**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.**

**Code**

#include <stdio.h>

#define MAX\_FRAMES 100

#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], pages[MAX\_PAGES], time[MAX\_FRAMES];

int frameCount, pageCount, i, j, page, pos, faults = 0;

int counter = 0;

int isHit;

printf("Enter number of frames: ");

scanf("%d", &frameCount);

printf("Enter number of pages: ");

scanf("%d", &pageCount);

printf("Enter page reference sequence (space-separated): ");

for(i = 0; i < pageCount; ++i) {

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

}

for(i = 0; i < frameCount; ++i) {

frames[i] = -1; // Initialize all frames as empty

}

printf("\nPage Loading Process:\n");

for(i = 0; i < pageCount; ++i) {

isHit = 0;

for(j = 0; j < frameCount; ++j) {

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

counter++;

time[j] = counter;

isHit = 1;

printf("Page %d is already in frame %d.\n", pages[i], j);

break;

}

}

if(!isHit) {

for(j = 0; j < frameCount; ++j) {

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

counter++;

faults++;

frames[j] = pages[i];

time[j] = counter;

printf("Page %d loaded into empty frame %d.\n", pages[i], j);

break;

}

}

if(j == frameCount) {

pos = findLRU(time, frameCount);

printf("Page %d removed from frame %d (LRU).\n", frames[pos], pos);

counter++;

faults++;

frames[pos] = pages[i];

time[pos] = counter;

printf("Page %d loaded into frame %d.\n", pages[i], pos);

}

}

}

printf("\nFinal Page-to-Frame Mapping:\n");

for(i = 0; i < frameCount; ++i) {

if(frames[i] != -1)

printf("Frame %d => Page %d\n", i, frames[i]);

else

printf("Frame %d => Empty\n", i);

}

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

return 0;

}

**Output**

A screenshot of a computer

AI-generated content may be incorrect.

**Question no.2**

**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 100

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

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

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

if(time[i] < min) {

min = time[i];

pos = i;

}

}

return pos;

}

int main() {

int frames[MAX], pages[MAX], time[MAX];

int n, f, i, j, k, pageFaults = 0, counter = 0;

int hit;

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

scanf("%d", &n);

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

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

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

}

printf("Enter number of frames: ");

scanf("%d", &f);

for(i = 0; i < f; i++) {

frames[i] = -1; // Initialize all frames to empty

}

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

printf("-----\t----------------\t-----------\n");

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

hit = 0;

// Check if page is already in frames

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

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

counter++;

time[j] = counter;

hit = 1;

break;

}

}

if(!hit) {

// Miss - need to insert page

int pos;

if(pageFaults < f) {

// There's space in memory

pos = pageFaults;

} else {

// Memory full - find LRU page to replace

pos = findLRU(time, f);

}

counter++;

frames[pos] = pages[i];

time[pos] = counter;

pageFaults++;

}

// Print current memory state

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

for(k = 0; k < f; k++) {

if(frames[k] != -1)

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

else

printf("- ");

}

if(!hit)

printf("\t\tYes");

else

printf("\t\tNo");

printf("\n");

}

printf("\nTotal Page Faults = %d\n", pageFaults);

return 0;

}

**Output**

A screen shot of a computer

AI-generated content may be incorrect.

**Question no.3**

**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.**

**Code**

#include <stdio.h>

#include <limits.h>

#define MAX 100

typedef struct {

int page;

int freq;

int time; // To break ties in frequency (Least Recently Used among equal frequency)

} Frame;

int findLFU(Frame frames[], int f) {

int i, minFreq = INT\_MAX, oldestTime = INT\_MAX, pos = -1;

for(i = 0; i < f; i++) {

if(frames[i].freq < minFreq) {

minFreq = frames[i].freq;

oldestTime = frames[i].time;

pos = i;

} else if(frames[i].freq == minFreq) {

// If frequencies are equal, choose the least recently used

if(frames[i].time < oldestTime) {

oldestTime = frames[i].time;

pos = i;

}

}

}

return pos;

}

int main() {

Frame frames[MAX];

int pages[MAX];

int n, f, i, j, k, pageFaults = 0, timeCounter = 0;

int hit;

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

scanf("%d", &n);

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

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

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

}

printf("Enter number of frames: ");

scanf("%d", &f);

// Initialize all frames

for(i = 0; i < f; i++) {

frames[i].page = -1;

frames[i].freq = 0;

frames[i].time = 0;

}

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

printf("-----\t----------------\t-----------\n");

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

hit = 0;

// Check if page is already in memory

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

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

frames[j].freq++;

timeCounter++;

frames[j].time = timeCounter;

hit = 1;

break;

}

}

if(!hit) {

// Page not in memory

int pos = -1;

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

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

// Empty frame found

pos = j;

break;

}

}

if(pos == -1) {

// No empty frame, find LFU

pos = findLFU(frames, f);

}

// Replace page

frames[pos].page = pages[i];

frames[pos].freq = 1;

timeCounter++;

frames[pos].time = timeCounter;

pageFaults++;

}

// Display memory state

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

for(k = 0; k < f; k++) {

if(frames[k].page != -1)

printf("%d ", frames[k].page);

else

printf("- ");

}

if(!hit)

printf("\t\tYes");

else

printf("\t\tNo");

printf("\n");

}

printf("\nTotal Page Faults = %d\n", pageFaults);

return 0;

}

**Output**

A screen shot of a computer

AI-generated content may be incorrect.

**Question no.2**

**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>

#define MAX 100

int main() {

int numPages, numFrames;

int referenceString[MAX];

int pageTable[MAX]; // pageTable[i] stores the frame number of page i

int frames[MAX]; // To check which frames are occupied

int usedFrames = 0;

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

scanf("%d", &numPages);

printf("Enter number of frames: ");

scanf("%d", &numFrames);

printf("Enter the number of page references: ");

int n;

scanf("%d", &n);

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

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

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

}

// Initialize page table and frame table

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

pageTable[i] = -1;

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

frames[i] = -1;

printf("\nPage Loading Simulation:\n");

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

int page = referenceString[i];

if (pageTable[page] != -1) {

// Page is already loaded

printf("Page %d is already in Frame %d\n", page, pageTable[page]);

} else {

// Need to load the page

if (usedFrames < numFrames) {

frames[usedFrames] = page;

pageTable[page] = usedFrames;

printf("Loaded Page %d into Frame %d\n", page, usedFrames);

usedFrames++;

} else {

printf("Cannot load Page %d: No free frame available.\n", page);

}

}

}

// Final page-to-frame mapping

printf("\nFinal Page-to-Frame Mapping:\n");

printf("Page\tFrame\n");

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

if (pageTable[i] != -1)

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

else

printf("%d\tNot Loaded\n", i);

}

return 0;

}

**Output**

A computer screen shot of a program

AI-generated content may be incorrect.