

32. Construct a C program to simulate the Least Recently Used paging technique of memory management.

AIM

To construct a C program that simulates the Least Recently Used (LRU) paging technique of memory management, which replaces the page that has not been used for the longest time when a new page needs to be loaded, and all frames are full.

PROCEDURE

- 1. Start**
2. Input the number of pages, the sequence of page references, and the number of frames.
3. Initialize the frames with -1 (empty), and set the page fault counter to 0.
4. For each page reference:
 - Check if the page is already present in one of the frames.
 - If it is found, move to the next page (no page fault).
 - If it is not found, increment the page fault counter.
 - If there is space in the frames, place the page in an empty frame.
 - If all frames are full, find the least recently used page (the one that hasn't been used for the longest time) and replace it with the new page.
5. Display the status of the frames after each page reference and the total number of page faults at the end.
- 6. Stop**

CODE:

```
#include <stdio.h>
```

```
void lruPaging(int pages[], int n, int frames[], int f) {  
    int pageFaults = 0, i, j, found, min, minIndex;
```

```
    printf("Page Reference\tFrames\n");
```

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

```
        found = 0;
```

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

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

```
                found = 1;
```

```
                break;
```

```
            }
```

```
        }
```

```
        if (!found) {
```

```
            if (pageFaults < f) {
```

```
                frames[pageFaults] = pages[i];
```

```
            } else {
```

```
                min = 9999;
```

```

        for (j = 0; j < f; j++) {
            int usageCount = 0;
            for (int k = i - 1; k >= 0; k--) {
                if (pages[k] == frames[j]) {
                    usageCount = i - k;
                    break;
                }
            }
            if (usageCount < min) {
                min = usageCount;
                minIndex = j;
            }
        }
        frames[minIndex] = pages[i];
    }
    pageFaults++;
}

printf("%d\t\t", pages[i]);
for (j = 0; j < f; j++) {
    if (frames[j] != -1) {
        printf("%d ", frames[j]);
    } else {
        printf("- ");
    }
}
printf("\n");
}

printf("Total Page Faults: %d\n", pageFaults);
}

int main() {
    int n, f, i;

    printf("Enter the number of pages: ");
    scanf("%d", &n);

    int pages[n];
    printf("Enter the page reference sequence: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }

    printf("Enter the number of frames: ");
    scanf("%d", &f);

```

```

int frames[f];
for (i = 0; i < f; i++) {
    frames[i] = -1;
}

lruPaging(pages, n, frames, f);

return 0;
}

```

OUTPUT:

```

Enter the number of pages: 12
Enter the page reference sequence: 11
10
9
8
7
6
5
3
4
2
1
12
Enter the number of frames: 3
Page Reference  Frames
11              11 - -
10              11 10 -
9               11 10 9
8               11 10 8
7               11 10 7
6               11 10 6
5               11 10 5
3               11 10 3
4               11 10 4
2               11 10 2
1               11 10 1
12              11 10 12
Total Page Faults: 12

...Program finished with exit code 0
Press ENTER to exit console.

```