S.Y.B.Tech

Computer Engineering

Lab : CE 2207 Operating Systems Laboratory (OSL)

Assignment #8: (Group-'B4')

Title: Write a Java program to implement following page replacement algorithms: First In First Out (FIFO) and Least Recently Used (LRU)

Objective: Implementation of page replacement algorithms.

Theory:

There are various memory management strategies that have the same goal→ keep many processes in memory in simultaneously to allow multiprogramming. However they require the entire process be in memory before process may execute.

Virtual memory: is a technique that allows the execution of processes that may not be completely in memory. **Virtual memory** is separation of user logical memory from physical memory.

- **Page replacement algorithms**
  There are three page replacement algorithms designed to achieve the lowest page fault rate. Every operating system has its own unique replacement scheme. We need to consider only page number not entire address.

1. **FIFO (First in first out)**
   Selects first page of main memory as victim page. Input is reference string of demand page numbers. Replaces one by one pages from main memory & satisfy one by one demand from the reference string

Advantages	**Disadvantages**
1) Very simple algorithm	1) it's facing Belady's anomaly
2) Easy to apply	2) Poor efficiency
	3) Not able to control page faults

Algorithm (FIFO):-

1) **Step1**: Accept the Reference pages String from User (rs)
2) **Step 2** : Find out first free frame
3) **Step 3** : If not free frame then free frame which is at the front and update rear and front increment the page fault counter
4) **Step 4** : Load frame[k]=rs[current] page from secondary memory(from reference string)
5) **Step5** : Update rear and front and current page and page fault counter
6) **Step 6**:Repeat steps 1 to 4 till reference string not over
7) **Step 7:** Display the status of the queue

Implementation

1) Read the input String rs[n] , fsize=3 (frame size /Queue)
2) Initialize queue rear= -1 front=0,frame[fsize]
3) Check queue is full or not → **If Yes** Go to step **5)**

→ **If not** then add rs[i] into frame [rear], Page_fault ++, Store the Queue state into an array other empty Frames indicated by **'-' or blank**

4) Check current page of input string (rs[i]) present in previous frame

→ **Yes** →don't increment page_fault, don't add that page, read next page from rs[n]

→ **If not** present→ add rs[i] at the frame [rear], increment Page_fault, Store the Queue state into an array

5) Replace the first page from frame (from front of queue), page_fault++, Store the Queue state into an array.
6) Follow the above steps till end of reference string rs[n].
7) Display the Sequence of Pages Replaced, Total no of Page Faults Occurred

Example---FIFO

Rs	**7**	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
F2		**0**	**0**	**0**	**0**	**3**	**3**	**3**	**2**	**2**	**2**	**2**	**2**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
F3			**1**	**1**	**1**	**1**	**0**	**0**	**0**	**3**	**3**	**3**	**3**	**3**	**2**	**2**	**2**	**2**	**2**	**1**
PF	**1**	**2**	**3**	**4**	**X**	**5**	**6**	**7**	**8**	**9**	**10**	**X**	**X**	**11**	**12**	**X**	**X**	**13**	**14**	**15**

Total page fault = 15

2. Least Recently Used (LRU) :

Selects least frequently used page of the main memory as a victim page. The page which is not used for long time (Used less number of times in past references string) Counters: here count is for each page whether page is referenced or not. Selects largest count page which is least recently used for victim page.

Advantages

1) Very simple algorithm

2) Easy to apply

3) Number of page faults are decreases

Disadvantages

1) it's facing Belady's anomaly for some strings

2) Not able to control page fault

3) Most of the time act as FIFO

Algorithm (LRU)

1) **Step1**: Accept the Reference pages String from User (rs[n])
2) **Step 2** : Find out first free frame
3) **Step 3**: If not free frame then free LRU (least recently used) page from frame using past reference.
   That means search pages present in frame from rs[0] to rs[current]

Calculate LRI (Last Referenced Index- selects page that has least recently used means that has occurred before the current page in past reference string of pages means Minimum index value)

4) **Step 4** : Load frame[k]=rs[current] page from secondary memory(from reference string)
5) **Step5** : Update rear and front and current page and page fault counter

6) Step 6: Follow the above steps till end of reference string rs[n].

7) Step 7: Display the Sequence of Pages Replaced, Total no of Page Faults Occurred

Implementation

1) Read the input String rs[n] , fsize=3 (frame size /Queue), array of frame[fsize]
2) Array for Index[fsize] → Store indexes of rs[n]
3) **Check all 3 are frames full or not a) → If Yes** Go to step **6)**

b) → If not then add rs[i] into frame[k], Page_fault ++, index **[0] = index of rs[i]** Store the array state into an array → **if any empty Frames→** indicate by **'-' or blank**

4) Check current page of input string (rs[i]) present in previous frame

a) → **yes** → Store index for frame in index[], don't increment page_fault , don't add that page, read next page from rs[n] then go to step 4

b) → **If not** present→ then add rs[i] into frame[k], Page_fault ++,

index[j] = index of rs[i] Store the frame state into an array → **if any empty Frames→** indicate by **'-' or blank,** read next page from rs[n] then go to step 4

5) **Replace the LRU page** from frame **Using LRI as follows**

Check current page of input string (rs[i]) present in previous frame

a) → **yes** → Store index for frame in index[], don't increment page_fault , don't add that page, read next page from rs[n] then go to step 5

b) → **If not** present→ Set min=index [0] , Calculate the min_index.

Replace page having minimum index, Store the index of replaced rs[i] for frame in index [] page_fault++, Store the Frame status into an array.

6) Follow the above steps till end of reference string rs[n].
7) Display the Sequence of Pages Replaced (status of the frame for every reference page), total no of Page Faults Occurred.

Example --LRU

Rs	**7**	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
F2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
F3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
PF	**1**	**2**	**3**	**4**	**X**	**5**	**X**	**6**	**7**	**8**	**9**	**X**	**X**	**10**	**X**	**11**	**X**	**12**	**X**	**X**

Total page fault = 12

Sample Output:

```
Enter the number of frames: 3
Enter the number of pages in the reference string: 12
Enter the reference string: 144 11 144 236 144 168 144 11 179 11 12 263

Menu:
1.FIFO
2.LRU
3.Exit
Enter your choice : 1

Reference string: 144 11 144 236 144 168 144 11 179 11 12 263

F1:  144  144  144  144  144  168  168   168   179  179  179  179
F2:   -    11   11   11   11   11   144  144   144  144  12    12
F3:   -    -    -    236  236  236  236  11    11   11   11   263
PF:   1    2    2    3    3    4    5    6     7    7    8    9
Do you want to continue? Press y for yes: y

Menu:
1.FIFO
2.LRU
3.Exit
Enter your choice : 2

Reference string: 144 11 144 236 144 168 144 11 179 11 12 263
F1:  144   144   144   144   144   144   144   144   144   144   12    12
F2:   -     11    11    11    11   168   168   168   179   179  179  263
F3:   -     -     -     236   236   236   236   11    11    11    11    11
PF:   1     2     2     3     3     4     4     5     6     6     7     8
Do you want to continue? Press y for yes: y

Menu:
1.FIFO
2.LRU
3.Exit
Enter your choice : 3
Termination of Program!!!
```