

# 2020 OS Project 1 — Process Scheduling

B05902038 資工四 林詩苓

## 1. 設計

### 1. 整體架構

我採取單核心的方式，也就是將排程和所有子程序都綁到同一顆 CPU 上執行，來完成這份作業。無論是哪種排程方式，每一個 time unit 都會經歷這個流程：

- 排程檢查是否有新 ready 的子程序，若有即 fork 出來，並把其綁至和自己相同的 CPU，優先度調至低於自己（也就是該子程序不會立即執行）。
- 排程檢查有沒有想要執行（ready 了，但還沒完成）的子程序，若有即依據排程方式決定此 time unit 要執行的是哪一個子程序，並將其優先度調至高於自己。此刻 CPU 會 context switch 至該子程序，子程序跑完一個 time unit 後，將自己的優先度調至低於排程，CPU context switch 回排程。
- 排程檢查這個 time unit 有沒有執行過任何子程序，若無，則排程自己跑一個 time unit，然後進入下一個 time unit 周期，直到所有子程序都執行完為止。

### 子程序實作

```
1. main() {
2.     print(name, pid); //印至 stdout
3.     start = get time by system call;
4.     for(int i = 0; i < T; i++){
5.         run 1 unit time;
6.         set myself to low priority (except the last time);
7.         //要把自己切成 T 份，只需 context switch(T-1) 次
8.     }
9.     end = get time by system call;
10.    print to dmesg by system call(pid, start, end);
11.    return;
12. }
```

## 2. FIFO

### 演算法

First in, first out. 根據每個子程序的 ready 時間排序，先 ready 的先執行。

### 實作

```
1. SIGCHLD_handler(){
2.     wait();
3.     done++;
4. }
5.
6. int adjust_priority(){
7.     if(ready > done){ //有子程序等待執行
8.         set process[done] to high priority;
9.         return 1;
10.    }
11.    return 0;
12. }
13.
14. fifo(){
15.     sort(R); //依照 ready 時間排序
16.     for(t = 0; done < N; t++){
17.         while (ready < N && R of process[ready] <= t){
18.             //有新的子程序 ready
19.             fork process[ready];
20.             ready++;
21.         }
22.         child = adjust_priority();
23.         if(child == 0){ //此 time unit 沒有執行任何子程序
24.             run 1 unit time;
25.         }
26.     }
27.     return;
28. }
```

## 3. RR

### 演算法

Round-robin. 各子程序的優先度與 FIFO 相似，但與其不同的是，RR 若有子程序連續占用一定量的時間，會將他踢出，讓他重新回到 ready 中排隊。

### 實作

使用 queue 來實作排隊。

```
1. SIGCHLD_handler(){
2.     wait();
3.     done++;
```

```

4.     count = 0;
5.     run = 0;
6. }
7.
8. int adjust_priority(){
9.     if(ready > done){ //有子程序等待執行
10.         run = 1;
11.         if(count == 500){ //把子程序踢出
12.             push(current);
13.             current = pop();
14.             count = 0;
15.         }
16.         count++;
17.         set process[current] to high priority;
18.         return 1;
19.     }
20.     return 0;
21. }
22.
23. RR(){
24.     sort(R); //依照 ready 時間排序
25.     for(t = 0; done < N; t++){
26.         while (ready < N && R of process[ready] <= t){
27.             //有新的子程序 ready
28.             push(process[ready]);
29.             fork process[ready];
30.             ready++;
31.         }
32.         if(run == 0 && queue size != 0){
33.             //沒有子程序在執行，而且 queue 不為空
34.             current = pop();
35.         }
36.         child = adjust_priority();
37.         if(child == 0){ //此 time unit 沒有執行任何子程序
38.             run 1 unit time;
39.         }
40.     }
41.     return;
42. }

```

## 4. SJF

### 演算法

Shortest job first. 在所有 ready 的子程序中，選擇執行時間最小的來執行。

### 實作

使用 min heap 來取得執行時間最小的子程序。

```

1. SIGCHLD_handler(){
2.     wait();
3.     done++;

```

```

4.     run = 0;
5. }
6.
7. int adjust_priority(){
8.     if(ready > done){ //有子程序等待執行
9.         run = 1;
10.         set process[current] to high priority;
11.         return 1;
12.     }
13.     return 0;
14. }
15.
16. sjf(){
17.     sort(R); //依照 ready 時間排序
18.     for(t = 0; done < N; t++){
19.         while (ready < N && R of process[ready] <= t){
20.             //有新的子程序 ready
21.             push(process[ready]); //以 T 作為 heap node 的 value
22.             fork process[ready];
23.             ready++;
24.         }
25.         if(run == 0 && heap head != NULL){
26.             //沒子程序在執行，而且 heap 不為空
27.             current = pop();
28.         }
29.         child = adjust_priority();
30.         if(child == 0){ //此 time unit 沒有執行任何子程序
31.             run 1 unit time;
32.         }
33.     }
34. }

```

## 5. PSJF

### 演算法

Preemptive shortest job first. 在所有 ready 的子程序中，選擇剩餘執行時間最小的來執行。所以若新 ready 的子程序的執行時間，比現在正在執行的子程序的剩餘執行時間小，那麼現在正在執行的子程序就會被插斷，換成新 ready 的子程序來執行。

### 實作

使用 min heap 來取得剩餘執行時間最小的子程序。

```

1. SIGCHLD_handler(){
2.     wait();
3.     done++;
4. }
5.
6. int adjust_priority(){

```

```

7.     if(ready > done){ //有子程序等待執行
8.         current = pop();
9.         T of process[current]--;
10.        if(T of process[current] > 0){ //此程序還沒結束
11.            push(process[current]); //以 T 作為 heap node 的 value
12.        }
13.        set process[current] to high priority;
14.        return 1;
15.    }
16.    return 0;
17. }
18.
19. psjf(){
20.     sort(R); //依照 ready 時間排序
21.     for(t = 0; done < N; t++){
22.         while (ready < N && R of process[ready] <= t){
23.             //有新的子程序 ready
24.             push(process[ready]); //以 T 作為 heap node 的 value
25.             fork process[ready];
26.             ready++;
27.         }
28.         child = adjust_priority();
29.         if(child == 0){ //此 time unit 沒有執行任何子程序
30.             run 1 unit time;
31.         }
32.     }
33. }

```

## 2. 核心版本

Ubuntu, with Linux 4.14.25

## 3. 實際結果與理論結果

### 1. TIME\_MEASUREMENT

測資：

FIFO

10

P0 0 500

P1 1000 500

P2 2000 500

P3 3000 500

P4 4000 500

P5 5000 500

P6 6000 500

P7 7000 500

P8 8000 500

P9 9000 500

stdout :

P0 3048

P1 3049

P2 3050

P3 3058

P4 3059

P5 3064

P6 3068

P7 3069

P8 3077

P9 3078

dmesg :

```
[ 453.295368] [Project1] 3048 1587907234.227194833 1587907235.234569536
[ 455.247530] [Project1] 3049 1587907236.249126477 1587907237.187707108
[ 457.265989] [Project1] 3050 1587907238.213167919 1587907239.207175780
[ 459.250719] [Project1] 3058 1587907240.214212271 1587907241.192897496
[ 461.203293] [Project1] 3059 1587907242.132884034 1587907243.146448250
[ 463.141811] [Project1] 3064 1587907244.088628331 1587907245.085935612
[ 465.203603] [Project1] 3068 1587907246.126341188 1587907247.148757918
[ 467.186674] [Project1] 3069 1587907248.106843140 1587907249.132820423
[ 469.211229] [Project1] 3077 1587907250.150060718 1587907251.158388315
[ 471.255567] [Project1] 3078 1587907252.199944869 1587907253.203747976
```

理論值（為了方便與 dmesg 做比較，採結束順。後面的 case 不再重複說明）：

```
P0: 0- 500
P1: 1000-1500
P2: 2000-2500
P3: 3000-3500
P4: 4000-4500
P5: 5000-5500
P6: 6000-6500
P7: 7000-7500
P8: 8000-8500
P9: 9000-9500
```

由此可知，500 個 time unit 為 0.9990044634 秒。

## 2. FIFO

### FIFO\_1

測資：

```
FIFO
```

```
5
```

```
P1 0 500
```

```
P2 0 500
```

```
P3 0 500
```

```
P4 0 500
```

```
P5 0 500
```

stdout :

```
P1 2865
```

```
P2 2866
```

```
P3 2867
```

```
P4 2868
```

```
P5 2869
```

dmesg :

```
[ 362.141417] [Project1] 2865 1587907143.052953675 1587907144.035041835
```

```
[ 363.135923] [Project1] 2866 1587907144.035500570 1587907145.030044903
```

```
[ 364.011951] [Project1] 2867 1587907145.030522013 1587907145.906510288
```

```
[ 365.011989] [Project1] 2868 1587907145.906982387 1587907146.907047742
```

```
[ 366.013930] [Project1] 2869 1587907146.907498264 1587907147.909485181
```

理論值：

右側為實際值與理論值的差異，中間欄為起始時間，右欄為結束時間，所有的數字都是與最早開始的子程序的開始時間相比，也就是累積的誤差。負數代表實際值較理論值小，程序實際上跑得比理論快。後面的 case 不再重複說明。



P1: 0- 500	±0	-0.0169163034
P2: 500-1000	-0.0164575684	-0.0209176988
P3: 1000-1500	-0.0204405888	-0.1434567772
P4: 1500-2000	-0.1429846782	-0.1419237866
P5: 2000-2500	-0.1414732646	-0.138490811

誤差為理論總秒數之 2.78%。

## FIFO\_2

測資：

```
FIFO
4
P1 0 80000
P2 100 5000
P3 200 1000
P4 300 1000
```

stdout：

```
P1 7139
P2 7140
P3 7141
P4 7142
```

dmesg：

```
[ 3147.289662] [Project1] 7139 1587918504.882573791 1587918666.792786902
[ 3157.038618] [Project1] 7140 1587918666.793288732 1587918676.541742864
[ 3158.896307] [Project1] 7141 1587918676.542209873 1587918678.399432131
[ 3160.913539] [Project1] 7142 1587918678.399872533 1587918680.416663282
```

理論值：

P1:	0-80000	±0	+2.069498967
P2:	80000-85000	+2.070000797	+1.828410295
P3:	85000-86000	+1.828877304	+1.6280906352
P4:	86000-87000	+1.6885309372	+1.7073128594

誤差為理論總秒數之 0.98%。

### FIFO\_3

測資：

FIFO
7
P1 0 8000
P2 200 5000
P3 300 3000
P4 400 1000
P5 500 1000
P6 500 1000
P7 600 4000

stdout：

P1 3430
P2 3434
P3 3435
P4 3436
P5 3437
P6 3438
P7 3439

dmesg：

```
[ 721.326182] [Project1] 3430 1587907487.283363222 1587907503.399399075
[ 731.021694] [Project1] 3434 1587907503.399873115 1587907513.099757942
[ 736.873659] [Project1] 3435 1587907513.100232713 1587907518.954649302
[ 738.843609] [Project1] 3436 1587907518.955120521 1587907520.925584727
[ 740.837503] [Project1] 3437 1587907520.926032352 1587907522.920475254
[ 742.790227] [Project1] 3438 1587907522.920921754 1587907524.874175656
[ 750.950857] [Project1] 3439 1587907524.874625404 1587907533.038886658
```

理論值：

P1: 0- 8000	±0	+0.1319644386
P2: 8000-13000	+0.1324384786	-0.1577213284
P3: 13000-16000	-0.1572465574	-0.2968567488
P4: 16000-17000	-0.2963855298	-0.329302506
P5: 17000-18000	-0.3234826356	-0.3270486504
P6: 18000-19000	-0.3266021504	-0.3713571752
P7: 19000-23000	-0.3709074272	-0.1986818804

誤差為理論總秒數之 0.43%。

## FIFO\_4

測資：

```
FIFO
4
P1 0 2000
P2 500 500
P3 500 200
P4 1500 500
```

stdout：

P1 3527

P2 3528

P3 3529

P4 3537

dmesg :

[ 768.985329] [Project1] 3527 1587907547.015104139 1587907551.082376101

[ 769.993475] [Project1] 3528 1587907551.082841156 1587907552.091025179

[ 770.392845] [Project1] 3529 1587907552.091465082 1587907552.490594935

[ 771.310437] [Project1] 3537 1587907552.491040403 1587907553.408645638

理論值：

P1: 0-2000	±0	+0.0712541084
------------	----	---------------

P2: 2000-2500	+0.0717191634	+080898723
---------------	---------------	------------

P3: 2500-2700	+0.081338626	+0.08086669364
---------------	--------------	----------------

P4: 2700-3200	+0.08131216164	-0.00008706676
---------------	----------------	----------------

誤差小於理論總秒數之 0.01%。

## FIFO\_5

測資：

FIFO

7

P1 0 8000

P2 200 5000

P3 200 3000

P4 400 1000

P5 400 1000

P6 600 1000

P7 600 4000

stdout：

P1 5066

P2 5069

P3 5070

P4 5071

P5 5072

P6 5073

P7 5074

dmesg :

[ 1605.989421] [Project1] 5066 1587917109.431324708 1587917125.319014504

[ 1616.100158] [Project1] 5069 1587917125.319510270 1587917135.434806821

[ 1622.223518] [Project1] 5070 1587917135.435266070 1587917141.561217766

[ 1624.235522] [Project1] 5071 1587917141.561707942 1587917143.574237987

[ 1626.051643] [Project1] 5072 1587917143.574679071 1587917145.391267315

[ 1627.914325] [Project1] 5073 1587917145.391711142 1587917147.254880406

[ 1635.939335] [Project1] 5074 1587917147.255344350 1587917155.283902774

理論値：

P1: 0- 8000	±0	-0.0963816184
-------------	----	---------------

P2: 8000-13000	-0.0958858524	+0.0293660646
----------------	---------------	---------------

P3: 13000-16000	+0.0298253136	+0.1617502292
-----------------	---------------	---------------

P4: 16000-17000	+0.1622404052	+0.1767615234
-----------------	---------------	---------------

P5: 17000-18000	+0.1772026074	-0.0042180754
-----------------	---------------	---------------

P6: 18000-19000	-0.0037742484	-0.1386139112
-----------------	---------------	---------------

P7: 19000-23000	-0.1381499672	-0.1016272504
-----------------	---------------	---------------

誤差為理論總秒數之 0.22%。

### 3. RR

RR\_1

測資：

RR

5

P1 0 500

P2 0 500

P3 0 500

P4 0 500

P5 0 500

stdout :

P1 3666

P2 3667

P3 3668

P4 3669

P5 3670

dmesg :

[ 849.218142] [Project1] 3666 1587907630.357859043 1587907631.355305273

[ 850.199814] [Project1] 3667 1587907631.355749852 1587907632.337467162

[ 851.100936] [Project1] 3668 1587907632.337945542 1587907633.239039526

[ 851.962375] [Project1] 3669 1587907633.239518387 1587907634.100909171

[ 852.839120] [Project1] 3670 1587907634.101370716 1587907634.978092644

理論値：

P1: 0- 500	±0	-0.0015582334
P2: 500-1000	-0.0011136544	-0.0184008078
P3: 1000-1500	-0.0179224278	-0.1158329072
P4: 1500-2000	-0.1153540462	-0.2529677256
P5: 2000-2500	-0.2525061806	-0.374788716

誤差為理論總秒數之 7.5%。

RR\_2

測資：

```
RR
2
P1 600 4000
P2 800 5000
```

stdout：

```
P1 3712
P2 3713
```

dmesg：

```
[ 886.736937] [Project1] 3712 1587907653.820365098 1587907668.892859423
[ 889.768535] [Project1] 3713 1587907654.801789674 1587907671.925972403
```

理論值：

P1: 600-8100	$\pm 0$	+0.087427374
P2: 1100-9600	-0.0175798874	+0.1235269638

誤差為理論總秒數之 0.69%。

### RR\_3

測資：

```
RR
6
P1 1200 5000
P2 2400 4000
P3 3600 3000
P4 4800 7000
P5 5200 6000
P6 5800 5000
```

stdout：

P1 3764

P2 3772

P3 3773

P4 3778

P5 3782

P6 3783

dmesg :

[ 938.669460] [Project1] 3773 1587907693.187835250 1587907720.851348164

[ 941.583702] [Project1] 3764 1587907687.279788021 1587907723.767047298

[ 942.608438] [Project1] 3772 1587907690.179254642 1587907724.792295138

[ 958.477765] [Project1] 3783 1587907701.225288271 1587907740.669557866

[ 962.420927] [Project1] 3782 1587907698.175889595 1587907744.614691192

[ 964.272759] [Project1] 3778 1587907697.220710308 1587907746.467449447

理論値：

P3: 4200-18200	-0.0859795514	-0.3945916126
----------------	---------------	---------------

P1: 1200-19700	±0	-0.4759058688
----------------	----	---------------

P2: 2700-20200	-0.0975467692	-0.4496624922
----------------	---------------	---------------

P6: 8200-28200	-0.0405622376	-0.5564711786
----------------	---------------	---------------

P5: 6700-30200	-0.0929475234	-0.6073557062
----------------	---------------	---------------

P4: 6200-31200	-0.049122347	-0.752606378
----------------	--------------	--------------

誤差為理論總秒數之 1.26%。

**RR\_4**

測資：



RR

7

P1 0 8000

P2 200 5000

P3 300 3000

P4 400 1000

P5 500 1000

P6 500 1000

P7 600 4000

stdout :

P1 3879

P2 3880

P3 3881

P4 3882

P5 3883

P6 3884

P7 3889

dmesg :

[ 988.764409] [Project1] 3882 1587907763.225325887 1587907770.971344969

[ 989.756633] [Project1] 3883 1587907764.215620832 1587907771.964065411

[ 990.771023] [Project1] 3884 1587907765.166091848 1587907772.978961875

[ 1006.413407] [Project1] 3881 1587907762.232479884 1587907788.629166845

[ 1014.223482] [Project1] 3889 1587907767.144559833 1587907796.443147801

[ 1016.979210] [Project1] 3880 1587907761.415146500 1587907799.200253003

[ 1022.705436] [Project1] 3879 1587907760.419020915 1587907804.929342177

理論値 :

P4: 1500- 5500	-0.1907084182	-0.4367250434
P5: 2000- 6000	-0.1994179366	-0.4430090648
P6: 2500- 6500	-0.247951384	-0.4271170642
P3: 1000-14500	-0.1845499578	-0.7609835086
P7: 3500-18500	-0.2674923258	-0.9390382598
P2: 500-20000	-0.0028788784	-1.178946448
P1: 0-23000	±0	-1.4438840544

誤差為理論總秒數之 3.14%。

## RR\_5

測資：

RR
7
P1 0 8000
P2 200 5000
P3 200 3000
P4 400 1000
P5 400 1000
P6 600 1000
P7 600 4000

stdout：

P1 3155
P2 3177
P3 3178
P4 3179
P5 3180
P6 3274
P7 3275

dmesg :

```
[ 237.658481] [Project1] 3179 1588000802.843070828 1588000810.490062563
[ 238.658994] [Project1] 3180 1588000803.803129626 1588000811.491075469
[ 240.579490] [Project1] 3274 1588000805.696785839 1588000813.412531075
[ 254.534706] [Project1] 3178 1588000801.826229700 1588000827.374725013
[ 261.876224] [Project1] 3275 1588000806.690973340 1588000834.719914144
[ 264.437332] [Project1] 3177 1588000801.003057410 1588000837.282302379
[ 269.953300] [Project1] 3155 1588000800.069875674 1588000842.801028288
```

理論值：

P4: 1500- 5500	-0.2238182362	-0.5688622084
P5: 2000- 6000	-0.2627639016	-0.5668537658
P6: 3000- 7000	-0.3671166154	-0.6434070866
P3: 1000-14500	-0.2416549008	-1.6662800996
P7: 3500-18500	-0.3719335778	-2.3131266758
P2: 500-20000	-0.0658227274	-2.747751831
P1: 0-23000	$\pm 0$	-3.2230527024

誤差為理論總秒數之 7.01%。

## 4. SJF

SJF\_1

測資：

```
SJF
4
P1 0 7000
P2 0 2000
P3 100 1000
P4 200 4000
```

stdout：

P2 9858

P3 9859

P4 9860

P1 9857

dmesg :

[ 4760.838046] [Project1] 9858 1587911541.013850203 1587911544.892433105

[ 4762.699980] [Project1] 9859 1587911544.892958511 1587911546.754367682

[ 4770.551864] [Project1] 9860 1587911546.754827620 1587911554.606252569

[ 4784.854437] [Project1] 9857 1587911554.606714950 1587911568.908824780

理論値：

P2: 0- 2000	±0	-0.1174349516
-------------	----	---------------

P3: 2000- 3000	-0.1169095456	-0.2535093014
----------------	---------------	---------------

P4: 3000- 7000	-0.2530493634	-0.3936601216
----------------	---------------	---------------

P1: 7000-14000	-0.3931977406	-0.0771503982
----------------	---------------	---------------

誤差為理論總秒數之 0.28%。

## SJF\_2

測資：

SJF

5

P1 100 100

P2 100 4000

P3 200 200

P4 200 4000

P5 200 7000

stdout：

P1 9946

P3 9948

P2 9947

P4 9949

P5 9950

dmesg :

[ 4816.438857] [Project1] 9946 1587911600.296450616 1587911600.493244839

[ 4816.829355] [Project1] 9948 1587911600.493812116 1587911600.883742485

[ 4824.834112] [Project1] 9947 1587911600.884188263 1587911608.888500020

[ 4832.779306] [Project1] 9949 1587911608.888980083 1587911616.833693569

[ 4846.919822] [Project1] 9950 1587911616.834136499 1587911630.974209472

理論值：

P1: 100- 200	±0	-0.00300666968
--------------	----	----------------

P3: 200- 400	-0.00243939268	-0.01211080904
--------------	----------------	----------------

P2: 400- 4400	-0.01166503104	+0.00061101876
---------------	----------------	----------------

P4: 4400- 8400	+0.00109108176	-0.04623113944
----------------	----------------	----------------

P5: 8400-15400	-0.04578820944	+0.10822227596
----------------	----------------	----------------

誤差為理論總秒數之 0.35%。

**SJF\_3**

測資：

SJF

8

P1 100 3000

P2 100 5000

P3 100 7000

P4 200 10

P5 200 10

P6 300 4000

P7 400 4000

P8 500 9000

stdout :

P1 10029

P4 10032

P5 10033

P6 10034

P7 10035

P2 10030

P3 10031

P8 10036

dmesg :

[ 4874.122918] [Project1] 10029 1587911652.205929707 1587911658.177306186  
 [ 4874.142351] [Project1] 10032 1587911658.177787280 1587911658.196738517  
 [ 4874.162814] [Project1] 10033 1587911658.197151765 1587911658.217201810  
 [ 4881.915855] [Project1] 10034 1587911658.217862533 1587911665.970242285  
 [ 4889.792364] [Project1] 10035 1587911665.970741100 1587911673.846751290  
 [ 4899.605792] [Project1] 10030 1587911673.847289600 1587911683.660179383  
 [ 4913.325354] [Project1] 10031 1587911683.660660725 1587911697.379742291  
 [ 4931.619063] [Project1] 10036 1587911697.380196390 1587911715.673450736

理論值：

P1: 100- 3100	±0	-0.0226503014
P4: 3100- 3110	-0.0221692074	-0.023198059668
P5: 3110- 3120	-0.022784811668	-0.022714855936
P6: 3120- 7120	-0.022054132936	-0.261710088136
P7: 7120-11120	-0.261211273136	-0.377236790336
P2: 11120-16120	-0.376698480336	-0.553853331336
P3: 16120-23120	-0.553371989336	-0.820352910936
P8: 23120-32120	-0.819898811936	-0.508724807136

誤差為理論總秒數之 0.8%。

## SJF\_4

測資：

SJF  
 5  
 P1 0 3000  
 P2 1000 1000  
 P3 2000 4000  
 P4 5000 2000  
 P5 7000 1000

stdout :

```
P1 10151
P2 10159
P3 10160
P5 10176
P4 10168
```

dmesg :

```
[ 4950.980791] [Project1] 10151 1587911729.904497974 1587911735.035178905
[ 4952.639406] [Project1] 10159 1587911735.035642233 1587911736.693793739
[ 4960.645601] [Project1] 10160 1587911736.694260387 1587911744.699988838
[ 4962.681288] [Project1] 10176 1587911744.700460335 1587911746.735675922
[ 4966.125705] [Project1] 10168 1587911746.736125752 1587911750.180091902
```

理論值：

P1: 0- 3000	±0	-0.8633458494
P2: 3000- 4000	-0.8628825214	-1.2027399422
P3: 4000- 8000	-1.2022732942	-1.1885805504
P5: 8000- 9000	-1.1881090534	-1.1509023932
P4: 9000-11000	-1.1504525632	-1.7025042668

誤差為理論總秒數之 7.75%。

## SJF\_5

測資：

```
SJF
4
P1 0 2000
P2 500 500
P3 1000 500
P4 1500 500
```



stdout :

```
P1 10227
P2 10228
P3 10229
P4 10230
```

dmesg :

```
[ 4993.746374] [Project1] 10227 1587911774.086452726 1587911777.800761387
[ 4994.750980] [Project1] 10228 1587911777.801247680 1587911778.805367350
[ 4995.754470] [Project1] 10229 1587911778.805804187 1587911779.808857675
[ 4996.715778] [Project1] 10230 1587911779.809337493 1587911780.770165013
```

理論值：

P1: 0-2000	$\pm 0$	-0.2817091926
P2: 2000-2500	-0.2812228996	-0.276107693
P3: 2500-3000	-0.275670856	-0.2716218314
P4: 3000-3500	-0.2711420134	-0.3093189568

誤差為理論總秒數之 4.42%。

## 5. PSJF

PSJF\_1

測資：

```
PSJF
4
P1 0 10000
P2 1000 7000
P3 2000 5000
P4 3000 3000
```

stdout :

P1 2441

P2 2446

P3 2447

P4 2460

dmesg :

[ 200.040199] [Project1] 2460 1588006270.548047049 1588006275.988330253

[ 208.064715] [Project1] 2447 1588006268.547272629 1588006284.016858831

[ 220.033849] [Project1] 2446 1588006266.522193682 1588006295.991976125

[ 236.846124] [Project1] 2441 1588006264.597289363 1588006312.812657842

理論值：

P4: 3000- 6000	-0.0432690944	-0.5970126708
----------------	---------------	---------------

P3: 2000-10000	-0.0460345876	-0.5605198
----------------	---------------	------------

P2: 1000-16000	-0.0731046078	-0.5734560668
----------------	---------------	---------------

P1: 0-25000	$\pm 0$	-1.734854691
-------------	---------	--------------

誤差為理論總秒數之 3.47%。

## PSJF\_2

測資：

PSJF

5

P1 0 3000

P2 1000 1000

P3 2000 4000

P4 5000 2000

P5 7000 1000

stdout：

P1 10382

P2 10390

P3 10391

P4 10399

P5 10409

dmesg :

[ 5094.176689] [Project1] 10390 1587911876.236817619 1587911878.231077155

[ 5098.160382] [Project1] 10382 1587911874.230568124 1587911882.214769412

[ 5103.458149] [Project1] 10399 1587911884.200221375 1587911887.512536384

[ 5105.482162] [Project1] 10409 1587911887.513082221 1587911889.536549377

[ 5111.326042] [Project1] 10391 1587911882.215251728 1587911895.380429530

理論値 :

P2: 1000- 2000	+0.0082405682	+0.0044911774
----------------	---------------	---------------

P1: 0- 4000	±0	-0.0078344192
-------------	----	---------------

P4: 5000- 7000	-0.020391383	-0.7040942276
----------------	--------------	---------------

P5: 7000- 8000	-0.7035483906	-0.6780901614
----------------	---------------	---------------

P3: 4000-11000	-0.0073521032	-0.8282367888
----------------	---------------	---------------

誤差為理論總秒數之 3.77%。

### PSJF\_3

測資 :

PSJF

4

P1 0 2000

P2 500 500

P3 1000 500

P4 1500 500

stdout :

P1 10444

P2 10445

P3 10453

P4 10454

dmesg :

[ 5128.234062] [Project1] 10445 1587911911.283904665 1587911912.288449839

[ 5129.226695] [Project1] 10453 1587911912.288974324 1587911913.281082599

[ 5130.247425] [Project1] 10454 1587911913.281589169 1587911914.301811966

[ 5133.266730] [Project1] 10444 1587911910.289929637 1587911917.321118311

理論値 :

P2: 500-1000	-0.0050294354	+0.0005112752
--------------	---------------	---------------

P3: 1000-1500	+0.0010357602	+0.0058604282
---------------	---------------	---------------

P4: 1500-2000	-0.0053538582	+0.0158644754
---------------	---------------	---------------

P1: 0-3500	$\pm 0$	+0.0381574302
------------	---------	---------------

誤差為理論總秒數之 0.55%。

**PSJF\_4**

測資 :

PSJF

4

P1 0 7000

P2 0 2000

P3 100 1000

P4 200 4000

stdout :

P2 10490

P3 10491

P4 10492

P1 10489

dmesg :

[ 5154.347952] [Project1] 10491 1587911936.755436416 1587911938.402339786

[ 5157.745380] [Project1] 10490 1587911936.572263611 1587911941.799767475

[ 5165.736071] [Project1] 10492 1587911941.800244300 1587911949.790458625

[ 5179.380618] [Project1] 10489 1587911949.790906826 1587911963.435006197

理論值：

P3: 100- 1100	-0.01662808768	-0.36773364448
---------------	----------------	----------------

P2: 0- 3000	±0	-0.7665229164
-------------	----	---------------

P4: 3000- 7000	-0.7660460914	-0.7678674736
----------------	---------------	---------------

P1: 7000-14000	-0.7674192726	-1.1093823892
----------------	---------------	---------------

誤差為理論總秒數之 3.97%。

PSJF\_5

測資：

PSJF

5

P1 100 100

P2 100 4000

P3 200 200

P4 200 4000

P5 200 7000

stdout：

P1 10557

P3 10559

P2 10558

P4 10560

P5 10561

dmesg：

[ 5193.373346] [Project1] 10557 1587911977.231859339 1587911977.427733757

[ 5193.767238] [Project1] 10559 1587911977.428303768 1587911977.821625656

[ 5201.738378] [Project1] 10558 1587911977.822077056 1587911985.792765822

[ 5209.757182] [Project1] 10560 1587911985.793260439 1587911993.811570079

[ 5223.858375] [Project1] 10561 1587911993.812026690 1587912007.912762744

理論值：

P1: 100-200	±0	-0.00392647468
-------------	----	----------------

P3: 200-400	-0.00335646368	-0.00963636104
-------------	----------------	----------------

P2: 400-4400	-0.00918496104	-0.03053190224
--------------	----------------	----------------

P4: 4400-8400	-0.03003728524	-0.00376335244
---------------	----------------	----------------

P5: 8400-15400	-0.00330674144	+0.11136682496
----------------	----------------	----------------

誤差為理論總秒數之 0.36%。

## 6. 討論與實作過程中遇到的困難

### 雙核與單核的比較與抉擇

雖然最後我是以單核的方式來實作，但其實我一開始是採取雙核，也就是排程綁在一顆 CPU 上，子程序綁在另一顆 CPU 上的做法。因為這種做法不會讓排程與子程序競爭同一顆 CPU，理應執行時間會比較短，而且比較不會出現某一個 time unit 的時間特別長的情形。然而途中，我卻遇到了難以解決的難題。

雙核的設計，我是採取排程和子程序都會在迴圈中跑 unit time，理論上兩者的時間就會同步。然而，排程因為要檢查新 ready 的子程序，和決定下一個執行的子程序，勢必會花費比子程序多的時間去處理，導致排程的 unit time 比子程序的長。

因為這個現象，測試 SJF\_2 時，理論上 P1 應在 200 時跑完，但 P1 的 200 卻是排程的 18X，讓排程以為 P3 還沒 ready，P1 就跑完了，便會先執行已 ready 的 P2，導致結果錯誤。

經過一番思考及測試後，發現單核的演算法完全不會遇到這個問題，所有排程方式的子程序執行的順序完全可預期且正確，而且測試資料的誤差也都不超過 10%，還算是個可接受的結果，所以最後就改以單核的方式實作了。

### 分析實驗數據

由以上結果可以看出，大部分測試資料的誤差都是負的，代表實際時間比理論時間短，推測原因為：當有一個子程序長時間連續執行，排程所須花費的工夫就會因不需要做改變而比較少，所以整體的執行時間就下降了。

當然，上述情形也不一定百分之百如此，例如 FIFO\_2 的 P1，明明連續執行了 80000 個 time unit，卻耗費了比理論值還長的時間。所幸多出來的誤差和 80000 個 time unit 的總時長相比不大，仍然是可接受的值。

### 可改善之處

我在實作 PSJF 的時候，想說既然是 preemptive，我就每個 time unit 都重新判斷剩餘執行時間最小的子程序就好了。然而，在撰寫這份 report，計算理論值時才想到，PSJF 要重新判斷剩餘執行時間最小的子程序的時間點，只有兩個：有新的子程序 ready 時、現在執行的子程序結束時，所以在每個 time unit 都做判斷完全是做白工。

然而，如果要重新去修改程式，所有的測試以及開頭的設計都要全部重測、重寫。想想現在的做法其實也不算錯，只是不夠聰明而已，測試的誤差也不算太大，所以便不修改程式，而把這件事寫在 report 中。