**OPERATING SYSTEMS**

LAB EXPERIMENT - 8

Submitted To: Prashant Giridhar Shambharkar Sir.

Submitted By: Manav Agrawal (2K19/CO/216)

---------------------------------------------------------------------

Aim:

Write a C program to simulate page replacement algorithms a) FIFO b) LRU

Introduction:

Page Replacement Algorithms:

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

Page Fault: A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

The following are the page replacement algorithms which we will be implementing:

### First In First Out (FIFO):

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the 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.

*Example:* Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames.Find number of page faults. 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.

6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>1 Page Fault. Finally when 3 come it is not available so it replaces 0 1 page fault

*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.

### Least Recently Used:

In this algorithm page will be replaced which is least recently used.

*Example-* Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames. Find number of page faults.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.

Algorithms:

Let capacity be the number of pages that memory can hold. Let set be the current set of pages in memory.

1. **First In First Out (FIFO)**

1- Start traversing the pages.

i) If set holds less pages than capacity.

1. Insert page into the set one by one until

the size of set reaches capacity or all

page requests are processed.

1. Simultaneously maintain the pages in the

queue to perform FIFO.

1. Increment page fault

ii) Else

If current page is present in set, do nothing.

Else

1. Remove the first page from the queue

as it was the first to be entered in

the memory

1. Replace the first page in the queue with

the current page in the string.

1. Store current page in the queue.
2. Increment page faults.

2- Return page faults.

1. **Least Recently Used**

1- Start traversing the pages.

i) If set holds less pages than capacity.

1. Insert page into the set one by one until

the size of set reaches capacity or all

page requests are processed.

1. Simultaneously maintain the recent occurred

index of each page in a map called indexes.

1. Increment page fault

ii) Else

If current page is present in set, do nothing.

Else

1. Find the page in the set that was least

recently used. We find it using index array.

We basically need to replace the page with

minimum index.

1. Replace the found page with current page.
2. Increment page faults.
3. Update index of current page.

2. Return page faults.

Implementation:

**A) First In First Out (FIFO)**

#include<stdio.h>

#include<conio.h>

int main()

{

int a[5],b[20],n,p=0,q=0,m=0,h,k,i,q1=1;

char f='F';

printf("Enter the Number of Pages:"); scanf("%d",&n);

printf("Enter %d Page Numbers:",n); for(i=0;i<n;i++)

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

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

{if(p==0)

{

if(q>=3)

q=0;

a[q]=b[i];

q++;

if(q1<3)

{

q1=q;

}

}

printf("\n%d",b[i]);

printf("\t");

for(h=0;h<q1;h++)

printf("%d",a[h]);

if((p==0)&&(q<=3))

{

printf("-->%c",f);m++;

}

p=0;

for(k=0;k<q1;k++)

{

if(b[i+1]==a[k])

p=1;

}

}

printf("\nNo of faults:%d",m);

getch();

return 0;

}

**B) Least Recently Used**

#include<stdio.h>

#include<conio.h>

int main()

{

int g=0,a[5],b[20],p=0,q=0,m=0,h,k,i,q1=1,j,u,n;

char f='F';

printf("Enter the number of pages:"); scanf("%d",&n);

printf("Enter %d Page Numbers:",n); for(i=0;i<n;i++)

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

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

{if(p==0)

{

if(q>=3)

q=0;

a[q]=b[i];

q++;

if(q1<3)

{

q1=q;

}

}

printf("\n%d",b[i]);

printf("\t");

for(h=0;h<q1;h++)

printf("%d",a[h]);

if((p==0)&&(q<=3))

{

printf("-->%c",f);m++;

}

p=0;

g=0;

if(q1==3)

{

for(k=0;k<q1;k++)

{

if(b[i+1]==a[k])

p=1;

}

for(j=0;j<q1;j++)

{

u=0;

k=i;while(k>=(i-1)&&(k>=0))

{

if(b[k]==a[j])

u++;k--;

}

if(u==0)

q=j;

}

}

else

{

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

{

if(b[i+1]==a[k])

p=1;

}

}

}

printf("\nNo of faults:%d",m);

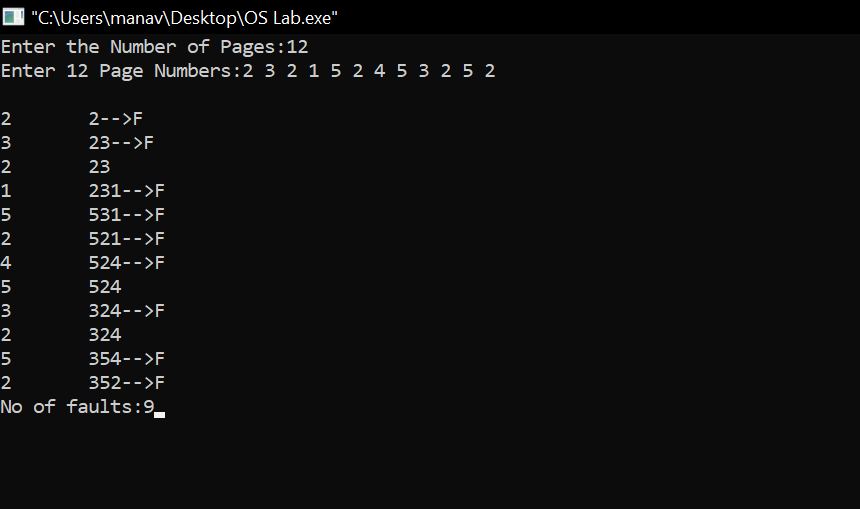
getch();

return 0;

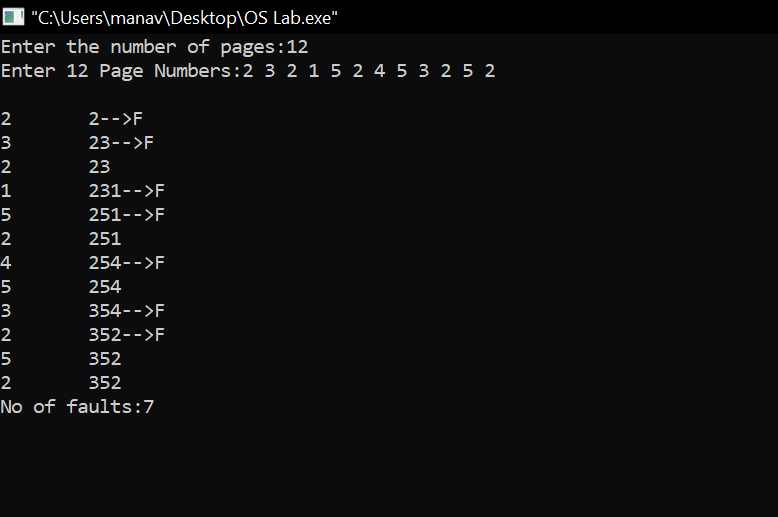
}

Output:

**A) First In First Out (FIFO)**



**B) Least Recently Used**



Learning From The Experiment:

Both the algorithms are have their pros and cons.

The only advantage of FIFO is that it is simple and easy to understand & implement. However, the disadvantage is that the process effectiveness is low and when we increase the number of frames while using FIFO, we are giving more memory to processes. So, page fault should decrease, but here the page faults are increasing. This problem is called as Belady’s Anomaly.

The advantage of LRU is that it is open for full analysis. We replace the page which is least recently used in LRU, thus free from Belady’s Anomaly. It is easy to choose page which has faulted and hasn’t been used for a long time. However, it requires additional Data Structure to be implemented.The hardware assistance is high.

***THANK YOU!***