**資訊三甲 10827102 沈柏融**

**開發環境 :**

**Dev-c++ 5.11 version**

**實作方法和流程 :**

**依照使用者鍵入的檔案名稱讀取資料，並儲存進特定list中，完成指定的排程方法後將結果輸出至新的.txt檔中**

**FCFS :**

**依照Arrival\_Time的先後次序，對原先讀入的list進行bubble sort ，如遇到相同的 Arrival\_Time 則比較其ID大小，每完成一個ID紀錄其完成時間，並算出Turnaround\_Time 和 Waiting\_Time儲存至vector中**

**RR:**

**依照Arrival\_Time的先後次序，若Arrival\_Time相同則比較其ID大小，將已到達的Process放進queue中，每一個Process的Time\_Slice用完時就必須回到queue的最後面，若此時剛好有新的Process到達，則必須讓新的Process排在前面，直到所有的Process執行完畢，且每完成一個ID紀錄其完成時間，並算出Turnaround\_Time 和 Waiting\_Time儲存至vector中**

**SJF:**

**依照CPU\_Brust的大小排序，若剩餘CPU\_Brust相同則依照Arrival\_Time的先後次序，若Arrival\_Time相同則比較其ID大小，執行完一Process再重新依照CPU\_Brust的大小排序，且每完成一個ID紀錄其完成時間，並算出Turnaround\_Time 和 Waiting\_Time儲存至vector中**

**SRTF:**

**依照CPU\_Brust的大小排序，若剩餘CPU\_Brust相同則依照Arrival\_Time的先後次序，若Arrival\_Time相同則比較其ID大小，決定下一個時間單位要執行哪個Process，且每完成一個ID紀錄其完成時間，並算出Turnaround\_Time 和 Waiting\_Time儲存至vector中**

**PPRR:**

**依照Priority的大小排序，若有多個Priority的大小相同，則**

**採用RR的原則將多個Process放進queue中，若有更小於之前所有的Process出現，則優先處理Priority最小的排序，且每完成一個ID紀錄其完成時間，並算出Turnaround\_Time 和 Waiting\_Time儲存至vector**

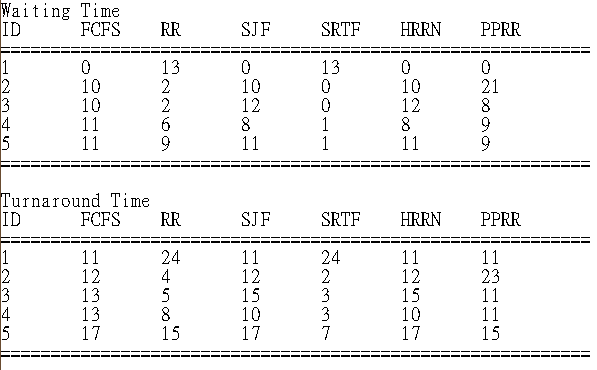
**HRRN**

**依照反應時間比率的大小排序，若反應時間比率的大小相同則依照Arrival\_Time的先後次序，若Arrival\_Time相同則比較其ID大小，決定下一個時間單位要執行哪個Process，且每完成一個ID紀錄其完成時間，並算出Turnaround\_Time 和 Waiting\_Time儲存至vector中**

**不同排程法的比較 :**

**以範例input2.txt來比較**

**FCFS**

**Average waiting times : 8.4**

**Average turnaround times: 13.2**

**RR**

**Average waiting times : 6.4**

**Average turnaround times: 11.2**

**SJF**

**Average waiting times : 8.2**

**Average turnaround times: 13**

**SRTF**

**Average waiting times : 3**

**Average turnaround times: 7.8**

**PPRR**

**Average waiting times : 9.4**

**Average turnaround times: 14.2**

**HRRN**

**Average waiting times : 8.2**

**Average turnaround times: 13**

**解果與討論 :**

**FCFS**

**因為一定要處理完當前程序才會讓出CPU，所以I/O繁忙的環境非常不利，比較有利於長時間的作業**

**RR**

**因為必須要執行完一個時間片段，因此非常看重時間片段的長度設定，過長的話跟FCFS沒什麼區別，過短的話因為要多個時間片段才能完成，響應時間會被拉長，像是範例的input1.txt**

**SJF**

SJF單純討論誰最快就先跑誰**，雖然有了更好的排班效益，2但嚴重偏好short job，對long-burst-time的排程非常不公平，需要等待非常久的時間**

**SRTF**

**比起SJF，他要求的是剩下的時間，如果有Process更快，將會允許他插隊，雖然有了更好的排班效益，但會付出更多的contest swich 的代價**

**PPRR**

**比起RR，多了Priority這一個參考值，但我目前不知道其存在的原因**

**HRRN**

**每當要進行作業排程的時候，都會計算一次反應時間比率，也因為如此，就算進行長時間作業，隨著時間增加，後面的排程也有機會獲得排程執行，但也因為每次都多了計算反應時間比率，增加了系統的開銷**