**TASK 4**

**Description**

This task involved the simulation and comparison of the 2 page replacement policies local and global, a page reference string was use to compare, this simulation showed the efficiency and effectiveness of different replacement algorithms in multiprogramming.

**Code**

**Local page replacement**

#include <stdio.h>  
//local page replacment  
int simulateLocalPageReplacement(int \*pageRefs, int length, int numFrames) {  
    int frames[numFrames];  
    for (int i = 0; i < numFrames; i++) frames[i] = -1; // Initialize frames to -1 (empty)  
     
    int pageFaults = 0;  
    int index = 0; // Pointer to keep track of the next frame to evict  
     
    for (int i = 0; i < length; i++) {  
        int page = pageRefs[i];  
        int found = 0;  
         
        // Check if the page is already in one of the frames  
        for (int j = 0; j < numFrames; j++) {  
            if (frames[j] == page) {  
                found = 1;  
                break;  
            }  
        }  
         
        if (!found) {  // Page fault occurred  
            pageFaults++;  
            frames[index] = page;  // Replace the page  
            index = (index + 1) % numFrames;  // Move to the next frame  
        }  
    }  
    return pageFaults;  
}  
  
int main() {  
    int pageRefs[] = {0, 1, 2, 3, 0, 1, 4, 0, 1, 2};  // Example page reference string  
    int length = sizeof(pageRefs) / sizeof(pageRefs[0]);  
  
    int pageFaults = simulateLocalPageReplacement(pageRefs, length, 3);  // 3 frames  
    printf("Local Page Faults: %d\n", pageFaults);  
  
    return 0;  
}

**Global page replacement**

#include <stdio.h>  
//global page replacment  
  
int simulateGlobalPageReplacement(int \*pageRefs1, int \*pageRefs2, int length, int numFrames) {  
    int frames[numFrames];  
    for (int i = 0; i < numFrames; i++) frames[i] = -1; // Initialize frames to -1 (empty)  
  
    int pageFaults = 0;  
    int index = 0; // Pointer to keep track of the next frame to evict  
     
    // Simulate global page replacement for both processes  
    for (int i = 0; i < length; i++) {  
        int page = (i % 2 == 0) ? pageRefs1[i / 2] : pageRefs2[i / 2]; // Alternate between two processes  
        int found = 0;  
         
        // Check if the page is already in one of the frames  
        for (int j = 0; j < numFrames; j++) {  
            if (frames[j] == page) {  
                found = 1;  
                break;  
            }  
        }  
  
        if (!found) {  // Page fault occurred  
            pageFaults++;  
            frames[index] = page;  // Replace the page  
            index = (index + 1) % numFrames;  // Move to the next frame  
        }  
    }  
    return pageFaults;  
}  
  
int main() {  
    int pageRefs1[] = {0, 1, 2, 3, 0, 1, 4, 0, 1, 2};  // Example page reference string for process 1  
    int pageRefs2[] = {1, 2, 3, 0, 1, 2, 4, 0, 1, 3};  // Example page reference string for process 2  
    int length = sizeof(pageRefs1) / sizeof(pageRefs1[0]);  
  
    int pageFaults = simulateGlobalPageReplacement(pageRefs1, pageRefs2, length, 5);  // 5 frames  
    printf("Global Page Faults: %d\n", pageFaults);  
  
    return 0;  
}

**Page reference string**

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
//page refrence string  
  
#define MAX\_PAGES 10  // Set to a small number for demonstration  
  
// Function to generate a page reference string based on a Markov model  
void generatePageReferenceString(int N, float \*probabilities, int \*pageRefs, int length) {  
    int currentPage = rand() % N;  // Random starting page  
    pageRefs[0] = currentPage;  
     
    for (int i = 1; i < length; i++) {  
        float randVal = (float)rand() / RAND\_MAX; // Random number between 0 and 1  
        if (randVal < probabilities[currentPage]) {  
            pageRefs[i] = currentPage; // Stay at the same page  
        } else {  
            pageRefs[i] = rand() % N; // Choose a new page  
        }  
        currentPage = pageRefs[i];  
    }  
}  
  
int main() {  
    srand(time(NULL)); // Seed for random number generation  
  
    int N = 4; // Number of states (pages)  
    int length = 20; // Length of the reference string  
    float probabilities[] = {0.7, 0.6, 0.8, 0.5}; // Probabilities of staying at the same page  
  
    int pageRefs[length];  
  
    generatePageReferenceString(N, probabilities, pageRefs, length);  
  
    // Print generated page reference string  
    printf("Generated Page Reference String:\n");  
    for (int i = 0; i < length; i++) {  
        printf("%d ", pageRefs[i]);  
    }  
    printf("\n");  
  
    return 0;  
}

**Git link**

[**https://github.com/FirasAhmed2/Operating-systems-coursework.git**](https://github.com/FirasAhmed2/Operating-systems-coursework.git)