中 国 地 质 大 学

本科生课程论文封面

课 程 名 称 C#程序设计

教 师 姓 名 杨鸣

本科生姓名 郑召作

本科生学号 20161001451

本科生专业 信息安全

所 在 院 系 计算机科学与技术

日期: 2019/11/10

课程设计评语

对课程论文的评语:

平时成绩：	课程论文成绩：
总 成 绩：	评阅人签名：

注：1、无评阅人签名成绩无效；
2、必须用钢笔或圆珠笔批阅，用铅笔阅卷无效；
3、如有平时成绩，必须在上面评分表中标出，并计算入总成绩。

目 录

1. 课程论文题目

监视文件和文件夹的变化（FileSystemWatcher 类）：选择一个文件夹，单击“开始监视”，该文件夹处于监视状态，此时对该文件夹如果进行操作，这些行为将显示出来。

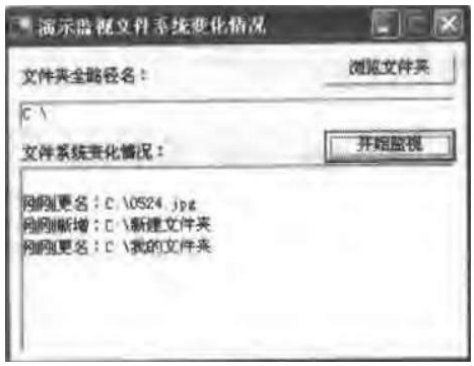

图 1 题目例图

2. 实验原理

本文将使用 FileSystemWatcher 组件监视文件系统，并对文件系统的改变作出反应。通过使用 FileSystemWatcher 组件，在特定的文件或目录被创建、修改或删除时，可以快速和便捷地启动业务流程。例如，如果一组用户在合作处理一个存储在服务器共享目录下的文档时，可以使用 FileSystemWatcher 组件编写应用程序来监视对共享目录的更改情况。当检测到更改时，该组件可以运行处理过程，通过电子邮件通知每个用户。

可以配置组件来监视整个目录及其内容，或特定目录下一个特定的文件或一组文件。若要监视所有文件中的更改，应将 Filter 属性设置为空字符串（""）；若要监视特定的文件，应将 Filter 属性设置为该文件的文件名（例如，若要监视文件 MyDoc.txt 中的更改，将 Filter 属性设置为"MyDoc.txt"）；也可以监视特定文件类型中的更改，例如若要监视文本文件中的更改，将 Fillter 属性设置为"*.txt"。

因此本文通过创建一个 FileSystemWatche 组件来监视运行时指定的目录。组件被设置用来监视 LastWrite 和 LastAccess 时间的更改，以及目录中文本文件的创建、删除或重命名。如果文件被更改、创建或删除，则文件的路径就会被输出到控制台。当一个文件被重命名时，旧的和新的路径都被输出到控制台。

3. 实验截图

本文件夹监控软件由一个 Form 窗体、两个 Label、两个 Button 以及两个 TextBox 组成，如图 2 所示：

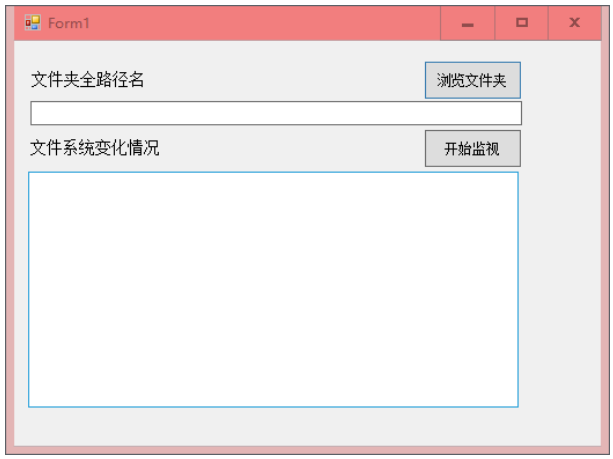

图 2 应用界面图

点击文件夹监控软件的浏览文件夹按钮，如图 3 所示：

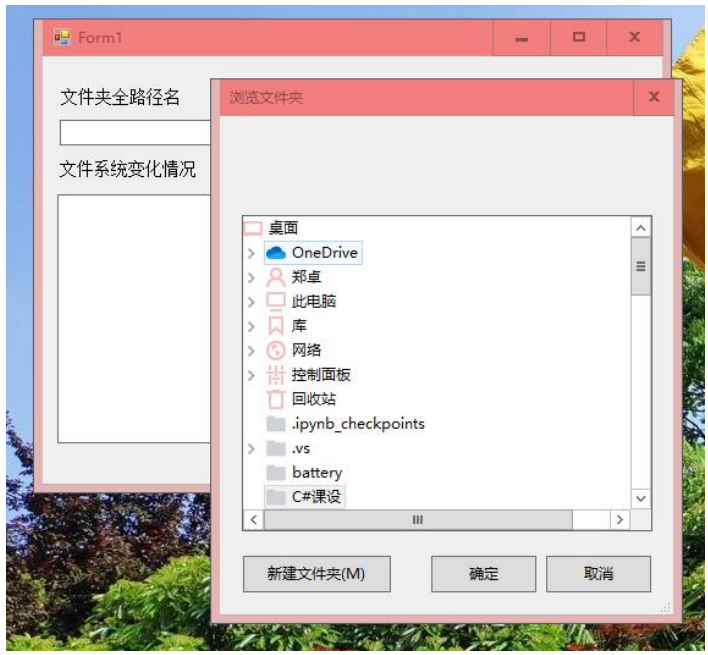

图 3 选择文件

在浏览文件夹窗体里选择待监控的文件夹，选择结束后，点击确认。可以看到在文件夹全路径名下的文本框出现了刚刚选择的文件夹路径：

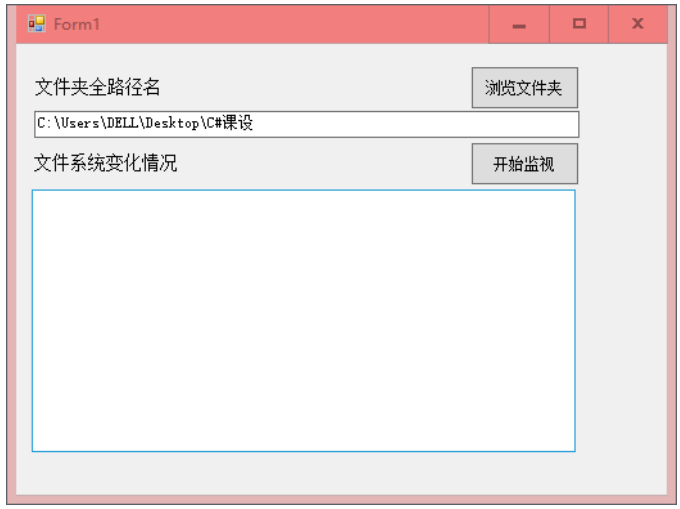

图 4 路径选择结束

我们进入所选择的文件夹，可以看到里面有三个文件，分别是 1.txt，2.txt，3.jpg，接下来我们对这三个文件分别进行操作：

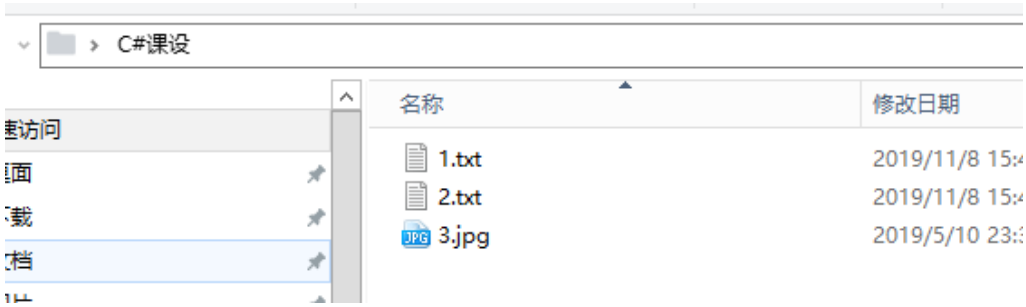

图 5 初始文件信息

点击文件夹监控软件的开始监视按钮，如图 6 所示：

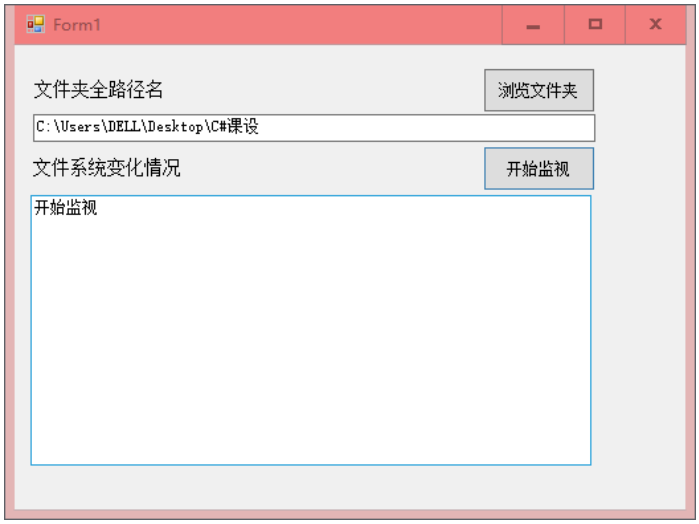

图 6 开始监视

接着，我将 1.txt 更名为 new_1.txt，文件系统变化情况显示如下：

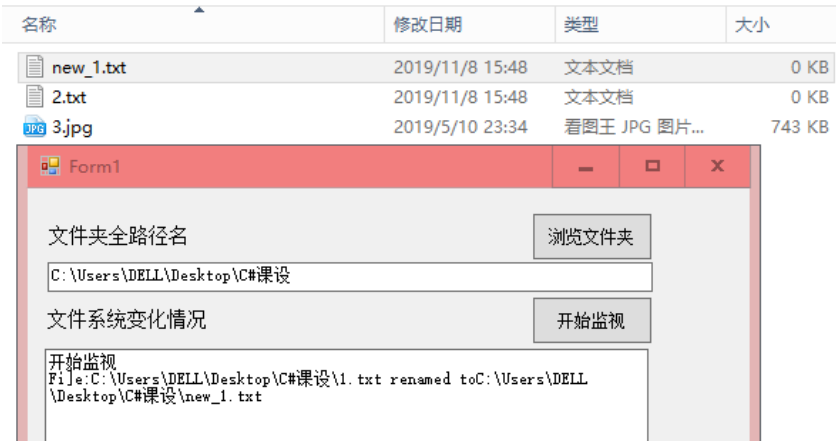

图 7 文件更名

接着，我将 2.txt 进行删除，文件系统变化情况显示如下：

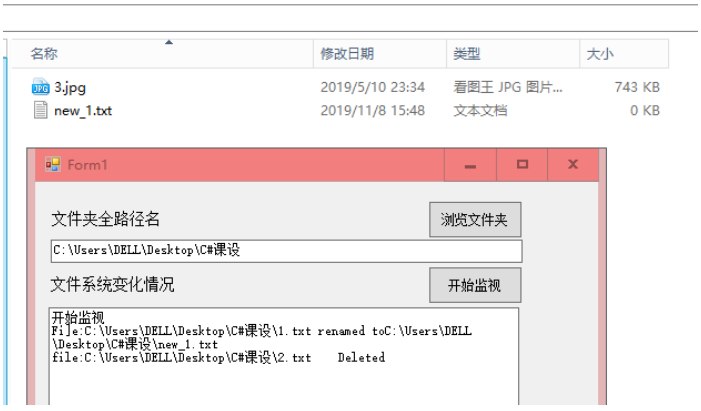

图 8 文件删除

接着，我将 3.jpg 更名为 3.png，文件系统变化情况显示如下：

图 9 文件更名

最后，我新建了一个 4.txt 文件，文件系统变化情况显示如下：

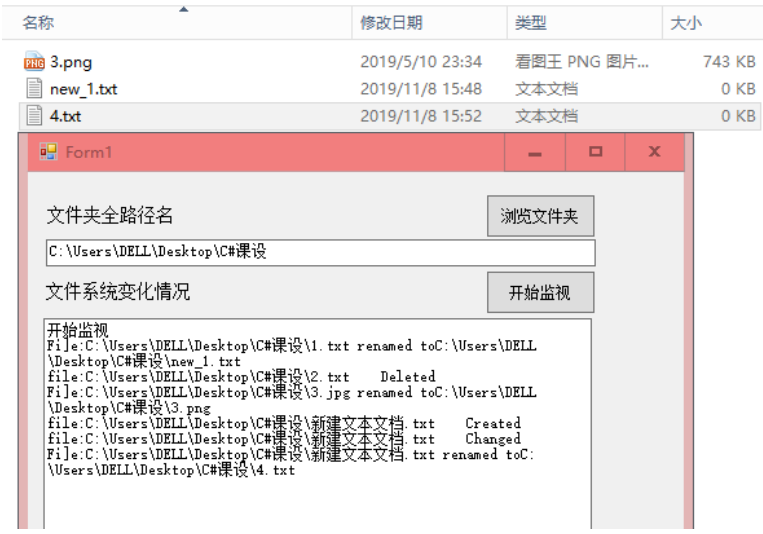

图 10 新建文件

4. 遇到的问题

1、未处理 InvalidPoerationException：由于线程间对控件的操作无效，即尝试通过不是创建控件的线程访问该线程。这是由于，随着程序复杂度的提高，程序不可避免会出现多个线程，此时就很可能存在跨线程操作控件的问题，但跨线程调用窗体控件不被编译器允许。

图 11 遇到问题

解决方案：通过在窗体的创建或者 load 方法中加入 Control.CheckForIllegalCrossThreadCalls=false;就可以跨线程调用了。CheckForIllegalCrossThreadCalls 容许子线呈随时更新 UI,，在同一个 test 函数体内，不能保证自身事务的一致性。

2、非静态的字段、方法或属性要求对象引用：说明该类的中方法没有 static，不是静态的，或者有非静态的字段、方法或属性，要求该类必须实例化。

解决办法：我将所定义的事件处理程序全部改为非静态的，如 public void OnCreated(object sender, FileSystemEventArgs e)

5．实验总结

在学习C#的一个学期以及完成了两次的课程实习后，我们进行了C语言课程设计，尝试编写一个比较复杂的程序系统。我抽到的题目是实现一个文件与文件夹的监控。在为期一周的时间中，我的感受是：C#课程设计和平时上课所接触的程序以及前几次上机实习是有很大不同的，所经受的考验和克服的困难是平时所无法比拟的。在这段时间内，我和同学们一起交流，共同讨论。在完成之后，感触良多。

虽然这只是一个很小的项目，但是接触的方面有很多，比如 Windows 编程、文件操作、事件处理。遇到了蛮多问题是上课时老师没有教授的，但是通过网络查询到了相应的解决方案。在课程中深切体会到了教师认真负责的伟大的精神和热情为同学指导的促学方式，虽然对有些时候教师没给我们指出解决问题的方法有些小抱怨，但是到了结束时才知道，这种教学让我们自己学会了自学，学会了去看懂别人的代码。

通过这次实训，也使我们发现了许多问题。 在实训中，我们认识到自己还有很多的知识没学好，基础知识没理清，而且许多东西还要去翻书，去上网搜索。而且遇到一些小错误运行不出来，就会烦躁不安，觉得有些自暴自弃或者抱怨项目的变态，以后要克服，尽量保持一颗良好的心态。

6．实验代码

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace FileMonitor
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
```

```
            Control.CheckForIllegalCrossThreadCalls = false;
        }
        private void label1_Click(object sender, EventArgs e){
        }
        private void textBox1_TextChanged(object sender, EventArgs e) {
        }
        private void label2_Click(object sender, EventArgs e) {
        }

        private void button1_Click(object sender, EventArgs e) {
            int i;
            string[] filen;
            string filea;
            textBox1.Clear();
            FolderBrowserDialog folderBrowserDialog1 = new FolderBrowserDialog();
            folderBrowserDialog1.ShowDialog();
            textBox1.Text = folderBrowserDialog1.SelectedPath;
            if (folderBrowserDialog1.SelectedPath == "") return;
            if (!Directory.Exists(folderBrowserDialog1.SelectedPath))
                MessageBox.Show(folderBrowserDialog1.SelectedPath + "文件夹不存在", "信息提示",
MessageBoxButtons.OK);
            else
            {
                filen = Directory.GetFiles(folderBrowserDialog1.SelectedPath);
                Console.WriteLine("ni"+filen);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string filea = textBox1.Text;
            FileSystemWatcher watcher = new FileSystemWatcher();
            watcher.Path = @filea;
            textBox2.AppendText("开始监视\n");


            /*监视LastAcceSS和LastWrite时间的更改以及文件或目录的重命名*/
            watcher.NotifyFilter = NotifyFilters.LastAccess | NotifyFilters.LastWrite |
                    NotifyFilters.FileName | NotifyFilters.DirectoryName;
            //只监视文本文件
            //watcher.Filter = "*.txt";
```

```
            //添加事件句柄
            //当由FileSystemWatcher所指定的路径中的文件或目录的
            //大小、系统属性、最后写时间、最后访问时间或安全权限
            //发生更改时，更改事件就会发生
            watcher.Changed += new FileSystemEventHandler(OnChanged);
            //由FileSystemWatcher所指定的路径中文件或目录被创建时，创建事件就会发生
            watcher.Created += new FileSystemEventHandler(OnChanged);
            //当由FileSystemWatcher所指定的路径中文件或目录被删除时，删除事件就会发生
            watcher.Deleted += new FileSystemEventHandler(OnChanged);
            //当由FileSystemWatcher所指定的路径中文件或目录被重命名时，重命名事件就会发生
            watcher.Renamed += new RenamedEventHandler(OnRenamed);
            //开始监视
            watcher.EnableRaisingEvents = true;
            //等待用户退出程序

        }



        //定义事件处理程序
        public    void OnCreated(object sender, FileSystemEventArgs e)
        {
            textBox2.AppendText("file:" + e.FullPath + "       " + e.ChangeType + "\n");
        }

        //定义事件处理程序
        public void OnChanged(object sender, FileSystemEventArgs e)
        {

            textBox2.AppendText("file:" + e.FullPath + "       " + e.ChangeType + "\n");
        }
        public    void OnRenamed(object sender, RenamedEventArgs e)
        {
            textBox2.AppendText("Fi]e:"+e.OldFullPath+" renamed to"+ e.FullPath + "\n");
        }

        private void textBox2_TextChanged(object sender, EventArgs e)    {
        }
    }
}
```