Работа с файлами

Преподаватель: Тазиева Рамиля Фаридовна

Основные понятия

Под **файлом** подразумевается именованная информация на внешнем носителе.

Логически файл можно представить как конечное количество последовательных байтов.

Передача данных с внешнего устройства в оперативную память называется **чтением**, или вводом, обратный процесс – **записью**, или выводом.

Обмен данных реализуется с помощью **потоков**.

Поток (stream) –это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику. Поток определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен.

Обмен с потоком для повышения скорости передачи производится через специальную область оперативной памяти – **буфер**.

При записи в файл вся **информация** сначала **направляется в буфер** и там накапливается, пока буфер не заполнится. После этого происходит передача данных на внешнее устройство.

При чтении из файла **данные** вначале **считываются в буфер**, что позволяет быстро и эффективно обмениваться информацией с внешними устройствами.

Пространство имен System.IO

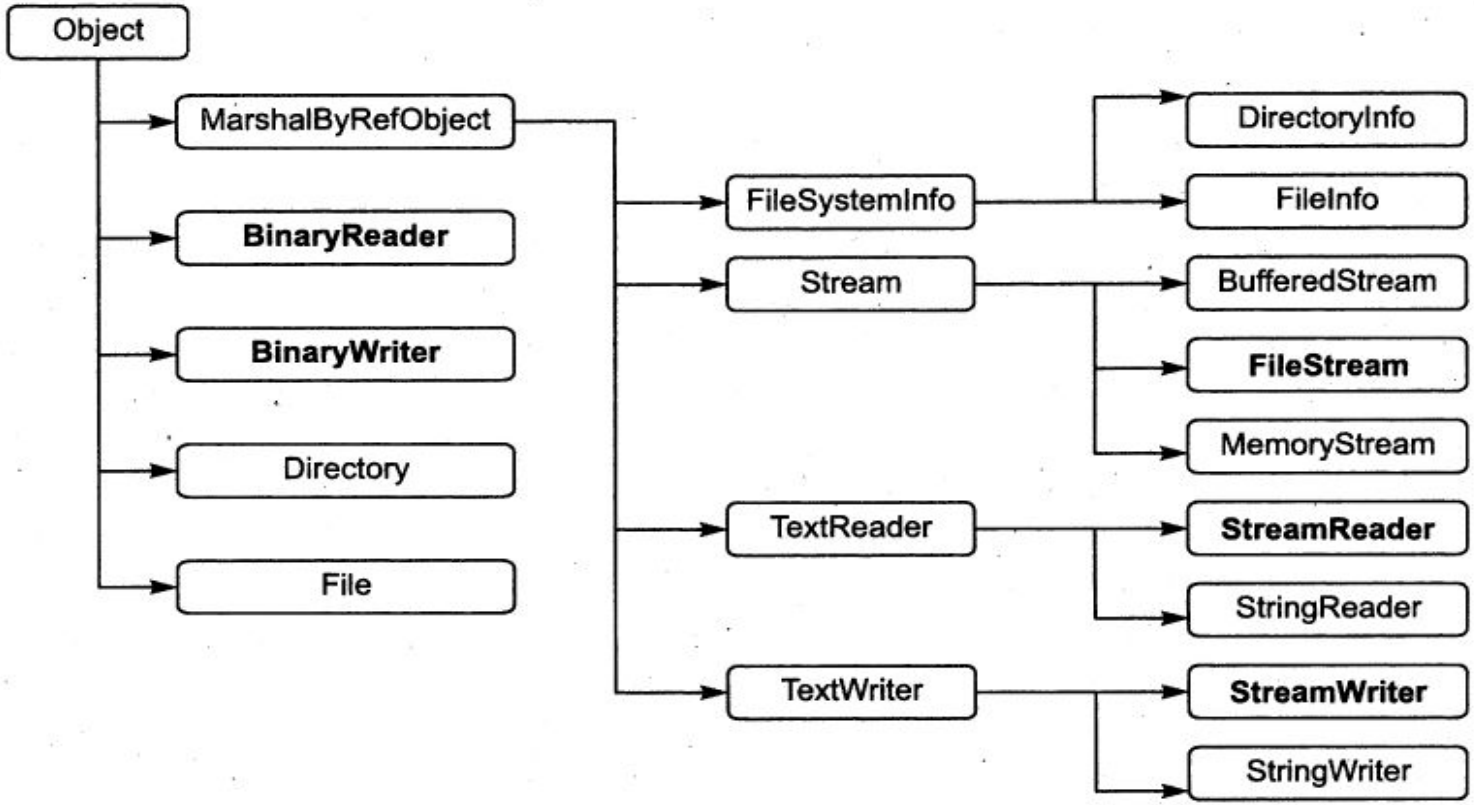

Рис. 11.1. Классы библиотеки .NET для работы с потоками

Обмен с внешними устройствами можно выполнять на уровне:

- Двоичного представления данных (BinaryReader, BinaryWriter).
- Байтов (FileStream).
- Текста, то есть символов (StreamWriter, StreamReader).

Описание классов для работы с файлами

Класс	Описание
BinaryReader BinaryWriter	Чтение и запись простых встроенных типов (целочисленных, логических, строковых и т.п.) во внутренней форме представления
BufferedStream	Временное хранение потока байтов (например для последующего переноса в постоянное хранилище)
Directory, DirectoruInfo File, FileInfo	Работа с каталогами или физическими файлами: создание, удаление, получение свойств
FileStream	Произвольный доступ к файлу, представленному как поток байтов
MemoryStream	Произвольный доступ к потоку байтов в оперативной памяти
StreamWriter StreamReader	Чтение из файла и запись в файл текстовой информации (произвольный доступ не поддерживается)
StringWriter StringReader	Работа с текстовой информацией в оперативной памяти

Доступ к файлам

Последовательный – очередной элемент можно прочитать (записать) только после аналогичной операции с предыдущим элементом.

Произвольный (прямой) – выполняется чтение (запись) произвольного элемента по заданному адресу.

Использование классов файловых потоков предполагает следующие операции.

1. Создание потока и связывание его с физическим файлом.
2. Обмен (ввод-вывод).
3. Закрытие файла.

Режимы доступа к файлу в перечислении FileAccess

Значение	Описание
Read	Открыть файл только для чтения
ReadWrite	Открыть файл для чтения и записи
Write	Открыть файл только для записи

Режимы открытия файла в перечислении FileMode

Значение	Описание
Append	Открыть файл, если он существует, и установить текущий указатель в конец файла. Если файл не существует, создать новый файл
Create	Создать новый файл. Если в каталоге уже существует файл с таким же именем, он будет стерт
CreateNew	Создать новый файл. Если в каталоге уже существует файл с таким же именем, возникает исключение IOException
Open	Открыть существующий файл
OpenOrCreate	Открыть файл, если он существует. Если нет, создать файл с таким именем
Truncate	Открыть существующий файл. После открытия он должен быть обрезан до нулевой длины

Потоки байтов. Класс FileStream

Элемент	Описание
CanRead, CanSeek, CanWrite	Свойства, определяющие, какие операции поддерживает поток: чтение, прямой доступ и/или запись
Close	Закрыть текущий поток и освободить связанные с ним ресурсы (сокеты, указатели на файлы и т. п.)
EndRead, EndWrite	Ожидать завершения асинхронного ввода; закончить асинхронный вывод
Flush	Записать данные из буфера в связанный с потоком источник данных и очистить буфер. Если для данного потока буфер не используется, то этот метод ничего не делает
Length	Возвратить длину потока в байтах
Position	Возвратить текущую позицию в потоке
Read, ReadByte	Считать последовательность байтов (или один байт) из текущего потока и переместить указатель в потоке на количество считанных байтов
Seek	Установить текущий указатель потока на заданную позицию
SetLength	Установить длину текущего потока
Write, WriteByte	Записать последовательность байтов (или один байт) в текущий поток и переместить указатель в потоке на количество записанных байтов

Пример работы с потоком байтов

```
namespace ConsoleApp1
{   class Program
   {      static void Main(string[] args)
      {
         FileStream f = new FileStream("text.txt", FileMode.Create, FileAccess.ReadWrite);
         f.WriteByte(100);  // записывает число 100 в начало файла

         byte[] x = new byte[10];
         for (byte i=0;i<10;++i)
         {
            x[i] = (byte)(10 - i);
            f.WriteByte(i);  // записывает 10 чисел о 0 до 9
         }

         f.Write(x, 0, 5);  // записывает первые 5 элементов массива

         byte[] y = new byte[20];
         f.Seek(0, SeekOrigin.Begin); //установка указателя на нача
         f.Read(y, 0, 20);   // чтение из файла в массив

         foreach (byte elem in y) Console.Write(" " + elem);
         Console.WriteLine();

         f.Seek(5, SeekOrigin.Begin);//установка указателя на 5-й элемент
         int a = f.ReadByte(); // чтение 5-го элемента
         Console.WriteLine(a);
         a = f.ReadByte(); // чтение 6-го элемента
         Console.WriteLine(a);

         Console.WriteLine("Текущая позиция в потоке "+f.Position);
         f.Close();
      }   }}
```

C:\Windows\system32\cmd.exe

```
 100 0 1 2 3 4 5 6 7 8 9 10 9 8 7 6 0 0 0 0
4
5
Текущая позиция в потоке 7
Для продолжения нажмите любую клавишу . . .
```

Исключения

Исключение	Описание
FileNotFoundException	файл с указанным именем в указанном каталоге не существует
DirectoryNotFoundException	Не существует указанный каталог
ArgumentException	Неверно задан режим открытия файла
IOException	Файл не открывается из-за ошибок ввода-вывода

```
static void Main(string[] args)
        { try
            {
                FileStream f = new FileStream("text1.txt", FileMode.Open, FileAccess.ReadWrite);
                f.Close();
            }
            catch (FileNotFoundException e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine("Проверьте правильность имени файла");
                return;
            }
            catch (Exception  e)
            {
                Console.WriteLine("Error: "+ e.Message);
                return;
            }
 }
```

C:\Windows\system32\cmd.exe

```
Файл 'C:\Users\Рамиля\source\repos\ConsoleApp2\ConsoleApp2\bin\Debug\text1.txt' не найден.
Проверьте правильность имени файла
Для продолжения нажмите любую клавишу . . .
```

StreamWriter

Методы	Чтение из файла. Класс StreamReader Описание
Close	закрывает считываемый файл и освобождает все ресурсы
Peek	возвращает следующий доступный символ, если символов больше нет, то возвращает -1
Read	считывает и возвращает следующий символ в численном представлении. Имеет перегруженную версию: Read(char[] array, int index, int count), где array - массив, куда считываются символы, index - индекс в массиве array, начиная с которого записываются считываемые символы, и count - максимальное количество считываемых символов
ReadLine	считывает одну строку в файле
ReadToEnd	считывает весь текст из файла

Методы	Запись в файл. . Класс StreamWriter Описание
Close	закрывает записываемый файл и освобождает все ресурсы
Flush	записывает в файл оставшиеся в буфере данные и очищает буфер
Write	записывает в файл данные простейших типов, как int, double, char, string и т.д.
WriteLine	также записывает данные, только после записи добавляет в файл символ окончания строки

Пример чтения из файла

```
string path= @"C:\SomeDir\hta.txt";
try
{
    Console.WriteLine("***считываем весь файл***");
    using (StreamReader sr = new StreamReader(path))
    {
        Console.WriteLine(sr.ReadToEnd());
    }
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
```

```
Console.WriteLine("**считываем построчно**");
    using (StreamReader sr = new StreamReader(path,
System.Text.Encoding.Default))
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
    }
```

```
Console.WriteLine("******считываем блоками********");
using (StreamReader sr = new StreamReader(path, System.Text.Encoding.Default))
{
    char[] array = new char[4];
    // считываем 4 символа
    sr.Read(array, 0, 4);
```

Пример записи в текстовый файл

```
string readPath= @"C:\SomeDir\hta.txt";
string writePath = @"C:\SomeDir\ath.txt";

string text = "";
try
{
    using (StreamReader sr = new StreamReader(readPath, System.Text.Encoding.Default))
    {
        text=sr.ReadToEnd();
    }
    using (StreamWriter sw = new StreamWriter(writePath, false, System.Text.Encoding.Default))
    {
        sw.WriteLine(text);
    }

    using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))
    {
        sw.WriteLine("Дозапись");
        sw.Write(4.5);
    }
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
```

true - новые данные добавляются в конце к уже имеющимся данным. false, то файл перезаписывается

Кодировка, в которой записывается файл.

Двоичные потоки: BinaryWriter и BinaryReader

Методы	Класс BinaryWriter Описание
Close()	закрывает поток и освобождает ресурсы
Flush()	очищает буфер, дописывая из него оставшиеся данные в файл
Seek()	устанавливает позицию в потоке
Write()	записывает данные в поток

Методы	Класс BinaryReader Описание
Close()	закрывает поток и освобождает ресурсы
ReadBoolean()	считывает значение bool и перемещает указатель на один байт
ReadDecimal()	считывает значение decimal и перемещает указатель на 16 байт
ReadByte()	считывает один байт и перемещает указатель на один байт
ReadChar()	считывает значение char, то есть один символ, и перемещает указатель на столько байтов, сколько занимает символ в текущей кодировке
ReadDouble()	считывает значение double и перемещает указатель на 8 байт
ReadInt16()	считывает значение short и перемещает указатель на 2 байта
ReadInt32()	считывает значение int и перемещает указатель на 4 байта
ReadSingle()	считывает значение float и перемещает указатель на 4 байта
ReadString()	считывает значение string. Каждая строка предваряется значением длины строки, которое представляет 7-битное целое число

Пример формирования двоичного файла

```
class Program
    {
        static void Main(string[] args)
        {
            try
            {
                BinaryWriter f = new BinaryWriter(new FileStream(@"D:\Информационные
технологии\states.txt", FileMode.Create));
                double d = 0;
                while (d<4)
                {
                    f.Write(d);
                    d +=0.33;
                }

                f.Seek(16, SeekOrigin.Begin); // второй элемент файла
                f.Write(888d);
                f.Close();
            }
            catch (FileNotFoundException e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine("Проверьте правильность имени файла");
                return;
            }
            catch (Exception  e)
            {
                Console.WriteLine("Error: "+ e.Message);
                return;
            }        }    }
```

Пример чтения из двоичного файла

```
FileStream f=new FileStream(@"D:\Информационные технологии\states.txt", FileMode.Open);
      BinaryReader fin = new BinaryReader(f);

      long n = f.Length /8;  //количество чисел в файле
      double[] x = new double[n];

      long i = 0;
      try
         {
            while (true) x[i++] = fin.ReadDouble();
         }

      catch (EndOfStreamException e) { }

      foreach (double d in x) Console.Write(" " + d);
      fin.Close();
      f.Close();
```

C:\Windows\system32\cmd.exe

```
0 0,33 888 0,99 1,32 1,65 1,98 2,31 2,64 2,97 3,3 3,63 3,96
```

Работа с каталогами. Класс Directory и DirectoryInfo

Методы	Класс Directory . Описание
CreateDirectory(path)	создает каталог по указанному пути path
Delete(path)	удаляет каталог по указанному пути path
Exists(path)	определяет, существует ли каталог по указанному пути path. Если существует, возвращается true, если не существует, то false
GetDirectories(path)	получает список каталогов в каталоге path
GetFiles(path)	получает список файлов в каталоге path
Move(sourceDirName, destDirName):	перемещает каталог
GetParent(path)	получение родительского каталога

Методы	Класс DirectoryInfo. Описание
Create()	создает каталог
CreateSubdirectory(path)	создает подкаталог по указанному пути path
Delete()	удаляет каталог
Свойство **Exists**	определяет, существует ли каталог
GetDirectories()	получает список каталогов
GetFiles()	получает список файлов
MoveTo(destDirName)	перемещает каталог
Свойство **Parent**	получение родительского каталога
Свойство **Root**	получение корневого каталога

Создание каталога

```
string path = @"C:\SomeDir";
string subpath = @"program\avalon";
DirectoryInfo dirInfo = new
DirectoryInfo(path);
if (!dirInfo.Exists)
{
    dirInfo.Create();
}
dirInfo.CreateSubdirectory(subpath);
```

Перемещение каталога

```
string string oldPath = @"C:\SomeFolder";
string newPath = @"C:\SomeDir";
DirectoryInfo dirInfo = new
DirectoryInfo(oldPath);
if (dirInfo.Exists &&
Directory.Exists(newPath) == false)
{
    dirInfo.MoveTo(newPath);
}
```

Получение информации о каталоге

```
string dirName = "C:\\Program Files";

DirectoryInfo dirInfo = new DirectoryInfo(dirName);

Console.WriteLine("Название каталога: {0}", dirInfo.Name);
Console.WriteLine("Полное название каталога: {0}",
dirInfo.FullName);
Console.WriteLine("Время создания каталога: {0}",
dirInfo.CreationTime);
Console.WriteLine("Корневой каталог: {0}", dirInfo.Root);
```

Получение списка файлов и подкаталогов

```
string dirName = "C:\\";

if (Directory.Exists(dirName))
{
    Console.WriteLine("Подкаталоги:");
    string[] dirs = Directory.GetDirectories(dirName);
    foreach (string s in dirs)
    {
        Console.WriteLine(s);
    }
    Console.WriteLine();
    Console.WriteLine("Файлы:");
    string[] files = Directory.GetFiles(dirName);
    foreach (string s in files)
    {
        Console.WriteLine(s);
    }
}
```

Удаление каталога

```
string dirName = @"C:\SomeFolder“;
try
{
    DirectoryInfo dirInfo = new
DirectoryInfo(dirName);
    dirInfo.Delete(true);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

Работа с файлами. Классы File и FileInfo

Методы	Класс FileInfo . Описание
CopyTo(path)	копирует файл в новое место по указанному пути path
Create()	создает файл
Delete():	удаляет файл
MoveTo(destFileName)	перемещает файл в новое место
Свойство Directory	получает родительский каталог в виде объекта DirectoryInfo
Свойство DirectoryName	получает полный путь к родительскому каталогу
Свойство Exists	указывает, существует ли файл
Свойство Length	получает размер файла
Свойство Extension	получает расширение файла
Свойство Name	получает имя файла
Свойство FullName	получает полное имя файла

Методы	Класс File . Описание
Copy()	копирует файл в новое место
Create()	создает файл
Delete()	удаляет файл
Move	перемещает файл в новое место
Exists(file)	определяет, существует ли файл

Получение информации о файле

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
    Console.WriteLine("Имя файла: {0}",
fileInf.Name);
    Console.WriteLine("Время создания: {0}",
fileInf.CreationTime);
    Console.WriteLine("Размер: {0}",
fileInf.Length);
}
```

Перемещение файла

```
string path = @"C:\apache\hta.txt";
string newPath = @"C:\SomeDir\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
   fileInf.MoveTo(newPath);
   // альтернатива с помощью класса File
   // File.Move(path, newPath);
}
```

Удаление файла

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
   fileInf.Delete();
   // альтернатива с помощью класса File
   // File.Delete(path);
}
```

Копирование файла

```
string path = @"C:\apache\hta.txt";
string newPath = @"C:\SomeDir\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
   fileInf.CopyTo(newPath, true);
   // альтернатива с помощью класса File
   // File.Copy(path, newPath, true);
}
```

Метод CopyTo класса FileInfo принимает два параметра: путь, по которому файл будет копироваться, и булевое значение, которое указывает, надо ли при копировании перезаписывать файл (если true, как в случае выше, файл при копировании перезаписывается). Если же в качестве последнего параметра передать значение false, то если такой файл уже существует, приложение выдаст ошибку.