|  |  |  |  |
| --- | --- | --- | --- |
| 名稱 | 定義 | 範例 | 分析 |
| FIFO | 最早載入的page視為victim page |  | 1. 效果不好page fault rate高 2. 有Belady anomaly(異常現象) |
| OPT(optimal) | 選擇未來長期不會使用的page當作victim page |  | 1. 是最佳的法則，page fault rate最低 2. 沒有Belady anomaly 3. 無法製作出來 |
| LRU | 最近最不常使用的page當作victim page(歷史資料做決定的OPT) |  | 1. 效能不錯，尚可接受 2. 沒有Belady anomaly 3. 製作成本高 |
| LRU近似法則 | LRU成本高才有近似法則，在page table再加入一個欄位reference bit  {0=未存取過  {1=曾被參考過 |  |  |
| LRU及MFU | 以page的累積參考次數當作挑選依據   1. 挑累積次數最小的當victim 🡪 LFU 2. 挑累積次數最多的當victim 🡪 MFU   多個page具相同值，以FIFO為準 |  | 1. 效果不好，page fault rate高 2. 有Belady anomaly(異常現象) 3. 製作成本高 |

Page replacement algo

* Page replacement algo沒有所謂最差的algo，因為每個範例不同它的結果也不同
* Belady anomaly – process分配到的frame數增加，其page fault rate不降反升
* Stack property – n個frame包含的page set必定是n+1個frame所包含page set的子集，具有此性質的法則不會有Belady anomaly。 (OPT及LRU具有此性質，其餘沒有)
* reference bit由MMU set (0 🡪1)；OS reference reset (1 🡪0)

|  |  |  |
| --- | --- | --- |
| LRU製作法 | Counter法 | 以一個counter作為logical timer 當有memory access發生則counter++，要選擇victim page時以the least reference time最小的page置換 |
| Stack法 | 最後存取的page會push到stack頂端，stack的bottom端即  LRU page(frame數 = stack大小 ) |

|  |  |  |  |
| --- | --- | --- | --- |
| 三種LRU近似法則 | 名稱 | 做法 | 範例 |
| Reference bit usage | 每一個page有一個register紀錄最近幾次的Reference bit值每隔一段時間做   1. 每個page的register值，右移一位空出最左位元 2. Copy Reference bit到最左位元 3. Reference bit值清為0   要選擇victim page挑register值最小的，若有多個相同則以FIFO為主 | 如果移位暫存器中保存的是00000000，那麼這一頁在前八段時間內都沒有被使用過；某頁如果每段時間之內最少都被使用過一次，那麼它的移位暫存器內的值就是11111111。某頁暫存器中的值若是11000100，那它就一定比值是01110111的那一頁較常被用(11000100 > 01110111) |
| Second chance | 1. 以FIFO order排出候選page 2. 檢查此page的Reference bit，若為0則為victim page 3. 若為1 🡪就給那一頁第二次機會並且繼續去選出下一個FlFO頁。   若所有page的Reference bit皆相同，退化成FIFO |  |
| Enhanced second chance | 以 < Reference bit ,Modification bit >配對值為挑victim的依據，最小的作為victim，若多個page具相同值則以FIFO為準 |  |

|  |  |  |
| --- | --- | --- |
| Page buffering機制 | OS建立一個free frame pool (內放一些free frame不配置給process)，當page replacement完之後則   1. 從free frame pool中拿出一個free frame，供lost page載入 2. 載入完之後process可resume exection (不需等victim寫回disk) 3. 等page I/O device有空寫回victim page再將此free frame加入free frame pool |  |
| OS keep Modified pages list  只要page I/O device有空，OS就將一些Modified page寫回disk，清除它們的Modification bit =0，可增加將來選victim page時的unmodified page的機率，大幅降低寫回次數，lost page直接載入可用frame，可以加快resume exection速度 |  |
| 架構在法一的基礎，但是OS會紀錄這些pool中的free frame內放的是哪個process的哪個page(最新的內容)，當page fault發生要做page replacement   1. 挑選victim page 2. 先到free frame pool找有無lost page 3. 若有則取出，process可resume exection 4. 若沒有依照法一處理 |  |

* Page buffering機制可搭配任何法則使用，降低選錯付出的成本 (早期VAX / VMS常用)