# Document

資訊三甲。10727138。游子諭

## 1. 開發環境

本次我用的開發環境為 Windows 系統上的 Dev C++。

## 2. 實作方法和流程

**讀 input**

我利用 struct 去儲存每個 process 的資料，而 process 中的 ID 、 CPU Burst 、 Arrival Time 以及 Priority 各是 struct 中的一個欄位。利用 struct 去創造 vector 以此來儲存多筆的 processes 資料。而在一開始，也必須要記錄 command 和 time 。

**FCFS**

以 Arrival Time 當作第一順位，ID 為第二順位，藉此來排定使用 CPU 的排程。

每一次在找新的 Process 來放進 CPU 時，都利用 for 迴圈一個一個找，當找到後，就利用 Boolean 函數 來記錄他是否已經進入過 CPU，而我也創造一個專屬於 CPU Schedule 的 Vector，裡面專門記錄每個當下是哪個 Process 正在使用 CPU。

因此，每個 Process 在被選擇後，會依據該 Process 的 CPU Burst 在 CPU Schedule 中新增對應數量的資料，也因為 FCFS 沒有 Preemptive ，所以才可以這樣用。

**RR**

因為 RR 有 Time out 的問題，因此我在此段 code 增加了 time 的變數，目的在於紀錄現在當下的 CPU 時間。

我仍然在 CPU Schedule 上利用和 FCFS 相同的資料結構，同時，為了實現 RR 對於 Process 的優先序，我額外寫別的 Function ，他們能夠根據當下的 time 來回傳給我最優先該處理的 Process ，以此來方便我做 CPU Schedule 的工作，同時在確定該 Process 的 CPU Burst 都結束後，才利用 Boolean 函數紀錄。

**SRTF**

基本上和 RR 相同的程式結構，比較麻煩是有 Preemptive 的問題，所以必須隨注意當下 time 是否有更應該先被做的 Process ，並

讓她插隊，同時也要記錄說每個 Process 剩餘的 CPU Burst。

**PPRR**

簡單來說就是 RR 和 Preemptive 機制的融合，這必須整合我在 RR 和 SRTF 的兩樣東西，但因為要同時考慮 Time Slice 和 Priority 的問題，所以每個時段的 time 都必須確認好每個 Process ，現在這個當下最應該先做的 Process 為何。

**HRRN**

跟第一個 FCFS 是相同結構，只是這次的 Priority 計算上比較複雜，但只要找到當下最優先的 Process 後，就直接依據 CPU Burst 直接在 CPU Schedule 上做紀錄就可以了。

**Waiting Time && Turnaround Time**

我主要是在各 Process 結束 CPU Burst 的時候，利用當下時間扣除 Arrival Time 來計算 Turnaround Time，在用這項結果扣除 CPU Burst 來計算 Waiting Time，並利用 Vector 去紀錄。

## 3. 不同排程法的比較

waiting

| ID | FCFS | RR | SRTF | PPRR | HRRN |
|---|---|---|---|---|---|
| 0 | 19 | 18 | 0 | 0 | 19 |
| 1 | 13 | 8 | 0 | 0 | 5 |
| 2 | 22 | 19 | 2 | 14 | 16 |
| 3 | 18 | 25 | 6 | 0 | 14 |
| 4 | 13 | 19 | 0 | 11 | 13 |
| 5 | 20 | 27 | 19 | 21 | 23 |
| 6 | 0 | 15 | 6 | 11 | 0 |
| 7 | 15 | 2 | 0 | 55 | 3 |
| 8 | 21 | 14 | 0 | 9 | 11 |
| 9 | 5 | 13 | 1 | 0 | 6 |
| 10 | 8 | 37 | 49 | 45 | 18 |
| 13 | 18 | 3 | 0 | 0 | 4 |
| 20 | 13 | 17 | 0 | 40 | 13 |
| 27 | 16 | 28 | 19 | 10 | 9 |
| 29 | 14 | 31 | 19 | 4 | 20 |

**（以上為老師測資中 input1 的 waiting time）**

平均等待時間

FCFS：14.334
RR：16.4
SRTF：8.067
PPRR：14.667
HRRN：11.6

這只是其中一個測試資料，但有趣的是，當該排程法具 Preemptive 的特性，容易出現某幾個 Process 的 waiting time 特別久的問題，就像 SRTF 和 PPRR，而這也有可能發生如果一直出現 Priority 相當靠前的 Process ，那有些 Process 確實會 starvation。

而雖然大家都希望有公平的排程法，但在某些狀況下，他們的平均等待時間又比 Preemptive 多很多，根據不同的測資，各有不同的排程法有優勢。