

Jazyk C# 2

3. seminář

Jakub Večeřa

Univerzita Palackého v Olomouci

24.2.2025

Reakce na úkoly (1/3)

- Velikost náhodného pole
- Efektivita paralelizace
- Sdílené proměnné

Doplnění (1/3): Práce s vlákny

- Nevýhody přímého použití vláken
 - ▶ Nevracejí výsledek přímo - použití sdílené proměnné / přes objekt
 - ▶ Neposkytují pokročilé řízení výpočtu
 - ▶ Může mít jenom jeden proces
- `Task` je abstrakce nad vlákny
 - ▶ Interně používá `TaskScheduler`, který využívá `ThreadPool` = velmi optimalizovaný systém přidělování zdrojů
 - ▶ Řízení výpočtu: `CancellationToken`
 - ▶ Návratová hodnota: `Task.Result`

Doplnění (2/3): Scheduler a ThreadPool

- Nejdražší operací u vláken je jejich vytvoření
- Implementace:
 - ▶ ThreadSafe Queue
 - ▶ Workers Pool
- Neřeší problémy kritické sekce
- Lze manipulovat s prioritou úkonů i vláken
- Často se stává, že proces tráví dlouhý čas čekáním, blokují operace
 - ▶ Stále hrozí thread starvation
- Event based řízení vláken

Doplnění (3/3): Task metody a property

- `Task.IsCompleted`, `Task.IsCanceled` - dotaz na stav tasku
- `Task.Status` - `TaskStatus` enum: `Created`, `Started`, `Canceled`,
- `Task.Wait` - ekvivalentní k `await`, nebo `Thread.Join()`
- `Task.Dispose` - uklidí veškeré zdroje tasku, uvolní vlákna

Třída DirectoryInfo (1/2)

```
1 DirectoryInfo di = new DirectoryInfo("E:\\1\\");
```

di	[E:\1\]	System.IO.DirectoryInfo
Attributes	Directory	System.IO.FileAttributes
CreationTime	{6. 3. 2017 10:40:00}	System.DateTime
CreationTimeUtc	{6. 3. 2017 9:40:00}	System.DateTime
DisplayPath	"E:\\1\\"	string
Exists	true	bool
Extension	""	string
FullName	"E:\\1\\"	string
FullPath	"E:\\1\\"	string
Identity	null	object
LastAccessTime	{6. 3. 2017 10:40:24}	System.DateTime
LastAccessTimeUtc	{6. 3. 2017 9:40:24}	System.DateTime
LastWriteTime	{6. 3. 2017 10:40:24}	System.DateTime
LastWriteTimeUtc	{6. 3. 2017 9:40:24}	System.DateTime
Name	"1"	string
OriginalPath	"E:\\1\\"	string
Parent	{}	System.IO.DirectoryInfo
Root	[E:\]	System.IO.DirectoryInfo
UnsafeGetFullName	"E:\\1\\"	string

- Pozor! – zadaná cesta v konstruktoru nemusí existovat

Třída DirectoryInfo (2/2)

- Metody pro výčet souborů a složek:
`EnumerateFiles()`, `EnumerateDirectories()`
- Vs. pro vrácení pole souborů/složek `GetFiles()`, `GetDirectories()`
- Enum `FileAttributes` – Archive, Compressed, Device, Directory, Encrypted, Hidden, IntegrityStream, Normal, NoScrubData, NotContentIndexed, Offline, ReadOnly, ReparsePoint, SparseFile, System, Temporary
- statická třída `Directory` vs. `DirectoryInfo`

Třída FileInfo





















```
1 FileInfo fi = new FileInfo("E:\\1\\test.txt");
```

- Atributy stejný význam jako u DirectoryInfo
- Metody pro vytváření, mazání, zápis (viz. dále)

Třída DriveInfo

- Statická metoda `GetDrives()` – vrací pole `DriveInfo[]` s informacemi od discích

```
1 DriveInfo diskE = new DriveInfo("E:\\");
```

 <code>drInf</code>	<code>{E:\}</code>		<code>System.IO.DriveInfo</code>
 <code>AvailableFreeSpace</code>	<code>5592346624</code>		<code>long</code>
 <code>DriveFormat</code>	<code>"NTFS"</code>	 	<code>string</code>
 <code>DriveType</code>	<code>Fixed</code>		<code>System.IO.DriveType</code>
 <code>IsReady</code>	<code>true</code>		<code>bool</code>
 <code>Name</code>	<code>"E:\\"</code>	 	<code>string</code>
  <code>RootDirectory</code>	<code>{E:\}</code>		<code>System.IO.DirectoryInfo</code>
 <code>TotalFreeSpace</code>	<code>5592346624</code>		<code>long</code>
 <code>TotalSize</code>	<code>64421359616</code>		<code>long</code>
 <code>VolumeLabel</code>	<code>"Data"</code>	 	<code>string</code>
 <code>_name</code>	<code>"E:\\"</code>	 	<code>string</code>

- Enum `DriveType`: `CDRom`, `Fixed`, `Network`, `NoRootDirectory`, `Ram`, `Removable`, `Unknown`

Čtení obsahu souboru

- Statická třída `File`
- Načtení **celého** obsahu do stringu

```
1 FileInfo f = new FileInfo(@"E:\1\test.txt");
2 if (f.Exists)
3 {
4     string str = File.ReadAllText(f.FullName);
5 }
```

- Načtení **celého** obsahu do pole řádků

```
1 FileInfo f = new FileInfo(@"E:\1\test.txt");
2 if (f.Exists)
3 {
4     string[] array = File.ReadAllLines(f.FullName);
5 }
```

Proudy (stream)

- Obecný mechanismus pro čtení/zápis dat "od někud někam"
 - ▶ Z venku do našeho programu → čtení z proudu
 - ▶ Z našeho programu „někam“ → zápis do proudu
- Abstrakce – stejný přístup pro různé zdroje a cíle např.
 - ▶ Síť
 - ▶ Soubor
 - ▶ Paměť
 - ▶ ...

Čtení souboru pomocí proudů

```
1 FileInfo f = new FileInfo(@"E:\1\test.txt");
2 if (f.Exists) {
3     StreamReader sr = null;
4     try {
5         sr = f.OpenText();
6         string s = "";
7         while ((s = sr.ReadLine()) != null) {
8             Console.WriteLine(s);
9         }
10    } catch (Exception e) {
11        Console.WriteLine("Chyba pri cteni souboru");
12    } finally {
13        if (sr!=null) {
14            sr.Close();
15        }
16    }
17 }
```

Alternativně

- Na konci deklarovaného bloku se objekt automaticky zahodí:

```
1 using (StreamReader sr = new StreamReader(@"E:\1\test.txt"))
2 {
3 // Kod pouzivajici StreamReader
4 }
```

- Také lze pouze:

```
1 using StreamReader sr = new StreamReader(@"E:\1\test.txt");
```

Zápis do souboru (1/4)

- Vytvoření souboru – vrací stream

```
1 File.Create(path);
```

- Zápis

```
1 StreamWriter sw = null;
2 try {
3     sw = new StreamWriter(File.Create(@"E:\1\test2.txt"));
4     for (int i = 0; i < 10; i++) {
5         sw.WriteLine($"Line {i}");
6     }
7 } catch (Exception e) {
8     Console.WriteLine("Zapis souboru selhal");
9 } finally {
10     if (sw!=null) {
11         sw.Close();
12     }
13 }
```

Zápis do souboru (2/4)

- Komprimovaný soubor – pouhé přidání `GZipStream`

```
1 StreamWriter sw = null;
2 try
3 {
4     GZipStream gzs = new GZipStream(File.Create(@"E:\1\test2.gzip"),
5         CompressionLevel.Optimal);
6     sw = new StreamWriter(gzs);
7     for (int i = 0; i < 10; i++)
8     {
9         sw.WriteLine($"Line {i}");
10    }
11 catch (Exception e) {
12    Console.WriteLine("Zapis souboru selhal");
13 } finally {
14     if (sw!=null) {
15         sw.Close();
16     }
17 }
```

Zápis do souboru (3/4) – binární vs. textový zápis

- Jak efektivně zapsat sekvenci jedniček a nul

- Co udělá:

```
1 sw.WriteLine($"1110001001000");
```

- Výsledkem je správně řetězec jedniček a nul, ale s velikostí 15B

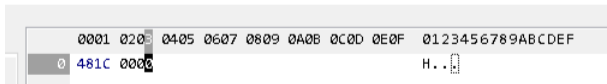
- Proč?

0001	0203	0405	0607	0809	0A0B	0C0D	0E0F	0123456789ABCDEF	
0	3131	3130	3030	3130	3031	3030	300D	0A	1110001001000.

Zápis do souboru (4/4) – binární

- Lze použít `BinaryWriter` <https://docs.microsoft.com/en-us/dotnet/api/system.io.binarywriter?view=netframework-4.7.2>
- Zapisuje se pak binárně, nikoliv textově
- Soubor pak nepůjde číst klasicky (textový editor zobrazí „rozsypaný čaj“)
- Je to ale efektivnější, ale režie je na vás (oddělovače, přesný formát, počítání bajtů)

```
1 BinaryWriter sw = null;
2 try {
3     sw = new BinaryWriter(File.Create(@"C:\cs\test2.txt"));
4     sw.Write(0b1110001001000);
5 }
```



- Vyzkoušejte třeba `sw.write(0.33f);` vs. `sw.write(0.33);`. Bude nějaký rozdíl?

Úkol (1/3)

- Vypište seznam disků s jejich jménem, velikostí a volným místem
- Uživatel zadá vstupní adresář; vypište seznam jeho adresářů a souborů s datem jejich vytvoření, velikostí(u souborů) a názvem
- Navrhněte třídu BinaryMatrix, která bude efektivně reprezentovat binární matici + konstruktor
- Naprogramujte metody WriteMatrix a ReadMatrix pro **efektivní** uložení a načtení matice ze souboru
- Zjistěte a vypište, kolik bajtů bylo potřeba pro zápis dané matice
 - ▶ (a zamyslete se nad tím)
- Vhodně ošetřete vstupy, výstupy, výjimky

Úkol (2/3)

- Náznak volání

```
1 > Diskova zarizeni v pocitaci:
2 Name           Type           Size (bytes)   Free Space
3 A:\            Removable
4 C:\            Fixed          321544777728   140893417472
5 D:\            CDROM
6 U:\            Network        8100172001280   7275337273344
7 V:\            Network        8100172001280   7275337273344
8 Z:\            Network        4055406592     43767193600
9
10 > zadejte adresar: C:\test[enter]
11 Directory of C:\test
12
13 03.12.2016 19:14 <DIR>      app
14 07.11.2007 07:00      17 734 test.txt
15 07.11.2007 07:00      45 123 test2.pdf
```

Úkol (3/3)

● Příklad volání

```
1 BinaryMatrix matrix = new BinaryMatrix(...)  
2 Console.WriteLine(matrix);  
3 matrix.set(2,1,0);  
4 Console.WriteLine(matrix);  
5 matrix.Writematrix(path);  
6  
7 BinaryMatrix readed = BinaryMatrix.ReadMatrix(path)
```

● Výstup

```
1 1 1 0  
2 0 1 0  
3 1 1 1 -> Zapsana matice o velikost xyz bajtu.  
4  
5 1 1 0  
6 0 1 0  
7 1 0 1 -> Nactena matice z disku
```