C# Szálkezelés

Tóth Zsolt

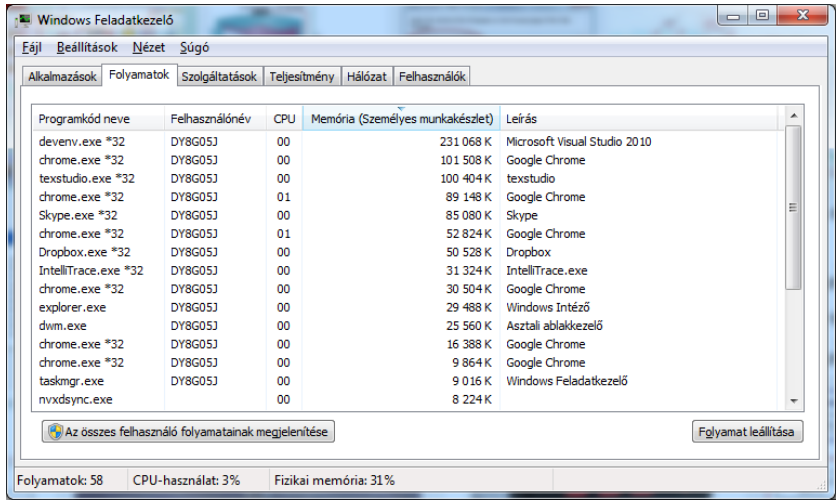
Miskolci Egyetem

2013

Tartalomjegyzék

Process

- Futó program egy példánya
- Egy program többször is futhat $\rightarrow$ több process
- Ütemezés egysége
- Kontextus
- Tulajdonságai
  - Azonosító (pid)
  - Erőforrás adatok
    - ★ CPU idő
    - ★ Memória
    - ★ Nyitott adatfolyamok



Thread

- Végrehajtás szála
- Process kontextusban futnak
- Klasszikus process egy szálon fut.
- Egy process több szál
- Párhuzamosság
- Egyszer implementáció

Előnyök

- Közös kontextus
- Kevesebb process
- Nem kell IPC
- Alacsony menedzselési költség

Párhuzamosítás

The whole is more than the sum of its parts.

- Korszerű CPU
- Teljesítmény ↑
- Feladat felbontása
- Egyszerű részfeladatok
- Egy folyamat egy feladat
- Menedzselési költség

Használata

- GUI
  - Megjelenítés
  - Feldolgozás
- Nagy adatmennyiség
  - pl.: rendezés
  - $O(n^2)$
  - Nagy n esetén lassú
  - Részek rendezése
  - Összefésülés
- Párhuzamos feldolgozás

Tartalomjegyzék

Thread

- System.Threading
- Szál reprezentálása
- Szál kezelése
- Konstruktor
  - ThreadStart
  - ParameterizedThreadStart

  Properties
  - CurrentThread
  - IsAlive
  - ThreadState

Műveletek

- Start
- Abort
- Join
- Suspend
- Resutme
- Sleep
- Interupt

ThreadStart

```
public delegate void ThreadStart()
```

- System.Threading.ThreadStart
- Delegátum
- Függvényt azonosít
- Thread konstruktor paraméter
- Nagy rugalmasság
- Implementációt futtatja a Thread
- Nincs paraméter

Thread

```
public delegate void ParameterizedThreadStart(
Object obj)
```

- System.Threading.ParameterizedThreadStart
- Delegátum
- Függvényt azonosít
- Thread konstruktor paraméter
- Object paraméter $\rightarrow$ tetszőleges paraméter
- Castolás
  - as
  - is

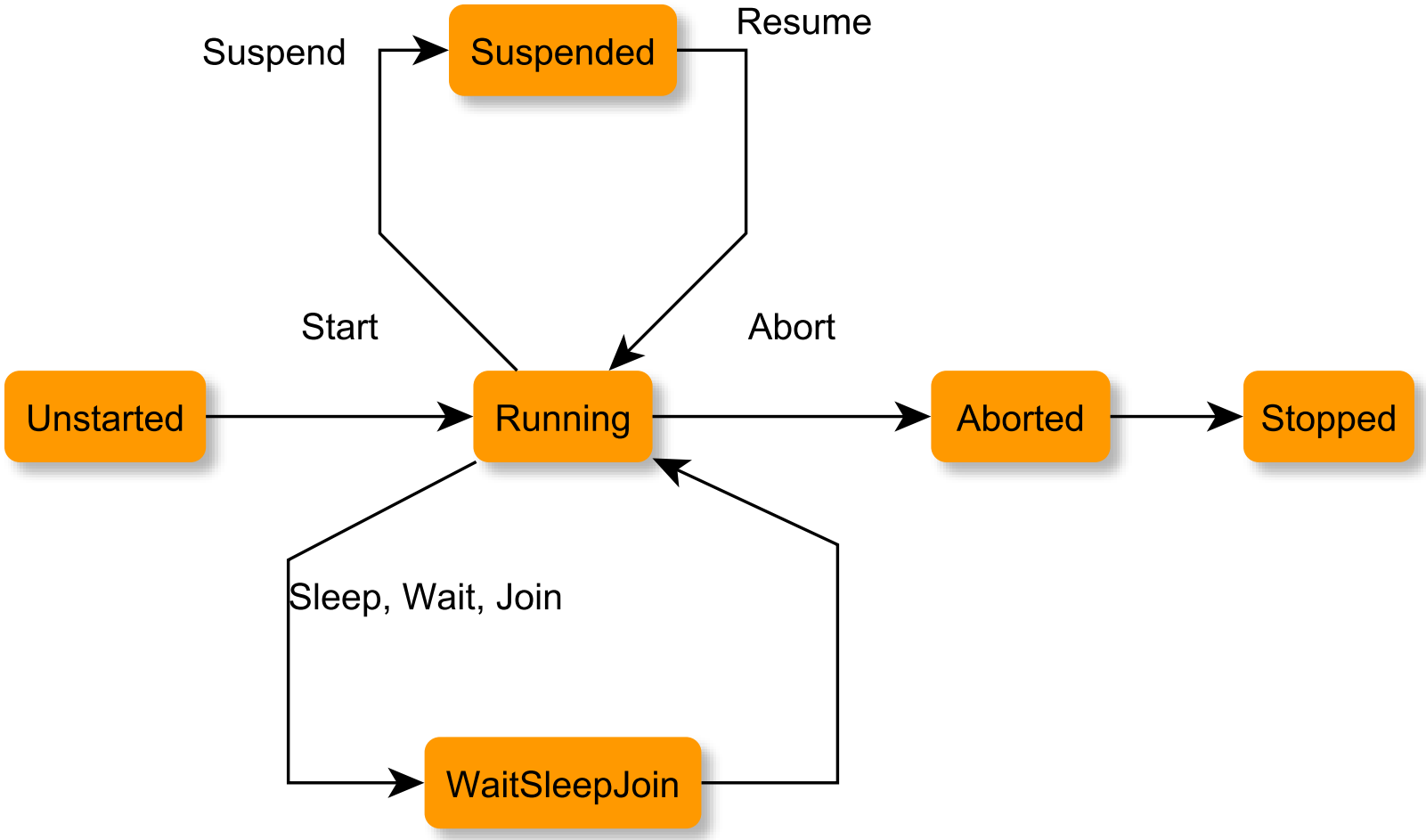
Thread Állapot

- Enumeráció
- Bitek (egyszerre több állapot)

Állapotok

Unstarted	Thread.Start nem lett meghívva
Running	A szál fut
Background	Háttér szál
WaitSleepJoin	Blokkolt
StopRequested	Meg fog állni
Stopped	A szál megállt
SuspendRequested	A szál fel lesz függesztve
Suspended	Felfüggesztett
AbortRequested	Thread.Abort meg lett hívva
Aborted	A szál halott, Stopped-re vált.

Thread Állapot



Kritikus Szakasz

- Abort megszakítja a futást!
- Tetszőleges időpont
- Inkonzisztens állapot
- Kritikus szakasz
  - Előtte konzisztens állapot
  - Fontos kód, inkonzisztens állapot
  - Utána konzisztens állapot

```
Thread.
    BeginCriticalRegion();
//Kritikus szakasz
Thread.
    EndCriticalRegion();
```

Thread Prioritás

- Ütemezést befolyásolja
- OS figyelembe veheti (nem kötelező)
- Szálanként különböző
- Magasabb prioritás $\rightarrow$ fontosabb szál

1. Highest
2. AboveNormal
3. Normal
4. BelowNormal
5. Lowest

Tartalomjegyzék

Osztott erőforrás kezelés

- Közös kontextus
- Közös erőforrások
- Egyszerre hozzáférnek ✗
- Problémák
- Nem elemi műveletek

Lost Update

1. x = 5;
2. t1: read(x);
3. t2: read(y);
4. t1: write(x+3);
5. t2: write(x-4);

Osztott erőforrás kezelés

- Zárolás
- Elemi
- Csak egy szál férhet hozzá.
- Többi várakozik

```
lock(objektum)
{
//Zarolt resz, csak az
    adott szal kezelheti
    az objektumot
}
```

Producer–Consumer Probléma

- Klasszikus példa
- Szereplők
  - Producer
  - Consumer
  - Storage

Storage

- Osztott erőforrás
- Mind a termelők, mind a fogyasztók elérik

Producer

- Időközönként termel
- Eredményét a Storage–be rakja
- Vár, előállítja a következő terméket

Consumer

- Időközönként fogyaszt
- Storage–ből veszi ki
- Vár, feldolgozza a terméket

Holtpont

- Több erőforrás
- Több szál
- Felengedés a szakasz végén
- Lockolás sorrendje kötött

```
lock(A)
{
lock(B)
{
//muveletek
}
}
```

```
lock(B)
{
lock(A)
{
//muveletek
}
}
```

Kölcsönösen lock–olják a másiknak szükséges objektumot és egyik sem tud tovább lépni!

Monitor

- Osztály
- Statikus metódusok gyűjteménye
- Nem lehet példányosítani
- Kontextus független
- Thread safe
- Objektumokhoz való szinkronizált hozzáférés

Műveletek

- Enter
- TryEnter
- Wait
- Pulse (Signal)
- PulseAll
- Exit

Monitor

Enter

- Object paraméter
- Zárolja az objektumot
- Blokkolódik és vár

TryEnter

- bool visszatérési érték
- Holtpont elkerülés
  - blokkolódás mentes
  - Timeout

Exit

- Feloldja a zárolást

Wait

- Feloldja a zárolást
- Másik szál zárolásáig blokkolódik
- Várakozó sor

Pulse

- Elengedi az objektumot
- Következő szálat értesíti

PulseAll

- Összes várakozó szálat értesíti

Lock–olás finomítása

- Csak 1 szál férhet hozzá
- Különböző műveletek
  - Írás
  - Olvasás
- Különböző zárolások
- Finomabb szabályozás
- Egy írhat
- Többen olvashatják
- Ha írják nem lehet olvasni

Metódusok

- AcquireReaderLock
- AcquireWriterLock
- ReleaseReaderLock
- ReleaseWriterLock
- UpgradeToWriterLock