21BAI1217
Mainak Chattopadhyay
OS LAB - 9

Question

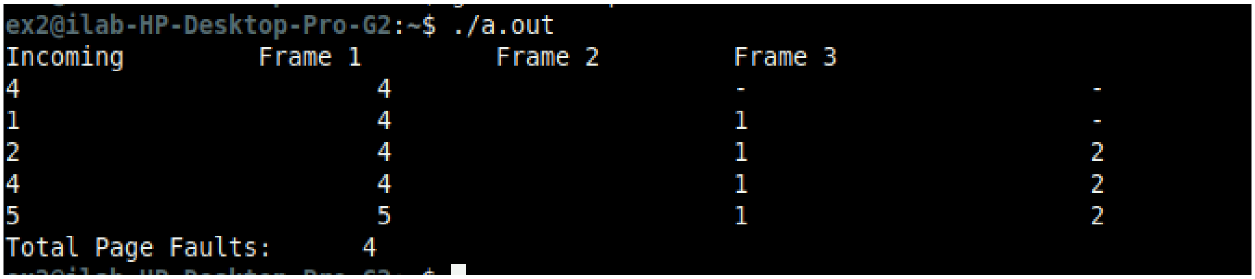
1. Write a C program to implement the following page replacement algorithms: FIFO,LRU and Optimal algorithm.

FIFO

CODE -

```
#include<stdio.h>
int main()
{
    int incomingStream[] = {4, 1, 2, 4, 5};
    int pageFaults = 0;
    int frames = 3;
    int m, n, s, pages;

    pages = sizeof(incomingStream)/sizeof(incomingStream[0]);

    printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");
    int temp[frames];
    for(m = 0; m < frames; m++)
    {
        temp[m] = -1;
    }

    for(m = 0; m < pages; m++)
    {
        s = 0;

        for(n = 0; n < frames; n++)
        {
            if(incomingStream[m] == temp[n])
            {
                s++;
                pageFaults--;
            }
        }
        pageFaults++;

        if((pageFaults <= frames) && (s == 0))
        {
            temp[m] = incomingStream[m];
        }
        else if(s == 0)
        {
            temp[(pageFaults - 1) % frames] = incomingStream[m];
        }

        printf("\n");
        printf("%d\t\t\t",incomingStream[m]);
        for(n = 0; n < frames; n++)
        {
            if(temp[n] != -1)
                printf(" %d\t\t\t", temp[n]);
            else
                printf(" - \t\t\t");
        }
    }

    printf("\nTotal Page Faults:\t%d\n", pageFaults);
    return 0;
}
```

OUTPUT-



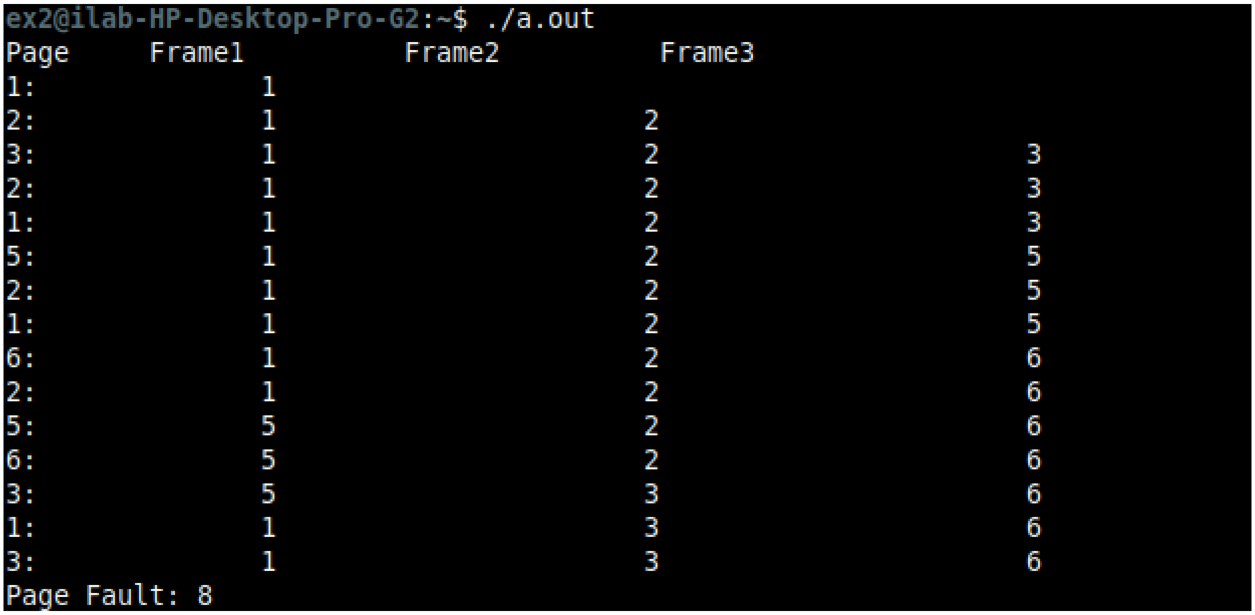
```
ex2@ilab-HP-Desktop-Pro-G2:~$ ./a.out
Incoming         Frame 1         Frame 2         Frame 3
4                        4                       -                       -
1                        4                       1                       -
2                        4                       1                       2
4                        4                       1                       2
5                        5                       1                       2
Total Page Faults:      4
ex2@ilab-HP-Desktop-Pro-G2:~$
```

LRU

CODE-

```
#include<stdio.h>
#include<limits.h>

int checkHit(int incomingPage, int queue[], int occupied){

    for(int i = 0; i < occupied; i++){
        if(incomingPage == queue[i])
            return 1;
    }

    return 0;
}

void printFrame(int queue[], int occupied)
{
    for(int i = 0; i < occupied; i++)
        printf("%d\t\t\t",queue[i]);
}

int main()
{

//    int incomingStream[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1};
//    int incomingStream[] = {1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3};
    int incomingStream[] = {1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3};

    int n = sizeof(incomingStream)/sizeof(incomingStream[0]);
    int frames = 3;
    int queue[n];
    int distance[n];
    int occupied = 0;
    int pagefault = 0;

    printf("Page\t Frame1 \t Frame2 \t Frame3\n");

    for(int i = 0;i < n; i++)
    {
        printf("%d:  \t\t",incomingStream[i]);
        // what if currently in frame 7
        // next item that appears also 7
        // didnt write condition for HIT

        if(checkHit(incomingStream[i], queue, occupied)){
            printFrame(queue, occupied);
        }
        // filling when frame(s) is/are empty
        else if(occupied < frames){
            queue[occupied] = incomingStream[i];
            pagefault++;
            occupied++;

            printFrame(queue, occupied);
        }
        else{

            int max = INT_MIN;
            int index;
            // get LRU distance for each item in frame
            for (int j = 0; j < frames; j++)
            {
                distance[j] = 0;
                // traverse in reverse direction to find
                // at what distance  frame item occurred last
                for(int k = i - 1; k >= 0; k--)
                {
                    ++distance[j];

                    if(queue[j] == incomingStream[k])
                        break;
                }

                // find frame item with max distance for LRU
                // also notes the index of frame item in queue
                // which appears furthest(max distance)
                if(distance[j] > max){
                    max = distance[j];
                    index = j;
                }
            }
            queue[index] = incomingStream[i];
            printFrame(queue, occupied);
            pagefault++;
        }

        printf("\n");
    }

    printf("Page Fault: %d \n",pagefault);

    return 0;
}
```

OUTPUT-



```
ex2@ilab-HP-Desktop-Pro-G2:~$ ./a.out
Page     Frame1          Frame2          Frame3
1:              1
2:              1                       2
3:              1                       2                       3
2:              1                       2                       3
1:              1                       2                       3
5:              1                       2                       5
2:              1                       2                       5
1:              1                       2                       5
6:              1                       2                       6
2:              1                       2                       6
5:              5                       2                       6
6:              5                       2                       6
3:              5                       3                       6
1:              1                       3                       6
3:              1                       3                       6
Page Fault: 8
```

Optimal Algorithm

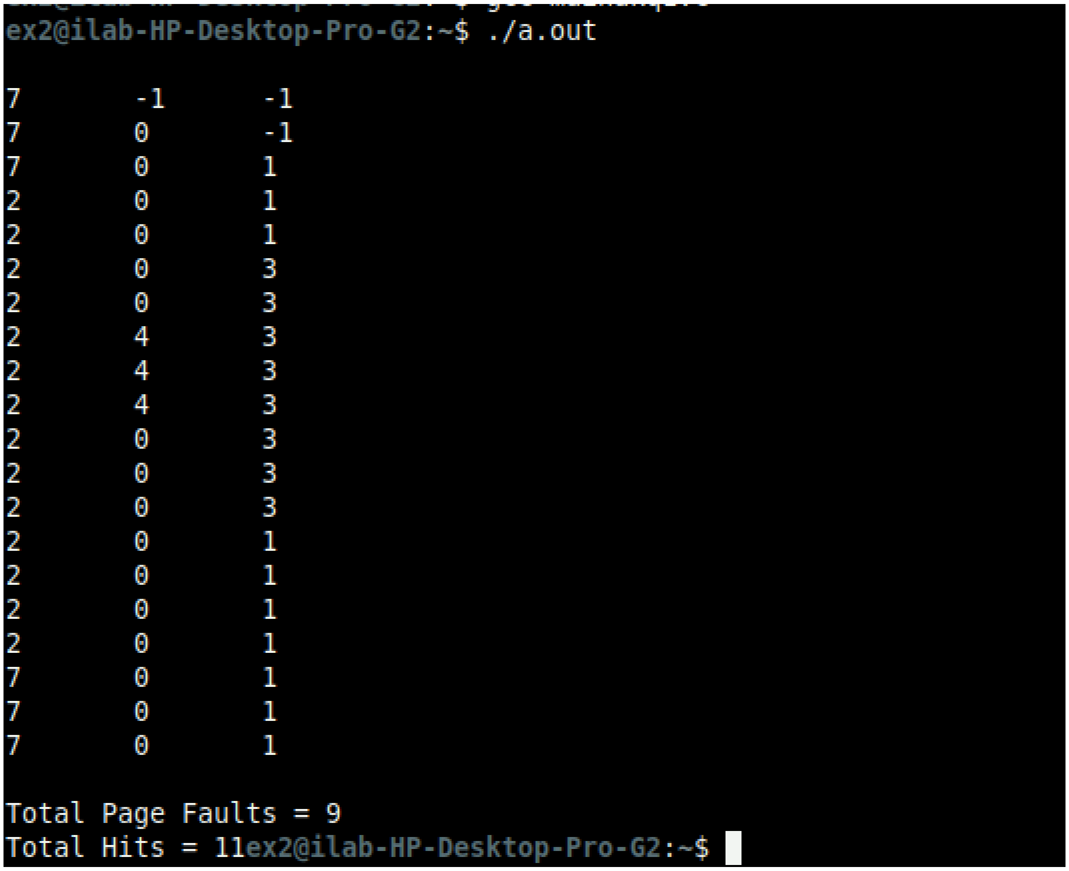
CODE-

```
#include <stdio.h>
 int main()
{
    int flag1, flag2, flag3, i, j, k, position, max, faults = 0;
    int num_frames = 3;

    int frames[num_frames];
    int temp[num_frames];

    int inputStream[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1};

    int num_pages = sizeof(inputStream) / sizeof(inputStream[0]);

    for(i = 0; i < num_frames; i++){
        frames[i] = -1;
    }

    for(i = 0; i < num_pages; i++){

        flag1 = flag2 = 0;

        for(j = 0; j < num_frames; j++){
            if(frames[j] == inputStream[i]){
                   flag1 = flag2 = 1;
                   break;
               }
        }

        if(flag1 == 0){
            for(j = 0; j < num_frames; j++){
                if(frames[j] == -1){
                    faults++;
                    frames[j] = inputStream[i];
                    flag2 = 1;
                    break;
                }
            }
        }

        if(flag2 == 0){
```

```
        flag3 =0;

        for(j = 0; j < num_frames; j++){
            temp[j] = -1;

            for(k = i + 1; k < num_pages; k++){
                if(frames[j] == inputStream[k]){
                    temp[j] = k;
                    break;
                }
            }
        }

        for(j = 0; j < num_frames; j++){
            if(temp[j] == -1){
                position = j;
                flag3 = 1;
                break;
            }
        }

        if(flag3 ==0){
            max = temp[0];
            position = 0;

            for(j = 1; j < num_frames; j++){
                if(temp[j] > max){
                    max = temp[j];
                    position = j;
                }
            }
        }

        frames[position] = inputStream[i];
        faults++;
    }

    printf("\n");


    for(j = 0; j < num_frames; j++){
        printf("%d\t", frames[j]);
    }
}
```

```
    printf("\n\nTotal Page Faults = %d", faults);
    printf("\nTotal Hits = %d", num_pages-faults);

    return 0;
}
```

OUTPUT-



```
ex2@ilab-HP-Desktop-Pro-G2:~$ ./a.out

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

Total Page Faults = 9
Total Hits = 11ex2@ilab-HP-Desktop-Pro-G2:~$
```