**LAB 8**

**PAGE REPLACEMENT ALGORITHM**

The page replacement algorithms we use in this lab are as follows:

1. Optimal replacement

2. FIFO replacement

3. LRU replacement

**For Optimal Page Replacement:**

#include <stdio.h>

int main() {

int no\_of\_frames, no\_of\_pages, frames[10], ref\_string[30], temp[10], flag1, flag2, flag3;

int i, j, k, pos, max, faults = 0;

int flag = 0; // for hit if flag = 1 ; then hit else fault

printf("Enter number of frames: ");

scanf("%d", &no\_of\_frames);

printf("Enter number of pages: ");

scanf("%d", &no\_of\_pages);

printf("Enter page reference string: ");

for(i = 0; i < no\_of\_pages; ++i){

scanf("%d", &ref\_string[i]);

}

for(i = 0; i < no\_of\_frames; ++i){

frames[i] = -1;

}

printf("\n==============================================================================\n" );

for(int i=0;i<no\_of\_frames; i++)

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

printf("page fault" );

printf("\n==============================================================================" );

for(i = 0; i < no\_of\_pages; ++i){

flag1 = flag2 = 0;

for(j = 0; j < no\_of\_frames; ++j){

if(frames[j] == ref\_string[i]){

flag1 = flag2 = 1;

flag = 1;

break;

}

}

if(flag1 == 0){

for(j = 0; j < no\_of\_frames; ++j){

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

faults++;

frames[j] = ref\_string[i];

flag2 = 1;

break;

}

}

}

if(flag2 == 0){

flag3 =0;

for(j = 0; j < no\_of\_frames; ++j){

temp[j] = -1;

for(k = i + 1; k < no\_of\_pages; ++k){

if(frames[j] == ref\_string[k]){

temp[j] = k;

break;

}

}

}

for(j = 0; j < no\_of\_frames; ++j){

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

pos = j;

flag3 = 1;

break;

}

}

if(flag3 ==0){

max = temp[0];

pos = 0;

for(j = 1; j < no\_of\_frames; ++j){

if(temp[j] > max){

max = temp[j];

pos = j;

}

}

}

frames[pos] = ref\_string[i];

faults++;

}

printf("\n");

for(j = 0; j < no\_of\_frames; ++j){

printf("%4d\t\t", frames[j]);

}

if (flag == 0)

printf(" F" );

flag = 0;

}

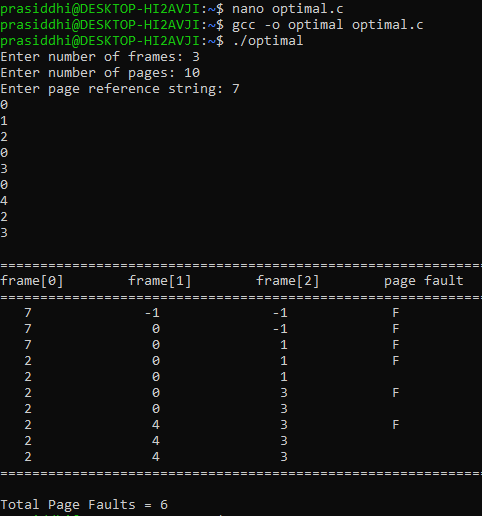
printf("\n==============================================================================" );

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

return 0;

}

**Output:**

****

**For FIFO Replacement:**

#include <stdio.h>

int main() {

int num\_pages, num\_frames, page\_faults = 0;

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

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

int pages[num\_pages];

printf("Enter the page reference string: ");

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

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

}

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

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

int frames[num\_frames];

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

frames[i] = -1;

}

int oldest\_frame = 0;

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

int page = pages[i];

int page\_found = 0;

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

if (frames[j] == page) {

page\_found = 1;

break;

}

}

if (!page\_found) {

frames[oldest\_frame] = page;

oldest\_frame = (oldest\_frame + 1) % num\_frames;

page\_faults++;

}

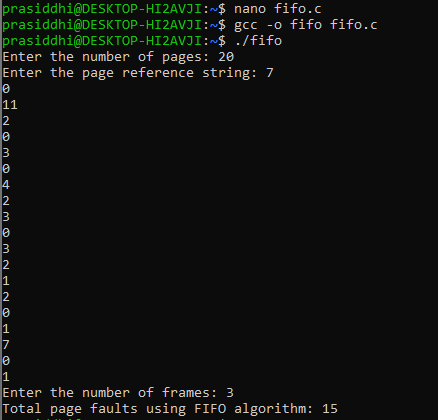
}

printf("Total page faults using FIFO algorithm: %d\n", page\_faults);

return 0;

}

**Output:**

****

**For LRU Replacement:**

#include <stdio.h>

int main() {

int num\_pages, num\_frames, page\_faults = 0;

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

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

int pages[num\_pages];

printf("Enter the page reference string: ");

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

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

}

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

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

int frames[num\_frames];

int last\_used[num\_frames];

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

frames[i] = -1;

last\_used[i] = -1;

}

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

int page = pages[i];

int page\_found = 0;

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

if (frames[j] == page) {

page\_found = 1;

last\_used[j] = i;

break;

}

}

if (!page\_found) {

int lru\_frame = 0;

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

if (last\_used[j] < last\_used[lru\_frame]) {

lru\_frame = j;

}

}

frames[lru\_frame] = page;

last\_used[lru\_frame] = i;

page\_faults++;

}

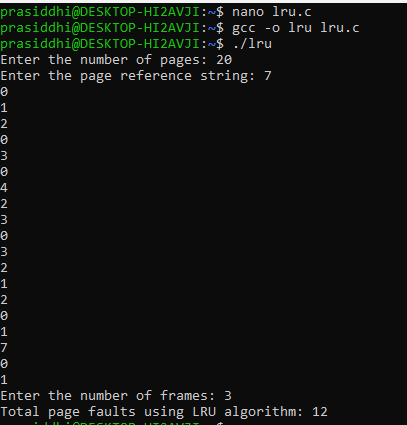
}

printf("Total page faults using LRU algorithm: %d\n", page\_faults);

return 0;

}

**Output:**

****