# File

# 一、方法

****1、读文件：****

// 打开一个文件，将文件的内容读入一个字符串，然后关闭该文件。

public static byte[] ReadAllBytes(string path);

// 打开一个文本文件，读取文件的所有行，然后关闭该文件。

public static string[] ReadAllLines(string path);

// 打开一个文件，使用指定的编码读取文件的所有行，然后关闭该文件。

public static string[] ReadAllLines(string path, Encoding encoding);

// 打开一个文本文件，读取文件的所有行，然后关闭该文件。

public static string ReadAllText(string path);

// 打开一个文件，使用指定的编码读取文件的所有行，然后关闭该文件。

public static string ReadAllText(string path, Encoding encoding);

// 读取文件的文本行。

public static IEnumerable<string> ReadLines(string path);

// 读取具有指定编码的文件的文本行。

public static IEnumerable<string> ReadLines(string path, Encoding encoding);

****2、写文件：****

// 创建一个新文件，在其中写入指定的字节数组，然后关闭该文件。如果目标文件已存在，则覆盖该文件。

public static void WriteAllBytes(string path, byte[] bytes);

// 创建一个新文件，在其中写入一组字符串，然后关闭该文件。

public static void WriteAllLines(string path, IEnumerable<string> contents);

// 创建一个新文件，在其中写入指定的字符串数组，然后关闭该文件。

public static void WriteAllLines(string path, string[] contents);

// 使用指定的编码创建一个新文件，在其中写入一组字符串，然后关闭该文件。

public static void WriteAllLines(string path, IEnumerable<string> contents, Encoding encoding);

// 创建一个新文件，使用指定的编码在其中写入指定的字符串数组，然后关闭该文件。

public static void WriteAllLines(string path, string[] contents, Encoding encoding);

// 创建一个新文件，在其中写入指定的字符串，然后关闭文件。如果目标文件已存在，则覆盖该文件。

public static void WriteAllText(string path, string contents);

// 创建一个新文件，在其中写入指定的字符串，然后关闭文件。如果目标文件已存在，则覆盖该文件。

public static void WriteAllText(string path, string contents, Encoding encoding);

****3、追加内容：****

// 在一个文件中追加文本行，然后关闭该文件。

public static void AppendAllLines(string path, IEnumerable<string> contents);

// 使用指定的编码向一个文件中追加文本行，然后关闭该文件。

public static void AppendAllLines(string path, IEnumerable<string> contents, Encoding encoding);

// 打开一个文件，向其中追加指定的字符串，然后关闭该文件。如果文件不存在，此方法创建一个文件，将指定的字符串写入文件，然后关闭该文件。

public static void AppendAllText(string path, string contents);

// 将指定的字符串追加到文件中，如果文件还不存在则创建该文件。

public static void AppendAllText(string path, string contents, Encoding encoding);

// 创建一个 System.IO.StreamWriter，它将 UTF-8 编码文本追加到现有文件。

public static StreamWriter AppendText(string path);

****4、创建文件：****

// 在指定路径中创建或覆盖文件。

public static FileStream Create(string path);

// 创建或覆盖指定的文件。

public static FileStream Create(string path, int bufferSize);

// 创建或覆盖指定的文件，并指定缓冲区大小和一个描述如何创建或覆盖该文件的 System.IO.FileOptions 值。

public static FileStream Create(string path, int bufferSize, FileOptions options);

// 创建或覆盖具有指定的缓冲区大小、文件选项和文件安全性的指定文件。

public static FileStream Create(string path, int bufferSize, FileOptions options, FileSecurity fileSecurity);

****5、打开文件：****

// 打开指定路径上的 System.IO.FileStream，具有读/写访问权限。

public static FileStream Open(string path, FileMode mode);

// 以指定的模式和访问权限打开指定路径上的 System.IO.FileStream。

public static FileStream Open(string path, FileMode mode, FileAccess access);

// 打开指定路径上的 System.IO.FileStream，具有指定的读、写或读/写访问模式以及指定的共享选项。

public static FileStream Open(string path, FileMode mode, FileAccess access, FileShare share);

// 打开现有文件以进行读取。

public static FileStream OpenRead(string path);

****6、获取和设置文件属性：****

// 获取一个 System.Security.AccessControl.FileSecurity 对象，它封装指定文件的访问控制列表 (ACL) 条目。

public static FileSecurity GetAccessControl(string path);

// 获取一个 System.Security.AccessControl.FileSecurity 对象，它封装特定文件的指定类型的访问控制列表 (ACL)项。

public static FileSecurity GetAccessControl(string path, AccessControlSections includeSections);

// 获取在此路径上的文件的 System.IO.FileAttributes。

public static FileAttributes GetAttributes(string path);

// 返回指定文件或目录的创建日期和时间。

public static DateTime GetCreationTime(string path);

// 返回指定的文件或目录的创建日期及时间，其格式为协调世界时 (UTC)。

public static DateTime GetCreationTimeUtc(string path);

// 返回上次访问指定文件或目录的日期和时间。

public static DateTime GetLastAccessTime(string path);

// 返回上次访问指定的文件或目录的日期及时间，其格式为协调世界时 (UTC)。

public static DateTime GetLastAccessTimeUtc(string path);

// 返回上次写入指定文件或目录的日期和时间。

public static DateTime GetLastWriteTime(string path);

// 返回上次写入指定的文件或目录的日期和时间，其格式为协调世界时 (UTC)。

public static DateTime GetLastWriteTimeUtc(string path);

// 对指定的文件应用由 System.Security.AccessControl.FileSecurity 对象描述的访问控制列表 (ACL) 项。

public static void SetAccessControl(string path, FileSecurity fileSecurity);

// 设置指定路径上文件的指定的 System.IO.FileAttributes。

public static void SetAttributes(string path, FileAttributes fileAttributes);

// 设置创建该文件的日期和时间。

public static void SetCreationTime(string path, DateTime creationTime);

// 设置文件创建的日期和时间，其格式为协调世界时 (UTC)。

public static void SetCreationTimeUtc(string path, DateTime creationTimeUtc);

// 设置上次访问指定文件的日期和时间。

public static void SetLastAccessTime(string path, DateTime lastAccessTime);

// 设置上次访问指定的文件的日期和时间，其格式为协调世界时 (UTC)。

public static void SetLastAccessTimeUtc(string path, DateTime lastAccessTimeUtc);

// 设置上次写入指定文件的日期和时间。

public static void SetLastWriteTime(string path, DateTime lastWriteTime);

// 设置上次写入指定的文件的日期和时间，其格式为协调世界时 (UTC)。

public static void SetLastWriteTimeUtc(string path, DateTime lastWriteTimeUtc);

****7、复制、移动、替换：****

// 将现有文件复制到新文件。不允许覆盖同名的文件。

public static void Copy(string sourceFileName, string destFileName);

// 将现有文件复制到新文件。允许覆盖同名的文件。

public static void Copy(string sourceFileName, string destFileName, bool overwrite);

// 将指定文件移到新位置，并提供指定新文件名的选项。

public static void Move(string sourceFileName, string destFileName);

// 使用其他文件的内容替换指定文件的内容，这一过程将删除原始文件，并创建被替换文件的备份。

public static void Replace(string sourceFileName, string destinationFileName, string destinationBackupFileName);

// 用其他文件的内容替换指定文件的内容，删除原始文件，并创建被替换文件的备份和（可选）忽略合并错误。

public static void Replace(string sourceFileName, string destinationFileName, string destinationBackupFileName, bool ignoreMetadataErrors);

****8、加密解密、删除、判定是否存在：****

// 将某个文件加密，使得只有加密该文件的帐户才能将其解密。

public static void Encrypt(string path);

// 解密由当前帐户使用 System.IO.File.Encrypt(System.String) 方法加密的文件。

public static void Decrypt(string path);

// 删除指定的文件。如果指定的文件不存在，则不引发异常

public static void Delete(string path);

// 确定指定的文件是否存在。

public static bool Exists(string path);