

ESEIG

**POLITÉCNICO
DO PORTO**

Algoritmia e Estruturas de Dados

2014/2015

Módulo II – C# Ficheiros de Texto

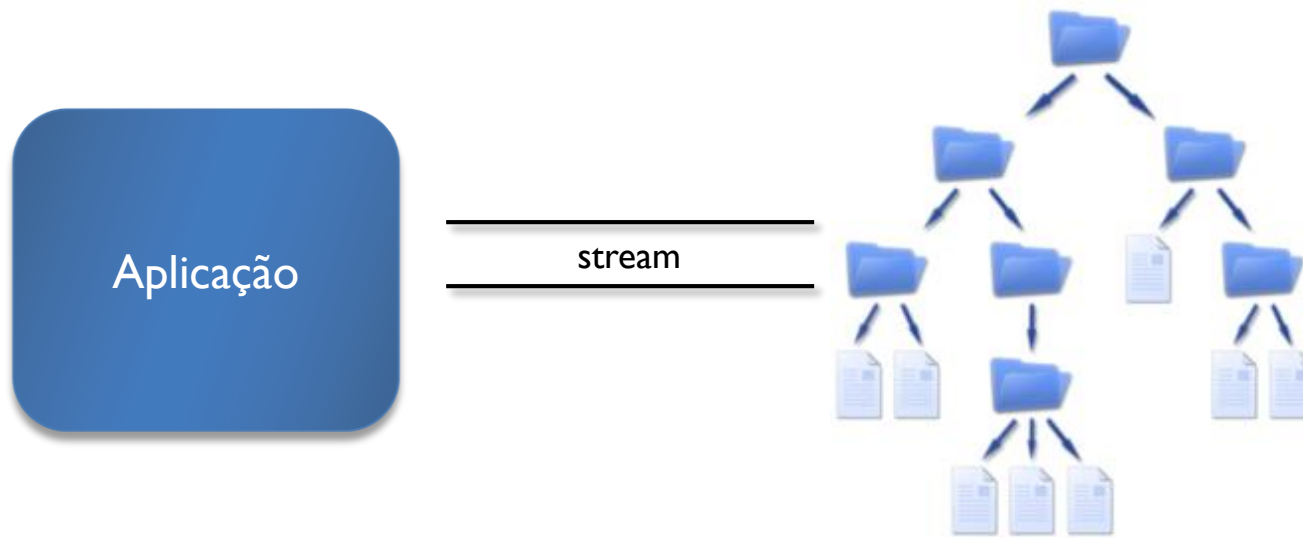


I. Ficheiros de Texto

1. Conceito
2. Tipos de ficheiros
3. O namespace `System.IO` e classes específicas
4. `StreamReader` e `StreamWriter`
5. A classe `File`
6. A classe `Directory`

▶ Conceito

- ▶ Armazenamento persistente de dados
- ▶ Principais conceitos a reter
 - ▶ Ficheiros – objetos físicos que contêm dados
 - ▶ Pastas – contentores de subpastas e ficheiros
 - ▶ Streams – canais de comunicação entre a aplicação e o ficheiro

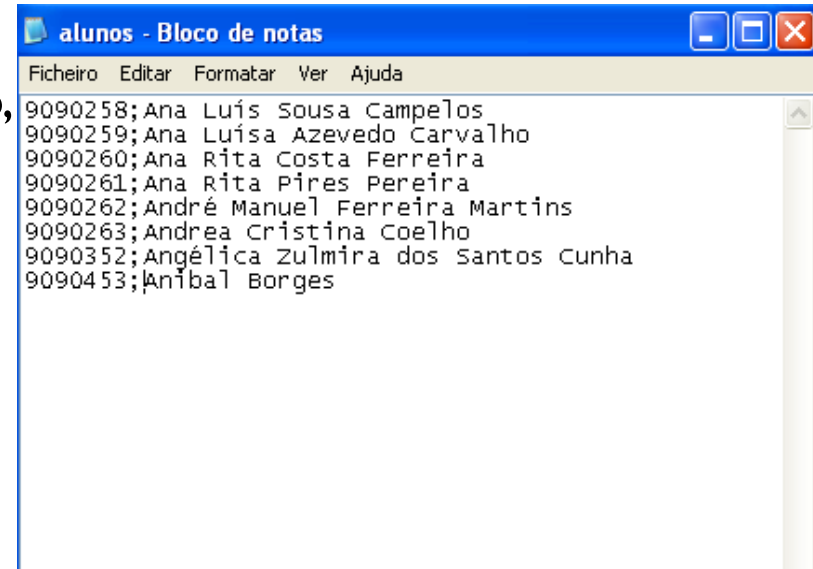


► Tipo de ficheiros

► Text Files – ficheiros de texto

Ficheiros constituídos por linhas de texto, de acesso sequencial.

São ficheiros de texto que podem ser acedidos por um editor de texto, como o bloco de notas



► Data Files – ficheiros de dados

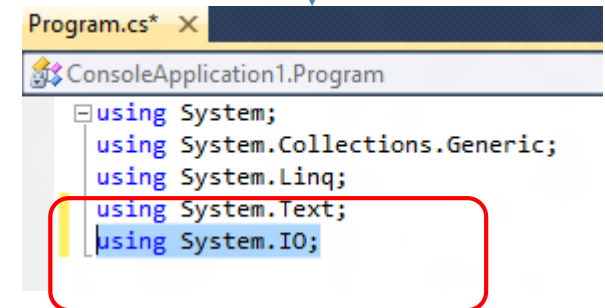
Ficheiros de acesso directo, não é necessário ler linha a linha; podemos aceder directamente aos dados que necessitamos.

São ficheiros indexados, pelo que não podem ser acedidos por um editor de texto como o bloco de notas

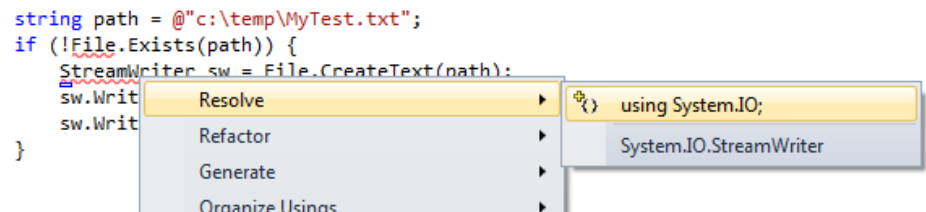
- ▶ O namespace System.IO
 - ▶ Conjunto de classes e tipos de dados que suportam a leitura, escrita e manipulação de ficheiros
 - ▶ Obrigatória a sua declaração no código

- ▶ Principais classes

- ▶ Classe File
- ▶ Classe Directory
- ▶ Classe Path
- ▶ Classe FileInfo
- ▶ Classe StreamReader
- ▶ Classe StreamWriter



```
Program.cs*  
ConsoleApplication1.Program  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.IO;
```



```
string path = @"c:\temp\MyTest.txt";  
if (!File.Exists(path)) {  
    StreamWriter sw = File.CreateText(path);  
    sw.Writ  
    sw.Writ  
}
```

Resolve
Refactor
Generate
Organize Usings

using System.IO;
System.IO.StreamWriter

► As Classes StreamReader e StreamWriter

As classes *StreamReader* e *StreamWriter*, ambas definidas no *namespace System.IO*, possibilitam a implementação da leitura e escrita de dados em ficheiros de acesso sequencial.

Classe	Objetivo
StreamReader	Leitura em ficheiro
StreamWriter	Escrita em ficheiro

Os streams são canais de comunicação entre a aplicação e o ficheiro.

▶ A classe File

▶ Principais métodos

Método	Descrição
File.CreateText(ficheiro)	Cria ficheiro para escrita de texto
File.OpenText(ficheiro)	Abre ficheiro para leitura
File.ReadAllText(ficheiro)	Lê o conteúdo do ficheiro para uma única string
File.AppendText(ficheiro)	Abre ficheiro para acrescentar texto
File.Exists(ficheiro)	Verifica se ficheiro existe (devolve verdadeiro ou falso)
File.Delete(ficheiro)	Remove ficheiro
File.Move(ficheiro, destino)	Move ficheiro
File.Copy(ficheiro, destino, overwrite)	Copia ficheiro de uma origem para um destino

- ▶ File.CreateText(caminhoFicheiro)
 - ▶ Cria ficheiro para escrita de texto

```
// Definição do caminho e nome do ficheiro para escrita
string meuFicheiro = @"c:\temp.txt";
//criação de ficheiro (se já existir remove-o e cria um novo)
StreamWriter sw = File.CreateText(meuFicheiro);
// Escrita no canal obtido da linha anterior
sw.WriteLine("Olá");
sw.WriteLine("Mundo!");
// Fecho do canal
sw.Close();
```



- ▶ File.OpenText(caminhoFicheiro)
 - ▶ Abre ficheiro para leitura de texto



```
C:\Windows\system32\cmd.exe
Olá
Mundo!
Prima qualquer tecla para continuar . . .
```

```
// Definição do caminho e nome do ficheiro para escrita
string meuFicheiro = @"c:\temp.txt";
//Verificação de existência de ficheiro
If(File.Exists(meuFicheiro))
{
    //Abertura de ficheiro para leitura
    StreamReader sr = File.OpenText(meuFicheiro);
    String linha = " ";
    // Leitura do ficheiro até obtenção de nulo
    while(linha!=null)
    {
        linha = sr.ReadLine();
        Console.WriteLine(linha);
    }
    sr.Close();
}
Else
{
    Console.WriteLine("Ficheiro inexistente!");
}
```

- ▶ `File.ReadAllText(caminhoFicheiro)`
 - ▶ Lê o conteúdo de um ficheiro para uma string

```
// Definição do caminho e nome do ficheiro para escrita
string meuFicheiro = @"c:\temp.txt";
//Verificação de existência de ficheiro
If(File.Exists(meuFicheiro))
{
    //Abertura de ficheiro para leitura
    String texto = File.ReadAllText(meuFicheiro);
    Console.WriteLine("O texto do ficheiro é \n{0}", texto);
}
```

temp.txt

Olá
Mundo!



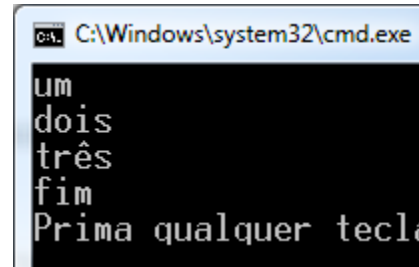
Usando o método `ReadAllLines` devolve um array de strings onde cada string é uma linha do ficheiro!

```
C:\Windows\system32\cmd.exe
O texto do ficheiro é
Olá
Mundo!
Prima qualquer tecla para continuar . . .
```

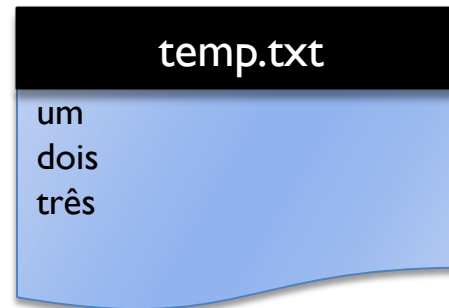
► File.AppendText(caminhoFicheiro)

► Abre ficheiro para acrescentar texto no fim do ficheiro

```
string meuFicheiro = @"c:\temp.txt";  
StreamWriter sw;  
if(File.Exists(meuFicheiro))  
// Se ficheiro existe abre-o para acrescentar  
// dados no fim  
    sw = File.AppendText(meuFicheiro);  
else  
// Se ficheiro não existe cria-o  
    sw = File.CreateText(meuFicheiro);  
string linha;  
// Lê dados da consola e escreve-os no  
// ficheiro até que seja escrito "fim"  
linha = Console.ReadLine();  
while(linha!="fim")  
{  
    sw.WriteLine(linha);  
    linha = Console.ReadLine();  
}  
sw.Close();
```

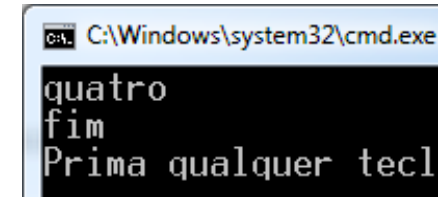


```
C:\Windows\system32\cmd.exe  
um  
dois  
três  
fim  
Prima qualquer tecla
```

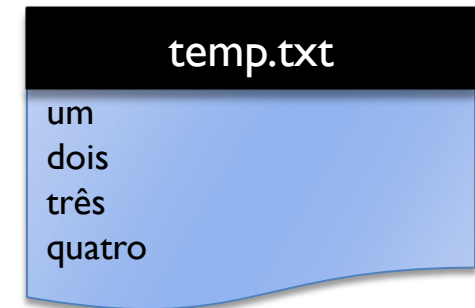


temp.txt

```
um  
dois  
três
```



```
C:\Windows\system32\cmd.exe  
quatro  
fim  
Prima qualquer tecla
```



temp.txt

```
um  
dois  
três  
quatro
```

- ▶ File.Delete(caminhoFicheiro)
 - ▶ Remove ficheiro

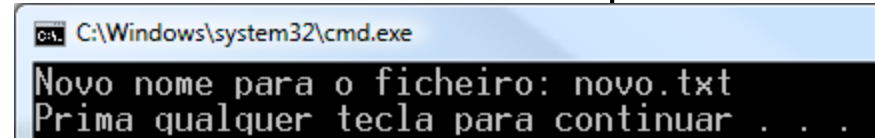
```
// Definição do caminho e nome do ficheiro para escrita
string meuFicheiro = @"c:\temp.txt";
// Se o ficheiro existe remove-o, confirma a remoção e
// escreve na consola a confirmação
if(File.Exists(meuFicheiro))
{
    File.Delete(meuFicheiro);
    if(File.Exists(meuFicheiro) == false)
        Console.WriteLine("Ficheiro removido com sucesso!");
}
else
    Console.WriteLine("Ficheiro inexistente!");
```



- ▶ `File.Move(ficheiroAntigo, ficheiroNovo)`
 - ▶ Move ficheiro para nova localização com hipótese de renomear

// Renomear ficheiro

```
string pastaMeusDocumentos =  
    Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);  
string meuFicheiro = pastaMeusDocumentos + @"\temp.txt";  
if (File.Exists(meuFicheiro))  
{  
    Console.WriteLine("Novo nome para o ficheiro:");  
    String novoFicheiro = Console.ReadLine();  
    if(novoFicheiro != String.Empty)  
        File.Move(meuFicheiro, pastaMeusDocumentos + @"\" + novoFicheiro);  
}
```



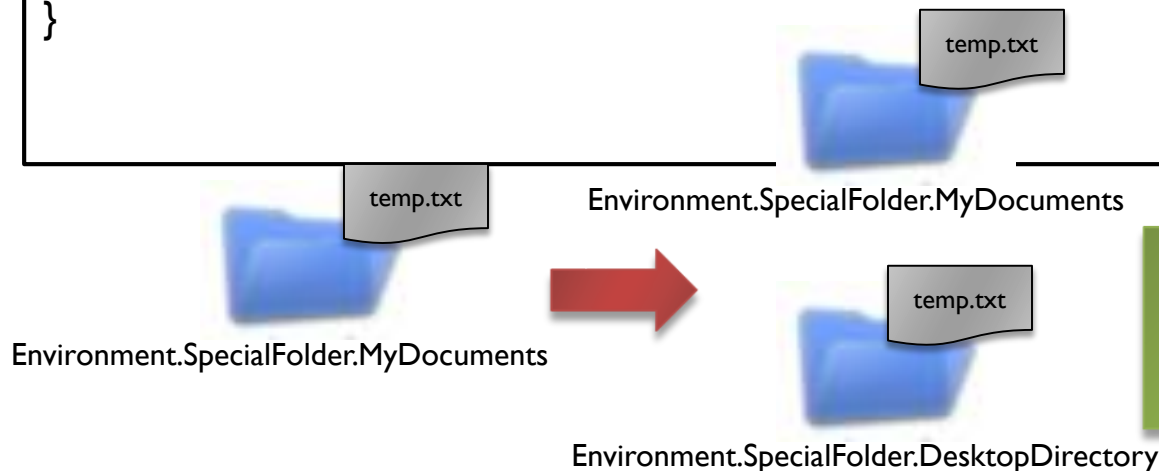
C:\Windows\system32\cmd.exe
Novo nome para o ficheiro: novo.txt
Prima qualquer tecla para continuar . . .



- ▶ `File.Copy(ficheiroAntigo, ficheiroNovo, sobreposição)`
 - ▶ Copia ficheiro para nova localização com hipótese de sobreposição no caso de ficheiro com igual nome no destino

// Mover ficheiro da pasta dos meus documentos para o desktop

```
string meuFicheiro = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) +  
    @"\temp.txt";  
string novoFicheiro = Environment.GetFolderPath(Environment.SpecialFolder.DesktopDirectory) +  
    @"\temp.txt";  
if (File.Exists(meuFicheiro)) {  
    File.Copy(meuFicheiro, novoFicheiro, true);  
    Console.WriteLine("Ficheiro copiado dos meus documentos para o desktop!");  
}
```



Usar o código
`AppDomain.CurrentDomain.BaseDirectory`
para obter a pasta do executável da
aplicação!

UM EXEMPLO...

```
static void Main(string[] args)
{
    // -----
    //          MENU
    //-----

    Console.Title = "Ficha 8 - Ficheiros";
    char opcao=' ';
    while (opcao != '0')
    {
        Console.BackgroundColor = ConsoleColor.DarkBlue;
        Console.Clear();
        Console.WriteLine(" -----");
        Console.WriteLine("          MENU");
        Console.WriteLine(" -----");
        Console.WriteLine("1 - Criar Registo de Aluno");
        Console.WriteLine("2 - Consultar lista geral de alunos");
        Console.WriteLine("3 - Consultar alunos por turma");
        Console.WriteLine("0 - Sair");
        Console.Write("          Opção: ");
        opcao = Convert.ToChar(Console.ReadLine());
        switch (opcao)
        {
            case '1' : criar_registo(); // Cria registo de aluno no ficheiro de texto
                break;
            case '2': consulta1(); // Consulta geral de alunos
                break;
            case '3': consulta2(); // consulta por turma
                break;
            case '0' : break;
            default: Console.WriteLine("Opção Inválida");
                Console.ReadLine();
                break;

        } // fim do switch

    } // fim do while

} /// fim do MAIN
```

UM EXEMPLO...

```
static void criar_registo()
{

    // Imprime nomes dos atributos a ler
    Console.Clear();
    Console.SetCursorPosition(10, 4);
    Console.WriteLine("Número :");
    Console.SetCursorPosition(10, 5);
    Console.WriteLine("Nome  :");
    Console.SetCursorPosition(10, 6);
    Console.WriteLine("Turma  :");

    // faz a leitura do numero, nome e turma
    Console.SetCursorPosition(22, 4);
    int numero = int.Parse(Console.ReadLine());
    Console.SetCursorPosition(22, 5);
    string nome= Console.ReadLine();
    Console.SetCursorPosition(22, 6);
    string turma = Console.ReadLine();

    // grava dados em ficheiro
    string ficheiro = @"alunos.txt";
    StreamWriter sw;
    if (File.Exists(ficheiro) == true)
    {
        sw = File.AppendText(ficheiro);
    }
    else
    {
        sw = File.CreateText(ficheiro);
    }
    string linha = numero.ToString() + ";" + nome + ";" + turma;
    sw.WriteLine(linha);
    sw.Close();

}
```


UM EXEMPLO...

```
static void consulta()
{
    Console.Clear();
    int lin = 5;
    string ficheiro = @"alunos.txt";
    StreamReader sr;
    // verifica se o ficheiro existe
    if (File.Exists(ficheiro) == true)
    {
        sr = File.OpenText(ficheiro); // Abre ficheiro para leitura
        string linha = "";
        while ((linha= sr.ReadLine()) != null) // Lê ficheiro até ao final
        {
            int pos = linha.IndexOf(";");
            string num = linha.Substring(0, pos); // Obtem o 1º campo de cada linha
            // Refaz a variavel linha: a partir do 1º ";" até ao final da linha
            linha = linha.Substring(pos+1, linha.Length-pos-1);
            // procura o ";" e retira o 2º campo
            pos = linha.IndexOf(";");
            string nome = linha.Substring(0, pos);
            string turma = linha.Substring(pos+1, linha.Length -pos-1);
            // Imprime os dados
            Console.SetCursorPosition(12, lin);
            Console.WriteLine(num);
            Console.SetCursorPosition(17, lin);
            Console.WriteLine(nome);
            Console.SetCursorPosition(40, lin);
            Console.WriteLine(turma);
            lin++;
        }
        sr.Close(); // fecha o ficheiro
    }
    else
    {
        Console.WriteLine("O ficheiro não existe");
    }
    Console.ReadLine();
}
```

- ▶ A classe **Directory**
- ▶ Principais métodos

Método	Descrição
Directory.CreateDirectory(pasta)	Cria pasta
Directory.Delete(pasta)	Remove pasta
Directory.Move (pastaOriginal, pastaDestino)	Move pasta de uma origem para um destino
Directory.GetFiles (pasta)	Obtém nomes dos ficheiros de uma pasta (array de strings)
Directory.Exists(pasta)	Verifica a existência de uma pasta (devolve verdadeiro ou falso)

- ▶ A Classe Directory
 - ▶ Directory.Exists e Directory.Delete

```
static void Main(string[] args)
{
    if(Directory.Exists("testdir"))
    {
        Directory.Delete("testdir");
        if(Directory.Exists("testdir") == false)
            Console.WriteLine("Pasta eliminada...");
    }
    else
        Console.WriteLine("A pasta testedir não existe!");

    Console.ReadLine();
}
```

- ▶ A Classe Directory
 - ▶ Directory.Move

```
static void Main(string[] args)
{
    if(Directory.Exists("testedir"))
    {
        Console.WriteLine("Novo nome da pasta:");
        string newDirName = Console.ReadLine();
        if(newDirName != String.Empty)
        {
            Directory.Move("testedir", newDirName);
            if(Directory.Exists(newDirName))
            {
                Console.WriteLine("O nome da pasta foi alterado para" + newDirName);
                Console.ReadKey();
            }
        }
    }
}
```

- ▶ A Classe Directory
 - ▶ **Directory.CreateDirectory**

```
static void Main(string[] args)
{
    Console.WriteLine("Nome da pasta:");
    string newDirName = Console.ReadLine();
    if(newDirName != String.Empty)
    {
        Directory.CreateDirectory(newDirName);
        if(Directory.Exists(newDirName))
        {
            Console.WriteLine("A pasta foi criada!");
            Console.ReadKey();
        }
    }
}
```

► A Classe Directory

► Exemplo...

```
static void Main(string[] args)
{
    if (Directory.Exists("c:\\projecto"))
    {
        Console.WriteLine("Pasta projecto já existe");
    }
    else
    {
        // cria pasta projecto
        Directory.CreateDirectory("c:\\projecto");
        Console.WriteLine("Pasta projecto criada com sucesso ! ");
        //criar sub-pasta programa
        Directory.CreateDirectory("c:\\projecto\\programa");
        Console.WriteLine("Pasta programa criada com sucesso ! ");
        //mover pasta programa para c:\
        Directory.Move("c:\\projecto\\programa", "c:\\programa");
        Console.WriteLine("Pasta programa foi movida para c:\\ ");
        //eliminar pasta programa
        Directory.Delete("c:\\programa");
        Console.WriteLine("pasta programa removida ");
    }
    Console.ReadLine();
}
```