OPERATING SYSTEMS

CECSC09 - 1

Submitted by :-

NAME: Harshit Gupta

ROLL NO: 2019UCO1580

Program 10

To show the working of the FIFO page replacement algorithm

- The following program highlights how the page frames are allocated inside a system when the replacement strategy is *first in first out*
- This is one of the simplest page replacement strategies used in the demand paging type systems but suffers from the belady's anomaly and is not very efficient.

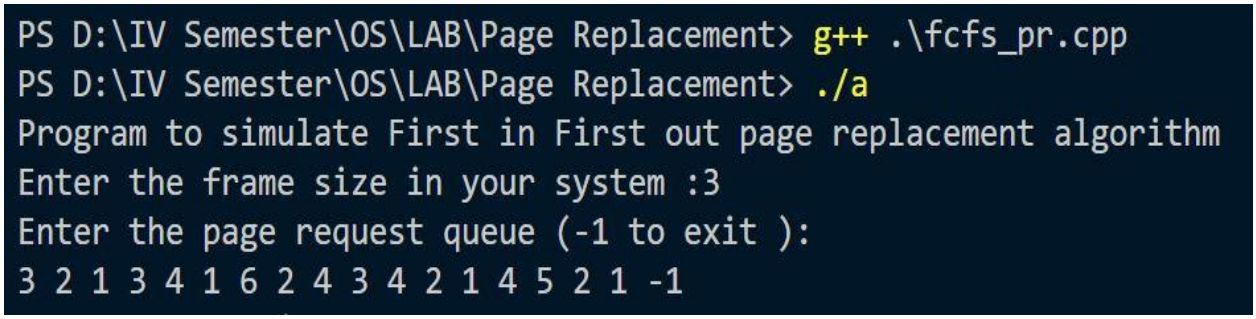
CODE

```
#include<bits/stdc++.h>
#define all(x) x.begin(),x.end()
using namespace std;

int main(){
    ios::sync_with_stdio(0);
    cout<<"Program to simulate First in First out page replacement
algorithm\n";
    int frame_size;
    cout<<"Enter the frame size in your system :";
    cin>>frame_size;

    set<int> frame;
    map<int,int> indices;
    int hits,misses;
    hits = misses = 0;
    // Page queue
    cout<<"Enter the page request queue (-1 to exit ):\n";
    int c = 0,min,to_replace,page;
    vector<int> pages;
    pages.reserve(100);

    while(true){
        cin>>page;
        if(page == -1) break;
```

```
        pages.push_back(page);
    }

    // Evaluate
    for(auto page : pages){

        // associate earliest time stamps
        if(indices.find(page) == indices.end()){
            indices[page] = c;
        }

        // check if the page is in the frame or not
        if(frame.find(page)!= frame.end()){
            // page is found
            hits++;
            cout<<"Hit! "<<page<<" found in frame\n";
        }
        else{
            misses++;
            if(frame.size()!= frame_size){
                // no replacement required , just insert
                frame.insert(page);
                cout<<"Free frame available!\n";
            }
            else{
                // replacement required
                min = 1e5;
                for(auto k:frame){
                    if(indices[k] < min){
                        min = indices[k];
                        to_replace = k;
                    }
                }
                cout<<to_replace<<" is replaced by "<<page<<endl;

                // page is removed
                frame.erase(to_replace);
                indices.erase(to_replace);
                frame.insert(page);
```

```
            }
        }

        cout<<"Current Page Frame :\n";
        for(auto k:frame) cout<<k<<" ";

        cout<<endl;

        c++;

    }
    // Hit and miss ratio
    cout<<"Total numbers of hits :"<<hits<<endl;
    cout<<"Total number of misses :"<<misses<<endl;

    float hr,mr;
    hr = float(hits)/(c);
    mr = 1.0 - hr;
    cout<<"Hit ratio :"<<hr<<endl;
    cout<<"Miss ratio :"<<mr<<endl;
    return 0;
}
```

SCREENSHOTS



```
Free frame available!
Current Page Frame :
3
Free frame available!
Current Page Frame :
2 3
Free frame available!
Current Page Frame :
1 2 3
Hit! 3 found in frame
Current Page Frame :
1 2 3
3 is replaced by 4
Current Page Frame :
1 2 4
Hit! 1 found in frame
Current Page Frame :
1 2 4
2 is replaced by 6
Current Page Frame :
1 4 6
1 is replaced by 2
Current Page Frame :
2 4 6
Hit! 4 found in frame
Current Page Frame :
```

```
Current Page Frame :
2 4 6
4 is replaced by 3
Current Page Frame :
2 3 6
6 is replaced by 4
Current Page Frame :
2 3 4
Hit! 2 found in frame
Current Page Frame :
2 3 4
2 is replaced by 1
Current Page Frame :
1 3 4
Hit! 4 found in frame
Current Page Frame :
1 3 4
3 is replaced by 5
Current Page Frame :
1 4 5
4 is replaced by 2
Current Page Frame :
1 2 5
6 is replaced by 4
Current Page Frame :
2 3 4
```

```
1 2 5
6 is replaced by 4
Current Page Frame :
2 3 4
Hit! 2 found in frame
Current Page Frame :
2 3 4
2 is replaced by 1
Current Page Frame :
1 3 4
Hit! 4 found in frame
Current Page Frame :
1 3 4
3 is replaced by 5
Current Page Frame :
1 4 5
4 is replaced by 2
Current Page Frame :
1 2 5
Hit! 1 found in frame
Current Page Frame :
1 2 5
Total numbers of hits :6
Total number of misses :11
Hit ratio :0.352941
Miss ratio :0.647059
```

Program 11

To show the working of the LRU page replacement algorithm

- The following program highlights how the page frames are allocated inside a system when the replacement strategy is *least recently used*
- This strategy looks for the page ‘use’ in the request queue and replaces the page which was used the earliest and is present in the current frame

CODE

```
#include<bits/stdc++.h>
#define all(x) x.begin(),x.end()
#define pb(x) push_back(x)
using namespace std;
typedef vector<int> vi;
int main(){
   cout<<"LEAST RECENTLY USED PAGE REPLACEMENT ....\n";
   cout<<"Please enter the page request queue ( -1 to exit ) :\n";
    int miss,hits;
    hits = miss = 0;
    vi pages;
    int f_size;
    set<int> frame;

    cout<<"Enter the frame size of the system :";
    cin>>f_size;
    int ele;
    while(ele!=-1){
        cin>>ele;
        if(ele == -1) break;

        pages.pb(ele);
    }
    // GOALS
    // 1.Find the number of page faults encountered
    // 2.Find the hit rate and miss rate
```

```
    int c = 0;
    map< int,int > indices;
    int i,min_ind;
    for(auto k:pages){

        indices[k] = c;
        // if frame contains element

        // hit
        if(frame.find(k) != frame.end()){
            // got a hit
            cout<<"\nHit! Page "<<k<<" found in the frame.";
            hits++;
            c++;
            continue;
        }

//

        //else
        miss++;
        // frame does not have the page
        if(frame.size()!= f_size){
            frame.insert(k);

        }
        else{
            // Frame is full

                min_ind = 1e6;
                for(auto j:frame){
                    i = indices[j];
                    if(i < min_ind){

                        ele = j;
                        min_ind = i;
                    }
                }
```

```
                // at this point I have the minimum index element
                cout<<endl<<ele<<" is replaced with "<<k;
                // replace it
                frame.erase(ele);
                indices.erase(ele);
                // insert the current page
                frame.insert(k);
            }


    c++;
    cout<<endl;
    cout<<"Current Frame :\n";
    for(auto k:frame) cout<<k<<" ";

    }

    cout<<"\nNumber of misses :"<<miss<<endl;
    cout<<"Number of hits :"<<hits<<endl;
    float hr = float(hits)/(hits + miss);
    cout<<"The hit ratio of the algorithm :"<<hr<<endl;
    cout<<"Miss ratio of the algorithm :"<<1.0 - hr;

    return 0;}
```

SCREENSHOTS

```
PS D:\IV Semester\OS\LAB\Page Replacement> g++ lru.cpp
PS D:\IV Semester\OS\LAB\Page Replacement> ./a
LEAST RECENTLY USED PAGE REPLACEMENT ....
Please enter the page request queue ( -1 to exit ) :
Enter the frame size of the system :3
3 2 1 3 4 1 6 2 4 3 2 4 1 4 5 2 1 -1
```

```
Current Frame :
3
Current Frame :
2 3
Current Frame :
1 2 3
Hit! Page 3 found in the frame.
2 is replaced with 4
Current Frame :
1 3 4
Hit! Page 1 found in the frame.
3 is replaced with 6
Current Frame :
1 4 6
4 is replaced with 2
Current Frame :
1 2 6
1 is replaced with 4
Current Frame :
2 4 6
6 is replaced with 3
Current Frame :
2 3 4
Hit! Page 2 found in the frame.
Hit! Page 4 found in the frame.
3 is replaced with 1
```

```
Current Frame :
1 2 4
Hit! Page 4 found in the frame.
2 is replaced with 5
Current Frame :
1 4 5
1 is replaced with 2
Current Frame :
2 4 5
4 is replaced with 1
Current Frame :
1 2 5
Number of misses :12
Number of hits :5
The hit ratio of the algorithm :0.294118
Miss ratio of the algorithm :0.705882
```

Program 12

To show the working of the Second Chance page replacement algorithm

- In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and a page that has been accessed between consecutive considerations will not be replaced.
- The page replaced is the one that, when considered in a round robin matter, has not been accessed since its last consideration.

CODE

```
#include<bits/stdc++.h>
#include<windows.h>

using namespace std;
int main(){
    ios::sync_with_stdio(0);

    cout<<"Program to simulate SECOND CHANCE ALGORITHM\n";
    int f_size,len;
    cout<<"Enter frame size : ";
    cin>>f_size;
    list <int> frame;
    set <int> frame_copy;
    map <int,bool> reference;
    cout<<"Enter the number of page references : ";
    cin>>len;
    cout<<"Simulating pages...\n";
    int c;
    list<int>::iterator it2;

    bool found;
    int hit,miss,count,page,to_replace;
    hit = count = miss = 0;
    while(len--){

        page = rand()%9 + 1;
```

```
        // page was referenced
        reference[page] = 1;

        if(frame_copy.find(page) != frame_copy.end()){
            cout<<"Hit! "<<page<<" found in frame\n";
            hit++;
        }
        else{
            miss++;


            if(frame.size() != f_size){
                // fifo insertion
                frame.push_back(page);
                frame_copy.insert(page);
            }

            else{

                // replacement needs to be done
                found = false;
                auto it = frame.begin();
                while(found == false){

                    if(reference[*it] == 1){
                        reference[*it] = 0;
                        //second chance given
                        it++;
                        if(it==frame.end()) // loop back
                            it = frame.begin();
                    }
                    // found a page
                    else{
                        found = true;
                        to_replace = *it;

                        //it2 is the iterator pointing to the
                        // element before which we want to insert
                        // new page
```

```
                        it2 = frame.erase(it);
                        frame.insert(it2,page);

                        // update the copy
                        frame_copy.erase(to_replace);
                        frame_copy.insert(page);
                    }
                }

                cout<<to_replace<<" was replaced by "<<page<<endl;
            }
        }

    cout<<"Current frame : ";
    for(auto k:frame) cout<<k<<" ";
    cout<<endl;

    cout<<"Reference : ";
    for(auto k:frame) cout<<reference[k]<<" ";
    cout<<endl;
    count++;
    _sleep(400);
    }


    cout<<"Total hits :"<<hit<<endl;
    cout<<"Total missed :"<<miss<<endl;
    float hr = float(hit)/count;
    cout<<"Hit ratio :"<<hr<<endl;
    cout<<"Miss ratio :"<<1.0 - hr;

    return 0;
}
```

SCREENSHOTS

```
PS D:\IV Semester\OS\LAB\Page Replacement> ./a
Program to simulate SECOND CHANCE ALGORITHM
Enter frame size : 4
Enter the number of page references : 12
Simulating pages...
Current frame : 6
Reference : 1
Current frame : 6 9
Reference : 1 1
Current frame : 6 9 8
Reference : 1 1 1
Current frame : 6 9 8 5
Reference : 1 1 1 1
Hit! 9 found in frame
Current frame : 6 9 8 5
Reference : 1 1 1 1
6 was replaced by 2
Current frame : 2 9 8 5
Reference : 1 0 0 0
9 was replaced by 4
```

```
Current frame : 2 4 8 5
Reference : 0 1 0 0
2 was replaced by 1
Current frame : 1 4 8 5
Reference : 1 1 0 0
Hit! 8 found in frame
Current frame : 1 4 8 5
Reference : 1 1 1 0
5 was replaced by 3
Current frame : 1 4 8 3
Reference : 0 0 0 1
1 was replaced by 9
Current frame : 9 4 8 3
Reference : 1 0 0 1
Hit! 3 found in frame
Current frame : 9 4 8 3
Reference : 1 0 0 1
Total hits :3
Total missed :9
Hit ratio :0.25
Miss ratio :0.75
```

Program 13

Program to show the working of the LFU page replacement algorithm

- In this algorithm, the operating system keeps track of all pages in the memory in a queue.
- When a page needs to be replaced, the operating system chooses the page which is least frequently used for the replacement with the incoming page.
- *Hash Table* data structure is used to keep the frequency array for a page and *set* is used to simulate a frame in C++.

CODE

```
#include<bits/stdc++.h>
#define for0(i,n) for(int i=0;i<n;i++)
#include<windows.h>
using namespace std;
int main(){
    ios::sync_with_stdio(0);

    //Declare frame
    set<int> frame;

    //Declare  hash table for frequencies
    unordered_map <int,int> freq;

    cout<<"Program to simulate LEAST FREQUENTLY USED page replacement\n";
    int f_size,len,min,to_replace;

    //SIMULATION
    cout<<"Enter the frame size :";
    cin>>f_size;
    cout<<"Enter how many pages you want to simulate :";
    cin>>len;
    cout<<"Simulating page request queue :\n";

    int hit,miss,count,page;
    hit = count = miss = 0;
```

```
    for0(i,len){
        page = rand()%9 + 1;
        cout<<"Page :"<<page<<endl;

        // referenced
        freq[page]++;

        if(frame.find(page) != frame.end()){
            // hit
            cout<<"Hit! "<<page<<" found in frame\n";
            hit++;
        }
        else{
            //miss
            miss++;
            cout<<"Miss! ";
            if(frame.size() != f_size){
                // some empty space available
                cout<<page<<" inserted\n";
                frame.insert(page);
            }

            else{
                // no empty space, replacement required
                min = 1e7;

                // choose minimum used page
                for(auto k:frame){
                    if(freq[k] <= min){
                        min = freq[k];
                        to_replace = k;
                    }
                }
                cout<<to_replace<<" replace with "<<page<<endl;
                // replace the page
                frame.erase(to_replace);
                frame.insert(page);
                freq[to_replace] = 0; // erased page
            }
```

```
        }

    cout<<"Current frame : ";
    for(auto k:frame) cout<<k<<" ";

    cout<<endl;
    count++;
    _sleep(400);
    }

    cout<<"Total hits :"<<hit<<endl;
    cout<<"Total missed :"<<miss<<endl;
    float hr = float(hit)/count;
    cout<<"Hit ratio :"<<hr<<endl;
    cout<<"Miss ratio :"<<1.0 - hr;
    return 0;
}
```

SCREENSHOTS

```
PS D:\IV Semester\OS\LAB\Page Replacement> ./a
Program to simulate LEAST FREQUENTLY USED page replacement
Enter the frame size :4
Enter how many pages you want to simulate :12
Simulating page request queue :
```

```
Page :6
Miss! 6 inserted
Current frame : 6
Page :9
Miss! 9 inserted
Current frame : 6 9
Page :8
Miss! 8 inserted
Current frame : 6 8 9
Page :5
Miss! 5 inserted
Current frame : 5 6 8 9
Page :9
Hit! 9 found in frame
Current frame : 5 6 8 9
Page :2
Miss! 8 replace with 2
Current frame : 2 5 6 9
Page :4
Miss! 6 replace with 4
Current frame : 2 4 5 9
```

```
Page :1
Miss! 5 replace with 1
Current frame : 1 2 4 9
Page :8
Miss! 4 replace with 8
Current frame : 1 2 8 9
Page :3
Miss! 8 replace with 3
Current frame : 1 2 3 9
Page :9
Hit! 9 found in frame
Current frame : 1 2 3 9
Page :3
Hit! 3 found in frame
Current frame : 1 2 3 9
Total hits :3
Total missed :9
Hit ratio :0.25
Miss ratio :0.75
```