**CS305作業系統概論 Prog. #4 Page Replacement 說明報告.docx**

**作者　：1043335 賴詩雨**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修改日期** | **說明** | **修改者** |
| **1.0.0.0** | **2017-06-12** | **初稿** | **賴詩雨** |
| **1.0.0.1** | **2017-06-13** | **修改** | **賴詩雨** |
|  |  |  |  |
|  |  |  |  |

**目錄**

[第1章 程式完成部分 2](#_Toc484119947)

[第2章 設計理念 2](#_Toc484119948)

[2.1 Optimal Algorithm 2](#_Toc484119949)

[2.2 FIFO Algorithm 3](#_Toc484119950)

[2.3 LRU Algorithm 4](#_Toc484119951)

[2.4 LfU Algorithm 5](#_Toc484119951)

[2.5 Clock Algorithm 6](#_Toc484119951)

[第3章 程式如何編譯 7](#_Toc484119954)

[第4章 如何操作 7](#_Toc484119955)

1. 程式完成部分
2. 4種演算法 ( Optimal、FIFO、LRU、LFU )
3. 加分項目 : Clock algorithm
4. 設計理念
   1. Optimal Algorithm

* 變數介紹

|  |  |
| --- | --- |
| vector **<**int**>** frame**;**  vector **<**int**>** first**(**Fsize**,** **-**1**);**  int fault **=** Fsize**;**  int point **=** 0**,** Fcount **=** 0**;** | // Frame 的數量  // 計算frame中進來數字的陣列編號(編號小代表  越早進來)  // page fault 的數量  // 目前處理到數字的位置, 目前frame要換的位置 |

* **實作方法**

一開始等frame 全部先放滿，再來判斷接下來要使用的數字是否已存在frame中。

1. 若frame中無，則利用 for迴圈去計算未來看的到的數字中有幾個存在 ( count )
2. Frame 中的數字都存在 ( count == Fsize )，則最遙遠的那個數字極為要替換掉的
3. Frame 中有數字不存在，則替換掉不存在當中最早進入的 ( FIFO )
4. 若frame中已存在此數字，則直接印出即可

* **簡單示意圖**

**A**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2** | **0**  **B** | **3** | **0** | **4** | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 2 | 2 | 2 | 2 | 2 | 2 |
| 0 | 0 | 0 | 0 | 0 | 4 | 4 |
| 1 | 1 | 1 | 3 | 3 | 3 | 3 |

**7xx，70x，701**

**共三次page fault**

**Page fault : 6**

* **流程簡介 ( 以上述例子為例 )**

以上述例子為例

1. 藍框A要判斷2是否再frame中，發現沒有，因此進入for迴圈，計算count
2. 得到count = 1 ( 未來只能看到 0 一個數字)
3. 7 和 1兩數當中，7較早進入，因此替換掉7
4. 可得 2 0 1
5. 藍框B要判斷0是否再frame中，發現有，因此直接印出
6. 其餘的依照上述方法完成
   1. FIFO Algorithm

* 變數介紹

|  |  |
| --- | --- |
| vector **<**int**>** frame**;**  int fault **=** Fsize**;**  int point **=** 0**,** Fcount **=** 0**;**  int fifo **=** 0**;** | // Frame 的數量  // page fault 的數量  // 目前處理到數字的位置, 目前frame要換的位置  // 目前要替換掉的frame位置 |

* **實作方法**

一開始等frame 全部先放滿，再來判斷接下來要使用的數字是否已存在frame中。

1. 若frame中無，將先進來的替換掉 ( fifo 指到的位置 )
2. 若frame中已存在此數字，則直接印出即可

* **簡單示意圖**

**A**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2** | **0**  **B** | **3**  **C** | **0** | **4** | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 2 | 2 | 2 | 2 | 4 | 4 |
| 0 | 0 | 0 | 3 | 3 | 3 | 2 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

**7xx，70x，701**

**共三次page fault**

**Page fault : 8**

* **流程簡介 ( 以上述例子為例 )**

1. 藍框A要判斷2是否再frame中，發現沒有，因此將最早進入的數字替換掉 ( 7 )

NOTE: int fifo 一開始在7的位置，被替換掉後會往下一格，到底再回第一格位置

1. Fifo 從 7 的位置移到 0 的位置
2. 藍框B要判斷0是否再frame中，發現有，因此直接印出，且fifo不變
3. 藍框C要判斷3是否再frame中，發現沒有，因此將3替換到fifo在的位置 ( 0 )
4. 其餘的依照上述方法完成
   1. LRU Algorithm

* 變數介紹

|  |  |
| --- | --- |
| vector **<**int**>** frame**;**  int fault **=** Fsize**;**  int point **=** 0**,** Fcount **=** 0**;**  vector **<**bool**>** check**(**Fsize**,** 0**);** | // Frame 的數量  // page fault 的數量  // 目前處理到數字的位置, 目前frame要換的位置  // 往回找，先找到的標示為1，最後變1便替換掉 |

* **實作方法**

一開始等frame 全部先放滿，再來判斷接下來要使用的數字是否已存在frame中。

1. 若frame中無，則利用 for迴圈去計算過去的數字誰最晚進就先標成1 ( check )，最早進的數字會最晚被標成1，因此就是要被替換的
2. 若frame中已存在此數字，則直接印出即可

* **簡單示意圖**

**A**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2** | **0**  **B** | **3** | **0** | **4** | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 2 | 2 | 2 | 2 | 4 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 3 | 3 | 3 | 2 |

**7xx，70x，701**

**共三次page fault**

**Page fault : 7**

* **流程簡介 ( 以上述例子為例 )**

1. 藍框A要判斷2是否再frame中，發現沒有，因此進入for迴圈往前找，先將1的check標成1，再將0的check標成1，最後才將7的check標成1
2. 因此7為要替換掉的
3. 可得 2 0 1
4. 藍框B要判斷0是否再frame中，發現有，因此直接印出
5. 其餘的依照上述方法完成
   1. LFU Algorithm

* 變數介紹

|  |  |
| --- | --- |
| vector **<**int**>** frame**;**  vector **<**int**>** first**(**Fsize**,** **-**1**);**  vector **<**int**>** count**(**Fsize**,** 1**);**  int fault **=** Fsize**;**  int point **=** 0**,** Fcount **=** 0**, Min;** | // Frame 的數量  // 數字進來時的陣列編號，越小代表越早  // frame中數字目前被使用的次數  // page fault 的數量  // 處理到數字,frame要換的位置,count中最小數 |

* **實作方法**

一開始等frame 全部先放滿，再來判斷接下來要使用的數字是否已存在frame中。

1. 若frame中無，則利用 for迴圈去找出count最少的，再去比較誰較早進來，較早且使用較少的被替換掉
2. 若frame中已存在此數字，將frame中此數字的count加一，再直接印出即可

* **簡單示意圖**

**A**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2** | **0**  **B** | **3**  **C** | **0** | **4** | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 2 | 2 | 2 | 2 | 4 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 3 | 3 | 3 | 2 |

**7xx，70x，701**

**共三次page fault**

**Page fault : 7**

* **流程簡介 ( 以上述例子為例 )**

1. 藍框A要判斷2是否再frame中，發現沒有，去找出count最少的，發現三個數字的count都一樣，因此要再找最早進來的，發現是7，因此7為要替換掉的，可得 2 0 1
2. 藍框B要判斷0是否再frame中，發現有，因此將0的count加一再直接印出
3. 藍框C要判斷3是否再frame中，發現沒有，比對count，發現2 1的次數一樣，因此從兩數當中找最早的，發現是1，替換掉1後為2 0 3
4. 其餘的依照上述方法完成
   1. Clock Algorithm

* 變數介紹

|  |  |
| --- | --- |
| vector **<**int**>** frame**;**  vector **<**int**>** Rbit**(**Fsize**,** 1**);**  int fault **=** Fsize**;**  int point **=** 0**,** Fcount **=** 0**;**  int fifo **=** 0**;** | // Frame 的數量  // 記錄第二次機會是否還在  // page fault 的數量  // 目前處理到數字的位置, 目前frame要換的位置  // 目前要替換掉的frame位置 |

* **實作方法**

一開始等frame 全部先放滿，再來判斷接下來要使用的數字是否已存在frame中。

1. 若frame中無，則利用 while迴圈去檢查第二次機會是否還在，若還有機會就把機會扣掉，若沒有機會了就是要替換掉的，利用fifo代表目前要被檢查的frame
2. 若frame中已存在此數字，將frame中此數字的Rbit設成一，再直接印出即可

* **簡單示意圖**

**A**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2** | **0**  **B** | **3**  **C** | **0** | **4** | **2** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 4 | 1 | 4 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 3 | 1 | 3 | 1 | 3 | 0 | 3 | 0 |

灰底為Rbit

**7xx，70x，701**

**共三次page fault**

**Page fault : 7**

* **流程簡介 ( 以上述例子為例 )**

1. 藍框A要判斷2是否再frame中，發現沒有，fifo依序檢查7 0 1發現他們都有一次機會，因次扣掉他們的機會再次回到7
2. 發現他沒有機會了，所以將7替換掉，而替換進來的2有一次機會，此時fifo到0那格
3. 藍框B要判斷0是否再frame中，發現有，因此將0的Rbit設成1，代表他現在有了一次機會
4. 藍框C要判斷3是否再frame中，發現沒有，因此從fifo所在那格 ( 0 ) 開始檢查機會，0有一次機會，所以扣掉，再往下發現1沒機會了，因此將他替換掉，變成2 0 3，且3的機會有一次
5. 其餘的依照上述方法完成
6. 程式如何編譯
7. 有附上Makefile檔提供操作

OR

1. 在命令列輸入 : g++ 1043335\_04.cpp -o 1043335\_04
2. 編譯成功後，輸入 : ./1043335\_04 replace.cfg

**NOTE : replace.cfg可改成其他檔名，可在命令列改也可以從Makefile中更改**

1. 如何操作
2. 將 Makefile 中 replace.cfg 改成其他檔名，可換不同的測資
3. make main => 編譯
4. make run => 執行
5. make start => 編譯 & 執行
6. make clean => 清除編譯、執行後產生的檔案
7. make all => 清除檔案後編譯執行
8. 或是可以直接在命令列輸入 **第3章** 提供的指令