操作系统——文件管理

1854062 许之博

操作系统——文件管理
1.背景
1.1 项目目的
1.2 操作说明
1.3 开发环境
2.核心原理
2.1 内存块模拟
2.2 存储空间管理
2.3 文件物理结构
2.4 FCB
2.5 FCB目录
3. 主要功能实现
3.1 文件内容编辑
3.2 文件关键操作
3.2.1 打开文件
3.2.2 保存文件
3.3.3 新建文件
3.3.4 格式化
4.效果

1.背景

1.1 项目目的

结合操作系统课堂中所学的文件管理相关知识，模拟一个文件管理系统，实现文件管理的基本功能。

1.2 操作说明

1. 双击.exe文件打开文件管理项目，所有功能都集中在界面右上角的"文件"栏中。
2. 对文件或文件夹进行任何操作都需要单机选中文件或文件夹后，再打开“文件”选择要进行的操作，本项目中没有任何双击功能(如双击打开文件或文件夹)和右键功能。
3. 左侧的树形目录只是对文件系统结构的展示，选中其中的内容没有作用。对文件进行操作需要单击选中右侧当前目录栏中的文件。
4. 格式化功能清空整个文件系统中的内容。剪切，复制，粘贴功能都是对整个文件或文件夹进行的操作。
5. 本项目中实现了对数据的保存，当对文件系统进行修改后退出，再打开，修改的记录仍然存在。与exe在同一文件夹中的还有.dat文件，存储保存的数据，没有.dat文件，exe不能正常运行。
6. 通过前进或后退功能可以回退或前进到新的目录，父目录功能及返回到当前目录的父目录。
7. 在修改文件过后，若不在文本编辑界面的左上角“文件”处选择“保存”，则修改不会保存。

1.3 开发环境

- 开发平台：Windows 10
- 开发软件：Visual Studio Community 2019 16.9.4
- 开发语言：c#

2.核心原理

2.1 内存块模拟

MemorySpace类实现对某一个内存数据块的模拟，每个内存数据块可以存储长度为64位的字符串。对文件内容进行的读写，最终都是转化为对内存块数据的读写，同时内存数据块也是最终存储数据的结构。MemorySpace类包括最基本的获取数据和写入数据功能。

```
public class MemorySpace
    {
        public static int memorySize = 64;//内存块存储的字符串的最大长度
        private char[] fileData = new char[64];//内存块数据
        private int fileSize = 0;//内存块中已经被使用的空间大小
        public MemorySpace(){ }
        public string getData();//获取内存块中内容
        public void setData(string data);//写入内存块
    }
```

2.2 存储空间管理

建立一个BlankSpace类实现对所有内存数据块的管理。

- 建立一个位图bool类型数组bitMap用来表示各个数据块是否被使用。
- 该类的核心是一个内存数据块数组spaces,类型为2.1中定义的MemorySpace类
- 在类中也定义两个变量用于表示存储空间的容量和已经被使用的内存数据块数目。
- 实现从某一块内存数据块中取内容。
- 可以清空下标为i的内存数据块的内容，也可以清空某一文件中所有内存数据块的内容。
- 为某一文件中的内容建立文件结构FileStructrue(2.3提及），也即将输入的内容保存到内存块中。

```
   public class BlankSpace//管理空白区
    {
        public int usedSpace = 0;//已经使用的空间
        public static int capacity = 10 * 1024;//10k空间
        private bool[] bitMap = new bool[capacity];//位图，描述内存块使用情况
        private MemorySpace[] spaces = new MemorySpace[capacity];//所有内存块
        public BlankSpace();
        public string getDataFromMemory(int i);//获取
        public void FreeSpace(int id);//释放下标为id的内存块空间
        public void FreeSpace(List<int>ids);//释放多个内存块空间
        public FileStructrue CreateSpaces(string data);//根据文件内容分配内存空间
        public int newSpace(string data);//根据data新分配一个内存块
    }
```

2.3 文件物理结构

基于2.1中设计的内存数据块类，一个文件的内容最终都是存储在内存数据块之中。因此为每一个文件建立一个物理结构管理类FileStructrue，用来管理该文件用到的内存数据块的索引号，根据索引号可以在内存空间BlankSpace中找到对应的内存数据块。本项目中选用的文件物理结构是多级索引结构，为了使文件空间足够大，选用三级索引结构。二级索引FirstIndex与三级索引SecondIndex都是FileStructrue的属性。FileStructrue自带一个index数组，然而只能存储十个内存块的索引，当空间不够时，会开辟二级索引与三级索引空间，一个FirstIndex的容量为128，SecondIndex又包含有128个FirstIndex，容量为128*128，故FileStructrue可以调用的内存块的最大数目为10+128+128*128。

- FileStructrue包含一个getSpaceList方法，外界可以获取该文件所调用的所有内存块的索引号列表。
- 当为文件增调内存数据块时，将内存数据块的索引通过AddIndex方法加入到FileStructrue中。

```
public class FileStructrue//索引结构
    {
        public static int capacity = 10;
        private int[] index = new int[capacity];
        public int usedSize =0;
        private FirstIndex fIndex;//所属于的一级索引
        private SecondIndex sIndex;//所属于的二级索引
        public FileStructrue();
        public bool checkFull();
        public bool AddIndex(int id);
        public List<int>getSpaceList();//获取全部索引项
    }
public class FirstIndex
    {
        public static int capacity = 128;
        public int[] index = new int[capacity];
        public int usedSize = 0;
        public FirstIndex();
        public void AddIndex(int id);
        public bool checkFull();
    }
public class SecondIndex
    {
        public static int capacity = 128;
        public List<FirstIndex>fIndex
            = new List<FirstIndex> { new FirstIndex() };
        public int usedSize = 0;
        public SecondIndex();
        public bool checkFull();
        public void AddIndex(int id);
    }
```

2.4 FCB

FCB是文件管理系统的核心，他包括一个文件的基本属性，如编号，名称，类型，大小等。本项目中的文件类型只有两种，文本文件与文件夹，以下分这两种进行讨论。

文本文件：

- 核心是文件物理结构FileStructrue类型的变量blocks，其中包含该文件所调用的内存块的索引号，而文件的内容都存在内存块中。

文件夹：

- 由于一个文件夹可以包括文件夹与文件，因此形成了一个多叉树的结构，文件夹是树中的内部节点，文本文件是树中的叶子节点，但无论是文件夹还是文本文件都是FCB类型。根据多叉树的特性，因此想使用左孩子，右兄弟的方法模拟多叉树结构。
- 包含父目录指针parent，子目录指针child，后继兄弟指针nextBro，前驱兄弟指针preBro。指向不同的FCB对象，可能是文本文件也可能是文件夹。

```
public class FCB//文件的目录结构
    {
        public int fileID;//文件编号
        public string fileName;//文件名称
        public DateTime editTime;//修改时间
        public int fileSize;//文件大小
        public enum fileType {txt,folder};//文件类型
        public fileType type;
        public int fileNum=0;//如果是文件夹，包含的文件个数
        public FileStructrue blocks;//文件物理结构

        public FCB parent = null, child = null;//父目录指针和子目录指针
        public FCB nextBro=null,preBro=null;//左右兄弟文件指针

        public FCB(FCB fCB);//从fCB复制文件
        public FCB(string name,fileType tp);//新建名称为name类型为tp的文件
        public void Delete();//删除文件以及包含的子内容
        public void addChild(FCB child);//文件夹状态下添加子内容
        public string getPath();//获取文件所在路径
    }
```

2.5 FCB目录

文件系统中的所有FCB对象都有一个fileID编号，在FCBIndex中可以对所有的FCB进行同一管理。

```
public class FCBIndex//管理FCB
    {
        public Dictionary<int, FCB> fileIndex = new Dictionary<int, FCB>();
        //存储FCB
        public FCB getFCB(int num);//获取fileID为num的FCB
        public void Delete(FCB fCB);//删除fCB
        public void Add(FCB fCB);//添加fCB
    }
```

3. 主要功能实现

3.1 文件内容编辑

核心部分为以下两部分：

- showTexts：将FCB中内容写到界面中，根据FCB中的FileStructrue存储的所调用的内存块的索引号找到所有的内存块，并获得其中存储的内容。
- saveToMemory：首先获取FCB中所调用的所有内存块的索引，释放这些内存块，然后再将界面中的文本内容(string)写入存储空间中的内存块中，最后FCB获得这些内存块的索引。

```
public partial class FileEditor : Form
    {
        private FCB fCB;
        private BlankSpace blankSpace = MainPage.blankSpace;
```

```
        private bool dirt = false;
        public FileEditor()
        {
            InitializeComponent();
        }
        public FileEditor(FCB fcb)
        {
            InitializeComponent();
            this.fCB = fcb;
            this.Text = fcb.fileName;
            showTexts();
        }
        private void showTexts()
        {
            List<int> indexs = fCB.blocks.getSpaceList();
            string content = "";
            foreach (int i in indexs)
            {
                content += blankSpace.getDataFromMemory(i);
            }
            richTextBox1.Text = content;
        }
        private void saveToMemory()
        {
            string content = richTextBox1.Text;
            fCB.fileSize = content.Length
            List<int> indexs = fCB.blocks.getSpaceList();
            blankSpace.FreeSpace(indexs);// 先清空原来的所有数据块
            fCB.blocks = blankSpace.CreateSpaces(content);
            // 先清空原来的所有数据块
        }
        private void 保存ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            saveToMemory();
            dirt = false;
            this.Text = fCB.fileName;
        }
        private void 关于ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Code by 许之博 (1854062)", "关于",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        private void 退出ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void Editor_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (dirt && MessageBox.Show("是否保存文件？", "提示",
MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                saveToMemory();
            }
        }
    }
```

3.2 文件关键操作

3.2.1 打开文件

```
private void Open(FCB fcb, bool isBack = false)//打开文件
        {
            switch (fcb.type)
            {
                case FCB.fileType.folder://文件夹类型
                    (isBack ? furtStack : pastStack).Push(currentContent);
                    currentContent = fcb;
                    comboBox1.Text = currentContent.getPath();
                    UpdateListView(fcb);
                    break;
                case FCB.fileType.txt://文本文件类型
                    FileEditor fileEditor = new FileEditor(fcb);
                    //打开文本编辑界面
                    fileEditor.FormClosed += (s, e) => { this.UpdateView(); };
                    fileEditor.Show();
                    break;
                default:
                    break;
            }
        }
```

3.2.2 保存文件

```
public void Serialize()//保存文件，保存到root.dat，fcbIndex.dat，blankSpace.dat
{
   FileStream fileStream;
   BinaryFormatter b = new BinaryFormatter();
   string dir = Directory.GetCurrentDirectory();

   fileStream = new FileStream(Path.Combine(dir, "root.dat"), FileMode.Create);
   b.Serialize(fileStream, root);
   fileStream.Close();

   fileStream = new FileStream(Path.Combine(dir, "fcbIndex.dat"),
FileMode.Create);
   b.Serialize(fileStream, fcbIndex);
   fileStream.Close();

   fileStream = new FileStream(Path.Combine(dir, "blankSpace.dat"),
FileMode.Create);
   b.Serialize(fileStream, blankSpace);
   fileStream.Close();
}
```

3.3.3 新建文件

```
private void 文件ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    string file_name = NameCheck("新建文本文件.txt");// 创建一个FCB
    FCB fcb = new FCB(file_name, FCB.fileType.txt);
```

```
    currentContent.addChild(fcb);// 将FCB加入到FCB表中
    fcbIndex.Add(fcb);
    UpdateView();
}
private void 文件夹ToolStripMenuItem_Click(object sender, EventArgs e)
{
    string file_name = NameCheck("新建文件夹");// 创建一个FCB
    FCB fcb = new FCB(file_name, FCB.fileType.folder);
    currentContent.addChild(fcb);// 将FCB加入到FCB表中
    fcbIndex.Add(fcb);
    UpdateView();
}
```

3.3.4 格式化

```
private void 格式化ToolStripMenuItem_Click(object sender, EventArgs e)
{
    currentContent = root = new FCB("我的文件", FCB.fileType.folder);
    blankSpace = new BlankSpace();
    fcbIndex = new FCBIndex();
    UpdateView();
}
```

4.效果


FileManagement
文件
关于
帮助
我的文件
操作系统
电梯调度
内存管理
文件管理
数据库
期末复习
课件
算法
计算机系统结构
ddl
前进
后退
我的文件
父目录
名称
大小
类型
修改时间
操作系统
0 字节
文件夹
2021/6/21 21:04:53
数据库
0 字节
文件夹
2021/6/21 20:37:14
算法
0 字节
文件夹
2021/6/21 20:34:01
计算机系统结构
0 字节
文件夹
2021/6/21 20:35:00
ddl
51 字节
文本文件
2021/6/21 21:48:19
ddl
文件
关于
6.28 操作系统考试
6.29 毛概考试
7.1 数据库考试
7.2 系统结构考试
7.9 算法考试
系统总空间：640KB 已用空间：256B 剩余空间：639KB