Haladó fejlesztési technikák

Folyamatok
Folyamatok közötti kommunikáció

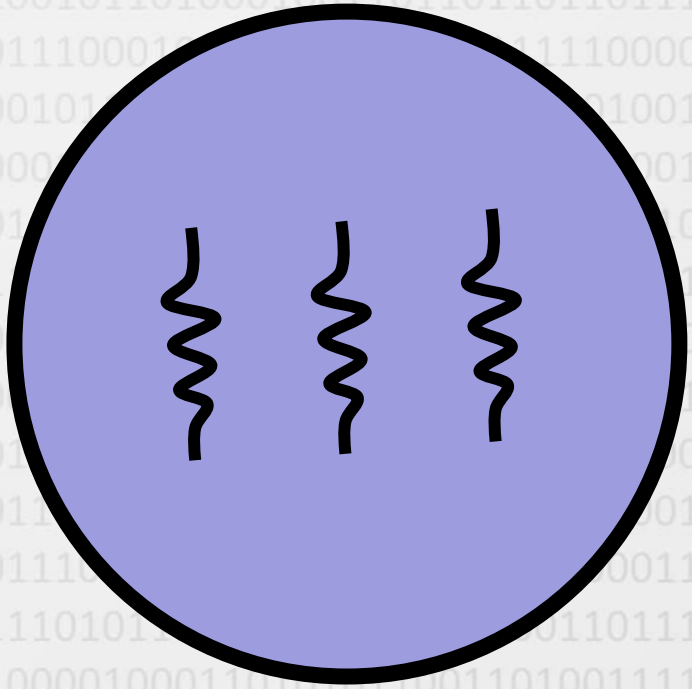
BEVEZETÉS

Folyamatok

- **Az operációs rendszer egyidejűleg több programot is végrehajthat**
  - Ehhez a programhoz erőforrásokat rendel, memóriaterületet allokál, prioritást határoz meg
  - A végrehajtás alatt álló programot folyamatnak nevezzük
- **Több folyamat *szimultán* futtatásához több processzor(mag) szükséges**
  - Ezt nevezzük párhuzamos feldolgozásnak
- **De lehetőség van arra is, hogy a különböző folyamatok időosztásos felbontásban férjenek hozzá az erőforrásokhoz**
  - Ezt nevezzük konkurenciának

Szálak

- **A folyamaton belüli egységnyi végrehajtási szekvenciát szálnak nevezzük**
  - Egy folyamaton belül több szál is lehet
  - A szálak a gazdafolyamatuk memóriaterületét használják
  - Külön CPU magot használhatnak



Párhuzamosság

- **Multiprocessing**
  - Több folyamat dolgozik ugyanazon probléma megoldásán
  - Az egyes folyamatok külön CPU-t használhatnak a számításaikhoz
  - Az egyes folyamatok különálló memóriaterületen dolgoznak, információ cseréjéhez kommunikálniuk kell
- **Multithreading**
  - Egy folyamaton belül több szál dolgozik a probléma megoldásán
  - A szálak külön CPU magokat használhatnak a számításaikhoz
  - A szálak közös memóriaterületen dolgoznak, adatok hozzáférésekor számíthat a hozzáférési sorrend

FOLYAMATOK

System.Diagnostics.Process

- A Process osztály példányosításával hozható létre folyamat
- De az osztályszintű Start() metódus is használható ilyen célra

```
Process p = new Process()
{
    StartInfo = new ProcessStartInfo("hello") {
        CreateNoWindow = true,
        UseShellExecute = false,
        RedirectStandardOutput = true
    }
};
p.Start();

Process.Start("cmd");
Process.Start("explorer");
Process.Start("chrome", "http://users.nik.uni-obuda.hu/prog3");
Process.Start("http://nik.uni-obuda.hu");
```

System.Diagnostics.Process osztály (kivonatos referencia)

Metódusok	
Start()	Folyamat indítása
CloseMainWindow()	Folyamat főablakának bezárása (GUI alkalmazásoknál)
Kill()	Folyamat leállítása
GetCurrentProcess()	Aktuális folyamatot reprezentáló objektum lekérése
GetProcesses()	Összes folyamat adatainak lekérése a helyi számítógépről
WaitForExit()	Várakozás az adott folyamat befejeződésére
Tulajdonságok	
StartInfo	A folyamathoz tartozó ProcessStartInfo példány
PriorityClass	A folyamat prioritása (fontossági szintje)
EnableRaisingEvents	A folyamat kiválthat-e eseményeket
HasExited	A folyamat kilépett-e
ExitCode, ExitTime	Kilépési kód, illetve a kilépés (vagy leállítás) időpontja
StandardInput, StandardOutput	Alapértelmezett be- és kimeneti csatorna (adatfolyam)
Események	
Exited	A folyamat kilépett (vagy leállították)

System.Diagnostics.ProcessStartInfo osztály (kivonatos referencia)

Tulajdonságok	
FileName	Fájlnév megadása az indítandó folyamathoz (program vagy programmal társított fájltípusba tartozó fájl neve)
Arguments, WorkingDirectory	Parancssori paraméterek és munkakönyvtár megadása az indítandó folyamathoz
Domain, UserName, Password	Folyamat indítása adott felhasználó nevében
RedirectStandardInput, RedirectStandardOutput	Alapértelmezett be- és kimeneti csatorna átirányítása
ErrorDialog	Hibaüzenet jelenjen-e meg, ha a folyamat indítása sikertelen
UseShellExecute	Operációs rendszerhéj programindító funkciójának használata folyamat indításához
Verb	A társított fájl megnyitásakor végrehajtandó művelet
WindowStyle	Kezdeti ablakméret megadása (normál, minimalizált vagy maximalizált méret)

Folyamatok listázása

- Process.GetProcesses()

```
static void Main(string[] args)
{
    foreach (var p in Process.GetProcesses().OrderBy(x => x.Id))
        Console.WriteLine($"#{p.Id}\t {p.ProcessName}");
}
```

```
#0       Idle
#4       System
#96      Registry
#104     RuntimeBroker
#220     tv_x64
#372     WUDFHost
#412     smss
#556     svchost
#564     csrss
#652     egui
#660     wininit
#668     csrss
```

Több példányban futó folyamat

- **Előfordulhat, hogy a programunk logikájában tiltani szeretnénk azt, hogy egyidejűleg több folyamat induljon ugyanabból a futtatható állományból**

Több példányban futó folyamat

```
if (Process.GetProcesses().Where(
        x => x.ProcessName == Process.GetCurrentProcess().ProcessName &&
        x.Id != Process.GetCurrentProcess().Id
    ).Count() > 0)
{
    Console.WriteLine("Ez a folyamat már fut!");
    return;
}

Console.WriteLine("Helló világ!");
Console.ReadLine();
```

```
Ez a folyamat már fut!
```

```
Ez      Ez a folyamat már fut!
```

```
Helló világ!
```

Folyamat kimenetének olvasása

- Hello.exe

```
static void Main(string[] args)
{
    if (args.Length > 0)
        Console.WriteLine("Hello " + args[0] + "!");
    else
        Console.WriteLine("Hello világ!");
}
```

```
c:\Users\kerteszg\source\repos\Process_IPC\03_Hello\bin\Debug>hello.exe
Hello világ!

c:\Users\kerteszg\source\repos\Process_IPC\03_Hello\bin\Debug>hello.exe Gábor
Hello Gábor!
```

Folyamat kimenetének olvasása

```
Process p = Process.Start(new ProcessStartInfo("hello") {
    CreateNoWindow = true,
    UseShellExecute = false,
    RedirectStandardOutput = true
});
p.WaitForExit();

Console.WriteLine(p.StandardOutput.ReadToEnd());
```

```
Hello világ!
```

Folyamat kimenetének olvasása

```
Process p = Process.Start(new ProcessStartInfo("hello", "Pistike") {
    CreateNoWindow = true,
    UseShellExecute = false,
    RedirectStandardOutput = true
});
p.WaitForExit();

Console.WriteLine(p.StandardOutput.ReadToEnd());
```

```
Hello Pistike!
```

Blokkolásmentes felépítés

- A p.WaitForExit() hívás addig blokkolja a végrehajtást, amíg p folyamat véget nem ér.
- Ez a viselkedés folyamatok esetén megkerülhető például események használatával

```
p.EnableRaisingEvents = true;
p.Exited += P_Exited;
```

Blokkolásmentes felépítés

```
foreach (var h in hosts)
{
    Process p = Process.Start(new ProcessStartInfo("ping", "-n 10 " + h)
    {
        CreateNoWindow = true,
        UseShellExecute = false,
        RedirectStandardOutput = true
    });
    p.EnableRaisingEvents = true;
    p.Exited += P_Exited;
}
Console.ReadLine();
//...

private static void P_Exited(object sender, EventArgs e)
{
    Console.WriteLine((sender as Process).StandardOutput.ReadToEnd());
}
```

Feladat

Egy gyűjteményben eltárolunk a világ különböző országaiban bejegyzett domaineket. A feladat az ezekhez tartozó útvonalak elemzése, tracert segítségével.

Állapítsuk meg minden címről, hogy hány állomáson (hop) keresztül érhetőek el a számítógépünkről!

a) **Az eredmények akkor jelenjenek meg, amikor a program minden vizsgálat feldolgozásával végzett!**
b) **Az eredmények azonnal jelenjen meg a konzolon, amint a háttérben futó folyamat végzett!**

FOLYAMATOK KÖZÖTTI KOMMUNIKÁCIÓ

Inter-process Communication

- **Folyamatok közötti üzenetváltás**
- **Többféle megközelítés létezik:**
  - Socket
  - Pipe
  - Message queue
  - Shared memory
  - Message passing
  - stb.

Pipe

- Amennyiben a host ugyanaz, a legegyszerűbb megközelítési mód a pipe (System.IO.Pipes)

```
//Szerver
var server = new
NamedPipeServerStream("EzEgyEgyediNev");
Console.WriteLine("Várakozás kliensre...");
server.WaitForConnection();
Console.WriteLine("Kliens csatlakozott!");
StreamReader reader = new StreamReader(server);
StreamWriter writer = new StreamWriter(server);
while (true)
{
    var line = reader.ReadLine();
    writer.WriteLine(string.Join("",
line.Reverse()));
    Console.WriteLine($"Azt az üzenetet kaptam
hogy {line}, azt feleltem hogy {string.Join("",
line.Reverse())}");
    writer.Flush();
}
```

```
//Kliens
var client = new NamedPipeClientStream("EzEgyEgyediNev");
Console.WriteLine("Csatlakozom a szerverhez...");
client.Connect();
Console.WriteLine("Csatlakozva!");
StreamReader reader = new StreamReader(client);
StreamWriter writer = new StreamWriter(client);
while (true)
{
    Console.WriteLine("Mit üzensz?");
    string input = Console.ReadLine();
    if (String.IsNullOrEmpty(input)) break;
    writer.WriteLine(input);
    writer.Flush();
    Console.WriteLine(reader.ReadLine());
}
```

NamedPipe demo

```
Várakozás a kliensre...
Kliens csatlakozott!
Azt az üzenetet kaptam hogy
asdasdasd, azt feleltem hogy
dsadsadsa
Azt az üzenetet kaptam hogy
Indul a görög aludni, azt
feleltem hogy indula görög a
ludnI
```

```
Csatlakozom a szerverhez...
Csatlakozva!
Mit üzensz?
asdasdasd
dsadsadsa
Mit üzensz?
Indul a görög aludni
indula görög a ludnI
Mit üzensz?
```

MPI

- A Message Passing Interface egy szabvány, amelyet implementál többek között az OpenMPI nevű keretrendszer is
- Segítségével megoldható a folyamatok közötti kommunikáció, akár hálózaton keresztül is

```
using (new MPI.Environment(ref args))
{
        Console.WriteLine(
                "Hello, from process number " +
                MPI.Communicator.world.Rank.ToString() +
                " of " +
                MPI.Communicator.world.Size.ToString()
        );
}
```

```
C:\tmp\MPI_Hello\bin\Debug>mpiexec -n 8 MPI_Hello.exe
Hello, from process number 1 of 8
Hello, from process number 0 of 8
Hello, from process number 4 of 8
Hello, from process number 3 of 8
Hello, from process number 6 of 8
Hello, from process number 2 of 8
Hello, from process number 5 of 8
Hello, from process number 7 of 8
```

Források

- Szabó-Resch Miklós Zsolt és Cseri Orsolya Eszter Haladó Programozás előadásfóliái
- Miklós Árpád prezentációi
- MSDN
- Jeffrey Richter: CLR via C#