**COMPUTER SCIENCE DEPARTMENT**

|  |
| --- |
| **Total Marks:** |
| **Obtained Marks:** |

Operating System Lab

**LAB TASK#15**

**Submitted To: Sir Jawad Naseer**

**Student Name: Ubaid Bin Waris**

**Reg Number: 2212416**

**COMPUTER SCIENCE DEPARTMENT**

**Question no.1**

**A virtual memory system divides the logical address space into pages and the physical memory into frames. You are given a list of logical page references made by a running process. Write a program to simulate the basic paging technique of memory management.**

1. **Take the number of pages and number of frames as input.**
2. **Map each page to a frame. If a frame is not available, show that the page cannot be loaded.**
3. **Display the final page-to-frame mapping.**
4. **A web browser keeps track of the last time each tab (page) was used. e. If new tabs exceed available memory, it closes the least recently used tab.**

**Simulate the LRU page replacement algorithm in C:**

* + **Accept the reference string and number of frames as input.**
  + **Replace the page that hasn’t been used for the longest time.**
  + **Display page faults and the state of memory after each access.**

**Code**

#include <stdio.h>

#define MAX\_FRAMES 10

#define MAX\_PAGES 100

int findLRU(int time[], int n) {

int i, minimum = time[0], pos = 0;

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

if(time[i] < minimum) {

minimum = time[i];

pos = i;

}

}

return pos;

}

int main() {

int frames[MAX\_FRAMES], time[MAX\_FRAMES];

int pages[MAX\_PAGES], n\_pages, n\_frames, i, j, k, pos, faults = 0, counter = 0, flag1, flag2;

printf("Enter number of pages: ");

scanf("%d", &n\_pages);

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

for(i = 0; i < n\_pages; ++i) {

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

}

printf("Enter number of frames: ");

scanf("%d", &n\_frames);

for(i = 0; i < n\_frames; ++i) {

frames[i] = -1;

}

printf("\nPage\tFrames\t\tPage Fault\n");

for(i = 0; i < n\_pages; ++i) {

flag1 = flag2 = 0;

// Check if page is already in frame

for(j = 0; j < n\_frames; ++j) {

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

counter++;

time[j] = counter; // Update last used time

flag1 = flag2 = 1;

break;

}

}

// Empty frame available

if(flag1 == 0) {

for(j = 0; j < n\_frames; ++j) {

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

counter++;

faults++;

frames[j] = pages[i];

time[j] = counter;

flag2 = 1;

break;

}

}

}

// No empty frame: apply LRU

if(flag2 == 0) {

pos = findLRU(time, n\_frames);

counter++;

faults++;

frames[pos] = pages[i];

time[pos] = counter;

}

// Display state of frames

printf("%d\t", pages[i]);

for(j = 0; j < n\_frames; ++j) {

if(frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

if(flag1 == 0)

printf("\tYes\n");

else

printf("\tNo\n");

}

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

return 0;

}

**Output**

A screen shot of a computer

AI-generated content may be incorrect.

**Question no.2**

**A system has limited cache size and wants to evict the page that is least frequently used to free up space. Write a program to:**

* 1. **Simulate LFU page replacement**
  2. **Input a reference string and number of frames.**
  3. **Keep track of frequency of each page.**
  4. **Replace the page with the lowest frequency when a new page needs to be loaded.**
  5. **Display total page faults and memory states.**

**A virtual memory system divides the logical address space into pages and the physical emory into frames. You are given a list of logical page references made by a running process. Write a program to simulate the basic paging technique of memory management.**

* 1. **Take the number of pages and number of frames as input.**
  2. **Map each page to a frame.**
  3. **If a frame is not available, show that the page cannot be loaded.**
  4. **Display the final page-to-frame mapping.**

**Code**

#include <stdio.h>

#include <limits.h>

#define MAX\_FRAMES 10

#define MAX\_PAGES 100

// Part 1: Basic Paging Technique

void basicPaging() {

int n\_pages, n\_frames;

printf("\n--- Basic Paging Simulation ---\n");

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

scanf("%d", &n\_pages);

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

scanf("%d", &n\_frames);

int page\_to\_frame[n\_pages];

int frame\_used[n\_frames];

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

page\_to\_frame[i] = -1;

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

frame\_used[i] = 0;

int frame\_index = 0;

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

if(frame\_index < n\_frames) {

page\_to\_frame[i] = frame\_index;

frame\_used[frame\_index] = 1;

frame\_index++;

} else {

page\_to\_frame[i] = -1; // Cannot be loaded

}

}

printf("\nPage\tFrame\n");

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

if(page\_to\_frame[i] != -1)

printf("%d\t%d\n", i, page\_to\_frame[i]);

else

printf("%d\tNot Loaded (No Frame)\n", i);

}

}

// Part 2: LFU Page Replacement

void lfuReplacement() {

int pages[MAX\_PAGES], frequency[MAX\_PAGES] = {0};

int frames[MAX\_FRAMES], freq\_in\_frame[MAX\_FRAMES];

int n\_pages, n\_frames, faults = 0;

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

printf("Enter number of pages in reference string: ");

scanf("%d", &n\_pages);

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

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

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

printf("Enter number of frames: ");

scanf("%d", &n\_frames);

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

frames[i] = -1;

freq\_in\_frame[i] = 0;

}

printf("\nPage\tFrames\t\tPage Fault\n");

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

int current = pages[i];

frequency[current]++;

int found = 0;

// Check if page is already in frame

for(int j = 0; j < n\_frames; ++j) {

if(frames[j] == current) {

freq\_in\_frame[j] = frequency[current];

found = 1;

break;

}

}

// If not in frame

if(!found) {

faults++;

int min\_freq = INT\_MAX;

int replace\_index = -1;

// Find empty frame or least frequently used

for(int j = 0; j < n\_frames; ++j) {

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

replace\_index = j;

break;

}

if(freq\_in\_frame[j] < min\_freq) {

min\_freq = freq\_in\_frame[j];

replace\_index = j;

}

}

frames[replace\_index] = current;

freq\_in\_frame[replace\_index] = frequency[current];

}

// Output state

printf("%d\t", current);

for(int j = 0; j < n\_frames; ++j) {

if(frames[j] != -1)

printf("%d ", frames[j]);

else

printf("- ");

}

if(!found)

printf("\tYes\n");

else

printf("\tNo\n");

}

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

}

// Main

int main() {

basicPaging();

lfuReplacement();

return 0;

}

**Output**

A screen shot of a computer

AI-generated content may be incorrect.