OS Project 1 report

韓兆駿 r08922127

環境

a. Kernel 版本：linux 4.14.177
b. 虛擬機 OS：Ubuntu 18.04
c. 虛擬機硬體：4 CPU(1 個給主程式,1 個給子程式,其他執行 system process)，8G ram
d. 虛擬機程式：VM station pro 15

Design

a. 主程式與子程式使用不同的 CPU，減少對彼此的干擾。
b. 所有實作皆使用兩種底層排程規則來實現，分別為 SCHED_FIFO 和 SCHED_IDEL。
c. 主程式的排程固定為 SCHED_FIFO，讓其可以保證順利執行。
d. 在子程式中，SCHED_FIFO 表示正在執行的子程序；SCHED_IDEL 表示正在等待的子程序。

Scheduler

主程式在偵測子程式 ready time 到達時，會將其 fork 出來，並且 block 子程式。該子程式會等待，直到搶到 CPU 執行。(以下皆採用 for loop 來找到符合條件的子程式)

FIFO

a. 會依照子程式 ready time 將其 fork。
b. 按照先來先執行。
c. 直到所有子程式執行結束，主程式結束。

RR

a. 會依照子程式 ready time 將其 fork。
b. 正在使用 CPU 的子程式，執行 500 個 unit 後會使其 Block，並將 CPU 給下一個子程式。
c. 直到所有子程式執行結束，主程式結束。

SJF/PSJF

a. 會依照子程式 ready time 將其 fork。
b. 目前使用 CPU 的子程式結束後，SJF 會選擇下一個 execution time 最短的給予 CPU 讓其執行。(PSJF 在偵測到有 execution time 更小的子程式時，會 block 正在使用 CPU 的子程式，讓 execution 更短的子程式執行)
c. 直到所有子程式執行結束，主程式結束。

System call

a. 只使用一個 system call，其設計如下

```
asmlinkage int sys_my_new(int isStart, int pid, unsigned long __user *start_sec, unsigned long __user *start_nsec)
```

b. 這個 system call 會呼叫兩次。
c. 第一次會計算開始時間並且回傳到 start_sec 與 start_nsec。
d. 第二次會計算結束時間並且將 PID 還有開始和結束時間用 printk 打印到 dmesg 中。

Discussion

因為主程式與子程式的 unit time 是分開來算的，並且主程式除了執行 one unit time，還會執行其他判斷，所以主程式的時間相較子程式 unit time 是比較長一點的。也因此會造成子程式應該 ready 的時間，主程式的 local 所計算的還沒到達，在子程式結束的時候也是大同小異。
由於是採用一個 CPU 跑主程式，另一個 CPU 跑子程式，雖然這樣可以大幅減小兩者的干擾，但要分配 CPU 也會造成主程式的延遲，還有子程式呼叫 system call 會從 user space 轉換到 kernel space 造成延遲，也因此實際時間跟理想時間會有所誤差。

Reference

1. Linux kernel space get time interval
2. getnstimeofday
3. sched_setscheduler
4. [Linux C] fork 觀念由淺入深
5. struct sched_param 结构体结构
6. Linux process priorities and scheduling 心得
7. System Call
8. How does copy_from_user from the Linux kernel work internally?
9. copy_to_user