OS Project 1 Report

資工二 謝心默 **B06505017**

Design

General

- main process 和 child processes 分別使用兩顆 CPU 來跑以防止互搶資源

- main process 維護一個 local clock，並在相對應的時間執行、block、結束 child processes

- 在 child processes 的那顆 CPU 上，只有當下正在執行的 process 會有較高的 priority

- 利用 share memory 來達到讓 child processes 照著 schedule 的時間 terminate

FIFO

First-Come, First-Served Scheduling

原理：

對於每個 process，先 ready 的先執行

實作方式：

- 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，則將其標記為 ready 的狀態，當其 ready time 為當前 ready 狀態最小的 process，執行直到結束

RR

原理

利用計時器固定週期，讓正在執行的 process 在執行一個週期後進入 queue 中，並從 queue 中選定下一個執行的 process。

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 queue 中

- 如果目前沒有在執行 process，並且 queue 中有 process，則從 queue 中取出 process 執行，若在執行一個 Time Quantum 後沒有結束，則將其 execution time 減少一個 Time Quantum，並將該 process 放回 queue 中

SJF

原理

選擇執行時間最短的 process 來執行，此處應用的 SJF 是 non-preemptive 的，不會中斷仍在執行的 process

執行方式

- 將所有 process 照 ready time 由小到大排序，若 ready time 相同，再按照 execution time 由小到大排序

- 對於每個 process，若其 ready time 為當前的時間，則將其標記為 ready 的狀態，當其為當前 ready 狀態最小 execution time 的 process，執行直到結束

PSJF

原理

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（preemption）

執行方式

- 將所有 process 照 ready time 由小到大排序，若 ready time 相同，再按照 execution time 由小到大排序

- 對於每個 process，若其 ready time 為當前的時間，則將其標記為 ready 的狀態，當其為當前 ready 狀態的 process 中最小 execution time 的 process，則將原本在執行的 process 替換，並讓原本執行的 process 回到 ready 的狀態中。

- 與 SJF 相似，唯一不同點在於 main process 在 fork 新 process 後，如果新的 process 執行時間較短，則之後轉移 cpu 時，不是轉移回去，而是轉給新 process

kernel version

Environment: linux ubuntu 16.04 LTS

kernel: linux-4.14.25

Comparison

使用程式(calc.py)計算每支 process 用了幾個 unit time

1. TIME_MEASUREMENT：每支 process 的範圍在[482, 512]之間，誤差大約 3.6%

```
ZenB:OS_Project1 momo$ cat OS_PJ1_Test/TIME_MEASUREMENT.txt
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 500ZenB:OS_Project1 momo$ python calc.py TIME_MEASUREMENT
a time unit is 0.00197118935585 sec.
process P0  runs 482.034 in total.
process P1  runs 491.038 in total.
process P2  runs 504.944 in total.
process P3  runs 509.746 in total.
process P4  runs 499.817 in total.
process P5  runs 507.993 in total.
process P6  runs 511.742 in total.
process P7  runs 507.282 in total.
process P8  runs 488.052 in total.
process P9  runs 497.351 in total.
```

2. FIFO_1: 每支 process 的範圍在[505, 519]之間，誤差大約 3.8%

```
ZenB:OS_Project1 momo$ cat OS_PJ1_Test/FIFO_1.txt
FIFO
5
P1 0 500
P2 0 500
P3 0 500
P4 0 500
P5 0 500ZenB:OS_Projectpython calc.py FIFO_1
a time unit is 0.00197118935585 sec.
process P1  runs 517.584 in total.
process P2  runs 512.955 in total.
process P3  runs 518.311 in total.
process P4  runs 511.263 in total.
process P5  runs 505.483 in total.
```

3. PSJF_2: 因為有 preemption，因此 P1 的理論總執行時間為 4000，P2 為 1000，P3 為 7000，P4 為 2000，P5 為 1000，實際誤差最大約 5.6%

```
ZenB:OS_Project1 momo$ cat OS_PJ1_Test/PSJF_2.txt
PSJF
5
P1 0 3000
P2 1000 1000
P3 2000 4000
P4 5000 2000
P5 7000 1000ZenB:OS_Propython calc.py PSJF_2
a time unit is 0.00197118935585 sec.
process P2  runs 945.850 in total.
process P1  runs 3894.274 in total.
process P4  runs 2064.817 in total.
process P5  runs 958.261 in total.
process P3  runs 6901.003 in total.
```

4. RR_3: 因為會一直 context switch，因此總執行時間比原定的執行時間增加了很多。

理論值應為: P1:18500，P2:17500，P3:14000，P4:25000，P5:23500，P6:20000

實際誤差約為 0.9%

```
ZenB:OS_Project1 momo$ cat OS_PJ1_Test/RR_3.txt
RR
6
P1 1200 5000
P2 2400 4000
P3 3600 3000
P4 4800 7000
P5 5200 6000
P6 5800 5000
ZenB:OS_Project1 momo$ python calc.py RR_3
a time unit is 0.00197118935585 sec.
process P3  runs 13879.388 in total.
process P1  runs 18341.791 in total.
process P2  runs 17363.152 in total.
process P6  runs 19816.318 in total.
process P5  runs 23289.728 in total.
process P4  runs 24795.688 in total.
```

5. SJF_4: 誤差約 1.5%

```
ZenB:OS_Project1 momo$ cat OS_PJ1_Test/SJF_4.txt
SJF
5
P1 0 3000
P2 1000 1000
P3 2000 4000
P4 5000 2000
P5 7000 1000ZenB:OS_Project1 momo$ python calc.py SJF_4
a time unit is 0.00197118935585 sec.
process P1  runs 3042.509 in total.
process P2  runs 1029.115 in total.
process P3  runs 4062.169 in total.
process P5  runs 992.080 in total.
process P4  runs 2021.878 in total.
```

Discussion

由以上比較可看出排程的順序是正確的，執行時間誤差也相當小。造成誤差的原因可能為以下：

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 與資料結構的操作等。
2. 同樣執行時間的兩支 process，理論上會跑相同次數的迴圈數，但實際跑的時間也不一樣，可能是因為 CPU 本身造成差異。
3. 電腦上不只有我們的程序，我們的 process 可能因為 context switch 而被中斷，然而我們的計時不會跟著中斷，因此造成程序執行時間增加。