

# Operating Systems Project 1 Report

資工四 B05902116 陳昱鈞

## 1. Design

### FIFO (First-In-First-Out)

- 原理
  - 對於每個process，先來的先執行。
- 執行方式
  - 將所有process按照ready time由小到大排序。
  - 對於每個process，若其ready time為當前的時間，則將該process放入waiting\_list中。
  - 如果目前沒有在執行process，並且waiting\_list中有process，則從waiting\_list中取出process並執行，直到該process執行結束。
  - 執行中的process會被賦予一個較高的priority。

### PSJF (Preemptive Shortest Job First)

- 原理概念
  - 每次選擇剩餘執行時間最短的 process來執行，當新產生的 process 執行時間更短時，就可以插隊 (insert)。
- 執行方式
  - 將所有process照ready time由小到大排序。
  - 對於每個process，若其ready time為當前的時間，則將該process放入waiting\_list中。
  - 每個unit time開始前都會從waiting\_list 中選擇execution time最小的拿出來跑。
  - 每跑完一個unit time會將process放回waiting\_list。
  - 執行中的process會被賦予一個較高的priority。

### RR (Round Robin)

- 原理
  - 讓正在執行的process在執行一個time quantum後進入waiting\_list中，並從waiting\_list中選定下一個執行的process。
  - waiting\_list會像queue一樣，所以waiting\_list裡的每一個process都會輪流執行。
- 執行方式
  - 將所有process按照ready time由小到大排序。
  - 對於每個process，若其ready time為當前的時間，則將該process放入waiting\_list中。
  - 如果目前沒有在執行process，並且waiting\_list中有process，則從waiting\_list中取出process執行，並計算該process是否會在此time quantum內執行結束。如果沒有結束，在執行一個

time quantum後將remaining execution time減少一個time quantum，並將該process放入waiting\_list中。

- 執行中的process會被賦予一個較高的priority。

## SJF (Shortest Job First)

- 原理概念
  - 每次選擇當前waiting\_list執行時間最短的 process來執行。
- 執行方式
  - 將所有process照ready time由小到大排序。
  - 對於每個process，若其ready time為當前的時間，則將該process放入waiting\_list中，並把execution time最小者放到最前面。
  - 如果目前沒有在執行process，並且waiting\_list中有process，則取出execution time最小的process並執行，直到該process執行結束。
  - 執行中的process會被賦予一個較高的priority。

## 2. Kernel Version

4.14.25

## 3. Analyzing results

TIME\_MEASUREMENT.txt

```
1 FIFO
2 10
3 P0 0 500
4 P1 1000 500
5 P2 2000 500
6 P3 3000 500
7 P4 4000 500
8 P5 5000 500
9 P6 6000 500
10 P7 7000 500
11 P8 8000 500
12 P9 9000 500
```

```
1 P0 23143
2 P1 23144
3 P2 23145
4 P3 23146
5 P4 23147
6 P5 23148
7 P6 23149
```

8	P7	23150			
9	P8	23151			
10	P9	23152			
11	[13962.184430]	[Project1]	23143	1588133688.867052727	1588133689.564206334
12	[13963.524647]	[Project1]	23144	1588133690.232533001	1588133690.904380885
13	[13964.858019]	[Project1]	23145	1588133691.570861491	1588133692.237709322
14	[13966.192946]	[Project1]	23146	1588133692.902936130	1588133693.572593395
15	[13967.529314]	[Project1]	23147	1588133694.237092232	1588133694.908918623
16	[13968.865375]	[Project1]	23148	1588133695.576152369	1588133696.244936976
17	[13970.200599]	[Project1]	23149	1588133696.911581170	1588133697.580117241
18	[13971.535852]	[Project1]	23150	1588133698.244392433	1588133698.915328399
19	[13972.876678]	[Project1]	23151	1588133699.580926092	1588133700.256111122
20	[13974.216993]	[Project1]	23152	1588133700.924290016	1588133701.596382498

500 unit of time = 0.6732868 sec

## FIFO\_1.txt

1	FIFO				
2	5				
3	P1	0	500		
4	P2	0	500		
5	P3	0	500		
6	P4	0	500		
7	P5	0	500		

1	P1	23158			
2	P2	23159			
3	P3	23160			
4	P4	23161			
5	P5	23162			
6	[13991.842598]	[Project1]	23158	1588133718.550444859	1588133719.221421697
7	[13992.514938]	[Project1]	23159	1588133719.221811403	1588133719.893740578
8	[13993.184864]	[Project1]	23160	1588133719.894111209	1588133720.563644298
9	[13993.858031]	[Project1]	23161	1588133720.564013704	1588133721.236789596
10	[13994.530508]	[Project1]	23162	1588133721.237154849	1588133721.909245824

Process	start time	end time	theoretical start time	theoretical end time
P1	0	0.6709768	0	0.6732868
P2	0.671366544	1.3432957	0.6732868	1.3465736
P3	1.3436663	2.0131994	1.3465736	2.0198604
P4	2.01356884	2.6863447	2.0198604	2.6931472
P5	2.6867099	3.3588009	2.6931472	3.366434

## PSJF\_2.txt

```
1 | PSJF
2 | 5
3 | P1 0 3000
4 | P2 1000 1000
5 | P3 2000 4000
6 | P4 5000 2000
7 | P5 7000 1000
```

```
1 | P1 23168
2 | P2 23169
3 | P3 23170
4 | P4 23173
5 | P5 23174
6 | [14013.889130] [Project1] 23169 1588133739.921486600 1588133741.267244963
7 | [14016.583564] [Project1] 23168 1588133738.551207989 1588133743.961592027
8 | [14020.595021] [Project1] 23173 1588133745.295563003 1588133747.972920649
9 | [14021.932677] [Project1] 23174 1588133747.973396564 1588133749.310533685
10 | [14025.979654] [Project1] 23170 1588133743.962012521 1588133753.357380251
```

Process	start time	end time	theoretical start time	theoretical end time
P2	1.3702786	2.7160370	1.3465736	2.6931472
P1	0	5.4103839	0	5.3862944
P4	6.7443549	9.4217126	6.732868	9.4260152
P5	9.4221885	10.7593257	9.4260152	10.772588
P3	5.4108045	14.8061721	5.3862944	14.812309

## RR\_2.txt

```
1 | RR
2 | 2
3 | P1 600 4000
4 | P2 800 5000
```

```
1 | P1 3711
2 | P2 3722
3 | [ 4249.172825] [Project1] 3721 1588088187.286447891 1588088196.714433724
4 | [ 4251.887538] [Project1] 3722 1588088188.779615932 1588088199.534284810
```

Process	start time	end time	theoretical start time	theoretical end time
P1	0	9.427985783	0	9.367822170257568
P2	1.49316786	12.24783673	1.48123096	12.082452058792114

## SJF\_4.txt

```

1 SJF
2 5
3 P1 0 3000
4 P2 1000 1000
5 P3 2000 4000
6 P4 5000 2000
7 P5 7000 1000

```

```

1 P1 23202
2 P2 23203
3 P3 23204
4 P4 23205
5 P5 23206
6 [14101.736337] [Project1] 23202 1588133825.042401252 1588133829.111629007
7 [14103.078494] [Project1] 23203 1588133829.112005184 1588133830.453742295
8 [14108.442307] [Project1] 23204 1588133830.454157531 1588133835.817383267
9 [14109.782936] [Project1] 23206 1588133835.817804563 1588133837.157969457
10 [14112.467035] [Project1] 23205 1588133837.158309444 1588133839.841981624

```

Process	start time	end time	theoretical start time	theoretical end time
P1	0	4.0692276	0	4.0397208
P2	4.0696039	5.4113409	4.0397208	5.3862944
P3	5.4117562	10.7749819	5.3862944	10.7725888
P5	10.7754032	12.1155681	10.7725888	12.1191624
P4	12.1159081	14.7995803	12.1191624	14.8123096

## Conclusion

在實際執行時，使用兩種priority給予不同狀態的程序，搭配雙核心，可以讓程序間的銜接最好，同時也較容易進行排程管理。priority的分級如下：

1. 執行中的process：最高的priority

2. 其他的process：最低的priority
3. scheduler跑在另一個核心上，priority介於上述兩者之間

理論與實際結果的差異主要來自於以下三點：

1. scheduler 除了排程外，還有其他執行所花費的時間，例如動態調整各process的priority、執行新的process所花費的時間等。
2. 因為電腦上不只有這個program，所以有可能program因為context switch而被中斷，然而繼續計時，因此造成program執行時間上的增加。
3. scheduler跑完一個unit time跟process跑完一個unit time花費的時間不同，進而造成排成的誤差。