

Ex. No.: 11a)

Date: 12.04.25

FIFO PAGE REPLACEMENT

Aim:

To find out the number of page faults that occur using First-in First-out (FIFO) page replacement technique.

Algorithm:

1. Declare the size with respect to page length
 2. Check the need of replacement from the page to memory
 3. Check the need of replacement from old page to new page in memory 4.
- Form a queue to hold all pages
5. Insert the page require memory into the queue
 6. Check for bad replacement and page fault
 7. Get the number of processes to be inserted
 8. Display the values

Program Code:

```
#include <stdio.h>
int main() {
    int refstr[100], frames[10];
    int refsize, framesize;
    int index = 0, i, j, pf = 0;
    printf("Enter size of ref string: ");
    scanf("%d", &refsize);
    for (int i = 0; i < refsize; i++) {
        printf("Enter [%d]: ", i);
        scanf("%d", &refstr[i]);
    }
```

```

    printf("Enter page frame size: ");
    scanf("%d", &framesize);
    for (int i = 0; i < refsize; i++) {
        int it = 0;
```

```

for (int j=0; j < framesize; j++) {
    if (frames[j] == refstr[i]) {
        istit = 1;
        break;
    }
}

if (!istit) {
    frames[index] = refstr[i];
    index = (index + 1) % framesize;
    p++;
    printf("%d → ", refstr[i]);
    for (int k=0; k < framesize; k++) {
        if (frames[k] != -1)
            printf("%d", frames[k]);
    }
    printf("\n");
}
else {
    printf("%d → No page faults in",
           refstr[i]);
}

printf("\n Total page faults: %d\n", pf);
}

```

Sample Output:

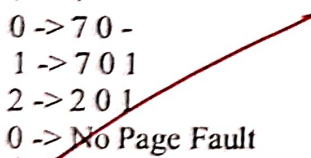
```
[root@localhost student]# python fifo.py
```

Enter the size of reference string: 20

```
Enter [ 1] : 7
Enter [ 2] : 0
Enter [ 3] : 1
Enter [ 4] : 2
Enter [ 5] : 0
Enter [ 6] : 3
Enter [ 7] : 0
Enter [ 8] : 4
Enter [ 9] : 2
Enter [10] : 3
Enter [11] : 0
Enter [12] : 3
Enter [13] : 2
Enter [14] : 1
Enter [15] : 2
Enter [16] : 0
Enter [17] : 1
Enter [18] : 7
Enter [19] : 0
Enter [20] : 1
```

Enter page frame size : 3

```
7 -> 7 - -
0 -> 7 0 -
1 -> 7 0 1
2 -> 2 0 1
0 -> No Page Fault
```



```
3 -> 2 3 1
0 -> 2 3 0
4 -> 4 3 0
2 -> 4 2 0
3 -> 4 2 3
0 -> 0 2 3
3 -> No Page Fault
2 -> No Page Fault
1 -> 0 1 3
2 -> 0 1 2
0 -> No Page Fault
1 -> No Page Fault
7 -> 7 1 2
0 -> 7 0 2
```

1 -> 7 0 1

Total page faults: 15.

[root@localhost student]#

Enter the size of ref string: 7

Enter page frame size: 3

Enter [1]: 1

Enter [2]: 3

Enter [3]: 0

Enter [4]: 3

Enter [5]: 5

Enter [6]: 6

Enter [7]: 3

1 -> 1

3 -> 1 3

0 -> 1 3 0

No page fault

5 -> 5 3 0

6 -> 5 6 0

3 -> 5 6 3

total page faults: 6

Result :

A program for finding the page fault using FIFO replacement has been executed successfully

Ex. No.: 11b)

Date: 18-04-25

LRU

Aim:

To write a c program to implement LRU page replacement algorithm.

Algorithm:

- 1: Start the process
- 2: Declare the size
- 3: Get the number of pages to be inserted
- 4: Get the value
- 5: Declare counter and stack
- 6: Select the least recently used page by counter value
- 7: Stack them according the selection.
- 8: Display the values
- 9: Stop the process

Program Code:

```
#include <stdio.h>
int main() {
    int refst[100], frames[20], recent[20];
    int rsize, framesize;
    int i, j, k, time = 0, pf = 0, issu, index;
    printf("Enter the number of pages: ");
    scanf("%d", &rsize);
    for (int i = 0; i < rsize; i++) {
        printf(" [ %d ]": i+1);
        scanf("%d", &refst[i]);
    }
    printf("Enter page frame size: ");
    scanf("%d", &framesize);
    for (i = 0; i < framesize; i++) {
        frame[i] = -1;
        recent[i] = -1;
    }
}
```

```
printf("\n");
```

```
for (int i = 0; i < refsize; i++) {
```

```
    isHit = 0;
```

```
    for (int j = 0; j < framesize; j++) {
```

```
        if (frames[j] == refstr[i]) {
```

```
            isHit = 1;
```

```
            recent[j] = time++;
```

```
            break;
```

```
        }
```

```
    } if (isHit) {
```

```
        printf("Hit No page fault\n", refstr[i]);
```

```
        continue;
```

```
    }
```

```
    int empty ind = -1;
```

```
    for (j = 0; j < framesize; j++) {
```

```
        if (frames[j] == -1) {
```

```
            empty ind = j;
```

```
            break;
```

```
        }
```

```
    }
```

```
    if (empty ind != -1) {
```

```
        frames[empty ind] = ref[i];
```

```
        recent[empty ind] = time++;
```

```
    } else {
```

```
        int min = recent[0];
```

```
        trindex = 0;
```

```
        for (j = 1; j < framesize; j++) {
```

```
            if (recent[j] < min) {
```



```

        min = recent[j];
        tindex = j;
    }
    frames[tindex] = refstr[i];
    recent[tindex] = time + 1;

}
pf++;
printf("%d → ", refstr[i]);
for(int k=0; k < framesize; k++) {
    if (frames[k] != -1)
        printf("%d", frames[k]);

}
printf(" ⇒ page fault\n");

}
printf("\n Total page faults : %d\n", pf);
}

```

output

Enter number of page : 14

Enter[1]=7
 Enter[2]=0
 Enter[3]=1
 Enter[4]=2
 Enter[5]=0
 Enter[6]=3
 Enter[7]=0
 Enter[8]=4
 Enter[9]=2
 Enter[10]=3
 Enter[11]=0
 Enter[12]=3
 Enter[13]=2
 Enter[14]=3

Enter page frame

7 → 7 → page fault
 0 → 7 0 ⇒ page fault
 1 → 7 0 1 ⇒ page fault
 2 → 7 0 1 2 ⇒ page fault
 0 → No page fault
 4 → 8 0 4 2 ⇒ page fault
 2 ⇒ No page fault
 3 ⇒ No page fault
 0 ⇒ No page fault
 3 ⇒ No page fault
 2 ⇒ No page fault
 3 ⇒ No page fault
 Total page faults : 6

Sample Output :

Enter number of frames: 3

Enter number of pages: 6

Enter reference string: 5 7 5 6 7 3

5 -1 -1

5 7 -1

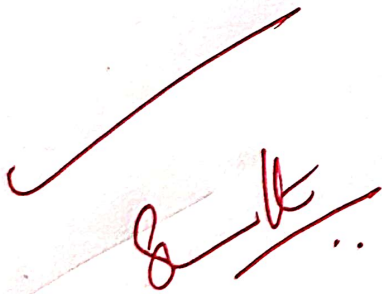
5 7 -1

5 7 6

5 7 6

3 7 6

Total Page Faults = 4

A handwritten signature in red ink, consisting of a stylized 'S' followed by 'K' and a double underline, with a large checkmark above it.

Result:

A C program for finding the page fault using LRU page replacement technique is implemented successfully.

Ex. No.: 11c)

Date: 18.04.25

Optimal

Aim:

To write a c program to implement Optimal page replacement algorithm.

ALGORITHM:

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least frequently used page by counter value
7. Stack them according to the selection.
8. Display the values
9. Stop the process

PROGRAM:

```
#include <stdio.h>
int main() {
    int refstr[100], frames[10];
    int n, p, i, j, k, pageFaults = 0, hit;
    printf("Enter the size of reference string: ");
    scanf("%d", &n);
    for(i = 0; i < n; i++) {
        printf("Enter s%d: ", i+1);
        scanf("%d", &refstr[i]);
    }

    printf("Enter page frame size: ");
    scanf("%d", &p);
```

```
for (i=0; i < F; i++)
```

```
frames[i] = -1;
```

```
printf("\n");
```

```
for (i=0; i < n; i++) {
```

```
    hit = 0;
```

```
    for (j=0; j < F; j++) {
```

```
        if (frames[j] == refstr[i]) {
```

```
            hit = 1;
```

```
            break;
```

```
        }
```

```
    } if (hit) {
```

```
        printf("y. 2d -> NO page fault\n", refstr[i]);
```

```
        continue;
```

```
    }
```

```
    int empty = -1;
```

```
    for (j=0; j < F; j++) {
```

```
        if (frames[j] == -1) {
```

```
            empty = j;
```

```
            break;
```

```
        }
```

```
    }
```

```
    if (empty != -1) {
```

```
        frames[empty] = refstr[i];
```

```
    } else {
```

```
        int farthest = -1, index = -1;
```

```
        for (j=0; j < F; j++) {
```

```
            int found = 0;
```

```
            for (k=i+1; k < n; k++) {
```

```
                if (frames[j] == refstr[k]) {
```

```
                    74
```

```
                    found = 1;
```

```
                    if (k > farthest) {
```

```
                        farthest = k;
```

```
                        index = j;
```

```
                    }
```

```
                }
```

```
} frames[idx] = refs[i];
```

```
}
```

```
pageFaults++;
```

```
printf("%2d → ", refs[i]);
```

```
for (k=0; k<P; k++) {
```

```
    if (frames[k] != -1)
```

```
        printf("%d ", frames[k]);
```

```
}
```

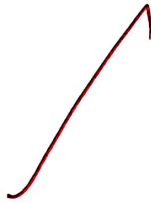
```
printf(" ⇒ Page Fault\n");
```

```
}
```

```
printf("In total page faults: %d\n", PageFaults);
```

```
printf("Total page hits: %d\n", n - pageFaults);
```

```
}
```



Output:

Enter the size of reference string = 10

Enter page frame size: 3

Enter [1]: 7

7 → 7 ⇒ page fault

Enter [2]: 0

0 → 70 ⇒ page fault

Enter [3]: 1

1 → 701 ⇒ page fault

Enter [4]: 2

2 → 201 ⇒ page fault

Enter [5]: 0

0 → No page

Enter [6]: 3

3 → 203 ⇒ No page fault

Enter [7]: 0

4 → 243 ⇒ page fault

Enter [8]: 4

2 → No page fault

Enter [9]: 2

3 → No page fault

Enter [10]: 3

Total page faults: 6

Total page hit: 4

Result:

A C program for finding the page fault using optimal page replacement technique is implemented successfully