**# OS Project 1 Report**

**tags: OS project OS project**

**資工三 黃奕鈞 B06902033**

**設計**

**General**

main process及之後fork的child們使用兩個不同的prioirty，只有當下正在執行的process使用較高的priority(因此我只使用一顆CPU)。 使用share memory來存取clock以及每個task的資訊(進來時間、剩餘執行時間)等需要分享的資訊，這樣每支process都能看到現在會不會有人要進來，如果有就(以調整priority的方式)將cpu歸還給main process，然後將這個時間點進來的task全部fork出來，再看是要交給上一位執行的process還是有人要preempt轉移cpu。

**FIFO**

使用Queue來存取執行中的程式，如果做完就DeQueue然後回main process，main process如果看到Queue不是空的就作Queue.front的process。在EnQueue的時候，因為一開始按照1. 進來時間 2. input出現順序 的方式排序，故EnQueue順序正確，以此來達到FIFO。

**RR**

與FIFO相同用Queue來實作。唯一不同是每支Process在執行的時候會記錄自己做了幾個回合(unit of time)了，如果大於零且為500的倍數就放在Queue的尾巴(EnQueue(Queue.front), DeQueue())。如此就能達到RR且新進來的process會進到Queue的尾巴了。

**SJF**

與上述兩者不同，在完成child process之後，main process不是看Queue來找process，而是看目前已經進來的process中，剩餘時間最短的(O(N))。因此一開始排序方式為 1. 進來時間 2. 剩餘執行時間 3. input出現順序

**PSJF**

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。其他都與SJF相同。

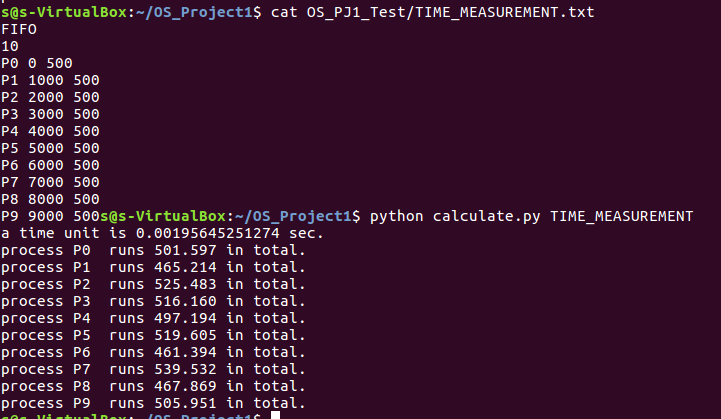
**核心版本**

環境:linux ubuntu 16.04 LTS Kernel:linux-4.14.25

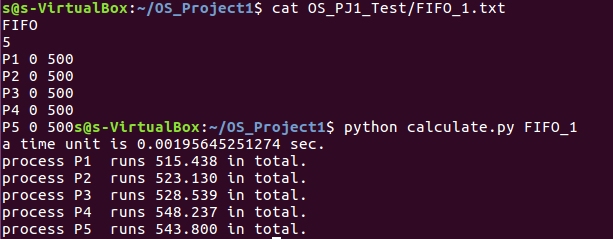
**比較**

使用程式(calculate.py)計算每個process用了幾個unit\_of\_time

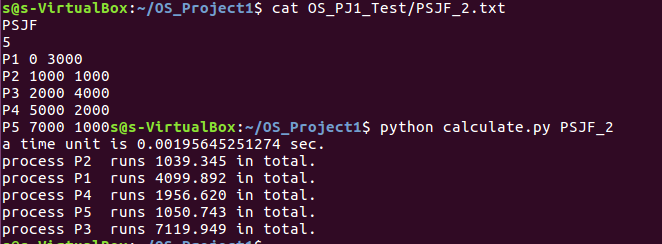
1. 首先看到TIME\_MEASUREMENT，每支process的範圍大概落在[500-40, 500+40]之間，也就是誤差在8%左右



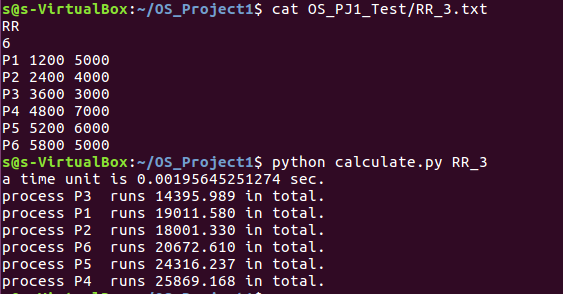
1. FIFO\_1的部分，誤差在+10%內



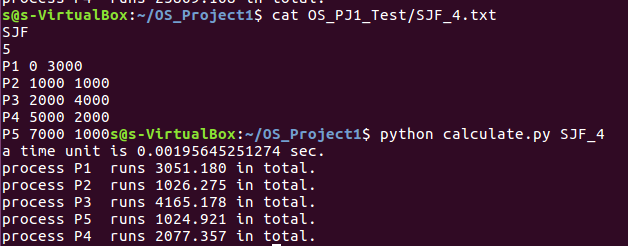
1. PSJF\_2的部分，因為有preempt，因此P1的總執行時間為4000，P2為1000，P3為7000，P4為2000，P5為1000，可看到誤差皆在5%內



1. RR\_3的部分，因為他們會一直切換，因此總執行時間比他們原定的執行時間增加了很多。理論值則應該各為:P1:18500，P2:17500，P3:14000，P4:25000，P5:23500，P6:20000，可以看到誤差都在5%內



1. SJF\_4的部分，因為沒有Preempt，因此誤差與要求的時間只在5%內



因此大概可以推論我的這份scheduler的誤差約會在5%內

**討論**

1. 在我的程式中，我是讓child來看是否有新的task出現，再喚醒scheduler去新增，但這部分應該是要交給scheduler來確認的，child不應該有辦法知道其他child的狀況。
2. 時間的定義上，在此使用unit of time，而我讓child也能更改目前的時間，但實際上會有專門的process來執行，且是用實際時間來看，會比較精準。
3. 不過因為這裡是跑空迴圈，但如果涉及到storage的操作，一個操作所需要使用的時間會大幅上升，因此若用幾個操作來看可能不太好，所以可能真的在做的時候可以用clock的值。
4. 我只有使用一顆CPU，因此main process在跑的時候可能會拖延到執行的時間，但因為操作數目不多，因此影響應該不大。main process會影響到只有在中途有新的task進來，或者是需要Preempt的時候，才會從child process退到main process。
5. 如果沒有涉及到其他process的情形，誤差的來源來自於非迴圈的操作，如if-else判斷是否該結束or換人等等

**備註**

* output資料夾底下的"xxx\_finish"紀錄執行該測資時，每個process開始(首次進入CPU)及結束的時間。