使用windows操作系统提供的DLL，实现对注册表的操作

1） 创建C#控制台项目

2） 引入操作注册表类库和文件读写的命名空间

```
using Microsoft.Win32;
using System.IO;
```

3） 使用Registry类，RegistryKey类读取注册表内容

```
//从注册表HKEY_CURRENT_USER\MyDefine读取SourceDirectory的值
RegistryKey myDefine = Registry.CurrentUser.OpenSubKey("MyDefine");
SourceDirectory = myDefine.GetValue("SourceDirectory").ToString();
```

4）先Directory.Exists判断文件夹是否存在，再获取文件名，利用FileStream和StreamReader打开并读取文件同时合并文本内容

```
if (Directory.Exists(SourceDirectory)){
string [] fnames = Directory.GetFiles(SourceDirectory);
Array.ForEach(fnames, (fname) =>{
using (FileStream fsRead = new FileStream(fname, FileMode.Open, FileAccess.Read,
FileShare.None)){
using (StreamReader reader = new StreamReader(fsRead, Encoding.Default)){
richTextBox1.Text += reader.ReadToEnd()+"\n";
reader.Close();}
fsRead.Close();}});}
```

5）更新注册表

```
RegistryKey myDefine = Registry.CurrentUser.CreateSubKey("MyDefine");
//创建Save子键存TargetDirectory、TargetFile
using (RegistryKey
save = myDefine.CreateSubKey("Save"))
{
save.SetValue("TargetDirectory", TargetDirectory);
save.SetValue("TargetFile", TargetFile);
}
```

6）写入文件

```
using (FileStream fsWrite = new FileStream(fname, FileMode.Create,
FileAccess.Write, FileShare.None){
using (StreamWriter writer = new StreamWriter(fsWrite, Encoding.Default)){
writer.WriteLine(richTextBox1.Text);
writer.Close();}
fsWrite.Close();}
```

通过重定向机制实现进程间通信

1.调用getmac获取网卡mac--"getmac"
2.调用shutdown命令关闭--"shutdown -s -t 0"或重启电脑--"shutdown - r"

1）异步接收消息

创建Process进程类对象并设置要启动的应用程序为cmd

```
Process process = new Process();
process.StartInfo.FileName = "cmd.exe";
```

重定向输入流、输出流

```
process.StartInfo.RedirectStandardInput = true;
process.StartInfo.RedirectStandardOutput = true;
```

OutputDataReceived绑定消息接收函数strOutputHandler

```
process.OutputDataReceived += new DataReceivedEventHandler(strOutputHandler);
```

调用Start方法启动进程，向输入流写入相应的cmd指令

```
string strCmd = "getmac";
process.Start();
process.StandardInput.WriteLine(strCmd);
```

最后调用BeginOutputReadLine()开始异步接收消息。

```
process.BeginOutputReadLine();
```

其中，消息接收函数通过FindWindow API的方式找到目标进程句柄再向窗体发送消息，同时重载Form的DefWndProc方法来把接收到的消息显示在窗体控件上。

2)同步接收消息

创建Process进程类对象并设置要启动的应用程序为cmd，重定向输入流、输出流，调用Start方法启动进程，向输入流写入相应的cmd指令。(以上部分代码同异步接收消息，只是不绑定消息处理函数)

获取输出信息，最后调用WaitForExit方法等待关联进程退出并Close该进程。

```
tbShow.Text = process.StandardOutput.ReadToEnd();
process.WaitForExit();
process.Close();
```

通过管道机制，实现进程间通信客户端向服务器端发送数据，服务器显示数据

1.创建客户端项目，引入命名空间

```
using System.IO.Pipes;
```

2.创建与服务端同名的NamedPipeClientStream对象作为客户端

```
NamedPipeClientStream pipeClient = new NamedPipeClientStream(".", "testpipe",
PipeDirection.Out, PipeOptions.Asynchronous, TokenImpersonationLevel.None);
```

3.连接服务端

```
pipeClient.Connect();
```

4.StreamWriter发送数据

```
StreamWriter sw = new StreamWriter(pipeClient);
sw.AutoFlush = true;
sw.WriteLine(textBox2.Text);
sw.Close();
```

5.创建服务端项目，引入管道命名空间

6.创建服务端对象

```
NamedPipeServerStream pipeServer =  new NamedPipeServerStream("testpipe",
PipeDirection.InOut, 1, PipeTransmissionMode.Byte, PipeOptions.Asynchronous);
```

7.创建工作线程异步等待客户端连接，并将其设置为后台线程，最后启动。

```
Thread thread = new Thread(new ThreadStart(() => {
pipeServer.BeginWaitForConnection(Recieve, pipeServer); }));
thread.IsBackground = true;
thread.Start();
```

8.实现自定义的消息处理函数void Recieve(IAsyncResult result);

```
NamedPipeServerStream pServer = (NamedPipeServerStream)result.AsyncState;
//结束等待
pServer.EndWaitForConnection(result);
//读取数据
StreamReader sr = new StreamReader(pServer);
//使用委托将数据显示到窗体控件上
this.Invoke((MethodInvoker)delegate { textBox1.Text = sr.ReadToEnd(); });
sr.Close();
```

采用信号量机制实现消费者与生产者的线程同步

1.导入命名空间using System.Threading;

2.使用Mutex实现生产者、消费者互斥访问仓库、

```
public Mutex mMutex = new Mutex();
```

3.Empty和Full信号量总和为仓库的最大容量（10）

```
//仓库产品数量
public Semaphore Full = new Semaphore(0, 10);
//仓库空位数量
public Semaphore Empty = new Semaphore(10, 10);
```

4.定义指示生产消费启动结束的信号量

```
bool StopFlag = false;
```

5.为每个生产者、消费者都创建一个线程，并启动

```
for (int i = 0; i < proNum; i++){
Thread ProThread = new Thread(new ParameterizedThreadStart(Produce));
ProThread.Start(i);}
for (int i = 0; i < conNum; i++){
Thread ConThread = new Thread(new ParameterizedThreadStart(Consume));
ConThread.Start(i);}
```

6.生产者每生产一个产品需先申请mMutex再申请一个Empty资源,生产结束释放mMutex并释放一个Full资源

```
while (!StopFlag){
Empty.WaitOne();
mMutex.WaitOne();
生产
mMutex.ReleaseMutex();
Full.Release(1);}
```

7.消费者每消费一个产品需先申请Empty资源再申请mMutex，消费结束后释放mMutex并释放一个Empty资源。

```
while (!StopFlag){
Full.WaitOne();
mMutex.WaitOne();
消费
Empty.Release(1);
mMutex.ReleaseMutex();}
```

8.停止生产消费活动

```
StopFlag = true;
```

使用excel的com组件

- 编写一个窗体应用程序，将给定Excel表格中信息显示到窗体界面。

1.创建控制台程序后向项目添加引用Microsoft.Office.Interop.Excel

2.在程序源文件添加Excel命名空间

3.创建OleDbConnection 对象

```
String path = @"..\..\hotel.xls"
String excelConnString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + path
+ ";Extended Properties='Excel 12.0 Xml;HDR=YES';";
OleDbConnection excleConn = new OleDbConnection(excelConnString);
```

4..打开连接，进行查询，并将查询保存到一个DataTable对象

```
try{
     excleConn.Open();
     //得到所有sheet的名字
     DataTable sheetsName =
excleConn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, new object[] { null, null,
null, "Table" });
     string firstSheetName = sheetsName.Rows[0][2].ToString(); //得到第一个sheet的
名字
     string sql = string.Format("SELECT * FROM [{0}]", firstSheetName); //查询字
符串
     OleDbDataAdapter ada = new OleDbDataAdapter(sql, excelConnString);
     DataSet set = new DataSet();
     ada.Fill(set);
     excleConn.Close();
     return set.Tables[0];
  }
 catch (Exception){
     excleConn.Close();
     return null;
}
```

5.在Form设计器上添加dataGridView控件，并将数据源设置为4中的DataTable对象

```
DataTable dt = ReadToTableExcelSheet_1();
dataGridView1.DataSource = dt;
```

- 编写功能设置Excel表格的边框为黑实线。P82.

  ```
  range.Interior.ColorIndex = 15;//15代表灰色
  ```

- 向Excel文档中添加图表。P83

WinForm实现两个窗体应用程序的消息传递

1.创建.NET Framework的WinForm项目

2.在Form1设计器上添加按钮使得Form2显示，编写Click函数

```
public partial class Form1 : Form{
        public Form2 nextForm { get; set; }
        .....
        private void btnToForm2_Click(object sender, EventArgs e){
            if (nextForm == null){
            //创建与Form1通信的Form2对象
                nextForm = new Form2();
                nextForm.deskTop=this; }
            //显示Form2
            nextForm.Show();
            //将焦点转到Form2
            nextForm.Focus();}}
```

3.在当前项目添加项，选择窗体类型。在Form2设计器上添加按钮使得重新聚焦到Form1，编写Click函数

```
public partial class Form2 : Form{
        //指向Form1窗体
        public Form1 deskTop { get; set; }
        ......
        private void btnBackForm1_Click(object sender, EventArgs e){
            deskTop.Show();
            deskTop.Focus();}}
```

WinForm使用消息机制通信

1.定义结构体

```
//lpData是我们传输的字符串数据
public struct COPYDATASTRUCT
 {
 public IntPtr dwData;
 public int cbData;
 [MarshalAs(UnmanagedType.LPStr)]
 public string lpData;
 }
```

2.消息发送者

```
1引用系统动态连接库中的SendMessage函数和FindWindow函数
[DllImport("User32.dll", EntryPoint = "SendMessage")]
private static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, ref
COPYDATASTRUCT lParam);
[DllImport("User32.dll", EntryPoint = "FindWindow")]
private static extern int FindWindow(string lpClassName, string lpWindowName);
2寻找消息接收者
IntPtr WINDOW_HANDLER = FindWindow(null, @"消息接受者");
3调用SendMessage函数发送信息
 public const int WM_COPYDATA = 0x004A;
 private void button3_Click(object sender, EventArgs e)
 {
 byte[] sarr = Encoding.Default.GetBytes(messageSendTextBox.Text);
 int len = sarr.Length;
```

```
COPYDATASTRUCT cds;
cds.dwData = (IntPtr)Convert.ToInt16(0);//可以是任意值
cds.cbData = len + 1;//指定lpData内存区域的字节数
cds.lpData = messageSendTextBox.Text;//发送给目标窗口所在进程的数据
SendMessage(hWnd, WM_COPYDATA, 0, ref cds); //main_whandle是消息接受窗体的句柄
}
```

3.消息接受者

```
1.重载消息处理函数DefWndProc
protected override void DefWndProc(ref System.Windows.Forms.Message m)
        {
            switch (m.Msg)
            {
                case WM_COPYDATA:
                    COPYDATASTRUCT mystr = new COPYDATASTRUCT();
                    Type mytype = mystr.GetType();
                    mystr = (COPYDATASTRUCT)m.GetLParam(mytype);
                    this.tbShow.Text += mystr.lpData + "\r\n";
                    break;
                default:
                    base.DefWndProc(ref m);
                    break;
            }
        }
```

WPF消息通信机制

1 2用Winform

3.消息接受者

```
1.创建一个消息处理方法对收到的消息进行处理，返回值是IntPtr类型
IntPtr WndProc(IntPtr hwnd, int msg, IntPtr wParam, IntPtrlParam, ref bool
handled){
     if (msg == WM_COPYDATA){
         messageReceiveTextBox.Clear();
         COPYDATASTRUCT cds = (COPYDATASTRUCT)
System.Runtime.InteropServices.Marshal.PtrToStructure(lParam,typeof(COPYDATASTRU
CT));
         messageReceiveTextBox.AppendText(cds.lpData); }
         return hwnd;}
2.在主窗体加载时，为我们创建的消息处理方法注册钩子事件，使其能够接受消息
private void mainWindowLoad(object sender, RoutedEventArgse){
 (PresentationSource.FromVisual(this) as
System.Windows.Interop.HwndSource).AddHook(new
System.Windows.Interop.HwndSourceHook(WndProc));
 define_Event();}
```

Invoke函数

在Windows下，窗体（Form）是由单独的线程控制的，这意味着，我们如果在控件的触发逻辑代码中新建了线程，这个线程是没有办法修改我们现有的窗体控件属性的。这时为了达到修改窗体控件属性的效果，我们需要调用控件的Invoke函数。Invoke方法首先检查发出调用的线程(即当前线程)是不是UI线程，如果是，直接执行委托指向的方法，如果不是，它将切换到UI线程，然后执行委托指向的方法。
1.创建一个委托

```
public delegate void AppendTextBoxDelegate(string msg);
```

2.创建一个方法作为委托的具体实现

```
public void AppendTextBox(string msg){textBox3.AppendText(msg);}
```

3.在需要的时候调用Invoke方法，传入的2个参数分别为委托对象和委托的参数

```
textBox3.Invoke(new AppendTextBoxDelegate(AppendTextBox), "期末复习要注意休息呀
\r\n");
private void button16_Click(object sender, EventArgs e) {
}
```

Demoleft

```
public static void DSToExcel(String sheetName, DataSet oldds)
        {
            //先得到汇总EXCEL的DataSet 主要目的是获得EXCEL在DataSet中的结构
            OleDbConnection myConn = new OleDbConnection(strConn);
            string strCom = "select * from [" + sheetName + "$]";
            myConn.Open();
            OleDbDataAdapter myCommand = new OleDbDataAdapter(strCom, myConn);
            System.Data.OleDb.OleDbCommandBuilder builder = new
OleDbCommandBuilder(myCommand);
            //QuotePrefix和QuoteSuffix主要是对builder生成InsertComment命令时使用。
            builder.QuotePrefix = "[";     //获取insert语句中保留字符（起始位置）
            builder.QuoteSuffix = "]"; //获取insert语句中保留字符（结束位置）
            DataSet newds = new DataSet();
            myCommand.Fill(newds, "table1");
            for (int i = 0; i < oldds.Tables[0].Rows.Count; i++)
            {
                //在这里不能使用ImportRow方法将一行导入到news中，因为ImportRow将保留原来
DataRow的所有设置(DataRowState状态不变)。
                //在使用ImportRow后newds内有值，但不能更新到Excel中因为所有导入行的
DataRowState != Added
                DataRow nrow = newds.Tables["Table1"].NewRow();
                for (int j = 0; j < newds.Tables[0].Columns.Count; j++)
                {
                    nrow[j] = oldds.Tables[0].Rows[i][j];
                }
                newds.Tables["table1"].Rows.Add(nrow);
            }
            myCommand.Update(newds, "table1");
            myConn.Close();
        }
        public Excel.Application m_xlApp = null;
        /// <summary>
        /// 将DataTable数据导出到Excel表
        /// </summary>
        /// <param name="tmpDataTable">要导出的DataTable</param>
        /// <param name="strFileName">Excel的保存路径及名称</param>
```

```
        public void DataTabletoExcel(System.Data.DataTable tmpDataTable, string
strFileName)
        {
            if (tmpDataTable == null)
            {
                return;
            }
            long rowNum = tmpDataTable.Rows.Count;//行数
            int columnNum = tmpDataTable.Columns.Count;//列数
            Excel.Application m_xlApp = new Excel.Application();
            m_xlApp.DisplayAlerts = false;//不显示更改提示
            m_xlApp.Visible = false;
            Excel.Workbooks workbooks = m_xlApp.Workbooks;
            Excel.Workbook workbook =
workbooks.Add(Excel.XlWBATemplate.xlWBATWorksheet);
            Excel.Worksheet worksheet =
(Excel.Worksheet)workbook.Worksheets[1];//取得sheet1
            try
            {
                if (rowNum > 65536)//单张Excel表格最大行数
                {
                    long pageRows = 65535;//定义每页显示的行数,行数必须小于65536
                    int scount = (int)(rowNum / pageRows);//导出数据生成的表单数
                    if (scount * pageRows < rowNum)//当总行数不被pageRows整除时，经
过四舍五入可能页数不准
                    {
                        scount = scount + 1;
                    }
                    for (int sc = 1; sc <= scount; sc++)
                    {
                        if (sc > 1)
                        {
                            object missing = System.Reflection.Missing.Value;
                            worksheet =
(Excel.Worksheet)workbook.Worksheets.Add(
                                        missing, missing, missing, missing);//添
加一个sheet
                        }
                        else
                        {
                            worksheet =
(Excel.Worksheet)workbook.Worksheets[sc];//取得sheet1
                        }
                        string[,] datas = new string[pageRows + 1, columnNum];

                        for (int i = 0; i < columnNum; i++) //写入字段
                        {
                            datas[0, i] = tmpDataTable.Columns[i].Caption;//表头
信息
                        }
                        Excel.Range range = worksheet.Range[worksheet.Cells[1,
1], worksheet.Cells[1, columnNum]];
                        range.Interior.ColorIndex = 15;//15代表灰色
                        range.Font.Bold = true;
                        range.Font.Size = 9;

                        int init = int.Parse(((sc - 1) * pageRows).ToString());
                        int r = 0;
```

```
                        int index = 0;
                        int result;
                        if (pageRows * sc >= rowNum)
                        {
                            result = (int)rowNum;
                        }
                        else
                        {
                            result = int.Parse((pageRows * sc).ToString());
                        }

                        for (r = init; r < result; r++)
                        {
                            index = index + 1;
                            for (int i = 0; i < columnNum; i++)
                            {
                                object obj = tmpDataTable.Rows[r]
[tmpDataTable.Columns[i].ToString()];
                                datas[index, i] = obj == null ? "" : "'" +
obj.ToString().Trim();//在obj.ToString()前加单引号是为了防止自动转化格式
                            }
                            System.Windows.Forms.Application.DoEvents();
                            //添加进度条
                        }
                        Excel.Range fchR = worksheet.Range[worksheet.Cells[1,
1], worksheet.Cells[index + 1, columnNum]];
                        fchR.Value2 = datas;
                        worksheet.Columns.EntireColumn.AutoFit();//列宽自适应。
                        m_xlApp.WindowState =
Excel.XlWindowState.xlMaximized;//Sheet表最大化
                        range = worksheet.Range[worksheet.Cells[1, 1],
worksheet.Cells[index + 1, columnNum]];
                        //range.Interior.ColorIndex = 15;//15代表灰色
                        range.Font.Size = 9;
                        range.RowHeight = 14.25;
                        range.Borders.LineStyle = 1;
                        range.HorizontalAlignment = 1;
                    }
                }
                else
                {
                    string[,] datas = new string[rowNum + 1, columnNum];
                    for (int i = 0; i < columnNum; i++) //写入字段
                    {
                        datas[0, i] = tmpDataTable.Columns[i].Caption;
                    }
                    //Excel.Range range = worksheet.get_Range(worksheet.Cells[1,
1], worksheet.Cells[1, columnNum]);
                    Excel.Range range = worksheet.Range[worksheet.Cells[1, 1],
worksheet.Cells[1, columnNum]];
                    range.Interior.ColorIndex = 15;//15代表灰色
                    range.Font.Bold = true;
                    range.Font.Size = 9;
                    int r = 0;
                    for (r = 0; r < rowNum; r++)
                    {
                        for (int i = 0; i < columnNum; i++)
                        {
```

```
                            object obj = tmpDataTable.Rows[r]
[tmpDataTable.Columns[i].ToString()];
                            datas[r + 1, i] = obj == null ? "" : "'" +
obj.ToString().Trim();//在obj.ToString()前加单引号是为了防止自动转化格式
                        }
                        System.Windows.Forms.Application.DoEvents();
                        //添加进度条
                    }
                    Excel.Range fchR = worksheet.Range[worksheet.Cells[1, 1],
worksheet.Cells[rowNum + 1, columnNum]];
                    fchR.Value2 = datas;

                    worksheet.Columns.EntireColumn.AutoFit();//列宽自适应。
                    m_xlApp.WindowState = Excel.XlWindowState.xlMaximized;

                    range = worksheet.Range[worksheet.Cells[1, 1],
worksheet.Cells[rowNum + 1, columnNum]];
                    //range.Interior.ColorIndex = 15;//15代表灰色
                    range.Font.Size = 9;
                    range.RowHeight = 14.25;
                    range.Borders.LineStyle = 1;
                    range.HorizontalAlignment = 1;
                }
                workbook.Saved = true;
                workbook.SaveCopyAs(strFileName);
            }
            catch (Exception ex)
            {
                //MessageBox.Show("导出异常：" + ex.Message, "导出异常",
MessageBoxButton.OK);
               // Log.Write(ex.Message);
            }
            finally
            {
                EndReport();
            }
        }
        /// <summary>
        /// 退出报表时关闭Excel和清理垃圾Excel进程
        /// </summary>
        private void EndReport()
        {
            object missing = System.Reflection.Missing.Value;
            try
            {
                m_xlApp.Workbooks.Close();
                m_xlApp.Workbooks.Application.Quit();
                m_xlApp.Application.Quit();
                m_xlApp.Quit();
            }
            catch { }
            finally
            {
                try
                {

System.Runtime.InteropServices.Marshal.ReleaseComObject(m_xlApp.Workbooks);
```

```
System.Runtime.InteropServices.Marshal.ReleaseComObject(m_xlApp.Application);

System.Runtime.InteropServices.Marshal.ReleaseComObject(m_xlApp);
                    m_xlApp = null;
                }
                catch { }
                try
                {
                    //清理垃圾进程
                    this.killProcessThread();
                }
                catch { }
                GC.Collect();
            }
        }
        /// <summary>
        /// 杀掉不死进程
        /// </summary>
        private void killProcessThread()
        {
            //           ArrayList myProcess = new ArrayList();
            Process[] myProcess = Process.GetProcesses();
            for (int i = 0; i < myProcess.Length; i++)
            {
                if (myProcess[i].ProcessName.StartsWith("Excel"))
                {
                    try
                    {

//System.Diagnostics.Process.GetProcessById(int.Parse((string)myProcess[i])).Kil
l();
                        myProcess[i].Kill();
                    }
                    catch { }
                }
            }
        }
```