# 作業系統-HW2

資工三乙 11027222 黃彥霖

開發環境: Windows11, Visual Studio Code

語言: C++

**任務簡介&實作方法與流程**:總共分為 7 種任務

儲存資料的方法:

一個 vector 裡面自訂 struct 型別，struct 有 process 的 id,cpu burst time, arrival time, priority, waiting time, turnaround time 和 rr(Response Ratio),去對資料做儲存。

流程簡介:

設定一個 time=0，每做完一次迴圈會加一，去判斷有沒有 arrival time 符合當前時間的 process，有的話加入 queue 裡面,queue 會根據不同排程法而有不同設計，以下再詳細補充。並且選擇佇列最前面的 process 當作下一個執行的，如果佇列是空的話，會在檔案上記錄('-')做為中央處理器閒置中。不是空的話也會記錄當前中央處理器正在執行的 ID，並減掉 cpu burst time，如果為 0 就結束這個 process，紀錄 waiting time 和 turnaround time 並存起來。

1. FCFS(First Come First Serve):

說明:先到先執行的排程法。佇列的實作方式是根據抵達時間來加入，如果有相同的話去比較 ID。有新的 process 加入，丟入佇列再根據以上條件做排序，可滿足先到先執行的概念。此方法會等一個 process 完全做完才換下一個。

2. RR(Round Robin):

說明:知更鳥排程法。佇列的實作方式是根據抵達時間來加入，如果有相同的話去比較 ID。有新的 process 加入，丟入佇列再根據以上條件做排序。這個方法會給一個占用中央處理器的時間，時間用完會再回去佇列排隊，與此同時，有新加入的 process 可以比用完 time slice 的更優先進入佇列，每個 process 都享有完整使用 time slice 的權利，除非 process 本身做完，否則無法在執行中去搶奪中央處理器資源。

3. SJF(Shortest Job First):

說明:最短任務優先排程法。佇列的實作方式是根據中央處理器執行時間來加入，如果有相同的話去比較抵達時間，最後根據 ID。有新的 process 加入，丟入佇列再根據以上條件做排序，可滿足先到先執行的概念。此方法會等一個 process 完全做完才換下一個。

4. SRTF(Shortest Remaining Time First):

說明:可奪取最短任務優先排程法。佇列的實作方式是根據中央處理器執行時間來加入，如果有相同的話去比較抵達時間，最後根據 ID。有新的 process 加入，丟入佇列再根據以上條件做排序。因為是可奪取，每個時間都會檢查佇列中的剩餘工作時間有沒有比當前正在做的工作的剩餘時間小，有的話會把佇列最前面的 process 和中央處理器當前的 process 交換。

5. HRRN(Highest Response Ratio Next):

說明:最高反應時間比率優先排程法。佇列的實作方式是根據計算每個 process 的 response ratio 來加入，如果有相同的話去比較抵達時間，最後根據 ID。有新的 process 入加入，丟入佇列再根據以上條件做排序。此方法會等一個 process 完全做完才換下一個，每次都選擇有最高 response ratio 的 process，而在選擇 process 之前，會去更新佇列中的資料，去更新他們的 waiting time，以確保當前選擇的 process 在當下有正確的反應時間比率。

6. PPRR(Preemptive Priority + Round Robin):

說明:Preemptive Priority 和 Round Robin 結合的排程法，概念類似多階層佇列，會根據優先度在分類一次，相同優先度的則是以 RR 的方式進行排程。因為是可奪取的，所以每次有新的 process 近來都會直接加入佇列，且當前正在執行的 process 也會加入佇列，因為我的實作方法都放在同一個佇列內，使用 stable sort 可以確保已對優先度排序，而相同優先度的 process 會按照 RR 的方式排隊。有新 process 近來直接打斷當前 process 的方法可行，是因為他如果有更高的優先度，中央處理器會去選擇執行它，沒有最高優先度的話，也代表他會被打斷。確定在 time slice 結束或有新 process 加入佇列時，有做好排序即可。

7. ALL

說明:把上述 1~6 的排程法各做一次。

計算公式:

Waiting time = Turnaround time – cpu burst time – arrival time

Turnaround time = terminated time – arrival time

Response ratio = (cpu burst time + waiting time) / cpu burst time

## 不同排程法的比較:

根據範例 input1, input2, input3, input4 的輸出和計算結果，討論以上六種排程演算法各自的平均 waiting time 和 turnaround time。

(表 1)Average Waiting Time:

| | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| Input1 | 14.33 | 18.40 | 8.87 | 8.07 | 11.60 | 14.67 |
| Input2 | 8.40 | 6.40 | 8.20 | 3.00 | 8.20 | 9.40 |
| Input3 | 6.67 | 11.67 | 6.67 | 6.67 | 6.67 | 12.50 |
| Input4 | 3.75 | 5.50 | 3.50 | 3.25 | 3.75 | 4.50 |

(表 2)Average Turnaround Time:

| | FCFS | RR | SJF | SRTF | HRRN | PPRR |
|---|---|---|---|---|---|---|
| Input1 | 18.20 | 22.27 | 12.73 | 11.93 | 15.47 | 18.53 |
| Input2 | 13.20 | 11.20 | 13.00 | 7.80 | 13.00 | 14.2 |
| Input3 | 24.17 | 29.17 | 24.17 | 24.17 | 24.17 | 30.00 |
| Input4 | 8.75 | 10.50 | 8.50 | 8.25 | 8.75 | 9.50 |

➔ FCFC

會因為數據同時近來，而全部數據在排隊等待 cpu 執行，cpu burst time 越大所有的等待時間和結束時間會跟著被影響，如過 cpu burst time 比較小且 arrival time 相差比較遠的話，整體的 waiting time 和 arrival time 就不會太大。

➔ RR & PPRR

從上面兩張圖可以得知，有使用到 time slice 的排程法，會有相對較高的 waiting time 和 turnaround time，是因為全部的 process 都會輪流使用 cpu，只要這個 process 要花費的時間大於 time slice，就代表它一定會進入佇列去等待，會被奪取也是一樣的意思，一個 process 不能馬上做完，turnaround time 一定會大幅增加，在佇列等待也代表 waiting time 會大幅增加。

➔ SJF & SRTF

從 SRTF 的 average waiting time 和 turnaround time 可以知道，它在這六種排程法之中都是最佳的。因為它把會長期占用 cpu 的 process 往後排，burst time 較小的都能先做，大大減少平均的 waiting time，但實驗做出的每個 process 的 waiting time 和 turnaround time 來看，雖然平均下來等待時間很低，但一定會有幾個 process 因為一直被插隊而有很大的 waiting time 和 turnaround time。

而 SJF 和 SRTF 都會有這個因為 cpu burst time 小的 process 先做，而犧牲掉 cpu burst time 大的 process，這種比較不公平的現象出現。

| ID | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 13 | 20 | 27 | 29 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SJF | 5 | 5 | 2 | 6 | 7 | 19 | 5 | 2 | 0 | 0 | 49 | 4 | 5 | 9 | 15 |
| SRTF | 0 | 0 | 2 | 6 | 0 | 19 | 6 | 0 | 0 | 1 | 49 | 0 | 0 | 19 | 19 |

表 3:SJF 和 SRTF 的 waiting time(以 input1 為例)

➔ HRRN

以 response ratio 來決定下一個要執行的 process，這排程執行結果較接近 SRTF 但還是略高於它。跟 SRTF 比就是有限度地等待，有了 aging 的機制，等太久就有可能先去執行，不會像 SRTF 或 SJF，cpu burst time 大的 process，會在佇列中無限等待。也避免了餓死的問題。

**結果與討論:**討論各排程法的優缺點

➔ FCFS

優點:

1.方法實驗簡單

2.很公平的方法，每個 process 都會被執行

3.對於 cpu burst time 較小且不會 process 都集中在同一時間出現時，表現較好。

缺點:

1.如果有一個 cpu burst time 很長的 process 在前面，會導致後面抵達的 process 塞在佇列裡面。

➔ RR

優點:

1.每個 process 都能公平地被分配到 cpu 使用時間。

缺點:

1. time slice 的大小會影像到 RR 的效能。
   time slice 太大:表現得像 FCFS，失去 RR 的目的。
   time slice 太小:cpu 不斷的 context switch 降低效能。

➔ SJF

優點:

1. 提供較小的平均等待時間
2. 非常適合短 cpu burst time 的 process

缺點:

1.實際情況無法得知 cpu burst time，難以實現。

➔ SRTF

優點:

1.最小的平均等待時間。

2.可以搶奪的 SJF，可以更有效地處理 process。

缺點:

1.實際情況無法得知 cpu burst time，難以實現。

2.可能會有餓死的情況。

➔ HRRN

優點:

1.考慮了 process 的等待時間和執行時間，嘗試避免餓死(有 aging 的機制)。

2.相比 SJF，比較公平，給予等待很久的 process 更高的優先度

缺點:

1.每次有新的 process 進入佇列就要計算 response ratio，計算它也要用到 cpu burst time，實際使用較難達成

➔ PPRR

優點:

1.結合了優先級和 time slice 的方法，可以靈活處理各種類型的任務。

2.比較符合實際上的運作方式，因為高優先級的 process 通常不會做太久，反之，低優先級的 process 會有較長的執行時間，概念類似多階層佇列。

缺點:

1.有可能低優先級的 process 會有餓死的情況，因為它沒有提供 aging 的機制，提供等太久的 process 有先做的機會。

心得:各個演算法沒有說絕對的好或壞，而是在不同情況下，可以根據情況使用當下最優的排程法，而不同的需求也會用到不同的程法，如:要最小平均等待時間會用到 SRTF 等等，而且不同排程法也會因為 process 執行時間大小，同時進來的數量而影響，在不同情況下都會有表現良好和不好的時候。