辽 宁 科 技 大 学

课程设计报告

设计题目： 虚拟页面置换算法的实现

学院、系： 电信学院计算机系

专业班级： 计算机15-2班

学生姓名： 韦博文

指导教师： 柴玉梅

成 绩：

2018年 3 月 6 日

**目录**

一．概述 .......................................................1

二．使用的基本概念和原理 .......................................1

三、总体设计 .......................... .........................1

四、详细设计及编码设计 ...... ..................................1

五、测试时出现的问题及解决方法 ... .............................7

六、总结 .......................................................5

七、参考文献 ...................................................5

1. **概述**

1、目的：实现OPT、FIFO、LRU三种置换算法。

2、主要完成任务：实现三种算法并计算出每种算法的却页数，且以表格形式输出结果。

3、使用开发工具：Eclipse—Java。

4、解决主要问题：编写出可视化窗口输出运行结果。

二**、使用的基本概念和原理**

OPT：在分配内存页面数（本实验为3）小于进程页面数（本实验为10）时，先运行的3个页面放入内存中。有需要处理的新页面，则将原来内存中的3个页面中将来不再使用的调出，然后将新页面放入

FIFO：在分配内存页面数（本实验为3）小于进程页面数（本实验为10）时，先运行的3个页面放入内存中。这时有需要处理的新页面，则将原来内存中的3个页面中最先进入的调出，然后将新的页面放入。

LRU：在分配内存页面数（本实验为3）小于进程页面数（本实验为10）时，先运行的3个页面放入内存中。当需要调页面进入内存，而当前分配的页面全部不空闲时，选择其中最长时间没有使用的页面调出，以空出内存来放置新调入的页面。

三**、总体设计**

**OPT算法**

**手动输入**

**虚拟页面置换算法窗口**

**生成页面走向**

**缺页数**

**LRU算法**

**FIFO算法**

**随机生成**

**四、详细设计及编码设计**

1、首先设计可视化窗口：

**public** Xnym()

{

**super**("虚拟页面置换算法");

**final** JPanel panel0 = **new** JPanel(**new** GridLayout(2,1,3,3));

**final** JPanel panel1 = **new** JPanel(**new** BorderLayout());

**final** JPanel panel2 = **new** JPanel(**new** GridLayout(1,7,2,2));

**final** JPanel panel3 = **new** JPanel(**new** GridLayout(3,1,2,2));

**final** JPanel panel4 = **new** JPanel(**new** GridLayout(5,10));

JSplitPane split = **new** JSplitPane(JSplitPane.*VERTICAL\_SPLIT*,panel0,panel1);

**this**.add(split);

group0.add(BO);

group0.add(BF);

group0.add(BL);

group1.add(SJ);

group1.add(SD);

panel0.add(labelBT);

panel0.add(panel2);

panel1.add(panel3, BorderLayout.*WEST*);

panel1.add(panel4);

panel2.add(BO);BO.addActionListener(**this**);

panel2.add(BF);BF.addActionListener(**this**);

panel2.add(BL);BL.addActionListener(**this**);

panel2.add(SJ);SJ.addActionListener(**this**);

panel2.add(SD);SD.addActionListener(**this**);

panel2.add(labelQYS);

panel2.add(QYS);QYS.setEditable(**false**);

panel3.add(labelYM);

panel3.add(labelNC);

panel3.add(labelQY);

Font font0 = **new** Font("宋体",0,28);

Font font1 = **new** Font("宋体",0,24);

//初始化表格内容并添加入各自面板，设置部分不允许手动修改

**for**(**int** i = 0;i<50;i++)

{

t[i] = **new** JTextField(**null**);

t[i].setFont(font0);

}

**for**(**int** i = 0;i<50;i++)

{

panel4.add(t[i]);

}

**for**(**int** i = 10;i<50;i++)

{

t[i].setEnabled(**false**);

}

labelBT.setFont(font1);

**this**.setSize(600,370);//设置窗口大小

**this**.setLocation(30,240);//设置窗口位置

**this**.setVisible(**true**);

}

窗口设计尽量简约，易操作。所以在原有按钮的基础上改进为单选框，更加便宜操作

2、监听事件及各算法

本实验总需要三个算法：OPT、FIFO、LRU

//OPT算法

**if**(an == BO)

{

reset();qys = 0;

//存入页面走向数据

**int** num[] = **new** **int**[10];

**for**(**int** i = 0;i<10;i++)

{

num[i] = Integer.*parseInt*(t[i].getText());

}

//处理第一个数据

t[10].setText(t[0].getText());

t[40].setText("Y");qys++;

//处理第二列数据

**if**(num[1]==num[0])

{

t[11].setText(t[10].getText());

}**else**

{

t[11].setText(t[10].getText());

t[21].setText(t[1].getText());

t[41].setText("Y");qys++;

}

**if**(num[2]==num[0]||num[2]==num[1])

{

t[12].setText(t[11].getText());

t[22].setText(t[21].getText());

}**else**

{

t[12].setText(t[11].getText());

t[22].setText(t[21].getText());

t[32].setText(t[2].getText());

t[42].setText("Y");qys++;

}

//处理剩下数据

**for**(**int** i = 3;i<10;i++)

{

**if**(t[29+i].getText()==**null**)

{

t[10+i].setText(t[9+i].getText());

t[20+i].setText(t[19+i].getText());

t[30+i].setText(t[i].getText());

t[40+i].setText("Y");qys++;

}

**else**

{

//比较当前页面和内存中的页面并进行相应处理

**if**(num[i]==Integer.*parseInt*(t[9+i].getText())||num[i]==Integer.*parseInt*(t[19+i].getText())||num[i]==Integer.*parseInt*(t[29+i].getText()))

{

t[10+i].setText(t[9+i].getText());

t[20+i].setText(t[19+i].getText());

t[30+i].setText(t[29+i].getText());

}

**else**

{

**for**(**int** k = i+1;k<10;k++)

{

**if**(num[k]==Integer.*parseInt*(t[9+i].getText()))

{

n1 = k;**break**;

}

**if**(k == 9)n1 = 10;

}

**for**(**int** k = i+1;k<10;k++)

{

**if**(num[k]==Integer.*parseInt*(t[19+i].getText()))

{

n2 = k;**break**;

}

**if**(k == 9)n2 = 10;

}

**for**(**int** k = i+1;k<10;k++)

{

**if**(num[k]==Integer.*parseInt*(t[29+i].getText()))

{

n3 = k;**break**;

}

**if**(k == 9)n3 = 10;

}

**if**(((n1>=n2)&&(n1>=n3)))

{

t[10+i].setText(t[i].getText());

t[20+i].setText(t[19+i].getText());

t[30+i].setText(t[29+i].getText());

}

**if**((n2>n1)&&(n2>=n3))

{

t[10+i].setText(t[9+i].getText());

t[20+i].setText(t[i].getText());

t[30+i].setText(t[29+i].getText());

}

**if**((n3>n2)&&(n3>n1))

{

t[10+i].setText(t[9+i].getText());

t[20+i].setText(t[19+i].getText());

t[30+i].setText(t[i].getText());

}

t[40+i].setText("Y");qys++;

}

}

}

//输出缺页数

QYS.setText(String.*valueOf*(qys));

}

//FIFO算法

**if**(an == BF)

{

reset();qys = 0;

//存入页面走向数据

**int** num[] = **new** **int**[10];

**for**(**int** i = 0;i<10;i++)

{

num[i] = Integer.*parseInt*(t[i].getText());

}

//处理第一个数据

t[10].setText(t[0].getText());

t[40].setText("Y");qys++;

//处理第二列数据

**if**(num[1]==num[0])

{

t[11].setText(t[10].getText());

}

**else**

{

t[11].setText(t[1].getText());

t[21].setText(t[0].getText());

t[41].setText("Y");qys++;

}

**if**(num[2]==num[1]||num[2]== num[0])

{

t[12].setText(t[11].getText());

t[22].setText(t[21].getText());

t[32].setText(t[31].getText());

}

**else**

{

t[12].setText(t[2].getText());

t[22].setText(t[11].getText());

t[32].setText(t[21].getText());

t[42].setText("Y");qys++;

}

//处理其余数据

**for**(**int** i = 3;i<10;i++)

{

//将当前页面和已存入内存的页面进行比较并进行相应操作

**if**(num[i]==Integer.*parseInt*(t[9+i].getText())||num[i]==Integer.*parseInt*(t[19+i].getText())||num[i]==Integer.*parseInt*(t[29+i].getText()))

{

t[10+i].setText(t[9+i].getText());

t[20+i].setText(t[19+i].getText());

t[30+i].setText(t[29+i].getText());

}

**else**

{

t[10+i].setText(t[i].getText());

t[20+i].setText(t[9+i].getText());

t[30+i].setText(t[19+i].getText());

t[40+i].setText("Y");qys++;

}

}

//输出缺页数

QYS.setText(String.*valueOf*(qys));

}

//LRU算法

**if**(an == BL)

{

reset();qys = 0;

//存入页面走向数据

**int** num[] = **new** **int**[10];

**for**(**int** i = 0;i<10;i++)

{

num[i] = Integer.*parseInt*(t[i].getText());

}

//处理第一个数据

t[10].setText(t[0].getText());

t[40].setText("Y");qys++;

//处理第二列数据

**if**(num[1]==num[0])

{

t[11].setText(t[10].getText());

}

**else**

{

t[11].setText(t[1].getText());

t[21].setText(t[10].getText());

t[41].setText("Y");qys++;

}

//处理第三列数据

**if**(num[2]==num[0]||num[2]==num[1])

{

t[12].setText(t[11].getText());

t[22].setText(t[21].getText());

}

**else**

{

t[12].setText(t[2].getText());

t[22].setText(t[11].getText());

t[32].setText(t[21].getText());

t[42].setText("Y");qys++;

}

//处理其余数据

**for**(**int** i = 3;i<10;i++)

{

**if**(num[i]==num[i-1])

{

t[10+i].setText(t[9+i].getText());

t[20+i].setText(t[19+i].getText());

t[30+i].setText(t[29+i].getText());

}

**else**

{

t[10+i].setText(t[i].getText());

t[20+i].setText(t[9+i].getText());

t[30+i].setText(t[19+i].getText());

//如果内存中的页面并未发生变化应当算作未缺页！！！！！！！ **if**(num[i]!=Integer.*parseInt*(t[9+i].getText())&&num[i]!=Integer.*parseInt*(t[19+i].getText())&&num[i]!=Integer.*parseInt*(t[29+i].getText()))

t[40+i].setText("Y");qys++;

}

}

//输出缺页数

QYS.setText(String.*valueOf*(qys));

}

三个算法各有其特点，设计时思路需清晰明了。除此之外，需要被监听的还有“手动”、“随机”两个单选框，其方法如下：

//随机选项处理算法

**if**(an == SJ)

{

reset();qys = 0;

**for**(**int** i = 0;i < 10;i++)

{

**int** number = **new** Random().nextInt(8)+1;

String a = String.*valueOf*(number);

t[i].setText(a);

t[i].setEnabled(**false**);

}

}

//手动选项处理算法

**if**(an == SD)

{

reset();qys = 0;

**for**(**int** i = 0;i < 10;i++)

{

t[i].setEnabled(**true**);

t[i].setText(**null**);

}

}

以上所有方法在调用时第一步要进行对一次数据处理的清零

4、清零程序

//清零方法

**public** **void** reset()

{

**for**(**int** i = 10;i<50;i++)

t[i].setText(**null**);

}

清零程序极其重要，如未添加将致使程序只能单独处理一次数据。重复操作极其复杂不便捷

5、主函数——调用窗口方法

//主函数

**public** **static** **void** main( String[] args)

{

**new** Xnym();

}

**五、测试时出现过的问题及解决方法**

测试结果如下：

1. 手动输入下的OPT算法：



1. 手动输入下的FIFO算法：



1. 手动输入下的LRU算法：



1. 随机产生下的OPT算法：



1. 随机产生下的FIFO算法：



1. 随机产生下的LRU算法：



1、OPT算法的设计

OPT算法较其他两种算法难了很多。在思路上也很不同。所以一开始并未成功写出，后来在网上查阅一些资料提供了一些新的思路，终于解决。难处在于如何处理好当前内存中的页面和将来页面的比价并记录下将来页面出现的时间

2、LRU算法和FIFO算法中缺页的判断

和OPT算法不同的是这两种算法更容易出现当前页面已经在内存中出现，遂不能算作缺页。一开始并未考虑周到，设计成只要内存中数据发生变化（包括顺序变化）即缺页，后发现问题后及时调整解决问题

**六、总结**

本次实验收获颇丰。由于C++语言是最初学习的一种高级语言，现在基本忘记，所以用了自己比较熟悉的Java语言进行设计。还好遇见问题虽多但慢慢克难攻坚也都悉数解决。整个实验完成后自己对页面置换的三种算法也有了新的理解，较之前的只会做题而言对其的整个只换过程有了更加深入的了解和记忆。

**七、参考文献**

1、《计算机操作系统（第2版）》 主编：陆丽娜

2、《java程序设计实用教程（第4版）》 主编：叶核亚

3、百度文库、百度搜索