**Write a C program to simulate page replacement algorithms.**

**a) FIFO**

**b) LRU**

**c) Optimal**

Program:

#include <stdio.h>

#include <stdlib.h>

#define EMPTY -1

void printFrames(int frame[], int capacity) {

for (int i = 0; i < capacity; i++) {

if (frame[i] == EMPTY)

printf("- ");

else

printf("%d ", frame[i]);

}

printf("\n");

}

int isInFrame(int frame[], int page, int capacity) {

for (int i = 0; i < capacity; i++) {

if (frame[i] == page)

return 1;

}

return 0;

}

int fifo(int pages[], int n, int capacity) {

int \*frame = (int \*)malloc(capacity \* sizeof(int));

for (int i = 0; i < capacity; i++) frame[i] = EMPTY;

int pageFaults = 0, pointer = 0;

printf("\n--- FIFO Page Replacement ---\n");

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

printf("Page %d: ", pages[i]);

if (!isInFrame(frame, pages[i], capacity)) {

frame[pointer] = pages[i];

pointer = (pointer + 1) % capacity;

pageFaults++;

}

printFrames(frame, capacity);

}

free(frame);

return pageFaults;

}

int lru(int pages[], int n, int capacity) {

int \*frame = (int \*)malloc(capacity \* sizeof(int));

int \*recent = (int \*)malloc(capacity \* sizeof(int));

for (int i = 0; i < capacity; i++) frame[i] = EMPTY;

int pageFaults = 0, count = 0;

printf("\n--- LRU Page Replacement ---\n");

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

printf("Page %d: ", pages[i]);

int pos = -1;

for (int j = 0; j < count; j++) {

if (frame[j] == pages[i]) {

pos = j;

break;

}

}

if (pos == -1) {

if (count < capacity) {

frame[count] = pages[i];

recent[count] = i;

count++;

} else {

int lru\_index = 0;

for (int j = 1; j < capacity; j++) {

if (recent[j] < recent[lru\_index])

lru\_index = j;

}

frame[lru\_index] = pages[i];

recent[lru\_index] = i;

}

pageFaults++;

} else {

recent[pos] = i;

}

printFrames(frame, capacity);

}

free(frame);

free(recent);

return pageFaults;

}

int optimal(int pages[], int n, int capacity) {

int \*frame = (int \*)malloc(capacity \* sizeof(int));

for (int i = 0; i < capacity; i++) frame[i] = EMPTY;

int pageFaults = 0, count = 0;

printf("\n--- Optimal Page Replacement ---\n");

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

printf("Page %d: ", pages[i]);

int found = isInFrame(frame, pages[i], capacity);

if (!found) {

if (count < capacity) {

frame[count++] = pages[i];

} else {

int farthest = -1, index = -1;

for (int j = 0; j < capacity; j++) {

int k;

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

if (frame[j] == pages[k]) break;

}

if (k > farthest) {

farthest = k;

index = j;

}

}

frame[index] = pages[i];

}

pageFaults++;

}

printFrames(frame, capacity);

}

free(frame);

return pageFaults;

}

int main() {

int choice, n, capacity;

printf("Enter the number of pages: ");

scanf("%d", &n);

int \*pages = (int \*)malloc(n \* sizeof(int));

printf("Enter the reference string (space-separated): ");

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

scanf("%d", &pages[i]);

}

printf("Enter the number of frames: ");

scanf("%d", &capacity);

do {

printf("\n--- Page Replacement Menu ---\n");

printf("1. FIFO\n");

printf("2. LRU\n");

printf("3. Optimal\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

int faults;

switch (choice) {

case 1:

faults = fifo(pages, n, capacity);

printf("Total Page Faults (FIFO): %d\n", faults);

break;

case 2:

faults = lru(pages, n, capacity);

printf("Total Page Faults (LRU): %d\n", faults);

break;

case 3:

faults = optimal(pages, n, capacity);

printf("Total Page Faults (Optimal): %d\n", faults);

break;

case 4:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

} while (choice != 4);

free(pages);

return 0;

}

Output:



