作業系統　HW3 文件

資訊四甲 10827111 袁本誠

一、開發環境

作業系統: windows

程式語言: C++

開發軟體: dev C++

二、實作方法與流程

主程式流程:

1. 使用者輸入檔名，取得欲行方法、字串、頁框架
2. 根據欲行方法，執行相應的函式
3. 寫檔
4. 回到步驟 1

FIFO:

1. 檢查目前頁框架是否已滿
2. 如果已滿，檢查是否重複
3. 有重複則不將字串裡的字元放入 vector；沒有重複則將 vector 最後一個移除，新的字元插入 vector 最前方。
4. 重複步驟 1、2、3，直到全部字元皆處理完

LRU:

1. 檢查目前頁框架是否已滿
2. 如果已滿，檢查是否重複
3. 有重複則不將字串裡的字元放入 vector，但要把其插入至 vector 最前方；沒有重複則將 vector 最後一個移除，新的字元插入 vector 最前方
4. 重複步驟 1、2、3，直到全部字元皆處理完

LFU:

1. 檢查目前頁框架是否已滿
2. 如果已滿，檢查是否重複
3. 有重複則將該框架的 count 增加 1；沒有重複則，將 vector 中 count 最小的移除，新的字元插入 vector 最前方。
4. 重複步驟 1、2、3，直到全部字元皆處理完

MFU:

1. 檢查目前頁框架是否已滿

2. 如果已滿，檢查是否重複
3. 有重複則將該框架的 count 增加 1；沒有重複則，將 vector 中 count 最大的移除，新的字元插入 vector 最前方。
4. 重複步驟 1、2、3，直到全部字元皆處理完

LFU+LRU:

1. 檢查目前頁框架是否已滿
2. 如果已滿，檢查是否重複
3. 有重複則將該框架的 count 增加 1，並把其插入至 vector 最前方；沒有重複則，將 vector 中 count 最小的移除，新的字元插入 vector 最前方。
4. 重複步驟 1、2、3，直到全部字元皆處理完

三、 不同方法的比較

不同的置換策略， Page Replaces、Page Fault 次數也不同，以 input2 的情況來說

FIFO :	Page Replaces = 12	Page Fault = 15
LRU :	Page Replaces = 9	Page Fault = 12
LFU :	Page Replaces = 10	Page Fault = 13
MFU :	Page Replaces = 12	Page Fault = 15
LFU+LRU :	Page Replaces = 8	Page Fault = 11

四、 結果與討論

畢雷迪反例是增加頁框架，反而造成更多頁錯誤與頁置換，以不同 Page Frame 的 input1 執行 FIFO 策略來說:

1. Page Frame = 2 ： Page Fault = 12
1. Page Frame = 3 ： Page Fault = 9
2. Page Frame = 4 ： Page Fault = 10
3. Page Frame = 5 ： Page Fault = 5

從 Page Frame 2 到 3 確實降低了 Page Fault；但從 Page Frame 3 到 4，Page Fault 卻從 9 增加至 10，這就是畢雷迪反例。