17BCE0581
SATYAM SINGH CHAUHAN

OPERATING SYSTEMS
CSE2005

AIM:
Paging and Segmentation

Algorithm:

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

2. Return page faults.

LRU
Let capacity be the number of pages that
memory can hold. Let set be the current
set of pages in memory.

1- Start traversing the pages.
 i) If set holds less pages than capacity.
   a) Insert page into the set one by one until
      the size of set reaches capacity or all

page requests are processed.
b) Simultaneously maintain the recent occurred index of each page in a map called indexes.
c) Increment page fault

ii) Else
If current page is present in set, do nothing.
Else
a) 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.
b) Replace the found page with current page.
c) Increment page faults.
d) Update index of current page.

2. Return page faults.

Optimal

1. If referred page is already present, increment hit count.
2. If not present, find if a page that is never referenced in future. If such a page exists, replace this page with new page. If no such page exists, find a page that is referenced farthest in future. Replace this page with new page.

CODE

```
#include<stdio.h>
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;

void getData()
{
    printf("\nEnter length of page reference sequence:");
    scanf("%d",&n);
    printf("\nEnter the page reference sequence:");
    for(i=0; i<n; i++)
        scanf("%d",&in[i]);
    printf("\nEnter no of frames:");
    scanf("%d",&nf);
}

void initialize()
{
    pgfaultcnt=0;
```

```
    for(i=0; i<nf; i++)
        p[i]=9999;
}

int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
}

int getHitIndex(int data)
{
    int hitind;
    for(k=0; k<nf; k++)
    {
        if(p[k]==data)
        {
            hitind=k;
            break;
        }
    }
    return hitind;
}

void dispPages()
{
    for (k=0; k<nf; k++)
    {
        if(p[k]!=9999)
            printf(" %d",p[k]);
    }

}

void dispPgFaultCnt()
{
    printf("\nTotal no of page faults:%d",pgfaultcnt);
}

void fifo()
{
    initialize();
```

```
    for(i=0; i<n; i++)
    {
        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(k=0; k<nf-1; k++)
                p[k]=p[k+1];

            p[k]=in[i];
            pgfaultcnt++;
            dispPages();
        }
        else
            printf("No page fault");
    }
    dispPgFaultCnt();
}


void optimal()
{
    initialize();
    int near[50];
    for(i=0; i<n; i++)
    {

        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
            {
                int pg=p[j];
                int found=0;
                for(k=i; k<n; k++)
                {
                    if(pg==in[k])
                    {
                        near[j]=k;
                        found=1;
                        break;
                    }
                    else
                        found=0;
                }
                if(!found)
                    near[j]=9999;
            }
            int max=-9999;
```

```
            int repindex;
            for(j=0; j<nf; j++)
            {
                if(near[j]>max)
                {
                    max=near[j];
                    repindex=j;
                }
            }
            p[repindex]=in[i];
            pgfaultcnt++;

            dispPages();
        }
        else
            printf("No page fault");
    }
    dispPgFaultCnt();
}

void lru()
{
    initialize();

    int least[50];
    for(i=0; i<n; i++)
    {

        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
            {
                int pg=p[j];
                int found=0;
                for(k=i-1; k>=0; k--)
                {
                    if(pg==in[k])
                    {
                        least[j]=k;
                        found=1;
                        break;
                    }
                    else
                        found=0;
                }
                if(!found)
                    least[j]=-9999;
            }
            int min=9999;
```

```
            int repindex;
            for(j=0; j<nf; j++)
            {
                if(least[j]<min)
                {
                    min=least[j];
                    repindex=j;
                }
            }
            p[repindex]=in[i];
            pgfaultcnt++;

            dispPages();
        }
        else
            printf("No page fault!");
    }
    dispPgFaultCnt();
}

int main()
{
    int choice;
    while(1)
    {
        printf("\nPage Replacement Algorithms\n1.Enter
data\n2.FIFO\n3.Optimal\n4.LRU\n5.Exit\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            getData();
            break;
        case 2:
            fifo();
            break;
        case 3:
            optimal();
            break;
        case 4:
            lru();
            break;
        default:
            return 0;
            break;
        }
    }
}
```

OUTPUT

```
17bce0581@sjt419scs066:~$ vi pagereplacement.c
17bce0581@sjt419scs066:~$ cc pagereplacement.c
17bce0581@sjt419scs066:~$ ./a.out

Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.Exit
Enter your choice:1

Enter length of page reference sequence:15

Enter the page reference sequence:0
2
1
4
3
3
6
4
0
1
0
3
1
2
1

Enter no of frames:3
```

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.Exit
Enter your choice:2

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 4 : 2 1 4
For 3 : 1 4 3
For 3 :No page fault
For 6 : 4 3 6
For 4 :No page fault
For 0 : 3 6 0
For 1 : 6 0 1
For 0 :No page fault
For 3 : 0 1 3
For 1 :No page fault
For 2 : 1 3 2
For 1 :No page fault
Total no of page faults:10
```

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.Exit
Enter your choice:3

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 4 : 0 4 1
For 3 : 0 4 3
For 3 :No page fault
For 6 : 0 4 6
For 4 :No page fault
For 0 :No page fault
For 1 : 0 1 6
For 0 :No page fault
For 3 : 3 1 6
For 1 :No page fault
For 2 : 2 1 6
For 1 :No page fault
Total no of page faults:9
```

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.Exit
Enter your choice:4

For 0 : 0
For 2 : 0 2
For 1 : 0 2 1
For 4 : 4 2 1
For 3 : 4 3 1
For 3 :No page fault!
For 6 : 4 3 6
For 4 :No page fault!
For 0 : 4 0 6
For 1 : 4 0 1
For 0 :No page fault!
For 3 : 3 0 1
For 1 :No page fault!
For 2 : 3 2 1
For 1 :No page fault!
Total no of page faults:10
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.Exit
Enter your choice:5
17bce0581@sjt419scs066:~$
```