李富荣

软件工程 | 2012118068

操作系统实训项目报告

项目二：段页式存储管理

1. **基本信息**

1，项目信息：段页式存储管理

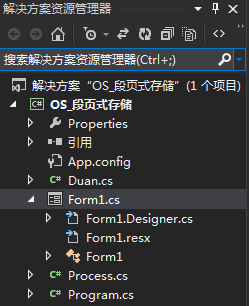
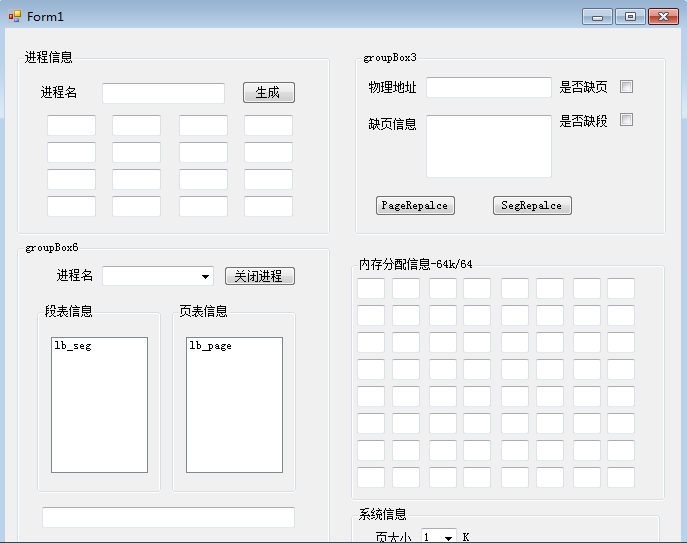
2，完成人信息：李富荣 2012118068

3，完成日期：2015.6.15

1. **实验内容与目的**
2. 加深理解段页式虚拟存储管理的概念和原理。
3. 掌握段页式存储管理中存储分配（和回收）方法；
4. 深入了解段页式虚拟存储管理中地址重定位（即地址映射）方法。
5. 深入理解段页式虚拟存储管理中缺段、缺页中断处理方法。

**三、主要设计思路和流程图**

使用Microsoft Visual Studio 2013为开发工具，在windows环境下开发，类视图及界面视图如下所示：

根据题目要求主存有64k的大小，每个页大小设为1k，因此这里把主存设为64个页框。题目要求进程最多有16个段，在界面的左上角是进程的创建部分，输入进程名，按顺序输入所需要的段的大小，点击生成，就会将信息存入List<Process> processes中，下面的ComboBox中就会显示进程。

创建进程后，系统为其分配两个驻留集，最多可以存放进程的两个段，其余段要运行会发生缺段中断，其中每个驻留集大小为8，段大小大于这个值，则只能放入部分页，其余页的访问则会造成缺页中断。

进程创建后，段表信息和页表信息会给出，在ComboBox中选择进程，段表信息一览将会显示段表，点击相应段，该段的页表信息会在页表信息一览显示出来，左侧最下的文本框输出相应的提示信息

如果选择的段不在内存中，界面右侧上部的CheckBox缺段中断的状态会变为选中，同时SegReplace按钮的可见性打开（之前是关闭的）如果点击SegRePlace按钮，则会根据置换算法，将主存中的相应段调出，放入置换段，页类似，如果段大小大于8（驻留集大小）当选择不再内存中的页时缺页中断，PageReplace按钮事件，调出相应页，会在右侧下方的内存状态信息中显示相应的操作结果。

如果选择的页在内存中，则在物理地址一栏输出相应的物理地址

界面右侧最下部的页大小可以改变，算是对题目的扩展。

**四、主要数据结构及其说明**

List<Process> processes; //进程队列存放创建的所有进程

List<Duan> duans //Process的类成员，即每个进程的所有段信息

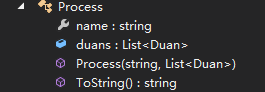
LinerX int[8] //驻留集，X从0取到7

Memory int[64] //主存模拟

Process类

*//属性成员name 进程名、duans List<Duan>类型的数据结构，存放每一段的情况*

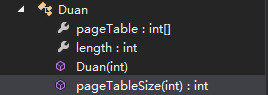
*//成员方法 Process() 构造方法、toString()返回进程名*



Duan类

*//属性成员length 每一段的长度信息，pageTable[] 为每一段分配的段表放在这里*

*//成员方法 Process() 构造方法、pageTableSize(int)根据分页的大小返回页表项的数目，这里是对题目做的扩展*



1. **程序运行时的初值和运行结果**

开始的时候创建2个进程，分别是banana（3,9,7）、orange（3,6,9），可以看到内存为banana和orange各分配了2个驻留集，内存第一个驻留集即内存信息第一行存放了banana第一个段3，第二行存放了8，因为第二段大于驻留集长度，只装了部分页进去。同理主存3，4行分给orange，第三行存放orange第一个段3，第四行存放6,第三段缺段没有在主存中

现在创建pear（1,4,2）可以预见主存5,6行将分配给pear，第五行前一个块的值会变成9（主存占用标志）第六行前4个块会置为9，第二段不能放入主存

选择banana进程，段表信息显示，选择第一段，显示页表信息，第一段都在主存中，随便显示一页，右侧显示出物理地址，这里显示12

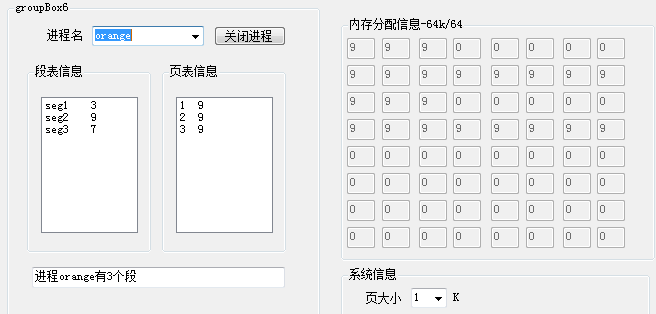
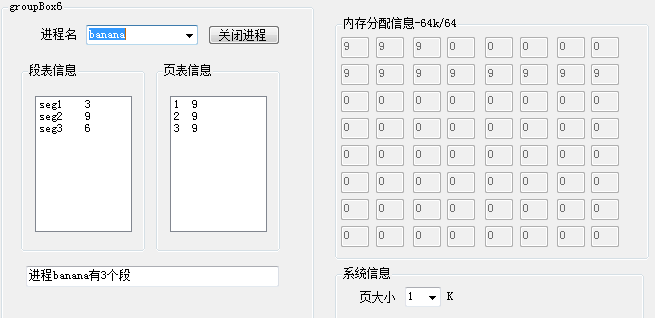
这时选择第三段，第三段没有在主存中，所以值都为0，缺段复选框置为选中态，替换按钮出现，点击发现原有的一号段为即主存第一行状态已经改变，7个框的值置为9，表示三段已经装入，3段所以位置已经变成9

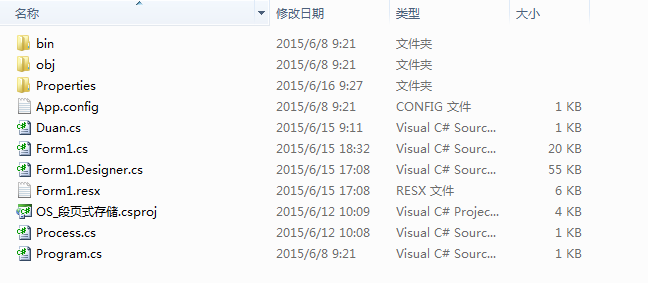
选择进程banana的第二段的最后，可以发现并不在主存中，点选按钮，主存第二段第一个框的值已经置为2，标志着页面置换成功

扩展，点击关闭进程按钮可以发现主存中相应驻留集已经清空

**六、源程序&注释**



**核心代码展示**

namespace OS\_段页式存储

{

public partial class Form1 : Form

{

List<Process> processList;

public static int pageSize=1;

int[] memory; //模拟主存

//8个驻留集

int[] Linger1;

int[] Linger2;

int[] Linger3;

int[] Linger4;

int[] Linger5;

int[] Linger6;

int[] Linger7;

int[] Linger8;

List<Process> ProcessListCanPutInMemory;

private void distributeMemory(Process tempProcess)

{

if (ProcessListCanPutInMemory != null&&ProcessListCanPutInMemory.Count<4)

{

ProcessListCanPutInMemory.Add(tempProcess);

}

else

{

}

for (int i = 0; i < ProcessListCanPutInMemory.Count; i++)

{

for (int j = 0; j < ProcessListCanPutInMemory[i].duans.Count; j++)

{

for (int k = 0; k < ProcessListCanPutInMemory[i].duans[j].length; k++)

{

if(i<4&&j<2&&k<8){

ProcessListCanPutInMemory[i].duans[j].pageTable[k] = 9;

}else{

}

}

}

}

try

{

Linger1 = ProcessListCanPutInMemory[0].duans[0].pageTable;

Linger2 = ProcessListCanPutInMemory[0].duans[1].pageTable;

Linger3 = ProcessListCanPutInMemory[1].duans[0].pageTable;

Linger4 = ProcessListCanPutInMemory[1].duans[1].pageTable;

Linger5 = ProcessListCanPutInMemory[2].duans[0].pageTable;

Linger6 = ProcessListCanPutInMemory[2].duans[1].pageTable;

Linger7 = ProcessListCanPutInMemory[3].duans[0].pageTable;

Linger8 = ProcessListCanPutInMemory[3].duans[1].pageTable;

}

catch { }

}

private void lb\_seg\_SelectedIndexChanged(object sender, EventArgs e)

{

isLackPage.Checked = false;

lb\_page.Items.Clear();

Process temp = (Process)cb\_ProcessList.SelectedItem;

string tempStr="";

for (int i = 0; i < temp.duans[lb\_seg.SelectedIndex].pageTable.Length; i++)

{

tempStr = tempStr + (i + 1) + " " + temp.duans[lb\_seg.SelectedIndex].pageTable[i];

lb\_page.Items.Add(tempStr);

tempStr="";

}

if (lb\_seg.SelectedIndex < 2)

{

isLackSeg.Checked = false;

SegPlace.Visible = false;

}

else

{

isLackSeg.Checked = true;

SegPlace.Visible = true;

}

showData();

}

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

{

if (cb\_ProcessList.SelectedItem != null)

{

if (cb\_ProcessList.SelectedIndex < 4)

{

int temp = cb\_ProcessList.SelectedIndex;

cb\_ProcessList.Items.RemoveAt(cb\_ProcessList.SelectedIndex);

switch (temp)

{

case 0:

for (int i = 0; i < ProcessListCanPutInMemory[1].duans.Count; i++)

{

for (int j = 0; j < ProcessListCanPutInMemory[1].duans[i].pageTable.Length; j++)

{

ProcessListCanPutInMemory[1].duans[i].pageTable[j] = 0;

}

}

break;

case 1:

for (int i = 0; i < ProcessListCanPutInMemory[1].duans.Count; i++)

{

for (int j = 0; j < ProcessListCanPutInMemory[1].duans[i].pageTable.Length; j++)

{

ProcessListCanPutInMemory[1].duans[i].pageTable[j] = 0;

}

}

break;

case 2:

for (int i = 0; i < ProcessListCanPutInMemory[2].duans.Count; i++)

{

for (int j = 0; j < ProcessListCanPutInMemory[2].duans[i].pageTable.Length; j++)

{

ProcessListCanPutInMemory[2].duans[i].pageTable[j] = 0;

}

}

break;

case 3:

for (int i = 0; i < ProcessListCanPutInMemory[3].duans.Count; i++)

{

for (int j = 0; j < ProcessListCanPutInMemory[3].duans[i].pageTable.Length; j++)

{

ProcessListCanPutInMemory[3].duans[i].pageTable[j] = 0;

}

}

break;

}

ProcessListCanPutInMemory.RemoveAt(temp);

showData();

}

}

}

private void lb\_page\_SelectedIndexChanged(object sender, EventArgs e)

{

physicalPosition.Text = "";

if (lb\_page.SelectedItem != null && lb\_seg.SelectedItem != null)

{

if (lb\_page.SelectedIndex > 7)

{

isLackPage.Checked = true;

isReplacePage.Visible = true;

}

else

{

isLackPage.Checked = false;

isReplacePage.Visible = false;

}

//如果不缺页不缺段打印物理地址

if (isLackPage.Checked == false && isLackSeg.Checked == false)

{

int tempInt=cb\_ProcessList.SelectedIndex \* 20 + (lb\_seg.SelectedIndex+1) \* 10 + (lb\_page.SelectedIndex+1);

physicalPosition.Text = tempInt.ToString();

}

}

showData();

}

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

{

for (int i = 0; i < ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[lb\_seg.SelectedIndex].pageTable.Length; i++)

{

ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[lb\_seg.SelectedIndex].pageTable[i] = 9;

}

for (int i = 0; i < ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[0].pageTable.Length; i++)

{

ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[0].pageTable[i]=0;

}

Linger1 = ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[lb\_seg.SelectedIndex].pageTable;

showData();

}

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

{

if (ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[lb\_seg.SelectedIndex].pageTable[0] == 9)

{

ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[lb\_seg.SelectedIndex].pageTable[0] = 2;

}

else if(ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[lb\_seg.SelectedIndex].pageTable[0] == 2)

{

ProcessListCanPutInMemory[cb\_ProcessList.SelectedIndex].duans[lb\_seg.SelectedIndex].pageTable[0] = 9;

}

showData();

}

//UI界面组

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

{

int[] temp = new int[16];

process\_Info.Text = "";

if (tb\_PID.Text != null)

{

try

{

temp[0] = Convert.ToInt32(seg1.Text);

temp[1] = Convert.ToInt32(seg2.Text);

temp[2] = Convert.ToInt32(seg3.Text);

temp[3] = Convert.ToInt32(seg4.Text);

temp[4] = Convert.ToInt32(seg5.Text);

temp[5] = Convert.ToInt32(seg6.Text);

temp[6] = Convert.ToInt32(seg7.Text);

temp[7] = Convert.ToInt32(seg8.Text);

temp[8] = Convert.ToInt32(seg9.Text);

temp[9] = Convert.ToInt32(seg10.Text);

temp[10] = Convert.ToInt32(seg11.Text);

temp[11] = Convert.ToInt32(seg12.Text);

temp[12] = Convert.ToInt32(seg13.Text);

temp[13] = Convert.ToInt32(seg14.Text);

temp[14] = Convert.ToInt32(seg15.Text);

temp[15] = Convert.ToInt32(seg16.Text);

}

catch { }

if (!isSomeDuanOver64Seg(temp))

{

List<Duan> tempDuans = new List<Duan>();

for (int i = 0; i < temp.Length; i++)

{

int j = 0;

if (temp[i] > 0)

{

Duan tempDuan = new Duan(temp[i]);

tempDuans.Add(tempDuan);

j++;

}

}

Process tempProcess = new Process(tb\_PID.Text, tempDuans);

processList.Add(tempProcess);

cb\_ProcessList.Items.Add(tempProcess);

clearUI();

distributeMemory(tempProcess);

showData();

}

else

{

process\_Info.Text = "有块大于64k";

clearUI();

}

}

}

public Form1()

{

InitializeComponent();

init();

}

private void init()

{

processList = new List<Process>();

pageSize = 1;

memory = new int[64];

for (int i = 0; i < memory.Length; i++)

{

memory[i] = 0;

}

showMemory();

Linger1 = new int[8];

Linger2 = new int[8];

Linger3 = new int[8];

Linger4 = new int[8];

Linger5 = new int[8];

Linger6 = new int[8];

Linger7 = new int[8];

Linger8 = new int[8];

//四个可以允许在内存中放的进程

ProcessListCanPutInMemory = new List<Process>();

}

private void cb\_pageSize\_SelectedIndexChanged(object sender, EventArgs e)

{

if (cb\_pageSize.SelectedItem != null)

{

try

{

pageSize = Convert.ToInt32(cb\_pageSize.SelectedItem.ToString());

}

catch { }

}

}

private bool isSomeDuanOver64Seg(int[] temp)

{

for (int i = 0; i < temp.Length; i++)

{

if (temp[i] > 64)

{

return true;

}

}

return false;

}

private void cb\_ProcessList\_SelectedIndexChanged(object sender, EventArgs e)

{

Process temp = null;

int duanNum = 0;

if (cb\_ProcessList.SelectedItem != null)

{

temp = (Process)cb\_ProcessList.SelectedItem;

lb\_seg.Items.Clear();

for (int i = 0; i < temp.duans.Count; i++)

{

if (temp.duans[i].length > 0)

{

string tempString = "seg" + (i + 1) + "\t" + temp.duans[i].length;

lb\_seg.Items.Add(tempString);

duanNum++;

}

}

}

process\_Info.Text = "进程" + temp.name + "有" + duanNum + "个段";

}

}

}