程式簡介

10727248 資訊三乙 鄭珮慈

一、開發平台

Intel® Core™ I5-8250U 處理器、RAM 8GB、64位元作業系統。

二、開發環境

Windows 10、CLion (IDE)。

三、程式設計說明

A、功能

1. 於程式開始執行時便提供使用者介面選項，可讓使用者決定要退出(QUIT)結束程式或是開始(START)執行程式。
2. 能夠依照使用者所輸入之檔案名稱讀檔，並依照檔案內所指定之Page Frame個數執行各種指定的Page Replacement方法，最後輸出每次Page Reference時，所有Page Frame所記錄的內容，並計算出每種方法發生的Page Fault次數以及Page Replace次數。
3. 於排程結束時能夠將結果依標準格式寫檔輸出，且能夠重複執行(直到使用者於使用者介面選擇QUIT(退出)選項為止)。

→ 各個Page Replacement方法運作說明：

(1). First In First Out (FIFO)

先進先出頁置換法可簡稱為FIFO頁置換法。當某個Page被載入至Page Frame時，就會以時間戳記(Time Stamp)記錄其被載入的時間。當Page Frame被放滿無法再放入其他Page時，就會將時間標記最小的頁，也就是在Memory內最久的頁置換出去(先進Memory的Page先出去)。

(2). Least Recently Used (LRU)

最近罕用頁置換法可簡稱為LRU頁置換法。是以過去記錄來當作Page未來行為的參考，會將過去最久不被使用到的Page先置換。和FIFO頁置換法最大的不同在於，FIFO在Page被載入時給予其時間戳記後，就不會再去改變時間戳記，但LRU頁置換法除了在載入時給予時間戳記外，每當此Page被參考到時，就會重新給予一個新的時間戳記。

(3). Least Frequently Used (LFU) + FIFO

此方法簡單來說，就是在FIFO頁置換法的基底下加入LFU的功能。最不常使用頁置換法又可簡稱為LFU頁置換法，它讓每個Page Frame都有一個對應的計數器(Counter)，起始值為0，當對應的Page Frame被參考或修改時，Counter值便會加1。當遇到Page需要置換時，系統會選用Counter值最小的Page Frame做頁置換。

(4). Most Frequently Used (MFU) + FIFO

此方法簡單來說，就是在FIFO頁置換法的基底下加入MFU的功能。最常使用頁置換法又可簡稱為MFU頁置換法，它讓每個Page Frame都有一個對應的計數器(Counter)，起始值為0，當對應的Page Frame被參考或修改時，Counter值便會加1。當遇到Page需要置換時，系統會選用Counter值最大的Page Frame做頁置換。

(5). LFU + LRU

此方法簡單來說，就是在LRU頁置換法的基底下加入LFU的功能。最不常使用頁置換法又可簡稱為LFU頁置換法，它讓每個Page Frame都有一個對應的計數器(Counter)，起始值為0，當對應的Page Frame被參考或修改時，Counter值便會加1。當遇到Page需要置換時，系統會選用Counter值最小的Page Frame做頁置換。

(6). MFU + LRU

此方法簡單來說，就是在LRU頁置換法的基底下加入MFU的功能。最常使用頁置換法又可簡稱為MFU頁置換法，它讓每個Page Frame都有一個對應的計數器(Counter)，起始值為0，當對應的Page Frame被參考或修改時，Counter值便會加1。當遇到Page需要置換時，系統會選用Counter值最大的Page Frame做頁置換。

B、流程

程式一開始，便會秀出使用者介面，使用者可選擇結束(QUIT)或開始執行(START)，若選擇START，就會請使用者輸入想模擬Page Replacement方法的檔案，交由Method::ReadFile()讀檔，將指定的Page Frame大小與Page Reference string(Page被參考到的先後次序)儲存。

接著便開始依資料內所指定的Page Frame大小對Page Reference string進行頁置換的模擬，各方法模擬功能如上「A、功能→各個Page Replacement方法運作說明」所述。

模擬頁置換完畢後便會用Method:: WriteFile()產出一個輸出檔，裡面存放的是經過頁置換後的資料及相關輸出資訊，會依照FIFO、LRU、Least Frequently Used Page Replacement(LFU+FIFO)、Most Frequently Used Page Replacement(MFU+FIFO)、Least Frequently Used

LRU Page Replacement(LFU+LRU)、Most Frequently Used LRU Page Replacement(MFU+LRU)的順序寫檔。

C、使用的Data Structure

我分別定義了兩種struct，分別叫pageDetail (用於存放最終要輸出的結果資料，包含Page Reference、Page被參考時Page Frame內的情形與是否發生Page Fault)與pageCurrent (用於記錄目前在Page Frame的情形，且幫每個Page Frame都設置了一個Counter初始值為0 =>用於LFU與MFU)。詳細如下圖所示：

```
10  struct pageDetail {
11    char ref = ' ' ;
12    vector <char> frame ;
13    bool fault = false ;
14  };
15
16  struct pageCurrent {
17    char ref = ' ' ;
18    int count = 0 ;
19  };
```

另外，於Method這個class內，我宣告了三個array及四個vector，以下將針對此做說明。

AllFaults與AllReplaces皆為int型別的一維靜態array，大小為6，主要用來分別儲存每個頁置換模擬方法後，所計算出的Page Faults數量與Page Replace次數，方便於最後寫檔時寫入模擬結果相關資訊。

還有一個string型別的一維靜態array叫做methodName，大小同樣為6，主要負責寫檔時寫入對應的頁置換方法名稱，會搭配Method::WriteFile()內的for迴圈運作。

接著是vector的部分。

result為型別pageDetail的一維vector，用來記錄一個模擬頁置換方法的結果資訊，包含所有Page Reference被參考時Page Frame內的情形與當時是否發生Page Fault。當一個模擬頁置換的方法執行完畢後，便會將result存入型別為vector<pageDetail>的AllResult此二維vector中。

AllResult會負責收集所有模擬頁置換方法所產出的結果(vector <pageDetail> result)，且會依照FIFO、LRU、LFU+FIFO、MFU+FIFO、LFU+LRU、MFU+LRU的寫檔順序依序存入，方便於最後寫檔時寫入各個方法之模擬結果。

currentFrame是一個型別為char的一維vector，負責記錄當前Page Frame的狀態(有哪些Page在裡面，且其順序為何)，當一個Page被參考完要存入當前Page Frame資訊時便用此vector內的資訊儲存至結果中。

Counter是一個一維vector型別為pageCurrent，同樣也是用來記錄當前Page Frame的狀態，但多儲存了每個Page Frame內Counter的值，

當模擬方法須搭配LFU與MFU時，便能夠拿來使用。

詳細如下圖所示：

```
class Method {
private:
  int page_frames ;       // 1-9
  string page_reference ; // 0-9
  int page_faults ;       // 記錄頁錯誤次數
  int page_replaces ;     // 記錄頁置換次數
  int AllFaults[6] ;
  int AllReplaces[6] ;
  vector <pageDetail> result ; // 記錄各reference資訊
  vector <vector <pageDetail>> AllResult ; // 記錄各方法的答案
  vector <char> currentFrame ; // 紀錄目前的frame資訊
  vector <pageCurrent> counter ;
  pageDetail tempBlock ;
  pageCurrent tempCounter ;
  string methodName[6] = { "FIFO", "LRU", "Least Frequently Used Page Replacement",
                           "Most Frequently Used Page Replacement ",
                           "Least Frequently Used LRU Page Replacement",
                           "Most Frequently Used LRU Page Replacement " };
```

四、不同方法之間的比較

A、Page Fault與Page Replace次數

(Page Fault／Page Replace)

【輸入檔案Input1.txt】

FIFO：(9／6)

LRU：(10／7)

LFU+FIFO：(10／7)

MFU+FIFO：(9／6)

LFU+LRU：(10／7)

MFU+LRU：(9／6)

【輸入檔案Input2.txt】

FIFO：(15／12)

LRU：(12／9)

LFU+FIFO：(13／10)

MFU+FIFO：(15／12)

LFU+LRU：(11／8)

MFU+LRU：(12／9)

B、結果與討論

單就FIFO和LRU來比較，在Input1時LRU發生Page Fault與Page Replace的次數皆比FIFO多1次，但在Input2中LRU發生Page Fault與Page Replace的次數皆比FIFO少3次。因此我認為LRU對於資料量較多的情況相較於FIFO更能避免Page Fault與Page Replace，因為它是以過去紀錄做為參考並在每次Page被參考時就會更新Page的時間戳記，而不是單看第一次的時間戳記就做判斷。

至於搭配LFU和MFU的效果，LFU有可能因為取代的Page Frame是新進來不久的Page，而造成取代錯誤的情形。MFU則是會把待很久的Page選為置換對象，但若此Page為很重要的Page，就會造成需再進行置換的問題。因此我認為搭配這兩種置換方法所得出的結果沒有絕對的好壞，模擬結果很大部分取決於輸入資料的型態。