**Project 1: CPU Scheduling**

by 黃謙仁, 林群崴, 傅家靖, 黃懋翔, 胡程維, 高新造

1. **設計：**

**1-1 實驗方法：**

本次實驗使用 Linux Kernel 4.19.37搭配Ubuntu 16.04 OS System。 利用修改（增加system call)過的Ubuntu 16.04 System call 取得系統時間並配合Linux Kernel 4.19.37 原生文件來達到模擬計算各Process Executiom Time。

為了精準計算時間，我們利用 sched\_setaffinity() 將parent process 與 child process 分別assign 至不同 cpu，以避免當 parent process 在計算 time unit 時 cpu time 被 child process 搶走。

**1-2 增加系統System call：**

在/home/linux-4.19.36/kernel中實作欲新增的System call，即 SYSCALL\_DEFINE3(hello\_3,int,state,int,pid,timespec\*,start）並於/home/linux-4.19.37/arch/x86/entry/syscall\_64.tbl 中增加System call entry。

此次Project增加取得系統時間的system call，並在每次fork() child process時使用syscall(336, 1, mypid, start) 取得開始執行時間，並於child process結束時利用 syscall(336, 0, mypid, start)取得結束時間。

**1-3 FIFO / RR**

將每個要執行的child process依照ready time進行non-decreasing排序， 並從ready queue 中依first in first out(ready time最早的先完成)的規則抓取執行，同時parent process 會以Unit time的標準計算時間來判斷child process執行了多少 time unit。

RR(Round Robin)步驟同上,只是每個 child process最長執行時間 (time quantum) 為固定的 500 time units。

**1-4 SJF：**

將每個要執行的child process依照 ready time 與 execution time 進行non-decreasing排序, 排愈前面priority愈高. 步驟同FIFO，差別只在於ready queue 的排序方法(執行時間最短的先完成)。

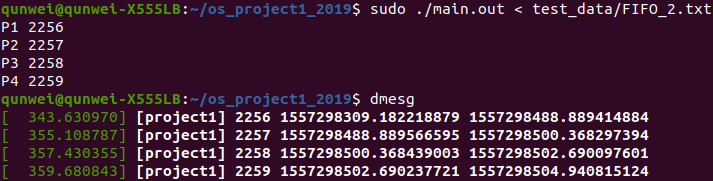
**1-5 Preemptive-SJF：**

步驟同SJF，差別在於 Preemptive 版本會在規定的時間檢查ready queue中 child process 剩餘執行時間，並把擁有最短剩餘時間的 child process 插隊置於 ready queue 最前端。

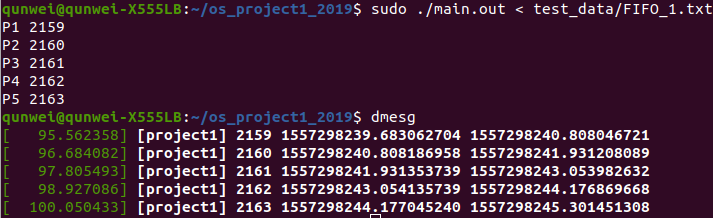
**2. 執行範例測資的結果：**

**First In First Out組(已修正)**

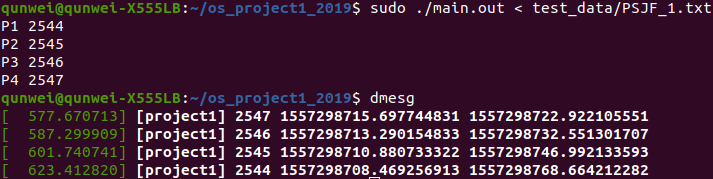
**FIFO\_1**



**FIFO\_2**

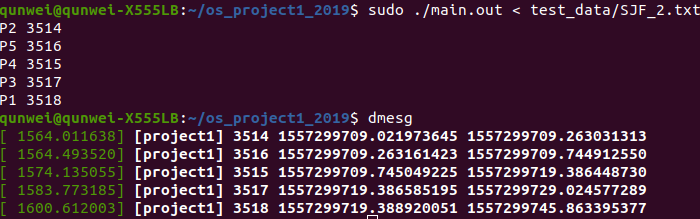
****

**FIFO\_3**

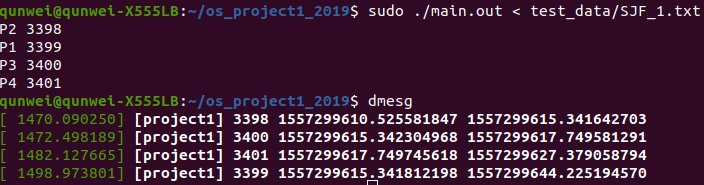


**Shortest Job First組**

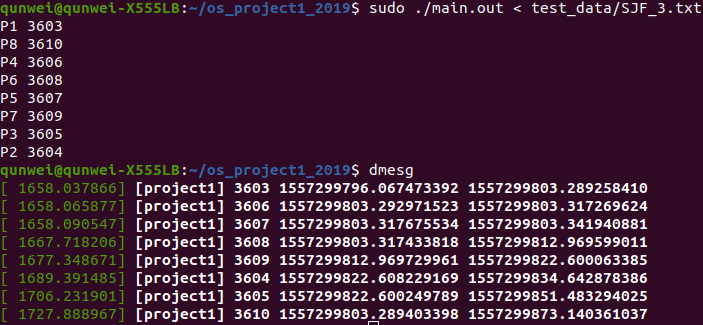
**SJF\_1**



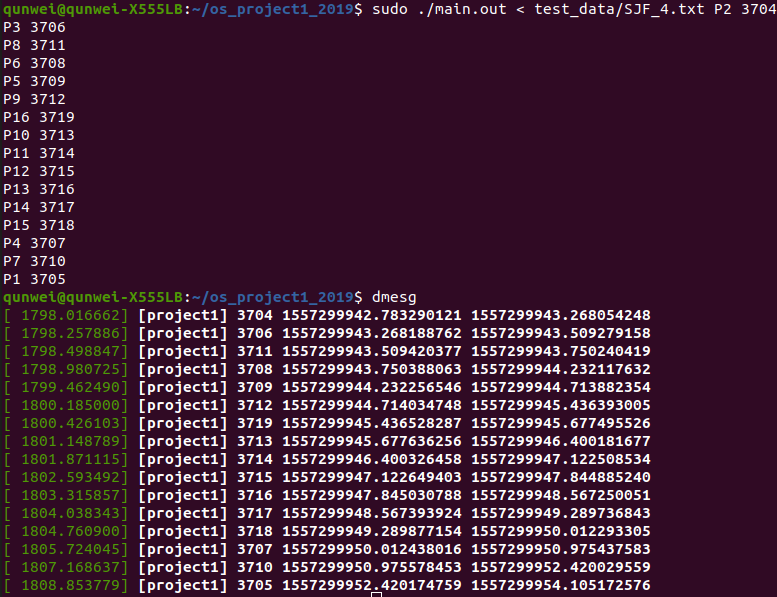
**SJF\_2**

****

**SJF\_3**

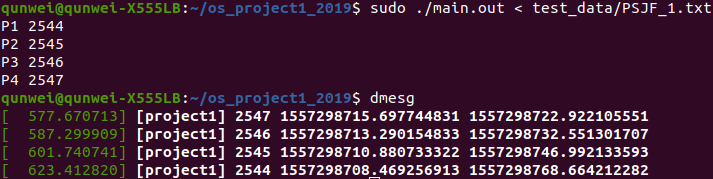


**SJF\_4**

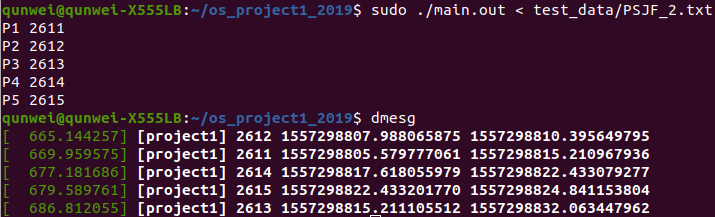


**Preemptive Shortest Job First組**

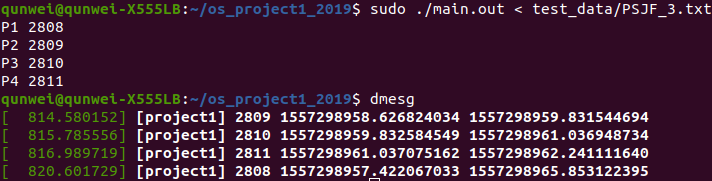
**PSJF\_1**



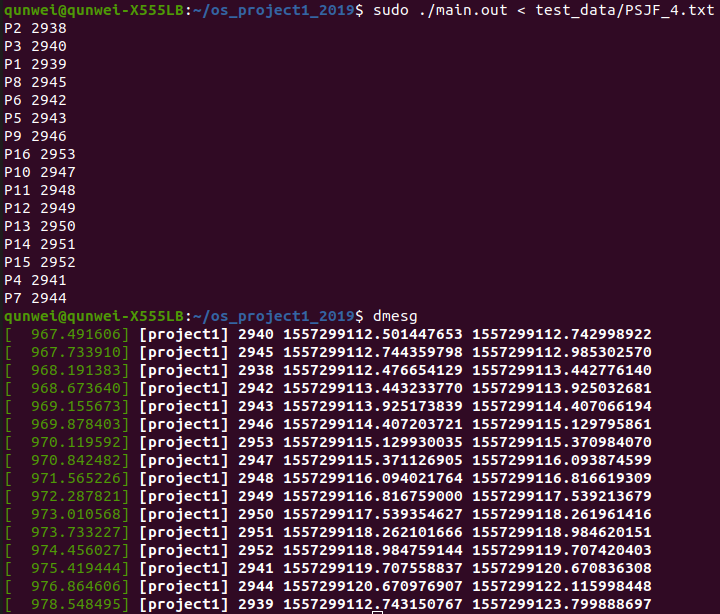
**PSJF\_2**



**PSJF\_3**

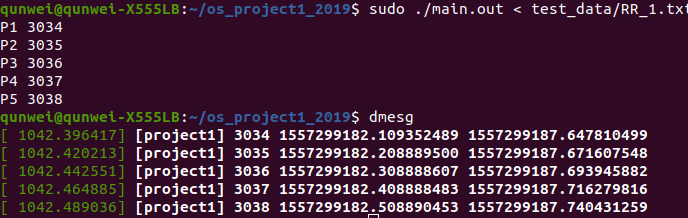


**PSJF\_4**

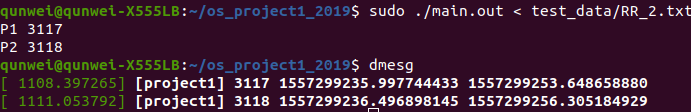


**Round-Robin組**

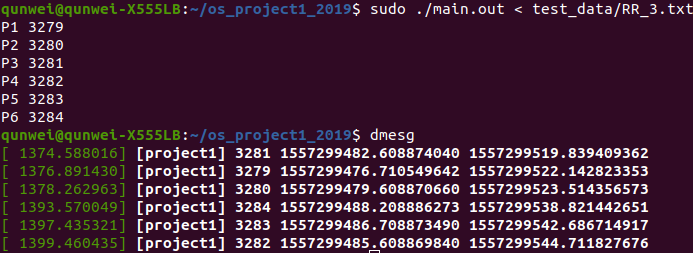
**RR\_1**



**RR\_2**



**RR\_3**



**3. 比較實際結果與理論結果：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time Unit =  1unit/0.0013571963198s  (小數點以後四捨五入) | ＦＩＦＯ | ＳＪＦ | ＰＳＪＦ | ＲＲ |
| \_1 | **實際:2434**  **理論:2500** | **實際:13602**  **理論:14000** | **實際:22270**  **理論:25000** | **實際:2432**  **理論:2500** |
| \_2 | **實際:85136**  **理論:87000** | **實際:14860**  **理論:15300** | **實際:9704**  **理論:11000** | **實際:10829**  **理論:9000** |
| \_3 | **實際:22349**  **理論:23000** | **實際:31176**  **理論:32020** | **實際:2916**  **理論:3500** | **實際:26826**  **理論:30000** |
| \_4 |  | **實際:4571**  **理論:4700** | **實際:4561**  **理論:4700** |  |

註：會造成實際值比理論值還小，猜測是因為Time unit的計算過大，導致實際值偏小。

**4. 各組員的貢獻：**

黃謙仁：搜索多數資料提供組員參考，並將觀念和技巧確實運用於報告加速整體流程。

林群崴：實際編組system call，實現利用真實linux kernel 來模擬排程的狀況。

傅家靖：詳實記錄方法並完成報告架構，實際執行測資結果並列表。

黃懋翔：組織組員會議，訂定具體完成時間，反覆測試程式正確性。

胡程維：推進計畫流程，提出關鍵個人意見，對報告進行校正和核對。

高新造：因為亂流加入的組員，未來有望成為潛力股生力軍！