**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。

1. 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。

1. 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。

1. 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。

1. 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。

1. 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。

1. 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。