**Mirette Boules Hanna**

**ID=5515**

**Group 2**

**Lab4**

**Paging and Replacement**

Replacement Algortithms:

Fifo:

First-In-First-Out

..we replace the new page with (the oldest in array who came first )

LRU(least recently used):

we replace the current page with (least recently used)we keep track of the past and choose the page who has been least recently used to replace with.

Optimal:

we replace the current page with (the farest page or the farest duplicated element ) we keep track of the future pages.

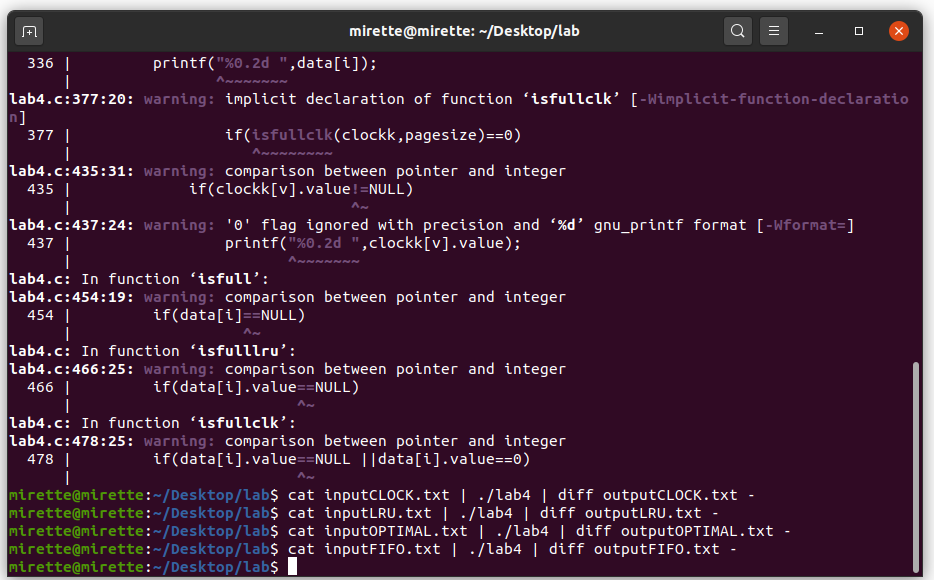
Clock:

We replace the new page with the page which its usebit =0 ,

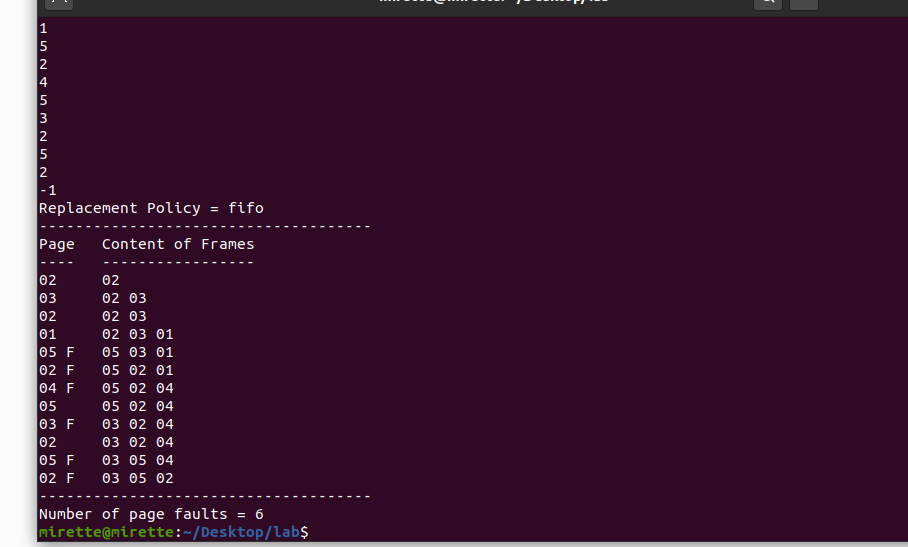
When a page enters the array we assign its use bit by default = 1 and

When the clock points to it .. it decrease it by 1 if its value its 0 it replace it with the new page .

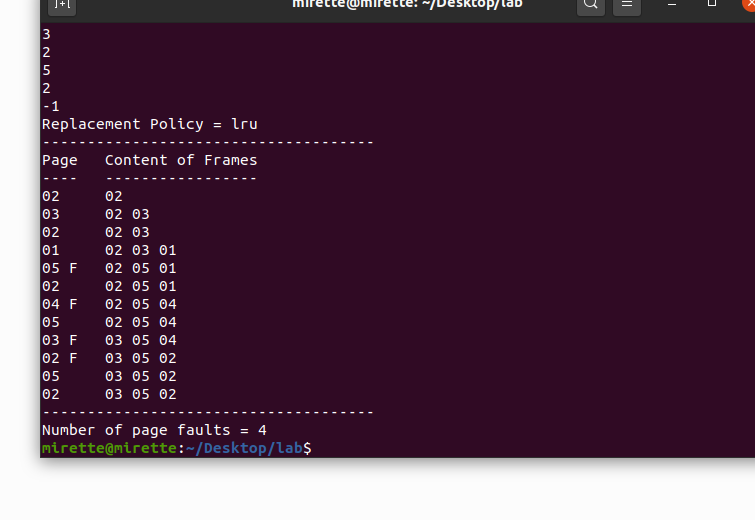
Output

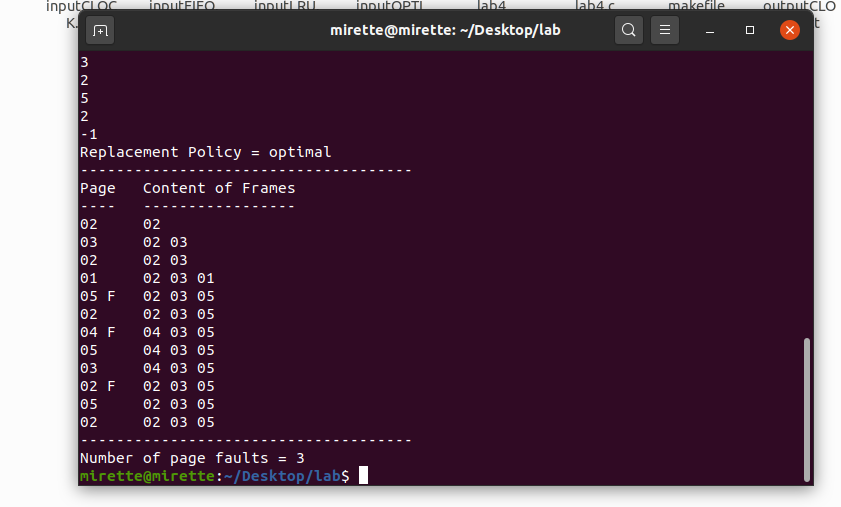
Makefile output : the output of the program matches the output in the output files so when I run the command it produce nothing in case of successful like in the photo 

**Output without makefile**

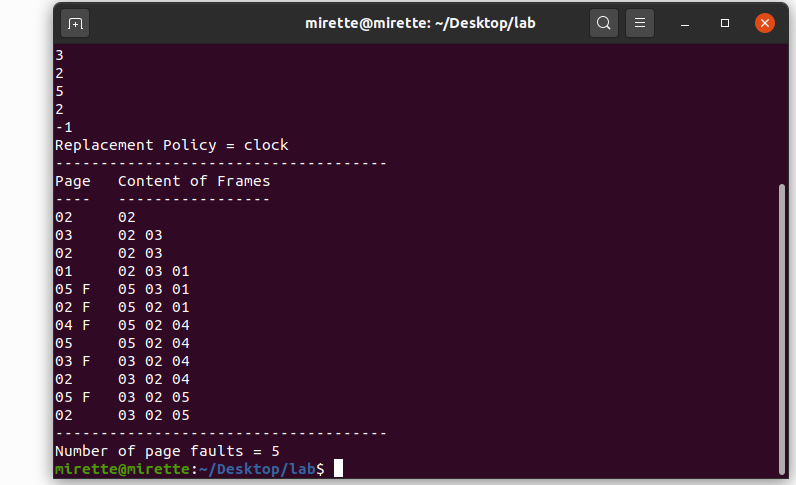
**Fifo**

**LRU**



**Optimal**

**Clock**



**Code well commented**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

int \*page,pagesize,datasize;

typedef struct

{

int value;

int bit;

} clock\_frame;

clock\_frame \*clockk;

typedef struct

{

int value;

int used;

} LRU\_frame;

void FIFO(int data[])

{

int i,j=0,k,flag,flag2,t,fault=0,v;

for(i=0; i<datasize; i++)

{//for circular array

if(j%pagesize==0)

{

j=0;

}

flag=0,flag2=0;

printf("%0.2d ",data[i]);

//if index is smaller than pagesize

//if it is present in array we set flag2 to 1

//else we append it immediatly

if(i<pagesize)

{

for(t=0; t<pagesize; t++)

{

if(page[t]==data[i])

{ printf(" ");

flag2=1;

break;

}

}

if(flag2==0)

{ printf(" ");

page[j]=data[i];

j++;

}

}//if index is grater than pagesize

//if present in [] we set flag to 1

//else we increase fault(if array is full) and replace it with the oldest index j

else if(i>=pagesize)

{

for(k=0; k<pagesize; k++)

{

if(page[k]==data[i])

{ printf(" ");

flag=1;

break;

}

}

if(flag==0)

{

if(isfull(page,pagesize)==0)

{

fault++;

printf("F");

page[j]=data[i];

j++;

}

else{

page[j]=data[i];

j++;

printf(" ");

}

}

}

printf(" ");

for(v=0; v<pagesize; v++)

{

if(page[v]!=NULL)

{

printf("%0.2d ",page[v]);

}

}

printf("\n");

}

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

printf("Number of page faults = %d",fault);

}

void LRU(int data[])

{

int j=0,i,v,t,flag2,fault=0,min,flag=0;

LRU\_frame \*frame=(LRU\_frame\*)malloc(sizeof(LRU\_frame)\*pagesize);

//flag2 when zero means that the element is not yet entred the lru[]

for(i=0; i<datasize; i++)

{

flag2=0;

printf("%0.2d ",data[i]);

//j like ptr to index in lru[]

//circular lru[]

if(j%pagesize==0)

{

j=0;

}

//append in lru[] because empty and set usedvalue with the index and falg2 to 1

if(i<pagesize)

{

for(t=0; t<pagesize; t++)

{

// check if data is already in lru[] ..if yes it change the value of use bit only

if(frame[t].value==data[i])

{printf(" ");

frame[t].used=i;

flag2=1;

break;

}

}

//if the elemnentis new it enters the lru[]

if(flag2==0)

{printf(" ");

frame[j].value=data[i];

frame[j].used=i;

j++;

}

}

//if the buffer is full and we want to replace an element or if it is present to update only the value of used

else

{

for(t=0; t<pagesize; t++)

{

//if prensent in lru[] update its used value only

if(frame[t].value==data[i])

{printf(" ");

flag2=1;

frame[t].used=i;

break;

}

}

//if nor present 1.increase page fault 2.repace it with the oldest recently used

if(flag2==0)

{

min=frame[0].used;

if(isfulllru(frame,pagesize)==0)

{

fault++;

printf("F");

//bringing the oldest recently used in min

for(v=0; v<pagesize; v++)

{

if(frame[v].used<min)

{

min=frame[v].used;

}

}

//replace it the the new element

for(v=0;v<pagesize;v++)

{

if(frame[v].used==min)

{j=v;break;}

}

frame[j].value=data[i];

frame[j].used=i;

}

else //if there is place in frame[]

{printf(" ");

frame[j].value=data[i];

frame[j].used=i;

j++;

}

}

}

printf(" ");

for(v=0; v<pagesize; v++)

{

if(frame[v].value!=NULL)

{

printf("%0.2d ",frame[v].value);

}

}

printf("\n");

}

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

printf("Number of page faults = %d",fault);

}

void Optimal(int data[])

{

int \*future=(int \*)malloc(sizeof(int)\*pagesize);

int i,j=0,k,flag,flag2,t,fault=0,v,c,max,max\_index;

for(i=0; i<datasize; i++)

{//for circular array

if(j%pagesize==0)

{

j=0;

}

flag=0,flag2=0;

printf("%0.2d ",data[i]);

//if index is smaller than the pagesize .. if it is present in in array update flag to 1 else append it

if(i<pagesize)

{

for(t=0; t<pagesize; t++)

{

if(page[t]==data[i])

{

printf(" ");

flag2=1;

break;

}

}

if(flag2==0)

{ printf(" ");

page[j]=data[i];

j++;

}

}

//if index is greater than page size

else if(i>=pagesize)

{

for(k=0; k<pagesize; k++)

{//if element is already present in array set flag to 1

if(page[k]==data[i])

{ printf(" ");

flag=1;

break;

}

}

if(flag==0)

{ //if array is full we need to keep track to the farset element to replace it

if(isfull(page,pagesize)==0)

{

for(t=0; t<pagesize; t++)

{

c=1;

for(v=i; v<datasize; v++)

{

if(page[t]==data[v])

{

future[t]=c;

break;

}

if(v==datasize-1)

{

future[t]=c;

}

c++;

}

}

max=future[0];

max\_index=0;

for(t=0; t<pagesize; t++)

{

if(future[t]>max)

{

max=future[t];

max\_index=t;

}

}

j=max\_index;

} // we increase fault as pagesize is full

if(isfull(page,pagesize)==0)

{

fault++;

printf("F");

page[j]=data[i];

j++;

}

else{

page[j]=data[i];

j++;

printf(" ");

}

}

}

printf(" ");

for(v=0; v<pagesize; v++)

{

if(page[v]!=NULL)

{

printf("%0.2d ",page[v]);

}

}

printf("\n");

}

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

printf("Number of page faults = %d",fault);

}

void clock(int data[])

{ //flag -> if 1 means no replacement or adding will occur

clockk=(clock\_frame\*)malloc(sizeof(clock\_frame)\*pagesize);

int i,j=0,k,flag,flag2,t,fault=0,v,hint=0,flag1=0;

for(i=0; i<datasize; i++)

{

if(j%pagesize==0)

{

j=0;

}

flag=0,flag2=0;

printf("%0.2d ",data[i]);

//this is to fill the clock[] first before fault because its empty

if(i<pagesize)

{

for(t=0; t<pagesize; t++)

{ //if the data is in the clock[] it will leave it without no replacement and update usebit to 1 if it were 0 and flag is 1

if(clockk[t].value==data[i])

{ printf(" ");

if(clockk[t].bit==0)

clockk[t].bit=1;

flag=1;

break;

}

}

//if there is no element in index ..we add one

//j is to know where i am in clock[]

if(flag==0)

{printf(" ");

clockk[j].value=data[i];

clockk[j].bit=1;

j++;

}

}

//this is when the clock[] is full and i want to choose which element to replace

else if(i>=pagesize)

{

for(k=0; k<pagesize; k++)

{ //if the data is in the clock[] it will leave it without no replacement and update usebit to 1 if it were 0 and flag is 1

if(clockk[k].value==data[i])

{printf(" ");

if(clockk[k].bit==0)

clockk[k].bit=1;

flag=1;

break;

}

}

//replacement will occur

if(flag==0)

{ //if clock[] is full and element isnot present we increase fault

if(isfullclk(clockk,pagesize)==0)

{

fault++;

printf("F");

//if used bit is 1 .. we decrease it by 1 and no rep will occure and we go to next elelment else we add our element and set its bit to 1 and go to next index

if(clockk[j].bit==1)

{

int p=0;

while(p<pagesize+1)

{

if(clockk[j].bit==1)

{

clockk[j].bit=0;

j++;

}

else

{

clockk[j].value=data[i];

clockk[j].bit=1;

j++;

break;

}

//for circular clock[] ;

if(j%pagesize==0)

j=0;

p++;

}

}

//if the use bit is 0 and ready for imideatly replace

else

{ clockk[j].value=data[i];

clockk[j].bit=1;

j++;

}

}

//when there is a place in clock[]

else

{printf(" ");

clockk[j].value=data[i];

clockk[j].bit=1;

j++;

}

}

}

printf(" ");

for(v=0; v<pagesize; v++)

{

if(clockk[v].value!=NULL)

{

printf("%0.2d ",clockk[v].value);

}

}

printf("\n");

}

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

printf("Number of page faults = %d",fault);

}

int isfull(int\* data,int size)

{

int i;

for(i=0; i<size; i++)

{

if(data[i]==NULL)

return 1;

}

return 0;

}

int isfulllru(LRU\_frame\* data,int size)

{

int i;

for(i=0; i<size; i++)

{

if(data[i].value==NULL)

return 1;

}

return 0;

}

int isfullclk(clock\_frame\* data,int size)

{

int i;

for(i=0; i<size; i++)

{

if(data[i].value==NULL ||data[i].value==0)

return 1;

}

return 0;

}

int main()

{

int data[150];

scanf("%d",&pagesize);

page=(int\*) malloc(sizeof(int)\*pagesize);

char input[10];

scanf("%s",input);

int i=0,j=0;

while(j!=-1)

{

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

j=data[i];

i++;

}

datasize=i-1;

printf("Replacement Policy = %s\n",input);

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

printf("Page Content of Frames\n");

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

if (strcasecmp(input,"FIFO")==0)

{

FIFO(data);

}

else if (strcasecmp(input,"OPTIMAL")==0)

{

Optimal(data);

}

else if (strcasecmp(input,"LRU")==0)

{

LRU(data);

}

else if(strcasecmp(input,"CLOCK")==0)

{

clock(data);

}

else

{

printf("Method doesn't exist");

}

printf("\n");

return 0;

}