OS Project 1 report B07902116 資工二 陳富春

核心版本：Ubuntu 16.04 with Linux4.14.25

設計：

1. FIFO

實作方式：先依照 process 的 ready time 由小到大排序，若 ready time 為當前時間，則將該 process 放進 queue 中，然後依照 queue 中的順序依序執行，直到該 process 結束，再從 queue 中找下一個 process 執行

2. RR

實作方式：先依照 process 的 ready time 由小到大排序，若 ready time 為當前時間，則將該 process 放進 queue 中，然後依照 queue 中的順序依序執行，直到執行了一個週期後，若仍未執行完，則再將其 push 進 queue 中，接著從 queue 中找下一個 process 執行

3. SJF

實作方式：先依照 process 的 ready time 由小到大排序，若 ready time 相同則依照 execution time 排序，若 ready time 為當前時間，則將該 process 以 execution time 為 key value 放進一個 minheap 當中，然後每次都選擇在 minheap 的 root 的 process 執行，直到該 process 結束，再從 minheap 中找下一個 process 執行

4. PSJF

實作方式：先依照 process 的 ready time 由小到大排序，若 ready time 相同則依照 execution time 排序，若 ready time 為當前時間，則將該 process 以 execution time 為 key value 放進一個 minheap 當中，當有新的 process 產生，先將目前執行的 process execution time 減去已執行的時間，然後將新的 process 加進 minheap 中，再選擇位在 root 的 process 執行，直到有新的 process 產生或是執行完畢

結果比較：

1. FIFO

1.
P1 0 500
P2 0 500
P3 0 500
P4 0 500
P5 0 500
理論執行順序：1 > 2 > 3 > 4 > 5
理論完成順序：1 > 2 > 3 > 4 > 5
實際完成順序：1 > 2 > 3 > 4 > 5

2.
P1 0 80000
P2 100 5000
P3 200 1000
P4 300 1000
理論執行順序：1 > 2 > 3 > 4
理論完成順序：1 > 2 > 3 > 4
實際完成順序：1 > 2 > 3 > 4

3.
P1 0 8000
P2 200 5000
P3 300 3000
P4 400 1000
P5 500 1000
P6 500 1000
P7 600 4000
理論執行順序：1 > 2 > 3 > 4 > 5 > 6 > 7
理論完成順序：1 > 2 > 3 > 4 > 5 > 6 > 7
實際完成順序：1 > 2 > 3 > 4 > 5 > 6 > 7

4.
P1 0 2000
P2 500 500
P3 500 200
P4 1500 500
理論執行順序：1 > 2 > 3 > 4
理論完成順序：1 > 2 > 3 > 4
實際完成順序：1 > 2 > 3 > 4

5.
P1 0 8000
P2 200 5000
P3 200 3000
P4 400 1000
P5 400 1000
P6 600 1000
P7 600 4000
理論執行順序：1 > 2 > 3 > 4 > 5 > 6 > 7
理論完成順序：1 > 2 > 3 > 4 > 5 > 6 > 7
實際完成順序：1 > 2 > 3 > 4 > 5 > 6 > 7

2. RR

1.
P1 0 500
P2 0 500
P3 0 500
P4 0 500
P5 0 500
理論執行順序：1 > 2 > 3 > 4 > 5
理論完成順序：1 > 2 > 3 > 4 > 5
實際完成順序：1 > 2 > 3 > 4 > 5

2.
P1 600 4000
P2 800 5000
理論執行順序：(1 > 2) for 8 times, > 2 > 2
理論完成順序：1 > 2
實際完成順序：1 > 2

3.
P1 1200 5000
P2 2400 4000
P3 3600 3000
P4 4800 7000
P5 5200 6000
P6 5800 5000
理論執行順序：1 > 1 > 1 > 2 > (1 > 2 > 3) for 2 times, (4 > 1 > 5 > 6 > 2 > 3) for 4 times, 4 > 1 > 5 > 6 > 2, (4 > 5 > 6) for 5 times, (4 > 5) for 2 times, > 4 > 4
理論完成順序：3 > 1 > 2 > 6 > 5 > 4
實際完成順序：3 > 1 > 2 > 6 > 5 > 4

4.
P1 0 8000
P2 200 5000
P3 300 3000
P4 400 1000
P5 500 1000
P6 500 1000
P7 600 4000
理論執行順序：1 > (2 > 3 > 4 > 1 > 5 > 6 > 7) for 2 times, (2 > 3 > 1 > 7) for 4 times, (2 > 1 > 7) for 2 times, (2 > 1) for 2 times, (1) for 5 times
理論完成順序：4 > 5 > 6 > 3 > 7 > 2 > 1
實際完成順序：4 > 5 > 6 > 3 > 7 > 2 > 1

5.
P1 0 8000
P2 200 5000
P3 200 3000
P4 400 1000
P5 400 1000
P6 600 1000
P7 600 4000
理論執行順序：1 > (2 > 3 > 4 > 5 > 1 > 6 > 7) for 2 times, (2 > 3 > 1 > 7) for 4 times, (2 > 1 > 7) for 2 times, (2 > 1) for 2 times, (1) for 5 times
理論完成順序：4 > 5 > 6 > 3 > 7 > 2 > 1
實際完成順序：4 > 5 > 6 > 3 > 7 > 2 > 1

3. SJF

1.
P1 0 7000
P2 0 2000
P3 100 1000
P4 200 4000
理論執行順序：2 > 3 > 4 > 1
理論完成順序：2 > 3 > 4 > 1
實際完成順序：2 > 3 > 4 > 1

2.
P1 100 100
P2 100 4000
P3 200 200
P4 200 4000
P5 200 7000
理論執行順序：1 > 3 > 2 > 4 > 5
理論完成順序：1 > 3 > 2 > 4 > 5
實際完成順序：1 > 3 > 2 > 4 > 5

3.
P1 100 3000
P2 100 5000
P3 100 7000
P4 200 10
P5 200 10
P6 300 4000
P7 400 4000
P8 500 9000
理論執行順序：1 > 4 > 5 > 6 > 7 > 2 > 3 > 8
理論完成順序：1 > 4 > 5 > 6 > 7 > 2 > 3 > 8
實際完成順序：1 > 4 > 5 > 6 > 7 > 2 > 3 > 8

4.
P1 0 3000
P2 1000 1000
P3 2000 4000
P4 5000 2000
P5 7000 1000
理論執行順序：1 > 2 > 3 > 5 > 4
理論完成順序：1 > 2 > 3 > 5 > 4
實際完成順序：1 > 2 > 3 > 5 > 4

5.
P1 0 2000
P2 500 500
P3 1000 500
P4 1500 500
理論執行順序：1 > 2 > 3 > 4
理論完成順序：1 > 2 > 3 > 4
實際完成順序：1 > 2 > 3 > 4

4. PSJF

1.
P1 0 10000
P2 1000 7000
P3 2000 5000
P4 3000 3000
理論執行順序：1 > 2 > 3 > 4 > 3 > 2 > 1
理論完成順序：4 > 3 > 2 > 1
實際完成順序：4 > 3 > 2 > 1

2.
P1 0 3000
P2 1000 1000
P3 2000 4000
P4 5000 2000
P5 7000 1000
理論執行順序：1 > 2 > 1 > 3 > 4 > 5 > 3
理論完成順序：2 > 1 > 4 > 5 > 3
實際完成順序：2 > 1 > 4 > 5 > 3

3.
P1 0 2000
P2 500 500
P3 1000 500
P4 1500 500
理論執行順序：1 > 2 > 3 > 4 > 1
理論完成順序：2 > 3 > 4 > 1
實際完成順序：2 > 3 > 4 > 1

4.
P1 0 7000
P2 0 2000
P3 100 1000
P4 200 4000
理論執行順序：2 > 3 > 2 > 4 > 1
理論完成順序：3 > 2 > 4 > 1
實際完成順序：3 > 2 > 4 > 1

5.
P1 100 100
P2 100 4000
P3 200 200
P4 200 4000
P5 200 7000
理論執行順序：1 > 3 > 2 > 4 > 5
理論完成順序：1 > 3 > 2 > 4 > 5
實際完成順序：1 > 3 > 2 > 4 > 5

結論

在實作過程中，將排程程式(scheduler)指定在一顆 CPU，並將其他 child process 指定在另一顆 CPU，這樣便能在保持 process 正常執行的情況下，scheduler 能夠同時 fork 出新的 process、或是控制 process 的優先度和執行順序，藉此完成 process 的排程。

實際上執行後，雖然各 child process 的結束順序正確，但在執行時間上可能會和理論預期的有些差異，我認為造成的可能原因如下：

1. 由於 scheduler 還要額外處理像是 create process, signal handler, operations of queue/heap 等等工作，因此 scheduler 的時間可能會和 child process 的時間不一致，導致產生多餘的時間。
2. 由於 CPU 在更換正在跑的 child process 時需要進行 context switch，便會導致產生多餘的時間。
3. 當一個 child process 結束並將 SIGCHLD 送回給 scheduler，到 scheduler 處理完並選出下一個應該執行的 process 的這段時間，由於負責 child process 的 CPU 會繼續選擇其他的 child process 執行，可能因此導致原本在比較後面才會開始執行的 process 提早執行，導致其開始執行的時間比預期的早。
4. 負責 scheduler 和負責 child process 的兩顆 CPU 的執行速度有可能不相同，導致雙方的時間不一致。