RM 和 EDF 排程器
專題報告

尤呈譽（A1055533)

這次專題主要的目標為使用程式來模擬 RM 和 EDF 排程的機制，專題也規定不能使用 array 來實作，必須要使用 linked list 來存放系統中的任務資訊和等待被執行工作的 ready queue，為了達成這個目標我也特地去複習了之前大二資料課程所教的 linked list 的程式碼與實作，最後我是使用 C++ 中內建的 list 以及 vector 的函式來實作 linked list，而整個程式也參考了老師給我們的 word 檔中 struct 和 pseudocode 的提示，在接下來的報告中，我按照老師給我們的參考資料中將我的程式碼分為 7 個步驟來介紹我的程式架構，我會以程式碼內容順序從上到下的方式介紹。

程式的最開始為相關變數的宣告，我參考了老師範例中 struct 的架構，將 struct 分為兩個部分，一個部分為儲存 task 使用而另一個部分則為儲存 job，在 task 和 job 分別有變數負責儲存該 task 或 job 的執行時間、週期、截止時間等等重要資訊，而 task 和 job 兩個共通可以相互連結的變數為 TID，這個變數的主要為儲存該工作的編號，讓排程時可以追蹤該工作的時間資訊。其他變數也有總共 task 個數的 Total_Task_Number 和儲存最大 phase time 的 MaxPH 等等。為了將主程式縮小，我將程式中的功能分為其他小的 function，像是 gcd 和 lcm 就是為了計算週期的最小公倍數而加入的功能，而 showlist 則是顯示 list 資料型態內容的函示。

接下來程式即為 main 的部分了，我按照參考資料中第一個項目資料的讀入開始實作，此部分主要是在程式碼 readData 的函式中，我使用 fstream 將資料按照指定的目錄讀入後，使用 getline 將資料進行處理，因為輸入的資料格式是以逗號（, ）分開，必須將資料進行切割才可以分別抓取到正確的資料，抓出資料後，我直接將資料依照輸入的順序存入 task 的 structure 中。

第二個步驟即為 schedulability test 和計算週期最小公倍數的部分，使用前面定義的 lcm 和 gcd 函式以及每項 task 中的週期以遞迴方式算出週期的最小公倍數，而 schedulability test 的部分使用課本中的公式判斷，分為 EDF 和 RM 兩種計算方式，計算結果若該種排程方式可能不能排程的話，即會立即將測試結果顯示出來。

第三步驟和第四步驟是所有部分中較為簡單的地方，目的是初始化序列和

clock，初始化序列的部分使用 list 函式中 clear() 的 function 將序列清空，而 clock 初始化即是將 clock 變數重新歸零。

接下來的步驟五到九即進入整個程式中最重要的部分了，這個部分負責了 RM 或 EDF 的排班判斷和 clock 的判斷，步驟五根據週期的最大公倍數和所有任務的最大 phase time 判斷是否要結束 clock，而步驟六則是判斷序列中的每一個工作是否可以在他的絕對截止時間內完成，這個我的實作方式為先判斷序列中是否有工作，若有工作才會進行絕對截止時間的判斷，我以迭代器從序列的頭抓取所有的工作一路到序列的尾巴，如果在該工作的絕對截止時間 － 目前的 clock － 剩下的要執行的時間小於零的話，即判斷該工作無法完成，將該工作的 PID 紀錄到另一個要移除的序列中，也在這邊將紀錄 miss 工作的變數 Miss_Deadline_Job_Number 加一，使用這種方式是因為如果在判斷該工作需要移除，而在迭代器遞迴中的話，即會讓原本的迭代器走到超過範圍的數值，造成程式錯誤，找到需要移除的工作 PID 後，最後再將工作從待執行的序列中移除即完成檢查所有工作是否可以排的部分。

下一個步驟七為判斷目前的 clock 是否為該工作抵達的時間，我以遞迴的方式檢查所有的工作，每個工作以目前的 clock － 該工作的 phase 和該工作的 period 取餘數做檢查，若結果為零的話即表示當前的 clock 是該工作的抵達時間，將該工作加入等待被執行的序列中，我也在這個步驟中將該 job 的 release time, remain execution time, absolute deadline 等資訊存入 job 的 struct 中。

步驟八則是 RM 和 EDF 排班函式擁有不同判斷的地方，RM 排班函式的部分，因爲 RM 排班在尋找優先權最高的工作的方式為檢視所有工作中擁有最小 period 值的工作，我將變數 shortest_period 初始化為 INT_MAX，int 變數的最大值，這樣在之後的比較最小值時會有比較保險的結果，我也宣告 RM_PID 變數來儲存在執行 RM 排班後要決定要執行的工作，一開始以迭代器方式將整個等待執行的序列完整走過，若找到比上一個更低 period 的工作即將該 TID 紀錄下來，這邊有再加上一個判斷為如果兩個工作擁有相同 period 時，則 RM_TID 紀錄的為 TID 較小的工作，以符合數字較小的工作優先權較高的原則，判斷完後即為工作執行的部分，若 RM_TID 為 -1 的話，即表示該 clock 沒有要執行的工作，但若 RM_TID 是 -1 以外的值，就將該工作剩下執行的時間減一，若該工作的剩餘執行時間減完後為零，則將該工作從待執行的序列中移除，最後再將執行過的工作儲存到 RM 已執行序列中。

步驟八在 EDF 排班函式的部分，變數宣告的的部分 RM 排班大同小異，因為 EDF 是以擁有最小 deadline 的工作的優先權最高，分別宣告了 earliest_deadline 和 EDF_PID 變數儲存，而 earliest_deadline 的部分也有使用 INT_MAX 的方式處理，一開始也是以迭代器方式將整個等待執行的序列完整走過，若找到比上一個擁有更早 deadline 的工作即將該 TID 紀錄下來，並有加入相同工作擁有相同 deadline 時執行較小 TID 的工作，工作執行的部分和 RM 相同，若 EDF_PID 為 -1 的話，即表示該 clock 沒有要執行的工作，但若 EDF_PID 是 -1 以外的值，就將該工作剩下執行的時間減一，若該工作的剩餘執行時間減完後為零，則將該工作從待執行的序列中移除，最後再將執行過的工作儲存到 EDF 已執行的序列中。

完成了這個 RM 和 EDF 排程器之後，我覺得我對於 RM 和 EDF 排班機制有了許多更深的了解，原本只是透過老師上課的講義看過並且練一些題目而已，但是為了完成這個專題，我必須將需多排班中可能遇到可能性都考慮進去，才可以讓程式在執行時擁有正確的結果，雖然老師有提供架構和 pseudocode 讓我們參考，但我還是覺得這個程式實作起來具有一些挑戰性，相信在寫完這個程式後對於未來在學習即時計算機系統其他的排班時也很有幫助。