Name: Kalpesh Kagresha **UBIT**: kkagresh **Date:** 11-20-12

CSE 521: Project 2: Implement Virtual Memory Manager Design Document

In this project, we implemented the following page replacement algorithms

1. FIFO(First In First Out)
2. LFU (Least Frequently Used)
3. LRU-STACK (Least Recently Used)
4. LRU-CLOCK (Second Chance Algorithm)
5. LRU-REF8 (Reference Bit Algorithm)
6. Optimal Algorithm

The following data structure are used in the below page replacement algorithms

1.) ***First In First out (FIFO):*** In this algorithm, the oldest page is selected for replacement.
Method Declaration:
int implementFIFO(int availableFrames, int referenceString[], int len) => #pageReplaces
where availableFrames: Total No. of available frames
referenceString[]: List of pages that are referenced
len: Length of reference String Array
Data Structure:
1. int *frameArray: This pointer stores the pages from the reference string.
2. int fifoPointer: This pointer points to the oldest page. It becomes victim when we want to replace the page.
3. int pageReplacement: This variable stores the count of total page replacements.

2.) ***Least Frequently Used(LFU):*** In this algorithm, page with smallest count is replaced. In case of tie, FIFO algorithm is used.
Method Declaration:
int implementLFU(int availableframes,int referenceStrng[],int len) => #pageReplacement

Data Structure:

1. int *frameArray: Holds pages from reference String
2. int *indexInRefStr: Stores the index of pages in frameArray. This indexes are obtained from the reference string. This is requires in order to select page when their count is same.
3. Struct pageNode{int page, int count, pageNode *next}: It is linked list of all pages along with their counts in reference string.

Implementation: Whenever a page is found in frameArray, count of that page is incremented in pageNode. When is there is no empty frame, we find a page with minimum count i.e. Least Frequently Used. If there is a page with minimum count, it becomes victim. Otherwise, for all pages from frameArray with same count, we find minimum index from indexInRefStr. Thus in other words, oldest page with minimum count is replaced.

3.) ***Least Recently Used (Using Stack):*** In this algorithm, page which is not used in recent time is replaced. Thus page is replaced if it is not used for longest period of time.

Method Declaration:

int implementLRUStack(int availableFrames, int referenceString[], int len)=> #pageReplacement

Data Structure:

1. Struct frameNode{ int page, frameNode *next}: Linked List acting as stack
2. int lenLinkList: It is required when length of linked list equals available frames. Whenever there is page Fault, we select the page which is least recently used i.e. bottom of stack. The new page is inserted at top of the stack.

Implementation: Whenever there is page hit, we bring the page to top of stack. This makes it most recently used. Removing a frameNode involves changing the pointer next links which is pretty simple. In case of page fault, we remove the frameNode which is present at the bottom of stack. This node is not used for longest period of time.

4.) ***Least Recently Used - Clock (Second Chance Algorithm):*** In this algorithm, a reference bit is associated with each page in the frame array. If the value of bit is 0, we replace the page else we give page a second chance and move on to select the next page. When a page gets a second chance, its reference bit is cleared. Thus page that is given second chance will not be replaced until all other pages are replaced.

Method Declaration:

int implementLRUClock(int availableFrames, int referenceStrng[], int len)=> #pageReplacement

Data Structure:

1. struct frameNode{int page, int referenceBit, frameNode *next}:Use for circular linkedlist
2. frameNode *start: Head of linked list. Use for building the circular link list
3. frameNode * currentPostiton: Points to most recent node; Reqd during page replacement
4. int cLen: Length of circular linked list which is same as available frames.

Implemenation: Initially, when frame array is vacant, new frame node is created. It is repeated until cLen is less than available-Frames. When page is first observed, reference bit is set to 0. If cLen equals available-Frames, we join the last node link to start node making it a circular linked list. Whenever page is present in circular linkedlist, refBit is set to 1 otherwise we traverse until reach from where we started. If the page is not found, we again traverse the list making all Ref Bit 1=0 until we encounter a page with RefBit =0. This page becomes the victim and can be replaced. This page is pointed by the current Position frameNode.

5.) Least Recently Used-Reference Bits (LRU-REF8): In this algorithm, we maintain 8 bit for each unique page in the reference string. Whenever a page is referenced, all 8 bits are right shifted ignoring the lower order bit. In case of page hit, the higher order bit is set whereas in page replacement, we select that page which has minimum cost (obtained by converting binary to decimal form and taking minimum value).

Method Declaration:

int implementLruREf8(int availableFrames, int referenceString[], int len)=> #pageReplacement

shiftRight (int page, int refBit[][9], int count_unique)=>Right shifts and set higher order bit to 1

Data Structures:

1.) int *uniqueElements: Stores the unique pages from reference string.
2.) int count_unique: Stores the count of unique pages in uniqueElements array.
3.) int refBit[count_unique][9]: Row indicates the page no. whereas column = 1(page) + 8 (reference bits)
4.) int *frameArray: Size of availableFrames to store page from reference string

Implementation: It first finds the unique pages in reference string then initializes the 2D matrix of size [count_unique, 9]. In case of page Hit, we first make a right shift from bit 1to8 and then high order bit of referenced page is set. In case of page Fault, we convert 8 bit to decimal and select low value page for replacement.

6.) Optimal Algorithm: In this algorithm, we replace the page that will not be used for the longest period of time.

Method Declaration:

int implementOptimalAlgo (int availableFrames, int referenceString[], int len)=>#pageReplaced

Data Structures:

1. int *frameArray: Holds pages from reference string.
2. int victimPageIndex: Stores the page index from referenceString[] that will be not used for longest period of time.

Implemenatation: Whenever we want to replace page, we find the index of page present in frame array. This index tells when the page will be referred in future reference string. Thus the page with max index will be selected as victim page index. In case if the page is not referred in future, it becomes the most appropriate candidate for replacement.

--------------------X--------------------X--------------------X--------------------X--------------------X