作業系統　HW2 文件

資訊四甲 10827111 袁本誠

一、開發環境

作業系統: windows

程式語言: C++

開發軟體: dev C++

二、實作方法與流程

主程式流程:

1. 使用者輸入檔名，並將資料放入 vector
2. 根據方法及時間片段，進入相應的排程函式
3. 執行排程函式
4. 寫檔
5. 回到步驟 1

FCFS:

1. 依照抵達時間、 ID 排好最初的資料
2. 依照排程時間(count)，將資料依序取出並執行( 由於第一步已經排好抵達時間，只要排程的時間大於等於抵達時間，就可以直接按照順序執行 )
3. 計算 waiting time 和 turnaround time
4. 回到步驟 2，直到全部資料處理完畢。

RR:

1. 依照抵達時間、 ID 排好最初的資料
2. 依照排成時間(count)，當時間一到就把符合的時間丟進佇列(queue)
3. 如果佇列沒東西，代表沒有處理程序要執行，記錄甘特圖並回到步驟 2
4. 處理 queue 的第 0 個，並同時執行步驟 2，如果執行時間超過時間片段，重新回 queue 排隊
5. 如過 queue 的第 0 個執行完畢，計算 waiting time 和 turnaround time ，並將結果紀錄下來後，從 queue 中移除該 process
6. 重複步驟 2、3、4、5 ，直到處理程序全部完成

SJF:

1. 依照抵達時間、 ID 排好最初的資料
2. 排成時間(count)當作搜尋範圍，執行符合範圍內的資料中 burst 最小的 process 。
3. 計算 waiting time 和 turnaround time 並將結果紀錄下來後移除該筆資料

4. 重複步驟 2、3，直到處理程序全部完成

SRTF:

1. 依照抵達時間、 ID 排好最初的資料
2. 排成時間(count)當作搜尋範圍，執行符合範圍內的資料中剩餘 burst 最小的 process 一次( count+1 )
3. 如果剩餘 burst 是 0，代表執行完畢，計算 waiting time 和 turnaround time，並將結果紀錄下來後移除該筆資料
4. 重複步驟 2、3 直到處理程序全部完成

HRRN:

1. 依照抵達時間、 ID 排好最初的資料
2. 排成時間(count)當作搜尋範圍，執行符合範圍內的資料中 ratio 最大的 process 。
3. 計算 waiting time 和 turnaround time 並將結果紀錄下來後移除該筆資料
4. 依照排成時間更新符合範圍內的資料的 ratio
5. 重複步驟 2、3、4，直到處理程序全部完成

PPRR:

1. 依照抵達時間、 ID 排好最初的資料
2. 依照排成時間(count)，當時間一到就把符合的時間丟進佇列(queue)
3. 判斷佇列(queue)有沒有東西，如果沒有，代表沒有處理程序要執行，記錄甘特圖、更新 count 並回到步驟 2
4. 選取 process 丟到另一個準備執行的 queue 中(sameP)，方式分為以下兩種:

   判斷是否有 process 曾經被奪取

   (1) 如果沒有 process 曾經被奪取，或者有被奪取但還未開始執行奪取的 process，從佇列中(queue)選取高優先度的 process，丟到另一個準備執行的 queue 中(sameP)。

   (2) 如果有 process 曾經被奪取，且已執行完奪取的 process，從 preempQueue 中取出第一個佇列(即那些被奪取且相同優先度的 process) 丟到另一個準備執行的 queue 中(sameP)，並確保皆比 queue 中的優先度高，以及將 queue 中相同優先度的 process 加入 sameP
5. 處理 sameP 的第 0 個，同時執行步驟 2，在此期間能將相同優先度的 process 加入到 sameP，如果出現更高優先度 process，立即中斷執行

   。如果執行時間超過時間片段，重新回 queue 排隊。
6. 如過 sameP 的第 0 個執行完畢，計算 waiting time 和 turnaround time ，並將結果紀錄下來後，從 queue 中移除該 process
7. 如果第 5 步有出現更高優先的 process，將 sameP 的資訊保留到 preempQueue 中

8. 重複步驟 2、3、4、5 、6、7，直到處理程序全部完成

三、不同排成法比較

1. input1

| | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| Avrange_waitTime | 14.3333 | 18.4 | 8.86667 | 8.06667 | 11.6 | 14.6667 |

Turnaround Time:

Turnaround Time

| ID | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| 0 | 23 | 22 | 9 | 4 | 23 | 4 |
| 1 | 15 | 10 | 7 | 2 | 7 | 2 |
| 2 | 25 | 22 | 5 | 5 | 19 | 17 |
| 3 | 22 | 29 | 10 | 10 | 18 | 4 |
| 4 | 16 | 22 | 10 | 3 | 16 | 14 |
| 5 | 26 | 33 | 25 | 25 | 29 | 27 |
| 6 | 5 | 20 | 10 | 11 | 5 | 16 |
| 7 | 16 | 3 | 3 | 1 | 4 | 56 |
| 8 | 23 | 16 | 2 | 2 | 13 | 11 |
| 9 | 9 | 17 | 4 | 5 | 10 | 4 |
| 10 | 16 | 45 | 57 | 57 | 26 | 53 |
| 13 | 19 | 4 | 5 | 1 | 5 | 1 |
| 20 | 16 | 20 | 8 | 3 | 16 | 43 |
| 27 | 22 | 34 | 15 | 25 | 15 | 16 |
| 29 | 20 | 37 | 21 | 25 | 26 | 10 |

Input2

| | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| Avrange_waitTime | 8.4 | 6.4 | 3 | 3 | 8.2 | 9.4 |

Turnaround Time

| ID | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| 1 | 11 | 24 | 11 | 24 | 11 | 11 |
| 2 | 12 | 4 | 12 | 2 | 12 | 23 |
| 3 | 13 | 5 | 15 | 3 | 15 | 11 |
| 4 | 13 | 8 | 10 | 3 | 10 | 11 |
| 5 | 17 | 15 | 17 | 7 | 17 | 15 |

Input3

| | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| Avrange_waitTime | 6.66667 | 11.6667 | 6.66667 | 6.66667 | 6.66667 | 12.5 |

Turnaround Time

| ID | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| 1 | 20 | 20 | 20 | 20 | 20 | 20 |
| 2 | 25 | 45 | 25 | 25 | 25 | 55 |
| 3 | 45 | 55 | 45 | 45 | 45 | 60 |
| 4 | 30 | 30 | 30 | 30 | 30 | 15 |
| 5 | 10 | 10 | 10 | 10 | 10 | 20 |
| 6 | 15 | 15 | 15 | 15 | 15 | 10 |

Input4

| | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| Avrange_waitTime | 3.75 | 5.5 | 3.25 | 3.25 | 3.75 | 4.5 |

Turnaround Time

| ID | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| 1 | 6 | 11 | 6 | 9 | 6 | 15 |
| 2 | 7 | 5 | 7 | 3 | 7 | 3 |
| 3 | 10 | 14 | 15 | 15 | 10 | 15 |
| 4 | 12 | 12 | 6 | 6 | 12 | 5 |

四、結果與討論

1.FCFS：屬於不可奪取算法。以數據上來看，可能會導致稍後到達的任務等待時間過長，並且可能導致執行時間較長的任務性能不佳。

2.RR：屬於可奪取算法，可以讓資源的公平分配並防止飢餓。

3.SJF：屬於不可奪取算法，可以有最佳平均等待時間。但是，它可能導致長時間任務餓死。

4.SRTF：屬於可奪取算法，可以有最佳的平均等待時間，但是，由於頻繁的輪替與輪尋是否有剩餘最小，可能會導致程式負擔。

5.HRRN：屬於不可奪取算法，可以有較佳的平均等待時間，並且反應時間比率會隨等待時間增加而提升，因此不會餓死。

6.PPRR：屬於可奪取算法，與 RR 數據性質大致相同

由以上數據，可發現 SRTF 在每個 input 的 Waiting Time 最小；FCFS 排程法依照 Arrive Time 由小到大進行排程，效率取決 job 來的時間以及 CPU Burs，如果今天先來的 job 的 CPU Burs 很大，那後續 job 的等待時間也會變非常大；RR 排程法及 PPRR 排程法則取決於所切的 Time Slice；HRRN 排程法在以上三種基礎測資的情況，表現看來最平均，有時能像 SRTF、SJF 一樣小的 Waiting Time。

HRRN 排程法能讓各 Job 等待時間差不多即被執行（避免飢餓）；而 SRTF 、SJF 排程法容易使自己的各 Job 等待時間差距較大（容易造成某行程飢餓）。