# OS HW2 document

10827201 簡湘媛

1. 開發環境 :

使用語言:Python 3.8

使用notepad++撰寫

用CMD執行

2. 實作方法和流程 :

在一開始輸入檔案名稱，開啟並讀檔，區分出檔案內容，如第幾個方法、time slice、ID、arrival time 等等。首先將關於 Process 內容存成 list 後，在依據 arrival time 排序。不管是要哪種 case 皆會把所有排程都會做完，最後在分不同的 case，將需要內容寫到 output file 中。

FSCS：因為一開始已經把 process 依照 arrival time 排好。因為 FSCS 為不可奪取的排程法，所以利用迴圈將 list 中一個一個 Process 按照已排好的順序進行，在進行的途中需要注意是否 arrival time 已經小於或是等於現在排程時間(甘特圖 list 的大小)，若以到達將 ID 轉化成需要的格式，依照 CPU Burst 的大小寫入甘特圖的 list 中，如 ID：10, CPU burst：3, 甘特圖：AAA。一但一個 Process 做完就計算其 waiting time 以及 turnaround time，並且將分別存入 waiting time 的 list 以及 turnaround time 的 list。

RR：創一個 list，來負責取下一個 Process 在 Process 中的 index ，每次第一個，取出後會將其刪除。首先，將當前甘特圖 list 的大小等於其 process 的 arrival time 的 Process 依照若不只一個依照 ID 順序存入 list 中。每次皆從 list 中取排在最前面的 index，找到相對的 Process 以 time slice 大小進行甘特圖排入，每排入一次便將 Process 的 CPU Burst 減一，在此過程中，如果發現已到達的 Process 將其 index 存入 list 中。當 time out 或是 process 經完成時，判斷 Process 是否已經完成，若未完成會將其重新加到 list 中，但是若有新的 Process 到達，則先要讓其優先加入後再加入，若已完成計算其 waiting time 以及 turnaround time，並且將分別存入 waiting time 的 list 以及 turnaround time 的 list。再來會從 list 中取出下一個 index，進行下一個 Process 進行排程，直到所有 Process 的 CPU Burst 皆為 0 結束。

SRTF：比較甘特圖 list 的大小(現在時間)以及 Process 的 arrival time，若發現有 Process 到達時，比較已到達的 Process 中 CPU Burst 最小(不包含已結束)的 Process 先進入排程，每排入甘特圖 list 一次 CPU Burst 減一，若發現 Process 已經結束或是有新到達的 Process，再次進行 Process 中 CPU Burst 的比較，最小的進入排程。若 Process 已結束計算其 waiting time 以及

turnaround time，並且將分別存入 waiting time 的 list 以及 turnaround time 的 list。直到所有 Process 的 CPU Burst 皆為 0 結束。

PPRR：比較甘特圖 list 的大小(現在時間)以及 Process 的 arrival time，若發現有 Process 到達時，開始找 priority 最優先的，並且存在一個 list 中。若 list 中只有一個 Process 則直接進行排程，若有超過兩個以上的 Process 則進行 RR 的排程，有一個會記錄下一個 process 的 index 的 list，每次從 list 中取最前面的進行排程。若排程過程中，有新的 Process 進入則要開始找 priority 最優先，在依照先前的流程進行，若是在進行 RR 排程期間內遇到的話則要把先前的還在等待的排程紀錄下來，等到下次再度輪到此 priority 時，優先尋找上次紀錄的排程順序進行排程，若發現此排程未包含此 priority 的所有未完成的 process 在利用 arrival time 給入準備的排程 list 中，再次進行 RR 排程。若目前 Process 已經結束後，計算其 waiting time 以及 turnaround time，並且將分別存入 waiting time 的 list 以及 turnaround time 的 list。重複上述步驟，直到所有 Process 完成。

HRRN：比較甘特圖 list 的大小(現在時間)以及 Process 的 arrival time，若發現有 Process 到達時，比較已到達的 Process 中計算 Response Ratio 最小(不包含已結束 CPU Burst 為 0)的 Process 先進入排程，每排入甘特圖 list 一次 CPU Burst 減一，直到發現 Process 已經結束後，計算其 waiting time 以及 turnaround time，並且將分別存入 waiting time 的 list 以及 turnaround time 的 list。再次做 Process 的 Response Ratio 比較，最小的進入排程。重複到所有 Process 的 CPU Burst 皆為 0 結束。

最後依據指定方法，將甘特圖、Waiting time 及 turnaround time 的結果，利用迴圈依照規定格式寫入 output 檔中，最後輸出 output file，程式結束。

3. 不同排程法的比較

平均等待時間（Average Waiting Time）

| | FSCS | RR | SRTF | PPRR | HRRN |
|---|---|---|---|---|---|
| Input1 | 14.333 | 18.4 | 8.06 | 14.666 | 11.6 |
| Input2 | 8.4 | 6.4 | 3.0 | 9.4 | 8.2 |
| Input3 | 6.667 | 11.667 | 6.667 | 12.5 | 6.667 |

由上面表格可以看出，在所有不同的input中，SRTF為最快的，因為他會依照CPU Burst來進行排程，讓所需時間較小的先做，這樣時就不用讓所需時間小的等待所需時間大的做完，自然地降低了等待所需的時間。至於其他的排程，比

較看重已開始進來的順序，以及注重的細節比如說RR比較注重每個Process可以輪流的執行，減少後面Process等待的時間，而PPRR而是比較注重priority進行排程，而HRRN則是注重現在等待時間和所需時間的比例來進行排序。

工作往返時間（Turnaround Time）
工作往返時間是跟waiting time加上 CPU Burst，所以說若CPU Burst時間相同的話，影響Turnaround time的就是waiting time。若waiting減少自然而然的turnaround time也就會減少。所以說可以看到在output中相同的Process，一旦一排程的Waiting time 較長，而跟著Turnaround time 也較長。

4. 結果與討論
每種排程都會有各自注重的方向，並不是所有的排程注重的都是 waiting time。還有很多是注重公平性，優先權等等。每種排程都有自己的優缺點，

一、FCFS 程序不會被餓死，但有可能會因為前面的 Process 有很長的 waiting time。
二、RR 則是要求每 process 都會輪流的進行，若 CPU Burst 較少的但是排在後面，可以比起 FCFS 還要更快的完成，因為等待時間相對的可能減少，另外還有 time slice 的影響，若 time slice 太長的話會造成前述等待時間的優點無法呈現出來。
三、SRTF 的優點就是看 CPU Burst 來進行排程，所以可以找到 CPU burst 少的進行，有效的減少等待時間，但是在現行上這個很難達成，因為系統並不會知道這個 Process 會需要多少的 CPU Burst time，所以這個排程較難達成。
四、PPRR 而言就是優先看 Process 的優先權，讓優先權較高(Priority 小)的可以優先執行，若優先相同則是採取輪流執行的，可以使多個 process 公平的執行。
五、HRRN 這個就是算比例算出 turnaround time 和 CPU Burst 的比例，此比例會隨著等待時間的加長而變大，相對的若一個 Process 等待夠長的時間後，就有機會變成下一個的執行序。

所以說，所有排程都是有自己的一套道理，沒有什麼是最好的，要看目前的需求來決定要用甚麼樣的排程。