|  |
| --- |
| **Subject :Operating System Sub Teacher: Pro.S.S.Shete**  **Class: S.E. Computer Engg Roll no :-**  **Practical No.:**  **Date:** |

**Aim:** Write program to implement page replacement algorithms Demonstrate the working of page replacement algorithms

1. FIFO (first in first out)

2 .LRU(Least recently used)

**Theory:**

In a operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

**Page Fault –**

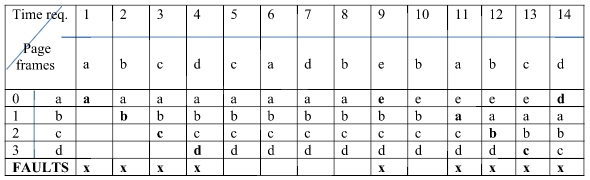
A page fault is a type of interrupt, raised by the hardware when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

**Page Replacement Algorithms :**

* **First In First Out (FIFO) –**  
  This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

For example-1, consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots.

Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —>**3 Page Faults.**  
when 3 comes, it is already in  memory so —>**0 Page Faults.**  
Then 5 comes, it is not available in  memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.**  
Finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**

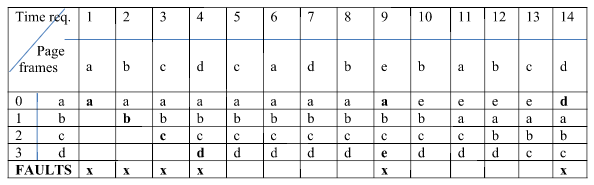
Example-2, Let’s have a reference string: a, b, c, d, c, a, d, b, e, b, a, b, c, d and the size of the frame be 4.  
  
There are 9 page faults using FIFO algorithm.

**Belady’s anomaly –**Belady’s anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.  For example, if we consider reference string 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

* **Optimal Page replacement –**  
  In this algorithm, pages are replaced which are not used for the longest duration of time in the future.

Let us consider page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 and 4 page slots.

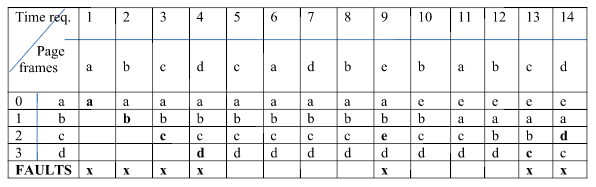
Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**  
0 is already there so —>**0 Page fault.**  
when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**  
0 is already there so —> **0 Page fault.**.  
4 will takes place of 1 —> **1 Page Fault.**  
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Example-2, Let’s have a reference string: a, b, c, d, c, a, d, b, e, b, a, b, c, d and the size of the frame be 4.  
There are 6 page faults using optimal algorithm.

Optimal page replacement is perfect, but not possible in practice as operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

* **Least Recently Used –**  
  In this algorithm page will be replaced which is least recently used.

Let say the page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 . Initially we have 4 page slots empty.  
Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**  
0 is already their so —>**0 Page fault.**  
when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**  
0 is already in memory so —> **0 Page fault**.  
4 will takes place of 1 —> **1 Page Fault**  
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Example-2, Let’s have a reference string: a, b, c, d, c, a, d, b, e, b, a, b, c, d and the size of the frame be 4.  
  
There are 7 page faults using LRU algorithm.

**ALGORITHM:** **To implement FIFO page replacement algorithm**

1. Start the process

2. Declare the size with respect to page length

3. Check the need of replacement from the page to memory

4. Check the need of replacement from old page to new page in memory

5. Forma queue to hold all pages

6. Insert the page require memory into the queue

7. Check for bad replacement and page fault

8. Get the number of processes to be inserted

9. Display the values

10. Stop the process.

**ALGORITHM: To implement LRU page replacement algorithm**

1. Start the process

2. Declare the size

3. Get the number of pages to be inserted

4. Get the value

5. Declare counter and stack

6. Select the least recently used page by counter value

7. Stack them according the selection.

8. Display the values

9. Stop the process.

**Aim:** Write a program to implement FIFO page replacement algorithm

#include<stdio.h>

intmain()

{

inti,j,n,a[50],frame[10],no,k,avail,count=0;

printf("\n enter the no of page\n:");

scanf("%d",&n);

printf("\nenter the page no:\n");

for(i=1;i<=n;i++)

scanf("%d",&a[i]);

printf("\n enter the no. of frames:");

scanf("%d",&no);

for(i=0;i<no;i++)

frame[i]=-1;

j=0;

printf("ref string\t page frames\n");

for(i=1;i<=n;i++)

{

printf("%d\t\t",a[i]);

avail=0;

for(k=0;k<no;k++)

if(frame[k]==a[i])

avail=1;

if(avail==0)

{

frame[j]=a[i];

j=(j+1)%no;

count++;

for(k=0;k<no;k++)

printf("%d\t",frame[k]);

}

printf("\n");

}

printf("page fault is %d",count);

return 0;

}

**Output:-**

gcoe@GCOE-LINUX:~/Desktop$ cc page\_r.c

gcoe@GCOE-LINUX:~/Desktop$ ./a.out

enter the no of page

:20

enter the page no:

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

enter the no. of frames:3

ref string page frames

7 7 -1 -1

0 7 0 -1

1 7 0 1

2 2 0 1

0

3 2 3 1

0 2 3 0

4 4 3 0

2 4 2 0

3 4 2 3

0 0 2 3

3

2

1 0 1 3

2 0 1 2

0

1

7 7 1 2

0 7 0 2

1 7 0 1

page fault is 15gcoe@GCOE-LINUX:~/Desktop$

**Aim :** write a program to implement LRU page replacement algorithm.

#include<stdio.h>

void main()

{

int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];

printf("enter the no of pages:");

scanf("%d",&n);

printf("enter the refference string:");

for(i=0;i<n;i++)

scanf("%d",&p[i]);

printf("enter the no of frames:");

scanf("%d",&f);

q[k]=p[k];

printf("\n\t%d\n",q[k]);

c++;

k++;

for(i=1;i<n;i++)

{

c1=0;

for(j=0;j<f;j++)

{

if(p[i]!=q[j])

c1++;

}

if(c1==f)

{

c++;

if(k<f)

{

q[k]=p[i];

k++;

for(j=0;j<k;j++)

printf("\t%d",q[j]);

printf("\n");

}

else

{

for(r=0;r<f;r++)

{

c2[r]=0;

for(j=i-1;j<n;j--)

{

if(q[r]!=p[j])

c2[r]++;

else

break;

}

}

for(r=0;r<f;r++)

b[r]=c2[r];

for(r=0;r<f;r++)

{

for(j=r;j<f;j++)

{

if(b[r]<b[j])

{

t=b[r];

b[r]=b[j];

b[j]=t;

}

}

}

for(r=0;r<f;r++)

{

if(c2[r]==b[0])

q[r]=p[i];

printf("\t%d",q[r]);

}

printf("\n");

}

}

}

printf("\n the no of page faults is %d",c);

}

**Output:-**

gcoe@GCOE-LINUX:~/Desktop$ cc lru.c

gcoe@GCOE-LINUX:~/Desktop$ ./a.out

enter the no of pages:10

enter the refference string:7 5 9 4 3 7 9 6 2 1

enter the no of frames:3

7

7 5

7 5 9

4 5 9

4 3 9

4 3 7

9 3 7

9 6 7

9 6 2

1 6 2

the no of page faults is 10gcoe@GCOE-LINUX:~/Desktop$ cd Desktop/