OS Project 1 - User-space Scheduler Design - Report

學號：B05902022　　姓名：張雅信　　系級：資工四

1. 設計

我實作了 2 核心的系統，其中一個核心只給 scheduler 使用，另一個給其他 process 使用。以下是 scheduler 的 pseudo code。其中「選擇 P」 以及「 P 此輪可以跑多久」會根據policy不同而改變：

```
SCHEDULE ( policy, N, processes ) {
    Sort processes by ready_time;
    Give scheduler core 0 and highest priority;
    Initialize queue waiting, running_P=NONE;
    current_time = 0;
    While ( not all processed finished ) {
        Fork new processes with [ready_time <= current_time] and put into waiting;
        /* Pick one process P to run according to policy */
        running_P = Choose_P( policy, waiting, running_P ) ;
        /* Decide how long it should run this turn */
        this_round = Round_Time ( policy, running_P ) ;
        Give process P core 1 and wake it up;
        (Busy) Wait for this_round;
        current_time += this_round;
        waitpid( P.pid ) if P is finished;
    }
}

Choose_P ( policy, waiting, running_P ) {
    if ( no process waiting ) return NONE;
    else if ( policy == FIFO or RR ) return ( next process in waiting ) ;
    else if ( policy == SJF or PSJF ) return ( waiting process with shortest exec_time ) ;
}
```

```
Round_Time ( policy, running_P ) {
        if ( running_P is NONE or  policy == PSJF ) return 1;
        else if ( policy == FIFO or SJF ) return ( exec_time of running_P ) ;
        else if ( policy == RR )
                return min ( remaining exec_time of running_P ,  TIME_QUANTUM ) ;
}
```

2. 核心版本

    linux 核心為 Linux 4.14.25。

    其他環境設定皆與作業一助教影片中相同 (VirtualBox , 系統為 Ubuntu 16.04.4)。

3. 比較實際結果與理論結果，並解釋造成差異的原因

    就執行順序來說，實際結果與理論結果相同。

    若假設 TIME_MEASUREMENT 的執行時間等於理論值，

    可建出以下表格，每格表示 **總執行時間** 的差距： 1 - (實際時間/理論時間)*100%

Policy \ Test Case	1	2	3	4	5
FIFO	0.56%	0.16%	-0.18%	0.45%	0.2%
RR	0.10%	0.19%	-10.35%	-6.90%	-7.18%
SJF	0.13%	0.42%	-0.15%	-0.85%	-0.49%
PSJF	-13.17%	-9.5%	-14.22%	-0.90%	0.45%

    在 FIFO 與 SJF 的策略中，與理論情況幾乎相符，造成誤差的原因可能為以下：

    (1) 以 TIME_MEASUREMENT 作為基礎的誤差 (+/-)

    (2) Scheduler 的 Busy waiting 與 process 的時間差，使 waitpid() 等地方delay

    不過都介於 1% 以內，個人認為尚可接受。

    但在 RR 與 PSJF 的策略中，出現許多實際結果比理論結果還要快的結果。

    照理來說，preemptive 的策略會造成 scheduler 的 overhead 更嚴重，

    花費時間應該會相對(理論值)增加較多。

    推測原因是 preemptive 策略中有許多 process 同時在等待，

    可能由於 sched_setscheduler( ) 僅給予很低的優先權，但並沒有完全停止。

    (也就是有偷跑的現象發生。)