開發平台

Window 10

使用開發環境

Dev C++

使用的程式語言

C++

功能

讀入一份包含所選擇的排程法 時間片段以及CPU的ID CPU Burst Arrival Time Priority 在依照各個排程法所定義的次序依序排程，最後會輸出該排程法得到的甘特圖 以及 每個process在做完的等待時間(waiting time ) 往返時間( Turnarround time ) 另外甘特圖的 id if > 9 會使用英文字母代替 ex A = 10, B = 11 以此類推

流程

1. 處理原則 – FCFS :

依Arrival Time 先後次序進行排程 ; 若Arrival Time 相同時 ,則依照Process Id 由小至大依序處理

1. 處理原則 – RR :

依ArrivalTime先後次序進行排序,時候未到的

Process不能執行 ; 若Arrival Time相同時,則依ProcessID由小至大 依序處理 ; 當Time out時,從佇列尾端開始排序,若有新來 的Process,則讓新來的Process排在前面;若Process的Time Slice未用完就結束時,必須讓下一個Process執行,且擁有完整的time slice

1. 處理原則 – SRTF :

由剩餘CPUBurst最小的Process先排序;若剩餘的CPUBurst相,則依Arrival Time小的先處理;若剩餘CPUBurst相单Arrival Time相同,則依 ProcessID由小至大依序處理。

1. 處理原則 – PPRR :

Preemptive Priority + Round Robin依Priority大小依序處理 , Priority的Process代表優先處理 ; 若Priority相的Process不只一個 , 採用RR原則進行排程:若有Priority相同的process正在執行中 , 須等待其時間片段用完 當Time out或被Preemptive時 , 從佇列尾端开始依Priority大小排序 , 若有新來的Process,則讓新來Process排在前面

1. 處理原則 – HRRN :

Highest Response Ratio Next 反應時間比率( Response Ratio ) 愈高的Process優先處理 ; 若Ratio相同的Process不只一個 , 則依Arrival Time小的先處理 ; 若Ratio相同且Arrival Time同 , 依Process ID由小至大依序處理。

使用的data structure :

使用到vector 動態的二維陣列 陣列裡面放的是 struct data

int slice = 0 ;int id = 0 ;int CpuBurstTime = 0 ;int ArrivalTime = 0 ; int Priority = 0 ;int TurnArround = 0 ; int Waiting = 0 ; int Finish = 0 ; double HRR ; bool completed = false ; bool inq = false ; bool havedobefore = false ;

vector <string > 也有使用到string的動態陣列 主要是用來存放每個排程法的甘特圖

queue<data> ;使用到 以陣列為基礎的佇列 Queue 用來存放需要等待執行的process 也有使用 vector 來存放需要等待執行的process

實作方法

一開始使用者輸入要開啟的文件檔 ex input1



Readfile() : 判斷檔案是否存在如果存在就開啟檔案 不存在就再次輸入文件檔，開啟後會將data 存在我們的vector 動態陣列 DataList ( 用來存放讀檔的資料 ) 以及要使用哪個排程法來排程( command ) , 讀到EOF後結束 關檔

==================================================================

FCFS :

先將資料依Arrival Time 先後次序進行排程 ; 若Arrival Time 相同時 ,則依照Process Id 由小至大依序處理 排序使用到 bubblesort() 排序完後將DataList 存放在mission1的 動態陣列中 用此陣列排程 我的寫法是先將第一筆資料做完 並且記錄當前的(duration) 的時間 之後從第二筆資料開始跑到 動態陣列的size結束為止 size 取決於資料量 中間要多判斷一 if( duration < mission1[i].ArrivalTime ) 如果目前的時間 小於進來的時間代表 資料尚未抵達 那為了讓程式正常運行 ,我們會強制讓下一筆資料抵達 duration = mission1[i].ArrivalTime ; 依此類推 一直跑, 算出當前資料的 TurnArround Waiting 時間 且記錄當前的(duration) 直到跑完 mission1 陣列裡的所有資料

那在每次process要做之前都會同步他的甘特圖記錄目前是誰執行那執行幾次在資料尚未抵達時會用 “-“ 來記錄 , 最後寫檔結束

==============================================

RR :

先將資料依Arrival Time 先後次序進行排程 ; 若Arrival Time 相同時 ,則依照Process Id 由小至大依序處理 排序使用到 bubblesort() 排序完後將DataList 存放在mission2的 動態陣列中 用此陣列排程, RR考慮到了 time slice 時間片段的問題 以及 Process 進來的次序 如果抵達時間與做完slice時間相同要讓還沒做過得先做 , 當排完序後我一樣會先將排完的第一筆資料先拉出來做 主要有兩個判斷 一個是判斷他的cpu burst time 是否 > slice 是的話就 更新當前的時間為 做完時間 將cpu time 扣一單位時間片段 ,如果 <= slice 一樣更新當前的時間 = cpu剩餘時間 並且算出當前process 的往返時間TurnArround 以及等待時間Waiting 做完之後去判斷是否有 其他的process 抵達要執行cpu 有的話我的做法是用一個 RRqueue 佇列用來存放 因為佇列是先進出所以方便管理 , 最後在判斷當前的process 是否完成了 之所以放在最後判斷是題目有說新進來的先做 , 如果完成了就不放進RRqueue裡面 未完成再放入 那有一點要注意!!就是如果 佇列已經空了 且下以筆資料沒有進入過佇列裡頭 ,代表前一個已經做完但下一個process 還未抵達if( duration < mission2[i].ArrivalTime )那為了讓程式正常運行 ,我們會強制讓下一筆資料抵達 duration = mission2[i].ArrivalTime ; 並且把下一筆資料push 進佇列 以此類推 算出當前資料的 TurnArround Waiting 時間 且記錄當前的(duration) 直到跑完 mission2 陣列裡的所有資料那在每次process要做之前都會同步他的甘特圖記錄目前是誰執行那執行幾次在資料尚未抵達時會用 “-“ 來記錄 , 最後寫檔結束

============================================

SRTF :

SRTF是由剩餘CPUBurst最小的Process先排序若剩餘的CPUBurst相,則依Arrival Time小的先處理在相同才是用process ID 排序

那我的作法是我一樣先依照抵達時間 然後id 這樣來排序 因為如果是按照剩餘cpu 來排序有可能最小的cpu 很後面才抵達 排序使用到 bubblesort() 排序完後將DataList 存放在mission3的 動態陣列中 用此陣列排程 我的作法是先找到第一筆資料 是最先抵達且 cpu時間最小的那個process先做 然後每做一次就去判斷他的cpu burst time 是否 > 0 是的話就 更新當前的時間為 做完時間 將cpu time 扣一次 目前時間加一時間,如果 == 0, 一樣更新當前的時間 cpu剩餘時間 = 0 並且算出當前process 的往返時間TurnArround 以及等待時間Waiting, completed = true 代表這process已經完成了 做完之後去判斷是否有 其他的process 抵達要執行cpu 有的話我的做法是用一個 動態陣列用來存放等待要執行的process 然後當前還沒做完的process也放進來 如果存放陣列的資料量 > 1 就排序一次找到在陣列中哪一個Process 的CPU執行時間最少將之取出並且在mission3陣列中找到他並將之從存放陣列移除 , 只有一筆資料就省略步驟

for( i = 0 ; i < mission3.size(); i++) {

if( SRTFqueue[0].id== mission3[i].id ){

mission3[i].inq = true ; / / 判斷是否進入過

SRTFqueue.erase(SRTFqueue.begin() +0) ;

break ;

} // if

} // for

那有一點要注意!!就是如果 存放陣列已經空了 且下以筆資料沒有進入過存放陣列裡頭 ,代表前一個已經做完但下一個process 還未抵達if( duration < mission3[i].ArrivalTime )那為了讓程式正常運行 ,我們會強制讓下一筆資料抵達 duration = mission3[i].ArrivalTime ; 並且把下一筆資料push 進存放陣列以此類推 如果process 已完成 completed = true 算出當前資料的 TurnArround Waiting 時間 且記錄當前的(duration) 直到跑完 mission3 陣列裡的所有資料 那在每次process要做之前都會同步他的甘特圖記錄目前是誰執行那執行幾次在資料尚未抵達時會用 “-“ 來記錄 , 最後寫檔結束

========================================================

PPRR :

PPRR 是根據Priority大小依序處理 , Priority的Process代表優先處理 ; 若Priority相的Process不只一個 , 採用RR原則進行排程輪流占用cpu 當發生Time out 或者 被奪取時, 從佇列尾端开始依Priority大小排序 如果有新的process 已經抵達要讓他們先做

排序使用到 bubbleSortpprr () 排序依照 Priority等級越小越做, 但一開始我還是要先找到最先抵達的process 且等級最小的 先做

排序做完後將DataList 存放mission4的 動態陣列中 用此陣列排程, 我的作法是 先將排序後的第一筆process 放到用來存放等待執行的process PPRRqueue 存放陣列裏頭 , 然後去看存放陣列裏頭的第一個 找到他在mission4 陣列中是哪一個位置 ,紀錄位置後將他從 存放陣列中去除 , 判斷他的CPU時間是否 > 0 是的話 在進階的去判斷他是否有相同的優先等級( 因為題目有說道如果相同的優先等級要做time sharing 輪流執行 ) 有的話我會將相同優先等級的process另外拿出來用另一個陣列存放 sameprio

Ity 方式一樣找到他在mission4 陣列中是哪一個位置 那這邊要判斷他是否之前執行過 例如 : process 2 跟 3 優先等級一樣 但 2比較早抵達 早抵達的process 只能做到他剩餘的slice結束 在輪流的跟3 執行cpu 並且更新當前的時間 以及他們剩餘的cpu時間

執行完如果還要繼續做就再放回存放陣列裏頭 如果 CPU時間 < = slice 代表當前process即將做完那我一樣 更新當前的時間 將他們cpu時間歸零, 紀錄process 已完成 completed = true 算出當前資料的 TurnArround Waiting 時間 且記錄當前的(duration)時間 那如果相同的優先等級要執行且沒有一個是之前先執行過 那我們就輪流地讓相同的process 占用cpu 但當在輪流時有別的process 近來且等級還比輪流執行的process 們 低就要先暫停 輪流執行讓 比較低的優先等級porcess 先執行 執行完一樣記錄他們的目前時間 往返時間以及等待時間並標示已完成 , 再回到他們剛剛中斷的地方繼續執行 回到剛剛那如果沒有相同的優先等級的話就沒執行一次更新他們的目前時間 以及 CPU Burst time 並且看在更新時間後有無新的process 近來 有的話再比較 找出較小的優先等級先給它執行 執行完一樣記錄他們的目前時間 往返時間以及等待時間並標示已完成, 那有一點要注意!!就是如果 佇列已經空了 且下一筆process沒有進入過佇列裡頭 ,代表前一個已經執行完但下一個process 還未抵達if( duration < mission4[i].ArrivalTime )那為了讓程式正常運行 ,我們會強制讓下一筆資料抵達 duration = mission4[i].ArrivalTime ; 並且把下一筆資料push 進佇列 以此類推 算出當前資料的 TurnArround Waiting 時間 且記錄當前的(duration) 直到跑完 mission4 陣列裡的所有資料那在每次process要做之前都會同步他的甘特圖記錄目前是誰執行那執行幾次在資料尚未抵達時會用 “-“ 來記錄 , 最後寫檔結束

HRRN :

HRRN 是根據反應時間比率愈高的Process優先處理 ; 若Ratio相同的Process不只一個 , 則依Arrival Time小的先處理 ; 若Ratio相同且Arrival Time同 , 依Process ID由小至大依序處理。排序使用到 bubblesort() 排序完後將DataList 存放mission5的 動態陣列中 用此陣列排程 我的作法是 , 先將排序後的第一筆資料先執行完 completed 設定為完成( true ) , 算出第一個process 的 TurnArround and Waiting time 的時間 再來是更新當前的時間(duration) , 做完後去找到當第一個process 在執行時 有哪些process 抵達需要執行 並將他們存放在HRRNqueue 陣列中 因為第一個process已經做完所以我們可以算出其他抵達的process 與 第一個process 的反應時間比率 那算出來最高的反應時間 的process 優先處理 我這邊是以 double 的型別去紀錄反應時間 , 將process 從存放陣列取出並且移除(erase) 算出當前 process 的 TurnArround and Waiting time 的時間 再來是更新當前的時間(duration) completed = true設定已經完成這筆process , 那有一點要注意!!就是如果 存放陣列已經空了 且下以筆資料沒有進入過存放陣列裡頭 ,代表前一個已經做完但下一個process 還未抵達if( duration < mission5[i].ArrivalTime )那為了讓程式正常運行 ,我們會強制讓下一筆資料抵達強制執行duration = mission5[i].ArrivalTime ; 並且把下一筆資料push 進存放陣列以此類推 那在每次process要做之前都會同步他的甘特圖記錄目前是誰執行那執行幾次在資料尚未抵達時會用 “-“ 來記錄 , 最後寫檔結束

未完成的功能 :

助教所給的範例input檔結果皆為正確

心得:

在我寫的這五個排程法中我認為最淺顯易懂的應該就是先進先出排程法，因為他只需要按照抵達時間去排程接著依序執行即可完成而且他又是不可奪取的排程法就不會有所謂的那一些timeout問題或者是被插隊，那再來是最高反應優先比率排程法 ，這個排程法其實只要懂得反應時間比率去計算出誰的反應比率最高並且持續更新等待時間即刻順利完成 ，我覺得最難的部分是任何有包含時間片段的排程法 ，尤其是 可奪取的知更鳥排程法 因為他要考慮的問題點很多像是Sandy打的那個如果是別的有相同優先等級的process要另外處理還有在輪流做sharing的時候 在考慮是否有其他的比較優先等級低的process進來