

Práce se soubory (System.IO)

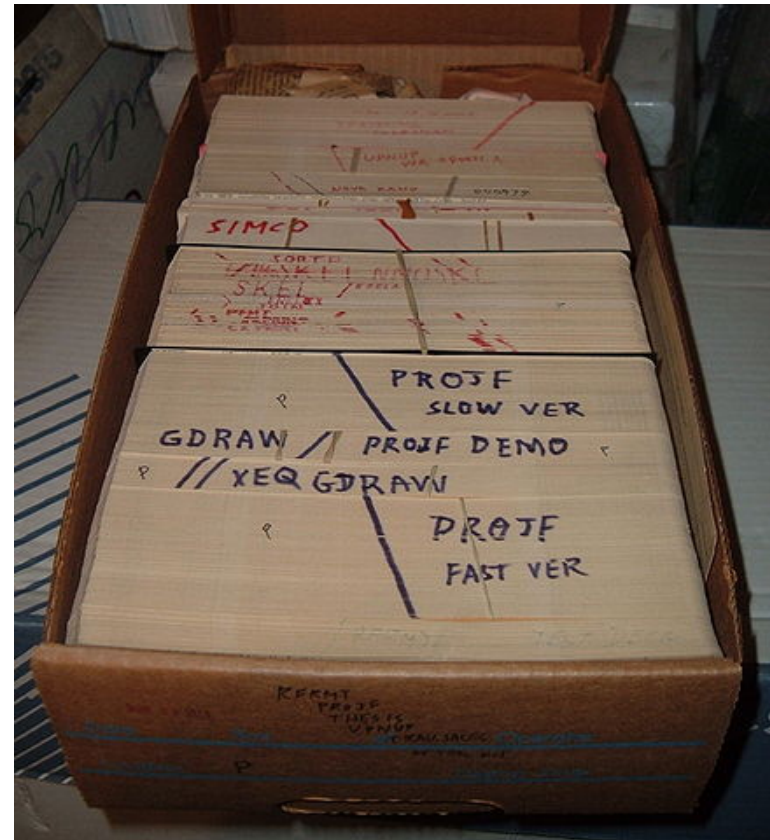
Class Directory

Class Path

Class File

Class StreamReader

Class StreamWriter



File (Soubor)

File je kolekce záznamů (nebo řádků)

Záznam je soubor dat, která spolu souvisejí

Druh souboru se charakterizuje příponou (.txg, .png)

FileName (název souboru) = Directory + Name

Directory (Adresář)

C:\Zamestnanci\zamestnanci.dat

Jan
Pospíšil
550225/0506
25. 2. 1955

Petra
Střítecká
875414/0201
14. 4. 1987

Zbyněk
Novák
611212/0802
12. 12. 1961

Pavel
Schuster
630506/0701
6. 5. 1963

...

2 druhy souborů

Soubor se sekvenčním přístupem (text)

C:\Zamestnanci\zamestnanci.csv

```
Jan,Pospíšil,550225/0506,25. 2. 1955  
Petra,Střítecká,875414/0201, 14. 4. 1987  
Zbyněk,Novák,611212/0802,12. 12. 1961  
Pavel,Schuster,630506/0701,6. 5. 1963  
...
```

Soubor s náhodným přístupem (záznamy)

C:\Zamestnanci\zamestnanci.dat

Jan Pospíšil 550225/0506 25. 2. 1955	Petra Střítecká 875414/0201 14. 4. 1987	Zbyněk Novák 611212/0802 12. 12. 1961	Pavel Schuster 630506/0701 6. 5. 1963	...
---	--	--	--	-----

Základní operace nad soubory

Vytvoření (Create)

Změna přístupových práv a atributů (Change)

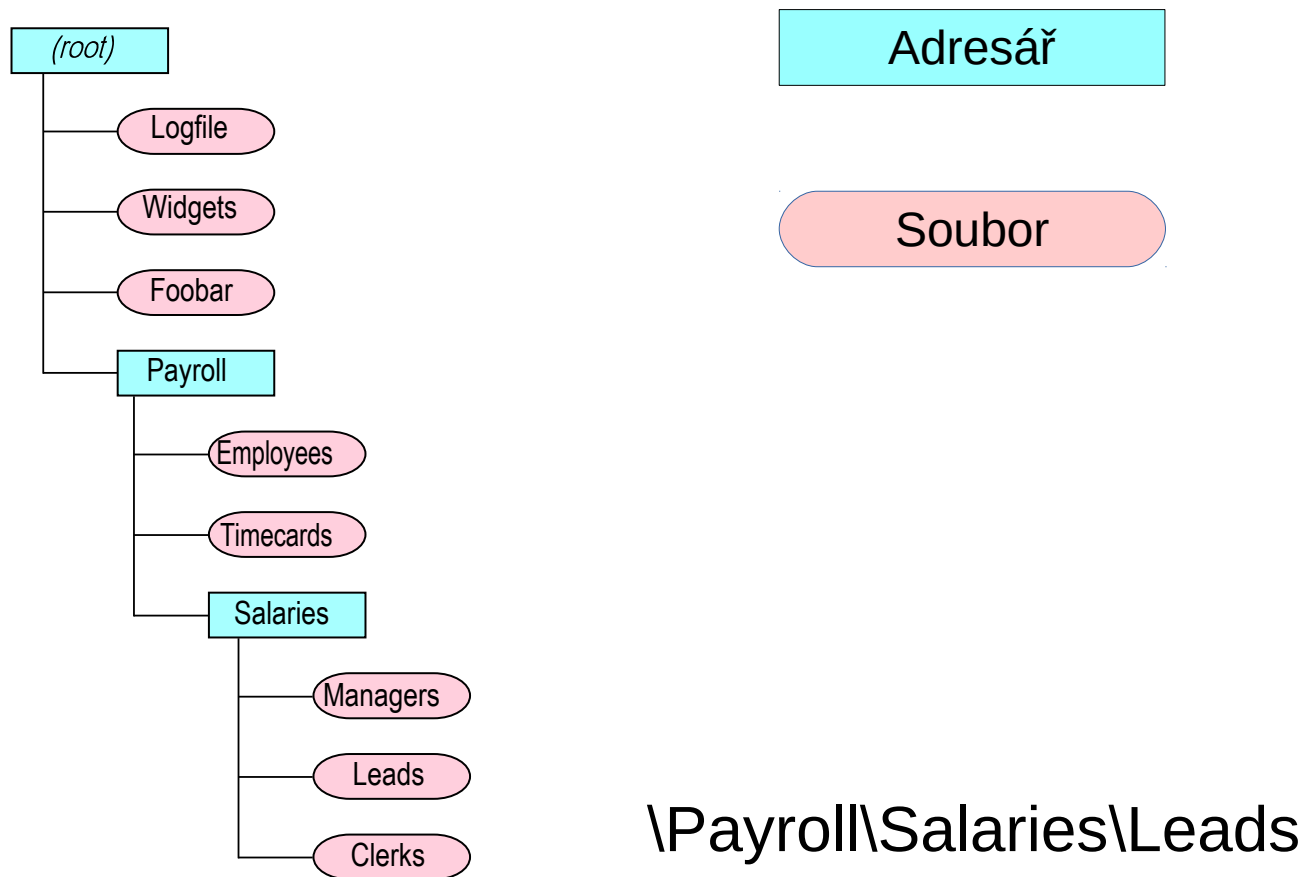
Otevření (Open)

Čtení (Read)

Zápis (Write)

Uzavření (Close)

Hierarchie souborů



1. Sekvenční přístup - třídy

- `StreamReader`
Načítání po znacích nebo po řádcích
- `StreamWriter`
Zapisování po znacích nebo řádcích
- `File`, `Directory`, `Path`
Zapisování nebo čtení celého stringu „na jeden zátah“
Operace se souborovým systémem

File a Stream

File (soubor) je uspořádaná a pojmenovaná kolekce bytů, která je uložaná na disku.

Stream (proud) je sekvence bytů, ze které můžeme číst nebo do ní zapisovat, ale která může používat různá úložiště (File, paměť, síť, apod.)

Class StreamReader

`StreamReader(string NazevSouboru)`

Inicializuje novou instanci třídy StreamReader.

`int Read()`

Vrací další znak ze StreamReader nebo -1, pokud nejsou další znaky.

`string ReadLine()`

Načte řádek ze StreamReader a vrátí jej jako string.

`int Peek()`

Přečte další znak beze změny stavu StreamReader.

Vrací další znak, který bude načten nebo -1, pokud není žádný další znak.

Class StreamWriter

`StreamWriter(string cesta)`

Inicializuje novou instanci třídy StreamWriter. Pokud soubor existuje, bude přepsán.

`StreamWriter(string cesta, bool append)`

Inicializuje novou instanci třídy StreamWriter. Může k souboru přidávat (`append = true`) nebo ho přepisovat.

`void Write(char c)`

Zapíše znak do StreamWriter.

`void WriteLine(string s)`

Zapíše string a zalomení řádku do StreamWriter.

Class StreamWriter

`Close()`

Zavře aktuální StreamReader nebo StreamWriter a uvolní všechny systémové zdroje přiřazené k StreamReader/StreamWriter

Class Stream*

using

Usnadňuje práci s objekty, které používají systémové zdroje. Pokud je nutné jejich uvolnění v případě vyvolání výjimky, použití příkazu `using` zaručí, že se zavolá odpovídající metoda `Dispose`. Je obdobou `try - catch - finally`.

```
using (var fsRead = new StreamReader (pathToSlovo))  
{  
    ...  
}  
// fsRead.Dispose není nutné volat, ani když dojde  
// k výjimce při čtení ze StreamReader
```

Class File

```
bool Exists(string cesta)
```

Vrací true, pokud soubor na cestě existuje

Class File

`string[] ReadAllLines(string cesta)`

Vrací pole stringů naplněné řádky přečteného souboru

`string ReadAllText(string cesta)`

Otevře soubor, celý ho přečte, vrátí ho jako string a uzavře soubor

```
/////
```

```
if (Exists(cesta))
```

```
    // string[] allLines = File.ReadAllLines(cesta)
```

```
    string allText = File.ReadAllText(cesta);
```

Class File

```
void WriteAllLines(string cesta,  
    string[] obsah)
```

Vytvoří (přepíše) soubor, zapíše **pole stringů obsah** do souboru a zavře soubor

```
void WriteAllText(string cesta,  
    string obsah)
```

Vytvoří (přepíše) soubor, zapíše **string obsah** do souboru a zavře soubor

```
File.WriteAllText(mydocpath + @"\WriteFile.txt",  
    text);
```

Class File

```
void Delete (string pathAndName)
```

Vymaže soubor na cestě, pokud existuje.

```
File.Delete("c:\\archives\\2008\\tmp1.txt");
```

Class Directory

```
string GetCurrentDirectory()
```

Vrací název pracovního adresáře

```
string path = Directory.GetCurrentDirectory();
```

```
void SetCurrentDirectory(string cesta)
```

Nastaví nový pracovní adresář, po ukončení programu se vrátí do původního

```
DirectoryInfo CreateDirectory(string path)
```

Vytvoří všechny adresáře na cestě, pokud už neexistují.

```
DirectoryInfo di = Directory.CreateDirectory(path);
```

```
// DirectoryInfo : Name, Root, Parent, CreationTime,
```

```
// LastAcceSTime, ...
```


Class Path

`Combine(string dir, string name)`

Ze dvou stringů vytvoří cestu k souboru

```
string path1 = "c:\\temp";  
string path2 = "subdir\\file.txt";  
string s = Path.Combine(path1, path2)
```

```
//c:\temp\subdir\file.txt
```

Class Path

`GetDirectoryName(string path)`

Vrací název adresáře (bez názvu souboru)

```
string filePath = "C:\MyDir\MySubDir\myfile.ext";  
directoryName = Path.GetDirectoryName(filePath);
```

```
//directoryName="C:\MyDir\MySubDir"
```

Class Path

`string GetTempPath()`

Vrací cestu k dočasným souborům

```
Console.WriteLine(Path.GetTempPath())  
// C:\Users\UserName\AppData\Local\Temp\
```

`string GetTempFileName()`

Vytváří dočasný soubor na disku a vrací úplnou cestu k tomuto souboru.

```
Console.WriteLine(Path.GetTempPath())  
// C:\Users\UserName\AppData\  
// Local\Temp\tmp70be1439.tmp
```

2. Náhodný přístup

- **FileStream**
Pro zapisování, čtení, otevírání a zavírání souborů na souborovém systému, umožňuje čtení a zápis po bytech (nebo polích bytů)
- **BinaryWriter**
Zapisuje primitivní data (int, long, float, double, string, ...) v binární podobě do streamu
- **BinaryReader**
Načítá primitivní data v binární podobě ze streamu

Class FileStream

`FileStream (string cesta,
 FileMode mod)`

cesta = cesta k souboru, který bude obsažený ve FileStream

mod = enum způsob otevření souboru

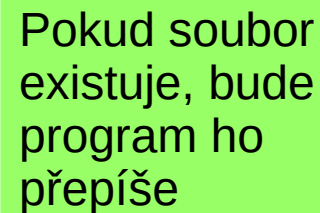
otevívá soubor pro čtení a zápis, pro ostatní procesy jen čtení

Append	Otevře soubor a přejde na konec nebo vytvoří nový soubor
Create	Vytvoří nový nebo přepíše existující soubor
CreateNew	Vytvoří nový soubor, ale vyhodí výjimku, pokud již existuje
Open	Otevře soubor, pokud existuje, jinak vyhodí výjimku
OpenOrCreate	Otevře soubor, pokud existuje, nebo vytvoří nový
Truncate	Otevře existující soubor a nastaví jeho velikost na 0

Class FileStream

```
const string fileName = "Test.dat";
```

```
FileStream fs =  
    new FileStream(fileName, FileMode.Create);
```



Pokud soubor
existuje, bude
program ho
přepíše

Ale to zajímavější az po metodě Seek ...

Náhodný přístup = Seek()

Umožňuje pohyb po streamu, nastavuje pozici ve streamu na konkrétní hodnotu:

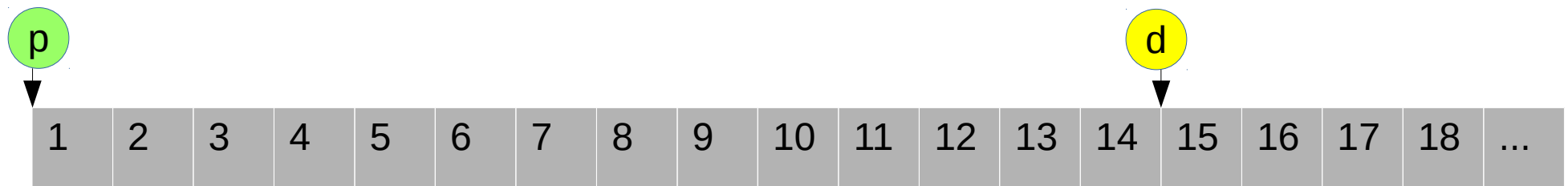
```
long Seek(long dalka, SeekOrigin pocatek)  
//  
// enum SeekOrigin { Begin, Current, End }  
//
```

Vrací novou pozici ve streamu

```
Seek(-2, SeekOrigin.End);
```



```
Seek(14, SeekOrigin.Begin);
```



Class FileStream

```
const string fileName = "Test.dat";
```

```
FileStream fs = new  
    FileStream(fileName, FileMode.OpenOrCreate);
```

Bez cesty se vytvoří v
domácím adresáři programu

```
fs.Seek (0, SeekOrigin.End);
```

```
fs.WriteByte(1);  
fs.WriteByte(2);  
fs.WriteByte(3);
```

Přejde na konec
souboru.

Pokud soubor
existuje,
program jej
otevře, jinak
vytvoří nový.

```
fs.Seek (0, SeekOrigin.Begin);
```

Přejde na začátek
souboru.

```
for (int i = 0; i < fs.Length; i++)  
    Console.WriteLine (fs.ReadByte ());
```


Class BinaryWriter

`BinaryWriter (FileStream vystup)`

vystup = obecně Stream, nejčastěji FileStream, do kterého se budou zapisovat binární data

Vytvoří FileStream pro zápis a potom ho použije pro vytvoření BinaryWriter

```
using (FileStream fs = new  
    FileStream ("BigIntArr.dat", FileMode.Create)){  
    using (BinaryWriter bw = new BinaryWriter (fs)) {  
        int SIZE = 2000;  
        // tady se bude zapisovat Int32 a potom string  
        bw.Write (SIZE);  
        bw.Write ("Záložní úložiště pro naše pole.");  
    }  
}
```

using zajistí, že se FileStream i BinaryWriter uvolní z paměti, až jej program přestane používat. Nahrazuje try/catch konstrukci pro oba objekty

Class BinaryWriter

`Write (typ)`

overload pro všechny běžné primitivní typy:

```
Write ( char c )  
Write ( char[] cArr )  
Write ( Int16 i )  
Write ( Int32 i )  
Write ( Int64 i )  
Write ( decimal d )  
Write ( string s )  
Write ( UInt16 )  
....
```

Vlastnost `BaseStream` odkazuje na stream, ze kterého byl vytvořen `BinaryWriter`

Class BinaryReader

`BinaryReader (FileStream vstup)`

vstup = obecně Stream, nejčastěji FileStream, ze kterého se budou číst binární data

Vytvoří FileStream pro čtení
potom ho použije pro vytvoření
BinaryReader

```
using (FileStream fs = new  
    FileStream ("BigIntArr.dat", FileMode.Open)) {  
    using (BinaryReader br = new BinaryReader (fs)) {  
  
        // tedy se bude číst Int32 a potom string  
        // první int je velikost pole  
        int sizeOfArray = br.ReadInt32 ();  
        string s = br.ReadString ();  
  
    }  
}
```

string je uložený
tak, že nejdřív se
načte délka
stringu a
potom celý string

Class BinaryReader

```
char ReadChar()  
char[] ReadChars(int kolik)  
Int16 ReadInt16()  
Int32 ReadInt32()  
decimal ReadDecimal()  
string ReadString()  
  
...
```

Vlastnost `BaseStream` odkazuje na stream, ze kterého byl vytvořen `BinaryReader`

3. Serializace

- BinaryFormatter