作業系統 OS HW2 scheduling

I.開發環境

資訊三乙 10827234 彭桂綺

開發環境:Windows 11 (21H2)
開發工具:visual studio code [1.66.2(2022.04.11)]
使用語言:C/C++

II.實作方法及流程

首先我會選擇在windows 以c/c++ 實作此次作業是因為我最習慣的環境及語言就是如此,如果遇到 bug 也熟悉如何 debug。

總的來說,我將 5 個方法用 5 個 class 分別包裝,各自擁有自己的排程 function 、 甘特圖、 waiting time 及 turnaround time 變數,方便呼叫執行。

我將依序解釋不同排程法的實作流程:

我先將所有資訊讀進 vector 中,並先以 process ID 排序、再以 arrival time 排序,這樣一來可確保queue從最前面拿出的順序為時間最小、如時間一樣則 process ID 小。

- FCFS -- > (First Come First Serve) non preemptive

  這個排程法會用到 queue ,我自己寫了一個 circular queue ,使記憶體空間不被浪費,要滿足先進先出的條件,當現在要放東西進 queue 但是發現 queue 滿了,此時將 queue 的 size 擴充為兩倍 , 再繼續將東西放進 queue。回到正題,我將在每個時間點(clock)作判斷,當有 process 在此時此刻到達時,就將它(們)丟到 queue 後面,接下來將從 queue 最前面取出一個 process 執行。因為 vector 中的資訊都是排序過的,所以可以確定執行順序沒問題。當 CPU Burst 到了(此 process 執行完了),就將此 process 丟掉,並計算 waiting time 及 turnaround time,並記錄 gantt 圖。 clock +1 後再從 queue 中拿下一個來執行,直到 ready queue 中沒有 process 且 已經執行完的 process 數量跟總數量一致時就結束。

- RR ---- > (Round Robin) preemptive

  跟 FCFS 很像,只是要去記錄目前執行中的 process 已經使用了多少 time slice,當 time slice 用完就要到 queue 後面排隊,這裡要注意,應該要讓新來的 process 先進 queue 再讓 time out 的 process 進入 queue。結束條件跟 FCFS 一樣,就不多加贅述。

- SRTF -- > (Shortest Remaining Time First) preemptive

  此方法不用 queue,而改用 array。前半段將 process 放進 ready

array 的方式都跟前面一樣,只是從 array 中取 process 的方式不同,已經不是先進先出了,而是每次(每個 clock)都取剩餘 cpu burst 時間最少的 process,當此 process 執行完就將此 process 丟掉,並計算 waiting time……,後續作法跟 FCFS 一樣。

- PPRR -- > (Preemptive Priority + Round Robin) preemptive

  此方法不用 queue,而改用 array。前處理皆相同,此種方法每次都會從 array 中取出 priority 最小(優先度最高)的 process 來執行。如遇到相同的 process 就依照 RR 來作。如果有高優先的 process 插入執行或 time out,則到 array 後面排隊(要讓新 process 排在前面) ,直到所有 process 執行結束。

- HRRN -- > (Highest Response Ratio Next) non preemptive

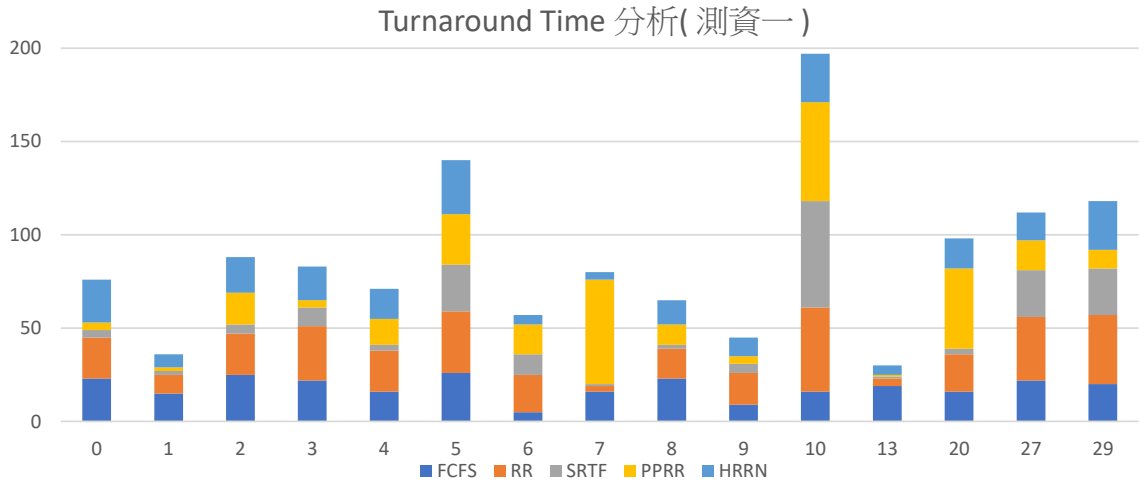
  此方法不用 queue,而改用 array。前處理皆相同,此種方法每次都會從 array 中取出 Response Ratio 最高的,並且是 non-preemptive,所以不管 time slice。做完一個 process 再執行 Response Ratio 次高的 process,直到所有 process 執行結束。

以我的實作方式來說, RR 及 PPRR 需要在執行次序上特別處理,要將 控制 clock 及 控制進入 ready queue 的 程式碼片段要放在 執行 process 的程式碼片段後,這樣邏輯才是對的。

III. 不同排程法的比較

- Turnaround Time 單位: 秒

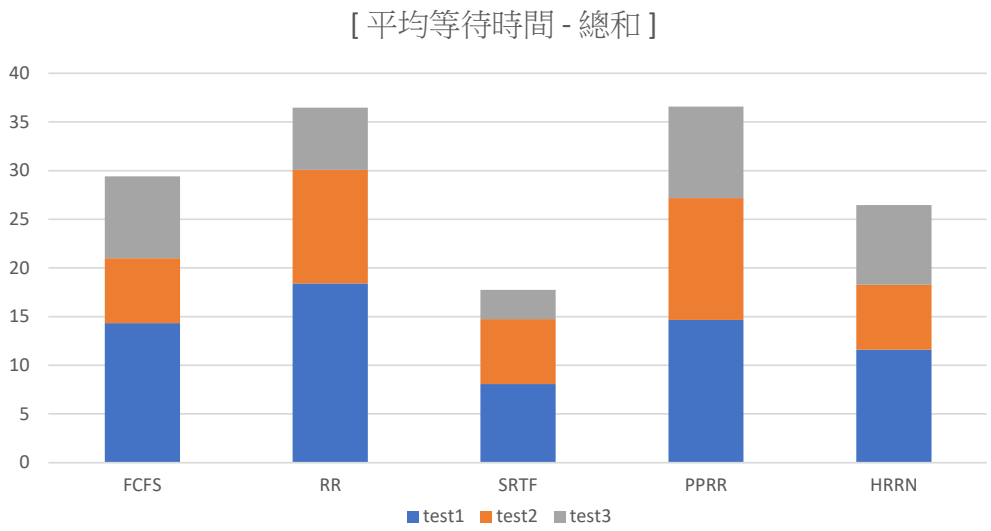
  Turnaround Time = finish time – arrival time 。



從上圖可以看出有些 process (5, 10),的 turnaround time 特別高,加上他們這兩個 process 使用 preemptive 的方法執行會有較高的 turnaround time ,這樣有可能發生等待較久的狀況。

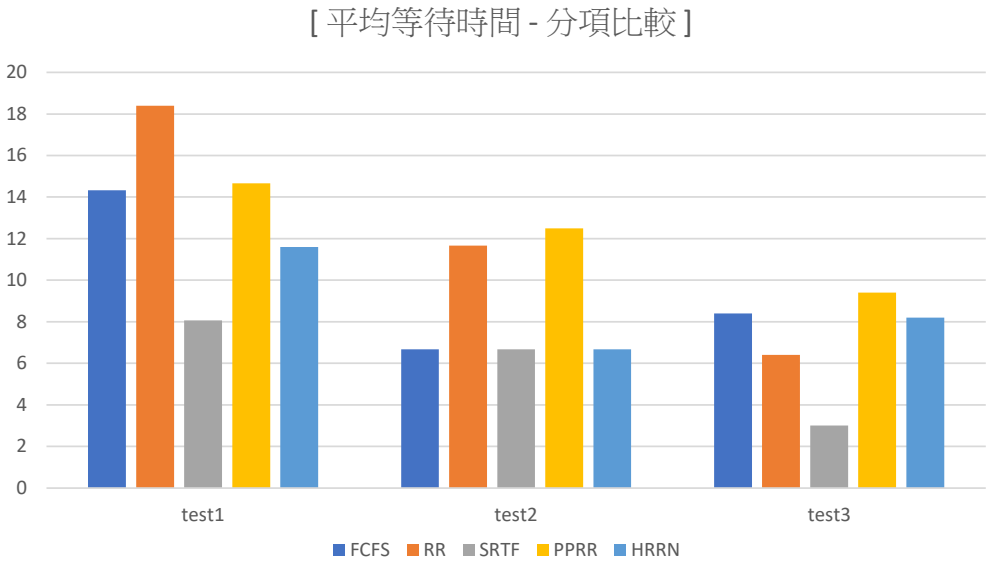
- Waiting Time 單位: 秒

waiting	FCFS	RR	SRTF	PPRR	HRRN	time slice
test1	14.33333	18.4	8.066667	14.66667	11.6	1
test2	6.666667	11.66667	6.666667	12.5	6.666667	3
test3	8.4	6.4	3	9.4	8.2	10
preemptive	non	Y	Y	Y	non	-



上圖是將三個測資的 平均 waiting time 相加之後的直條圖。從圖中可以看出 SRTF 為總等待時間最短的排程法,從下圖有可以看出,不論哪一個測資,SRTF 的平均等待時間都是最短的。

隨著 time slice 的上升,可以發現 preemptive 的排程法(RR, SRTF, HRRN) 的 waiting time 會跟著下降。



IV.結果與討論

中央處理器的其中幾個選用標準是:使得往返時間是最小的、使得所有處理程序之平均等待時間是最小,從前面的圖表可以知道 SRTF 是這五種方法最有效率的一種,他算是可以被插隊的 SJF,但是在實務上並不可行,因為不能知道這個 process 的 cpu 總執行時間是多少,所以沒辦法挑出真正剩餘時間最少的 process 來執行,但是 SRTF 也不失為一種理想的好方法。但 SRTF 也不一定就是最好的排程法,其他四個排程法也都有優缺點。

FCFS 看起來執行效率不錯,但是有機率發生 deadlock 進而造成餓死,當 FCFS 裡面有個 process 一直在執行,就不會換下個 process ,所以這個方法還是有風險(缺點)存在。

HRRN 的表現跟 FCFS 差不多,但是實務上也是無法達成,因為不知道 cpu burst 。

另外兩個 preemptive 的 procee (RR & PPRR) 就不太會發生 deadlock 的情況,因為它們會需要看 time slice,當 time out 就要換 下一個 process 來執行,有輪流的概念,也能運用在分時系統上。