Operating Systems

Assignment 11

- IMPLEMENTATION OF PAGE REPLACEMENT ALGORITHMS

    - First Come First Serve (FCFS) Page Replacement Algorithm

Input:

```c
#include "stdio.h"
#include <conio.h>
int fsize;
int frm[15];
void display();
int main()
{
  int pg[100], nPage, i, j, pf = 0, top = -1, temp, flag =
  printf("\n Enter frame size:");
  scanf("%d", &fsize);
  printf("\n Enter number of pages:");
  scanf("%d", &nPage);

  for (i = 0; i < nPage; i++)
  {
    printf("\n Enter page[%d]:", i + 1);
    scanf("%d", &pg[i]);
  }
  for (i = 0; i < fsize; i++)
    frm[i] = -1;
  printf("\n page | \t Frame content ");
  printf("\n--------------------------------------");
  for (j = 0; j < nPage; j++)
  {
    flag = 0;
    for (i = 0; i < fsize; i++)
    {
      if (frm[i] == pg[j])
      {
        flag = 1;
        break;
      }
    }

    if (flag == 0)
    {
      if (top == fsize - 1)
      {
        top = -1;
      }
      pf++;
      top++;
      frm[top] = pg[j];
    }
    printf("\n %d    |", pg[j]);
    display();
  }
  printf("\n--------------------------------------");
  printf("\n total page fault:%d", pf);
  getch();
}
void display()
{
  int i;
  for (i = 0; i < fsize; i++)
    printf("\t %d", frm[i]);
}
```

Output:

```
Enter number of pages:12

Enter page[1]:1

Enter page[2]:2

Enter page[3]:3

Enter page[4]:4

Enter page[5]:5

Enter page[6]:1

Enter page[7]:2

Enter page[8]:3

Enter page[9]:4

Enter page[10]:5

Enter page[11]:6

Enter page[12]:1

page |          Frame content
-------------------------------------
1    |  1        -1       -1       -1
2    |  1        2        -1       -1
3    |  1        2        3        -1
4    |  1        2        3        4
5    |  5        2        3        4
1    |  5        1        3        4
2    |  5        1        2        4
3    |  5        1        2        3
4    |  4        1        2        3
5    |  4        5        2        3
6    |  4        5        6        3
1    |  4        5        6        1
-------------------------------------
total page fault:12
```

- Least Recently Used (LRU) Page Replacement Algorithm

- **Input:**

```c
#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",pagefault);

    return 0;
}
```

Output:

```
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
```

Submitted By:

Harshita Pasupuleti

21BCE8421