文章编号：2096-1472(2016)-07-22-04

使用Windows服务实现文件夹同步的技术研究

邹 珺
（苏州农业职业技术学院，江苏 苏州 215008）

摘 要：为了让文件夹能够实时得以同步，需要一个工具不停地监控两个文件夹的异同，该工具实现了一个专门的Windows服务来实现文件夹的同步功能。本文主要描述文件夹同步服务程序的技术研究指定要同步的源文件夹，同步到的目标文件夹，设置同步项目、同步时间间隔，服务程序将根据指定的分钟数自动进行文件夹的同步。当同步服务完成后，可以看到同步的状态信息，比如已更新的文件个数或已删除的文件个数等。

关键词：文件夹；同步；监控；Windows服务
中图分类号：TP312 **文献标识码**：A

Technology Research of Realization of Folder Synchronization Making Use of Windows Service

ZOU Jun
(*Suzhou Agricultural Vocational College*,*Suzhou* 215008,*China*)

Abstract: In order to allow the folder to be synchronized in real time,a tool is needed to monitor the similarities and differences between the two folders,which implements a dedicated Windows service to achieve the synchronization of folders.The paper mainly describes the folder synchronization service program,designated to synchronize the source folder, synchronized to the target folder,set the synchronization,the synchronization interval,service procedures will be according to the specified number of minutes automatic folder synchronization.When the synchronization service is complete,you can see the status of the synchronization information,such as the number of files that have been updated or deleted files,etc..

Keywords:folder;synchronization;monitor;windows service

1 引言(Introduction)

在很多场合，用户需要在两个文件夹之间维持同步的工作(比如更新Web文件)，或基于安全的原因备份某个文件夹中的内容到其他文件夹等。一般的做法是使用Windows资源管理器手工实现文件夹的同步，但是人们时常忘记这一工作。

要实现两个文件夹同步，必须要有一个监控工具时刻监测这两个文件夹，并比较它们的不同之处。当两个文件夹其中一个发生变化，比如对某个文件进行修改后，同步工具能完成实时同步。事实上这种工具目前的需求量很大，很多文件夹需要与多个目标位置保持同步，如果通过手工实现同步，其弊端主要包括工作量大，容易出错等。

本文主要研究的是一个功能强大的文件夹同步服务程序，用户可以指定要同步的一个或多个文件夹，指定同步的时间，程序在一个Windows服务后台进行检测，将文件夹从一个源位置同步到目标位置。

2 文件夹同步功能概述(Summary of folder synchronization)

文件夹同步功能的技术提供了一个同步服务配置工具，这是一个Windows Forms项目，实现对服务的配置，配置结果保存为XML文件格式。同时为了调试Windows服务，程序实现了一个服务控制台工具。调试一个Windows服务非常麻烦，这个控制台程序使程序员可以调试包含在Windows服务中的核心代码的实现，不用去频繁地安装卸载服务。这个控制台程序还可以与文件服务配置程序进行通信。使用WCF命令管道在两个进程之间进行通信[1]。该系统组成结构如图1所示。

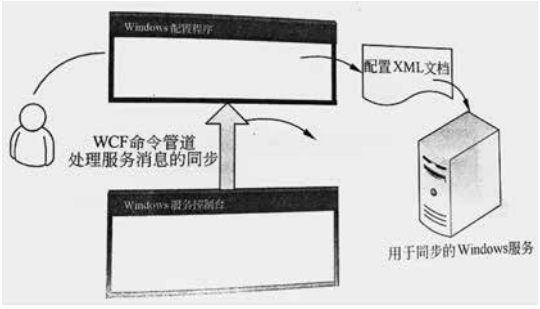

图1 文件同步工具的系统组成结构

Fig.1 System composition structure of file synchronization tool

3 文件夹同步主要功能(Main function of folder synchronization)

文件夹同步的主要功能通过一个类库项目SynchroLib实现，便于多次重用。该项目中的对象包括同步项集合对象、后台同步线程等。

3.1 实现文件夹同步项集合对象

SyncItemCollection集合实际上是一个泛型的

List<SyncItem>对象。但是SyncItemCollection需要具有从XML元素中获取同步对象的能力，以及允许用户开始所有同步对象的更新工作，因此从List<SyncItem>派生，也可以实现一个雷，在内部包含一个泛型的List<SyncItem>集合来处理多个同步对象[2]。

SyncItemCollection对象有一个XElement类型的属性，该属性将根据在构造函数中传入XML元素来解析出多个SyncItem对象并加载到List集合中。该属性的定义代码如下：

```
//使用XElement获取或者是设置SyncItem集合
public XElement XElement
{
  get
  {     //实例化一个新的XElement对象
XElement value=new XElement(“SyncItems”);
foreach(SyncItem item in this)
{   //将同步对象中的XML元素添加到XElement元素集合
  value.Add(item. XElement);
}
return value;                  //返回XElement对象实例
  }
  set
  {
if(value!=null)
{   //遍历XElement的子元素
  foreach(XElement element in value.Elements())
   {   //根据XML元素得到SyncItem对象实例，添加到集合
     this.Add(new SyncItem(element));
  }
}
  }
}
```

get设置区域中，通过实例化一个元素名称为SyncItems来构造一个XML片段。然后遍历集合中SyncItem对象，将SyncItem的XElement属性返回的XML片段加入到该XElement的子元素集合中。在set设置器中，通过遍历XElement的子元素集合来实例化新的SyncItem对象，再添加到泛型集合中构造了同步集合。

3.2 实现后台同步线程

SyncFiles实现了同步文件的操作，该方法将根据是否能进行同步工作来实现同步，实现代码如下：

```
//同步文件夹的线程委托，更新目标文件夹中的文件，且在完成时触发事件
private void SyncFiles()
{
  if(this.CanStartSync)          //如果允许文件夹同步
  {
try
{
    DateTime before=DateTime.Now;//得到同步前的时间
    //将源文件夹中的文件更新到目标文件夹中
     this.ToFilesList.Update(this.SyncFromPath,this.SyncSubfolders);
    DateTime after=DateTime.Now;   //更新后的时间
     TimeSpan elapsed=after-before;//得到所花费的时间
  int updates=this.ToFilesList.Updates;//得到更新的个数
  FileInfoEvent(this,new FileInfoArgs(updates,elapsed));      //触发更新完成事件
  }
  catch(ThreadAbortException ex)    //如果更新触发异常
  {
  if(ex!=null){}      //异常处理代码
  }
  catch(Exception)
  {
  throw;      //重新抛出
  }
  }
}
```

这个方法最核心的部分在于使用ToFilesList的Update将源文件夹中的文件更新到目标文件夹，更新前和更新后都记录了当前时间，以便记录下更新所花费的时间，并且记得更新的文件个数。在更新完成后，将调用FileInfoEvent事件处理代码[3]。

4 文件夹同步关键技术(Key technology of folder synchronization)

4.1 使用WCF开发命名管道程序

进程间通信(IPC)的应用非常广泛，其特点是消息无须跨越防火墙和主机。IPC是一种通信的方法，源于UNIX操作系统。在Windows操作系统中，使用进程间通信除了命名管道，还可以使用剪贴板、邮件槽、TCP/IP通信、内存映射文件等技术[4]。

WCF本身提供了NetNamedPipeBing对象，使创建命名管道变得非常简单。WCF把通信都进行了统一化，假如熟悉WCF开发，完全不用去学习与命名管道相关的细节，由WCF实现管道的创建与通信工作。

使用NetNamedPipeBinding绑定与其他类型绑定的不同之处在于路径的指定，例如使用地址：net.pipe://localhost/SynchroServiceWCF。地址中的net.pipe对应命名管道协议，任何使用了命名管道传输通道的绑定都使用net.pipe作为地址的协议部分。localhost是地址的主机部分，而SynchroServiceWCF是一个可选项，是为了使系统可读，通常是服务名称，命名应该与服务内容相关，让人更清楚服务的作用[5]。

4.2 实现监控和配置项目

配置和监控Windows Forms项目提供了用户界面与用户交互。该项目提供了如下功能让用户配置文件同步服务，以及查看由Windows服务返回的文件同步结果消息。

(1)允许添加和修改同步项，以及用于文件同步所需要的一些设置选项，这些选项将被保存到XML配置文件中。

(2)允许启动和停止Windows服务，以及作为WCF服务宿主来监听来自Windows服务的同步结果消息。

(3)允许安装和卸载Windows服务，并能检查Windows服务是否安装。

实现这个项目使文件同步程序能够提高使用性，也便于查看在文件同步过程中到底是成功还是失败，能深入了解文件同步的过程[6]。

4.2.1 在主窗口中启动或停止服务

用户主界面的启动、停止服务都将调用Globals静态类中定义的相关的方法来启动和停止服务。当服务启动后，如果Windows服务实现了同步的操作，将使用命名管道发送同步消息，WCF服务受到消息，触发Form1_SynchroHostEvent事件处理代码，代码如下：

```
void Form1_SynchroHostEvent(object sender,SynchroHostEventArgs e)
{
    e.Date=e.Date.ClearSeconds();
//使用扩展方法清除日期中的秒数
    string dateFormat="yyyy-MM-dd HH:mm";
//定义显示的日期格式
    if(this.listBoxActivity.Items.Count>0)
//移除任何大于24小时的记录
    {
bool deleted=false;
do
{
//循环移除日期的最后一条
    deleted=false;
    //获取ListBox中最后一个Item的索引号
    int lastItem=this.listBoxActivity.Items.Count-1;
    //得到最后一个ListViewItem的文本
    string oldestMsg=this.listBoxActivity.Items[lastItem].ToString();
    //如果字符串不为空或null且字符串的长度大于16个
    if(!string.IsNullOrEmpty(oldestMsg)&&oldestMsg.Length>16)
    {
      oldestMsg=oldestMsg.Substring(0,16);    //进行截止到16个为止
      DateTime datetime;
      //使用特定的格式解析日期
      if(DateTime.TryParse(oldestMsg,out datetime))
      {
        TimeSpan elapsed=datetime-e.Date;    //获取时间差
        if(elapsed.Days>0)          //如果天数大于0，表示多过一天
        {
          this.listBoxActivity.Items.RemoveAt(lastItem);
          deleted=true;          //标志已经删除
        }
      }
    }
    else
    {            //否则直接移除最后一行字符串
      this.listBoxActivity.Items.RemoveAt(lastItem);
      deleted=true;          //标志已删除
    }
  }while(deleted);    //当deleted标记为false则退出循环
    }
    //在ListBox的顶部插入新的文本信息
    string msg=string.Format("{0}-{1}",e.Date.ToString(dateFormat),e.Message);
    this.listBoxActivity.Items.Insert(0,msg);
}
```

在代码中，首先使用扩展方法ClearSeconds清除日期时间中的秒数，再判断ListBox中是否存在记录。如果存在，则在一个循环do中进行删除操作。在循环体中，总检查ListBox中的最后一行，得到最后一行显示的文本，先获取表示日期时间的前16个字符串，使用DateTime.TryParse将其转换为日期格式。然后判断该时间与从服务器端当前返回的时间是否相差多过一天，如果大于0，表示超过24小时，则进行项的移除；当delete标志为false时，表示没有大于1天的日志，则退出循环。最后将从WCF服务中传回的小时插入到ListBox中的顶部，即第0行[7]。

4.2.2 使用同步项更新用户界面

FormAddSyncItem类重载了默认的构造函数，用来接收一个List<string>集合和一个要编辑的SyncItem。一旦SyncItem被传入，会使用同步对象中的属性来更新用户界面。FormAddSyncItem类的构造函数代码如下：

```
public FormAddSyncItem(SyncItem item,List<string> names)
{
  m_existingNames=names;                    //得到同步项名称集合
  m_adding=(item==null);                    //判断是否为添加还是编辑状态
  this.SyncItem=item;                       //赋同步项对象实例
  InitializeComponent();
  //如果是添加一个新的项，需要提供一个唯一的名称以免用户输入重复的同步项名称
  if(m_adding)              //如果为添加
  {
  string tempName=“文件同步项{0}”;
  int count=0;              //初始化计数器
  do
  {
//通过循环得到新的名称增量
    this.textBoxName.Text=string.Format(tempName,++count);
  }while(NameIsDuplicate());  //判断名称是否重复
  }
  else
  {      //使用指定的同步项更新用户界面
  this.textBoxName.Text=item.Name;  //同步项名称
  this.textBoxSyncFrom.Text=item.SyncFromPath;  //源路径
  this.textBoxSyncTo.Text=item.SyncToPath;  //目标路径
  this.textBoxBackupFolder.Text=item.BackupPath;  //备份路径
  this.checkBoxBackupBeforeSync.Checked=item.BackupBeforeSync;  //是否备份
  this.checkBoxEnable.Checked=item.Enabled;  //是否允许
  this.checkBoxIncludeSubs.Checked=item.SyncSubfolders;  //是否包含子文件夹
  this.checkBoxRemoveAfterSync.Checked=item.DeleteAfterSync;//同步后是否删除
  }
}
```

之所以传入同步项名称集合，是因为程序要避免出现具有相同名称的同步项。如果是新增状态，程序通过循环的方式自动提供了一个同步项名称。如果不为新增状态，程序将从传入的SyncItem中更新用户界面，以便用户可以编辑现有的同步项[8]。

5 结论(Conclusion)

使用System.IO命名空间中的文件操作类能实现在两个或多个文件夹之前自动复制备份，实现同步工作，并使用OOP方法设计文件同步相关的类，将文件同步的核心功能实现在Windows服务中，为了在Windows服务与监控配置程序之间通信，使用WCF命名管道实现了进程间的通信。

服务类应用程序的开发在实现文件夹同步技术中发挥了很大的优势，同时能更深入地理解Windows服务的使用，其相关技术有待进一步研究。

参考文献(References)

[1] Qingfeng Jing,et al.Pseudo-noise preamble based joint frame and frequency synchronization algorithm in OFDM communication systems[J].Journal of Systems Engineering and Electronics,2014,(01):251-253.

[2] R.Rakkiyappan,N.Sakthivel,S.Lakshmanan.Exponential synchronization of complex dynamical networks with Markovian jumping parameters using sampled-data and mode-dependent probabilistic time-varying delays[J].Chinese Physics B,2014(02):891-892.

[3] Hui-Na Feng,Jun-Min Li.Distributed Adaptive Synchronization of Complex Dynamical Network with Unknown Time-varying Weights[J].International Journal of Automation and Computing,2015(03):475-476.

[4] 李鸣洋.Linux下实时文件同步传输系统的实现[J].电脑知识与技术,2014(31):89-91.

[5] 许圣明,等.基于有序哈希树的RPKI资料库数据同步方法[J].计算机系统应用,2016(06):132-134.

[6] 王宾,刘钊远.基于Rsync的远程文件同步优化模型[J].计算机与现代化,2015(04):292-294.

[7] 刘珺,叶勇,石竹.文件同步系统的研究和实现[J].信息安全与通信保密,2014(02):439-441.

[8] 周平,刘晓洁.基于两级分块的文件同步方法[J].计算机工程与设计,2014(03):59-62.

作者简介：

邹　珺(1981-)，女，硕士，讲师.研究领域：软件开发.