### 实验三、虚拟存储器管理

#### 目的要求

理解虚拟存储器概念，并掌握分页式存储管理地址转换和缺页中断的处理过程。用高级语言模拟请求分页式虚拟存储器的工作过程和页面置换算法LRU。

#### 准备知识

1. **分页式存储管理原理**

在存储器管理中，连续分配方式会形成许多“碎片”，虽然可通过“紧凑”方法将许多碎片拼接成可用的大块空间，但须为之付出很大开销。

如果允许将一个进程直接分散地装入到许多不相邻的分区中，则无须再进行“紧凑”。基于这一思想而产生了离散分配方式。如果离散分配的基本单位是页，则称为分页存储管理方式。在分页存储管理方式中，如果不具备页面对换功能，则称为基本分页存储管理方式，或称为纯分页存储管理方式，它不具有支持实现虚拟存储器的功能，它要求把每个作业全部装入内存后方能运行。

请求式分页系统是建立在基本分页基础上的，为了能支持虚拟存储器功能，而增加了请求调页功能和页面置换功能。

1. **LRU（Least recently used，最近最少使用）算法**

根据数据的历史访问记录来进行淘汰数据，其核心思想是“如果数据最近被访问过，那么将来被访问的几率也更高”。

1. **实现LRU的常用方法**
2. 用一个数组来存储数据，给每一个数据项标记一个访问时间戳，每次插入新数据项的时候，先把数组中存在的数据项的时间戳自增，并将新数据项的时间戳置为0并插入到数组中。每次访问数组中的数据项的时候，将被访问的数据项的时间戳置为0。当数组空间已满时，将时间戳最大的数据项淘汰。
3. 利用一个链表来实现，每次新插入数据的时候将新数据插到链表的头部；每次缓存命中（即数据被访问），则将数据移到链表头部；那么当链表满的时候，就将链表尾部的数据丢弃。

#### 实验内容

设计程序模拟请求分页式虚拟存储器的地址变换和缺页中断处理过程。

页面置换算法：最近最少使用（LRU）算法。

1. 建立一个存储分块表（MBT），表示内存所有物理页的当前状态，是否空闲。此表用来为进程分配所需的内存空间，每个进程可分配5个物理页。
2. 页面大小为1K个存储单元。
3. 允许多进程并发执行，为每个进程建立一个页表，表示该进程中各逻辑页是否已经调入内存，如果已经调入，该逻辑页对应的内存物理页号。
4. 当进程发生缺页中断，并且对应的5个物理页没有空闲时，实现采用LRU算法页面置换。
5. 程序模拟输出进程的访存过程：输入逻辑地址，转换为逻辑页号和页内地址，判断是否产生缺页中断完成调页，然后实现地址变换，查页表得到对应的物理页号，从而得到物理地址。

#### 实验结果

#### 

图 1 主界面

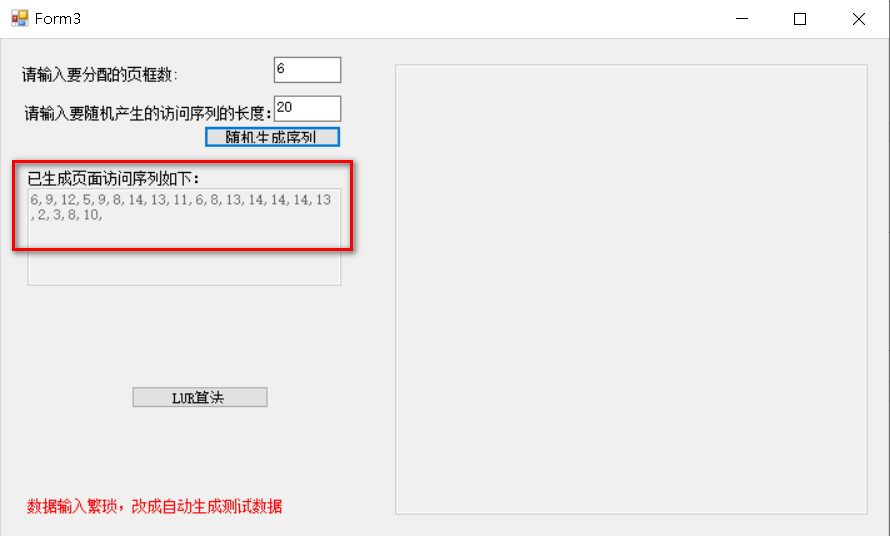


图 2 随机生成序列

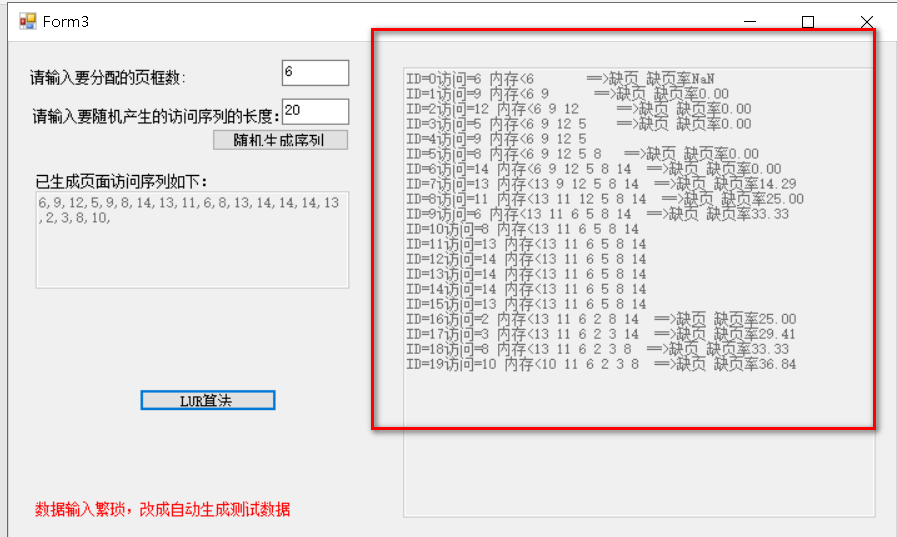


图 3 置换算法

#### 源程序

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace sy1234

{

public partial class Form3 : Form

{

int invalidcount = 0; // 缺页次数

int vpoint; //页面访问指针

int[] pageframe=new int[20]; // 分配的页框

int[] pagehistory = new int[20]; //用以记录页框中数据访问历史

int rpoint; //页面替换指针

int inpflag; //缺页标志，0为不缺页，1为缺页

PageInfo pageInfo = new PageInfo();

public Form3()

{

InitializeComponent();

}

private void Form3\_Load(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

textBox3.Text = "";

createps();

int i, n;

if (vpoint == 0)

{

string text = null;

for (i = 0; i < pageInfo.getTotal\_pn(); i++)

{

text += pageInfo.getSerial()[i]+",";

}

textBox3.Text = text;

}

}

void initialize(PageInfo pageInfo)

{

int i, pf;

inpflag = 0; //缺页标志，默认值为0不缺页

pageInfo.setDiseffect(0); //缺页次数，默认0

pageInfo.setFlag(0); //标志位，0表示无页面访问数据

pf = int.Parse(textBox1.Text);

pageInfo.setTotal\_pf(pf);

pageInfo.setSerial(null);// 清空页面序列

}

void createps()

{

initialize(pageInfo);

int s, i, pn;

initialize(pageInfo); //初始化相关数据结构

pn = int.Parse(textBox2.Text);

Random random = new Random();

s = pn; // 根据需求产生访问序列长度

pageInfo.setTotal\_pn(s);

int[] ser = new int[s];

for (i = 0; i < s; i++) //产生随机访问序列

{

ser[i] = random.Next(1, 16)+1;

}

pageInfo.setSerial(ser);

}

void displayinfo(PageInfo pageInfo)

{

int i, n;

textBox4.Text += "ID="+ vpoint + "访问=" + pageInfo.getSerial()[vpoint] + " 内存<";

for (n = 0; n < pageInfo.getTotal\_pf(); n++) // 页框信息

{

if (pageframe[n] >= 0)

textBox4.Text += pageframe[n]+" ";

else

textBox4.Text += " ";

}

if (inpflag == 1)

{

textBox4.Text += " ==>缺页 ";

textBox4.Text += "缺页率"+ ((float)(pageInfo.getDiseffect()) \* 100.00 / vpoint).ToString("F2");

}

textBox4.Text += "\r\n";

}

int findpage(int page, PageInfo pageInfo)

{

int n;

for (n = 0; n < pageInfo.getTotal\_pf(); n++)

{

pagehistory[n]++;

}

for (n = 0; n < pageInfo.getTotal\_pf(); n++)

{

if (pageframe[n] == page)

{

inpflag = 0;

pagehistory[0] = 0;

return 1;

}

}

inpflag = 1;

return 0;

}

//LUR页面置换算法

void LUR(PageInfo pageInfo)

{

int n, count, pstate;

rpoint = 0; // 页面替换指针

invalidcount = 0; // 缺页数初始化为0

count = 0; // 是否装满页框

for (n = 0; n < pageInfo.getTotal\_pf(); n++) // 清除页框信息

pageframe[n] = -1;

inpflag = 0;

for (vpoint = 0; vpoint < pageInfo.getTotal\_pn(); vpoint++) // 执行算法

{

pstate = findpage(pageInfo.getSerial()[vpoint],pageInfo);

if (count < pageInfo.getTotal\_pf()) // 开始时不计算缺页

{

if (pstate == 0)// 页不存在则装入页面

{

pageframe[rpoint] = pageInfo.getSerial()[vpoint];

rpoint = (rpoint + 1) % pageInfo.getTotal\_pf();

count++;

}

}

else // 正常缺页置换

{

if (pstate == 0)// 页不存在则置换页面

{

pageframe[rpoint] = pageInfo.getSerial()[vpoint];

rpoint = (rpoint + 1) % pageInfo.getTotal\_pf();

pageInfo.setDiseffect(pageInfo.getDiseffect()+1);

}

}

displayinfo(pageInfo); // 显示当前状态

} // 置换算法循环结束

return;

}

private void button1\_Click(object sender, EventArgs e)

{

textBox4.Text = "";

LUR(pageInfo);

}

}

}