MEMORY SCHEDULING ALGORITHMS
OPERATING SYSTEMS CSE: 223

BY

Mohammed Ehab Elsaeed

16P8160

Ahmed Sameh Shahin

16P6063

PRESENTED TO

DR. GAMAL ABDEL SHAFY EBRAHIM

ENG KARIM DARWISH

General Usage Instructions

1- **Enter the memory size to be simulated**

   Note: maximum memory size is 20, minimum is 1; If the entered number is out of range, 20 will be used.

2- Enter the length of the reference string that is to be generated

   Note: minimum is 1, if less is entered, 20 will be used

3- **Enter the maximum Page Number to be referenced in the reference string**

   Note: minimum is 1, maximum is 99; if the entered number is out of range, 100 will be used

Output Format

For every scheduling algorithm, the output will contain the following

1- A starting decorator

2- Memory status for each page referenced containing the following

- **a.** The page being referenced
  - i. For enhanced second chance, it is also shown whether it is a read-only or read-write (modify) operation
- **b.** Memory After Reference
- **c.** Statuses of other Variables/Queues/Bits maintained by the specific scheduling algorithm as following:
  - i. FIFO: Queue Front
  - ii. LRU: None
  - iii. LFU: Specific frequencies for each page in the memory
  - iv. Second Chance: Reference Bits, Full Queue
  - v. Enhanced Second Chance: Reference Bits, Mod Bits
  - vi. Optimal: None

3- An End Decorator

At the end of the whole output, each method’s miss number is printed line by line for easier comparison between algorithms.

Implementations for each Algorithm

1- **FIFO**

   a. **Implementation**

   A memory hit changes nothing in the memory or FIFO queue.

   Memory pages are replaced on a first-in first-out basis if there is a memory miss; a page is only enqueued if it replaces another in the memory.

   b. **Notable Assumptions**

   None

2- **LRU**

   a. **Implementation**

   A memory hit changes nothing in the memory; however, the referenced page becomes the most recently used one.

   Memory pages are replaced by looking for the page that wasn't recently used for the longest time among other pages in the memory.

   b. **Notable Assumptions**

   None

3- **LFU**

   a. **Implementation**

   A memory hit changes nothing in the memory; however, it increments the referenced page's frequency of use by one.

   Memory pages are replaced by looking for the page that was least frequently used among other pages in the memory

   b. **Notable Assumptions**

      i. When a page is replaced, its frequency counter is reset to Zero.
      ii. A newly referenced page's counter is 1.
      iii. If 2 pages have the same frequency, the first one is replaced regardless of which one came in the memory first.

4- Second Chance

a. Implementation

A memory hit changes nothing in the memory or FIFO queue; However, the referenced page's reference bit is set to one.

Memory pages are replaced as follows; on a FIFO basis, the next victim page is determined and if its reference bit is zero, it is immediately replaced with the newly referenced page; else, its reference bit is reset to zero giving it a second chance.

b. Notable Assumptions

i. A newly referenced page has a reference bit = 0.
ii. When the next victim page's reference bit is reset to 0 to give it a second chance, it is moved to the back of the FIFO queue.

5- Enhanced Second Chance

a. Implementation

Determining whether a reference will be read-only/read-write (Modifying) is random.

A memory hit changes nothing in the memory; However, the referenced page's reference bit is set to one and its modified bit is set to one if it was a modify operation.

Memory pages are replaced as follows; the next victim page is the one after the last replaced page in the memory.

We start looking from the next victim for a page with reference and modified bits equal to zero; once one is found, it is replaced. If no pages with reference and modified bits equal to zero is found, we start looking from the next victim for a page with reference bit equal to zero and mod bit equal to one while resetting reference bits to zero along the way,
once one is found, it is replaced. If no pages with reference bit equal to one and mod bit equal to zero, pages' reference bits would've been reset to zero then they'd fall in one of the first two categories.

The newly referenced page's reference bit is zero, and its modified bit is set to one if it was a modify operation, zero if else.

b. Notable Assumptions

i. Next Victim is always the page after the one last replaced
ii. Referenced page's reference bit is 0
iii. Referenced page's modified bit is randomly determined

6- Optimal

a. Implementation

A memory hit changes nothing in the memory.

Memory pages are replaced as follows; The page that will not be for the most time among the pages in memory is replaced with the referenced page.

b. Notable Assumptions

None

Test Cases

Important Note : To view the full output, go to http://cpp.sh/8bglgd, uncomment any of the lines 459 to 462 (which contain the below test cases), press run then enter the respective frame size the enter any 2 random numbers for the string size and page numbers.

1- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1. Frame Size = 3

```
FIFO: 15                         LFU: 14                          enhancedSecondChance: 13

LRU: 12                          secondChance: 11                 optimal: 9
```

2- 1, 2, 3, 4, 1, 3, 6, 2, 1, 5, 3, 7, 6, 3, 2, 1, 2, 3, 4, 6. Frame Size = 4

```
FIFO: 15                         LFU: 11                          enhancedSecondChance: 10

LRU: 14                          secondChance: 13                 optimal: 9
```

3- 0, 1, 3, 6, 2, 4, 5, 2, 5, 0, 3, 1, 2, 5, 4, 1, 0. Frame Size = 3

```
FIFO: 15                         LFU: 14                          enhancedSecondChance: 14

LRU: 15                          secondChance: 15                 optimal: 11
```

4- 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2. Frame Size = 3

```
FIFO: 9                          LFU: 8                           enhancedSecondChance: 8

LRU: 7                           secondChance: 6                  optimal: 6
```