College Of Engineering Trivandrum

System Software Lab

Mohammed Rabeeh
S5 CSE Roll No:35

TVE18CS036

Department of Computer Science

September 9, 2020

Contents

CS331 - System Software · 2020 ·

Cycle 1

Exp No 3

1 Aim

Simulate the following page replacement algorithms.

(a) FIFO

(b) LRU

(c) LFU

2 Algorithms

2.1 FIFO Page Replacement

1. Start traversing the pages.
    - If set holds less pages than capacity.
        - Insert page into the set one by one until the size of set reaches capacity or all page requests are processed.
        - Simultaneously maintain the pages in the queue to perform FIFO.
        - Increment page fault
    - Else, If current page is present in set, do nothing.
    - Else
        - Remove the first page from the queue as it was the first to be entered in the memory
        - Replace the first page in the queue with the current page in the string.
        - Store current page in the queue.
        - Increment page faults.
2. Return page faults.

2.2 LRU Page Replacement

1. Start traversing the pages.
    - If set holds less pages than capacity.
        - Insert page into the set one by one until the size of set reaches capacity or all page requests are processed.
        - Simultaneously maintain the recent occurred index of each page in a map called indexes
        - Increment page fault
    - Else, If current page is present in set, do nothing.
    - Else
        - 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.
        - Replace the found page with current page.
        - Increment page faults.
        - Update index of current page.
2. Return page faults.

2.3 LFU Page Replacement

1. Start traversing the pages.
    - If set holds less pages than capacity.
        - Insert page into the set one by one until the size of set reaches capacity or all page requests are processed.
        - Simultaneously maintain the recent occurred index of each page in a map called indexes
        - Increment page fault
    - Else, If current page is present in set, do nothing.
    - Else
        - 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.
        - Replace the found page with current page.
        - Increment page faults.
        - Update index of current page.
2. Return page faults.

3 Program Code

3.1 FIFO Page Replacement

```
#include <stdio.h>

void fifo(int[], int[], int, int);

int main()
{
  int i, pCount, fCount, pages[30], frames[20];
  printf("Number of Frames : ");
  scanf("%d", &fCount);

  for (i = 0; i < fCount; ++i)
  {
    frames[i] = -1;
  }
  printf("Number of Pages : ");
  scanf("%d", &pCount);
  printf("Enter the reference string\n");
  for (i = 0; i < pCount; ++i)
  {
    scanf("%d", &pages[i]);
  }

  fifo(pages, frames, pCount, fCount);

  return 0;
}

void fifo(int pages[], int frames[], int pCount, int fCount)
{
  printf("\nRef.String   |\tFrames\n");
  printf("------------------------------\n");
  int i, j, k, flag, faultCount = 0, queue = 0;
  for (i = 0; i < pCount; ++i)
  {
    printf("  %d\t|\t", pages[i]);
    flag = 0;
    for (j = 0; j < fCount; ++j)
```

```
    {
      if (frames[j] == pages[i])
      {
        flag = 1;
        printf("   Hit");
        break;
      }
    }
    if (flag == 0)
    {
      frames[queue] = pages[i];
      faultCount++;
      queue = (queue + 1) % fCount;

      for (k = 0; k < fCount; ++k)
      {
        printf("%d  ", frames[k]);
      }
    }
    printf("\n\n");
  }
  printf("Total Page Faults = %d\n", faultCount);
}
```

3.2 LRU Page Replacement

```
#include <stdio.h>

void LRU(int[], int[], int[], int, int);
int findLRU(int[], int);

int main()
{
  int i, pCount, fCount, pages[30], frames[20], time[20];
  printf("Number of Frames : ");
  scanf("%d", &fCount);

  for (i = 0; i < fCount; ++i)
  {
    frames[i] = -1;
  }
  printf("Number of Pages : ");
  scanf("%d", &pCount);
  printf("Enter the reference string\n");
  for (i = 0; i < pCount; ++i)
  {
    scanf("%d", &pages[i]);
  }
  LRU(pages, frames, time, fCount, pCount);

  return 0;
}

void LRU(int pages[], int frames[], int time[], int fCount, int pCount)
{
  printf("\nRef.String   |\tFrames\n");
  printf("-------------------------------\n");

  int i, j, k, pos, flag, faultCount, counter, queue;
  counter = 0, queue = 0, faultCount = 0;
```

```
  for (i = 0; i < pCount; ++i)
  {
    flag = 0;
    printf("  %d\t|\t", pages[i]);

    for (j = 0; j < fCount; ++j)
    {
      if (frames[j] == pages[i])
      {
        flag = 1;
        counter++;
        time[j] = counter;
        printf("   Hit\n\n");
        break;
      }
    }

    if ((flag == 0) && (queue < fCount))
    {
      faultCount++;
      counter++;
      frames[queue] = pages[i];
      time[queue] = counter;
      queue++;
    }

    else if ((flag == 0) && (queue == fCount))
    {
      faultCount++;
      counter++;
      pos = findLRU(time, fCount);
      frames[pos] = pages[i];
      time[pos] = counter;
    }

    if (flag == 0)
    {
      for (k = 0; k < fCount; ++k)
      {
        printf("%d  ", frames[k]);
      }
      printf("\n\n");
    }
  }
  printf("Total Page Faults = %d\n\n", faultCount);
}

int findLRU(int time[], int fCount)
{
  int k, min, pos;
  pos = 0;
  min = time[0];
  for (k = 1; k < fCount; ++k)
  {
    if (time[k] < min)
    {
      min = time[k];
      pos = k;
    }
  }
  return pos;
```

```
}
```

3.3 LFU Page Replacement

```
#include <stdio.h>
int main()
{
  int q[20], p[50], c = 0, c1, d, f, i, j, k = 0, n, r, t, b[20], c2[20];
  printf("Enter no of pages: ");
  scanf("%d", &n);
  printf("Enter the reference string: ");
  for (i = 0; i < n; i++)
    scanf("%d", &p[i]);
  printf("Enter 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("\nThe no of page faults is %d\n", c);

  return 0;
}
```

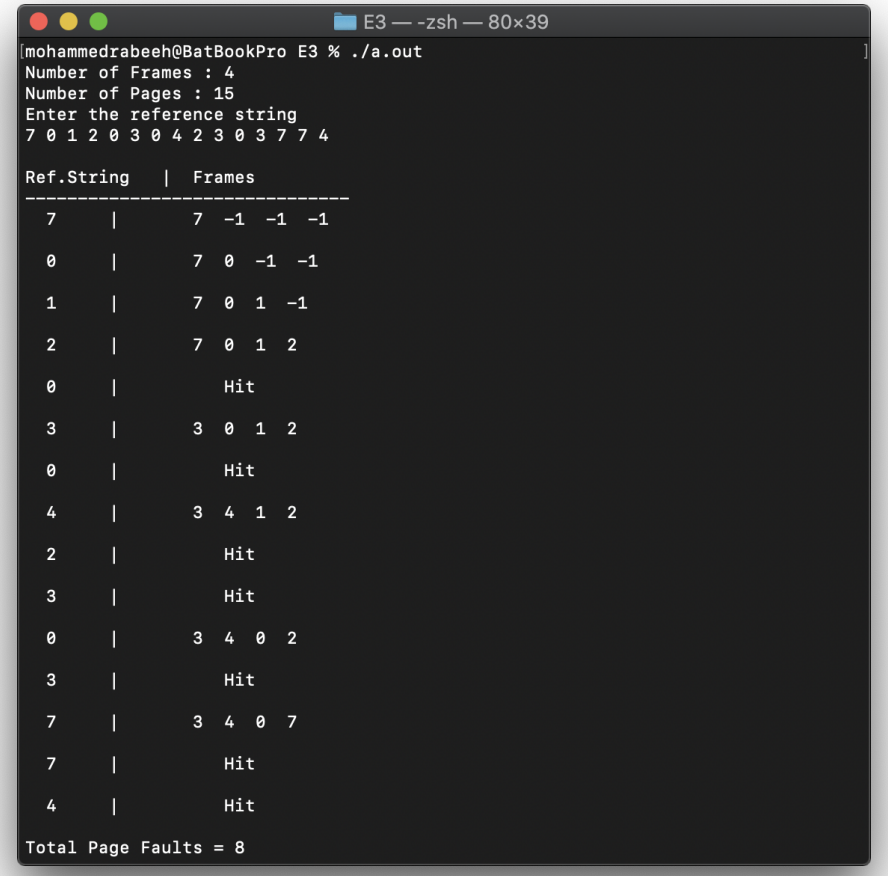
4 Program Output

Input

```
No of pages: 15
Reference string: 7 0 1 2 0 3 0 4 2 3 0 3 7 7 4
No of frames: 4
```

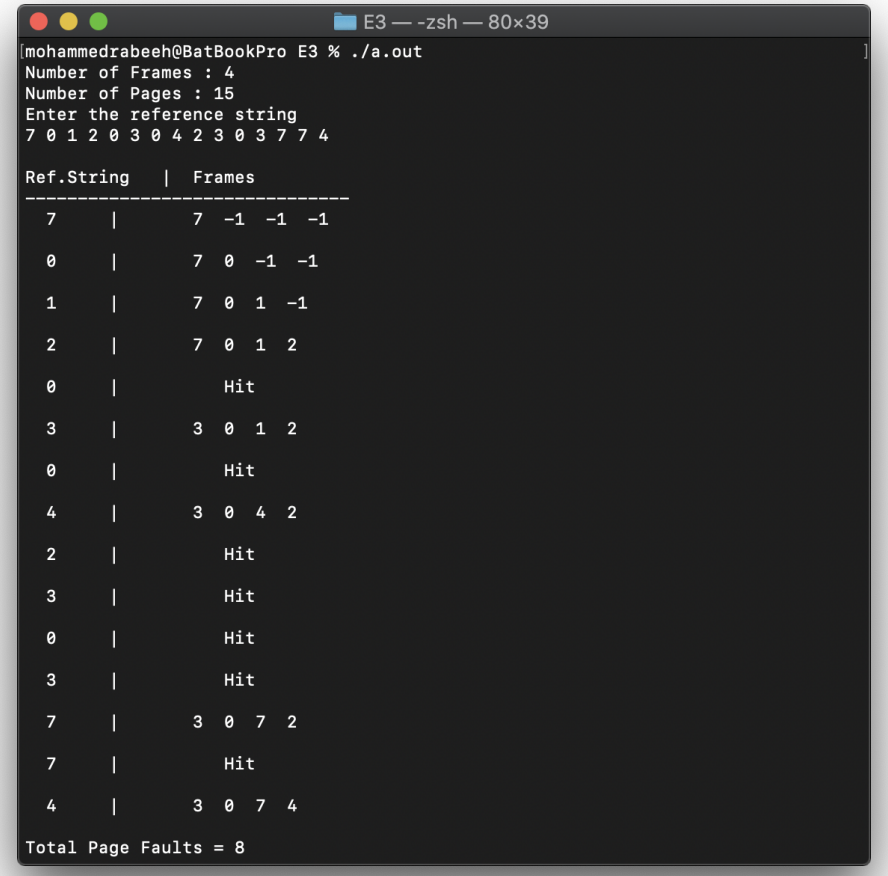
4.1 FIFO Page Replacement

Output



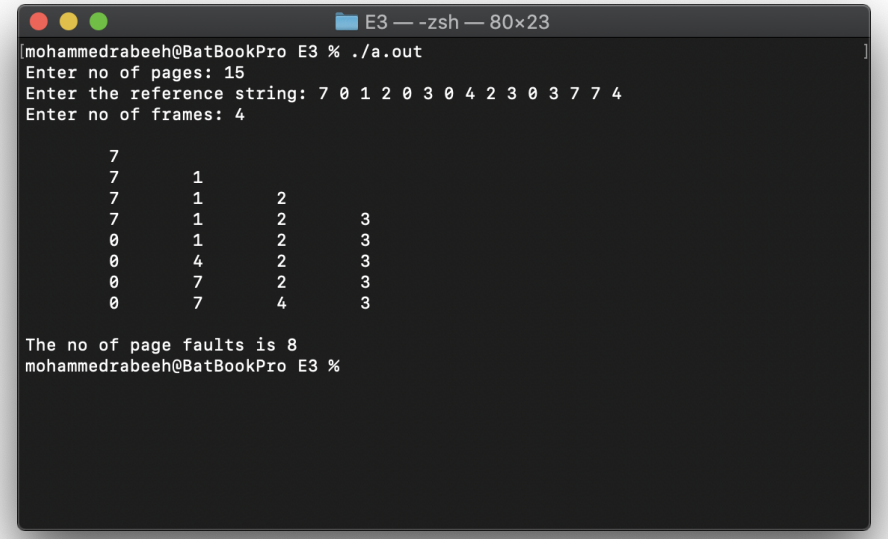
4.2 LRU Page Replacement

Output



4.3 LFU Page Replacement

Output



5 Result

The above mentioned page replacement algorithms, ie, FIFO, LRU, and LFU were implemented in C Language and their Output verified. The program was executed on macOS Catalina 10.15.3 operating system using ZSH Shell.