文件和流的操作在System.IO命名空间下

**概念：流（Stream）与byte**

流是用于读取数据（byte）的一个类，如我们要读取文件的数据使用文件流（FileStream）,读取内存数据使用内存流（MemoryStream），读取网络数据使用网络流（NetworkStream）

**流（Stream）**

了解了这个段落，就了解了本篇文章的精髓了，如下，看注释

Stream stream = File.Create("./text.txt");

// 将字符串转为byte

byte[] bs = System.Text.Encoding.Unicode.GetBytes("Text");

// 设置流操作的位置，从0开始操作会覆盖掉之前的数据

stream.Position = 0;

// 将byte写入流，第2、3个参数表示从bs的0位置开始，写入bs.Length个byte

// 由于我们写入整个bs，所以写0和bs.Length

stream.Write(bs, 0, bs.Length);

// 清空缓存，该操作会将流的byte写入目标，这里是文件（不调用可能会引起数据丢失）

stream.Flush();

// 新建一个byte数组，用于读取流

byte[] rbs = new byte[stream.Length];

// 设置操作位置为0，因为我们打算读取整个流

stream.Position = 0;

// 读取流到byte中，第2、3个参数表示从rbs的0位置开始，写入rbs.Length个byte到rbs中

stream.Read(rbs, 0, rbs.Length);

// 将byte转为字符串

string s = System.Text.Encoding.Unicode.GetString(rbs);

// 输出字符串

Console.WriteLine(s);

// 关闭流

stream.Close();

看不明白就看前面的文章《byte，字符，编码》

**文件操作File**

文件的操作再File类下，包括创建，拷贝，移动，删除

如果我们要写入byte数据，可以使用流进行操作，如果是字符串，可以使用File提供的静态方法操作

// 创建文件，返回文件流，如果文件存在，则覆盖

FileStream stream = File.Create("./text.txt");

stream.Close();

// 写入文本，该方法会清空之前的文本

File.WriteAllText("./text.txt", "AAAAA");

File.WriteAllText("./text.txt", "BBBBB");

// 追加文本

File.AppendAllText("./text.txt", "CCCCC");

// 读取文本

string txt = File.ReadAllText("./text.txt");

// 打开文件，如果文件不存在则创建

FileStream stream2 = File.Open("./text.txt", FileMode.OpenOrCreate);

**目录操作Directory**

Directory包含目录的创建，移动，删除等操作

// 创建目录

Directory.CreateDirectory("AAA");

Stream stream = File.Create("./AAA/Text");

stream.Close();

// 获取子目录

Directory.GetDirectories("./AAA");

// 获取子文件

Directory.GetFiles("./AAA");

**路径Path**

Path包含路径的操作方法

// 获取文件的完整路径

Path.GetFullPath("./AAA/Text.txt");

// 获取文件名（包含扩展名）

Path.GetFileName("./AAA/Text.txt");

// 获取文件名（不包含扩展名）

Path.GetFileNameWithoutExtension("./AAA/Text.txt");

// 获取扩展名

Path.GetExtension("./AAA/Text.txt");

**具体实现流**

.NET提供了如下的具体的流

FileStream - 用于对文件进行读取和写入操作。

IsolatedStorageFileStream - 用于对独立存储中的文件进行读取或写入操作。

MemoryStream - 用于作为后备存储对内存进行读取和写入操作。

BufferedStream - 用于改进读取和写入操作的性能。

NetworkStream - 用于通过网络套接字进行读取和写入。

PipeStream - 用于通过匿名和命名管道进行读取和写入。

CryptoStream - 用于将数据流链接到加密转换。

**Zip压缩和解压**

ZipFile和ZipArchive用于Zip压缩和解压缩，详情看API

string startPath = @".\start";

string zipPath = @".\result.zip";

string extractPath = @".\extract";

// 压缩目录

ZipFile.CreateFromDirectory(startPath, zipPath);

// 导出目录

ZipFile.ExtractToDirectory(zipPath, extractPath);

**Gzip压缩和解压**

<https://docs.microsoft.com/zh-cn/dotnet/api/system.io.compression.gzipstream?view=net-6.0>