Week 8

#include <stdio.h>

#define MAX 100

// Function to check if page is present

int isPresent(int frames[], int page, int n) {

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

if (frames[i] == page)

return 1;

return 0;

}

// FIFO Page Replacement

void FIFO(int pages[], int n, int capacity) {

int frames[capacity], front = 0, count = 0;

for (int i = 0; i < capacity; i++) frames[i] = -1; // initialize frames

printf("\nFIFO Page Replacement Process:\n");

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

if (!isPresent(frames, pages[i], capacity)) {

frames[front] = pages[i];

front = (front + 1) % capacity;

count++;

printf("PF No. %d: ", count);

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

(frames[j] == -1) ? printf("- ") : printf("%d ", frames[j]);

printf("\n");

}

}

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

}

// LRU Page Replacement

void LRU(int pages[], int n, int capacity) {

int frames[capacity], recent[capacity], count = 0;

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

frames[i] = -1;

recent[i] = -1;

}

printf("\nLRU Page Replacement Process:\n");

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

int found = 0;

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

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

found = 1;

recent[j] = i; // update recent use

break;

}

}

if (!found) {

int replace = -1;

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

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

replace = j;

break;

}

}

if (replace == -1) {

// Find least recently used

int lru\_index = 0;

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

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

lru\_index = j;

}

replace = lru\_index;

}

frames[replace] = pages[i];

recent[replace] = i;

count++;

printf("PF No. %d: ", count);

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

(frames[j] == -1) ? printf("- ") : printf("%d ", frames[j]);

printf("\n");

}

}

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

}

// Optimal Page Replacement

void Optimal(int pages[], int n, int capacity) {

int frames[capacity], count = 0;

for (int i = 0; i < capacity; i++) frames[i] = -1;

printf("\nOptimal Page Replacement Process:\n");

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

if (isPresent(frames, pages[i], capacity)) {

continue; // No page fault

}

int replace = -1;

// First, check for empty frame

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

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

replace = j;

break;

}

}

if (replace == -1) {

int farthest = -1;

int index = -1;

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

int k;

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

if (pages[k] == frames[j])

break;

}

if (k > farthest) {

farthest = k;

index = j;

}

}

replace = index;

}

frames[replace] = pages[i];

count++;

printf("PF No. %d: ", count);

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

(frames[j] == -1) ? printf("- ") : printf("%d ", frames[j]);

printf("\n");

}

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

}

int main() {

int capacity, n, pages[MAX];

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

scanf("%d", &capacity);

printf("Enter the length of reference string: ");

scanf("%d", &n);

printf("Enter the reference string: ");

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

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

FIFO(pages, n, capacity);

LRU(pages, n, capacity);

Optimal(pages, n, capacity);

return 0;

}

