Exp. No. 13a **FIFO Page Replacement**

Date:

Aim

To implement demand paging for a reference string using FIFO method.

FIFO

- Page replacement is based on when the page was brought into memory.
- When a page should be replaced, the oldest one is chosen.
- Generally, implemented using a FIFO queue.
- Simple to implement, but not efficient.
- Results in more page faults.
- The page-fault may increase, even if frame size is increased (Belady's anomaly)

Algorithm

```
1. Get length of the reference string, say l.
2. Get reference string and store it in an array, say rs.
3. Get number of frames, say nf.
4. Initalize frame array upto length nf to -1.
5. Initialize position of the oldest page, say j to 0.
6. Initialize no. of page faults, say count to 0.
7. For each page in reference string in the given order, examine:
     a. Check whether page exist in the frame array
     b. If it does not exist then
          i. Replace page in position j.
          ii. Compute page replacement position as (j+1) modulus nf.
          iii. Increment count by 1.
          iv. Display pages in frame array.
8. Print count.
9. Stop
```

Program

```
#include <stdio.h>

main()
{
   int i,j,l,rs[50],frame[10],nf,k,avail,count=0;

   printf("Enter length of ref. string : ");
   scanf("%d", &l);
   printf("Enter reference string :\n");
   for(i=1; i<=l; i++)
      scanf("%d", &rs[i]);
   printf("Enter number of frames : ");
   scanf("%d", &nf);
   for(i=0; i<nf; i++)
      frame[i] = -1;
   j = 0;
   printf("\nRef. str  Page frames");
   for(i=1; i<=l; i++)
   {
      printf("\n%4d\t", rs[i]);
      avail = 0;
      for(k=0; k<nf; k++)
         if(frame[k] == rs[i])
            avail = 1;
      if(avail == 0)
      {
         frame[j] = rs[i];
         j = (j+1) % nf;
         count++;
         for(k=0; k<nf; k++)
            printf("%4d", frame[k]);
      }
   }
   printf("\n\nTotal no. of page faults : %d\n",count);
}
```

Output

```
Enter length of ref. string : 20
Enter reference string :
1 2 3 4 2 1 5 6 2 1 2 3 7 6 3
Enter number of frames : 5

Ref. str  Page frames
   1       1  -1  -1  -1  -1
   2       1   2  -1  -1  -1
   3       1   2   3  -1  -1
   4       1   2   3   4  -1
   2
   1
   5       1   2   3   4   5
   6       6   2   3   4   5
   2
   1       6   1   3   4   5
   2       6   1   2   4   5
   3       6   1   2   3   5
   7       6   1   2   3   7
   6
   3
Total no. of page faults : 10
```

Result

Thus page replacement was implemented using FIFO algorithm.

Exp. No. 13b LRU Page Replacement

Aim

To implement demand paging for a reference string using LRU method.

LRU

- **Pages used in the recent past are used as an approximation of future usage.**
- **The page that has not been used for a longer period of time is replaced.**
- **LRU is efficient but not optimal.**
- **Implementation of LRU requires hardware support, such as counters/stack.**

Algorithm

```
1.  Get length of the reference string, say len.
2.  Get reference string and store it in an array, say rs.
3.  Get number of frames, say nf.
4.  Create access array to store counter that indicates a measure of recent usage.
5.  Create a function arrmin that returns position of minimum of the given array.
6.  Initalize frame array upto length nf to -1.
7.  Initialize position of the page replacement, say j to 0.
8.  Initialize freq to 0 to track page frequency
9.  Initialize no. of page faults, say count to 0.
10. For each page in reference string in the given order, examine:
      a. Check whether page exist in the frame array.
      b. If page exist in memory then
            i. Store incremented freq for that page position in access array.
      c. If page does not exist in memory then
            i. Check for any empty frames.
           ii. If there is an empty frame,
                  ➢ Assign that frame to the page
                  ➢ Store incremented freq for that page position in access array.
                  ➢ Increment count.
          iii. If there is no free frame then
                  ➢ Determine page to be replaced using arrmin function.
                  ➢ Store incremented freq for that page position in access array.
                  ➢ Increment count.
           iv. Display pages in frame array.
11. Print count.
12. Stop
```

Program

```
/* LRU page replacement - lrupr.c */

#include <stdio.h>
```

```
int arrmin(int[], int);

main()
{
   int i,j,len,rs[50],frame[10],nf,k,avail,count=0;
   int access[10], freq=0, dm;

   printf("Length of Reference string : ");
   scanf("%d", &len);
   printf("Enter reference string :\n");
   for(i=1; i<=len; i++)
      scanf("%d", &rs[i]);
   printf("Enter no. of frames : ");
   scanf("%d", &nf);

   for(i=0; i<nf; i++)
      frame[i] = -1;
   j = 0;

   printf("\nRef. str  Page frames");
   for(i=1; i<=len; i++)
   {
      printf("\n%4d\t", rs[i]);
      avail = 0;
      for(k=0; k<nf; k++)
      {
         if(frame[k] == rs[i])
         {
            avail = 1;
            access[k] = ++freq;
            break;
         }
      }
      if(avail == 0)
      {
         dm = 0;
         for(k=0; k<nf; k++)
         {
            if(frame[k] == -1)
               dm = 1;
               break;
         }
         if(dm == 1)
         {
            frame[k] = rs[i];
            access[k] = ++freq;
            count++;
         }
         else
         {
            j = arrmin(access, nf);
            frame[j] = rs[i];
            access[j] = ++freq;
            count++;
         }
         for(k=0; k<nf; k++)
            printf("%4d", frame[k]);
      }
   }
   printf("\n\nTotal no. of page faults : %d\n", count);
}

int arrmin(int a[], int n)
{
   int i, min = a[0];
   for(i=1; i<n; i++)
      if (min > a[i])
         min = a[i];
   for(i=0; i<n; i++)
      if (min == a[i])
         return i;
}
```

Output

```
Length of Reference string : 15
Enter reference string :
1 2 3 4 2 1 5 6 2 1 2 3 7 6 3
Enter no. of frames : 5
Ref. str  Page frames
   1        1  -1  -1  -1  -1
   2        1   2  -1  -1  -1
   3        1   2   3  -1  -1
   4        1   2   3   4  -1
   2
   1
   5        1   2   3   4   5
   6        1   2   6   4   5
   2
   1
   2
   3        1   2   6   3   5
   7        1   2   6   3   7
   6
   3
Total no. of page faults : 8
```

Result

Thus page replacement was implemented using LRU algorithm.