OS Project1 report

1. **設計**

   設計上，我將我的code分成main.c 以及 scheduler.c 以增加可讀性。在 main.c 中，主要是處理從 stdin 的輸入，並呼叫函式到scheduler.c 來做 cpu scheduling 的動作。而在scheduler.c 中，我採用的是雙核心的做法，一個core用來分配工作，而另一個core則實作cpu scheduling。在cpu scheduling 中，每一個UNIT_T都會先檢查剛剛在跑的process 是否結束，若剛好結束則交出cpu使用權。接著，會檢查這個時間點有無process剛好來到ready time，若有則fork一個新的child process，並等待進入cpu。接著會依據使用的排程演算法來決定在ready 的process 中（不含已經完成的），哪一個process可以獲得cpu的使用權力。最後，若發現這一個UNIT_T應該要由不同的process來使用cpu，則利用sched_setscheduler中的priority來實作context switch的過程。而system call 的部分，我則是實作了sys_get_time 及 sys_print來分別獲得當下時間以及printk訊息到dmesg。

2. **核心版本**

   輸入 *uname -a* 可得到：

   ```
   Linux ubuntu 4.14.25 #3 SMP Mon Apr 27 02:18:42 PDT 2020 x86_64 x86_64
   x86_64 GNU/Linux
   ```

3. **比較實際結果與理論結果，並解釋造成差異的原因**

   查看理論及實際結果後發現雖然趨勢相同，但是實際上花了更多時間。我想這可以歸因於許多原因，比如說context switch本身就有overhead，或者說我們的code跑到一半時根本就被context switch到其他的process去...等等。況且，我們程式執行的地方為user space，但我們並不清楚實際上kernel space是如何去跑的，因此實際跟理論可能難免會有一些出入。