VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A Computer Graphics & Visualization Mini Project Report
on

"LRU PAGE REPLACEMENT ALGORITHM"

Submitted in Partial fulfilment of the Requirements for the VI Semester of the Degree of

Bachelor of Engineering
In
Computer Science & Engineering
By

KAVYA SONI
(1CR19CS078)

GONDHI SAHANA
(1CR19CS063)

Under the Guidance of
Mrs. Krishna Sowjanya K
Asst. Professor, Dept. of CSE
and
Dr.Kiran Babu T S
Asst Professor,Dept. Of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Computer Graphics & Visualisation Project work entitled "**LRU page replacement algorithm"** has been carried out by **Kavya Soni (1CR19CS078) and Gondhi Sahana (1CR19CS063)** bonafide students of CMR Institute of Technology in partial fulfilment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2021-2022**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. This Computer Graphics Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

-----------------------	**----------------------**
Signature of Guide	**Signature of HOD**
Mrs.Krishna Sowjanya K **Assistant Professor** **Dept. of CSE, CMRIT**	**Dr. Shreekanth M Prabhu** **Professor, HOD** **Dept. of CSE, CMRIT**

<u>External Viva</u>

Name of the examiners	Signature with date
1.	
2.	

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. We take the privilege to express our sincere thanks to **Dr. Sanjay Jain**, our Principal for providing the encouragement and much support throughout our work. We thank him for making our stint thoroughly pleasant and enriching. We are deeply indebted to **Dr. Shreekanth M Prabhu** (Head of Department of CSE) and the entire team in the Computer Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work. All that we have done is only due to such supervision and assistance and we would not forget to thank them. We would like to extend our sincere esteems towards our guide, **Mrs.Krishna Sowjanya K** and **Dr.Kiran Babu T S** for the support, guidance and encouragement, he provided during the BE Project. This work would have not been possible without his valuable time, patience and motivation.

Kavya Soni (1CR19CS078)
Gondhi Sahana (1CR19CS063)

ABSTRACT

The purpose of this thesis work was to develop a trace-driven simulation to investigate the viability of applying a splay tree to a page replacement algorithm (new implementation). The basic idea of the splay tree is that fi'e-quently accessed items are placed near the root of the tree. This notion is compatible with the basic idea of the LRU page replacement algorithm. Reference strings consisting ofvirtual addresses were used as input for this simulation. To assess the performance of the splay tree, as applied to the implementation of the LRU page replacement algorithm, it was compared with other implementations of LRU approximations such as the clock algorithm and the additional- reference-bits algorithm. The perfonnance parameters were page fault rate, time and space complexities, and memory utilisation.

Four methods (leftmost, rightmost, highest, and LRU leaf) were used to select a victim page in the new implementation. Although the algorithm overhead (i.e., the time and space complexity) was lower in the leftmost and rightmost methods, the number of page faults and the memory utilization were not as good. The highest and LRU leaf methods generated the better results in terms of the number of page faults and memory utilization when compared with the clock and additional-reference-bits algorithms. The LRU leaf method had the demerit that its overhead was high. The highest leaf method, which did not need any hardware support, bad the most reasonable result over all

performance factors considered. Therefore, the highest leaf method of selecting a replacement victim in the new implementation using a splay tree could be recommended as a page replacement algorithm.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

The memory management of a computer system has a significant effect upon its operating system design. To execute a process, its instructions and data must be stored in main memory. Because of the restricted size of main memory, due to the fact that it is expensive relative to secondary memory, the execution of a process whose address space (i.e., instructions plus data) is larger than main memory is difficult. Also, as multiprogramming has been used to improve the utilisation of CPU, a single memory (i.e., only main memory) is not large enough to hold several processes. These problems may be solved by using virtual memory.

Silberschatz and Galvin state that "virtual memory is a technique that allows the execution of processes that may not be completely in memory" [Silberschatz and Galvin 94]. In this scenario, programs, each of wbjch can be larger than main memory, can be executed. So the programmer does not have to worry about the size of programs, The operating system keeps parts of the programs and data that are currently in use in main memory, and those parts that are not expected to be required soon are kept in secondary memory. Virtual memory is specially relevant to multiprogramming environments. Tanenbaum describes that '\while a program is waiting for part of itself to be swapped in, it is waiting for I/O and cannot run, so the CPU can be given to another process". In multiprogramming/time sharing systems, each user has the illusion that (s)he has a larger and individual memory of her/his own through the virtual memory scheme.

The basic idea of the virtual memory concept is separating the virtual addresses referenced in a running process from the real physical addresses in main memory. That is, the virtual address space and the real address space are separated. A programmer conceptualises a program in the virtual address space and the operating system links the program to the real address space locations. Actually, to execute a process, the virtual addresses of a process must be translated to real addresses dynamically.

Demand paging is frequently used to implement the fetching component of virtual memory management. In paged memory management scheme, the program and the data for each process are partitioned into equal-sized blocks called pages and stored in secondary memory. Main memory is also divided into fixed- sized blocks called frames. The pages and the frames are always the same size. When a process is executing, a page that is immediately needed is swapped into main memory and, unJess there is a free frame available, a page deemed not to be needed for a while is swapped out.

Thus, if there is no room in main memory for the page that has to be brought in, the operating system must choose a page to be removed from main memory, and replace it with the required page using a page replacement algorithm.

Since Belady's research on page replacement algorithms many algorithms have been introduced. The LRU (least recently used) replacement algorithm is considered to be close to the optimal algorithm (see Section 2.2 for a detailed discussion of replacement algorithms). The implementation of LRU requires special hardware support, which many systems do not provide, so various LRU approximations are usually used.

Splay tree, which is a self-adjusting binary search tree based on splaying (moving a referenced node to the root of a tree through a sequence of rotations), was developed by Slitter and Taiwan. As they claim, "splay tree approximately halves the depth of all nodes along the original path from the accessed node to the root". A splay tree does not require the maintenance of height or balance information. Thus it saves space and is simpler than a balanced tree. A splay tree has an amortised bound of O(log n) per operation. It is at least as efficient as a bal

anced tree and especially good in the case of a long sequence of accesses, because a node is likely to be accessed soon again when it is accessed once. Splay tree is practically useful in many applications.

Trace-driven simulation is one of the methods that can be used to evaluate the performance of a system. This method uses a dynamic sequence of addresses, which has been compiled during an actual execution, as input instead of actually executing instructions or generating results. Because designers do not have to be concerned about producing correct results or other overhead, they can focus on the performance of the designed system. The trace-driven model is thus frequently used to evaluate the performance of a proposed system.

The main goal of this thesis was to develop a trace-driven simulation to apply a splay tree to a page replacement algorithm. To execute the simulation, reference strings consisting of virtual addresses were used as input. lIDs new implementation was compared to traditional LRU approximation implementations.

The rest of this thesis is organised as follows. Chapter II provides a review of literature related to virtual memory management and splay tree. Chapter III contains the design and implementation issues. Chapter IV discusses evaluation. Finally, Chapter V gives the summary and future work.

Chapter 2

SYSTEM REQUIREMENTS

2.1 FEASIBILITY STUDY :-

It deals with handling system problem. The feasibility study proposes one or more conceptual solutions to the problem set for the project. The conceptual solutions give an idea of what the system will look like. They define what will be done on the computer and what will remain manual. They also indicate what input will be needed by the system and what output will be produced by the system. Also it tells that whether the plan of the project is been made by the people. They also indicate whether the input will be needed by the system and what manual the output produced i.e in feasibility study the analyst has to do evaluation of existing systems and procedures. He has to present a number of alternative solutions to the user requirements.

2.1.1 Operational :-

This feasibility study deals with the operation of the project. Here we are going to explain the whole operation of the project .In this project we have provide the ado connection which is useful for the connectivity of the database. This will provide the connection so that with which we can open more then one form due to this ado connection. This project is totally based on the visual basic language and in this visual basic we have provided the connection and also we have used ms-access so that we can create an database table. Here we have provided the user name and password facility so that there are some form which should provide security purpose. A system development project are likely to be feasible if it meets user requirements , needs , and expectation. User acceptance is an important determinant of operational feasibility. It requires careful consideration. Here also there are several reports generated by the database access so that which will be useful for entering the record of the employee .

2.1.2 Technical :-

This is a technical problem feasibility study. A system development project is likely to be feasible if it meets the user requirement. Here we have decided to use technical language called CODE BLOCKS and FREEGLUT. The purpose behind this is that we are more familiar with this language and it is easily access by the other user. Also the code for writing this is very easy and we can get the information in many books. Also it deal with knowledge of current and emerging technology solutions. Also there are several coding which we have done that will be very useful in terms of technical term. It is concerned with the capacity of the proposed system to meet initial performance expectation and accommodate new functionality over the medium term.

2.1.3 Cost/Benefit analysis :-

The cost of the project is less as compared to the other project. Also it is cheaper as compared to other project. Here only software is used so there is no need of extra cost that the project might be damaged. It is user friendly and can be interacted with other person. Also it can be used by any person if that person knows the language, then can be easily handled. Also there is no need of having any high range pc it can work with normal configuration pc.

2.2 SYSTEM DEVELOPMENT :-

In this System Development there are several steps which are to be followed they are:

2.2.1 System Engineering:-

This phase deals with the engineering part of the project. In system engineering work cannot be established before the system requirements for allocating the needs and subset of the requirements. In this we will be specifying that whether it will be easy to work with this project or not. Also we have here provided the requirements like html and css.

2.2.2 System Analysis:-

Analysis is a detailed of the various operation performed by the system and the relationship exist between the system. In our project we analysed the relationship that we will be using in our project. In analysisment we also decided how many module we will be including in our project. Also the brief overview of how our project will look like was done in the analysisment. We also decided what type of connectivity we will be providing in our project was done. In analysisment also we analyse that in how many days we will be completing our project so that it will be submitted in correct time as per the given schedule.

2.3 HARDWARE SPECIALISATION:-

a. Any Operating System

b. Any Computer Processors

c. Any screen resolution

2.4 SOFTWARE SPECIALISATION:-

This graphics package can be used in any platform.

Ø Development Platform: WINDOWS

Ø Development tool: CODE-BLOCKS

Ø Language Used In Coding: C

Chapter 3

DESIGN

3.1 Proposed System

To achieve three dimensional effects, OpenGL software is proposed. It is software which provides a graphical interface. It is an interface between application program and graphics hardware. The advantages are:

- Ø OpenGL is designed as a streamlined.
- Ø it is a hardware independent interface, it can be implemented on many different hardware platforms.
- Ø With OpenGL, we can draw a small set of geometric primitives such as points, lines and polygons etc.
- Ø It provides double buffering which is vital in providing transformations.
- Ø It is event driven software.
- Ø It provides call back function.

Transformation Functions

- Ø Translation: Translation is done by adding the required amount of translation quantities to each of the points of the objects in the selected area. If P(x,y) be the a point and (tx, ty) translation quantities then the translated point is given by glTranslatef(dx,dy,dz);
- Ø Rotation: The rotation of an object by an angle 'a' is accomplished by rotating each of the points of the object. The rotated points can be obtained using the OpenGL functions glRotatef(angle, vx,vy,vz);
- Ø Scaling: The scaling operation on an object can be carried out for an object by multiplying each of the points (x,y,z) by the scaling factors sx, sy and sz. glScalef(sx,sy,sz);

Chapter 4

IMPLEMENTATION:-

4.1 DESCRIPTION:

GL primitives can have either flat or smooth shading. Smooth shading, the default, causes the computed colors of vertices to be interpolated as the primitive is rasterized typically assigning different colors to each resulting pixel fragment. Flat shading selects the computed color of just one vertex and assigns it to all the pixel fragments generated by rasterizing a single primitive.

4.2 OBJECTIVE:

The main goal of the thesis was to develop a trace-driven simulation to apply a splay tree as a data structure to implement an LRU approximation page replacement algorithm. Reference strings consisting of virtual addresses were used as input to this simulation. The performance of this new implementation was evaluated by comparing it with two popular LRU approximation algorithms, namely the clock algorithm and the additional-reference-bits algorithm. The performance factors for the evaluation were number ofpage faults, memory utilization, and time and space complexities.

4.3 Input Parameters :

4.3.1 Input Traces

The traces used as input to the simulation were developed at the Parallel Architecture Research Laboratory of New Mexico State University. They were available in the public directory ofthe ftp site tracebase@nmsu. edu.

4.3.2 Process Number

The number of processes is limited to ten (i.e., the maXImum degree of multiprogramming is ten). Each process handles one file, which consists of a different reference string. A user can select the number ofprocesses through a standard input.

4.3.3 Memory Size

A critical parameter in the simulation is the memory size. Excessively large memory results in no page faults and excessively small memory results in thrashing (the typical range for the miss rate is from 0.00001% to

0.00]% [Hennessy and Patterson 90J). The degree of multiprogramming is constrained as a consequence of the availability of a limited number of traces. The memory size too indirectly depends on the traces.

In the absence of historical data, the same traces that were used to drive the simulation, were used in a pre-processing step to deternrine a plausible memory size. The necessary memory size for running each process was

obtained by gradually increasing the memory size and considering the start point for each process at which the number ofpage faults generated becomes stable (i.e., would not decrease) in the face offurther increasing the memory size. The memory size can also be selected through a standard input.
Having determined the memory size necessary for each process, two methods were used to arrive at the overall memory size for the simulation. The first method consisted of three steps. The first step was to take the median values among the start points of all processes obtained by using each approach of each different algorithm The second step was to calculate the average value ofthese median values. The final step was to decide the memory size based on the above two steps. The memory size was average value '" /lumber ofprocesses *page size because the memory was equally partitioned to each process for the simulation.
The second method had also consisted of three steps. The only a difference was in the first step compared to the first method. The average value of each process was taken instead of the median value (with the minimum and maximum values excluded as possible outliers).

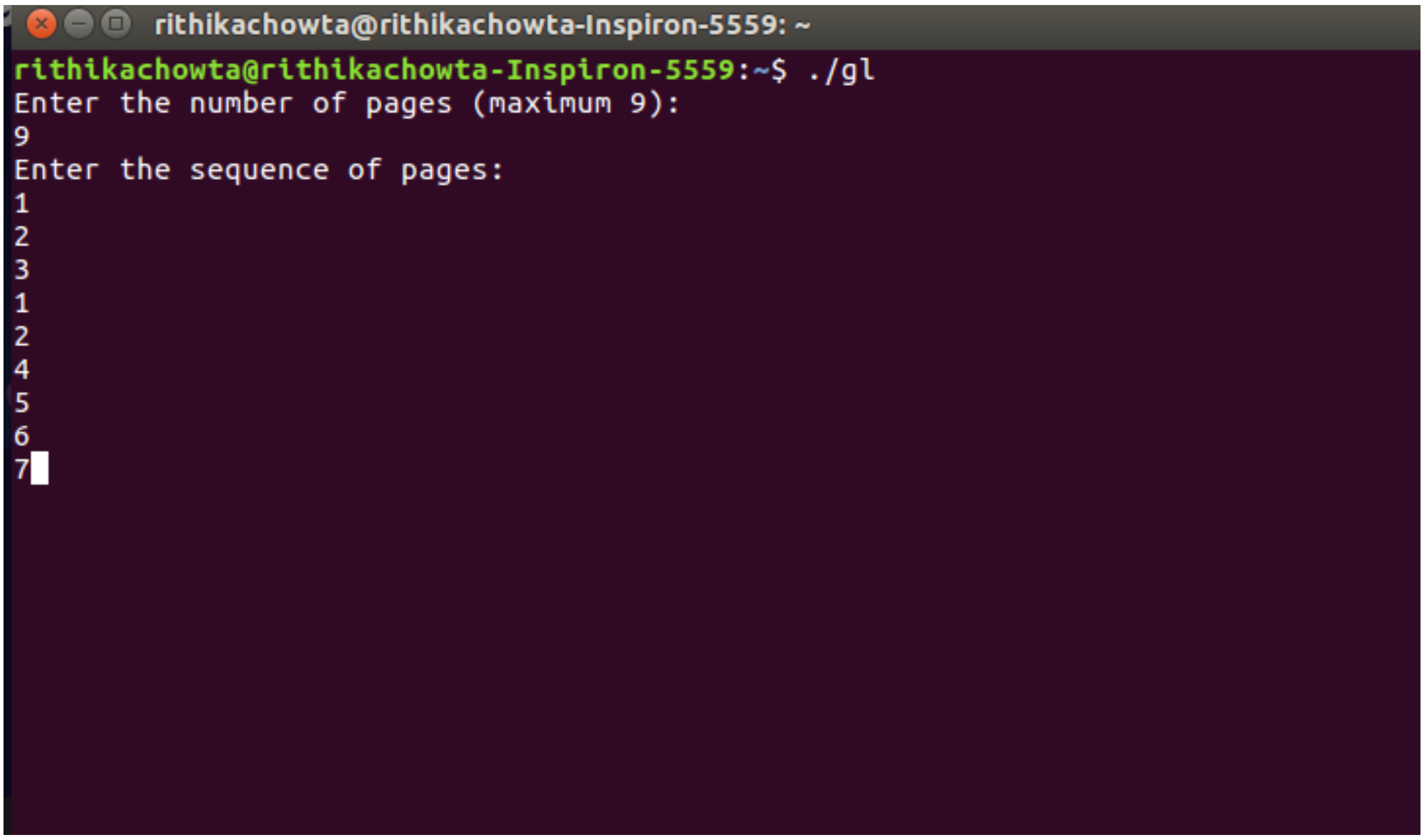
4.3.4 Page Replacement Algorithms

To investigate the perfonnance of the new LRU approximation implementation comparatively, two LRU approximation algorithms (i.e., clock and additional-reference- bits) were also implemented. A user can select any of the three algorithms and observe its performance by comparing it with the performance ofthe other two algorithms.
There are four different methods which a user can select to implement the new page replacement algorithm. First, the leftmost leaf page in a splay tree is replaced when a page replacement is needed. Second, the rightmost leaf page is replaced. Third, the highest leafpage in a splay tree is replaced. Fourth, the LRU page among the leaves is replaced.
The other two algorithms also have different methods by changing the time

intervals. Therefore, each implemented version of each algorithm also can be compared with the other versions of the same al!gorithm The best implemented version of each algorithm was compared with the best one of the other algorithms when checking the performance of three algorithms. Regular time intervals were assigned when clock and additional-reference-bits algorithms were executed.

Chapter 5

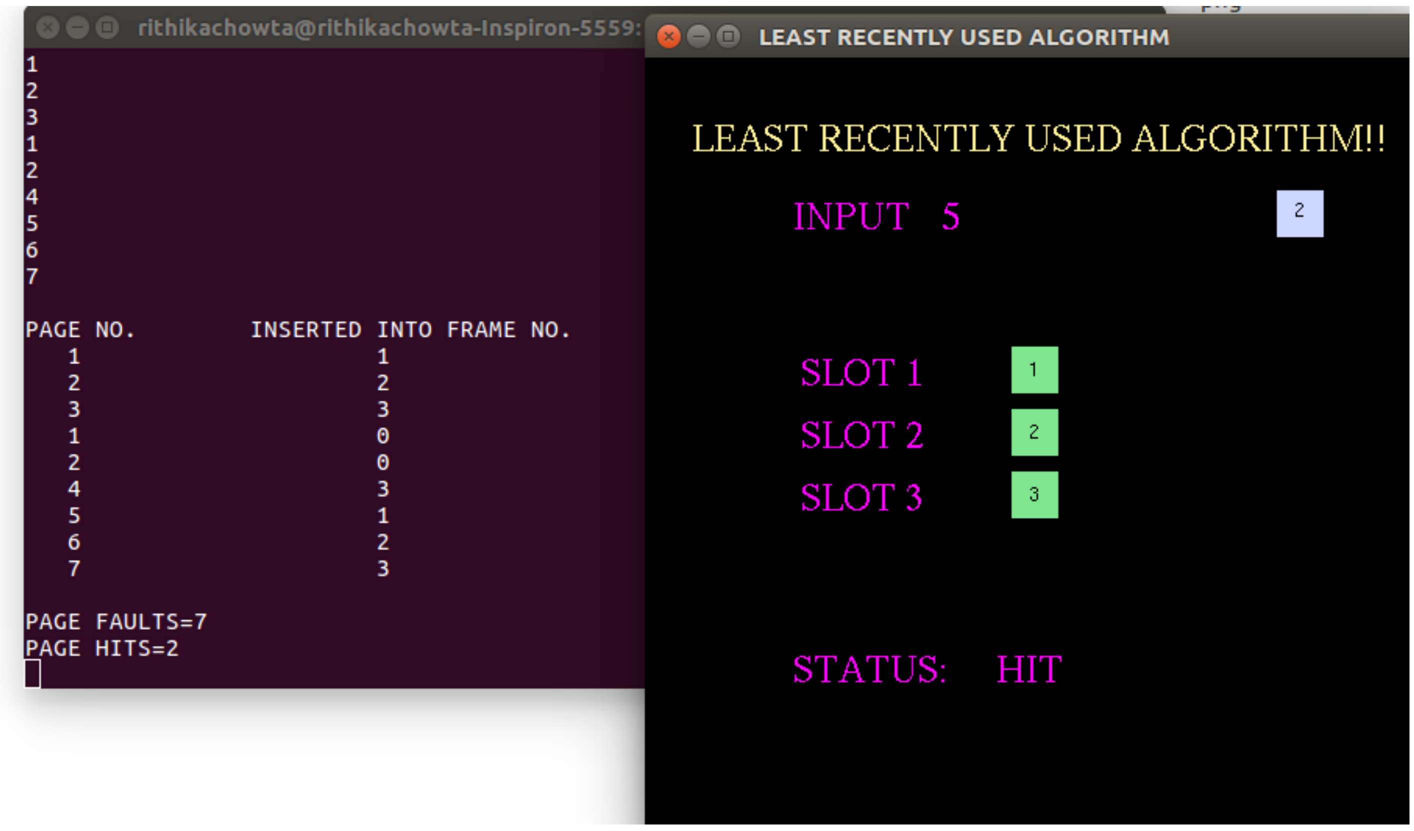
SNAPSHOTS:



```
rithikachowta@rithikachowta-Inspiron-5559: ~
rithikachowta@rithikachowta-Inspiron-5559:~$ ./gl
Enter the number of pages (maximum 9):
9
Enter the sequence of pages:
1
2
3
1
2
4
5
6
7
```

Figure: 5.1 INPUT

FIGURE 5.2 PAGE FAULT


rithikachowta@rithikachowta-Inspiron-5559:
LEAST RECENTLY USED ALGORITHM

```
1
2
3
1
2
4
5
6
7

PAGE NO.        INSERTED INTO FRAME NO.
   1                    1
   2                    2
   3                    3
   1                    0
   2                    0
   4                    3
   5                    1
   6                    2
   7                    3

PAGE FAULTS=7
PAGE HITS=2
```

LEAST RECENTLY USED ALGORITHM!!

INPUT 5   2

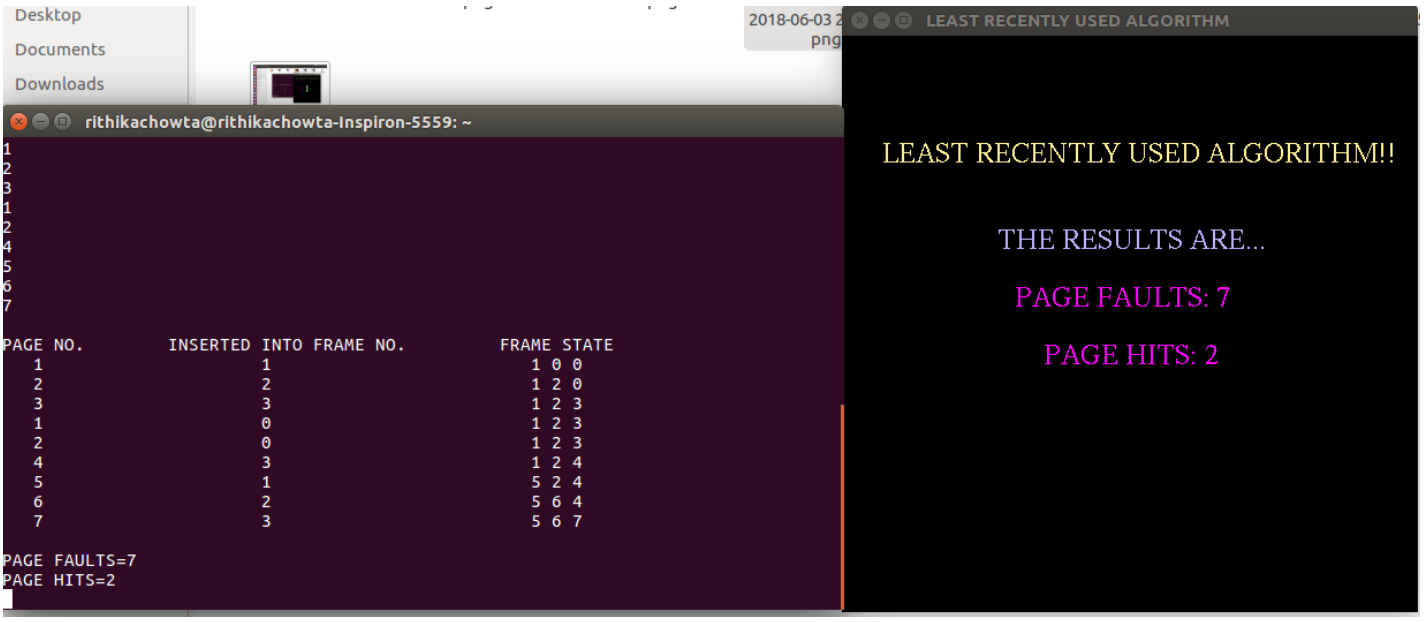
SLOT 1   1

SLOT 2   2

SLOT 3   3

STATUS: HIT

FIGURE 5.3 PAGE HIT


Desktop
Documents
Downloads
2018-06-03
png
LEAST RECENTLY USED ALGORITHM
rithikachowta@rithikachowta-Inspiron-5559: ~

```
1
2
3
1
2
4
5
6
7

PAGE NO.        INSERTED INTO FRAME NO.         FRAME STATE
   1                    1                          1 0 0
   2                    2                          1 2 0
   3                    3                          1 2 3
   1                    0                          1 2 3
   2                    0                          1 2 3
   4                    3                          1 2 4
   5                    1                          5 2 4
   6                    2                          5 6 4
   7                    3                          5 6 7

PAGE FAULTS=7
PAGE HITS=2
```

LEAST RECENTLY USED ALGORITHM!!

THE RESULTS ARE...

PAGE FAULTS: 7

PAGE HITS: 2

FIGURE 5.4 RESULTS

Chapter 6

CODE

```
#include<GL/gl.h>
#include<GL/glut.h>
#include<stdlib.h>
#include<unistd.h>
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<time.h>

int frmcontent[3]={0,0,0},frames[3]={0,0,0}, temp[3],pages[9],curpage=0,pos[9],pgf=0,n;
char status[9];
int diffx=0,diffy=0,i=0,window_id;

void init()
{
   glClearColor(0,0,0,0);
   glClear(GL_COLOR_BUFFER_BIT);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0,1000,0,1000);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glFlush();

}
void drawText(char *s,float x,float y,float z)

{
   char *c;
   glRasterPos3f(x,y,z);
   glColor3f(1,0,1);
```

```
    for (c=s; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
}

void clrsc()
{
    if(strcmp(status,"HIT")==0)
    {
         glClearColor(0,0,0,0);
         glClear(GL_COLOR_BUFFER_BIT);
         glColor3f(0.99,0.94,0.6);
     drawText("LEAST RECENTLY USED ALGORITHM!!",-500,800,0);
         glColor3f(0.75,0.7,1);
         drawText("THE RESULTS ARE...",-330,650,0);
         glColor3f(1,0,1);
         drawText("PAGE FAULTS:",-300,550,0);
         glRasterPos2f(50,550);
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, pgf+'0');
         drawText("PAGE HITS:",-250,450,0);
         glRasterPos2f(30,450);
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, (n-pgf)+'0');
         glFlush();

  }
    else
    {
         glClearColor(0,0,0,0);
         glClear(GL_COLOR_BUFFER_BIT);
         glColor3f(0.99,0.94,0.6);
     drawText("LEAST RECENTLY USED ALGORITHM!!",70,1150,0);
         glColor3f(0.75,0.7,1);
         drawText("THE RESULTS ARE...",270,1000,0);
         glColor3f(1,0,1);
         drawText("PAGE FAULTS:",300,900,0);
         glRasterPos2f(650,900);
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, pgf+'0');
         drawText("PAGE HITS:",350,800,0);
         glRasterPos2f(630,800);
         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, (n-pgf)+'0');
         glFlush();
    }
}

int lru(int pages[9],int cpg)
{
    int m, n, position, k, l;
int a = 0, b = 0;
    for(m = 0; m < 3; m++)
        if(frames[m] == pages[cpg])   //page hit
        {
            a = 1;

     b = 1;
            return -1;
        }
    if(a == 0)     //frame empty insert
    {
        for(m = 0; m < 3; m++)
            if(frames[m] == 0)
            {
                frames[m] = pages[cpg];
                b = 1;
                pgf++;
                return m;
```

```
         }
   }
   if(b == 0)  //page fault
   {
      for(m = 0; m < 3; m++)
                  temp[m] = 0;
      for(k = cpg-1, l = 1; l <=2; l++, k--)
      {
         for(m = 0; m <3; m++)
            if(frames[m] == pages[k])
                                        temp[m] = 1;
      }
      for(m = 0; m <3; m++)
         if(temp[m] == 0)
         {
            position = m;

           break;
         }

   frames[position] = pages[cpg];
      pgf++;
      return position;
   }
}

void boxpush(int x)
{
   glBegin(GL_POLYGON);
   glColor3f(0.8,0.84,1);
glVertex2f(470, 770);
glVertex2f(530, 770);
glVertex2f(530, 830);
```

```
glVertex2f(470, 830);
glEnd();
glColor3f(0,0,0);
glRasterPos2f(492, 796);
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, x+'0');
usleep(50000);
diffy-=10;
   if(pos[curpage]==0&&diffy>-200)
   {
       strcpy(status,"FAULT");
       glutPostRedisplay();
   }

 else if(pos[curpage]==1&&diffy>-300)
   {
       strcpy(status,"FAULT");

 glutPostRedisplay();
 }
   else if(pos[curpage]==2&&diffy>-380)
   {
       strcpy(status,"FAULT");
       glutPostRedisplay();
   }
    else if(pos[curpage]==-1&&diffx<550)
   {
       usleep(50000);
       diffy=0;
       diffx+=20;
       strcpy(status,"HIT");
       glutPostRedisplay();
   }
   else
   {
       if(curpage<n-1)
       {
            frmcontent[pos[curpage]]=pages[curpage];
            curpage++;
            diffy=0;

        diffx=0;
            glutPostRedisplay();
       }
       else
       {
            clrsc();

    }
   }
}
void drawframes()
{
   glBegin(GL_POLYGON);
   glColor3f(0.5,0.9,0.56);
glVertex2f(470, 570);
glVertex2f(530, 570);
glVertex2f(530, 630);
glVertex2f(470, 630);
glEnd();
glColor3f(0,0,0);
glRasterPos2f(492,592);
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, frmcontent[0]+'0');
glColor3f(0.5,0.9,0.56);
   glBegin(GL_POLYGON);
glVertex2f(470, 490);
glVertex2f(530, 490);
```

```
glVertex2f(530, 550);

glVertex2f(470, 550);

glEnd();
glColor3f(0,0,0);
glRasterPos2f(492,512);
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, frmcontent[1]+'0');
glColor3f(0.5,0.9,0.56);
   glBegin(GL_POLYGON);
glVertex2f(470, 410);
glVertex2f(530, 410);

glVertex2f(530, 470);
glVertex2f(470, 470);
glEnd();
glColor3f(0,0,0);
glRasterPos2f(492,432);
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, frmcontent[2]+'0');
}

void drawstatus()
{
   glColor3f(0.99,0.94,0.6);
drawText("LEAST RECENTLY USED ALGORITHM!!",60,880,0);
   glColor3f(1,0,1);
drawText("INPUT",190,780,0);
glColor3f(1,0,1);
glRasterPos2f(380, 780);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, curpage+1+'0');
drawText("SLOT 1",200,580,0);
drawText("SLOT 2”,200,500,0);
drawText("SLOT 3",200,420,0);
drawText("STATUS:",190,200,0);
drawText(status,450,200,0);
}

void mykey(unsigned char k, int c, int d)
{
if(k=='Q' || k=='q')
{
glutDestroyWindow(window_id);
}
}

void disp()
{
    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT);
    drawframes();
    drawstatus();
    glPushMatrix();
    glTranslatef(diffx,diffy,0);
    boxpush(pages[curpage]);
    glPopMatrix();
    glutSwapBuffers();
    glFlush();
}



int main(int argc,char **argv)
{
   int i;
   printf("Enter the number of pages (maximum 9):\n");
```

```
    scanf("%d",&n);
    printf("Enter the sequence of pages:\n");
    for(i=0;i<n;i++)
        scanf("%d",&pages[i]);
    glutInit(&argc,argv);
    glutInitWindowSize(500,500);
    window_id=glutCreateWindow("LEAST RECENTLY USED ALGORITHM");
    glutKeyboardFunc(mykey);
    init();
    glutDisplayFunc(disp);
    printf("\nPAGE NO.\tINSERTED INTO FRAME NO.\t\tFRAME STATE\n");
    for(i=0;i<n;i++)
    {
        pos[i]=lru(pages,i);
        printf("   %d\t\t\t %d\t\t\t   %d %d %d\n",pages[i],pos[i]+1,frames[0],frames[1],frames[2]);
    }
    printf("\nPAGE FAULTS=%d\n",pgf);
    printf("PAGE HITS=%d\n",n-pgf);
    glutMainLoop();
    return 0;
}
```

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

The significance of memory management, virtual memory, splay tree, trace-driven simulation, and the main objective of the thesis were stated. Chapter 11 contained a review ofthe virtual memory management schemes and splay tree operations. The topics covered in this chapter were paging, page replacement algorithms, splay tree, and penormance evaluation factors. Chapter ill presented the implementation platform and environment, and discussed the input parameters, the fundamental data structures used, and the implementation details to implement each algorithm. Chapter IV addressed the test programs (i.e., the test traces) used as input and the graphs obtained. This chapter also analyzed the results ofthe simulation using perfonnance graphs as well as time and space complexities.

The main goal of the thesis was to develop a trace-dJiven simulation to apply a splay tree to implement a page replacement algorithm. To drive the simulation, five traces consisting ofvirtual addresses, obtained from New Mexico State University, were used as input. The new implementation was compared to two traditional LRU approximations (i.e., dock and additional-reference-bits). The evaluation factors for penormance (i.e., page faults rate and memory utilization), were analyzed using graphs obtained from the

results of the simulation. The time and space complexities of the algorithms were also compared. Four methods were used to select a victim page in the new implementation: the leftmost leaf, the rightmost leaf, the highest leaf, and the LRU leaf methods. The highest leaf method, which does not need any hardware support, had the most reasonable result over the performance factors considered. Therefore, the highest leaf method could be recommended as a page replacement algorithm.

7.2 Future Work

The simulation (implemented as part of this thesis) handles the case where th.e memory is equally divided among processes. Equal allocation would not be an applicable approach when processes need to allocate memory according to their dynamic behaviors. Ifthe memory is divided among processes according to the estimated memory amount which each program needs, higher memory utilization and more tolerable page fau]t rate would be expected.

Parallel processes were not used in this simulation. Using two parallel processes for the new implementation (i.e., one for searching and the other for splaying) would be all attractive approach to decrease the execution time

Chapter 8

BIBLIOGRAPHY

This is a chapter that tells the references of the project. Means that it tells that from which we made our project. The bibliography of our project is:-

1. codeguru.com
1. google.com
2. blog.devart.com
3. creately.com
4. ignousupport.blogspot.com
5. geeksforgeeks.com
6. stackoverflow.com