

Konkurens programozás

IP-18KPROGEG előadás



Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

Egy program egyszerre több mindent is csinálhat

- Számítással egyidőben IO
- Több processzor: számítások egyidőben
- Egy processzor több programot futtat (process)
 - többfelhasználós rendszer
 - időosztásos technika
- Egy program több végrehajtási szálból áll (thread)



- Hatékonyság növelése: ha több processzor van
- Több felhasználó kiszolgálása
 - egy időben
 - interakció
- Program logikai tagolása
 - az egyik szál a felhasználói felülettel foglalkozik
 - a másik szál a hálózaton keresztül kommunikál valakivel



- Megosztott (shared) memória
 - Több processzor, ugyanaz a memóriaterület
- Elosztott (distributed) memória
 - Több processzor, mindnek saját memória
 - Kommunikációs csatornák, üzenetküldés



Kevesebb processzor, mint folyamat

- A processzorok kapcsolgatnak a különböző folyamatok között
- Mindegyiket csak kis ideig futtatják
- A párhuzamosság látszata
- A processzoridő jó kihasználása
 - Blokkolt folyamatok nem tartják fel a többit
 - A váltogatás is időigényes!



Párhuzamosság egy folyamaton belül

- Végrehajtási szálak (thread)
- Ilyenekkel fogunk foglalkozni
- Pehelysúlyú (Lightweight, kevés költségű)
- „Megosztott” jelleg: közös memória



Beépített nyelvi támogatás: `java.lang`

- Nyelvi fogalom: (végrehajtási) szál, `thread`
- Támogató osztály: `java.lang.Thread`



- Nemdeterminisztikusság
- Az ember már nem látja át
- Kezelendő
 - ütemezés
 - interferencia
 - szinkronizáció
 - kommunikáció
- Probléma: tesztelés, reprodukálhatóság



Végrehajtási szálak létrehozása

- A főprogram egy végrehajtási szál
- További végrehajtási szálak hozhatók létre
- A Thread osztály példányosítandó
- Az objektum `start()` metódusával indítjuk a végrehajtási szálát
- A szál programja az objektum `run()` metódusában van



```
class Hello {  
    public static void main( String args[] ){  
        Thread t = new Thread();  
        t.start();  
    }  
}
```

Hát ez még semmi különöset sem csinál, mert üres a run()



Példa – tömöröbber

```
class Hello {  
    public static void main( String args[] ){  
        (new Thread()).start();  
    }  
}
```



A run() felüldefiniálása

```
class Hello {  
    public static void main( String args[] ){  
        (new MyThread()).start();  
    }  
}  
  
class MyThread extends Thread {  
    @Override public void run(){  
        while(true) System.out.println("Hi!");  
    }  
}
```



Névtelen osztállyal

```
class Hello {  
    public static void main( String args[] ){  
        (new Thread(){  
            @Override public void run(){  
                while(true)  
                    System.out.println("Hi!");  
            }  
        }).start();  
    }  
}
```



- Nem elég létrehozni egy Thread objektumot
 - A Thread objektum nem a végrehajtási szál
 - Csak egy eszköz, aminek segítségével különböző dolgokat csinálhatunk egy végrehajtási szállal
- Meg kell hívni a `start()` metódusát
- Ez elindítja a `run()` metódust egy új szálaban
- Ezt a `run()`-t kell felüldefiniálni, megadni a szál programját



`class` MyThread `extends` Thread

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



`class` `MyThread` `extends` `Thread`

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



`class` `MyThread` `extends` `Thread`

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



```
class MyThread extends Thread
```

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



`class` MyThread `extends` Thread

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



`class` MyThread `extends` Thread

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



`class` MyThread `extends` Thread

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



`class` `MyThread` `extends` `Thread`

```
...  
  
public void m(){  
    ...  
    Thread t = new MyThread();  
    ...  
    t.start();  
    ...  
}
```

```
...  
  
public MyThread(){  
    ...  
}  
  
public void start(){  
    ...  
}  
  
public void run(){  
    ...  
}  
  
...
```



Mit csinál ez a program?

```
class Hello {  
    public static void main( String args[] ){  
        (new MyThread()).start();  
        while(true) System.out.println("Bye");  
    }  
}  
  
class MyThread extends Thread {  
    @Override public void run(){  
        while(true) System.out.println("Hi!");  
    }  
}
```



Lehetséges kimenetek

Hi!

Bye

Hi!

Bye

Hi!

Bye

Hi!

Bye

Hi!

Bye

Hi!

Bye

Hi!

Bye

Hi!

...



Lehetséges kimenetek

Hi!	Hi!
Bye	Hi!
Hi!	Hi!
Bye	Hi!
Hi!	Bye
Bye	Bye
Hi!	Bye
Bye	Bye
Hi!	Hi!
Bye	Hi!
Hi!	Hi!
Bye	Hi!
Hi!	Bye
Bye	Bye
Hi!	Bye
...	...



Lehetséges kimenetek

Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
...

Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
...

Hi!
Hi!
Bye
Bye
Bye
Hi!
Bye
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Hi!
...



Lehetséges kimenetek

Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
...

Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
Bye
Bye
...

Hi!
Hi!
Bye
Bye
Bye
Bye
Hi!
Bye
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
...

Bye
Bye
Bye
Bye
Hi!
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
...



Lehetséges kimenetek

Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
...

Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
Bye
...

Hi!
Hi!
Bye
Bye
Bye
Bye
Hi!
Bye
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Hi!
...

Bye
Bye
Bye
Bye
Hi!
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
...

Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
...



Lehetséges kimenetek

Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
Bye
Hi!
...

Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Bye
Bye
Bye
...

Hi!
Hi!
Bye
Bye
Bye
Bye
Hi!
Bye
Bye
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
...

Bye
Bye
Bye
Bye
Hi!
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
Bye
...

Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
Hi!
...

HiBye
!
Hi!
Hi!
Bye
HBiy!e

Bye
Hi!
Hi!
Hi!
Bye
ByeH
i!
Hi!
...



Lehetséges kimenetek

Hi!	Hi!	Hi!	Bye	Hi!	HiBye	Hi!
Bye	Hi!	Hi!	Bye	Hi!	!	... x3552
Hi!	Hi!	Bye	Bye	Hi!	Hi!	Hi!
Bye	Hi!	Bye	Bye	Hi!	Hi!	Bye
Hi!	Bye	Bye	Bye	Hi!	Bye	... x6242
Bye	Bye	Bye	Hi!	Hi!	HBiy!e	Bye
Hi!	Bye	Hi!	Bye	Hi!		...
Bye	Bye	Bye	Bye	Hi!	Bye	Hi!
Hi!	Hi!	Bye	Bye	Hi!	Hi!	... x5923
Bye	Hi!	Bye	Bye	Hi!	Hi!	Hi!
Hi!	Hi!	Hi!	Bye	Hi!	Hi!	Bye
Bye	Hi!	Hi!	Bye	Hi!	Bye	... x4887
Hi!	Bye	Hi!	Bye	Hi!	ByeH	Bye
Bye	Bye	Hi!	Bye	Hi!	i!	Hi!
Hi!	Bye	Hi!	Bye	Hi!	Hi!	Hi!
...



- Ütemezéstől függ
- A nyelv definíciója nem tesz megkötést az ütemezésre
- Különböző platformokon / virtuális gépeken különbözőképpen működhet
- A virtuális gép meghatároz(hat)ja az ütemezési stratégiát
- De azon belül is sok lehetőség van
- Sok mindentől függ (pl. hőmérséklettől)



Ütemezési stratégiák

Run to completion

egy szál addig fut, amíg csak lehet (hatékonyabb)

Preemptive

időosztásos ütemezés (igazságosabb)



Ütemezési stratégiák

Run to completion

egy szál addig fut, amíg csak lehet (hatékonyabb)

Preemptive

időosztásos ütemezés (igazságosabb)

Összefoglalva: írjunk olyan programokat, amelyek működése nem érzékeny az ütemezésre



Pártatlanság (fairness)

- Ha azt akarjuk, hogy minden szál „egyszerre”, „párhuzamosan” fusson
- Ha egy szál már „elég sokat” dolgozott, adjon lehetőséget más szálaknak is
- `Thread.yield()`
- Ezen metódus meghívásával lehet lemondani a vezérlésről
- A `Thread` osztály statikus metódusa



Felváltva írnak ki?

```
class Hello {  
    public static void main( String args[] ){  
        new Thread(){ @Override public void run(){  
            while(true){  
                System.out.println("Hi!");  
                Thread.yield();  
            }  
        }  
    }.start();  
    while(true){  
        System.out.println("Bye");  
        Thread.yield();  
    }  
}
```



A java.lang.Runnable interface

- Java-ban osztályok között egyszeres öröklődés
- Végrehajtási szálnál leszarmaztatás a Thread osztályból „elhasználja” azt az egy lehetőséget
- Megoldás: ne kelljen leszarmaztatni
- Megvalósítjuk a Runnable interfészt, ami előírja a run() metódust
- Egy Thread objektum létrehozásánál a konstruktornak átadunk egy futtatható objektumot



Szál programjának megadása: extends Thread

```
class Hello {  
    public static void main( String args[] ){  
        new MyThread().start();  
        ...  
    }  
}
```

```
class MyThread extends Thread {  
    @Override public void run(){  
        ...  
    }  
}
```



Szál programjának megadása: implements Runnable

```
class Hello {  
    public static void main( String args[] ){  
        new Thread(new MyRunnable()).start();  
        ...  
    }  
}
```

```
class MyRunnable implements Runnable {  
    @Override public void run(){  
        ...  
    }  
}
```



Szál programjának megadása: névtelen osztállyal

```
class Hello {  
    public static void main( String args[] ){  
        new Thread(new Runnable(){  
            @Override public void run(){  
                ...  
            }  
        }  
    ).start();  
    ...  
}
```



Szál programjának megadása: lambda-kifejezéssel

```
class Hello {  
    public static void main( String args[] ){  
        new Thread( () -> {  
            while(true)  
                System.out.println("Hi!");  
        }  
    ).start();  
    ...  
}  
}
```

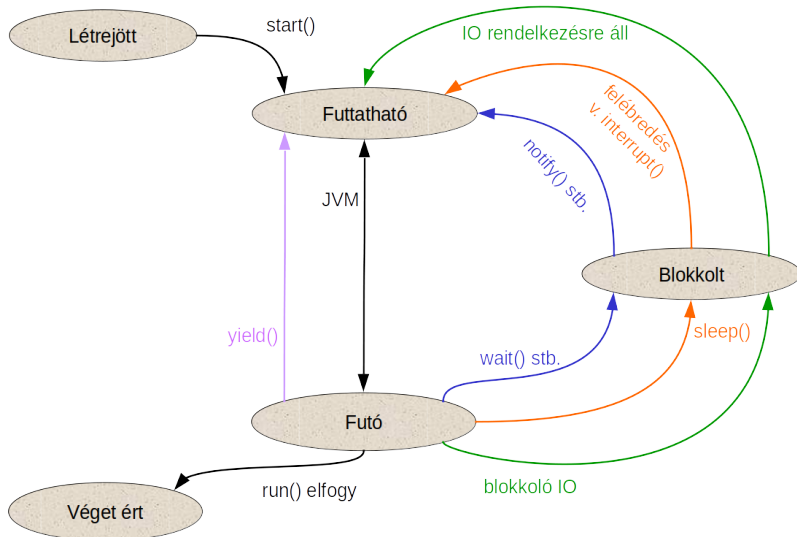


Szál bevárása

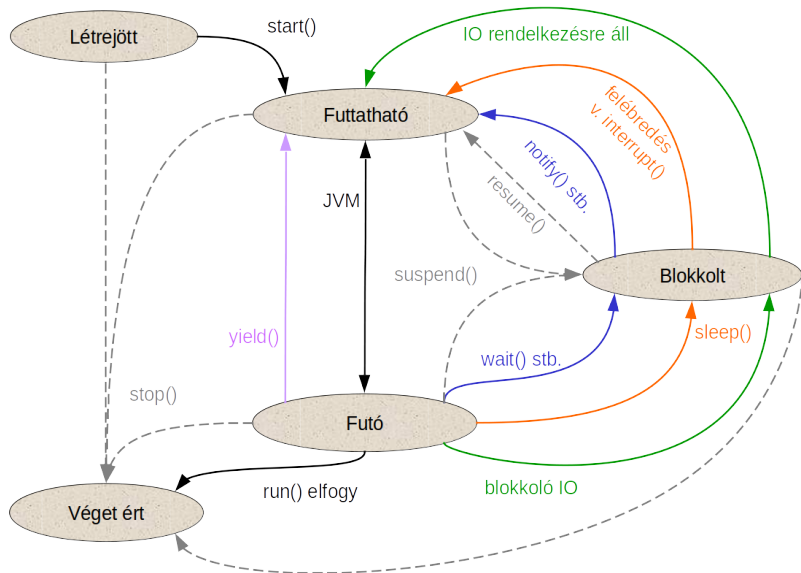
```
class Hello {  
    public static void main( String args[] ){  
        Thread t = new Thread( () -> ... );  
        t.start();  
        ...  
        try {  
            t.join();  
        } catch( InterruptedException e ){  
            // t has been interrupted  
        }  
        ...  
    }  
}
```



Életciklus



Életciklus – elítélendő állapotátmenetekkel



Thread.State

- NEW
- RUNNABLE
- BLOCKED
- WAITING
- TIMED_WAITING
- TERMINATED



```
class SleepDemo extends Thread {  
  
    public void run(){  
        while(true){  
            try { sleep(1000); }  
            catch ( InterruptedException ie ){ }  
            System.out.println(new java.util.Date());  
        }  
    }  
  
    public static void main(String[] args) {  
        (new SleepDemo()).start();  
        while(true) System.err.println();  
    }  
  
}
```



```
class IODemo extends Thread {  
  
    public void run(){  
        while(true){  
            try { System.in.read(); }  
            catch ( java.io.IOException ie ){ }  
            System.out.println(new java.util.Date());  
        }  
    }  
  
    public static void main(String[] args) {  
        (new IODemo()).start();  
        while(true) System.err.println();  
    }  
  
}
```



- A `stop()` metódus nem javasolt.
- Bízunk rá a szádra, hogy mikor akar megállni.
 - Erőforrások elengedése
- Ha a `run()` egy ciklus, akkor szabjunk neki feltételt.
- A feltétel egy *flaget* figyelhet, amit kívülről átbillenthetünk.



```
class MyAnimation extends ... implements Runnable {  
    ...  
    private volatile boolean running = false;  
    public void startAnimation(){...}  
    public void stopAnimation(){...}  
    @Override public void run(){...}  
    ...  
}
```



Példa – animáció megvalósítása

```
public void startAnimation(){
    running = true;
    (new Thread(this)).start();
}

public void stopAnimation(){
    running = false;
}

@Override public void run(){
    while(running){
        ...    // one step of animation
        try{ sleep(20); }
        catch(InterruptedException e){...}
    }
}
```

