Document

資訊三甲。10727138。游子諭

1. 開發環境：

Dev C++

2. 實作功能、資料結構、運行方式

這次主要是實現 **FIFO**、**LRU**、**LFU+FIFO**、**MFU+FIFO**、**LFU+LRU**、**MFU+LRU** 在 paging 中會出現的結果。以下我會介紹各個 paging 的功能。

讀檔

先讀入 framework 的大小，並將各個 page 切 token 分別放入 vector<int>中（稱為 **schedule**）。

FIFO

在一個以 framework 大小中的 vector<int>中（稱為 **frame**），因為 input 皆為非負整數，因此我將 **frame** 中的內容皆設為 -1，從 **schedule** 抓新的 page 進來時，先檢查 **frame** 裡面是否有相同的 page。

如果有相同的 page，就甚麼事都不用做。如果沒有相同，將 vector 的內容全部往後移一個，最後一個移除，第一個留空，並將新的 page 放進 vector 的第一個位置，同時為了要記錄 page fault，我會創一個 vector<char>（稱為 **fault**）去紀錄當下是否有發生 page fault，我也會創一個 int（稱為 **replace**）去紀錄 page replace 發生的次數。並且利用 vector<vector <int>>（稱為 **record**）來記錄當下的 **frame**。

LRU

和 FIFO 作對 **frame** 相同的前置動作，差別在於如果從 **schedule** 抓新的 page 時發現 **frame** 裡面有相同的 page，會把該 page 放到 **frame** 的最前端，其他的依序往後移。沒有相同的 page 的處理方法和 FIFO 相同。

LFU+FIFO

LFU 會在 **frame** 的各個欄位設一個 counter，當遇到相同的 page 在 **frame** 忠實，該 **frame** 欄位的 counter 會加一，目的在於當今天必須要移除一個 page 時，LFU 會將 counter 最小的那個欄位優先移除，剩下的 page 依序向後，新的 page 放在 LFU 的最前面，而 **fault** 一樣會去紀錄。而我利用的方式創造一個 vector<int>，一樣以 framework 的大小去創造，裡面專門放各個欄位的 counter

（稱為 **counter**），其實跟 FIFO 要做的事情差不多，只是在移動 **frame** 的同時，也要移動 **counter**，並且每次要移除 page 時，要從 **counter** 中找最小值。

MFU+FIFO

邏輯和實作上跟 LFU+FIFO 相同，差別在於當必須要移除 page 時，選擇 counter 大的來優先移除，其他步驟沒有改變。

LFU+LRU

套用和 LFU+FIFO 相同的模式，不同的地方在於當有相同 page 在 **frame** 中時，必須將該 page 移動到 **frame** 的最前面，同時，**counter** 也要把該欄位的 count 移動到最前面。

MFU+LRU

和 LFU+LRU 相同，差別一樣是在當必須要移除 page 時，選擇 counter 大的來優先移除，其他步驟沒有改變。

寫檔

在每個方法做完之後，我會利用 vector<**record**>、vector<**fault**>、vector<**replace**>去紀錄。Page reference 的部分用 **schedule** 即可，page fault 的次數去計算各個方法的 **fault** 也可以完成，最後依序將各個結果 output 出來

3. 不同方法之間的比較

FIFO VS LRU

因為 LRU 會把相同 page 移動到 frame 的最前面，因此在 “常用的 page 放前面”這個邏輯下，再加上正常使用下，相同的 page reference 也確實會有高頻率出現的狀況發生，這樣可以有效的讓 page fault 下降，也因為 page fault 下降的緣故，page replace 也可以因此下降一點。

LFU VS MFU

兩者比較的重點在於選擇 replace 時，一個選擇 counter 大，另一個選擇 counter 小，LFU 是選擇 counter 小，理由是讓最不常使用的 page 優先剃除，而 MFU 是讓最常使用的 page 優先剃除，理由是在 main memory 待很久了，可能已經處理完了。這兩種邏輯感覺上都說得通，主要還是要看 input 的資料來決定哪個方法比較好，假設兩種方法都能在自己預想的狀況下實行，兩者都能減少 page fault 和 page replace 的次數。