Streams, Files and Directories

File Types, Using Streams and Manipulating Files

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

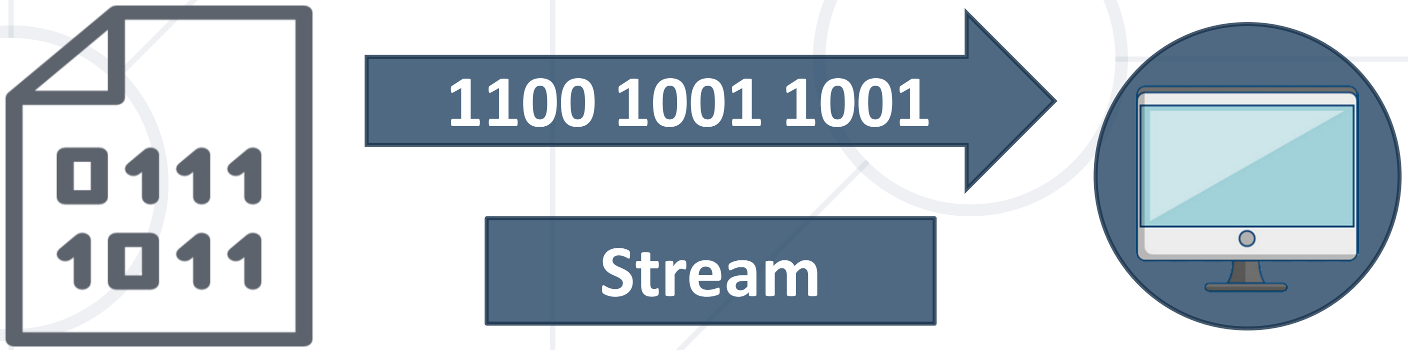
https://softuni.bg

Table of Contents

What Are Streams?

What is a Stream?

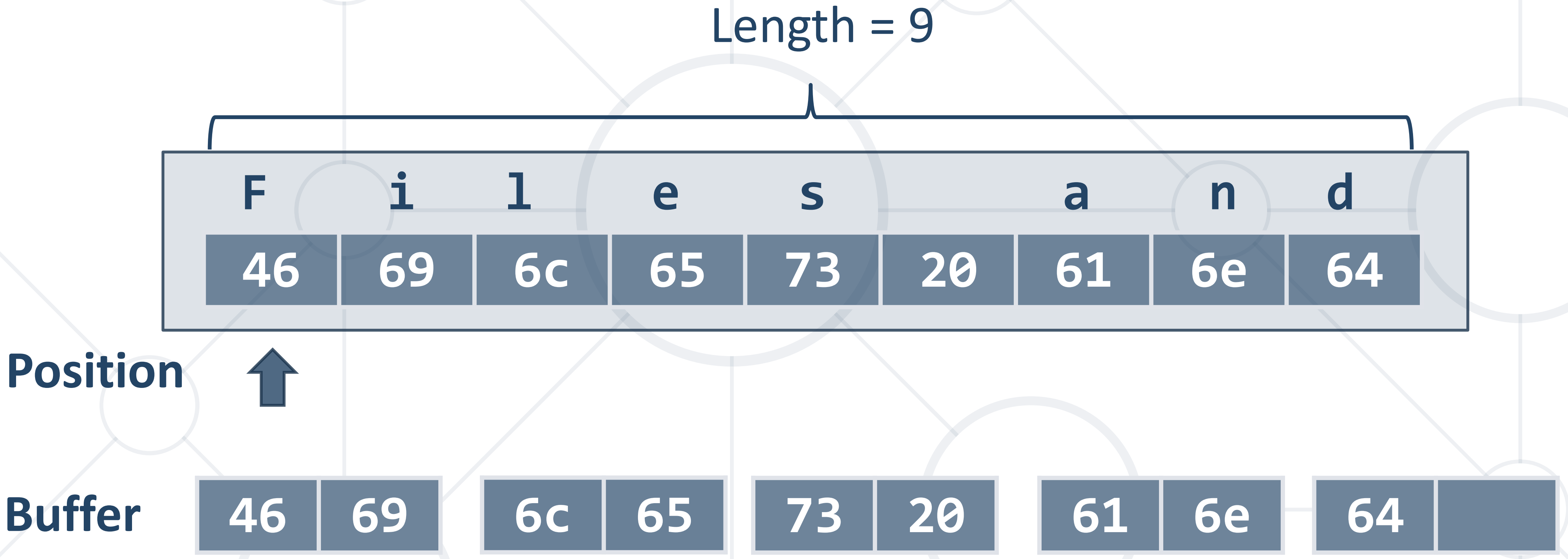
- Streams are used to **transfer data**
- We open a stream to:
  - **Read** data
  - **Write** data



Stream Basics

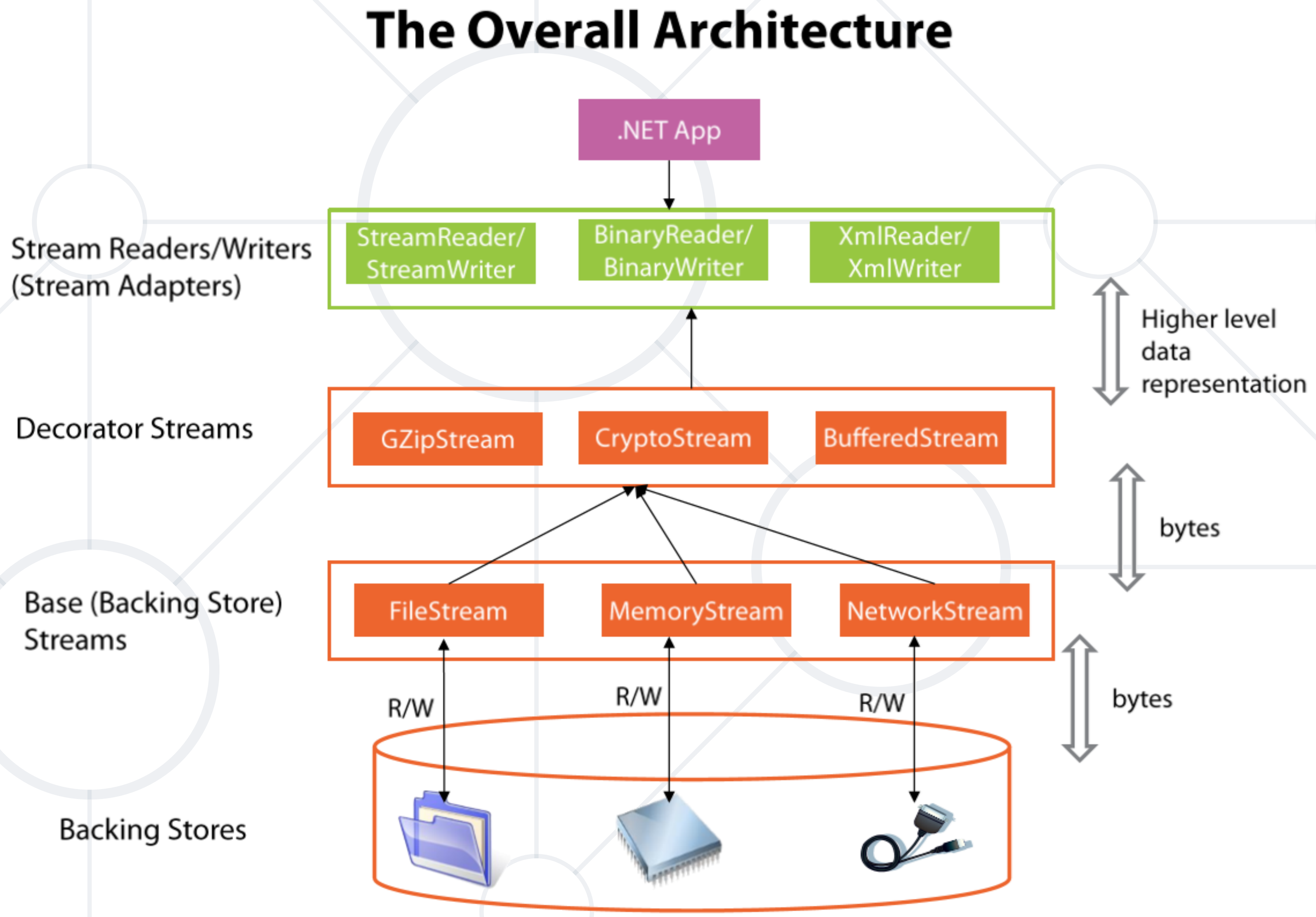
- Streams are means for **transferring** (reading and writing) **data**
- Streams are ordered **sequences of bytes**
  - Provide **sequential** access to its elements
- Different types of streams are available to access different data sources:
  - **File** access, **network** access, **memory** streams and others
- Streams are opened **before** using them and closed **after** that

Streams and Buffering – Example



- **Position** is the current position in the stream
- **Buffer** keeps **n** bytes of the stream from the current position

Stream Types in .NET



Readers and Writers in C#

Using StreamReader

- **StreamReader** in C# read **text** from a file / stream
- The `using(…)` statement closes properly the stream at the end

```
var reader = new StreamReader(fileName);
using (reader)
{
  // Use the reader here, e.g.
  // string line = reader.readLine();
}
```

Problem: Odd Lines

- Read the content from your **`input.txt`** file
- Print the **odd lines** on the console
- Counting starts from **0**

```
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
```

```
In fair Verona, where we lay our scene,
Where civil blood makes civil hands unclean.
```

Solution: Odd Lines

```
var reader = new StreamReader("input.txt");
using (reader) {
  int counter = 0;
  string line = reader.ReadLine();
  using (var writer = new StreamWriter("output.txt")) {
    while (line != null)
      if (counter % 2 == 1)
        writer.WriteLine(line);
    counter++;
    line = reader.ReadLine();
  }
}
```

Problem: Line Numbers

- Read the file **input.txt**
- Insert a **line number** in front of each line of the file
- Save it in **output.txt**

```
Two households, both alike in dignity,
In fair Verona, where we lay our scene,
From ancient grudge break to new mutiny,
Where civil blood makes civil hands unclean.
```

```
1. Two households, both alike in dignity,
2. In fair Verona, where we lay our scene,
3. From ancient grudge break to new mutiny,
4. Where civil blood makes civil hands unclean.
```

Solution: Line Numbers

```
using (var reader = new StreamReader("input.txt"))
{
  string line = reader.ReadLine();
  int counter = 1;
  using (var writer = new StreamWriter("output.txt"))
    while (line != null)
    {
      writer.WriteLine($"{counter}. {line}");
      line = reader.ReadLine();
      counter++;
    }
}
```

Try-Catch-Finally Example

```
StreamReader reader = null;
int linesCount = 0;
try {
  reader = new StreamReader("input.txt");
  while (reader.ReadLine() != null)
    linesCount++;
  Console.WriteLine("Lines count: {0}", linesCount);
}
catch (Exception ex) {
  Console.Error.WriteLine("Error reading file: {0}", ex);
}
finally {
  if (reader != null) reader.Close();
}
```

Instead of **try-finally**, you can use **using(reader)**

Reading / Writing Data from / to Files

File Streams

- **File streams** read / writes sequences of bytes from a file
- **Creating** a new binary file:

```
using (var fs = new FileStream("file.bin", FileMode.Create))
{
  // Write to the file: fs.Write(byte[]) …
}
```

- **Opening** existing file

```
using (var fs = new FileStream("file.bin", FileMode.Open))
{  // Read from file or write to the file …  }
```

Writing Text to File – Example

```
string text = "Кирилица";
var fileStream =
  new FileStream("log.txt", FileMode.Create);
using(fileStream)
{
  byte[] bytes = Encoding.UTF8.GetBytes(text);
  fileStream.Write(bytes, 0, bytes.Length);
}
```

Encoding.UTF8.GetBytes() returns the underlying bytes of the characters

Encrypt / Decrypt File with XOR

```
using (var fin = new FileStream("example.png", FileMode.Open))
using (var fout = new FileStream("example-encrypted.png", FileMode.Create))
{
  byte[] buffer = new byte[4096];
  while (true)
  {
    int bytesRead = fin.Read(buffer);
    if (bytesRead == 0) break;
    const byte secret = 183;
    for (int i = 0; i < bytesRead; i++)
      buffer[i] = (byte) (buffer[i] ^ secret);
    fout.Write(buffer, 0, bytesRead);
  }
}
```

Encrypting the read bytes with the constant parameter Secret using XOR operator

https://gist.github.com/nakov/1d39c4513cff83b8a735d7dc883dfe18

File Class in .NET

Reading Text Files

- File.ReadAllText() → string - reads a text file at once

```
using System.IO;
…
string text = File.ReadAllText("file.txt");
```

- File.ReadAllLines() → string[] - reads a text file's lines

```
using System.IO;
…
string[] lines = File.ReadAllLines("file.txt");
```

Writing Text Files

- Writing a **string** to a text file:

```
File.WriteAllText("output.txt", "Files are fun :)");
```

- Writing a **sequence** of strings to a text file, at separate lines:

```
string[] names = { "peter", "irina", "george", "maria" };
File.WriteAllLines("output.txt", names);
```

- **Appending** additional text to an existing file:

```
File.AppendAllText("output.txt", "\nMore text\n");
```

Reading / Writing Binary Files

- Writing a **byte[]** to a text file:

```
using System.IO;
…
byte[] bytesToWrite = { 0, 183, 255 };
File.WriteAllBytes("output.txt", bytesToWrite);
```

- Reading a binary file into **byte[]**:

```
using System.IO;
…
byte[] bytesRead = File.ReadAllBytes("binaryFile.txt");
```

Directory Class in .NET

Basic Directory Operations

- **Creating** a directory (with all its subdirectories at the specified path), unless they already exists:

```
Directory.CreateDirectory("TestFolder");
```

- **Deleting** a directory (with its contents):

```
Directory.Delete("TestFolder", true);
```

- **Moving** a file or a directory to a new location:

```
Directory.Move("Test", "New Folder");
```

Listing Directory Contents

- `GetFiles()` – returns the names of the files (including their paths) in the specified directory

```
string[] filesInDir =
  Directory.GetFiles("TestFolder");
```

- `GetDirectories()` – returns the names of the subdirectories (including their paths) in the specified directory

```
string[] subDirs =
  Directory.GetDirectories("TestFolder");
```

Problem: Calculate Folder Size

- You are given a folder named **TestFolder**
- Calculate the **size of all files in the folder** (with its subfolders)
- Print the result in a file "**output.txt**" in megabytes

output.txt
5.16173839569092

Solution: Calculate Folder Size

```
double sum = 0;

DirectoryInfo dir = new DirectoryInfo("TestFolder");
FileInfo[] infos = dir.GetFiles("*", SearchOption.AllDirectories);

foreach (FileInfo fileInfo in infos)
{
   sum += fileInfo.Length;
}

sum = sum / 1024 / 1024;

File.WriteAllText("output.txt", sum.ToString());
```

Gets all files from the given folder and its subfolders.

Summary

- **Streams** are ordered sequences of bytes
  - Can be **read** or **written**
  - Always close streams with **try-finally** or **using(…)**
- Use **StreamReader** / **StreamWriter** for text data
- Use **FileStream** to read / write binary files
- Use the **File** class to read / write files at once
- Use the **Directory** class to work with directories

Questions?

Software University

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

License