教育部与高通公司产学合作协同育人项目

燕山大学
机上作业报告

课程名称：计算机操作系统

主题：感受、体验和欣赏 OS 中的创新

题目： 操作系统 API 应用体验与编程设计

姓名	学号	班级	自评成绩
乔翱	201811040809	软件六班	A+

指导教师：申利民

2020 年 09 月 13 日

目录

一、实验任务

通过使用OS的API编写一个程序，满足下列要求：

（1）能够判断U盘是否存在；
（2）能够显示U盘的总容量、使用容量和剩余容量；
（3）能够将某个目录上的文件或整个目录复制到U盘上；
（4）可以删除U盘上文件；
（5）禁止U盘的使用及开启U盘的使用；
（6）推荐使用C# winform，也可以使用其它语言；
（7）体会OS的API的作用；
（8）尝试读取PCB信息;
(9) 其它创意。
（10）项目最终能以图形界面的形式完成。

Tips：C#对USB的管理有很多封装好的方法，本次作业重点是操作系统的API调用，因此这些方法禁止使用，只可使用WinAPI的相关函数。

二、实验目的和意义

1、了解并学会使用Windows API

操作系统的作用之一就是屏蔽一些复杂的直接对硬件操作，并提供给用户一个简单明确的应用接口，类外对于一些基本的或常用的操作也以API的形式提供给用户，比如内存管理、文件管理等。Windows API也叫作API函数，API（Application Programming Interface）是“应用程序编程接口”，开发人员在使用各种各样的编程语言进行Windows开发的时候，都可以使用API函数。在做程序设计、软件开发，肯定是要调用或者使用Windows里已经做好的API函数。微软在做Windows时，几万人做了十几年做出了大量的API函数，例如Windows用户界面相关应用程序接口、内存管理、数据的输入输出操作和中断处理等，Windows自身在用这些函数，程序员也可以调用这些函数，可以说，API函数是程序员的必学基础。

2、利用C#中的WinForm编程体验Windows API

Windows API是微软为了方便广大Windows开发人员调用系统底层功能而公开的一系列函数接口。.net中的函数很大都是对系统底层API的一些封装，但是在.net中并没有包含Windows所有的API函数。在.net中允许我们调用系统的API函数，并且可以根据需要向系统API传递输入或者输出参数。

C#在调用非托管API函数的时候，它将执行以下操作：查找包含该函数的DLL，将DLL加载到内存中。查找函数在内存中的地址并且将其参数推到堆栈上，以封送所需的数据，将控制权转移给非托管函数，对非托管 DLL 函数的“平台调用”会向托管调用方引发由非托管函数生成的异常。

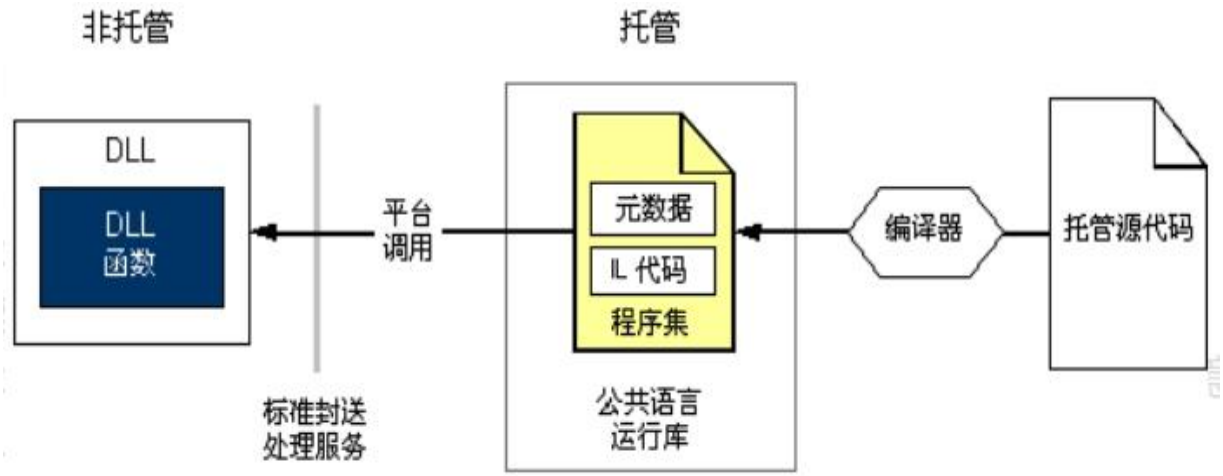

图 2.2 .net 调用非托管 API 示意图

3、C#调用 Windows API

C#调用 DLL 导出的函数的步骤：

（1）使用 C#关键字 static 和 extern 声明方法。

（2）将 DllImport 属性附加到该方法。DllImport 属性允许指定包含该方法的 DLL 的名称。

（3）如果需要，为方法的参数和返回值指定自定义封送处理信息，这将重写.NET Framework 的默认封送处理。

例如：

```
声明：
 [DllImport("User32.dll")]
 public static extern int MessageBox(int h, string m, string c, int type);

调用：
MessageBox(0, "hello win32 api", "yes", 4);
```

三、实现的具体功能

1、检测 U 盘是否插入，获取 U 盘的盘符

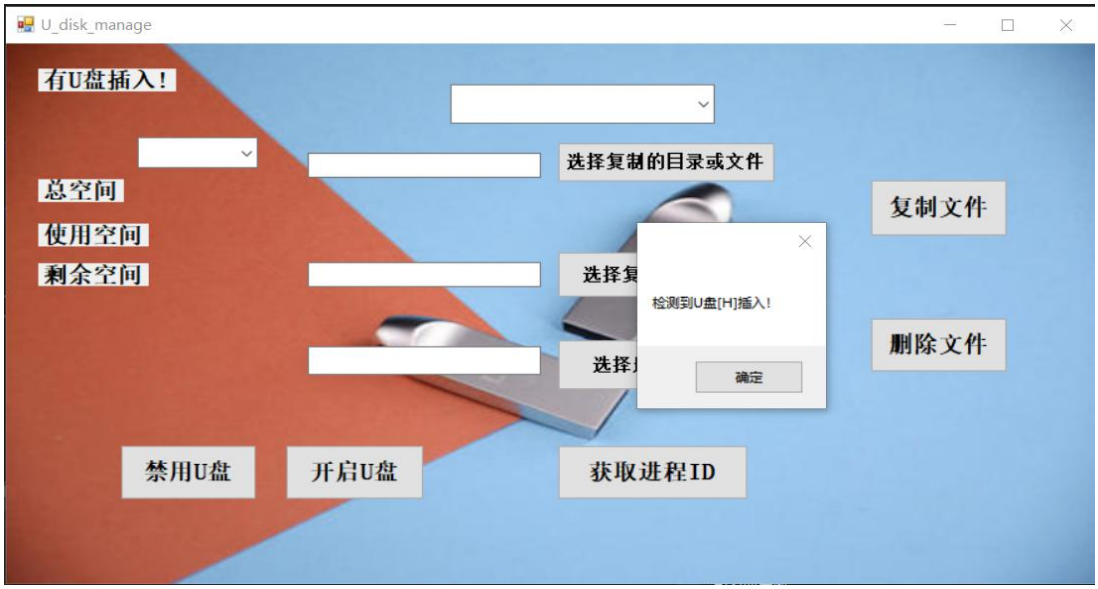

图 3.1 检测 U 盘是否插入，获取 U 盘盘符

2、时刻检测是否有 U 盘插入或者拔出，插入和拔出立刻提醒

图 3.2 实时检测 U 盘

3、获取 U 盘的总容量、使用容量、可用容量

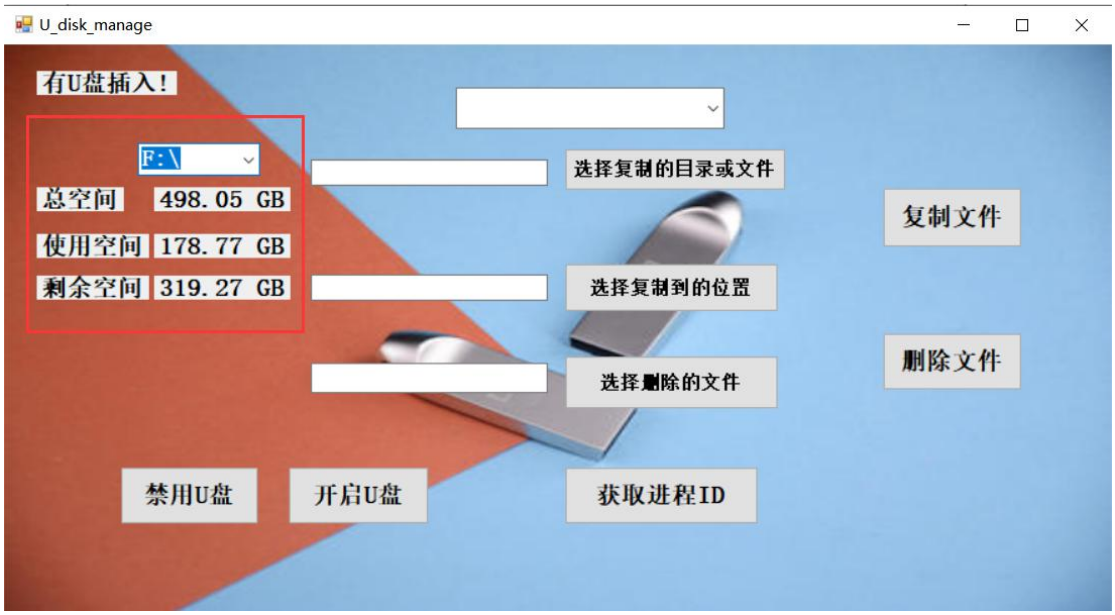

图 3.3　获取 U 盘容量信息

4、将电脑上的文件复制到 U 盘（可以选择复制整个文件夹或某个文件）

首先需要选择是复制整个目录中所有的文件还是复制某个文件，接着选择复制的目录或文件以及复制到的位置。

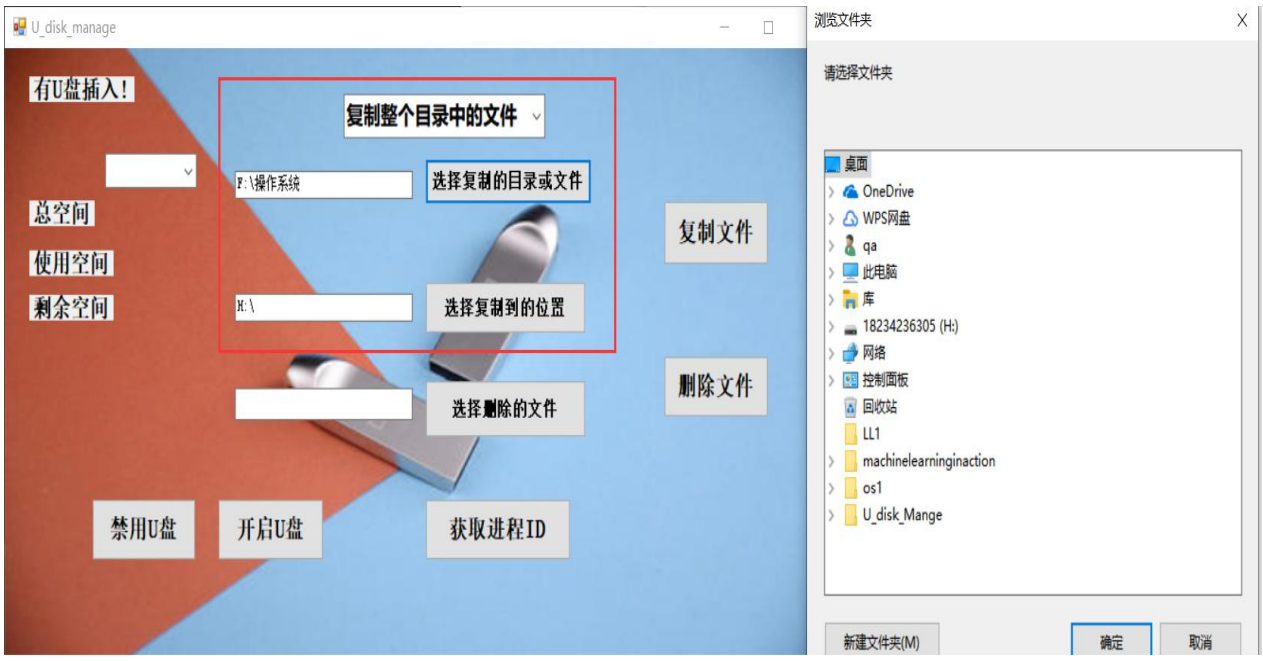

图 3.4.1　选择复制的目录

选好需要复制的文件或者文件夹以及复制的位置后，点击复制文件，则成功复制。

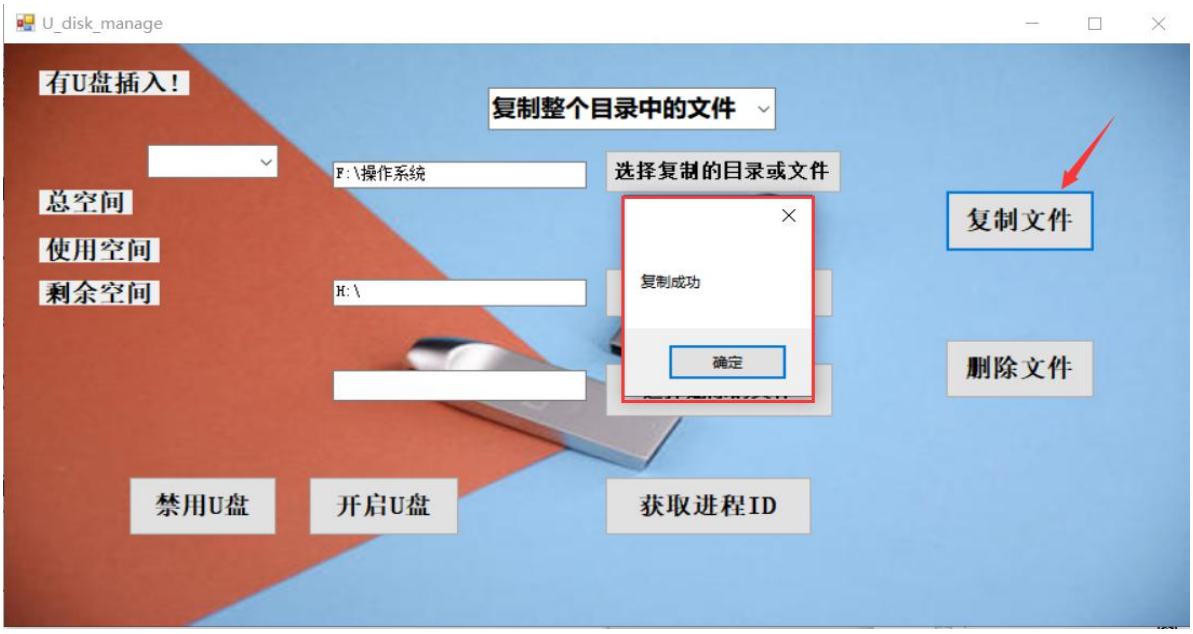

图 3.4.2　点击复制文件

5、删除文件或者文件夹

首先选择要删除的文件或者文件夹

图 3.5.1　选择要删除的文件或文件夹

选择好要删除的文件或者文件夹后，点击删除文件按钮，会弹出警示框，询问用户是否确定删除，确定删除就会成功删除。

图 3.5.2　点击删除文件

6、开启和禁止使用 U 盘

点击禁用 U 盘后，插入 U 盘，不再检测到有 U 盘插入，只有点击开启 U 盘后才能检测到 U 盘插入。

图 3.6.1　禁用 U 盘

图 3.6.2　开启 U 盘

7、获取 PCB 信息（获取进程名和进程号等信息）

图 3.7　点击获取进程 ID 获取进程信息

四、总体结构及框架

本项目采用 C#中的进行开发，利用 C#中的窗体应用 WinForm，利用其中的封装好的控件进行开发设计，调用 Windows API 进行各个模块功能的实现。

WinForm 非常方便本项目的开发，其中的控件极大的方便了开发，使用 label、button 等控件可以迅速构造一个界面，通过编写点击按钮触发的事件的代码以及其他事件的代码来实现本项目的全部功能。

窗体分为可视化界面和界面代码两部分，Form 是窗体可视化界面，.Designer.cs 结尾的文件，是窗体界面的源代码，界面是通过这个源代码来构建组成的；

整体项目采用模块化的设计方法，按照功能需求，一个模块一个模块来实现，最终整合成整个完整的项目。

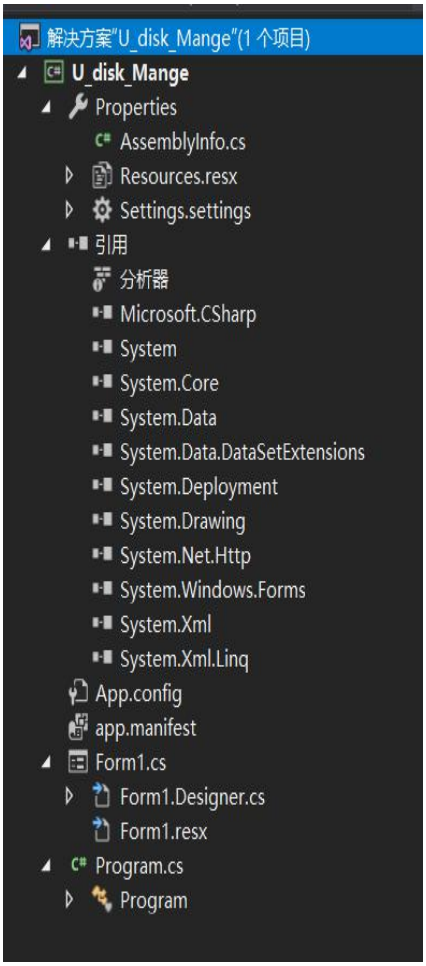

图 4.1　解决方案视图

图 4.2 界面设计视图

五、实现技术及处理流程

实现技术：C# WinForm，Windows API

处理流程：

1、实时检测 U 盘并获取盘符

对于 U 盘的检测首先是静态检测，通过获取 PC 中的盘符以及判断盘符的类型进行检测 PC 中是否插入 U 盘。对 U 盘的动态检测则需要在 PC 机的状态改变的时候重新获取盘符检验是否是 U 盘。

编写 checkDisk（）函数检验当前 PC 中存在的 U 盘盘符，以字符串形式返回。

这一功能主要使用的 Windows API 是 GetLogicalDrives()，返回值是 long 类型将其用二进制显示时，其中第 0 位表示 A 盘，第一位表示 B 盘，当某位为 1 时说明存在这个盘，为 0 表示不存在。

还使用了 Windows API 中获取磁盘类型的 GetDriveType（string lpRootPathName），该函数的参数是盘符的名字，例如：C\\，返回的是一个 int 类型的数字，根据返回值可以判断磁盘的类型。

GetDriveType（string lpRootPathName）返回值对应的类型：

0 "DRIVE_UNKNOWN"　　无法识别的设备

1 "DRIVE_NO_ROOT_DIR"　给出的名字不存在

2 "DRIVE_REMOVABLE"　　可移动设备

3 "DRIVE_FIXED"　　不可移动的磁盘

4 "DRIVE_REMOTE"　　网络硬盘

5 "DRIVE_CDROM"　　CD 光驱

6 "DRIVE_RAMDISK"　　内存虚拟盘

只有返回值是 2 才代表是 U 盘，则把该盘符假如到返回的字符串。

对于 U 盘状态的实时检测只需要重写 WndProc(ref Message m)函数并且在检测到设备改变的时候（DBT_DEVICEARRIVAL//U 盘插入，和 DBT_DEVICEREMOVECOMPLETE//U 盘卸载）更新 U 盘的状态（即调用 checkDisk（）函数重新检测 U 盘并且获取盘符）

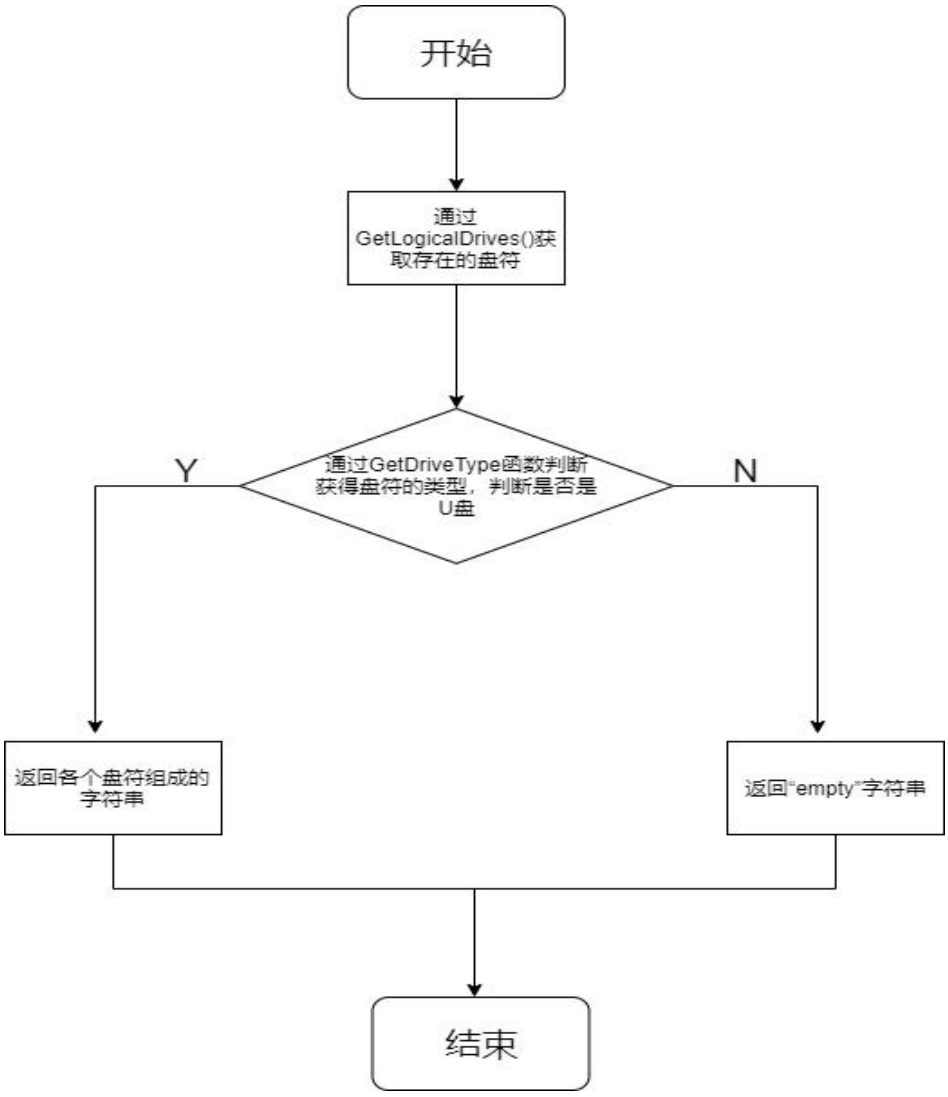

图 5.1 检测 U 盘获取盘符程序（checkDisk（）函数）流程图

2、获取 U 盘容量信息

通过调用 Windows API 中的 GetDiskFreeSpaceEx 直接获得 U 盘总容量，U 盘剩余容量等信息。

本项目可以显示多个 U 盘的容量信息通过一个选择框来选择显示哪个 U 盘的容量信息。

GetDiskFreeSpaceEx 示例：

```
[DllImport("Kernel32.dll")]
      public static extern bool GetDiskFreeSpaceEx(
    string lpDirectoryName,
    out UInt64 lpFreeBytesAvailable,
    out UInt64 lpTotalNumberOfBytes,
    out UInt64 lpTotalNumberOfFreeBytess);//得到盘符的容量以及剩余容量
```

3、复制文件或者文件夹到 U 盘

对于文件的复制可以调用 Windows API 中的 copyFile 进行文件的复制，但是该 API 无法进行对于整个文件夹的复制。

对于文件夹的复制，编写了一个 InteropSHFileOperation 类，在该类也是 Windows API 中存在的，实例化一个 InteropSHFileOperation 对象，调用其 execute 方法进行文件夹的复制。这种方法也可以实现对于文件的复制，本项目最终对于文件和文件夹的复制都采用此方法。

复制文件后由于 U 盘的容量信息发生了改变，所以还需要进行容量信息更新的操作。

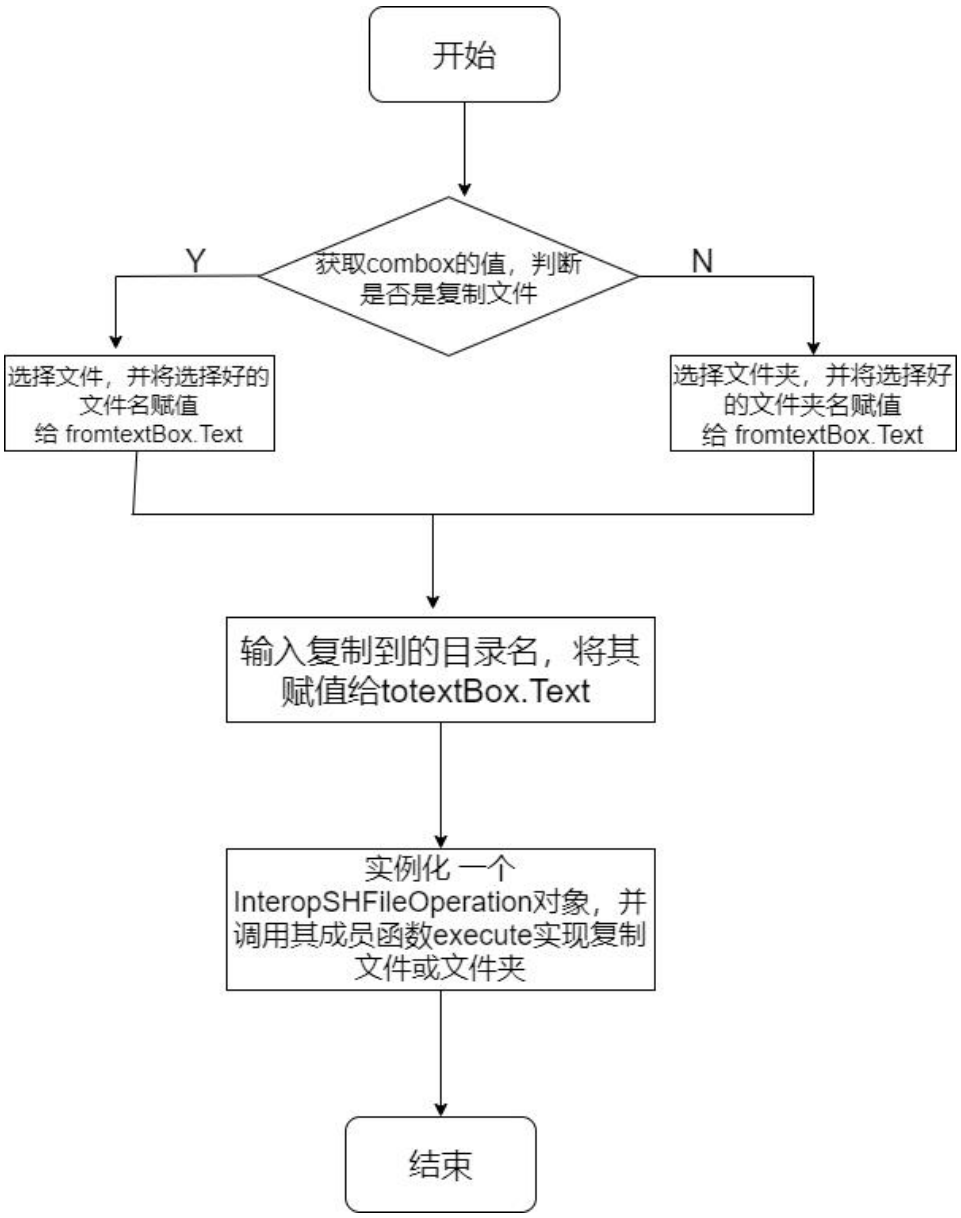

图 5.3 复制文件或文件夹程序流程图

4、删除 U 盘内的文件或者文件夹

对于删除文件有两种办法，一种是利用 Windows API 中的 DeleteFile 进行删除文件，但是该函数只能删除单个文件，如果要删除整个文件，还需要对文件夹进行遍历，假如文件夹里面还有文件夹，就需要进行一个深度搜索一个一个删除，这种方法比较繁琐，编码也比较复杂。

另一种是实例化一个 InteropSHFileOperation 对象，直接其中的成员函数可以很方便的对文件或者文件夹进行删除。

在删除操作后同样需要更新 U 盘容量信息。

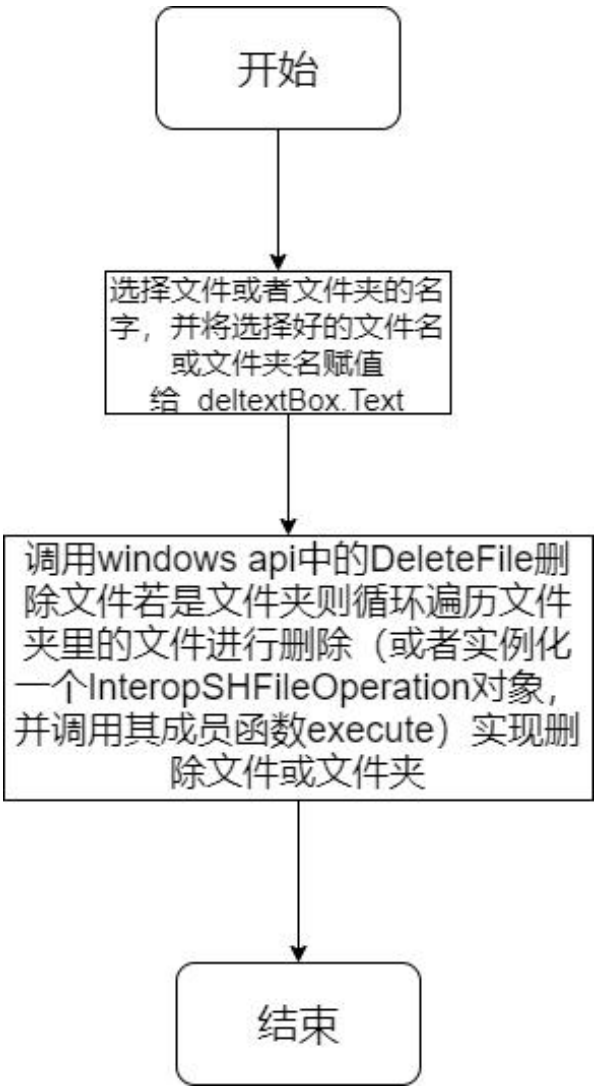

图 5.4 删除文件或文件夹程序流程图

5、禁用 U 盘和开启 U 盘

本项目对于此功能采用的是更改注册表的方法，C#中有封装好的类 Registry，利用该类可以很方便的对注册表进行操作。但是利用 Windows API 也可以对注册表进行修改实现禁用

U 盘和开启 U 盘，对于此功能的实现主要采取了下面的三个 Windows API 函数对注册表进行操作：

```
public static extern uint RegSetValueEx( //修改注册表
      UIntPtr hKey,
      [MarshalAs(UnmanagedType.LPStr)]
       string lpValueName,
      int Reserved,
      RegistryValueKind dwType,
      IntPtr lpData,
      int cbData);
     public static extern int RegOpenKeyEx(  //打开注册表
      UIntPtr hKey,
      string subKey,
      int ulOptions,
      int samDesired,
      out UIntPtr hkResult);
      public static extern int RegCloseKey(UIntPtr hKey)//关闭注册表
      static extern uint RegQueryValueEx( //查询注册表
       UIntPtr hKey,
      string lpValueName,
      int lpReserved,
      out uint lpType,
      StringBuilder lpData,  ref uint lpcbData);
```

通过打开注册表->查询注册表->修改注册表->关闭注册表实现该功能。

注意：对注册表的操作需要具有管理员权限。

6、获取 PCB 信息

本项目通过调用 Windows API 中的 getCurrentProcessId()来获取当前进程的 PID，通过该 PID 可以杀死进程以及对进程进行其他操作。

六、使用说明

1、项目路径

解压压缩包后，进入 bin/Debug 目录下，点击 U_disk_Mange.exe 即可运行。

如果需要使用开启和禁用 U 盘的功能则需要授予权限，即右击 U_disk_Mange.exe 选择以管理员身份运行。

2、项目开发环境

VS Studio 2017 版本

.NET Framework 3.5

3、进入项目界面

进入项目界面后，可以体验该项目，点击相应的功能的按钮体验该项目的功能，对 U 盘进行管理操作。

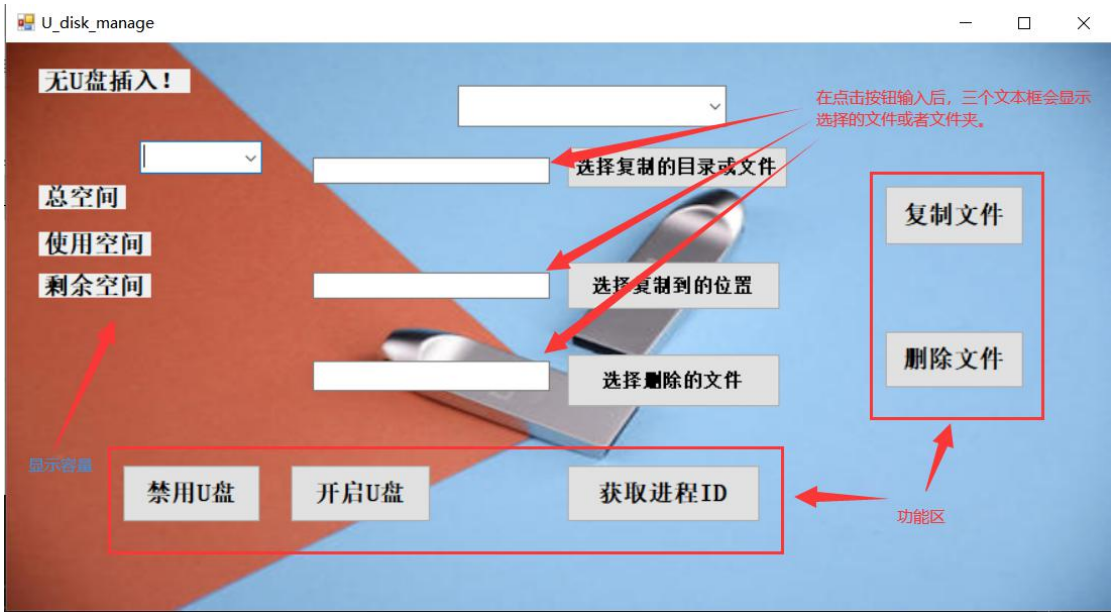

图 6.1 项目主界面

七、项目的特色与创新点

1、界面简洁、容易操作

本项目采用 C#窗体应用进行开发，C#对一些控件有着很好封装，使用起来非常方便，在设计的时候注意到用户友好性，例如在删除文件的时候弹出提示框，确定用户是否要删除，增强了友好性。

整个项目的界面很简单，用户操作起来很方便，用户可以轻松使用此项目进行对 U 盘的管理。

2、时刻检测 U 盘的状态

本项目可以实时检测 U 盘状态，只要有 U 盘的插入和拔出就会弹出消息框提醒用户。对于 U 盘的检测是随时进行的，这样可以使用户及时的了解 PC 中 U 盘状态的改变，增强了用户体验。

3、可以复制或删除文件和文件夹

对于文件的复制和删除操作，项目开发初期采用的是 Windows API 中的 copyFile 和 DeleteFile，但是这两个 API 只能实现对于文件的复制和删除，不能实现对于文件夹的复制

和删除。于是通过深度搜索进行对目录中的文件一个一个复制或者删除，但是此方法编码难度大，运行时间久。最终查阅了很久的资料，采用类 InteropSHFileOperation 来进行对文件或者文件夹的复制或者删除。

对于对文件的移动、复制、删除、重命名采用封装成类 InteropSHFileOperation 进行操作，在该类中可以采用类的成员函数进行对文件或者文件夹的操作，包括移动、复制、删除、重命名。开始的时候只是采用 copyfile 对文件进行复制，但是该 Windows API 不能实现对整个文件夹的复制，于是通过深搜整个目录进行一个一个文件的复制，但是这种方法编码难度大，运行时间久。

对于编写的类 InteropSHFileOperation 其中还有很多的方法可以实现对文件或者文件夹的操作，包括对文件或者文件夹的移动、删除、重命名、复制等，利用该类可以方便之后对于本项目其他功能的开发。

4、显示多个 U 盘容量信息

本项目可以同时检测到多个 U 盘的插入，并且通过对选择框内盘符的选择，可以选择显示各个 U 盘的容量信息，实现对于多个 U 盘同时插入也能选择盘符进行显示 U 盘的容量信息。

八、总结本次任务完成情况、经验教训和体会

本次项目实现了全部所需的功能。在开发过程中遇到了很多的困难，首先是对于利用 C# 来开发来说，网上的资料较少，大部分都是关于 C++和 MFC 的资源，所以找参考资料废了很大的力气。

在开发过程中对于文件的复制和注册表修改这两个模块遇到了很大的困难，因为对于 Windows API 中这两个操作的相关 api 都比较复杂，对于 C++，其中都封装好了很多结构体，而 C#没有封装好的，只能自己定义来使用，使用起来比较复杂，网上对于相关的资料比较少，在开发过程中也是遇到了很大的困难。不过最终还是通过查阅资料一点一点地解决了相关的一系列的问题。

通过本次实验体会到了 Windows API 编程的魅力，各种高级语言都对 Windows API 进行了很好的封装，越来越多的人都是直接去调用封装好的东西，而不去了解底层的东西。高级语言对 Windows API 的封装也是极大的方便了编程人员的开发。这些优秀可视化编程环境操作简单、界面友好，在这些工具中提供了大量的类库和各种控件，它们替代了 API 的神秘功能，事实上这些类库和控件都是构架在 WIN32 API 函数基础之上的，是封装了的 API 函数的集合。它们把常用的 API 函数的组合在一起成为一个控件或类库，并赋予其方便的使用方法，所以极大的加速了 Windows 应用程序开发的过程。有了这些控件和类库，程序员便可以把主要精力放在程序整体功能的设计上，而不必过于关注技术细节。

但是对于 Windows 应用开发来说，编程人员了解 Windows API 还是非常有必要的，只有了解了这些 API 的使用方法才能更有助于 Windows 应用的开发。

在 Windows 程序设计领域处于发展的初期，Windows 程序员所能使用的编程工具唯有 API 函数，这些函数是 Windows 提供给应用程序与操作系统的接口，他们犹如“积木块”一样，可以搭建出各种界面丰富，功能灵活的应用程序。API 函数就像是构筑整个 Windows 框架的

基石，在它的下面是 Windows 的操作系统核心，而它的上面则是所有的华丽的 Windows 应用程序。

如果要开发出更灵活、更实用、更具效率的应用程序，必然要涉及到直接使用 API 函数，虽然类库和控件使应用程序的开发简单的多，但它们只提供 Windows 的一般功能，对于比较复杂和特殊的功能来说，使用类库和控件是非常难以实现的，这时就需要采用 API 函数来实现。

通过开发本项目收获颇丰，了解到了很多 Windows API 的使用，最终开发出了这个对 U 盘管理的小项目，但是此项目还有待改进，例如：对界面的优化，对功能的丰富等，这是在接下来的时间里需要进一步做的地方。通过本项目还极大的锻炼了自己查阅资料的能力，提高了自己的编程能力。总体来说，通过本次实验，学到了很多知识，收获很多。