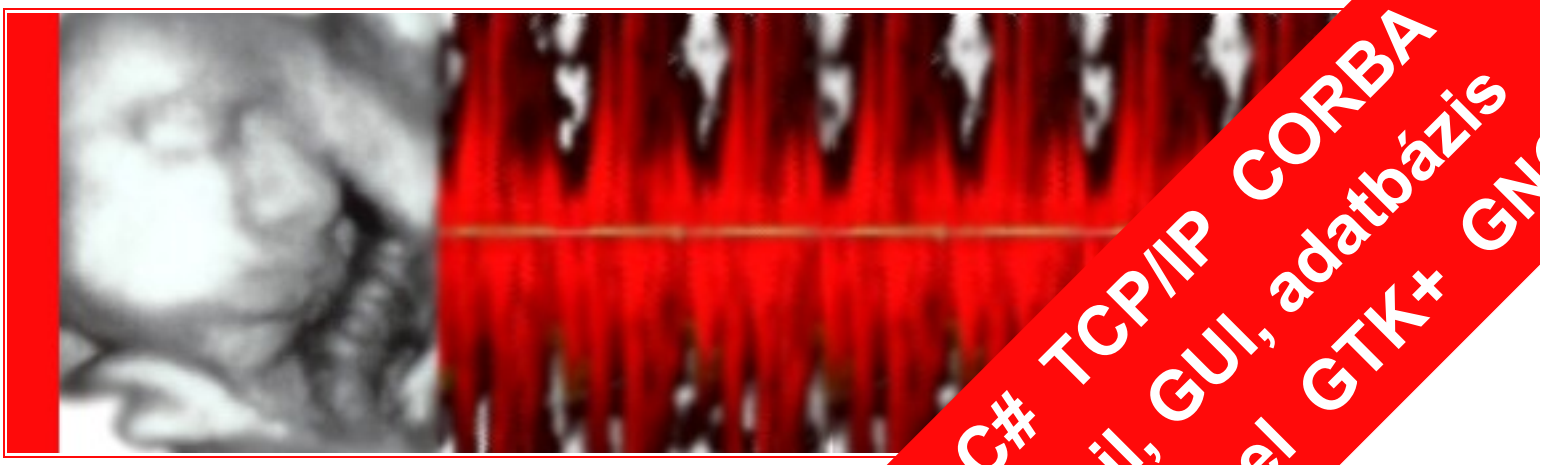
GNU FDL

PROGRAMOZÓ PÁTERNOSZTER

Kezdőknek és haladóknak egyaránt ajánljuk!



Megdobogtatja a programozók szívét!

GNU/Linux C C++ Java C# TCP/IP CORBA Swi
Hálózati, konkurens, mobil, GUI, adatbázis
programozás Linux kernel GTK+ GN

Bátfai Norbert

PROGRAMOZÓ PÁTERNOSZTER

http://www.inf.unideb.hu/~nbatfai/ProgramozoPaternoszter.pdf

„Belépés a gépek mesés birodalmába”

Ajánlás és vízió

Az anyagot elsősorban felsőoktatásbeli informatikus hallgatóknak ajánljuk. Víziónk, hogy a hallgatók a felsőoktatásból napjainkban kikerülve otthonosan dolgoznak **C**, **C++**, **Java™** és **C#** nyelveken, illetve az egyiken és a hozzá kapcsolódó platformon elmélyült tudással rendelkeznek. A jegyzet egyben a Debreceni Egyetem Informatikai Karának *Operációs rendszerek I.*, *II.*, *Mobil programozás* és *Hálózatok* laborgyakorlatainak anyagát is rögzíti. Fő célja a **GNU/Linux** rendszerek szeretetének átadása és általában a UNIX-típusú rendszerek **C** programozásának megismertetése, megszerettetése! Továbbá a **Java** programozás gyakorlása.

Ízelítő a tartalomból

- ✔ Unix típusú rendszerek használata *(felhasználói, rendszergazdai szint; bash programozás)*
- ✔ **C rendszerszint** *(folyamatok; jelek; szálak; IPC stb.)*
- ✔ **Hálózati programozás** *(TCP, UDP kliens-szerver; ICMP ping; RPC; Java RMI; CORBA, Java IDL, ORBit; Web programozás: Java szervletek, PHP; Java Bluetooth, **nem blokkolódó-multiplexelt**, **párhuzamos** és **elosztott** példák stb.)*
- ✔ Konkurens programozás *(SVr4 szemaforok; POSIX szálak stb.)*
- ✔ **C kernelszint** *(Linux 2.6; kernelmodulok; a Proc fájlrendszer stb.)*
- ✔ A **C** mellett **C++**, **Java** és **C#** nyelvű példák
- ✔ Mobil programozás *(Java ME; Symbian stb.)*
- ✔ GUI programozás *(AWT; Swing; GTK+; GNOME; Java-GNOME stb.)*
- ✔ Adatbázis programozás *(MySQL; PostgreSQL, libpq; JDBC stb.)*
- ✔ Kvantum számítások *(kísérletek; qubit; teleportáció stb.)*
- ✔ Bioinformatika *(bioqubit; Penrose-Hameroff 'Orch OR' tudatmodell stb.)*

Gép melletti fogyasztásra!

✔ ***C** példaprogramok, források száma:*	*110 db*
✔ ***C++** példaprogramok, források száma:*	*4 db*
✔ ***Java** példaprogramok, források száma:*	*40 db*
✔ ***C#** példaprogramok, források száma:*	*7 db*
✔ ***Kvantum számítógépes** algoritmusok:*	*5 db*

Köszönet

A Debreceni Egyetem Informatikai Karának [*Egyéb*/IK] PM, IT, PTM, PTI és MI szakos hallgatóinak, hogy az Operációs rendszerek I., II. és a Mobil programozás, illetve a Hálózatok gyakorlatokon lelkes részvételükkel katalizálták az anyag készítését, illetve, hogy az e formában való közös felhasználás, feldolgozás során keletkezett tapasztalatokat vissza tudtam forgatni az anyag írásába. A Solaris-os és Nokia N-Gage, Nokia 6600, 6131 és a LEGO® Mindstorms™ Robotics Invention System™, NXT-s és a 64 bites rendszeres példák kipróbálhatóságáért köszönet az EUROSMOBIL-nak [*Egyéb*/EM].

Debreceni Egyetem Informatikai Kar
Bátfai Norbert
nbatfai@inf.unideb.hu

verzió: 0.0.247, 2006. november 12.

A borítólapon Bátfai Mátyás Bendegúz szerepel, 31. hetes korában, 2005 decemberében. A jegyzetben található programokat és ábrákat a szerző készítette.

Személyes ajánlás

Feleségem pocakjában a programozással ismerkedő kisfiamnak, megnyugvással annak tudatában, hogy neki ezt fellapozni már csakis történeti visszatekintés lesz, nem izgalmas közelmút, jelen és jövő, mint például nekem.

Ábra 1: Bátfai Mátyás Bendegúz 24 hetes korában, 2005. októberében

Mottó

Mi a különbség a természettudósok és a hackerek között?

- Ha a természettudósokat felfedezőknek tekintjük, akkor ők mennek és meghódítják a Mount Everestet.
- Ha a hackereket tekintjük felfedezőknek, akkor ők összehordják a Mount Everestet, aztán meghódítják.

Szervezési könyvjelzők

- A jegyzet példaprogramjainak összefoglalását láthatjuk a *8*.[1] oldal, a *Programok és platformok* című pont táblázatában.
- A jegyzet anyagának más laborgyakorlatokhoz való kapcsolhatóságáról olvashatunk a *12*. oldalon, a *Feldolgozási javaslatok* című pontban.
- A jegyzet használatához fűzött figyelmeztetést olvassuk el a *6*.oldalon, a *FIGYELMEZTETÉS* pontban.
- Elgondolkodtató, programoztató kérdéseket, feladatokat találunk az *280*. oldalon, a *Gondolkodtató, programoztató kérdések* című fejezetben.
- Megvannak a válaszok és némi segítség is a válaszok megadásához a *287*. oldalon, a *Válaszok a programoztató kérdésekre* című fejezetben.

Főbb tartalmi könyvjelzők

1 Ha a jegyzetet nem papíron, hanem pdf-ben használjuk, akkor az aláhúzva és döntve szedett hivatkozásokat – mint amilyen ez is – könnyen tudjuk követni az egérrel!

Jóval részletesebb és programozási nyelvenkénti tartalmi bontást találunk az *8*. oldal, a *Programok és platformok* című pont táblázatában.

Technikai könyvjelzők

Előszó

Az anyagnak írása a Debreceni Egyetem, Informatikai Karának *Operációs rendszerek I.* és *II.* laborgyakorlata közben kezdődött meg, s egy része a gyakorlat mellett (illetve időközben a *Mobil programozás* és a *J2SE álózatok* laborgyakorlatok mellett), előtt - közben - után készül, így köszönettel veszem a visszajelzéseket, a hibák jelzését, vagy bármely más észrevételt elektronikus levélben a bn@javacska.hu avagy az nbatfai@inf.unideb.hu címekre.

Az anyag azt feltételezi, hogy a hallgatóknak nincs Unix-típusú rendszerekkel gyakorlata és a C programozást is csak ezzel a kurzussal párhuzamosan kezdik meg önképzés keretében. Ennek ellenére arra bátorítom az érdeklődő hallgatóságot, hogy C tanulmányai mellett kezdjen ismerkedni a C++, Java és a C# nyelvekkel, illetve a kvantum számításokkal is – ezt a C rendszerszint című fejezetben és az ezt követő részekben C, Java, C++ és C# példákkal próbálom majd segíteni. Illetve ennek megfelelően a tárgy második féléve gyakorlatainak példáit a hallgatóság a C mellett ezeken a további nyelveken is élvezheti.

A jegyzet írása – az *Operációs rendszerek* laborok mellett a – a *Mobil programozás* és a *J2SE - hálózatok* című laborok mellett is folytatódik, ennek megfelelően e gyakorlatok anyaga is itt, a Programozó Páternoszterben kap helyet, további részletek tekintetében lásd a *12*. oldal, *Feldolgozási javaslatok* című pontját!

Előfeltételek :)

A Diák-Robot Barátság lapok: *http://www.javacska.hu*, [*Egyéb*/DR]
A Jávácska Vortál lapjai: *http://javacska.lib.unideb.hu*, [*Egyéb*/JV]

Megjegyzés és FIGYELMEZTETÉS

Az anyag folyamatosan készül, a szereplő programokban az átgondolt hibakezelést elhanyagoljuk, inkább csak jelezzük, például egy kiíratással annak szükségességét, illetve számos esetben előfordul, hogy még ettől is eltekintünk.

A szerző a programok készítésénél a legjobb tudása szerint jár el, de lehetnek, bizonyára vannak hibák. Illetve vannak olyan lépésekben fejlesztett példák, amelyekben a kezdő változatok eleve hibásak, ezért nem is érhetők el például külön a programok, hanem csak ebből az anyagból, azaz ennek az anyagnak a részeként.

A jegyzet, a jegyzet programjainak használatából eredeztethető esetleges károkért a szerző semmilyen felelősséget nem vállal.
Ebből a szempotból külön is kiemelendők azok a nem felhasználói programok, amik, vagy az amikkel való kísérletezés az egész rendszert összeomlaszthatja! A szerző sajat szomorú tapasztalatából :) javasolja, hogy ezeket a programokat olyan gépen próbálgassuk, amit ennek tudatában választunk a tanuláshoz.

Hasonló óvatosságra intünk a jegyzet hálózati programjai kapcsán, ezeket, ezek módosításait ne az Interneten, hanem a `localhost`-on, vagy – a rendszergazdákkal egyeztetve – a lokális belső hálózatainkon teszteljük.

A szerzőről

Bátfai Norbert 1996-ban szerzett programozó matematikusi, majd 1998-ban kitüntetéses programtervező matematikusi oklevelet a Debreceni Egyetemen. 1998-ban megnyerte a Java Szövetség Java Programozási Versenyét.

2. ábra: A szerző és fia a LOKI pályán!
(http://www.dvsc.hu – Hajrá, LOKI!)

Bátfai Erikával közös mobil-információtechnológiai cége, az Eurosmobil [*Egyéb*/EM], második helyezést ért el 2004-ben a Motorola JavaJáték Versenyén, ugyancsak az Eurosmobil 2004-ben a Sun és a Nokia közös Mobil Java Fejlesztői Versenyén a „Ha hívsz, támadok!” (H.A.H) hálózati (Java EE szerver, Java ME kliens) játéksorozattal első díjat nyert.
Társszerzője a Fantasztikus programozás [*Egyéb*/DR, JV] című ismeretterjesztő kalandregény sorozatnak.
Jelenleg a Debreceni Egyetem Informatikai Karának munkatársa is, ahol most Operációs rendszerek gyakorlatokat tart. Oktatási tapasztalata az alábbi tárgyak gyakorlatain alapul: Java esettanulmányok, J2SE hálózatok, Java appletek, CORBA, Programozás, Hálózatok, Formális nyelvek és automaták, Algoritmuselmélet, Bevezetés az informatikába, Operációs rendszerek, Alkalmazások fejlesztése WWW-re, Objektumorientált programozás a középiskolában.

Milyen GNU/Linux rendszert használunk mi?

2005. november 27., Fedora Core 4, http://fedora.redhat.com/

```
$ more /etc/redhat-release
Fedora Core release 4 (Stentz)
```

Az anyagban az eddig (a pontos részletek tekintetében lásd a következő pontot) bemutatott példák legtöbbjéhez szükséges szoftverek rajta vannak a Fedora Core 4 telepítő lemezén, mint például a gcj, a GTK+ vagy akár a Java-GNOME, Tomcat, MySQL, PostgreSQL, stb.

2006. szeptember 18., Fedora Core 4, http://fedora.redhat.com/

```
$ more /etc/redhat-release
Fedora Core release 5 (Bordeaux)
```

Az anyag fejlődése

Az anyag verziótörténetét a dokumentum végén (*290*. oldal) találhatjuk meg, a jelenlegi állapot: **2006-11-12**, Az anyag elkészültségi foka ~ 42%.

A jegyzetet az OpenOffice.org 2.0 Writer szövegszerkesztővel készítettük/készítjük: http://www.openoffice.org (de szintén megtalálható a Fedora Core 4 telepítő lemezén is).

Oktatáspolitikai melléklet

A millió programozó országa: a tömegképzést úgy is fel lehet fogni, mint lehetőséget a jó programozók tömeges képzésére.

Oktatási történet

- 2004/2005 tanév, II. félév: **Operációs rendszerek 1.** labor, DE IK, IT, PM, PTM[1]. Tematika a jegyzet *12*.oldal *Az Operációs rendszerek 1. labor tematikája* c. pontja alapján.
- 2005/2006 tanév, I. félév: **Operációs rendszerek 2.** labor, DE IK, IT, PM, PTI. Tematika a jegyzet *13*.oldal *Az Operációs rendszerek 2. labor tematikája* c. pontja alapján.
- 2005/2006 tanév, II. félév:
  - **Operációs rendszerek 1.** labor, DE IK, MI, PTI. Tematika a jegyzet *12*.oldal *Az Operációs rendszerek 1. labor tematikája* c. pontja alapján.
  - **Alkalmazások fejlesztése WWW-re**, DE IK, levelező IT. Tematika a jegyzet *16*.oldal *Alkalmazások fejlesztése WWW-re labor tematikája* c. pontja alapján.
- 2006/2007 tanév, I. félév:
  - **Operációs rendszerek 2.** labor, DE IK PTM. Tematika eseti, hibrid (és eleinte csoportbontás), mert a csoportok fele nem e jegyzet alapján dolgozott az első félévben.
  - **Mobil programozás** labor, DE IK, MI, PTI, PTM. Tematika a jegyzet *15*.oldal *A Mobil programozás labor tematikája* c. pontja alapján.
  - **Alkalmazások fejlesztése WWW-re**, DE IK, levelező IT.
  - **J2SE - hálózatok** labor, DE IK, MI, PTI, PTM. Tematika a jegyzet *14*.oldal *J2SE – hálózatok labor tematikája* c. pontja alapján.

1 DE IK: Debreceni Egyetem Informatikai Kar, PM: programozó matematikus, PTM: programtervező matematikus, IT: informatika tanár, PTI: programtervező informatikus, MI: mérnök informatikus.

I. Az anyag szervezése

Az írás során igyekeztünk/igyekszünk az anyagot bőségesen ellátni kereszthivatkozásokkal, ezért javasoljuk az on-line böngészését. Így ahelyett, hogy a megfelelő oldalakra próbálnánk lapozni, érdemes figyelni a kurzort és egyszerűen ugrani, például gyakorlásképpen ide: *173*. oldal, *Kernel blog* című pont. Hasonlóan a legtöbb helyen kiírjuk az oldalszámot és a hivatkozott fejezet címét is, ezeket a részeket megdöntve és aláhúzva szedjük.

Az anyag feldolgozását szigorúan gép mellett ajánljuk! Több okból, de a legpraktikusabb, hogy a legnagyobb részt kitevő C programozással foglalkozó részekben folyamatosan hivatkozunk a manuál lapokra. Ezért például, a *43*. oldalon, a *Sigaction* című pontban a jegyzetbe nem vettük be a `sigaction` struktúra leírását, mert feltesszük, hogy az olvasó folyamatosan látja a

```
$ man sigaction
```

lapot. Tehát leírások helyett inkább működő kis példákat készítettünk, amit az olvasó könnyen át tud vágni kedvenc szövegszerkesztőjébe és máris mehet a fordítás – *265*. oldal, *Fordítás* című pont – kipróbálás... Hajrá!

I.1 Programok és platformok

A táblázatban szereplő számok azt az oldalt jelölik, ahol a szóban forgó programok, vagy azok első változatai találhatóak.

Táblázat 1: Programok és platformok (QC=Kvantum számítógép)

Programok, „programcsaládok”	***C***	***C++***	***Java***	***C#***	***QC***
Bevezetés					
Bevezetés/szóhossz	*25*				
Bevezetés/32, 64 bites rendszerek	*26*				
Bevezetés/végtelen ciklus	*27*				
Bevezetés/cat	*30*				
Bevezetés/”Helló Világ!”	*30*		*31*		
Bevezetés/véletlen számok	*32*		*35*		
Folyamatok					
Folyamatok/forkolás	*36*				
Folyamatok/zombik	*38*				
Folyamatok/riasztás	*39*				
Folyamatok/nem lokális ugrások	*42*				
Folyamatok/szignálkezelés	*41*				
Folyamatok/IPC/(SysV)szemaforok	*46*				
Folyamatok/IPC/socketek (lokális, anonim)	*47*				
Folyamatok/IPC/csővezetékek	*52*				
Folyamatok/IPC/(SysV)üzenetsorok	*55*				

Programok, „programcsaládok”	***C***	***C++***	***Java***	***C#***	***QC***
Folyamatok/IPC/(SysV)üzenetsorok példa: a Pi jegyei/BBP algoritmus több folyamattal (lásd Tudományos számítások is)	*231*				
Folyamatok/IPC/(SysV)osztott memória	*59*				
Konkurencia					
Konkurencia/szálak	*64*				
Konkurencia/mutex zárak	*66*				
Konkurencia/ebédelő filoszok (pthread szemaforok és feltételes változók, illetve általában a szálak szinkronizálása)	*70*		*78*	*79*	
Konkurencia/CPU idő	*66*				
Fájlrendszer, fájlkezelés					
Fájlrendszer/tulajdonságok	*81*				
Fájlrendszer/könyvtárak	*81*				
Fájlkezelés/bináris/szöveges kiír/beolvas	*82*	*84*	*86*	*87*	
Fájlkezelés/rendszer statisztika példa	*89*				
Fájlkezelés/rendszer statisztika/pidcs példa	*94*				
Hálózati programozás					
Hálózati/TCP socket szerverek/soros	*102*		*130*	*132*	
Hálózati/TCP socket szerverek/soros, IPv6	*104*				
Hálózati/TCP socket szerverek/párhuzamos, folyamatokkal	*106*		*130*		
Hálózati/TCP socket szerverek/párhuzamos, folyamatok sorával	*109*				
Hálózati/TCP socket szerverek/párhuzamos, folyamatok sorával, szemaforral védett accept	*111*				
Hálózati/TCP socket szerverek/párhuzamos, folyamatok sorával, zárolással védett accept	*113*				
Hálózati/TCP socket szerverek/párhuzamos, szálakkal	*116*				
Hálózati/TCP socket szerverek/párhuzamos, szálak sorával	*117*				
Hálózati/TCP socket szerverek/párhuzamos, mutex-el védett accept	*120*				
Hálózati/TCP socket szerverek/soros, nem blokkolódó IO multiplexeléssel	*121*		*131*		
Hálózati/TCP socket szerverek/párhuzamos, nem blokkolódó IO multiplexeléssel	*123*				
Hálózati/TCP socket kliensek	*132*		*134*	*134*	

Programok, „programcsaládok”	*C*	*C++*	*Java*	*C#*	*QC*
Hálózati/TCP socket kliensek, IPv6	*133*				
Hálózati/nem szálbiztos	*129*				
Hálózati/SMTP levél	*137*		*136*		
Hálózati/TCP alapú csevegő			*138*		
Hálózati/TCP alapú csevegő, nem blokkolódó IO multiplexeléssel			*141*		
Hálózati/UDP szerver, kliens	*148*		*156*		
Hálózati/SOCK_RAW, ICMP_ECHO	*158*			*159*	
Hálózati/Java RMI			*161*		
Hálózati/CORBA/ORBit/névszolgáltató	*164*				
Hálózati/CORBA/Gnorba/névszolgáltató	*166*				
Hálózati/Java szervletek			*169*		
Hálózati/Bluetooth					
Kernel programozás					
Kernel/első kernelmodulok	*174*				
Kernel/printk	*175*				
Kernel/task_struct	*176*				
Kernel/seq_file egyszerűen	*177*				
Kernel/seq_file kevésbé egyszerűen					
Kernel/fs/proc/generic.c-ben ajánlott					
Kernel/ue. lapméretnél kevesebb adattal					
Kernel/a `norbi()` rendszerhívás	*180*				
GUI programozás					
GUI/GTK+, ProgPáter TCP kliens	*190*				
GUI/GNOME, ProgPáter TCP kliens felülete					
GUI/Java-GNOME, ProgPáter TCP kliens felülete			*201*		
GUI/AWT, ProgPáter TCP kliens felülete			*186*		
GUI/Swing, ProgPáter TCP kliens			*183*		
GUI/Teljes képernyő			*188*		
Mobil programozás					
Mobil/Symbian C++ „Helló, Erika!”		*216*			
Mobil/Java ME/MIDlet életciklus			*217*		
Mobil/Java ME/Vászon/Pálcikaember 1-3			*218*		
Mobil/Java ME/Vászon/Pálcikaember sprite			*225*		

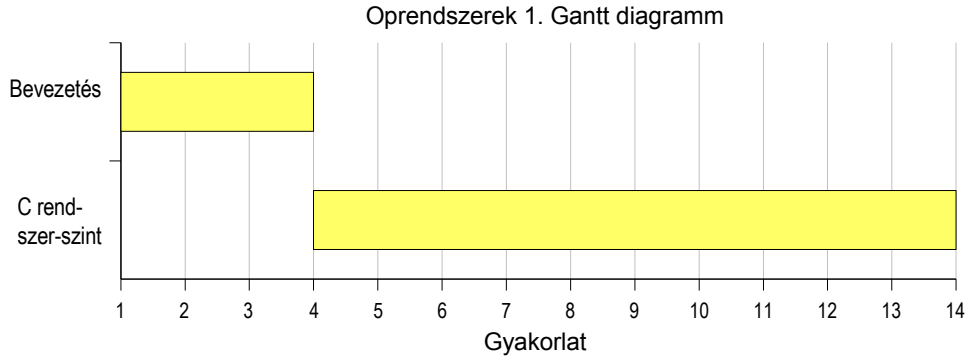
Programok, „programcsaládok”	*C*	*C++*	*Java*	*C#*	*QC*
Adatbázis programozás					
Adatbázis/MySQL/JDBC			*204*		
Adatbázis/MySQL/PHP					
Adatbázis/PostgreSQL/libpq					
Adatbázis/PostgreSQL/JDBC					
Web programozás					
Java szervletek			*169*		
PHP					
Tudományos számítások					
Pi jegyei/BBP algoritmus	*228*				
Pi jegyei/BBP algoritmus több folyamattal (párhuzamos példa)	*231*				
Pi jegyei/BBP algoritmus több géppel (elosztott példa)	*238*				
Kvantum algoritmusok					
Kvantum konstans orákulum					*254*
Kvantum sűrűségi kódolás					*257*
Kvantum teleportálás					*258*

I.2 Feldolgozási javaslatok

Azaz lehetséges válaszok arra a kérdésre, hogy hogyan szervezhetjük a jegyzet anyagának elsajátítását? Hogyan építhetjük fel rá az Operációs rendszerek című tárgy laborgyakorlatait, illetve hogyan kapcsolódhat a jelen anyag más laborgyakorlatokhoz?

I.2.1 Az Operációs rendszerek 1. labor tematikája

A GNU/Linux rendszert nem felhasználóként, hanem jóval mélyebben, programozóként, programokon, programozáson keresztül ismerjük meg, de mivel a C programozási kurzus egyszerre indul a jelen kurzussal, így az első néhány gyakorlaton szoktatjuk a hallgatóság szemét a körülbelül a negyedik gyakorlattól beinduló tartalmi anyaghoz, a C rendszerszinthez:



Ábra 3: Operációs rendszerek 1. laborgyakorlat Gantt diagramm

A labor részletes bontása a **C** nyelvű példák feldolgozásával a következő lehet:

Gyakorlat	***Téma***
1-3.	Bevezetés: *21*. oldaltól.
4-6.	C rendszerszint/folyamatok: *36*. oldaltól.
7.	C rendszerszint/jelkezelés: *41*. oldaltól.
8.	C rendszerszint/IPC: *46*. oldaltól.
9-10.	C rendszerszint/konkurencia: *64*. oldaltól, részlegesen.
11.	C rendszerszint/fájlrendszer, fájlkezelés: *81*. oldaltól.
12-14.	C rendszerszint/hálózati: *102*. oldaltól, részlegesen.

Táblázat 2 Operációs rendszerek 1. gyakorlat

I.2.1.1 Javasolt környezet

GNU/Linux, gcc fordító, manuál lapok.

I.2.1.2 Javasolt feldolgozás

A jegyzet alapján a források celebrálása, közben a megfelelő manuál lapok olvasása, a programok (parancsok) kipróbálása, illetve további irányított kísérletezgetés a programokkal.

I.2.2 Az Operációs rendszerek 2. labor tematikája

A **C**, **C++**, **Java**, **C#**, **QC** nyelvű példák feldolgozásával:

Gyakorlat	*Téma*
1-3.	Hálózati programozás folytatása, kimerítése.
4.	Konkurens programozás folytatása, kimerítése.
5-9.	C kernelszint.
10-11.	GUI programozás.
12.	Web programozás.
13.	Adatbázis programozás.
14.	Kvantum számítások, tudalmodellek figyelemfelhívó tárgyalása.

Táblázat 3 Operációs rendszerek 2. gyakorlat

I.2.2.1 Javasolt környezet

GNU/Linux, gcc fordító, manuál lapok. Java SE-JDK, NetBeans. GTK+, GNOME, Java-GNOME. ORBit. Tomcat. MySQL, PostgreSQL.

I.2.2.2 Javasolt feldolgozás

A jegyzet alapján a források celebrálása, közben a megfelelő manuál lapok olvasása, a programok (parancsok) kipróbálása, illetve további irányított kísérletezgetés a programokkal.

I.2.3 Önálló feldolgozás

Ez esetben javasoljuk, hogy telepítsük fel az otthoni gépünkre a *6*. oldal, *Milyen GNU/Linux rendszert használunk mi?* című pontjában ajánlott GNU/Linux rendszert vagy még inkább a feldolgozáskor éppen aktuális és népszerű rendszert, hogy a jegyzet példáit kipróbálhassuk.

I.2.4 Kapcsolat a kiegészítő és ráépülő tárgyakkal

I.2.4.1 Programozás

Érintett témák
Az anyagban C, C++, Java és C# példákat találunk, a források mennyisége is mutatja, hogy a hangsúlyt a C és a Java forrásokra helyeztük. A példaprogramok összefoglalását a 8. oldal, *Programok és platformok (QC=Kvantum számítógép)* táblázatban találjuk.

Táblázat 4 Programozás

I.2.4.2 Hálózati programozás

Érintett témák
TCP socket programozási bevezető példák.
UDP socket programozási bevezető példák.
Raw socket programozási bevezető példák.
Távoli metódushívás: RPC, Java RMI, CORBA.
Web programozás: Java szervletek, PHP.

Táblázat 5 Hálózati programozás

I.2.4.2.1 J2SE – hálózatok labor tematikája

Gyakorlat	*Téma*
1-2.	TCP/IP, hálózati programozási példák C-ben és Javaban.
3-4.	Multiplexelt, nem blokkolódó IO.
5-6.	Java RMI
7-11.	CORBA
12-14.	Java Bluetooth

Táblázat 6 J2SE - hálózatok gyakorlat

I.2.4.3 Mobil programozás

Érintett témák
Java ME a *217*. oldaltól
Symbian OS
Symbian C++ figyelemfelkeltés a *216*. oldaltól

Táblázat 7 Mobil programozás

I.2.4.3.1 A Mobil programozás labor tematikája

Gyakorlat	*Téma*
1.	Java ME fejlesztéssel kapcsolatos alapfogalmak.
2-3.	NetBeans 5.5 és a NetBeans Mobility telepítése és használata, például obfuscálás stb. Bevezető MIDlet példák, a fejlesztés menete, MIDlet életciklus.
4-5.	Java ME GUI programozás.
6.	Java ME multimédia programozás.
7.	Java ME vásznak és szálak, a játék vászon (GameCanvas) használata.
8.	Java ME adatbázis kezelés (RMS).
9-12.	Java ME hálózatkezelés.
13-14.	Java Bluetooth (JSR 82).

Táblázat 8 A Mobil programozás laborgyakorlat

I.2.4.4 Adatbázis programozás

Érintett témák
MySQL a *204*. oldaltól.
PostgreSQL a *213*. ldaltól.

Táblázat 9 Adatbázis programozás

I.2.4.5 Konkurens programozás

Érintett témák
SVr4 szemaforok, POSIX szálak, Java szálak, kölcsönos kizárás, szemaforok, monitorok, általában szálkezelés és szálak szinkronizációja.

Táblázat 10 Konkurens programozás

I.2.4.6 Algoritmuselmélet

Érintett témák
Kvantum számítási modellek, kvantum bonyolultsági osztályok a *251*. oldaltól.

Táblázat 11 Algoritmuselmélet

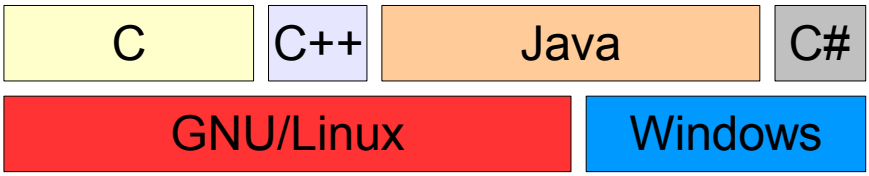
I.2.4.7 Alkalmazások fejlesztése WWW-re labor tematikája

Gyakorlat	*Téma*
1. alkalom	TCP/IP, hálózati programozási példák C-ben, HTTP protokoll.
2. alkalom	Hálózati programozási példák Java-ban.
3. alkalom	Mobil (Java ME) programozás vagy Java szervletek.

I.3 Áttekintés

I.3.1 Platformok és nyelvek

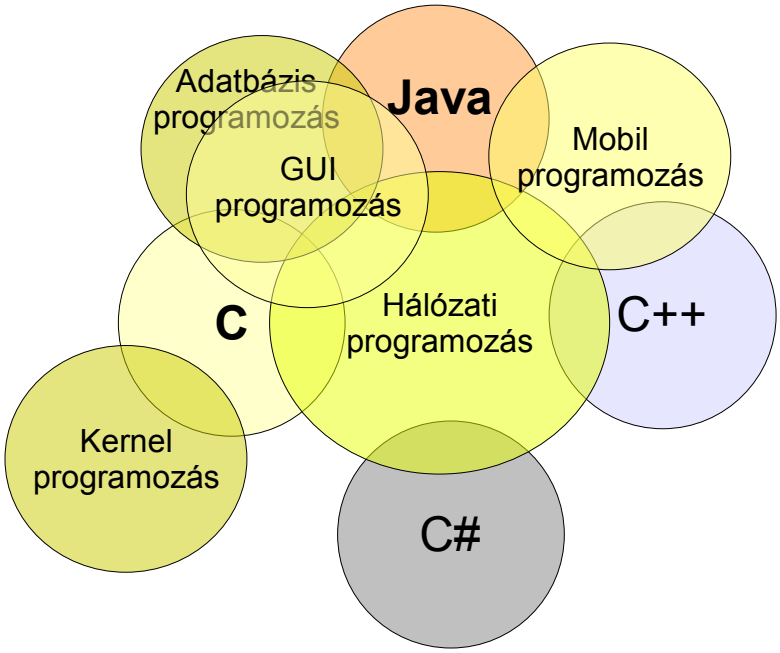
A jegyzetben használt platformok és nyelvek az alábbiak:



Ábra 4: Platformok és nyelvek a jegyzetben

I.3.2 Programozási témák

A jegyzetben használt programozási témák az alábbiak szerint is csoportosíthatóak:



Ábra 5: Programozási témák a jegyzetben

A példák részletesebb csoportosítását láthattuk a 8. oldal, a *Programok és platformok* című pont táblázatában.

I.4 Jelölések

I.4.1 Példaprogramok

A jegyzetben szereplő forráskódok legtöbbször teljes, futásra kész, kis példaprogramok. Nyelvek és további más szempotok szerint az alábbi bontást használjuk:

I.4.1.1 C

```
/* C források, header fájlok halványsárgában */
```

I.4.1.2 C++

```
/* C++ források kékesszürkében */
```

I.4.1.3 Java

```
/* Java források  narancsban */
```

I.4.1.4 C#

```
/* C# források  világosszürkében */
```

I.4.1.5 XML

```
<!-- XML fájlok kékben -->
```

de ilyenben szedjük a HTML fájlokat is.

I.4.1.6 Parancssor

```
$
$ ./parancs
```

vagy képernyőkép, például egy program kimenete:

```
$ more /proc/version
Linux version 2.6.13.4 (norbi@niobe.eurosmobil.hu) (gcc version
4.0.0 20050519 (
Red Hat 4.0.0-8)) #1 Fri Oct 28 18:40:26 CEST 2005
```

de ilyenbe illesztünk majd minden további fájlt, mint például a makefile-okat.

I.4.1.7 Parancsállomány

```
#!/bin/bash
# parancsállomány világoszöldben
```

Néha kombináljuk is:

```
$ more vissza
#!/bin/bash
exit 5
$ ./vissza
$ echo $?
5
```

I.4.1.8 További szempontok

A különleges jogosultság birtokában kipróbálható kódokat, parancsokat világospirossal keretezzük:

```
/* C forrás, a root tudja futtatni
a belőle fordított programot */
```

Parancs, amit a rendszergazdának kell futtatnia:

```
#
# ./parancs
```

I.4.1.9 Példa

```
public class Ford {
  public static void main(String [] args) {
    char[] tomb = args[0].toCharArray();
    for(int i=tomb.length-1; i>=0; --i)
      System.out.print(tomb[i]);
  }
}
```

A kérdésekre a választ ezzel a programmal megfordítva, sárga dobozban pirossal szerepeltetjük?

```
.lassorip ,nabzobod agrás ,avtídrofgem ,negI
```

Ha valamit nagyon meg akarunk magyarázni, akkor számozott sorokat használunk majd:

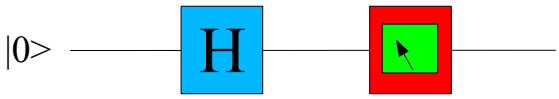
```
1.      if ((kapcsolat = accept (kapu_figyelo,
2.                       (struct sockaddr *) &kliens,
3.                       (socklen_t *) &kliensm)) == -1)
```

3. Nézzük meg, mit ír az `accept()` harmadik paraméteréről a manuál.

Vastaggal szedve is kiemelünk részeket, amikkel az adott tárgyalásban fontos momentumokra hívjuk fel a figyelmet. Illetve a laborgyakorlatokon segítjük a hallgatóságot a példaprogramok *celebrálásának*[1] követésében.

I.4.1.10 Kvantum algoritmusok

A kvantum számítógépes programoknak matematikai leírással vagy kvantum logikai hálózattal adjuk meg, például egy érme feldobása:



I.4.1.11 Tanári megjegyzések

A letanított laborokon kialakult, használt, elhangzó gondolatokat, megjegyzéseket többnyire ilyen kékkel, dőlten szedve fűzzük be az anyagba.

I.4.2 Általános jelölések

A kockás zárójelek között hivatkozásokat közlünk, a [NB] például azt jelenti, hogy a *293*. oldal *Felhasznált és ajánlott irodalom* című fejezetében meg kell keresnünk az NB bejegyzést, hogy fel tudjuk oldani a hivatkozást!

Sokszor ebben a keresésben segítjük az olvasót azzal, hogy [*Hálózati*/TB] alakban adjuk meg a hivatkozást, ami azonnal megadja, hogy az irodalomjegyzék melyik tematikus részében tudjuk feloldani a hivatkozást, a pdf-ben a dőlt és aláhúzott részre (vagy újabban magára a [] hivatkozásra, például az itt következő: [PI] hivatkozásra) kattintva azonnal ugorhatunk is erre a megfelelő részre.

I.4.3 A példaprogramok kipróbálása

A forráskódokat jelöljük ki a pdf olvasó programunkban majd illesszük be kedvenc szövegszerkesztőnkbe. Az Adobe Reader 7.0 pdf olvasót és a `vi` szövegszerkesztőt használva ez a módszer elegánsan működik. Viszont más pdf olvasókat használva előfordult például, hogy a több oldalas kijelölésbe az oldalszámot is belevette, ami a fordításnál nyilván hibát okoz. Továbbá az is hiba lehet, ha mi a jegyzetbe illesztett források tördelését tipikusan esztétikai okokból új sor jelekkel bővítjük és például egy ilyen sortöréssel ketté vágjuk egy `printf` formátum sztringjét. De ezek az esetlegesen felbukkanó szintaktikai hibák gyorsan felderíthetőek és kijavíthatóak.

E miatt az utóbbi időkben a jegyzetbe illesztett forrásokat nem formázzuk. Javasoljuk, hogy a kedves olvasó illessze be őket kedvenc (lehetőleg szintaxis kiemelést támogató) szövegszerkesztőjébe, s ott tanulmányozza őket!

1 A forrás celebrálása, azaz a program szövegének bemutatása, megbeszélése :)

II. Bevezetés

II.1 Történelem

Az Éric Lévénez-féle [OS/T] idővonalak tanulmányozása.

II.1.1 Feladat – operációs rendszer ajánlása

- Az idővonalak alapján mikor születtek az általatok eddig ismert-használt operációs rendszerek?
- Egy operációs rendszert ajánlj néhány mondatban a következő szereplők egyikének:
  - mikrovállalkozás (tetszőleges területen) vezetője
  - informatikus egyetemista ismerős (barát)
  - informatikus egyetemista ismerős (képzelt ellenség)
  - középiskolai szakkörvezető
  - gimnáziumi diák
  - a Paksi Atomerőmű

Ismételjük meg a feladat ajánlás részét programozási nyelvekre is! A megfelelő idővonalakat szintén a fent hivatkozott címen találjuk.

A feladat poénja, hogy a félév végén újra megoldjuk ezt a feladatot. A félév elején és végén beküldött megoldások összehasonlításával a hallgatóság és én is tudom érzékelni, mérni a kurzus hasznosságát.

II.2 Felhasználói szint – első találkozás a GNU/Linux rendszerrel

Azaz találkozás az igazival: nyomjunk `Alt+Ctrl+F2`-t és felhasználói nevünk, jelszavunk beírásával jelentkezzünk be! *(Természetesen kezdhetnénk a grafikus felületen történő bejelentkezéssel is, de – hallgassunk rám – kezdjünk karakteres konzolon, hogy észrevehessük, feltárulhassanak a rendszer belső értékei, már úgyis láttuk a felületet, láttuk, hogy gyönyörű szép, de a továbbiakban – az elején – csak elkalandoztatná a figyelmünket a lényegről :) Most még úgyis írok mindent a táblára, a következő gyakorlattól annyiból szerencsés a grafikus felület használata, hogy folyamatosan tudjuk követni a labor menetét a jelen pdf-ből, ez egyben azt is jelenti, hogy jegyzetelni felesleges!*

II.2.1 Bevezetés

Szimbólumok:

`$  #  .  ..  ~  /  \  &  *  ?  |`

Billentyűk (konzolok):

`Alt+Ctrl+F1-F6, Alt+<-, ->, Alt+Ctrl+F7, Shift+PageUp`

Ismerkedés a rendszerrel, kezdjük például a következő parancsokkal:

```
$ uname -a
$ last|more
$ more /proc/cpuinfo
$ more /proc/version
$ uptime
$ df -h
```

Az említett parancsokhoz a gyakorlatvezető szubjektív élményeinek bemutatása közben a `/proc/cpuinfo` BogoMIPS-ét külön is lefuttathatjuk: ez esetben ugorjunk a 277. oldal, BogoMIPS c. pontjára!

II.2.2 Alapvető parancsok

Parancsok:

`man`[1], `clear`, `uname`, `whoami`, `who`, `finger`, `chfn`, `passwd`, `w`, `last|more`, `pwd`, `ls -l`, `logout`

Néhány alapvető és további parancs tipikus használatát mutatjuk meg a 262. oldalon a *Az alapvető parancsok tipikus használata* című pontban.

Állományok:

`/`, `/proc`, `/proc/meminfo`, `/home`, `/tmp`

Billentyűk:

`Ctrl+d`

II.2.2.1 Fájlokkal kapcsolatos alapvető parancsok

A Windows ma már nagyon sikeresen testesíti meg azt a látomást, hogy bárki leül a grafikus felhasználói interfész elé és tudja használni a számítógépet! Viszont nekünk, mint informatikusoknak természetes módon nem szabad csupán ezt a királyi utat tudni járni... Ennek kapcsán lássuk, mit hoztunk a középiskolából! Ez van akinek ijesztő és olyan is, akinek unalmas lesz: mármint létrehozni az alma könyvtárat és belemásolni a dió.txt fájlt stb., ezért próbálunk közben néhány izgalmas húrt is megpendíteni. De ne aggódjunk néhány gyakorlat múlva eljutunk a 36. oldal C rendszerszint c. fejezetig, ahol is kezdjük az oprendszerek labor valódi tartalmi tárgyalását!

Parancsok:

`ls`, `ls -l`, `ls -la`, `mkdir`, `cd`, `cp`, `mv`, `rmdir`, `chmod`, `ln -s`, `du`, `find`, `grep`

```
$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
.
.
.
norbi:x:500:500:a:/home/norbi:/bin/bash
.
.
.
$ mkdir elso
$ cp /etc/passwd elso/passwd_az_etc-bol
$ ls -l elso/passwd_az_etc-bol
-rw-r--r--  1 norbi norbi 1971 dec 14 10:35 elso/passwd_az_etc-bol
$ find . -name '*etc-bol'
./elso/passwd_az_etc-bol
$ grep root `find . -name '*etc-bol'`
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
$ rm -rf[2] elso/
```

Nézzük meg a jelszó fájl mezőit! Az **x** *helyén régebben a titkosított jelszó volt, mostanában ezeket már az árnyékjelszó fájlban találjuk. Rendszergazdaként érdekes lehet*

1 A q billentyű a kilépés.

2 Így csak akkor használjuk az rm parancsot, ha kétszer is átgondoltuk, hogy mit fog törölni!

ellenőrizni, hogy a felhasználók nem használnak-e túl egyszerű jelszavakat. A gyengére választott jelszavak felderítésére szolgál például a John the Ripper program [OS/JR]. Az említett program doksiját követve, a `jelszofajl` *fájlt létrehozva és a John the Ripper programot erre ráengedve:*

```
$ more jelszofajl
.
.
.
norbi:$1$jmJCcLQX$hzvMzl3tzV0Kj4RIn7c6m0:500:500:a:/home/norbi:/bi
n/bash
.
.
.
matyi:$1$wZ3bMDHK$Xogj2CHjy4.o3MEB2nhp00:502:502::/home/matyi:/bin
/bash
erika:$1$X0FFXuk6$ikD/PljzJMBzQ05o2slyT.:503:503::/home/erika:/bin
/bash
$ ./john jelszofajl
Loaded 5 password hashes with 5 different salts (FreeBSD MD5
[32/32])
.
.
.
akire5           (erika)
matyi2006        (matyi)
4depeche         (norbi)

$ ./john -show jelszofajl
norbi:4depeche:500:500:a:/home/norbi:/bin/bash
.
.
.
matyi:matyi2006:502:502::/home/matyi:/bin/bash
erika:akire5:503:503::/home/erika:/bin/bash

5 password hashes cracked, 0 left
```

Hoppá! Magunk is kipróbálhatjuk, ha ráengedjük a fent mutatott alábbi fájlra:

```
norbi:$1$jmJCcLQX$hzvMzl3tzV0Kj4RIn7c6m0:500:500:a:/home/norbi:/bi
n/bash
matyi:$1$wZ3bMDHK$Xogj2CHjy4.o3MEB2nhp00:502:502::/home/matyi:/bin
/bash
erika:$1$X0FFXuk6$ikD/PljzJMBzQ05o2slyT.:503:503::/home/erika:/bin
/bash
```

Állományok:
`., .., /etc/passwd, ~/`

Kérdés: hogyan töröljük le a véletlenül létrehozott `*.txt` nevű fájlt?

```
$ ls
*.txt  a.txt
b.txt
$ rm \*.txt
$ ls
a.txt  b.txt
```

II.2.3 A Linux/UNIX account

Mi az utolsó mező a `/etc/passwd` jelszófájlban?

II.2.3.1 A bash parancsértelmező

Kurzor föl, parancsnév-kiegészítés a tabulátorral.

```
$ export PS1=”[\u@\h]\w> ”
```

Próbáljuk még ki, mit eredményeznek a következők: `\W, \a, \#, \t`

(Ha van accountunk egy olyan távoli unixos gépre[1], ahol megy a levelezés, akkor jelentkezünk be és futassuk ezen a távoli gépen a `pine` vagy az `elm` nevű levelező programot és dobjunk is egy levelet valakinek, mondjuk a szomszédunknak! Ha a `pine` vagy az `elm` nem akar futni, akkor a `TERM=vt100` parancs kiadása, azaz a `TERM` változó ilyen beállítása segít?)

Parancsok:

`echo, env, ., source, chmod`

Állományok:

`/bin/bash, ~/.bash_history, ~/.bash_profile` (vagy `~/.profile`), `~/.bashrc`

A `.bash_profile` fájl végére írjuk be az `echo "Udv a .bash_profile-bol"` parancsot, a `.bashrc` fájl végére írjuk be a `echo "Udv a .bashrc-bol"` parancsot, majd jelentkezzünk be újra!

II.2.3.1.1 Az elérési út beállítása

Állítsuk be a `.bash_profile` fájlban a számunkra szimpatikus `PS1` változót, illetve vizsgáljuk meg a következő példát: tegyük fel, hogy felhasználóként, azaz a saját home könyvtárunkba feltettük a legfrissebb Java fejlesztői csomagot (részletesen lásd a *268*. oldalon a *A Java fejlesztői csomag telepítése* című pontban), ekkor a `.bash_profile` fájlban érdemes beállítani, hol találhatóak a csomag parancsai (például a `java`, a `javac` stb.)

```
.
.
.
export PS1=”[\u@\h]\w> ”
export PATH=$HOME/Java/jdk1.5.0_05/bin:$PATH
.
.
.
```

Környezeti változók:

`PS1, PATH, TERM, HOME, USER, MAIL, MAILCHECK`

II.2.3.2 Feladat – felhasználói webterület

Hozzuk létre a `$HOME` könyvtárunkban a `public_html` könyvtárat és benne egy egyszerű `index.html` fájlt, majd egy böngészőből nézzük meg a következő lapot: `http://localhost/~loginnevunk`

1 A DE hallgatóinak tipikusan van ilyen.

```
$ mkdir public_html
$ chmod o+x public_html/
$ joe public_html/index.html
<html>
     <body>
          Hello, X. Y. vagyok. Ez a ~/index.html-em!
     </body>
</html>
$ chmod 644 public_html/index.html
```

(Persze a sikerhez az is kell, hogy a gépen fusson a webszerver, ha nem futna, akkor a .oldalon, a című pontban találunk segítséget.)

II.2.4 A programozó könnyűfegyverzete

Azaz egyszerű szövegszerkesztő.

II.2.4.1 joe

`Ctrl+kh` és mindent tudsz.

II.2.4.2 vi

`EscEsc` (pitty után parancsolgathatunk, különben szerkesztünk)
Ment és kilép: `EscEsc Shift zz`
!(azaz nem) ment és kilép: `EscEsc :q!`
Betűt töröl: `EscEsc x`
Sort töröl: `EscEsc dd`
Előző parancsot visszavon: `EscEsc u`
Elkezd szerkeszteni: `EscEsc i`
Egy karakterrel jobbra elkezd szerkeszteni: `EscEsc a`

II.2.4.3 emacs

Mentés: `Ctrl x s`
Keresés: `Ctrl s`
Kilépés: `Ctrl x c`

Dolgozzunk egy kicsit például a `vi`-al, nézzük meg a gép szóhosszát! Vágjuk be az alábbi szöveget például a `szohossz.c` nevű fájlba:

```
#include <stdio.h>
int
main(void)
{
  int h = 0;
  int n = 0x01;
  do
   ++h;
  while(n<<=1);
  printf("A szohossz ezen a gepen: %d bites\n", h);
  return 0;
}
```

Futtassuk a szövegszerkesztőt:

```
$ vi szohossz.c
```

Aztán `i` nyomása után beillesztjük a fájt, majd `EscEsc Shift zz` után:

```
$ gcc -o szohossz szohossz.c
$ ./szohossz
A szohossz ezen a gepen: 32 bites
```

Sokszor előfordul, hogy nem tudunk menteni a szerkesztőnkből, ilyenkor tipikusan az a hiba, hogy valami olyan helyre tévedtünk a fájlrendszerben, ahová nincs írási jogunk és innen indítottuk a szövegszerkesztőnket, ekkor például a vi-t használva:

Ment: `EscEsc w /home/hallgato/ezt_mar_kimenti.c`

Parancsok:
`indent`

Billentyűk:
`Ctrl+g, Ctrl+s, Ctrl+q`

II.3 Általános bevezetés

II.3.1 Néhány rövidítés és fogalom

Amiket érdemes tudni, mert biztosan összefutunk velük: SVr4, POSIX, BSD, RFC, ANSI, ...

II.3.1.1 Feladat – fussunk össze velük!

Lapozzunk bele néhány manuál lapba: `stdio, printf(3), socket, read(2)` stb.

```
$ man stdio
$ man 3 printf
.
.
.
```

Hol nézhetünk utána egy kívánt RFC-nek? A http://www.ietf.org/rfc lapon. Aki például a jegyzet IPv4 hálózati példáit akarja átírni IPv6-ra, annak az RFC 2553 [Hálózati/RFC2553] dokumentumot ajánljuk, ezt egyébként mi is megtesszük, a *104*. oldal, *IPv6 szerver oldal* című és a *133*. oldal *IPv6 kliens oldal* című pontokban.

II.3.2 32, 64 bites rendszerek

```
#include <stdio.h>
int main()
{
  printf("%d\n", sizeof(char) * 8);
  printf("%d\n", sizeof(short) * 8);
  printf("%d\n", sizeof(int) * 8);
  printf("%d\n", sizeof(long) * 8);
```

```
  printf("%d\n", sizeof(char*) * 8);
  return 0;
}
```

A lelkes manuál olvasó a `*8` helyett **`*CHAR_BIT`**-et ír:

```
$ man limits.h
```

Ezt kapjuk egy 32 bites rendszeren:

```
8
16
32
32
32
```

és ezt egy 64 bitesen:

```
8
16
32
64
64
```

II.3.3 Példaprogramok a kurzusban

II.3.3.1 Végtelen ciklus

Egyszer jó barát, máskor ellenség! Ki írja a legszebbet, íme egy javaslat az [*OS*/NX]-ből:

```
int
main(void)
{
  for(;;)
    ;
}
```

Billentyűk:
`Ctrl+c`

II.3.3.1.1 Folyamatokkal kapcsolatos alapvető parancsok

Futtassuk az imént látott végtelen ciklust:

```
$ gcc -o vegtelen vegtelen.c
$ ./vegtelen
```

Közben egy másik konzolon: `Alt+ <-vagy->` (avagy grafikus felületen: másik ablakban) futtassuk a `top`[1] parancsot! Figyeljük meg, hogy mennyi processzor időt visz el a `vegtelen`-nek keresztelt folyamatunk? A végtelen ciklust futtató konzolon a `Ctrl+c` billentyűkombi lenyomásával tudjuk megszakítani a folyamat futását, ekkor visszakapjuk a `$` promptot.

Futtassuk újra, majd a `Ctrl+z` billentyűkombi lenyomása után a másik ablakban figyeljük a futó `top -u felhasználói_név` parancsot!

```
$ ./vegtelen

[1]+  Stopped                 ./vegtelen
$ ps
  PID TTY          TIME CMD
 4969 pts/2    00:00:00 bash
 5064 pts/2    00:00:00 bash
 6065 pts/2    00:00:01 vegtelen
 6068 pts/2    00:00:00 ps
$ more /proc/6065/status
Name:   vegtelen
State:  T (stopped)
.
.
.
$ bg
[1]+ ./vegtelen &
$ more /proc/6065/status
Name:   vegtelen
State:  R (running)
.
.
.
$ fg
./vegtelen
```

Jól látható, hogy végül előtérbe tettük a folyamatot és a a `Ctrl+c` és a már ismert billentyükombival lőttük le, de használhatuk volna a

```
$ kill 6065
```

parancsot is. Programozói pályafutásunk során bizonyára fogunk olyan megátalkodott programokkal is találkozni, akik erre nem lesznek hajlandóak meghalni, ekkor az ellentmondást nem tűrő

```
$ kill -9 6065
```

parancsot használjuk majd! (Kapcsolódó részletek, finomságok majd a *41*. oldal, *Jelek és megszakítások* című pontjában.)

1 Magunk is írunk majd a top parancshoz hasonló programokat a későbbiekben, ilyen például a *89*. oldal, *Rendszer statisztika* című pontja.

```
$ man nohup
$ info coreutils nohup
```

Ha akkor is szeretnénk futtatni a programunkat, amikor kilépünk, akkor a `nohup` parancsot érdemes használnunk. Ez hasznos lehet, ha egy olyan szimulációs számítást futtatunk, ami például több hétig is elszámolgat.

```
$ nohup ./vegtelen&
[1] 6329
nohup: appending output to `nohup.out'
```

Lépjünk ki, majd vissza és lássuk, hogy fut-e még a programunk:

```
$ ps axu|grep vegtelen
norbi      6329 99.2  0.0    2484    260 pts/1     R     13:56    0:59
./vegtelen
```

Számításainkat érdemes `nice`-olva futtatnunk:

```
$ gcc -o v1 vegtelen.c
$ gcc -o v2 vegtelen.c
$ gcc -o v3 vegtelen.c
$ gcc -o v4 vegtelen.c
$ nice -n 3 ./v1&
[1] 6889
$ nice -n 6 ./v2&
[2] 6892
$ nice -n 9 ./v3&
[3] 6893
$ nohup nice -n 9 ./v4&
[4] 6896
nohup: appending output to `nohup.out'
$ top -u norbi
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6889 norbi     28   3  2484  260  200 R 33.8  0.0   0:27.16 v1
 6892 norbi     31   6  2488  264  200 R 23.8  0.0   0:18.51 v2
 6893 norbi     34   9  2488  264  200 R 19.9  0.0   0:12.74 v3
 6896 norbi     34   9  2488  264  200 R 19.9  0.0   0:10.13 v4
  .
  .
  .
$ killall v?
```

Régebben `nohup`-al futtattuk például a Seti@HOME klienseket. Aki nem ismeri ezt a projektet, annak a [Egyéb/SETI] lapot, illetve a [Egyéb/K] film/könyv megnézését/elolvasását ajánljuk. Fontosnak tartom megjegyezni, hogy a könyv mondanivalója jóval mélyebb, érdemes elolvasni!

Parancsok:
`top, ps, bg, fg, jobs, kill, nohup, nice, killall, sleep`

Állományok:
`/proc/processz_azonosító, nohup.out`

II.3.3.2 A példaprogramok szintjei

Milyen szinten, szinteken találkozunk majd programokkal a kurzusban?

cat

```
#include <stdio.h>
int
main(void)
{
  int c;
  while((c=getchar())!=EOF)
    putchar(c);
  return 0;
}
```

A `cat` három pillanata: a **kezdő** [*C, C++*/KR, a jelen példa, hasonló a KR 28. oldalán lévőhöz], a **haladó** [*C, C++*/KR, 159. oldalán] és a („**valódi**”) a GNU text utilities [*C, C++*/TU]-ből a `cat` forrása. Mi a kurzusban jobbára a haladó szinten fogunk majd dolgozni.

Billentyűk:
`Ctrl+d, Ctrl+z`

II.3.4 „Helló Világ!” – stdin, stdout, stderr

```
$ man stdio
```

```
#include <stdio.h>
int
main(void)
{
  printf("Hello, Vilag!\n");
  return 0;
}
```

Fordítsuk, majd futtassuk:

```
$ gcc -o hello hello.c
$ ./hello
Hello, Vilag!
```

A `main()` függvény kapcsán tanulmányozzuk a *272*. oldal *char **environ* című pontjának programját!

Miért nagyon rossz ötlet a következő parancs:

```
$gcc -o hello.c hello.c
```

Próbáljuk ki, hogy fut-e a sima

```
$ hello
```

parancs? Ha igen, ha nem, miért? Tanulmányozzuk a `PATH`-ot:

```
$ echo $PATH
```

Benne van az aktuális könyvtár?

Mit jelent a program végén a `return 0;` utasítás? (Lásd még `$ man exit`!)

.tlov dnog imalav ygoh ,latu arra kétré őzöbnölük lótállun a ,tlov nebdner nednim ygoh ,élef rezsdner sóicárepo za izlej lekkétré 0 a margorp A

Mit ír ki az

```
$ echo $?
```

Hol találjuk a Linuxunkon az `stdio.h`-t?

```
$ find /usr -name 'stdio.h'
```

Az `stdin`, `stdout`, `stderr` (ha megvan az `stdio.h`, akkor keressük is meg ezeket a difájnokat benne) és az átirányítások:

```
$ man fprintf
```

Ennek a manuál lapnak a kapcsán nézzük meg a *280*. oldal *const*-al kapcsolatos kérdését! Majd folytassuk az alábbi program kipróbálgatásával:

```
#include <stdio.h>
int
main(void)
{
  printf("Hello, Vilag!\n");
  fprintf(stdout, "STDOUT Hello\n");
  fprintf(stderr, "STDERR Hello\n");
  return 0;
}
```

Átirányítások:

```
$ ./hello >stdout.txt 2>stderr.txt
$ more stderr.txt
$ ./hello &>stdout_es_stderr.txt
$ ./hello 1>&2
$ ./hello 2> /dev/null
```

Ugyanezeket próbáljuk ki a `Hello.java` programmal is:

```
public class Hello {
  public static void main(String [] args) {
    System.out.println("Hello, STDOUT!");
    System.err.println("Hello, STDERR!");
  }
}
```

Fordítsuk, majd futtassuk:

```
$ javac Hello.java
$ java Hello
Hello, STDOUT!
Hello, STDERR!
$  java Hello >stdout.txt 2>stderr.txt
.
.
```

Szimbólumok:

> < >> >&

Állományok:

`/dev/null, /usr/include`

II.3.5 Játék a véletlennel – közben példa az átirányításokra

Próbáljuk többször is a következő parancsot:

```
$ echo $RANDOM
$ echo $RANDOM
```

Kiskoromban leültem a C+4-emmel random számokat generáltatni és közben erősen arra koncentráltam, hogy befolyásoljam… – akkor ez izgalmasnak ötletnek tűnt, miért nem az?

```
$ man 3 rand
```

ez alapján:

```
#include <stdio.h>
#include <stdlib.h>
int
main(void)
{
  int i, r;
  for(i=0; i<10; ++i)
  {
    r = 1+(int) (10.0*rand()/(RAND_MAX+1.0));
    printf("%d ", r);
  }
  printf("\n");
  return 0;
}
```

Többször futtassuk le egymás után! Mit tapasztalunk? A megoldás:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/types.h>
#include <unistd.h>
int
```

```
main(void)
{
  int i, r;
  srand(time(NULL)+getpid());
  for(i=0; i<10; ++i)
  {
    r = 1+(int) (10.0*rand()/(RAND_MAX+1.0));
    printf("%d ", r);
  }
  printf("\n");
  return 0;
}
```

(A probléma gyökere, hogy a számítógép – még a Turing gép sem tud „igazan véletlen” számokat generálni, aki ilyet akar, annak például 253. oldalon a Mit tud a kvantum számítógép, amit a Turing gép nem? című fejezetet ajánljuk.)

Érdemes beleolvasni a [*Egyéb*/KN] Véletlenszámok című fejezetébe, de ha már fellapoztuk, akkor pillantsunk bele a számrendszerekkel kapcsolatos részbe és lapozzuk fel e jegyzet További érdekes példák című fejezetét is ebben a témában, ahol többek közt megtudhatjuk, hogy miért bináris számrendszert használunk, illetve rafináltabb számrendszerekkel is találkozhatunk.

II.3.5.1 Generáljunk invertálható eloszlásból véletlen számokat

Ezt jó, ha tudjuk, szimulációknál még jól jöhet, de a lényeget tekintve továbbra is az átirányításokra koncentrálunk.

II.3.5.1.1 C

Generáljunk egy mintát standard exponenciálisból (az egyszerű módszer leírását az [Egyéb/VG, 60. oldal]-tól találjuk meg például, de e példákban a hangsúly nem ezen az algoritmuson, hanem a fordításon és az átirányításokon van.)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MINTA 10
int
main(void)
{
  int i;
  double r;
  for(i=0; i<MINTA; ++i)
  {
    r = rand()/(RAND_MAX+1.0);
    printf("%f\n", -log(r));
  }
  return 0;
}
```

Figyeljünk a fordításra, a

```
$ man log
```

alapján:

```
$ gcc -lm -o e e.c
$ ./e
0.174130
0.930433
0.244496
0.225095
0.092502
1.621757
1.092960
0.263667
1.280945
0.590645
```

Hogyan írjuk ki egy fájlba a számokat?

```
$ ./e > szamok.txt
$ more szamok.txt
0.174130
0.930433
.
.
.
```

Számoljuk ki a számok átlagát (tekintve a használt eloszlást, majd 1-et kell kapnunk)

```
#include <stdio.h>
#include <stdlib.h>
int
main(void)
{
  int i=0;
  double r, s=0.0;
  while(scanf("%lf", &r) > 0)
  {
    ++i;
    s += r;
  }
  printf("%lf\n", s/i);
  return 0;
}
```

Lássuk működés közben:

```
$ gcc -o a a.c
$ ./a < szamok.txt
0.651663
```

Emeljük most az előző generáló programban a minta `#define MINTA 10` nagyságát például `10000`-re!

```
$ gcc -lm -o e e.c
$ ./e > 10000szam.txt
$ ./a < 10000szam.txt
1.007993
```

Jónak látszik. Átlagló kis programunkat így is használhatjuk:

```
$ ./e | ./a
1.007993
```

Hogy működik ez?
Búcsúzóul nézzük meg, hogy a `scanf` függvény mikor ad vissza nullát:

```
$ man scanf
```

II.3.5.1.2 Java

Hasonló Java nyelven:

```
public class E {
  public static final int MINTA = 10;
  public static void main(String [] args) {
    for(int i=0; i<MINTA; ++i)
    System.out.println(-Math.log(Math.random()));
  }
}
```

II.3.5.1.2.a Feladat

Készítsük el és próbáljuk is ki az előző pont C-ben írt átlagolóját most Java-ban!

Eddig eljutva érkeztünk el ahhoz a ponthoz, ahol – a következőkben – kezdjük az operációs rendszerek labor tényleges tartalmi tárgyalását.

III. C rendszerszint

III.1 Folyamatok

```
$ man init
$ man proc
```

Bevezetés:
`init, getty, login, shell`

```
$ man fork
$ man getpid
```

Létrehozunk egy gyermekfolyamatot:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
int
main(void)
{
  int gyermekem_pid;
  printf("fork() elott PID: %d\n", getpid());
  if((gyermekem_pid = fork()) != -1)
  {
    if(gyermekem_pid)
    {
      printf("Szulo folyamat PID: %d, gyermekem: %d szulom %d\n",
        getpid(), gyermekem_pid, getppid());
    }
    else
    {
      sleep(1);
      printf("Gyermek folyamat PID: %d, szulom: %d\n",
        getpid(), getppid());
    }
  }
  else
  {
   printf("Sikertelen a gyermek folyamat letrehozasa\n");
   exit(-1);
  }
  printf("fork() utan PID: %d\n", getpid());
  return 0;
}
```

Futtassuk:

```
$ ./gyermek1
fork() elott PID: 19859
Szulo folyamat PID: 19859, gyermekem: 19860 szulom 19087
fork() utan PID: 19859
```

```
Gyermek folyamat PID: 19860, szulom: 1
fork() utan PID: 19860
```

Vegyük észre, hogy a szülő folyamat szülője maga a shell, nézzük meg:

```
$ ps
```

Hogy lehet, hogy a `19860` gyerek szülője nem a `19859`?

.atdagof ebkörö tamaylof tini 1 za totamaylof 06891 a ‚ttödőzejefeb bbőle rám tamaylof őlüzs 95891 a treM

Ez a szülő megvárja, amíg a gyermeke végez:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
int
main(void)
{
  int gyermekem_pid;
  int statusz;
  printf("fork() elott PID: %d\n", getpid());
  if((gyermekem_pid = fork()) != -1)
  {
    if(gyermekem_pid)
    {
      printf("Szulo folyamat PID: %d, gyermekem: %d szulom %d\n",
        getpid(), gyermekem_pid, getppid());
      printf("PID: %d gyermekfolyamat vege\n", wait(&statusz));
    }
    else
    {
      sleep(1);
      printf("Gyermek folyamat PID: %d, szulom: %d\n", getpid(),
        getppid());
    }
  }
  else
  {
   printf("Sikertelen a gyermek folyamat letrehozasa\n");
   exit(-1);
  }
  printf("fork() utan PID: %d\n", getpid());
  return 0;
}
```

Futtassuk:

```
$ ./gyermek2
fork() elott PID: 20015
Szulo folyamat PID: 20015, gyermekem: 20016 szulom 19087
Gyermek folyamat PID: 20016, szulom: 20015
```

```
fork() utan PID: 20016
PID: 20016 gyermekfolyamat vege
fork() utan PID: 20015
```

Már nem **árva** folyamat a `20016`.

Mi történik, ha a gyermek ágon elvégezzük az alábbi módosítást:

```
    else
    {
      sleep(1);
      printf("Gyermek folyamat PID: %d, szulom: %d\n",
        getpid(), getppid());
      exit(0);
    }
```

Mi történik, ha a szülő ágon elvégezzük az alábbi módosítást:

```
    if(gyermekem_pid)
    {
      printf("Szulo folyamat PID: %d, gyermekem: %d szulom %d\n",
        getpid(), gyermekem_pid, getppid());
      for(;;)
        sleep(1);
    }
```

(Ha ez utóbbi módosítást a korábbi `wait`-es verzióba tesszük, akkor a végtelen ciklust a `wait` elé tegyük. Akkor vagyunk rendben, ha futtatáskor a `top`-ban egy zombit látunk majd.)

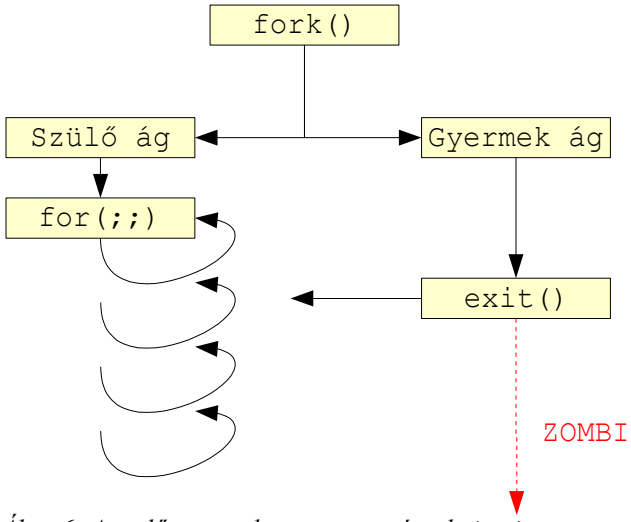
III.1.1 Zombik!

Ez történik:

```
$ top -u nbatfai
top - 13:10:55 up 175 days,  2:15,  9 users,  load average: ...
Tasks: 116 total,   1 running, 114 sleeping,   0 stopped,   1 zombie
```

Mikor lesz egy folyamat zombi?

.inzátguyn atdut men gém tze ejőlüzs a ed ,ttepélik le-tixe rám aH



Ábra 6: A szülő nem tud `wait()`-et végrehajtani...

Az iménti ábrát a *41*. oldal *Jelek és megszakítások* című részének megismerése után vessük össze a *44*. oldal *Játék zombikkal* című pontjának ábrájával! Illetve zombi témában olvassuk el a Hálózati programozás című fejezet párhuzamos szerverrel kapcsolatos „forkos” részei kapcsán felmerülő kérdéseket, lásd a *108*. oldaltól, a *Kérdések – a párhuzamos szerverről* című pontot!

Most pedig dolgozzuk fel a *281*. oldal, *Folyamatokkal kapcsolatos kérdések* című fejezetének kérdéseit, mert ezek szorosan kapcsolódnak a „forkoláshoz”.

III.1.2 Riasztás

Mit csinál a következő példa? (Próbáljuk ki! Ezzel egyben be is vezetjük a hamarosan következő, Jelek és megszakítások című részt.)

```
$ man 2 kill
```

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
int gyermekem_pid;
void riasztas_kezelo()
{
  printf("Gyermekfolyamat kilovese\n");
  kill(gyermekem_pid, SIGKILL);
}
int
main(void)
{
  int statusz;
  printf("fork() elott PID: %d\n", getpid());
  if((gyermekem_pid = fork()) != -1)
```

```
  {
    if(gyermekem_pid)
    {
      printf("Szulo folyamat PID: %d, gyermekem: %d szulom %d\n",
        getpid(), gyermekem_pid, getppid());
      signal(SIGALRM, riasztas_kezelo);
      alarm(5);
      printf("PID: %d gyermekfolyamat vege\n", wait(&statusz));
    }
    else
    {
      printf("Gyermek folyamat PID: %d, szulom: %d\n", getpid(),
        getppid());
      for(;;)
        sleep(1);
    }
  }
  else
  {
   printf("Sikertelen a gyermek folyamat letrehozasa\n");
   exit(-1);
  }
  printf("fork() utan PID: %d\n", getpid());
  return 0;
}
```

III.1.3 A `fork()` tipikus használata

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
int
main(void)
{
  int gyermekem_pid;
  int statusz;
  if((gyermekem_pid = fork()) == 0)
  {
    char *args[] = {"/bin/ls", NULL};
    execve("/bin/ls", args, NULL);
    exit(0);
  }
  else if(gyermekem_pid > 0)
  {
    wait(&statusz);
  }
  else
  {
   exit(-1);
  }
  return 0;
}
```

III.1.4 Jelek és megszakítások

Bevezetés:

```
$ man 7 signal
$ man signal
```

Próbáljuk ki az alábbi programot: a `SIGINT` (`Ctrl+c`) jelet figyelmen kívül hagyhatjuk (`SIG_IGN`), vagy kezelhetjük (`SIG_DFL`) alapértelmezés szerint:

```
#include <stdio.h>
#include <signal.h>
int
main(void)
{
  signal(SIGINT, SIG_IGN);
  sleep(5);
  signal(SIGINT, SIG_DFL);
  for(;;)
    sleep(5);
  return 0;
}
```

Definiálhatunk saját kezelőt is: `utolso_tennivalo()`

```
#include <stdio.h>
#include <signal.h>
void utolso_tennivalo(int sig)
{
  printf("Utolso tennivalo kesz, immar kilephetek\a\n");
  exit(0);
}
int
main(void)
{
  if(signal(SIGINT, utolso_tennivalo) == SIG_IGN)
    signal(SIGINT, SIG_IGN);
  for(;;)
    putchar(getchar());
  return 0;
}
```

Mondjunk erre a példára egy értelmes használati esetet!

Hogyan viszonyulna az alábbi szerkezet a példában szereplőhöz?

```
  if(signal(SIGINT, SIG_IGN) != SIG_IGN)
    signal(SIGINT, utolso_tennivalo);
```

Kapjuk el a `Ctrl+c` megszakítást:

```
#include <stdio.h>
#include <signal.h>
void ctrlc_kezelo(int sig)
{
```

```
  signal(SIGINT, ctrlc_kezelo);
  printf("Megprobaltal megallitani?\a\n");
}
int
main(void)
{
  if(signal(SIGINT, ctrlc_kezelo) == SIG_IGN)
    signal(SIGINT, SIG_IGN);
  for(;;)
    putchar(getchar());
  return 0;
}
```

Akkor mégis hogyan állítsuk meg elegánsan?

```
$ ./megszakit3
Megprobaltal megallitani?
Megprobaltal megallitani?
Megprobaltal megallitani?
Ctrl+z
[1]+  Stopped                 ./megszakit3
$ kill %1
[1]+  Stopped                 ./megszakit3
$
[1]+  Terminated              ./megszakit3
```

III.1.4.1 Nem lokális ugrások

Bevezetés:

```
$ man setjmp
$ man longjmp
```

```
#include <stdio.h>
#include <signal.h>
#include <setjmp.h>
sigjmp_buf jmpbuf;
void kezdjuk_ujra(int sig)
{
  signal(SIGINT, kezdjuk_ujra);
  printf("Megzavartal, ujra kezdjuk\a\n");
  siglongjmp(jmpbuf, 0);
}
int
main(void)
{
  if(signal(SIGINT, kezdjuk_ujra) == SIG_IGN)
    signal(SIGINT, SIG_IGN);
  sigsetjmp(jmpbuf, 1);
  for(;;)
    putchar(getchar());
  return 0;
}
```

Remek alkalmazását láthatjuk a nem lokális ugrásnak az 45. oldalon a *Crashme – az operációs rendszer robosztusságának vizsgálata* című fejezetben.

III.1.4.2 Sigaction

Bevezetés:

```
$ man sigaction
```

Korábbi `Ctrl+c` elkapó programunkat írjuk át a `sigaction`-t használva:

```
#include <stdio.h>
#include <signal.h>
void ctrlc_kezelo(int sig)
{
  printf("Megprobaltal megallitani?\a\n");
}
int
main(void)
{
  struct sigaction sa;
  sa.sa_handler = ctrlc_kezelo;
  sigemptyset(&sa.sa_mask);
  sa.sa_flags = SA_RESTART;
  sigaction(SIGINT, &sa, NULL);
  for(;;)
    putchar(getchar());
  return 0;
}
```

Jelen példa egy gyakorlati használatát találhatjuk az *123*. oldalon, a *Párhuzamos, IO multiplexeléssel* című fejezet második példájában.

III.1.4.2.1 A riasztás alkalmazása

Mit csinál a következő példa?

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
void
nem_nyomtal (int sig)
{
  printf ("\nNem nyomtal!!!\a\n");
}
int
main (void)
{
  struct sigaction sa;
  sa.sa_handler = nem_nyomtal;
  sigemptyset (&sa.sa_mask);
  sigaction (SIGALRM, &sa, NULL);
  printf ("Nyomj egy Enter-t 3 masodpercen belul!\n");
  alarm (3);
```

```
  getchar ();
  alarm (0);
  return 0;
}
```

Jelen példa egy gyakorlati használatát találhatjuk az *152*. oldalon, a *A nem megbízhatóság szimulálása* című pontban.

Remek alkalmazását láthatjuk a riasztásnak a 45. oldalon a *Crashme – az operációs rendszer robosztusságának vizsgálata* című fejezetben.

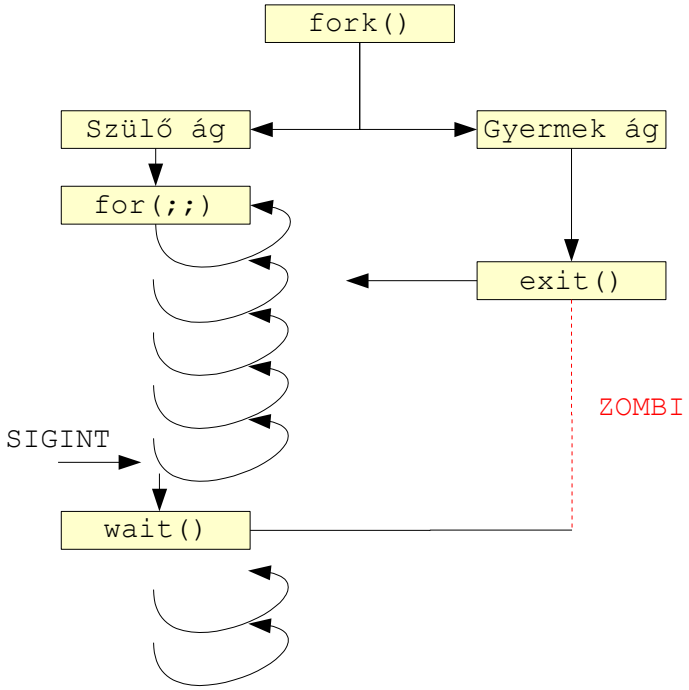
III.1.5 Játék zombikkal

Vessük össze az alábbi ábrát és programot!

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
void
styx (int sig)
{
  printf("%d\n", wait (NULL));
  fflush(stdout);
}
int
main (void)
{
  int gyermekem_pid;
  struct sigaction sa;
  sa.sa_handler = styx;
  sigemptyset (&sa.sa_mask);
  sa.sa_flags = SA_RESTART;
  sigaction (SIGINT, &sa, NULL);
  if ((gyermekem_pid = fork ()) != -1)
    {
      if (gyermekem_pid)
      {
      printf ("Szulo folyamat PID: %d, gyermekem: %d szulom %d\n",
            getpid (), gyermekem_pid, getppid ());
       for (;;)
         {
           printf ("Alszok...\n");
           fflush (stdout);
           sleep (1);
         }
      }
      else
      {
      printf ("Gyermek folyamat PID: %d, szulom: %d\n", getpid (),
            getppid ());
       exit (EXIT_SUCCESS);
      }
    }
```

```
  else
    {
      printf ("Sikertelen a gyermek folyamat letrehozasa\n");
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

Futtassuk, miközben egy másik ablakban (pl. `top`) folyamatosan figyeljünk a zombi megjelenésére! Aztán `Ctrl+C`. Elemezzük a működést a következő ábra segítségével:



Ábra 7: A Játék zombikkal programjának magyarázó rajza: a Ctrl+c megnyomásával a gyermeket a styx() kezelő kiviszi a zombi állapotból

III.1.5.1 Feladat – `SIGCHLD`

Módosítsuk úgy az iménti programot, hogy s szülő a `SIGCHLD` jelre `wait()`-eljen! Vessük össze ezt a megoldást a *108*. oldal *Kérdések – a párhuzamos szerverről* című pontjával!

III.1.6 Érdekes példák

III.1.6.1 Crashme – az operációs rendszer robosztusságának vizsgálata

A George J Carrette által írt Crashme program remek példa az előző két fejezetben megismertek gyakorlati alkalmazására, mert találunk benne megszakítás-kezelést, riasztást, nem lokális ugrást. (A program tanulmányozása előtt érdemes megnézni a Véletlen című fejezetet is.)

Letöltés:
http://packages.debian.org/stable/source/crashme

A működés vázlata:
A `badboy_loop()` függvénybeli főciklusból hívott `compute_badboy()` függvény véletlen bájtokkal tölti fel a parancssorban megadott méretű memóriaterületet, az ide mutató pointert `typedef void (*BADBOY)();` függvénymutatóra kényszeríti, amit majd egyszerűen meghív a következő lépések elvégzése után:

- `setjmp` mentést végez a nem lokális ugráshoz
- a megfelelő jelekre beállítja az `again_handler` megszakításkezelőt (ahonnan adott esetben végrehajtja majd a nem lokális ugrást)
- 10 másodperc múlvára kér egy riasztást

(Ha nem fordul le a make-re, akkor a 263. sorban az `act.sa_mask = 0;` helyett írjuk be, hogy `sigemptyset(&act.sa_mask);`!)

III.1.7 Folyamatok kommunikációja, IPC

III.1.7.1 Szemaforok

Nézzük meg a 111. oldal *A folyamatok összehangolása szemaforral* című pont példáját! Ezen az élő példán keresztül tanulmányozzuk a szemaforok használatát. Fordítsuk, futtassuk, majd a forrás alapján elemezzük az alábbi logolást: *(ugyanezt a kimenetet a hivatkozott részben is kiemeltük, de ott más részeit emeltük ki)*

```
$ ./szerver
127.0.0.1:2005
szemafor: 327688
zar elott: 2661
zar utan accept elott: 2661
zar elott: 2662
zar elott: 2663
zar elott: 2665
zar elott: 2664
     <-> 127.0.0.1:32884
zar elott: 2661
zar utan accept elott: 2662
.
.
.
```

Közben adjuk ki a következő parancsot:

```
$ ipcs
------ Shared Memory Segments --------
.
.
.
------ Semaphore Arrays --------
key        semid      owner      perms      nsems
.
.
.
0xffffffff 327688     norbi     600        1
------ Message Queues --------
.
```

```
.
.
```

A `/proc` fundamentalisták kedvéért:

```
$ more /proc/sysvipc/sem
```

A szerver lelövése után töröljük a szemafortömböt:

```
ipcrm -s 327688
```

Parancsok:
`ipc, ipcrm`

Állományok:
`/proc/sysvipc/sem`

III.1.7.2 Socketek

Lokális IPC.

```
$ man 7 unix
```

Ebben a pontban a lokális, azaz az ugyanazon a gépen futó folyamatok közötti, socketekkel megvalósított kommunikációval foglalkozunk. Ha a folyamatok különböző hosztokon vannak, akkor a 102. oldal, *Socket programozás* című fejezetét javasoljuk az érdeklődőknek. Egyébként ezt a hivatkozott fejezetet azért is ajánljuk, mert a szóban forgó két esetben a programozás gyakorlatilag ugyanaz. Olyannyira, hogy az itteni példaprogramokat a hálózati rész példáinak kis módosításával készítjük el. A szervert például a 102. oldal, *Soros, azaz iteratív* című pont példájából:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <time.h>
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int
main (void)
{
  int kapu_figyelo, kapcsolat, kliensm;
  struct sockaddr_un szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
```

```
  szerver.sun_family = AF_LOCAL;
  strncpy (szerver.sun_path, "szerver.socket",
           sizeof (szerver.sun_path));
  if ((kapu_figyelo = socket (PF_LOCAL, SOCK_STREAM, 0)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
         sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (listen (kapu_figyelo, SZERVER_SOR_MERET) == -1)
    {
      perror ("listen");
      exit (EXIT_FAILURE);
    }
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));
      if ((kapcsolat = accept (kapu_figyelo,
                      (struct sockaddr *) &kliens,
                      (socklen_t *) & kliensm)) == -1)
     {
       perror ("accept");
       exit (EXIT_FAILURE);
     }
      if (kiszolgal (kapcsolat) == -1)
     {
       perror ("kiszolgal");
     }
      close (kapcsolat);
    }
}
```

A klienst pedig a *132*. oldal, *C socket kliens* című pont példájából:

```
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#define BUFFER_MERET 256
int
main (void)
{
  int kapu, olvasva;
  struct sockaddr_un szerver;
  char buffer[BUFFER_MERET];
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sun_family = AF_LOCAL;
  strncpy (szerver.sun_path, "szerver.socket",
           sizeof (szerver.sun_path));
  if ((kapu = socket (PF_LOCAL, SOCK_STREAM, 0)) == -1)
```

```
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (connect (kapu, (struct sockaddr *) &szerver,
               sizeof (szerver)) == -1)
    {
      perror ("connect");
      exit (EXIT_FAILURE);
    }
  while ((olvasva = read (kapu, buffer, BUFFER_MERET)) > 0)
    write (1, buffer, olvasva);
  exit (EXIT_SUCCESS);
}
```

Mit mond a szerver a következő futtatáskor, ha nem töröljük a kötéskor létrehozott, alábbi socket típusú fájlt?

```
$ ls -l szerver.socket
srwxrwxr-x  1 norbi norbi 0 Oct 31 20:45 szerver.socket
```

Kifinomultabb szerkezetű szerveroldalról a már hivatkozott a 102. oldal, *Socket programozás* című pontjában olvashatunk.

A

```
$ man 7 unix
```

tanulmányozása mellett nézzük meg, mi történik, ha a szerverben és a kliensben elvégezzük az alábbi módosítást:

```
  strncpy (szerver.sun_path, "\0szerver.socket",
           sizeof (szerver.sun_path));
```

III.1.7.2.1 Feladat – ugyanez UDP-vel

Készítsük el az előző pont kliens-szerver IPC példáját, de most UDP-t használva, segítségül például a *148*. oldal *UDP szerver, kliens* című pontját megnézve!

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <sys/un.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 64
int
main (void)
{
```

```
  int kapu_figyelo, kliensm;
  char buffer[BUFFER_MERET];
  time_t t;
  struct sockaddr_un szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sun_family = AF_LOCAL;
  strncpy (szerver.sun_path, "szerver.socket",
           sizeof (szerver.sun_path));
  if ((kapu_figyelo = socket (PF_LOCAL, SOCK_DGRAM, 0)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
          sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));

      if (recvfrom (kapu_figyelo, buffer, BUFFER_MERET, 0,
                (struct sockaddr *) &kliens, (socklen_t *) & kliensm)
          < 0)
      {
       perror ("recvfrom");
       exit (EXIT_FAILURE);
      }
      t = time (NULL);
      ctime_r (&t, buffer);
      if (sendto (kapu_figyelo, buffer, strlen (buffer), 0,
             (struct sockaddr *) &kliens, (socklen_t) kliensm) < 0)
      {
       perror ("sendto");
       exit (EXIT_FAILURE);
      }
    }
}
```

A kliens:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/un.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 64
int
main (void)
{
```

```
  int kapu, olvasva, szerverm;
  struct sockaddr_un szerver, kliens;
  char buffer[BUFFER_MERET] = "Mennyi az ido?";
  memset ((void *) &szerver, 0, (szerverm = sizeof (szerver)));
  szerver.sun_family = AF_LOCAL;
  strncpy (szerver.sun_path, "szerver.socket", sizeof
(szerver.sun_path));
  if ((kapu = socket (PF_LOCAL, SOCK_DGRAM, 0)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  memset ((void *) &kliens, 0, sizeof (kliens));
  kliens.sun_family = AF_LOCAL;
  strncpy (kliens.sun_path, "kliens.socket",
           sizeof (kliens.sun_path));
  if (bind (kapu, (struct sockaddr *) &kliens, sizeof (kliens))
      == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (sendto (kapu, buffer, strlen (buffer), 0,
           (struct sockaddr *) &szerver, (socklen_t) szerverm) < 0)
    {
      perror ("sendto");
      exit (EXIT_FAILURE);
    }
  if ((olvasva = recvfrom (kapu, buffer, BUFFER_MERET, 0,
                  (struct sockaddr *) &szerver,
                  (socklen_t *) & szerverm)) < 0)
    {
      perror ("recvfrom");
      exit (EXIT_FAILURE);
    }
  printf("%s", buffer);
  exit (EXIT_SUCCESS);
}
```

Hasonlítsu össze ezt klienst a *148*. oldal *UDP szerver, kliens* című pontjának kliensével! Mi a különbség?

III.1.7.2.2 Anonim socketek

```
$ man socketpair
```

Az előző ponthoz hasonlóan most is egy daytime jellegű példát készítünk. A hálózati részben használt funkcionalítást a 40. oldal, a *A fork() tipikus használata* című pontjának példájába öntjük:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
```

```
#include <unistd.h>
#include <sys/socket.h>
#include <time.h>
#include <string.h>
#define BUFFER_MERET 256
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int
main ()
{
  int gyermekem_pid;
  int sv[2];
  if (socketpair (AF_LOCAL, SOCK_STREAM, 0, sv) == -1)
    {
      perror ("socketpair");
      exit (EXIT_FAILURE);
    }
  if ((gyermekem_pid = fork ()) == 0)
    {
      char buffer[BUFFER_MERET];
      int olvasva;
      close (sv[1]);
      olvasva = read (sv[0], buffer, BUFFER_MERET);
      write (1, buffer, olvasva);
      close (sv[0]);
    }
  else if (gyermekem_pid > 0)
    {
      close (sv[0]);
      kiszolgal (sv[1]);
      close (sv[1]);
    }
  else
    {
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

További példát találunk az anonim socketekre a következő pont összehasonlításánál.

III.1.7.2.3 Csővezetékek

```
$ man pipe
```

A példa megválasztásánál megint csak az előző ponthoz hasonlóan járunk el.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#define BUFFER_MERET 256
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int
main ()
{
  int gyermekem_pid;
  int sv[2];
  if (pipe (sv) == -1)
    {
      perror ("pipe");
      exit (EXIT_FAILURE);
    }
  if ((gyermekem_pid = fork ()) == 0)
    {
      char buffer[BUFFER_MERET];
      int olvasva;
      close (sv[1]);
      olvasva = read (sv[0], buffer, BUFFER_MERET);
      write (1, buffer, olvasva);
      close (sv[0]);
    }
  else if (gyermekem_pid > 0)
    {
      close (sv[0]);
      kiszolgal (sv[1]);
      close (sv[1]);
    }
  else
    {
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

Szinte csak a rendszerhívást írtuk át az előző példához képest, akkor miben több a `socketpair()` példa, mint a `pipe()` példa? Ebben:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/socket.h>
#include <time.h>
```

```
#include <string.h>
#define BUFFER_MERET 256
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int
main ()
{
  int gyermekem_pid;
  int sv[2];
  char buffer[BUFFER_MERET];
  int olvasva;
  if (socketpair (AF_LOCAL, SOCK_STREAM, 0, sv) == -1)
    {
      perror ("socketpair");
      exit (EXIT_FAILURE);
    }
  if ((gyermekem_pid = fork ()) == 0)
    {
      close (sv[1]);
      olvasva = read (sv[0], buffer, BUFFER_MERET);
      write (1, buffer, olvasva);
      kiszolgal (sv[0]);
      close (sv[0]);
    }
  else if (gyermekem_pid > 0)
    {
      close (sv[0]);
      kiszolgal (sv[1]);
      olvasva = read (sv[1], buffer, BUFFER_MERET);
      write (1, buffer, olvasva);
      close (sv[1]);
    }
  else
    {
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

III.1.7.2.3.a Játék az átirányítással

```
$ man dup
```

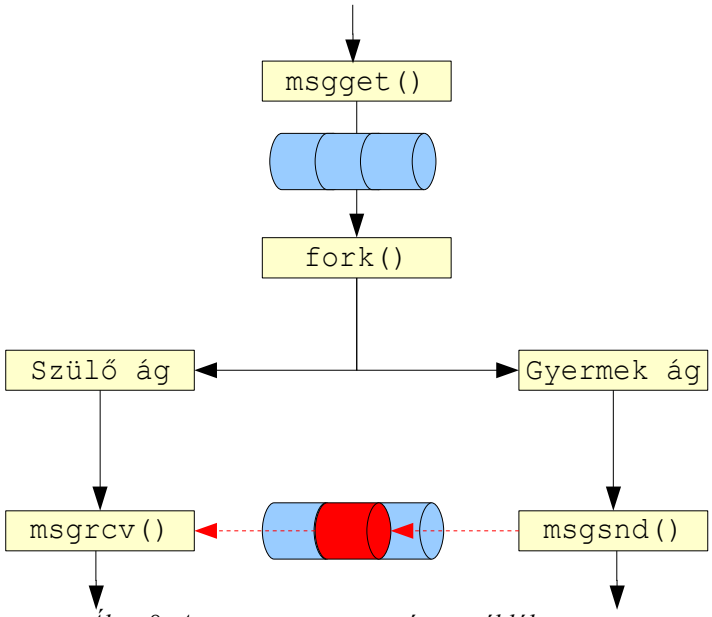
Induljunk ki a 40. oldal, a *A fork() tipikus használata* című pontjának példájából: a gyermek által indított program kimenetét juttassuk el a szülőnek:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#define BUFFER_MERET 256
int
main (void)
{
  int gyermekem_pid;
  int sv[2];
  if (pipe (sv) == -1)
    {
      perror ("pipe");
      exit (EXIT_FAILURE);
    }
  if ((gyermekem_pid = fork ()) == 0)
    {
      close (1);
      dup (sv[1]);
      char *args[] = { "/bin/ls", "-l", NULL };
      execve ("/bin/ls", args, NULL);
      exit (0);
    }
  else if (gyermekem_pid > 0)
    {
      char buffer[BUFFER_MERET];
      int olvasva;
      close (sv[1]);
      while ((olvasva = read (sv[0], buffer, BUFFER_MERET - 1)) >
0)
          write (1, buffer, olvasva);
      close (sv[0]);
    }
  else
    {
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

III.1.7.3 Üzenetsorok

Nézzük meg a `linux/msg.h`-ban [*OS*/KO] az `msgbuf` és az `msg_msg` struktúrák deklarációját! Ugyanitt nézzük meg, hogy mekkora a max üzenetméret? (`MSGMAX`)

Megintcsak a 40. oldal, a *A fork() tipikus használata* című pontjának példájának programjából indulunk ki: a gyermek megírja a szülőnek, hány az óra.



Ábra 8: Az üzenetsor megosztása a példában.

```
$ man 2 msgget
$ man 2 msgop
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define IDO_MERET 48
struct ido
{
  long mtype;
  char ido[IDO_MERET];
};
int
main ()
{
  int gyermekem_pid;
  int uzenetsor;
  if ((uzenetsor =
       msgget (ftok (".", 43), IPC_CREAT | S_IRUSR | S_IWUSR))
         == -1)
    {
      perror ("msgget");
      exit (EXIT_FAILURE);
    }
  if ((gyermekem_pid = fork ()) == 0)
    {
      struct ido ido;
```

```
      time_t t = time (NULL);
      ctime_r (&t, ido.ido);
      ido.mtype = 42;
      if (msgsnd (uzenetsor, &ido, sizeof (struct ido) -
                    sizeof (long), 0))
      {
       perror ("msgsnd");
       exit (EXIT_FAILURE);
      }
      exit (EXIT_SUCCESS);
    }
  else if (gyermekem_pid > 0)
    {
      struct ido ido;
      if (msgrcv (uzenetsor, &ido, sizeof (struct ido) -
                   sizeof (long), 42, 0) < 0)
      {
       perror ("msgrcv");
       exit (EXIT_FAILURE);
      }
      printf ("%s", ido.ido);
    }
  else
    {
      exit (EXIT_FAILURE);
    }
  if (msgctl (uzenetsor, IPC_RMID, NULL))
    {
      perror ("msgctl");
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

Állományok:
`/proc/sysvipc/msg, /proc/sys/kernel/msgmax`

III.1.7.3.1 Feladat

Írjunk két programot, az egyik írja be egy üzenetsorba az időt, a másik olvassa ki!

Ami közös, az az üzenet, helyezzük például az `uzenet.h` fájlba:

```
#define IDO_MERET 48
struct ido
{
  long mtype;
  char ido[IDO_MERET];
};
```

A berakást végző program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
```

```
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "uzenet.h"
int
main ()
{
  int uzenetsor;
  struct ido ido;
  time_t t = time (NULL);
  ctime_r (&t, ido.ido);
  ido.mtype = 42;
  if ((uzenetsor =
       msgget (ftok (".", 43), IPC_CREAT | S_IRUSR | S_IWUSR)) ==
-1)
    {
      perror ("msgget");
      exit (EXIT_FAILURE);
    }
  if (msgsnd (uzenetsor, &ido, sizeof (struct ido) - sizeof (long),
0))
    {
      perror ("msgsnd");
      exit (EXIT_FAILURE);
    }
  exit (EXIT_SUCCESS);
}
```

A kivételt végző program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "uzenet.h"
int
main ()
{
  int uzenetsor;
  struct ido ido;
  if ((uzenetsor =
       msgget (ftok (".", 43), IPC_CREAT | S_IRUSR | S_IWUSR)) ==
-1)
    {
      perror ("msgget");
      exit (EXIT_FAILURE);
    }
  if (msgrcv (uzenetsor, &ido, sizeof (struct ido) - sizeof (long),
42,
           0) < 0)
    {
```

```
      perror ("msgrcv");
      exit (EXIT_FAILURE);
    }
  printf ("%s", ido.ido);
  exit (EXIT_SUCCESS);
}
```

Mi történik, ha például háromszor futtatjuk a berakást, négyszer a kivételt, majd újra a berakást?

Ha végeztünk a kísérletezéssel, ne felejtsük el törölni az üzenetsort:

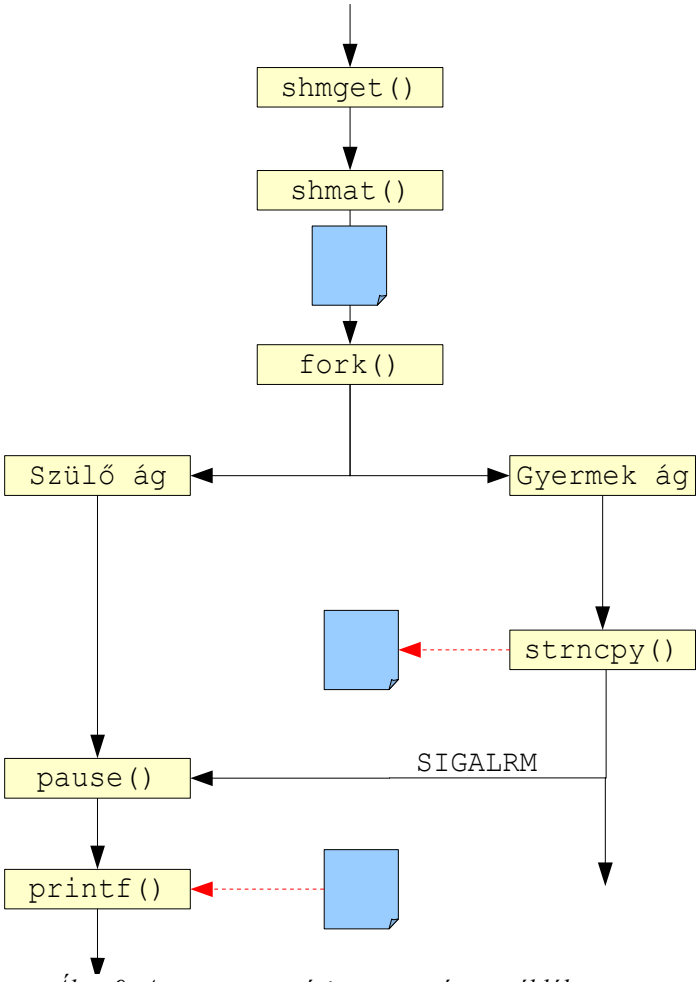
```
$ ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x2b05582b 32768      norbi      600        0            0

$ ipcrm msg 32768
resource(s) deleted
```

Mi történik, ha a két program közül az egyiket elmásoljuk egy másik könyvtárba és az egyiket innen, a másikat pedig onnan futtatva próbáljuk használni az üzenetsort?

III.1.7.4 Osztott memória

Nézzük meg az `asm/shmbuf.h`-ban [*OS*/KO] az `shmid64_ds` struktúra deklarációját!

Most is – mint az előző pont példájánál – a gyermek megírja a szülőnek, hány az óra. Megoldásként is az említett, előző példából indulunk ki.



Ábra 9: Az osztott memória megosztása a példában.

```
$ man 2 shmget
$ man 2 shmop
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#define IDO_MERET 48
void
ebreszto ()
{
  printf ("Ebreszto, Neo!\n");
}
int
main ()
{
```

```
  int gyermekem_pid;
  int osztott_memoria;
  char *osztott_memoria_terulet;
  if ((osztott_memoria =
       shmget (ftok (".", 44), 64, IPC_CREAT | S_IRUSR | S_IWUSR))
== -1)
    {
      perror ("shmget");
      exit (EXIT_FAILURE);
    }
  if ((osztott_memoria_terulet = shmat (osztott_memoria, NULL, 0))
       < 0)
    {
      perror ("shmat");
      exit (EXIT_FAILURE);
    }
  if ((gyermekem_pid = fork ()) == 0)
    {
      char buffer[IDO_MERET];
      time_t t = time (NULL);
      char *p = ctime_r (&t, buffer);
      strncpy (osztott_memoria_terulet, p, IDO_MERET);
      sleep (2);
      kill (getppid (), SIGALRM);
      if (shmdt (osztott_memoria_terulet) == -1)
      {
       perror ("shmdt");
       exit (EXIT_FAILURE);
      }
      exit (EXIT_SUCCESS);
    }
  else if (gyermekem_pid > 0)
    {
      signal (SIGALRM, ebreszto);
      pause ();
      printf ("%s", osztott_memoria_terulet);
      if (shmdt (osztott_memoria_terulet) == -1)
      {
       perror ("shmdt");
       exit (EXIT_FAILURE);
      }
    }
  else
    {
      exit (-1);
    }
  if (shmctl (osztott_memoria, IPC_RMID, NULL))
    {
      perror ("shmctl");
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

Állítsuk nagyobbra a gyermekben az alvási időt és közben:

```
$ ipcs
```

```
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
status
0x2c05582b 524290     norbi      600        64         2

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
```

Állományok:
`/proc/sysvipc/shm`

Válaszoljuk meg a *284*. oldal, *IPC-vel kapcsolatos kérdések* című fejezetének kérdéseit!

III.1.7.4.1 Feladat

Írjunk két programot, az egyik írja be az osztott memóriatartományba az időt, a másik olvassa azt ki onnan!

Az írást végző program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define IDO_MERET 48
int
main ()
{
  int osztott_memoria;
  char *osztott_memoria_terulet;
  char buffer[IDO_MERET];
  time_t t = time (NULL);
  char *p = ctime_r (&t, buffer);

  if ((osztott_memoria =
       shmget (ftok (".", 44), 64, IPC_CREAT | S_IRUSR | S_IWUSR))
== -1)
    {
      perror ("shmget");
      exit (EXIT_FAILURE);
    }
  if ((osztott_memoria_terulet = shmat (osztott_memoria, NULL, 0))
< 0)
    {
```

```
      perror ("shmat");
      exit (EXIT_FAILURE);
    }
  strncpy (osztott_memoria_terulet, p, IDO_MERET);
  if (shmdt (osztott_memoria_terulet) == -1)
    {
      perror ("shmdt");
      exit (EXIT_FAILURE);
    }
  exit (EXIT_SUCCESS);
}
```

Az olvasást végző program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define IDO_MERET 48
int
main ()
{
  int osztott_memoria;
  char buffer[IDO_MERET];
  char *osztott_memoria_terulet;
  if ((osztott_memoria =
       shmget (ftok (".", 44), 64, IPC_CREAT | S_IRUSR | S_IWUSR))
== -1)
    {
      perror ("shmget");
      exit (EXIT_FAILURE);
    }
  if ((osztott_memoria_terulet = shmat (osztott_memoria, NULL, 0))
< 0)
    {
      perror ("shmat");
      exit (EXIT_FAILURE);
    }
  strncpy (buffer, osztott_memoria_terulet, IDO_MERET);
  buffer[IDO_MERET] = '\0';
  printf ("%s", buffer);
  if (shmdt (osztott_memoria_terulet) == -1)
    {
      perror ("shmdt");
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

III.1.8 Konkurencia

III.1.8.1 POSIX szálak

```
#include <stdio.h>
#include <pthread.h>
void *szal(void *id)
{
  printf("Szal: %d, %d\n", *(int *)id, pthread_self());
  *(int *)id *= 2;
  return id;
}
int
main(void)
{
  pthread_t sz1, sz2;
  int s1 = 1, s2 = 2, *r;
  if(pthread_create(&sz1, NULL, szal, (void *)&s1))
    exit(-1);
  if(pthread_create(&sz2, NULL, szal, (void *)&s2))
    exit(-1);
  pthread_join(sz1, (void *) &r);
  printf("%d\n", *r);
  pthread_join(sz2, (void *) &r);
  printf("%d\n", *r);
  return 0;
}
```

Hogyan fordítjuk:

```
$ gcc -lpthread -o pszal1 pszal1.c
```

III.1.8.1.1 A `fork()` és pthread összehasonlítás

Hasonlítsuk össze az alábbi két programot:

`III.1.8.1.1.a fork()`

A `fork()`-osat:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>
#define FORKOK_SZAMA 1000
int
main(void)
{
  int i;
  int gyermekem_pid;
  int statusz;
  clock_t delta = clock();
  for(i=0; i<SZALAK_SZAMA; ++i)
  if((gyermekem_pid = fork()) == 0)
  {
```

```
    exit(0);
  }
  else if(gyermekem_pid > 0)
  {
    wait(&statusz);
  }
  else
  {
   printf("Hiba\n");
   exit(-1);
  }
  delta = clock() - delta;
  printf("delta: %f sec\n",
           (double)delta/CLOCKS_PER_SEC);
  return 0;
}
```

III.1.8.1.1.b pthread

És a pthread-eset:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define SZALAK_SZAMA 1000
void *szal(void *id)
{
  return id;
}
int
main(void)
{
  pthread_t sz[SZALAK_SZAMA];
  int s[SZALAK_SZAMA], *r, i;
  unsigned long delta = clock();
  for(i=0; i<SZALAK_SZAMA; ++i)
  {
    s[i] = i;
    if(pthread_create(&sz[i], NULL, szal, (void *)&s[i]))
    {
      printf("Hiba");
      exit(-1);
    }
  }
  for(i=0; i<SZALAK_SZAMA; ++i)
  {
    pthread_join(sz[i], (void *) &r);
  }
  delta = clock() - delta;
  printf("delta: %d\n", delta);
  return 0;
}
```

Ha már foglalkoztunk az idővel:

III.1.8.1.1.c CPU idő

```
man clock
man times
```

A `clock()` és `CLOCKS_PER_SEC`-et lásd az előző két példában!

Használjuk most ezt:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/times.h>
#define FORKOK_SZAMA 1000
int
main(void)
{
  int i;
  int gyermekem_pid;
  int statusz;
  struct tms tmsbuf1, tmsbuf2;
  times(&tmsbuf1);
  for(i=0; i<1000; ++i)
  if((gyermekem_pid = fork()) == 0)
  {
    exit(0);
  }
  else if(gyermekem_pid > 0)
  {
    wait(&statusz);
  }
  else
  {
   printf("Hiba\n");
   exit(-1);
  }
  times(&tmsbuf2);
  printf("%d\n", tmsbuf2.tms_utime-tmsbuf1.tms_utime
          +tmsbuf2.tms_stime-tmsbuf1.tms_stime);
  return 0;
}
```

III.1.8.1.2 Pthread-es mutex zárak

Hogy látványosabb eredményeket produkáljunk, ennek a fejezetnek a példáit egy több processzoros, az alábbi rendszerben futtatjuk. Részlet a `$ top -u nbatfai` –ból:

```
Cpu0  :   0.0% user,   0.4% system,   0.0% nice,  99.6% idle
Cpu1  :   3.2% user,   0.4% system,   0.0% nice,  96.4% idle
Cpu2  :   1.3% user,   0.7% system,   0.0% nice,  98.1% idle
Cpu3  :   3.6% user,   0.4% system,   0.0% nice,  96.0% idle
Cpu4  :   0.0% user,   1.0% system,   0.0% nice,  98.9% idle
```

```
Cpu5 :   0.7% user,   0.7% system,   0.0% nice,  98.5% idle
```

Futtassuk és hasonlítsuk össze az alábbi két programot:

III.1.8.1.2.a Zár nélkül

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define SZALAK_SZAMA 100
int szamlalo = 0;
void
var(void)
{
  int i, r = 1+(int) (10000.0*rand()/(RAND_MAX+1.0));
  for(i=0; i<r; ++i) ;
}
void *
novel_szal(void *id)
{
  int i;
  for(i=0; i<100; ++i)
  {
    printf("Szal: %d, %d\n", *(int *)id, pthread_self());
    fflush(stdout);
    var();
    szamlalo = szamlalo + 1;
  }
  return id;
}
void *
csokkent_szal(void *id)
{
  int i;
  for(i=0; i<100; ++i) {
    printf("Szal: %d, %d\n", *(int *)id, pthread_self());
    fflush(stdout);
    var();
    szamlalo = szamlalo - 1;
  }
  return id;
}
int
main(void)
{
   pthread_t sz[SZALAK_SZAMA];
   int s[SZALAK_SZAMA], *r, i;
   for(i=0; i<SZALAK_SZAMA; ++i)
   {
     s[i] = i;
     if(pthread_create(&sz[i], NULL,
       (i<SZALAK_SZAMA/2)?novel_szal:csokkent_szal,
       (void *)&s[i]))
         {
           perror("Hiba");
           exit(-1);
     }
```

```
    }
    for(i=0; i<SZALAK_SZAMA; ++i)
    {
         pthread_join(sz[i], (void *) &r);
    }
    printf("A szamlalo vegul: %d\n", szamlalo);
    return 0;
}
```

Idézet a kimenetből:

```
.
.
.
Szal: 98, 1622116
Szal: 96, 1589346
Szal: 98, 1622116
Szal: 96, 1589346
Szal: 96, 1589346
A szamlalo vegul: -2
```

a `szamlalo` változó láthatóan elromlott!

III.1.8.1.2.b Zárral

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define SZALAK_SZAMA 100
int szamlalo = 0;
pthread_mutex_t szamlalo_zar;
void
var(void)
{
  int i, r = 1+(int) (10000.0*rand()/(RAND_MAX+1.0));
  for(i=0; i<r; ++i) ;
}
void *
novel_szal(void *id)
{
  int i;
  for(i=0; i<100; ++i)
  {
    printf("Szal: %d, %d\n", *(int *)id, pthread_self());
    fflush(stdout);
    var();
    pthread_mutex_lock(&szamlalo_zar);
    szamlalo = szamlalo + 1;
    pthread_mutex_unlock(&szamlalo_zar);
  }
  return id;
}
void *
csokkent_szal(void *id)
{
  int i;
  for(i=0; i<100; ++i) {
    printf("Szal: %d, %d\n", *(int *)id, pthread_self());
```

```
    fflush(stdout);
    var();
    pthread_mutex_lock(&szamlalo_zar);
    szamlalo = szamlalo - 1;
    pthread_mutex_unlock(&szamlalo_zar);
  }
  return id;
}
int
main(void)
{
   pthread_t sz[SZALAK_SZAMA];
   int s[SZALAK_SZAMA], *r, i;
   for(i=0; i<SZALAK_SZAMA; ++i)
   {
     s[i] = i;
     if(pthread_create(&sz[i], NULL,
       (i<SZALAK_SZAMA/2)?novel_szal:csokkent_szal,
       (void *)&s[i]))
         {
           perror("Hiba");
           exit(-1);
     }
   }
   for(i=0; i<SZALAK_SZAMA; ++i)
   {
        pthread_join(sz[i], (void *) &r);
   }
   printf("A szamlalo vegul: %d\n", szamlalo);
   return 0;
}
```

Immár rendben van a kimenet:

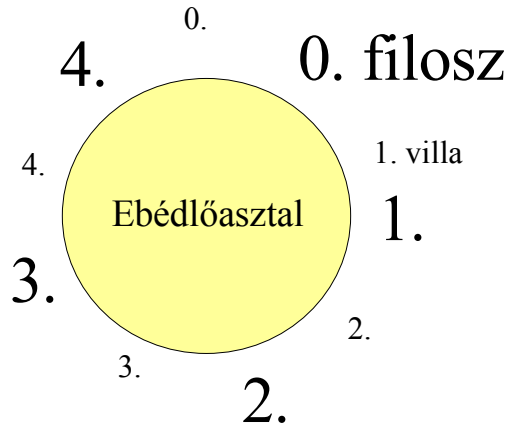
```
.
.
.
Szal: 40, 671786
Szal: 55, 917561
Szal: 97, 1605731
Szal: 40, 671786
Szal: 55, 917561
A szamlalo vegul: 0
```

IV. Konkurens programozás

IV.1 Szemaforok

IV.1.1 Programozzunk az ebédelő filozófusokkal

A feladat [*OS*/OS 93. oldal, vagy a FG előadás] elrendezését a következő ábra mutatja:



Ábra 10 Ebédelő filoszok

IV.1.1.1 C-ben szálakkal

A következőkben megismerkedünk a pthread szemaforokkal és a feltételes változókkal kapcsolatos függvényekkel.
Kezdjünk el foglalkozni a feladattal, egy filozófust egy szállal jelképezve:

```
#include <stdio.h>
#include <pthread.h>
#define FILOSZOK_SZAMA 5
void *egy_filosz(void *id)
{
  printf("%d. filosz jelen.\n", *(int *)id);
  fflush(stdout);
  return id;
}
int
main(void)
{
  pthread_t filosz_szal[FILOSZOK_SZAMA];
  int filosz_szal_arg[FILOSZOK_SZAMA];
  int i, *r;
  for(i=0; i<FILOSZOK_SZAMA; ++i) {
    filosz_szal_arg[i] = i;
    if(pthread_create(filosz_szal+i, NULL,
      egy_filosz, (void *)(filosz_szal_arg+i)))
      exit(-1);
  }
  for(i=0; i<FILOSZOK_SZAMA; ++i) {
    pthread_join(filosz_szal[i], (void *) &r);
    printf("%d\n", *r);
```

```
  }
  return 0;
}
```

Használjunk POSIX szemaforokat!

```
man sem_init
```

A villákhoz rendelt szemaforokkal jelezzük, hogy szabadok-e:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#define FILOSZOK_SZAMA 5
sem_t villa[FILOSZOK_SZAMA];
void *egy_filosz(void *id)
{
  int sorszam = *(int *)id;
  printf("%d. filosz jelen.\n", sorszam);
  fflush(stdout);
  for(;;)
  {
    sem_wait(villa+sorszam);
    sem_wait(villa+((sorszam+1) % FILOSZOK_SZAMA));
    printf("%d. filosz ebedel.\n", sorszam);
    fflush(stdout);
    sem_post(villa+sorszam);
    sem_post(villa+((sorszam+1) % FILOSZOK_SZAMA));
  }
  return id;
}
int
main(void)
{
  pthread_t filosz_szal[FILOSZOK_SZAMA];
  int filosz_szal_arg[FILOSZOK_SZAMA];
  int i, *r;
  for(i=0; i<FILOSZOK_SZAMA; ++i)
      sem_init(villa+i, 0, 1);
  for(i=0; i<FILOSZOK_SZAMA; ++i){
    filosz_szal_arg[i] = i;
    if(pthread_create(filosz_szal+i, NULL,
      egy_filosz, (void *)(filosz_szal_arg+i)))
      exit(-1);
  }
  for(i=0; i<FILOSZOK_SZAMA; ++i) {
    pthread_join(filosz_szal[i], (void *) &r);
    printf("%d\n", *r);
  }
  for(i=0; i<FILOSZOK_SZAMA; ++i)
    sem_destroy(villa+i);
  return 0;
}
```

A programmal valami alapvető baj van, mert ha fordítjuk, futtatjuk:

```
$ gcc -lpthread -o filo filo2f.c
.
.
.
3. filosz ebedel.
2. filosz ebedel.
1. filosz ebedel.
0. filosz ebedel.
0. filosz ebedel.
4. filosz ebedel.
```

egy darabig elfutkározik, de aztán lemerevedik és többé nem ezzel a programmal már nem esznek a filoszok! Mi okozza a problémát?
Mi történik, ha mind az öt filosz megszerzi az első villája szemaforját? Tapasztaljuk is meg ezt a szituációt:

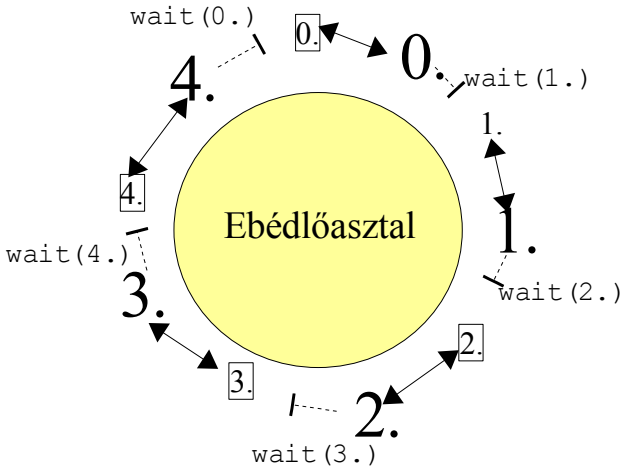
```
    sem_wait(villa+sorszam);
    printf("%d. filosz a 2. sem elott.\n", sorszam);
    fflush(stdout);
    sem_wait(villa+((sorszam+1) % FILOSZOK_SZAMA));
```

Lássuk:

```
.
.
.
2. filosz 2 sem elott.
4. filosz 2 sem elott.
1. filosz 2 sem elott.
3. filosz 2 sem elott.
0. filosz 2 sem elott.
```

Sikerült elkapnunk egy pompás kimenetet (persze az is előfordulhat, hogy az öt
„X. filosz 2 sem elott.”
nem ilyen szépen a végén sorakozik, akkor a hiányzókat följebb találjuk). Rajzoljuk most le, amit ez a kimenet mutat:

IV.1.1.1.1.a Holtpont



Ábra 11 Ebédelő filoszok holtpont

(Egy másik kérdés, hogy miért nem dolgozzuk be a sem_init-et tartalmazó ciklust a már meglévőbe?)

Próbálkozzunk a probléma megoldásával:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#define FILOSZOK_SZAMA 5
sem_t villa[FILOSZOK_SZAMA];
pthread_mutex_t villak_zar;
void *egy_filosz(void *id)
{
  int sorszam = *(int *)id;
  printf("%d. filosz jelen.\n", sorszam);
  fflush(stdout);
  for(;;){
    pthread_mutex_lock(&villak_zar);
    sem_wait(villa+sorszam);
    sem_wait(villa+((sorszam+1) % FILOSZOK_SZAMA));
    pthread_mutex_unlock(&villak_zar);
    printf("%d. filosz ebedel.\n", sorszam);
    fflush(stdout);
    sem_post(villa+sorszam);
    sem_post(villa+((sorszam+1) % FILOSZOK_SZAMA));
  }
  return id;
}
int
main(void)
{
  pthread_t filosz_szal[FILOSZOK_SZAMA];
  int filosz_szal_arg[FILOSZOK_SZAMA];
  int i, *r;
  for(i=0; i<FILOSZOK_SZAMA; ++i)
    sem_init(villa+i, 0, 1);
  for(i=0; i<FILOSZOK_SZAMA; ++i){
    filosz_szal_arg[i] = i;
    if(pthread_create(filosz_szal+i, NULL,
      egy_filosz, (void *)(filosz_szal_arg+i)))
      exit(-1);
  }
  for(i=0; i<FILOSZOK_SZAMA; ++i) {
    pthread_join(filosz_szal[i], (void *) &r);
    printf("%d\n", *r);
  }
  for(i=0; i<FILOSZOK_SZAMA; ++i)
    sem_destroy(villa+i);
  return 0;
}
```

Immár vidáman elfutkározik a program:

```
$ gcc -lpthread -o filo filo2f.c
.
.
.
```

```
0. filosz ebedel.
4. filosz ebedel.
2. filosz ebedel.
3. filosz ebedel.
1. filosz ebedel.
0. filosz ebedel.
4. filosz ebedel.
2. filosz ebedel.
0. filosz ebedel.
3. filosz ebedel.
.
.
.
```

Elemezzük ezt az iménti megoldást! Mi a hibája? Nézzük meg a bevezető ábrát és folytassik az alábbi gondolatmenetet! *„Tegyük fel, hogy a 0. filosz ebédel és közben az 1. filosz belép a mutex-el védett kritikus szakaszba, de a 0. filosz sokáig ebédel, így a 1. villára sokat kell várni... pedig közben például a 2. filosz ebédelhetne, (persze ha az 1. és 3. éppen nem ebédel.)”.* Tapasztaljuk is meg ezt a szituációt, tartson az ebéd 10 másodpercig és írjuk ki, ha egy filosz be/kilépett a kritikus szakaszba/ból:

```
void *egy_filosz(void *id)
{
  int sorszam = *(int *)id;
  for(;;){
    pthread_mutex_lock(&villak_zar);
    printf("%d. filosz belepett a kritikus szakaszba.\n",
      sorszam);
    fflush(stdout);
    sem_wait(villa+sorszam);
    sem_wait(villa+((sorszam+1) % FILOSZOK_SZAMA));
    pthread_mutex_unlock(&villak_zar);
    printf("%d. filosz kilepett a kritikus szakaszbol.\n",
      sorszam);
    fflush(stdout);
    printf("%d. filosz ebedel...\n", sorszam);
    fflush(stdout);
    sleep(10);
    printf("%d. filosz megebedelt.\n", sorszam);
    fflush(stdout);
    sem_post(villa+sorszam);
    sem_post(villa+((sorszam+1) % FILOSZOK_SZAMA));
  }
  return id;
}
```

és valóban tapasztaljuk is a fent elgondolt szituációt:

```
0. filosz belepett a kritikus szakaszba.
0. filosz kilepett a kritikus szakaszbol.
0. filosz ebedel...
1. filosz belepett a kritikus szakaszba.

Itt telik a 10 másodperc...

0. filosz megebedelt.
1. filosz kilepett a kritikus szakaszbol.
```

```
1. filosz ebedel...
2. filosz belepett a kritikus szakaszba.

1. filosz megebedelt.
2. filosz kilepett a kritikus szakaszbol.
2. filosz ebedel...
3. filosz belepett a kritikus szakaszba.
.
.
```

Próbáljuk ezt a problémát megoldani a `sem_trywait()` használatával:

```
man sem_trywait
```

```
void *egy_filosz(void *id)
{
  int sorszam = *(int *)id;
  int nincsmeg_ket_villa = 0;
  for(;;){
  do {
      pthread_mutex_lock(&villak_zar);
      nincsmeg_ket_villa = sem_trywait(villa+sorszam);
      if(nincsmeg_ket_villa == 0)
        {
          nincsmeg_ket_villa = sem_trywait(villa+((sorszam+1)
                                          % FILOSZOK_SZAMA));
          if(nincsmeg_ket_villa != 0)
             sem_post(villa+sorszam);
        }
      pthread_mutex_unlock(&villak_zar);
    } while (nincsmeg_ket_villa);
    printf("%d. filosz ebedel...\n", sorszam);
    fflush(stdout);
    sleep(10);
    printf("%d. filosz megebedelt.\n", sorszam);
    fflush(stdout);
    sem_post(villa+sorszam);
    sem_post(villa+((sorszam+1) % FILOSZOK_SZAMA));
  }
  return id;
}
```

Immár egyik filosz sem birtokolja fölöslegesen a kritikus szakasz zárját, de a mellett, hogy két szálunk folyamatosan eszik, a másik három próbálkozásai felemésztik az összes prociidőt.

```
.
.
.
2. filosz megebedelt.
2. filosz ebedel...
4. filosz megebedelt.
4. filosz ebedel...

2. filosz megebedelt.
1. filosz ebedel...
4. filosz megebedelt.
4. filosz ebedel...
```

```
1. filosz megebedelt.
1. filosz ebedel...
4. filosz megebedelt.
4. filosz ebedel...
.
.
.
```

Gondolkodjunk egy proci kímélő megoldáson! Feltételes változókkal:

```
man pthread_cond_wait
```

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#define FILOSZOK_SZAMA 5
sem_t villa[FILOSZOK_SZAMA];
pthread_mutex_t villak_zar = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t villak_letetele = PTHREAD_COND_INITIALIZER;
void *egy_filosz(void *id)
{
  int sorszam = *(int *)id, r;
  int nincsmeg_ket_villa = 0;
  for(;;){
  do {
      pthread_mutex_lock(&villak_zar);
      nincsmeg_ket_villa = sem_trywait(villa+sorszam);
      if(nincsmeg_ket_villa == 0)
      {
        nincsmeg_ket_villa = sem_trywait(villa+((sorszam+1)
                                          % FILOSZOK_SZAMA));
        if(nincsmeg_ket_villa != 0)
           sem_post(villa+sorszam);
      }
      if(nincsmeg_ket_villa)
      {
        printf("%d. filosz villara var...\n", sorszam);
        fflush(stdout);
        pthread_cond_wait(&villak_letetele, &villak_zar);
      }
      pthread_mutex_unlock(&villak_zar);
    } while (nincsmeg_ket_villa);
    r = 1+(int) (5.0*rand()/(RAND_MAX+1.0));
    printf("%d. filosz ebedel... %d secig\n", sorszam, r);
    fflush(stdout);
    sleep(r);
    printf("%d. filosz megebedelt.\n", sorszam);
    fflush(stdout);
    sem_post(villa+sorszam);
    sem_post(villa+((sorszam+1) % FILOSZOK_SZAMA));
    pthread_mutex_lock(&villak_zar);
    pthread_cond_broadcast(&villak_letetele);
    pthread_mutex_unlock(&villak_zar);
    r = 1+(int) (5.0*rand()/(RAND_MAX+1.0));
    printf("%d. filosz gondolkodik... %d secig\n", sorszam, r);
    fflush(stdout);
    sleep(r);
```

```
    printf("%d. filosz megehezett.\n", sorszam);
    fflush(stdout);
  }
  return id;
}
int
main(void)
{
  pthread_t filosz_szal[FILOSZOK_SZAMA];
  int filosz_szal_arg[FILOSZOK_SZAMA];
  int i, *r;
  for(i=0; i<FILOSZOK_SZAMA; ++i)
    sem_init(villa+i, 0, 1);
  for(i=0; i<FILOSZOK_SZAMA; ++i){
    filosz_szal_arg[i] = i;
    if(pthread_create(filosz_szal+i, NULL,
      egy_filosz, (void *)(filosz_szal_arg+i)))
      exit(-1);
  }
  for(i=0; i<FILOSZOK_SZAMA; ++i) {
    pthread_join(filosz_szal[i], (void *) &r);
    printf("%d\n", *r);
  }
  for(i=0; i<FILOSZOK_SZAMA; ++i)
    sem_destroy(villa+i);
  pthread_mutex_destroy(&villak_zar);
  pthread_cond_destroy(&villak_letetele);
  return 0;
}
```

Pillantsunk bele a kimenetbe:

```
.
.
.
0. filosz ebedel... 5 secig
1. filosz villara var...
2. filosz ebedel... 2 secig
3. filosz villara var...
4. filosz villara var...
2. filosz megebedelt.
2. filosz gondolkodik... 4 secig
1. filosz villara var...
3. filosz ebedel... 4 secig
4. filosz villara var...
0. filosz megebedelt.
0. filosz gondolkodik... 5 secig
4. filosz villara var...
1. filosz ebedel... 1 secig
.
.
.
```

[*OS*/OS 95. o]-on találunk egy olyan megoldást, amit a most megismert eszközökkel már könnyen be tudunk C-ben programozni.

IV.1.1.2 C-ben folyamatokkal

IV.1.1.3 Java-ban

Írjuk át az előző fejezet példáját Java nyelvre:

```
import java.util.concurrent.Semaphore;
class Asztal {
  int hanySzemelyes;
  Semaphore villak[];
  public Asztal(int hanySzemelyes) {
    this.hanySzemelyes = hanySzemelyes;
    villak = new Semaphore[hanySzemelyes];
    for(int i=0; i<hanySzemelyes; ++i)
      villak[i] = new Semaphore(1, true);
  }
  public synchronized void villakFel(int ki) {
    boolean nincsMegKetVilla = false;
    do {
      nincsMegKetVilla = villak[ki].tryAcquire();
      if(nincsMegKetVilla) {
        nincsMegKetVilla =
          villak[(ki+1)%hanySzemelyes].tryAcquire();
        if(!nincsMegKetVilla)
          villak[ki].release();
      }
      if(!nincsMegKetVilla)
        try {
          System.out.println(ki+". wait()");
          System.out.flush();
          wait();
          System.out.println(ki+". notify()");
          System.out.flush();
        } catch(InterruptedException e){};
    } while(!nincsMegKetVilla);
  }
  public synchronized void villakLe(int ki) {
    villak[ki].release();
    villak[(ki+1)%hanySzemelyes].release();
    notifyAll();
  }
}
class Filosz extends Thread {
  int sorszam;
  Asztal asztal;
  public Filosz(int sorszam, Asztal asztal) {
    this.asztal = asztal;
    this.sorszam = sorszam;
  }
  public void run() {
    for(;;) {
      asztal.villakFel(sorszam);
      System.out.println(sorszam+". ebedel...");
      System.out.flush();
      try {sleep(5000);}
      catch(InterruptedException e){};
      asztal.villakLe(sorszam);
      System.out.println(sorszam+". gondolkodik...");
      System.out.flush();
      try {sleep(5000);}
```

```
      catch(InterruptedException e){};
    }
  }
}
public class EbedeloFiloszok {
  public static void main(String args[]) {
    Asztal asztal = new Asztal(5);
    Filosz [] filoszok = {
      new Filosz(0, asztal),
      new Filosz(1, asztal),
      new Filosz(2, asztal),
      new Filosz(3, asztal),
      new Filosz(4, asztal),
    };
    for(int i=0; i<filoszok.length; ++i)
      filoszok[i].start();
  }
}
```

IV.1.1.4 C#

Ahogy az előző fejezetben az azt megelőző C programot írtuk meg Java nyelven, most írjuk meg C# nyelven:

```
using System;
using System.Threading;
class Asztal
{
  int hanySzemelyes;
  Semaphore [] villak;
  public Asztal(int hanySzemelyes)
  {
    this.hanySzemelyes = hanySzemelyes;
    villak = new Semaphore[hanySzemelyes];
    for(int i=0; i<hanySzemelyes; ++i)
      villak[i] = new Semaphore(1, 1);
  }
  public void villakFel(int ki)
  {
    Monitor.Enter(this);
    bool nincsMegKetVilla = false;
    do {
      nincsMegKetVilla = villak[ki].WaitOne(0, true);
      if(nincsMegKetVilla)
      {
        nincsMegKetVilla =
          villak[(ki+1)%hanySzemelyes].WaitOne(0, true);
        if(!nincsMegKetVilla)
          villak[ki].Release();
      }
      if(!nincsMegKetVilla)
      {
        Monitor.Wait(this);
      }
    } while(!nincsMegKetVilla);
```

```
    Monitor.Exit(this);
  }
  public void villakLe(int ki)
  {
    Monitor.Enter(this);
    villak[ki].Release();
    villak[(ki+1)%hanySzemelyes].Release();
    Monitor.PulseAll(this);
    Monitor.Exit(this);
  }
}
class Filosz
{
  int sorszam;
  Asztal asztal;
  public Filosz(int sorszam, Asztal asztal)
  {
    this.asztal = asztal;
    this.sorszam = sorszam;
  }
  public void run()
  {
    for(;;)
    {
      asztal.villakFel(sorszam);
      Console.WriteLine(sorszam+". ebedel...");
      Thread.Sleep(5000);
      asztal.villakLe(sorszam);
      Console.WriteLine(sorszam+". gondolkodik...");
      Thread.Sleep(5000);
    }
  }
}
class EbedeloFiloszok
{
  static void Main() {
    Asztal asztal = new Asztal(5);
    Filosz [] filosz = {
      new Filosz(0, asztal),
      new Filosz(1, asztal),
      new Filosz(2, asztal),
      new Filosz(3, asztal),
      new Filosz(4, asztal),
    };
    ThreadStart [] threadStart = new ThreadStart[filosz.Length];
    Thread [] filoszok = new Thread[filosz.Length];
    for(int i=0; i<filoszok.Length; ++i)
    {
      threadStart[i] = new ThreadStart(filosz[i].run);
      filoszok[i] = new Thread(threadStart[i]);
      filoszok[i].Start();
    }
  }
}
```

V. Fájlrendszer

V.1 Fájlok és könyvtárak

V.1.1 Infók a fájlokról

```
man 2 stat
```

Figyeljük meg a `stat` struktúra `st_size` tagját!

```
#include <stdio.h>
#include <sys/stat.h>
long int meret(char *nev)
{
  struct stat s;
  if(stat(nev, &s) == -1)
     return -1;
  return s.st_size; // man stat: off_t st_size
}
int
main(int argc, char *argv[])
{
  while(argc--)
    printf("%s : %d\n", argv[argc], meret(argv[argc]));
  return 0;
}
```

Fordítsuk, futtassuk!

```
$ gcc -o meret meret.c
$ ls -l
total 24
-rwxrwxr-x  1 norbi norbi 7209 Oct 30 09:21 meret
-rw-rw-r--  1 norbi norbi  306 Oct 30 09:16 meret.c
lrwxrwxrwx  1 norbi norbi    7 Oct 30 09:19 meret.c.forrasa -> meret.c
$ ./meret meret.c meret.c.forrasa nincs /etc/passwd /boot/vmlinuz-2.6.13.4
/boot/vmlinuz-2.6.13.4 : 1943304
/etc/passwd : 1816
nincs : -1
meret.c.forrasa : 306
meret.c : 306
./meret : 7209
```

V.1.2 Könyvtárak

```
man readdir
```

Figyeljük meg a `dirent` struktúra `d_name` tagját!

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>
```

```
int
main(int argc, char *argv[])
{
  DIR *dirp;
  struct dirent *direntp;
  if(dirp = opendir("."))
  {
    while(direntp = readdir(dirp))
      printf("%s\n", direntp->d_name);
  }
  else
  {
    perror("opendir");
    exit(EXIT_FAILURE);
  }
  closedir(dirp);
  exit(EXIT_SUCCESS);
}
```

(Ezt a példát felhasználjuk még a *89*. oldal *Rendszer statisztika* című pontja alatti második programban is.)

V.1.2.1 Állománykezelés: kiírás, beolvasás

A következő **4** (C, C++, Java, C#) **x 2** (szöveges, bináris) programban a gyökkettőt kiírjuk/beolvassuk egy fájlból/fájlba.

V.1.2.1.1 C

```
#include <stdio.h>
#include <math.h>
int
main()
{
  FILE *gyok2f;
  double gy2 = sqrt(2.0);
  gyok2f = fopen("GyokKetto.txt", "w");
  fprintf(gyok2f, "%lf", gy2);
  fclose(gyok2f);
  return 0;
}
```

Fordítsuk, futtassuk, majd nézzünk bele a létrehozott fájlba:

```
gcc -lm -o gyokki gyokki.c
$ ./gyokki
$ more GyokKetto.txt
1.414214
```

Azaz közvetlen emberi fogyasztásra alkalmas az **`1.414214`**.
Most visszaolvassuk:

```
#include <stdio.h>
int
main()
```

```
{
  FILE *gyok2f;
  double gy2;
  gyok2f = fopen("GyokKetto.txt", "r");
  fscanf(gyok2f, "%lf", &gy2);
  fclose(gyok2f);
  printf("%lf\n", gy2);
  return 0;
}
```

Ezt is próbáljuk ki:

```
$ gcc -o gyokbe gyokbe.c
$ ./gyokbe
1.414214
```

Dolgozzunk most alacsony szinten:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <math.h>
int
main()
{
  int gyok2f;
  double gy2 = sqrt(2.0);
  gyok2f = open("GyokKetto", O_CREAT|O_WRONLY,
                  S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
  write(gyok2f, (void *)&gy2, sizeof(double));
  close(gyok2f);
  return 0;
}
```

Fordítsuk, futtassuk, majd nézzünk bele a létrehozott fájlba:

```
$ gcc -lm -o gyokkib gyokkib.c
$ ./gyokkib
$ more GyokKetto
Íf ö?
```

Jól látható, hogy közvetlen emberi fogyasztásra nem alkalmas az **`Íf ö?`**.
Figyeljük meg, milyen jogokkal jött létre a `GyokKetto` nevű fájl?

V.1.2.1.1.a Feladat – ---------

Ennek kapcsán hozzunk létre egy olyan fájlt, amire

```
$ ls -l olyan_fajl
---------- 1 tulajdonos csoport meret datum olyan_fajl
```

Térjünk vissza az eredeti példához, olvassuk most vissza a binárisan kiírt gyökkettőt!

```
#include <stdio.h>
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int
main()
{
  int gyok2f;
  double gy2;
  gyok2f = open("GyokKetto", O_RDONLY);
  read(gyok2f, (void *)&gy2, sizeof(double));
  close(gyok2f);
  printf("%lf\n", gy2);
  return 0;
}
```

Sikerül visszaolvasni?

```
$ gcc -o gyokbeb gyokbeb.c
$ ./gyokbeb
1.414214
```

V.1.2.1.2 C++

```
#include <iostream>
#include <fstream>
#include <cmath>
int
main()
{
  const double gy2 = sqrt(2.0);
  std::fstream gyok2f("GyokKetto.txt", std::ios_base::out);
  gyok2f << gy2;
  gyok2f.close();
  return 0;
}
```

Fordítsuk, futtassuk, majd nézzünk bele a létrehozott fájlba:

```
$ g++ gyokki.cpp -o gyokki -lm
$ ./gyokki
$ more GyokKetto.txt
1.41421
```

Azaz megint csak közvetlen emberi fogyasztásra alkalmas az **1.414214**.
Olvassuk vissza:

```
#include <iostream>
#include <fstream>
int
main()
{
  double gy2;
  std::fstream gyok2f("GyokKetto.txt", std::ios_base::in);
  gyok2f >> gy2;
```

```
  gyok2f.close();
  std::cout << gy2 << std::endl;
  return 0;
}
```

Lássuk:

```
$ g++ gyokbe.cpp -o gyokbe
$ ./gyokbe
1.41421
```

Dolgozzunk binárisan:

```
#include <iostream>
#include <fstream>
#include <cmath>
int
main()
{
  const double gy2 = sqrt(2.0);
  std::fstream gyok2f("GyokKetto", std::ios_base::out);
  gyok2f.write((char *)&gy2, sizeof(double));
  gyok2f.close();
  return 0;
}
```

Teszteljük:

```
$ g++ gyokkib.cpp -o gyokkib -lm
$ ./gyokkib
$ more GyokKetto
Íf ö?
```

Azaz megint csak fogyasztásra alkalmatlan, de a beolvasó programunk meg tudja emészteni?

```
#include <iostream>
#include <fstream>
int
main()
{
  double gy2;
  std::fstream gyok2f("GyokKetto", std::ios_base::in);
  gyok2f.read((char *)&gy2, sizeof(double));
  gyok2f.close();
  std::cout << gy2 << std::endl;
  return 0;
}
```

Teszteljük:

```
$ g++ gyokbeb.cpp -o gyokbeb
$ ./gyokbeb
1.41421
```

V.1.2.1.3 Java

```
public class GyokKi {
  public static void main(String [] args) {
    double gy2 = Math.sqrt(2.0);
      try {
        java.io.PrintStream ps = new java.io.PrintStream(
          new java.io.FileOutputStream("GyokKetto.txt"));
        ps.print(gy2);
        ps.close();
     } catch(Exception e) {
      }
  }
}
```

Fordítsuk, futtassuk, majd nézzünk bele a létrehozott fájlba:

```
$ javac GyokKi.java
$ java GyokKi
$ more GyokKetto.txt
1.4142135623730951
```

Azaz megint csak közvetlen emberi fogyasztásra alkalmas az **`1.4142135623730951`**. Olvassuk vissza:

```
public class GyokBe {
  public static void main(String [] args) {
      double gy2 = 0.0;
      try {
        java.io.BufferedReader bf = new java.io.BufferedReader(
          new java.io.FileReader("GyokKetto.txt"));
        gy2 = Double.parseDouble(bf.readLine());
        bf.close();
      } catch(Exception e) {
      }
      System.out.println(gy2);
  }
}
```

Sikerül?

```
$ javac GyokBe.java
$ java GyokBe
1.4142135623730951
```

Jöjjön a szokásos második kör:

```
public class GyokKib {
  public static void main(String [] args) {
    double gy2 = Math.sqrt(2.0);
      try {
        java.io.DataOutputStream dos =
           new java.io.DataOutputStream(
              new java.io.FileOutputStream("GyokKetto"));
        dos.writeDouble(gy2);
        dos.close();
```

```
      } catch(Exception e) {
      }
  }
}
```

Sikerül?

```
$ javac GyokKib.java
$ java GyokKib
$ more GyokKetto
?ö ;Í
```

Megint csak fogyasztásra alkalmatlan, de a beolvasó programunk meg tudja emészteni?

```
public class GyokBeb {
  public static void main(String [] args) {
      double gy2 = 0.0;
      try {
        java.io.DataInputStream dis =
           new java.io.DataInputStream(
             new java.io.FileInputStream("GyokKetto"));
        gy2 = dis.readDouble();
        dis.close();
      } catch(Exception e) {
      }
      System.out.println(gy2);
  }
}
```

Nos:

```
$ javac GyokBeb.java
$ java GyokBeb
1.4142135623730951
```

Simán!

V.1.2.1.4 C#

```
class GyokKi
{
  public static void Main()
  {
      double gy2 = System.Math.Sqrt(2.0);
      using (System.IO.StreamWriter sw =
        new System.IO.StreamWriter("GyokKetto.txt"))
      {
        sw.WriteLine(gy2);
      }
  }
}
```

Fordítsuk, futtassuk, majd nézzünk bele a létrehozott fájlba:

```
> csc GyokKi.cs
> GyokKi
```

```
> more GyokKetto.txt
1.4142135623731
```

Azaz megint csak közvetlen emberi fogyasztásra alkalmas az **`1.4142135623731`**.
Olvassuk vissza:

```
class GyokBe
{
  public static void Main()
  {
      double gy2;
      using (System.IO.StreamReader sr =
        new System.IO.StreamReader("GyokKetto.txt"))
      {
        gy2 = System.Convert.ToDouble(sr.ReadLine());
      }
      System.Console.WriteLine(gy2);
  }
}
```

Sikerül?

```
> csc GyokBe.cs
> GyokBe
1.4142135623731
```

Jöjjön a szokásos második kör:

```
class GyokKib
{
  public static void Main()
  {
      double gy2 = System.Math.Sqrt(2.0);
      using (System.IO.BinaryWriter bw =
        new System.IO.BinaryWriter(
          new System.IO.FileStream("GyokKetto",
                                   System.IO.FileMode.Create)))
      {
        bw.Write(gy2);
      }
  }
}
```

Lássuk!

```
> csc GyokKib.cs
> GyokKib
> more GyokKetto
Ífž ö?
```

Megint csak fogyasztásra alkalmatlan, de a beolvasó programunk meg tudja emészteni?

```
class GyokBeb
{
  public static void Main()
```

```
  {
      double gy2;
      using (System.IO.BinaryReader br =
        new System.IO.BinaryReader(
          new System.IO.FileStream("GyokKetto",
                                   System.IO.FileMode.Open)))
      {
        gy2 = br.ReadDouble();
      }
      System.Console.WriteLine(gy2);
  }
}
```

Sikerül?

```
> csc GyokBeb.cs
> GyokBeb
1.4142135623731
```

Naná!

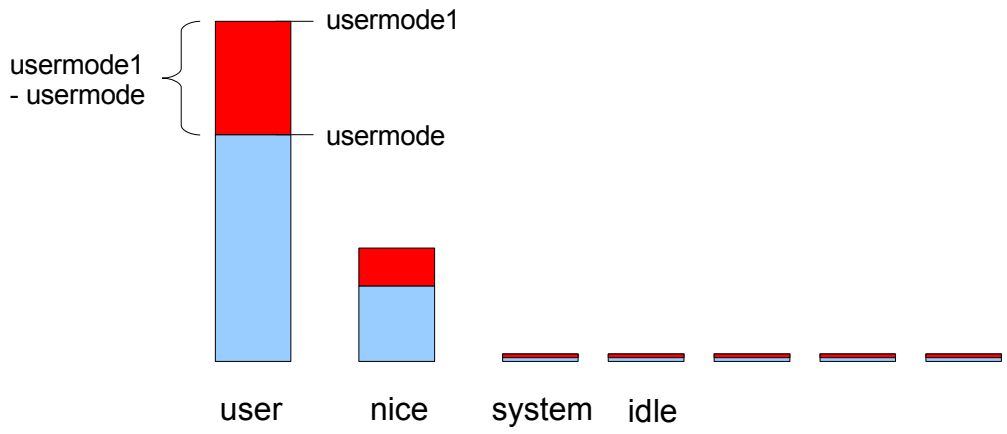
V.1.2.1.4.a Rendszer statisztika

Írjunk egy olyan programot, ami például megmondja, hogy „éppen most” mennyire idle a processzor? Tehát ahhoz hasonlót, mint amit a `top` parancs használatakor látunk:

```
top - 11:03:15 up 25 min,  3 users,  load average: 0.00, 0.00, 0.01
Tasks:  83 total,   2 running,  81 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   1024996k total,   327264k used,   697732k free,    19476k buffers
Swap:  2031608k total,        0k used,  2031608k free,   227848k cached
```

```
$ man proc
```

Itt olvassuk el a `/proc/stat` és a `/proc/uptime` bekezdést!



12. ábra: A /proc/stat fájlból egy másodperc különbséggel kiolvasott értékek ábrázolása.

Ezek alapján tudunk adni egy olyan megoldást, ami gyakorlatilag egyszerű fájlkezelési feladattá egyszerűsíti a példát:

```
#include <stdio.h>
#include <unistd.h>
#include <curses.h>
int
main ()
{
  FILE *procstat;
  unsigned long usermode, nice, systemmode, idle, iowait, irq,
                softirq;
  unsigned long usermode1, nice1, systemmode1, idle1, iowait1,
                irq1, softirq1;
  unsigned osszes;
  WINDOW *ablak;

  ablak = initscr ();
  noecho ();
  cbreak ();
  nodelay (ablak, true);

  for (;;)
    {
      procstat = fopen ("/proc/stat", "r");
      fscanf (procstat, "cpu %lu %lu %lu %lu %lu %lu %lu",
              &usermode, &nice,
              &systemmode, &idle, &iowait, &irq, &softirq);
      fclose (procstat);

      sleep (1);

      procstat = fopen ("/proc/stat", "r");
      fscanf (procstat, "cpu %lu %lu %lu %lu %lu %lu %lu",
              &usermode1,
              &nice1, &systemmode1, &idle1, &iowait1, &irq1,
              &softirq1);
      fclose (procstat);

      usermode1 -= usermode;
      nice1 -= nice;
      systemmode1 -= systemmode;
      idle1 -= idle;
      iowait1 -= iowait;
      irq1 -= irq;
      softirq1 -= softirq;

      osszes = usermode1 + nice1 + systemmode1 + idle1
        + iowait1 + irq1 + softirq1;

      clear ();

      printw
        ("user mode: %.2f%% nice: %.2f%% system mode: %.2f%% idle:
%.2f%% IO wait: %.2f%% irq: %.2f%% soft irq: %.2f%%\n",
         (double) usermode1 / osszes * 100,
         (double) nice1 / osszes * 100,
         (double) systemmode1 / osszes * 100,
```

```
          (double) idle1 / osszes * 100,
          (double) iowait1 / osszes * 100,
          (double) irq1 / osszes * 100,
          (double) softirq1 / osszes * 100);

      refresh ();

      if (getch () == 'q')
        break;

    }

  endwin ();

  return 0;
}
```

Lássuk: fordítsuk, futtassuk – miközben majd futtatunk két végtelen ciklust is, hogy lássuk működik-e a megoldásunk, s persze egy, futása közben kiadott top parancs megfelelő adataival is vessük össze a mi kimenetünket!

```
$ gcc proci.c -o proci -lcurses
$ ./proci
.
.
.
user mode: 0.00% nice: 0.00% system mode: 0.00% idle: 100.00% IO
wait: 0.00% irq: 0.00% soft irq: 0.00%
.
.
.
```

Egy másik ablakban futtassuk a beígért két végtelen ciklust:

```
$ nice -n 5 ./v&
[1] 4075
$ ./v
```

Programunk reagál:

```
user mode: 60.00% nice: 40.00% system mode: 0.00% idle: 0.00% IO
wait: 0.00% irq: 0.00% soft irq: 0.00%
```

További ismerkedés a /proc fájlrendszerrel

Fejlesszük tovább előző programunkat, írjuk ki a rendszerben jelen lévő folyamatok számát! Ehhez felhasználhatjuk a *81*. oldal *Könyvtárak* című pontjának programját, mert a

```
$ man proc
```

`/proc/[number]` bejegyzése alapján tudjuk, hogy minden folyamathoz tartozik egy számmal kezdődő (egészen pontosan a folyamat PID azonosító száma alkotta) könyvtár – tehát nincs más dolgunk, mint a `/proc` alatt megszámolni a számmal kezdődő könyvtárakat.

A *81*. oldal *Könyvtárak* című pontja programjának kis átalakításával egészítsük ki a korábbi rendszer statisztikát kiíró programunkat:

```
...
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>
int
folyamatok_szama()
{
  DIR *dirp;
  struct dirent *direntp;
  int folyamatok_szama = 0;
  if(dirp = opendir("/proc"))
    {
      while(direntp = readdir(dirp))
        if( direntp->d_name[0] >= '0'
            && direntp->d_name[0] <= '9')
          ++folyamatok_szama;
    }
  else
    {
      perror("opendir");
      return -1;
    }
  closedir(dirp);
  return folyamatok_szama;
}

...
      printw("%d folyamat van a rendszerben", folyamatok_szama());
...
```

Kreatív feladatként egészítsük még ki olyan infókkal a `/proc` alól programunk kimenetét, amivel csak akarjuk. Például írjuk ki a `/proc/cpuinfo` fájlból a processzor típusát, a processzort jellemző MHz értéket és mondjuk a BogoMIPS értéket, a `/proc/sys/kernel/osrelease` fájlból a kernel verzióját stb.

Ha ezzel megvagyunk, akkor készítsünk, avagy korábbi példáinkból ollózzunk össze még egy további változatot, melyben az eddigieken túl felhasználjuk az *81*. oldal, *Infók a fájlokról* című pontjában megismert `stat` struktúrát annak megoldásában, hogy kiíratjuk a saját folyamatainkról néhány információt. Nevezetesen csak azokat a folyamatokat tekintjük, amelyeknek megfelelő `/proc` alatti bejegyzésének tulajdonosai mi vagyunk.

```
...
void kiir_folyamat(char* bejegyzes)
{
  FILE *procpidstat;
  int pid;
  char buffer[1024];
  int allapot;
```

```
  procpidstat = fopen (bejegyzes, "r");
  fscanf (procpidstat, "%d %s %c", &pid, buffer, &allapot);
  fclose (procpidstat);

  printw("%s %c", buffer, allapot);

}

int
sajat_folyamatok()
{
  DIR *dirp;
  struct dirent *direntp;
  struct stat s;
  char buffer[1024];
  int folyamatok_szama = 0;
  int uid = getuid();

  if(dirp = opendir("/proc"))
    {
      mvprintw(3, 4, "PID Név Állapot");

      while(direntp = readdir(dirp))
        if( direntp->d_name[0] >= '0'
            && direntp->d_name[0] <= '9')
          {
            sprintf(buffer, "/proc/%s", direntp->d_name);
            if(stat(buffer, &s) != -1)
              {
                if(s.st_uid == uid)
                  {
                    mvprintw(3+(++folyamatok_szama), 4,
                             "%s ", direntp->d_name);
                    sprintf(buffer, "/proc/%s/stat",
                            direntp->d_name);
                    kiir_folyamat(buffer);
                  }
              }
            else
              perror("aa");
          }
    }
  else
    {
      perror("opendir");
      return -1;
    }
  closedir(dirp);
  return folyamatok_szama;
}

int
main ()
{
...
      mvprintw(2, 3, "%d saját folyamat van a rendszerben:",
               sajat_folyamatok());
```

```
...
```

Fordítva majd futtatva kapjuk, hogy

```
user mode: 0.00% nice: 0.00% system mode: 0.00% idle: 100.00%

   6 saját folyamat van a rendszerben:
    PID Név Állapot
    2283 (sshd) S
    2284 (bash) S
    2347 (sshd) S
    2348 (bash) S
    4285 (emacs-x) S
    4292 (rendstat) R
```

Az előzőek alapján a kreatív feladatra magunk az alábbi megoldást (pidcs) adjuk (feladatként kiegészíthetjük például a `/proc/meminfo`, a `/proc/loadavg` vagy például a `/proc/uptime` fájlokból mazsolázott információkkal, a jobb oldali üresen hagyott dobozt levehetjük, de tartalommal is megtölthetjük.)

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ncurses.h>
#include <time.h>
#include <string.h>
#include <sys/types.h>
#include <dirent.h>

typedef struct jellemzok
{
  char *os_tipus;
  char *kernel_verzio;
  char *processzor_model;
  char *processzor_mhz;
  char *bogomips;

} JELLEMZOK, *JELLEMZOK_MUTATO;

typedef struct folyamat
{
  struct folyamat *kovetkezo;
  struct folyamat *elozo;
  int pid;
  int ppid;
  char *nev;
  char allapot;

} FOLYAMAT_LISTA, *FOLYAMAT_LISTA_MUTATO;

JELLEMZOK_MUTATO
jellemzok_osszegyujtese ()
{

  FILE *f;
```

```
  char buffer[1024];
  JELLEMZOK_MUTATO konstans_jellemzok;

  konstans_jellemzok = (JELLEMZOK_MUTATO) malloc (sizeof
(JELLEMZOK));

  f = fopen ("/proc/cpuinfo", "r");

  do
    {
      fgets (buffer, 1024, f);
      if (!strncmp (buffer, "model name", strlen ("model name")))
     {
       konstans_jellemzok->processzor_model =
         strndup (strchr (buffer, ':') + 2,
               strlen (strchr (buffer, ':') + 2) - 1);
     }
      else if (!strncmp (buffer, "cpu MHz", strlen ("cpu MHz")))
     {
       konstans_jellemzok->processzor_mhz =
         strndup (strchr (buffer, ':') + 2,
               strlen (strchr (buffer, ':') + 2) - 1);

     }
      else if (!strncmp (buffer, "bogomips", strlen ("bogomips")))
     {
       konstans_jellemzok->bogomips =
         strndup (strchr (buffer, ':') + 2,
               strlen (strchr (buffer, ':') + 2) - 1);

     }

    }
  while (!feof (f));
  fclose (f);

  f = fopen ("/proc/sys/kernel/ostype", "r");
  fgets (buffer, 1024, f);
  konstans_jellemzok->os_tipus = strndup (buffer, strlen (buffer) -
1);
  fclose (f);

  f = fopen ("/proc/sys/kernel/osrelease", "r");
  fgets (buffer, 1024, f);
  konstans_jellemzok->kernel_verzio = strndup (buffer, strlen
(buffer) - 1);
  fclose (f);

  return konstans_jellemzok;
}

void
konstans_jellemzok_szabadit (JELLEMZOK_MUTATO konstans_jellemzok)
{

  free (konstans_jellemzok->processzor_model);
  free (konstans_jellemzok->processzor_mhz);
  free (konstans_jellemzok->bogomips);
```

```
  free (konstans_jellemzok->os_tipus);
  free (konstans_jellemzok->kernel_verzio);
  free (konstans_jellemzok);

}

void
folyamat_lista_feltolt (FOLYAMAT_LISTA_MUTATO aktualis, char
*procpidstat)
{

  FILE *f;
  char buffer[1024];

  f = fopen (procpidstat, "r");
  fscanf (f, "%d %s %c %d", &(aktualis->pid),
       buffer, &(aktualis->allapot), &(aktualis->ppid));
  aktualis->nev = strdup (buffer);
  fclose (f);
}

FOLYAMAT_LISTA_MUTATO
folyamatok_osszegyujtese ()
{
  FOLYAMAT_LISTA_MUTATO lista_feje, aktualis, elozo = NULL;
  DIR *dirp;
  struct dirent *direntp;
  char buffer[1024];

  if (dirp = opendir ("/proc"))
    {
      while (direntp = readdir (dirp))
     if (direntp->d_name[0] >= '0' && direntp->d_name[0] <= '9')
       {
         snprintf (buffer, 1024, "/proc/%s/stat", direntp->d_name);
         aktualis =
           (FOLYAMAT_LISTA_MUTATO) malloc (sizeof
(FOLYAMAT_LISTA));
         folyamat_lista_feltolt (aktualis, buffer);
         if (elozo != NULL)
           {
          aktualis->elozo = elozo;
          elozo->kovetkezo = aktualis;
          aktualis->kovetkezo = NULL;
           }
         else
           {
          lista_feje = aktualis;
          aktualis->elozo = NULL;
           }
         elozo = aktualis;
       }
    }

  return lista_feje;
}

void
```

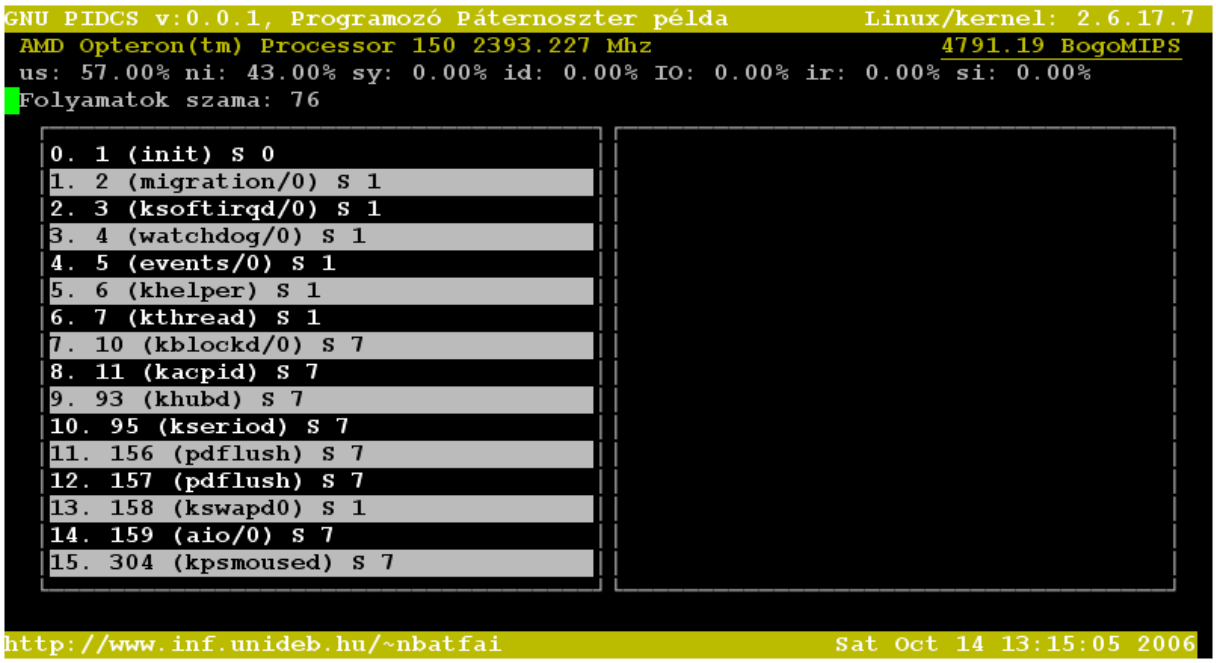
```
folyamatok_lista_szabadit (FOLYAMAT_LISTA_MUTATO fl)
{
  FOLYAMAT_LISTA_MUTATO akt;

  while (fl != NULL)
    {
      free (fl->nev);
      akt = fl;
      fl = fl->kovetkezo;
      free (akt);
    }

}

void
kiir_rendszer_statisztika ()
{
  FILE *proc_stat;
  static unsigned long usermode = 0, nice = 0, systemmode = 0, idle
=
    0, iowait = 0, irq = 0, softirq = 0;
  unsigned long dusermode, dnice, dsystemmode, didle, diowait,
dirq, dsoftirq;
  unsigned long usermode1, nice1, systemmode1, idle1, iowait1,
irq1, softirq1;
  unsigned osszes;

  proc_stat = fopen ("/proc/stat", "r");
  fscanf (proc_stat, "cpu %lu %lu %lu %lu %lu %lu %lu",
       &usermode1,
       &nice1, &systemmode1, &idle1, &iowait1, &irq1, &softirq1);
  fclose (proc_stat);

  dusermode = usermode1 - usermode;
  dnice = nice1 - nice;
  dsystemmode = systemmode1 - systemmode;
  didle = idle1 - idle;
  diowait = iowait1 - iowait;
  dirq = irq1 - irq;
  dsoftirq = softirq1 - softirq;

  usermode = usermode1;
  nice = nice1;
  systemmode = systemmode1;
  idle = idle1;
  iowait = iowait1;
  irq = irq1;
  softirq = softirq1;

  osszes = dusermode + dnice + dsystemmode + didle
    + diowait + dirq + dsoftirq;

  mvprintw
    (2, 1,
     "us: %.2f%% ni: %.2f%% sy: %.2f%% id: %.2f%% IO: %.2f%% ir:
%.2f%% si: %.2f%%\n",
     (double) dusermode / osszes * 100,
     (double) dnice / osszes * 100,
```

```
     (double) dsystemmode / osszes * 100,
     (double) didle / osszes * 100,
     (double) diowait / osszes * 100,
     (double) dirq / osszes * 100, (double) dsoftirq / osszes *
100);

}

void
kiir_jellemzok (JELLEMZOK_MUTATO konstans_jellemzok, int
folyamatok_szama)
{

  char buffer[128];
  time_t t;
  char *ido;

  attron (COLOR_PAIR (1));
  mvhline (0, 0, ' ', COLS);
  mvhline (LINES - 1, 0, ' ', COLS);
  attron (A_BOLD);
  mvprintw (0, 0, "GNU PIDCS v:0.0.1, Programozó Páternoszter
példa");
  mvprintw (0, COLS - (strlen (konstans_jellemzok->os_tipus) +
                 strlen (konstans_jellemzok->kernel_verzio) + 10),
         "%s/kernel: %s",
         konstans_jellemzok->os_tipus, konstans_jellemzok-
>kernel_verzio);
  mvprintw (LINES - 1, 0, "http://www.inf.unideb.hu/~nbatfai");
  t = time (NULL);
  ido = ctime_r (&t, buffer);
  mvprintw (LINES - 1, COLS - strlen (ido), "%s", ido);
  attroff (COLOR_PAIR (1));
  attroff (A_BOLD);

  attron (COLOR_PAIR (2));
  mvprintw (1, 1, "%s %s Mhz",
         konstans_jellemzok->processzor_model,
         konstans_jellemzok->processzor_mhz);
  attron (A_UNDERLINE);
  mvprintw (1, COLS - strlen (konstans_jellemzok->bogomips) - 11,
         "%s BogoMIPS", konstans_jellemzok->bogomips);
  attroff (A_UNDERLINE);
  attroff (COLOR_PAIR (2));

  mvprintw (3, 1, "Folyamatok szama: %d", folyamatok_szama);

}

int
folyamatok_szama (FOLYAMAT_LISTA_MUTATO fl)
{
  int n = 0;
  for (; fl != NULL; fl = fl->kovetkezo)
    ++n;
  return n;
}
```

```
void
kiir_folyamatok (WINDOW * ablak, int sorok, int irany,
           FOLYAMAT_LISTA_MUTATO folyamatok_lista)
{

  FOLYAMAT_LISTA_MUTATO fl;
  int i;
  static int mennyitol = 0;
  int mennyi = sorok - 2;

  irany = irany * (mennyi - 2);
  if (mennyitol + irany >= 0
      && mennyitol + irany + mennyi <
      folyamatok_szama (folyamatok_lista) + mennyi)
    mennyitol = mennyitol + irany;

  box (ablak, 0, 0);

  for (i = 0, fl = folyamatok_lista; fl != NULL; ++i, fl = fl-
>kovetkezo)
    if (i >= mennyitol && i < mennyitol + mennyi)
      {
     if (i % 2)
       {
         wattron (ablak, A_REVERSE);
         mvwhline (ablak, i + 1 - mennyitol, 1, ' ', COLS / 2 - 4);
         mvwprintw (ablak, i + 1 - mennyitol, 1, "%d. %d %s %c %d",
i,
                  fl->pid, fl->nev, fl->allapot, fl->ppid);
         wattroff (ablak, A_REVERSE);
       }
     else
       {
         wattron (ablak, A_BOLD);
         mvwprintw (ablak, i + 1 - mennyitol, 1, "%d. %d %s %c %d",
i,
                  fl->pid, fl->nev, fl->allapot, fl->ppid);
         wattroff (ablak, A_BOLD);
       }
      }
  wrefresh (ablak);

}

void
kiir_reszletek (WINDOW * ablak)
{

  box (ablak, 0, 0);
  wrefresh (ablak);
}

int
main ()
{
  WINDOW *ablak, *folyamatok_ablak, *reszletes_ablak;
  JELLEMZOK_MUTATO konstans_jellemzok;
```

```
  FOLYAMAT_LISTA_MUTATO folyamatok_lista;
  int sorok, billentyu, irany;

  ablak = initscr ();
  start_color ();
  init_pair (1, COLOR_WHITE, COLOR_YELLOW);
  init_pair (2, COLOR_YELLOW, COLOR_BLACK);
  clear ();
  noecho ();
  cbreak ();
  nodelay (ablak, true);

  sorok = LINES - 6;
  folyamatok_ablak = subwin (ablak, sorok, COLS / 2 - 2, 4, 2);
  reszletes_ablak = subwin (ablak, sorok, COLS / 2 - 2, 4, COLS /
2);

  konstans_jellemzok = jellemzok_osszegyujtese ();

  for (;;)
    {

      clear ();

      folyamatok_lista = folyamatok_osszegyujtese ();

      kiir_jellemzok (konstans_jellemzok,
                folyamatok_szama (folyamatok_lista));
      kiir_rendszer_statisztika ();
      kiir_folyamatok (folyamatok_ablak, sorok, irany,
folyamatok_lista);
      kiir_reszletek (reszletes_ablak);

      folyamatok_lista_szabadit (folyamatok_lista);

      if ((billentyu = getch ()) == 'q')
     break;
      else if (billentyu == 'f')
     irany = -1;
      else if (billentyu == 'l')
     irany = 1;
      else
     irany = 0;

      refresh ();
      sleep (1);

    }

  endwin ();

  konstans_jellemzok_szabadit (konstans_jellemzok);

  return 0;
}
```

Mivel ez egy viszonylag hosszabb forrás, nézzük meg a kimenetét is:



13. ábra: A pidcs 0.0.1 program felülete

Ennyi ismerkedés után írjunk egy olyan programot, amivel tudunk lépkedni, nézelődni a folyamatok között!

PID családfa

Dőljünk tehát hátra és tervezzük meg az előző pontban kívánt programot!

VI. Hálózati programozás

Az ISO OSI és TCP/IP hivatkozási modell összevetése a programozó nézőpontjából.

VI.1 Socket programozás

Folyamatok közötti kommunikáció. TCP/IP.

```
$ man 7 ip
$ man 7 tcp
```

VI.1.1 Socket szerverek

Állatorvosi példánk egy daytime jellegű TCP szerver, azaz nem csinál mást, mint küldi a rendszeridőt a jelentkező klienseknek. Tehát ha a futó szerverhez csatlakozunk, akkor valami ilyesmit kapunk majd:

```
Sun Oct 30 09:47:42 2005
```

Szervereinket tipikusan a `telnet` kliens programmal fogjuk tesztelni:

```
$ telnet localhost 2005
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Sun Oct 30 09:52:38 2005
Connection closed by foreign host.
```

(A következő szerver oldali programok kapcsán olvassuk majd el az 129. oldalon a Nem szálbiztos függvények című fejezetet is! A lokális folyamatok közötti socketes kommunikáció ténában a Folyamatok kommunikációja, IPC fejezetben a 47. oldalon, a Socketek című pontot keressük fel!)

VI.1.1.1 Soros, azaz iteratív

Ennél az első, a soros szervernél IPv4 és IPv6 példát is mutatunk.

VI.1.1.1.1 IPv4

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
```

```
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int
main(void)
{
  int kapu_figyelo, kapcsolat, kliensm, sockoptval=1;
  struct sockaddr_in szerver, kliens;
  memset((void *)&szerver, 0, sizeof(szerver));
  szerver.sin_family= AF_INET;
  inet_aton("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port= htons(SZERVER_PORT);
  if((kapu_figyelo = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE);
  }
  setsockopt(kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
    (void *)&sockoptval, sizeof(sockoptval));
  if(bind(kapu_figyelo, (struct sockaddr *)&szerver,
    sizeof(szerver)) == -1)
  {
    perror("bind");
    exit(EXIT_FAILURE);
  }
  if(listen(kapu_figyelo, SZERVER_SOR_MERET) == -1)
  {
    perror("listen");
    exit(EXIT_FAILURE);
  }
  printf("%s:%d\n",
    inet_ntoa(szerver.sin_addr), ntohs(szerver.sin_port));
  for(;;)
  {
    memset((void *) &kliens, 0, (kliensm = sizeof(kliens)));
    if((kapcsolat = accept(kapu_figyelo,
      (struct sockaddr *)&kliens, (socklen_t *)&kliensm)) == -1)
    {
      perror("accept");
      exit(EXIT_FAILURE);
    }
    printf("     <-> %s:%d\n",
      inet_ntoa(kliens.sin_addr), ntohs(kliens.sin_port));
    if(kiszolgal(kapcsolat) == -1)
    {
      perror("kiszolgal");
    }
    close(kapcsolat);
  }
}
```

Tanulmányozzuk a kiemelt függvények manuáljait, majd fordítsuk így:

```
$ gcc -lnsl -o szerver szerver.c
```

Teszteljük a `telnet` kliens programmal, ekkor a szerver oldalon az alábbi logolást használjuk:

```
$ ./szerver
127.0.0.1:2005
     <-> 127.0.0.1:35277
```

a kliens oldalon (közben persze a szervert nem lelőve :) így csatlakozunk:

```
$ telnet localhost 2005
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Sat Apr 16 12:24:21 2005
Connection closed by foreign host.
```

(Tesztelhetünk a későbbiekben tekintett kliens programokkal is, lásd az 132. oldaltól.)

E szerver kapcsán tanulmányozuk a *263*. oldal *netstat* című pontjának `netstat -tapc` példáját!

Parancsok:
`netstat, (/sbin/)ifconfig`

VI.1.1.1.2 IPv6 szerver oldal

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
int
kiszolgal (int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time (NULL);
  char *ts = ctime_r (&t, buffer);
  return write (kliens, ts, strlen (ts));
}

int
main (void)
{
  int kapu_figyelo, kapcsolat, kliensm, sockoptval = 1;
  struct sockaddr_in6 szerver, kliens;
  char buffer[INET6_ADDRSTRLEN];
```

```
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin6_family = AF_INET6;
  szerver.sin6_flowinfo = 0;
  inet_pton (AF_INET6, "::FFFF:C0A8:0101", &(szerver.sin6_addr));
  szerver.sin6_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET6, SOCK_STREAM, IPPROTO_TCP))
== -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  setsockopt (kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
          (void *) &sockoptval, sizeof (sockoptval));
  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
         sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (listen (kapu_figyelo, SZERVER_SOR_MERET) == -1)
    {
      perror ("listen");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n",
       inet_ntop (AF_INET6, &(szerver.sin6_addr), buffer,
                  sizeof (buffer)), ntohs (szerver.sin6_port));
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));
      if ((kapcsolat = accept (kapu_figyelo,
                      (struct sockaddr *) &kliens,
                      (socklen_t *) & kliensm)) == -1)
     {
       perror ("accept");
       exit (EXIT_FAILURE);
     }
      printf ("     <-> %s:%d\n",
           inet_ntop (AF_INET6, &(kliens.sin6_addr), buffer,
                sizeof (buffer)), ntohs (kliens.sin6_port));
      if (kiszolgal (kapcsolat) == -1)
     {
       perror ("kiszolgal");
     }
      close (kapcsolat);
    }
}
```

Fordítás után futtatva:

```
$ ./ipv6_soros
::ffff:192.168.1.1:2005
```

a *133*. oldal IPv6 kliensével tesztelve:

```
$ ./ipv6_kliens
Sun Nov 27 13:06:58 2005
```

közben a szerver logolása:

```
$ ./ipv6_soros
::ffff:192.168.1.1:2005
     <-> ::ffff:192.168.1.1:44642
```

A localhost megadása:

```
inet_pton (AF_INET6, "::1", &(szerver.sin6_addr));
```

Az IPv6 wildcard cím használata:

```
szerver.sin6_addr = in6addr_any;
```

Dolgozzuk össze a folyamatokkal foglalkozó fejezet összefoglaló, 40. oldali, *A fork() tipikus használata* című programját ezzel az iménti IPv4-es soros példával:

VI.1.1.2 Párhuzamos, azaz konkurens, folyamatokkal

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
void
zombi_elharito(int sig)
{
  signal(SIGCHLD, zombi_elharito);
  while(wait(NULL) > 0)
    ;
}
int
main(void)
{
```

```
  int kapu_figyelo, kapcsolat, kliensm, gyermekem_pid,
sockoptval=1;
  struct sockaddr_in szerver, kliens;
  memset((void *)&szerver, 0, sizeof(szerver));
  szerver.sin_family= AF_INET;
  inet_aton("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port= htons(SZERVER_PORT);
  if((kapu_figyelo = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE);
  }
  setsockopt(kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
    (void *)&sockoptval, sizeof(sockoptval));
  if(bind(kapu_figyelo, (struct sockaddr *)&szerver,
    sizeof(szerver)) == -1)
  {
    perror("bind");
    exit(EXIT_FAILURE);
  }
  if(listen(kapu_figyelo, SZERVER_SOR_MERET) == -1)
  {
    perror("listen");
    exit(EXIT_FAILURE);
  }
  printf("%s:%d\n",
    inet_ntoa(szerver.sin_addr), ntohs(szerver.sin_port));
  signal(SIGCHLD, zombi_elharito);
  for(;;)
  {
    memset((void *) &kliens, 0, (kliensm = sizeof(kliens)));
    if((kapcsolat = accept(kapu_figyelo,
       (struct sockaddr *)&kliens, (socklen_t *)&kliensm)) == -1)
    {
      perror("accept");
      exit(EXIT_FAILURE);
    }
    printf("     <-> %s:%d\n",
      inet_ntoa(kliens.sin_addr), ntohs(kliens.sin_port));

    if((gyermekem_pid = fork()) == 0)
    {
      close(kapu_figyelo);
      if(kiszolgal(kapcsolat) == -1)
      {
        perror("kiszolgal");
      }
      close(kapcsolat);
      exit(EXIT_SUCCESS);
    }
    else if(gyermekem_pid > 0)
    {
      // wait(&statusz); e miatt kezeljuk a SIGCHLD jelet,
      // l. a Zombik fejezetet!
      close(kapcsolat);
    }
    else
```

```
    {
      close(kapcsolat);
      perror("fork");
      exit(EXIT_FAILURE);
    }
  }
}
```

VI.1.1.2.1 Kérdések – a párhuzamos szerverről

- Miért válik szerverünk zombi szerverré, ha nem foglalkozunk a `SIGCHLD` jellel, vagy `signal(SIGCHLD, SIG_DFL);` ?
- Mi történne `signal(SIGCHLD, SIG_IGN);` mellett?
- Elemezzük a következő megoldásokat a kezelőben:

```
while(waitpid(-1, NULL, WNOHANG) > 0);
while(wait3(NULL, WNOHANG, NULL) > 0);
```

- Kapjunk el `$ top -u nbatfai` –ból egy olyan pillanatot, amikor láthatjuk, hogy több folyamatunk szolgál ki! *(Tegyünk például egy `sleep()`-et a `kiszolgal()` függvénybe és futtassuk az alábbi kis szkriptet.)*

```
telnet localhost 2005&
telnet localhost 2005&
telnet localhost 2005&
telnet localhost 2005&
telnet localhost 2005&
```

Azaz lássunk ilyet:

```
26737 nbatfai   14   0   384  380  368 S  0.0  0.1   0:00.00 szerver
26760 nbatfai   10   0   388  384  360 S  0.0  0.2   0:00.00 szerver
26761 nbatfai   11   0   388  384  360 S  0.0  0.2   0:00.00 szerver
26762 nbatfai   12   0   388  384  360 S  0.0  0.2   0:00.00 szerver
26765 nbatfai   13   0   388  384  360 S  0.0  0.2   0:00.00 szerver
26766 nbatfai   14   0   388  384  372 S  0.0  0.2   0:00.00 szerver
```

(Ha nem sikerül elkapnunk a több kiszolgáló folyamatot, akkor tesztelő szkriptünket módosítsuk, például így:

```
telnet localhost 2005&
telnet localhost 2005&
telnet localhost 2005&
telnet localhost 2005&
telnet localhost 2005&
ps axu|grep szerver > folyamatok
```

s ne felejtsünk egy `sleep(5)`-öt betenni a `kiszolgal()` elejére, aztán szervert fordít, futtat, tesztelő szkriptre futási jogot ad, futtat és végül kimóroljuk a `folyamatok` fájlt.)

Ugyanazt a `sleep()`-et tegyük be a soros szerverbe is és hasonlítsuk össze, hogy mennyi idő alatt szolgálja ki az öt klienst a soros, illetve a párhuzamos szerver.

VI.1.1.3 Párhuzamos, folyamatok sorával

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define SZERVER_FOLYAMATOK_SOR_MERET 5
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
void
kiszolgalo_folyamat(int kapu_figyelo)
{
  int kapcsolat, kliensm;
  struct sockaddr_in kliens;
  for(;;)
  {
    memset((void *) &kliens, 0, (kliensm = sizeof(kliens)));
    if((kapcsolat = accept(kapu_figyelo,
      (struct sockaddr *)&kliens, (socklen_t *)&kliensm)) == -1)
    {
      perror("accept");
      exit(EXIT_FAILURE);
    }
    printf("     <-> %s:%d\n",
    inet_ntoa(kliens.sin_addr), ntohs(kliens.sin_port));
    if(kiszolgal(kapcsolat) == -1)
    {
      perror("kiszolgal");
    }
    close(kapcsolat);
  }
}
int
main(void)
{
  int kapu_figyelo, kliensm, gyermekem_pid, sockoptval=1;
  int i;
  struct sockaddr_in szerver, kliens;
  memset((void *)&szerver, 0, sizeof(szerver));
  szerver.sin_family= AF_INET;
  inet_aton("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port= htons(SZERVER_PORT);
  if((kapu_figyelo = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))
```

```
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE);
  }
  setsockopt(kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
    (void *)&sockoptval, sizeof(sockoptval));
  if(bind(kapu_figyelo,
     (struct sockaddr *)&szerver,sizeof(szerver)) == -1)
  {
    perror("bind");
    exit(EXIT_FAILURE);
  }
  if(listen(kapu_figyelo, SZERVER_SOR_MERET) == -1)
  {
    perror("listen");
    exit(EXIT_FAILURE);
  }
  printf("%s:%d\n",
    inet_ntoa(szerver.sin_addr), ntohs(szerver.sin_port));
  for(i=0; i<SZERVER_FOLYAMATOK_SOR_MERET; ++i)
  {
    if((gyermekem_pid = fork()) == 0)
    {
      kiszolgalo_folyamat(kapu_figyelo);
    }
    else if(gyermekem_pid > 0)
    {
    }
    else
    {
      perror("fork");
      exit(EXIT_FAILURE);
    }
  }
  for(;;)
    pause();
}
```

(A program fordítása kapcsán olvassuk el a 266. oldal Üzenetek című pontját.)

Hasonlóan ez előző pédához, most is nézzük meg, hány folyamatunk van, most elég a

```
$ps axu|grep szerver
```

Miért elég?

Nézzük meg, hogy a szerverben milyen fájlleírók vannak:

```
$ ls -l /proc/2933/fd/
összesen 4
lrwx------  1 norbi norbi 64 okt 22 08:54 0 -> /dev/tty2
lrwx------  1 norbi norbi 64 okt 22 08:54 1 -> /dev/tty2
lrwx------  1 norbi norbi 64 okt 22 08:54 2 -> /dev/tty2
lrwx------  1 norbi norbi 64 okt 22 08:54 3 -> socket:[15400]
```

Ebben a példában az az érdekes, hogy egyszerre több folyamat `accept()`-el. Ennek a BSD-beli megoldásáról részletesen olvashatunk a [Hálózati/UNP, 739. oldalon]. A további néhány példában magában a szerver kódjába építjük be, hogy egyszerre egy valaki `accept()`-eljen.

VI.1.1.4 Párhuzamos, kölcsönös kizárással accept-elő folyamatok sorával

VI.1.1.4.1 A folyamatok összehangolása szemaforral

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define SZERVER_FOLYAMATOK_SOR_MERET 5
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int zar;
void
kiszolgalo_folyamat (int kapu_figyelo, int szemafor)
{
  int kapcsolat, kliensm;
  struct sockaddr_in kliens;
  struct sembuf zar, nyit;
  zar.sem_num = 0;
  zar.sem_op = -1;
  nyit.sem_num = 0;
  nyit.sem_op = 1;

  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));
      printf ("zar elott: %d\n", getpid ());
      fflush (stdout);
      if (semop (szemafor, &zar, 1) == -1)
     {
       perror ("semop");
       exit (EXIT_FAILURE);
     }
```

```
      printf ("zar utan accept elott: %d\n", getpid ());
      fflush (stdout);
      if ((kapcsolat = accept (kapu_figyelo,
                      (struct sockaddr *) &kliens,
                      (socklen_t *) & kliensm)) == -1)
     {
       perror ("accept");
       exit (EXIT_FAILURE);
     }
      if (semop (szemafor, &nyit, 1) == -1)
     {
       perror ("semop");
       exit (EXIT_FAILURE);
     }
      printf ("     <-> %s:%d\n",
           inet_ntoa (kliens.sin_addr), ntohs (kliens.sin_port));
      if (kiszolgal (kapcsolat) == -1)
     {
       perror ("kiszolgal");
     }
      close (kapcsolat);
    }
}
int
main (void)
{
  int kapu_figyelo, kliensm, gyermekem_pid, sockoptval = 1;
  int i, szemafor;;
  struct sockaddr_in szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP))
== -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  setsockopt (kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
           (void *) &sockoptval, sizeof (sockoptval));
  if (bind (kapu_figyelo,
         (struct sockaddr *) &szerver, sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (listen (kapu_figyelo, SZERVER_SOR_MERET) == -1)
    {
      perror ("listen");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr), ntohs
(szerver.sin_port));
  if ((szemafor =
       semget (ftok (".", 42), 1, IPC_CREAT | S_IRUSR | S_IWUSR))
== -1)
    {
```

```
      perror ("semget");
      exit (EXIT_FAILURE);
    }
  printf ("szemafor: %d\n", szemafor);
  fflush (stdout);
  if (semctl (szemafor, 0, SETVAL, 1))
    {
      perror ("semctl");
      exit (EXIT_FAILURE);
    }
  for (i = 0; i < SZERVER_FOLYAMATOK_SOR_MERET; ++i)
    {
      if ((gyermekem_pid = fork ()) == 0)
     {
       kiszolgalo_folyamat (kapu_figyelo, szemafor);
     }
      else if (gyermekem_pid > 0)
     {
     }
      else
     {
       perror ("fork");
       exit (EXIT_FAILURE);
     }
    }
  for (;;)
    pause ();
}
```

A példa egyben az *46*. oldal, *Szemaforok* című fejezetének is példája, ha nem emlékszünk a kapcsolódó parancsokra, akkor érdemes oda kattintani! Futtassuk és elemezzük a logolást:

```
$ ./szerver
127.0.0.1:2005
szemafor: 327688
zar elott: 2661
zar utan accept elott: 2661
zar elott: 2662
zar elott: 2663
zar elott: 2665
zar elott: 2664
     <-> 127.0.0.1:32884
zar elott: 2661
zar utan accept elott: 2662
.
.
.
```

VI.1.1.4.2 A folyamatok összehangolása állományzárolással

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define SZERVER_FOLYAMATOK_SOR_MERET 5
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int zar;
void
kiszolgalo_folyamat (int kapu_figyelo)
{
  int kapcsolat, kliensm;
  struct sockaddr_in kliens;
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));
      if (lockf (zar, F_LOCK, 0) == -1)
      {
       perror ("flock");
      }
      if ((kapcsolat = accept (kapu_figyelo,
                      (struct sockaddr *) &kliens,
                      (socklen_t *) & kliensm)) == -1)
      {
       perror ("accept");
       exit (EXIT_FAILURE);
      }

      if (lockf (zar, F_ULOCK, 0) == -1)
      {
       perror ("flock");
      }
      printf ("     <-> %s:%d\n",
           inet_ntoa (kliens.sin_addr), ntohs (kliens.sin_port));
      if (kiszolgal (kapcsolat) == -1)
      {
       perror ("kiszolgal");
      }
      close (kapcsolat);
    }
}
int
main (void)
{
  int kapu_figyelo, kliensm, gyermekem_pid, sockoptval = 1;
  int i;
```

```
  struct sockaddr_in szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_STREAM,
                IPPROTO_TCP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  setsockopt (kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
           (void *) &sockoptval, sizeof (sockoptval));
  if (bind (kapu_figyelo,
         (struct sockaddr *) &szerver, sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (listen (kapu_figyelo, SZERVER_SOR_MERET) == -1)
    {
      perror ("listen");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr),
                     ntohs (szerver.sin_port));
  if ((zar = creat ("szerver_fork_sor.zar", S_IRWXU)) == -1)
    {
      perror ("create");
      exit (EXIT_FAILURE);
    }
  for (i = 0; i < SZERVER_FOLYAMATOK_SOR_MERET; ++i)
    {
      if ((gyermekem_pid = fork ()) == 0)
      {
       kiszolgalo_folyamat (kapu_figyelo);
      }
      else if (gyermekem_pid > 0)
      {
      }
      else
      {
       perror ("fork");
       exit (EXIT_FAILURE);
      }
    }
  for (;;)
    pause ();
}
```

Hasonlóan az előző, szemaforra várakozó példához, pillantsunk bele a várakozásba, például így:

```
      printf ("zar elott: %d\n", getpid ());
      fflush (stdout);
      .
      .
      .
      printf ("zar utan accept elott: %d\n", getpid ());
      fflush (stdout);
```

VI.1.1.5 Párhuzamos, POSIX szálakkal

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <pthread.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
void *
kiszolgal(void *kapcsolat)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  write(*(int *)kapcsolat, ts, strlen(ts));
  close(*(int *)kapcsolat);
  free(kapcsolat);
  pthread_exit(NULL);
}
int
main(void)
{
  pthread_t nemkell_id;
  int kapu_figyelo, *kapcsolat, kliensm, sockoptval=1;
  struct sockaddr_in szerver, kliens;
  memset((void *)&szerver, 0, sizeof(szerver));
  szerver.sin_family= AF_INET;
  inet_aton("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port= htons(SZERVER_PORT);
  if((kapu_figyelo = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE);
  }
  setsockopt(kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
    (void *)&sockoptval, sizeof(sockoptval));
  if(bind(kapu_figyelo, (struct sockaddr *)&szerver,
    sizeof(szerver)) == -1)
  {
    perror("bind");
    exit(EXIT_FAILURE);
  }
  if(listen(kapu_figyelo, SZERVER_SOR_MERET) == -1)
  {
    perror("listen");
    exit(EXIT_FAILURE);
  }
```

```
  printf("%s:%d\n",
    inet_ntoa(szerver.sin_addr), ntohs(szerver.sin_port));
  for(;;)
  {
    if((kapcsolat = (int *)malloc(sizeof(int))) == NULL)
    {
      perror("memoria");
      exit(EXIT_FAILURE);
    }
    memset((void *) &kliens, 0, (kliensm = sizeof(kliens)));
    if((*kapcsolat = accept(kapu_figyelo, (struct
      sockaddr *)&kliens, (socklen_t *)&kliensm)) == -1)
    {
      perror("accept");
      exit(EXIT_FAILURE);
    }
    printf("     <-> %s:%d\n",
      inet_ntoa(kliens.sin_addr), ntohs(kliens.sin_port));
    if(pthread_create(&nemkell_id, NULL,
      kiszolgal, (void *)kapcsolat))
    {
      perror("pthread_create");
      exit(EXIT_FAILURE);
    }
  }
}
```

Dolgozzuk össze a korábbi `fork()`-os, sort használó példát ez előbbivel, azaz lássuk **szálak sorával**:

VI.1.1.6 Párhuzamos, szálak sorával

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <pthread.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define SZERVER_SZALAK_SOR_MERET 5
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
void *
kiszolgalo_szal(void *kapu_figyelo)
{
```

```
  int kapcsolat, kliensm;
  struct sockaddr_in kliens;
  for(;;)
  {
    memset((void *) &kliens, 0, (kliensm = sizeof(kliens)));
    if((kapcsolat = accept(*(int *)kapu_figyelo,
      (struct sockaddr *)&kliens, (socklen_t *)&kliensm)) == -1)
    {
      perror("accept");
      exit(EXIT_FAILURE);
    }
    printf("     <-> %s:%d\n",
    inet_ntoa(kliens.sin_addr), ntohs(kliens.sin_port));
    if(kiszolgal(kapcsolat) == -1)
    {
      perror("kiszolgal");
    }
    close(kapcsolat);
  }
}
int
main(void)
{
  pthread_t szalak[SZERVER_SZALAK_SOR_MERET];
  int kapu_figyelo, kliensm, gyermekem_pid, sockoptval=1, i;
  struct sockaddr_in szerver, kliens;
  memset((void *)&szerver, 0, sizeof(szerver));
  szerver.sin_family= AF_INET;
  inet_aton("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port= htons(SZERVER_PORT);
  if((kapu_figyelo = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE);
  }
  setsockopt(kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
    (void *)&sockoptval, sizeof(sockoptval));
  if(bind(kapu_figyelo, (struct sockaddr *)&szerver,
    sizeof(szerver)) == -1)
  {
    perror("bind");
    exit(EXIT_FAILURE);
  }
  if(listen(kapu_figyelo, SZERVER_SOR_MERET) == -1)
  {
    perror("listen");
    exit(EXIT_FAILURE);
  }
  printf("%s:%d\n",
    inet_ntoa(szerver.sin_addr), ntohs(szerver.sin_port));
  for(i=0; i<SZERVER_SZALAK_SOR_MERET; ++i)
  {
    if(pthread_create(&szalak[i], NULL, kiszolgalo_szal,
      (void *)&kapu_figyelo))
    {
      error("pthread_create");
      exit(EXIT_FAILURE);
```

```
    }
  }
  for(;;)
    pause();
}
```

Két különböző gépen is megnézük, hány folyamatunk, szálunk van, az egyik egy

```
$uname -r
2.4.19
```

gép, itt

```
$ ps axHo comm,pid,ppid,stat,tid,nlwp
COMMAND           PID  PPID STAT   TID NLWP
szerver         22265 22226 S+   22265    1
szerver         22266 22265 S+   22266    1
szerver         22267 22266 S+   22267    1
szerver         22268 22266 S+   22268    1
szerver         22269 22266 S+   22269    1
szerver         22270 22266 S+   22270    1
szerver         22271 22266 S+   22271    1
```

A másik

```
$uname -r
2.6.11-1.1369_FC4
```

itt

```
$ ps axHo comm,pid,ppid,stat,tid,nlwp
COMMAND           PID  PPID STAT   TID NLWP
bash            20077  2386 Ss   20077    1
szerver         20115 20077 Sl+  20115    6
szerver         20115 20077 Sl+  20116    6
szerver         20115 20077 Sl+  20117    6
szerver         20115 20077 Sl+  20118    6
szerver         20115 20077 Sl+  20119    6
szerver         20115 20077 Sl+  20120    6
```

Futtasuk újra a szervert és nézzünk szét a Proc-ban is:

```
$ ls -l  /proc/20115/task/
total 0
dr-xr-xr-x  4 norbi norbi 0 Nov  2 11:36 20115
dr-xr-xr-x  4 norbi norbi 0 Nov  2 11:36 20116
dr-xr-xr-x  4 norbi norbi 0 Nov  2 11:36 20117
dr-xr-xr-x  4 norbi norbi 0 Nov  2 11:36 20118
dr-xr-xr-x  4 norbi norbi 0 Nov  2 11:36 20119
dr-xr-xr-x  4 norbi norbi 0 Nov  2 11:36 20120

$ more /proc/20115/status
Name:     szerver
State:    S (sleeping)
```

```
SleepAVG: 78%
Tgid:     20115
Pid: 20115
PPid:     20077
.
.
.
Threads:        6
.
.
.
```

VI.1.1.7 Párhuzamos, kölcsönös kizárással accept-elő szálak sorával

Az előre elkészített szálak kölcsönös kizárással hívják az `accept()`-et, a 66. oldalon, a *Pthread-es mutex zárak* című fejezetben megismert módon:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <pthread.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define SZERVER_SZALAK_SOR_MERET 5
#define CTIME_BUFFER_MERET 128
pthread_mutex_t accept_zar;
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
void *
kiszolgalo_szal(void *kapu_figyelo)
{
  int kapcsolat, kliensm;
  struct sockaddr_in kliens;
  for(;;)
  {
    memset((void *) &kliens, 0, (kliensm = sizeof(kliens)));
    pthread_mutex_lock(&accept_zar);
    if((kapcsolat = accept(*(int *)kapu_figyelo,
      (struct sockaddr *)&kliens, (socklen_t *)&kliensm)) == -1)
    {
      perror("accept");
      exit(EXIT_FAILURE);
    }
    pthread_mutex_unlock(&accept_zar);
    printf("     <-> %s:%d\n",
    inet_ntoa(kliens.sin_addr), ntohs(kliens.sin_port));
    if(kiszolgal(kapcsolat) == -1)
    {
```

```
      perror("kiszolgal");
    }
    close(kapcsolat);
  }
}
int
main(void)
{
  pthread_t szalak[SZERVER_SZALAK_SOR_MERET];
  int kapu_figyelo, kliensm, gyermekem_pid, sockoptval=1, i;
  struct sockaddr_in szerver, kliens;
  memset((void *)&szerver, 0, sizeof(szerver));
  szerver.sin_family= AF_INET;
  inet_aton("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port= htons(SZERVER_PORT);
  if((kapu_figyelo = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE);
  }
  setsockopt(kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
    (void *)&sockoptval, sizeof(sockoptval));
  if(bind(kapu_figyelo, (struct sockaddr *)&szerver,
    sizeof(szerver)) == -1)
  {
    perror("bind");
    exit(EXIT_FAILURE);
  }
  if((listen(kapu_figyelo, SZERVER_SOR_MERET)) == -1)
  {
    perror("listen");
    exit(EXIT_FAILURE);
  }
  printf("%s:%d\n",
    inet_ntoa(szerver.sin_addr), ntohs(szerver.sin_port));
  for(i=0; i<SZERVER_SZALAK_SOR_MERET; ++i)
  {
    if(pthread_create(&szalak[i], NULL, kiszolgalo_szal,
      (void *)&kapu_figyelo))
    {
      error("pthread_create");
      exit(EXIT_FAILURE);
    }
  }
  for(;;)
    pause();
}
```

VI.1.1.8 Soros, IO multiplexeléssel

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
int
main(void)
{
  int kapu_figyelo, kapcsolat, kliensm, sockoptval=1, s;
  fd_set kapu_figyelok;
  struct timeval timeout;
  struct sockaddr_in szerver, kliens;
  memset((void *)&szerver, 0, sizeof(szerver));
  szerver.sin_family= AF_INET;
  inet_aton("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port= htons(SZERVER_PORT);
  if((kapu_figyelo = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP))
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE);
  }
  setsockopt(kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
     (void *)&sockoptval, sizeof(sockoptval));
  fcntl(kapu_figyelo, F_SETFL,
    fcntl(kapu_figyelo, F_GETFL) | O_NONBLOCK);
  if(bind(kapu_figyelo, (struct sockaddr *)&szerver,
    sizeof(szerver)) == -1)
  {
    perror("bind");
    exit(EXIT_FAILURE);
  }
  if(listen(kapu_figyelo, SZERVER_SOR_MERET) == -1)
  {
    perror("listen");
    exit(EXIT_FAILURE);
  }
  printf("%s:%d\n",
    inet_ntoa(szerver.sin_addr), ntohs(szerver.sin_port));
  FD_ZERO(&kapu_figyelok);
  for(;;)
  {
    FD_SET(kapu_figyelo, &kapu_figyelok);
    timeout.tv_sec=3;
    timeout.tv_usec=0;
```

```
    if((s=select(kapu_figyelo+1, &kapu_figyelok, NULL,
       NULL, &timeout)) == -1)
    {
      perror("select");
      exit(EXIT_FAILURE);
    }
    else if(!s)
    {
      printf("vartam...\n");
      fflush(stdout);
    }
    else
    {
      if(FD_ISSET(kapu_figyelo, &kapu_figyelok))
      {
        memset((void *) &kliens, 0, (kliensm = sizeof(kliens)));
        if((kapcsolat = accept(kapu_figyelo,
          (struct sockaddr *)&kliens,
          (socklen_t *)&kliensm)) == -1)
        {
          perror("accept");
          exit(EXIT_FAILURE);
        }
        printf("     <-> %s:%d\n",
          inet_ntoa(kliens.sin_addr), ntohs(kliens.sin_port));
        fflush(stdout);
        if(kiszolgal(kapcsolat) == -1)
        {
          perror("kiszolgal");
        }
        close(kapcsolat);
      }
    }
  }
}
```

Az `FD_ISSET()` itt biztos nem nullát adna vissza, mivel a példa túl egyszerű, de a teljesség kedvéért használjuk. Kicsit bonyolultabb esetben, azaz például egy visszhang jellegű szervernél már kéne.

VI.1.1.9 Párhuzamos, IO multiplexeléssel

Dolgozzuk most össze az iménti példát a 106. oldal, *Párhuzamos, azaz konkurens, folyamatokkal* című fejezet programjával:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
```

```
#include <signal.h>
#include <errno.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
void
zombi_elharito (int sig)
{
  signal (SIGCHLD, zombi_elharito);
  while (wait (NULL) > 0)
    ;
}
int
main (void)
{
  int kapu_figyelo, kapcsolat, kliensm, sockoptval = 1, s,
gyermekem_pid;
  fd_set kapu_figyelok;
  struct timeval timeout;
  struct sockaddr_in szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP))
      == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  setsockopt (kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
          (void *) &sockoptval, sizeof (sockoptval));
  fcntl (kapu_figyelo, F_SETFL, fcntl (kapu_figyelo, F_GETFL)
         | O_NONBLOCK);
  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
        sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (listen (kapu_figyelo, SZERVER_SOR_MERET) == -1)
    {
      perror ("listen");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr),
          ntohs (szerver.sin_port));
  signal (SIGCHLD, zombi_elharito);
  FD_ZERO (&kapu_figyelok);
  for (;;)
```

```
    {
      FD_SET (kapu_figyelo, &kapu_figyelok);
      timeout.tv_sec = 3;
      timeout.tv_usec = 0;
      if ((s = select (kapu_figyelo + 1, &kapu_figyelok, NULL,
                 NULL, &timeout)) == -1)
     {
       switch (errno)
         {
         case EBADF:
           printf ("EBADF\n");
           break;
         case EINTR:
           printf ("EINTR\n");
           break;
         case EINVAL:
           printf ("EINVAL\n");
           break;
         case ENOMEM:
           printf ("ENOMEM\n");
           break;
         }
       fflush (stdout);
       //exit (EXIT_FAILURE);
     }
      else if (!s)
     {
       printf ("vartam...\n");
       fflush (stdout);
     }
      else
     {
       if (FD_ISSET (kapu_figyelo, &kapu_figyelok))
         {

           memset ((void *) &kliens, 0, (kliensm =
                    sizeof (kliens)));
           if ((kapcsolat = accept (kapu_figyelo,
                            (struct sockaddr *) &kliens,
                            (socklen_t *) & kliensm)) == -1)
          {
            perror ("accept");
            exit (EXIT_FAILURE);
          }
           printf ("     <-> %s:%d\n",
                inet_ntoa (kliens.sin_addr),
                           ntohs (kliens.sin_port));

           if ((gyermekem_pid = fork ()) == 0)
          {
            close (kapu_figyelo);
            if (kiszolgal (kapcsolat) == -1)
              {
                perror ("kiszolgal");
              }
            close (kapcsolat);
            exit (EXIT_SUCCESS);
          }
```

```
           else if (gyermekem_pid > 0)
           {
             // wait(&statusz); e miatt kezeljuk a SIGCHLD jelet,
             // l. a Zombik fejezetet!
             close (kapcsolat);
           }
           else
           {
             close (kapcsolat);
             perror ("fork");
             exit (EXIT_FAILURE);
           }
         }
      }
}
```

Futtassuk, majd tegyünk egy `sleep(5);`-öt a `kiszolgal()`-ba és így próbáljuk ki:

```
$ ./szerver
127.0.0.1:2005
vartam...
     <-> 127.0.0.1:35913
vartam...
EINTR
```

Nézzünk utána a hibának:

```
man select
```

A problémát az okozza, hogy a `SIGCHLD` jel megszakítja a `select()` rendszerhívást. Ezért kommenteztük ki a kilépést. Használjuk fel a 43. oldal, *Sigaction* című fejezetében megismerteket a `sa.sa_flags = SA_RESTART;` beállítással.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <signal.h>
#include <errno.h>
#define SZERVER_PORT 2005
#define SZERVER_SOR_MERET 10
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
```

```
  char *ts = ctime_r(&t, buffer);
  sleep (5);
  return write(kliens, ts, strlen(ts));
}
void
zombi_elharito (int sig)
{
  while (wait (NULL) > 0)
    ;
}
int
main (void)
{
  int kapu_figyelo, kapcsolat, kliensm, sockoptval = 1, s,
gyermekem_pid;
  fd_set kapu_figyelok;
  struct timeval timeout;
  struct sockaddr_in szerver, kliens;
  struct sigaction sa;
  sa.sa_handler = zombi_elharito;
  sigemptyset (&sa.sa_mask);
  sa.sa_flags = SA_RESTART;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP))
      == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  setsockopt (kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
           (void *) &sockoptval, sizeof (sockoptval));
  fcntl (kapu_figyelo, F_SETFL, fcntl (kapu_figyelo, F_GETFL)
         | O_NONBLOCK);
  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
         sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (listen (kapu_figyelo, SZERVER_SOR_MERET) == -1)
    {
      perror ("listen");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr),
          ntohs (szerver.sin_port));
  sigaction (SIGCHLD, &sa, NULL);
  FD_ZERO (&kapu_figyelok);
  for (;;)
    {
      FD_SET (kapu_figyelo, &kapu_figyelok);
      timeout.tv_sec = 3;
      timeout.tv_usec = 0;
      if ((s = select (kapu_figyelo + 1, &kapu_figyelok, NULL,
                  NULL, &timeout)) == -1)
```

```
    {
      switch (errno)
        {
        case EBADF:
          printf ("EBADF\n");
          break;
        case EINTR:
          printf ("EINTR\n");
          break;
        case EINVAL:
          printf ("EINVAL\n");
          break;
        case ENOMEM:
          printf ("ENOMEM\n");
          break;
        }
      fflush (stdout);
      //exit (EXIT_FAILURE);
    }
     else if (!s)
    {
      printf ("vartam...\n");
      fflush (stdout);
    }
     else
    {
      if (FD_ISSET (kapu_figyelo, &kapu_figyelok))
        {

          memset ((void *) &kliens, 0, (kliensm =
                  sizeof (kliens)));
          if ((kapcsolat = accept (kapu_figyelo,
                           (struct sockaddr *) &kliens,
                           (socklen_t *) & kliensm)) == -1)
         {
           perror ("accept");
           exit (EXIT_FAILURE);
         }
          printf ("     <-> %s:%d\n",
                inet_ntoa (kliens.sin_addr),
                ntohs (kliens.sin_port));

          if ((gyermekem_pid = fork ()) == 0)
         {
           close (kapu_figyelo);
           if (kiszolgal (kapcsolat) == -1)
             {
               perror ("kiszolgal");
             }
           close (kapcsolat);
           exit (EXIT_SUCCESS);
         }
          else if (gyermekem_pid > 0)
         {
           // wait(&statusz); e miatt kezeljuk a SIGCHLD jelet,
           // l. a Zombik fejezetet!
           close (kapcsolat);
         }
```

```
            else
            {
              close (kapcsolat);
              perror ("fork");
              exit (EXIT_FAILURE);
            }
          }
      }
    }
}
```

VI.1.1.10 Összefoglalás

Akinek a socket programozás felkeltette az érdeklődését, annak további mély olvasmányként a [*Hálózati*/UNP] könyvet ajánljuk, azon belül is az alapozó fejezeteken túl a 727. oldalon kezdődő kliens-szerver tervezési alternatívák című fejezetet, ami illeszkedik a jelen feldolgozáshoz is.

VI.1.1.10.1 A wildcard cím – azaz ne csak a localhostról

Módosítsuk úgy a szervert, hogy elérjük egymását, azaz ne csak a localhostról tudjuk tesztelni őket! Az alábbi megszokott sort

```
inet_aton("127.0.0.1", &(szerver.sin_addr));
```

cseréljük le erre, a :

```
szerver.sin_addr.s_addr = htonl(INADDR_ANY);
```

Sikerült, ha szerverünk így indul:

```
0.0.0.0:2005
```

IPv6 esetén a wildcard cím megadása:

```
szerver.sin6_addr = in6addr_any;
```

ekkor:

```
:::2005
```

VI.1.1.10.2 Nem szálbiztos függvények

```
man ctime
man inet_ntoa
```

```
#define CTIME_BUFFER_MERET 128
int
kiszolgal(int kliens)
{
```

```
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time(NULL);
  char *ts = ctime_r(&t, buffer);
  return write(kliens, ts, strlen(ts));
}
```

VI.1.1.11 Java nyelven a szerveroldal

Java nyelven egy remek példát olvashatunk a [Java/WS] cikkben egy egyszerű többszálú szerverről. Nagyon jól olvasható a kód és szép gyakorlati példát mutat a `wait()` és a `notify()` használatára. Fix darab kiszolgáló szállal indul, ha ezek mind dolgoznak, akkor a bejövő kérések kiszolgálására új szálakat készít.

Az alábbi két példán (az első soros, a második párhuzamos) csak a kapcsolódó szerveroldali TCP-s Java osztályokat mutatjuk be:

```
public class EgyszeruSzerver {
    public static void main(String [] args) {
        try {
            java.net.ServerSocket serverSocket =
                    new java.net.ServerSocket(2005);
            while(true) {
                java.net.Socket socket = serverSocket.accept();
                java.io.PrintWriter kimenoCsatorna =
                        new java.io.PrintWriter(socket.
                                                getOutputStream());
                kimenoCsatorna.println(
                        new java.util.Date().toString());
                kimenoCsatorna.flush();
                socket.close();
            }
        } catch(java.io.IOException ioE) {
            ioE.printStackTrace();
        }
    }
}
```

A következő példa a kérések kiszolgálását külön szálban végzi, de többszálúság tekintetében korántsem olyan szép és hatékony példa, mint az imént említett [Java/WS] hivatkozás.

```
class KiszolgaloSzal implements Runnable {
    java.net.Socket socket;
    public KiszolgaloSzal(java.net.Socket socket) {
        this.socket = socket;
        new Thread(this).start();
    }
    public void run() {
        try {
            java.io.BufferedReader bejovoCsatorna =
                    new java.io.BufferedReader(
                    new java.io.InputStreamReader(socket.
                                                getInputStream()));
            java.io.PrintWriter kimenoCsatorna =
                    new java.io.PrintWriter(socket.
```

```
                                                    getOutputStream());
            String viszhang = bejovoCsatorna.readLine();
            kimenoCsatorna.println(viszhang);
            kimenoCsatorna.flush();
            socket.close();
        } catch(java.io.IOException ioE) {
            ioE.printStackTrace();
        }
    }
}
public class EgyszeruTobbszaluSzerver {
    public static void main(String [] args) {
        try {
            java.net.ServerSocket serverSocket =
                    new java.net.ServerSocket(2005);
            while(true) {
                java.net.Socket socket = serverSocket.accept();
                new KiszolgaloSzal(socket);
            }
        } catch(java.io.IOException ioE) {
            ioE.printStackTrace();
        }
    }
}
```

VI.1.1.11.1 Multiplexelt, nem blokkolódó Java szerver

```
public class MultiplexeltSzerver {
 public static void main(String [] args) {
   try {
    java.nio.channels.ServerSocketChannel szerverSocketCsatorna
                    = java.nio.channels.ServerSocketChannel.open();
    szerverSocketCsatorna.configureBlocking(false);
    java.net.ServerSocket szerverSocket
                    = szerverSocketCsatorna.socket();
    szerverSocket.bind(new java.net.InetSocketAddress(2005));
    java.nio.channels.Selector kliensSzelektor
                    = java.nio.channels.Selector.open();
    szerverSocketCsatorna.register(kliensSzelektor,
                    java.nio.channels.SelectionKey.OP_ACCEPT);
    while(true) {
     kliensSzelektor.select();
     java.util.Set<java.nio.channels.SelectionKey> kliensek
                        = kliensSzelektor.selectedKeys();
     for(java.nio.channels.SelectionKey kliens : kliensek) {
      if(kliens.isAcceptable()) {// biztos az
       java.nio.channels.ServerSocketChannel serverSocket
         = (java.nio.channels.ServerSocketChannel)kliens.channel();
       java.net.Socket socket = serverSocket.accept().socket();
       java.io.PrintWriter kimenoCsatorna =
         new java.io.PrintWriter(socket.getOutputStream());
       kimenoCsatorna.println(new java.util.Date().toString());
       kimenoCsatorna.flush();
       socket.close();
```

```
      }
     }
    }
   } catch(Exception e) {
           e.printStackTrace();
   }
 }
}
```

VI.1.1.12 C# szerveroldal

```
using System;
using System.Net;
using System.Net.Sockets;
using System.IO;
public class Szerver
{
  static void Main (String[]args)
  {
    try
    {
      IPAddress localhost = IPAddress.Parse ("127.0.0.1");
      TcpListener tcpListener
         = new TcpListener (localhost, 2005);
      tcpListener.Start ();
      while (true)
     {
       TcpClient socket = tcpListener.AcceptTcpClient ();
       StreamWriter sw = new StreamWriter (socket.GetStream ());
       sw.WriteLine (DateTime.Now);
       sw.Flush ();
       socket.Close ();
     }
    }
    catch (Exception e)
    {
      Console.WriteLine (e);
    }
  }
}
```

VI.1.2 Socket kliensek

A klienseket a korábban fejlesztett szerverekkel (lásd a *102*. oldaltól) próbálhatjuk ki, de ha más gépekről is akarunk tesztelni, akkor ne felejtsük az *129*. oldalon javasolt `INADDR_ANY` (wildcard cím) módosítást a C szerverekben. Javasoljuk, hogy a fentiekben bemutatott mindenféle – C, Java, C# – szerverekhez próbáljunk ki minden, az alábbiakban bemutatásra kerülő – C, Java, C# – klienst!

VI.1.2.1 C socket kliens

IPv4 és IPv6 példát is mutatunk.

VI.1.2.1.1 IPv4

```
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 256
int
main (void)
{
  int kapu, olvasva;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET];
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (connect (kapu, (struct sockaddr *) &szerver,
               sizeof (szerver)) == -1)
    {
      perror ("connect");
      exit (EXIT_FAILURE);
    }
  while ((olvasva = read (kapu, buffer, BUFFER_MERET)) > 0)
    write (1, buffer, olvasva);
  exit (EXIT_SUCCESS);
}
```

VI.1.2.1.2 IPv6 kliens oldal

```
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define SZERVER_PORT 2005
#define PUFFER_MERET 256
int
main (void)
{
  int kapu, olvasva;
  struct sockaddr_in6 szerver;
  char puffer[PUFFER_MERET];
  char buffer[INET6_ADDRSTRLEN];
```

```
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin6_family = AF_INET6;
  szerver.sin6_flowinfo = 0;
  inet_pton (AF_INET6, "::FFFF:C0A8:0101", &(szerver.sin6_addr));
  szerver.sin6_port = htons (SZERVER_PORT);
  if ((kapu = socket (PF_INET6, SOCK_STREAM, IPPROTO_TCP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (connect (kapu, (struct sockaddr *) &szerver,
               sizeof (szerver)) == -1)
    {
      perror ("connect");
      exit (EXIT_FAILURE);
    }
  while ((olvasva = read (kapu, puffer, PUFFER_MERET)) > 0)
    write (1, puffer, olvasva);
  exit (EXIT_SUCCESS);
}
```

VI.1.2.2 Java socket kliens

```
import java.net.*;
import java.io.*;
public class Kliens {
  public static void main(String[] args) {
    try {
      Socket socket = new Socket("127.0.0.1", 2005);
      BufferedReader br =
        new BufferedReader(
          new InputStreamReader(socket.getInputStream()));
      System.out.println(br.readLine());
      socket.close();
    } catch (Exception e) {
      System.out.println(e);
    }
  }
}
```

VI.1.2.3 C# socket kliens

```
using System;
using System.Net.Sockets;
using System.IO;
public class Kliens3 {
  static void Main(String[] args)
  {
    try
    {
      TcpClient tcpClient = new TcpClient("127.0.0.1", 2005);
```

```
      StreamReader sr = new StreamReader(tcpClient.GetStream());
      Console.WriteLine(sr.ReadLine());
      tcpClient.Close();
    }
    catch (Exception e)
    {
      Console.WriteLine(e);
    }
  }
}
```

VI.1.2.4 Feladat – socket opciók

A párhuzamos TCP szerverek rész (106. oldal) első programjának kiszolgáló rutinjába tegyünk be egy `for(;;);` utasítást! Hogyan viselkednek ekkor a kliensek ? (Például a 132. oldal kliensei?) Majd figyeljük meg, hogyan viselkedik az alábbi módosítás?

```
import java.net.*;
import java.io.*;
public class Kliens {
  public static void main(String[] args) {
    try {
      Socket socket = new Socket("127.0.0.1", 2005);
      socket.setSoTimeout(5000);
      BufferedReader br =
        new BufferedReader(
          new InputStreamReader(socket.getInputStream()));
      System.out.println(br.readLine());
      socket.close();
    } catch (SocketTimeoutException stE) {
      System.out.println(stE);
    } catch (Exception e) {
      System.out.println(e);
    }
  }
}
```

VI.1.2.5 Feladat – SMTP levél

Ebben a feladatban az egyszerű levéltovábbítási protokollal ismerkedünk meg, az alábbi bevágásban a `neumann.inf.unideb.hu` helyett `localhost`-ot használjunk!

```
$ telnet neumann.inf.unideb.hu 25
Trying 193.6.135.20...
Connected to neumann.inf.unideb.hu.
Escape character is '^]'.
220 neumann.inf.unideb.hu ESMTP Postfix (Debian/GNU)
HELO inf.unideb.hu
250 neumann.inf.unideb.hu
MAIL FROM: <nbatfai@inf.unideb.hu>
250 Ok
```

```
RCPT TO: <nbatfai@inf.unideb.hu>
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: nbatfai@inf.unideb.hu
To: nbatfai@inf.unideb.hu
Subject: Proba level
Hello!

Ez egy proba level.

Norbi
.
250 Ok: queued as ECC381FC15
QUIT
221 Bye
Connection closed by foreign host.
```

Tehát az iménti sorok mintájára mi a `localhost`-al dolgozzunk:

```
$ telnet localhost 25
```

Címzetként a `localhost` egy felhasználóját adjuk meg! Ha kész, akkor a `mail` programmal biztosan el tudjuk olvasni:

```
$ mail
```

VI.1.2.5.1 SMTP levél Java-ban

Csináljuk meg ugyanezt Java programból! Induljunk ki a *134*. oldal *Java socket kliens* című pontjának programjából:

```
import java.net.*;
import java.io.*;
public class SmtpLevel
{
  public static void main (String[]args)
  {
    if (args.length != 4)
      {
     System.err.println ("java SmtpLevel felado cimzett targy
uzenet");
     System.exit (-1);
      }
    try
    {
      Socket socket = new Socket ("127.0.0.1", 25);
      BufferedReader br
        = new BufferedReader (
                  new InputStreamReader (socket.getInputStream ()));
      PrintWriter pw
```

```
        = new PrintWriter (socket.getOutputStream (), true);
      System.out.println (br.readLine ());
      pw.println ("HELO inf.unideb.hu");
      System.out.println (br.readLine ());
      pw.println ("MAIL FROM: <" + args[0] + ">");
      System.out.println (br.readLine ());
      pw.println ("RCPT TO: <" + args[1] + ">");
      System.out.println (br.readLine ());
      pw.println ("DATA");
      System.out.println (br.readLine ());
      pw.println ("From: " + args[0]);
      pw.println ("To: " + args[1]);
      pw.println ("Subject: " + args[2]);
      pw.println (args[3]);
      pw.println (".");
      System.out.println (br.readLine ());
      pw.println ("QUIT");
      System.out.println (br.readLine ());
      socket.close ();
    }
    catch (Exception e)
    {
      System.out.println (e);
    }
  }
}
```

VI.1.2.5.2 SMTP levél C-ben

Most pedig C programból! Induljunk ki a *132*. oldal *C socket kliens* című pontjának programjából:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define SZERVER_PORT 25
#define BUFFER_MERET 4096
int
main (int argc, char **argv)
{
  int kapu, olvasva;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET];
  if (argc != 5)
    {
      printf ("smtp_level felado cimzett targy uzenet");
      exit (EXIT_FAILURE);
    }
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
```

```
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (connect (kapu, (struct sockaddr *) &szerver, sizeof
(szerver)) == -1)
    {
      perror ("connect");
      exit (EXIT_FAILURE);
    }
  olvasva = read (kapu, buffer, BUFFER_MERET);
  write (1, buffer, olvasva);

  snprintf(buffer, BUFFER_MERET, "HELO %s\n", "inf.unideb.hu");
  write (kapu, buffer, strlen (buffer));
  olvasva = read (kapu, buffer, BUFFER_MERET);
  write (1, buffer, olvasva);

  snprintf(buffer, BUFFER_MERET, "MAIL FROM: <%s>\n", argv[1]);
  write (kapu, buffer, strlen (buffer));
  olvasva = read (kapu, buffer, BUFFER_MERET);
  write (1, buffer, olvasva);

  snprintf(buffer, BUFFER_MERET, "RCPT TO: <%s>\n", argv[2]);
  write (kapu, buffer, strlen (buffer));
  olvasva = read (kapu, buffer, BUFFER_MERET);
  write (1, buffer, olvasva);

  snprintf(buffer, BUFFER_MERET, "DATA\n");
  write (kapu, buffer, strlen (buffer));
  olvasva = read (kapu, buffer, BUFFER_MERET);
  write (1, buffer, olvasva);

  snprintf(buffer, BUFFER_MERET,
    "From: %s\nTo: %s\nSubject: %s\n%s\n", argv[1], argv[2],
    argv[3], argv[4]);
  write (kapu, buffer, strlen (buffer));
  snprintf(buffer, BUFFER_MERET, ".\n");
  write (kapu, buffer, strlen (buffer));
  olvasva = read (kapu, buffer, BUFFER_MERET);
  write (1, buffer, olvasva);

  snprintf(buffer, BUFFER_MERET, "QUIT\n");
  write (kapu, buffer, strlen (buffer));
  olvasva = read (kapu, buffer, BUFFER_MERET);
  write (1, buffer, olvasva);

  exit (EXIT_SUCCESS);
}
```

VI.1.2.5.3 Csevegő Javaban

Az alábbiakban elkezdjük el egy TCP-s csevegő példa fejlesztését.

```
class CsevegőSzoba {
```

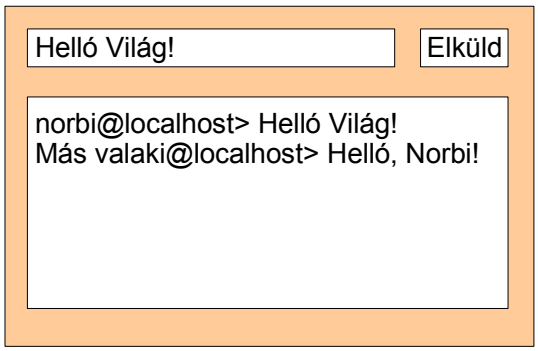
```
    private CsevegőSzál[] csevegőSzoba;

    public CsevegőSzoba(int méret) {

        csevegőSzoba = new CsevegőSzál[méret];

        for(int i=0; i<csevegőSzoba.length; ++i)
            csevegőSzoba[i] = new CsevegőSzál(this);

    }

    public CsevegőSzál belép() {

        for(int i=0; i<csevegőSzoba.length; ++i)
            if(csevegőSzoba[i].szabad())
                return csevegőSzoba[i];

        return null;
    }

    public void mindenkinek(String üzenet) {

        for(int i=0; i<csevegőSzoba.length; ++i)
            csevegőSzoba[i].üzenet(üzenet);

    }
}

class CsevegőSzál implements Runnable {

    CsevegőSzoba csevegőSzoba;
    private boolean szabad = true;
    private java.net.Socket socket;
    private java.io.PrintWriter kimenoCsatorna;

    public CsevegőSzál(CsevegőSzoba csevegőSzoba) {
        this.csevegőSzoba = csevegőSzoba;
        new Thread(this).start();
    }

    public boolean szabad() {
        return szabad;
    }

    public synchronized void belép(java.net.Socket socket) {
        this.socket = socket;
        szabad = false;
        notify();
    }

    public void üzenet(String üzenet) {
        if(!szabad) {
            kimenoCsatorna.println(üzenet);
            kimenoCsatorna.flush();
        }
    }
```

```
    public synchronized void run() {

        for(;;) {

            szabad = true;
            try{
                wait();
            } catch(InterruptedException e) {}

            String nick = socket.getInetAddress().getHostName()
            + ":"
            + socket.getPort()
            +"> ";

            try {
                java.io.BufferedReader bejovoCsatorna =
                        new java.io.BufferedReader(
                        new java.io.InputStreamReader(socket.
                        getInputStream()));
                kimenoCsatorna =
                        new java.io.PrintWriter(socket.
                        getOutputStream());

                String üzenet = bejovoCsatorna.readLine();
                do {
                    if("ki".equals(üzenet))
                        break;

                    csevegőSzoba.mindenkinek(nick+üzenet);
                } while((üzenet = bejovoCsatorna.readLine()) !=
null);

                socket.close();

            } catch(java.io.IOException ioE) {
                ioE.printStackTrace();
            }
        }
    }
}

public class CsevegőSzerver {

    public static void main(String [] args) {

        CsevegőSzoba csevegőSzoba =
                new CsevegőSzoba(25);

        try {
            java.net.ServerSocket serverSocket =
                    new java.net.ServerSocket(2006);
            while(true) {
                java.net.Socket socket = serverSocket.accept();

                CsevegőSzál csevegőSzál = csevegőSzoba.belép();

                if(csevegőSzál != null)
                    csevegőSzál.belép(socket);
```

```
                else {
                    java.io.PrintWriter kimenoCsatorna =
                            new java.io.PrintWriter(socket.
                            getOutputStream());
                    kimenoCsatorna.println("A csevegő szoba tele
van.");
                    kimenoCsatorna.println("Később próbálkozz
újra!");
                    kimenoCsatorna.flush();
                    socket.close();
                }
            }
        } catch(java.io.IOException ioE) {
            ioE.printStackTrace();
        }
    }
}
```

VI.1.2.5.3.a Swinges csevegő kliens

A *183*. oldal *ProgPáter TCP kliens* című pontja alapján készítsünk egy grafikus klienst az előző pontban fejlesztett szerverhez. A felületet bonyolultsága legalább az alábbi skicc mutatta legyen:



14. ábra: A csevegő kliens felülete

VI.1.2.5.3.b Multiplexelt, nem blokkolódó csevegő szerver

Fejlesszük tovább csevegő szerverünket, most egy multiplexelt, nem blokkolódó változatot készítsünk! A jelentkezők számát, szemben az előző példával most ne korlátozzuk. A választott megoldási módból következően most eltekinthetünk a szálak használatától.

Egyfajta bemelegítésként a C programozók a *121*. oldal, a *Soros, IO multiplexeléssel*, vagy a *123*. oldal, a *Párhuzamos, IO multiplexeléssel* című részeket nézhetik át, ahol ugyancsak multiplexelt, nem blokkolódó szervert készítettünk, bár az ezekben a pontokban tárgyalt „daytime szerver” a jelen csevegő szerverünkhöz képes egy fokozattal egyszerűbb.

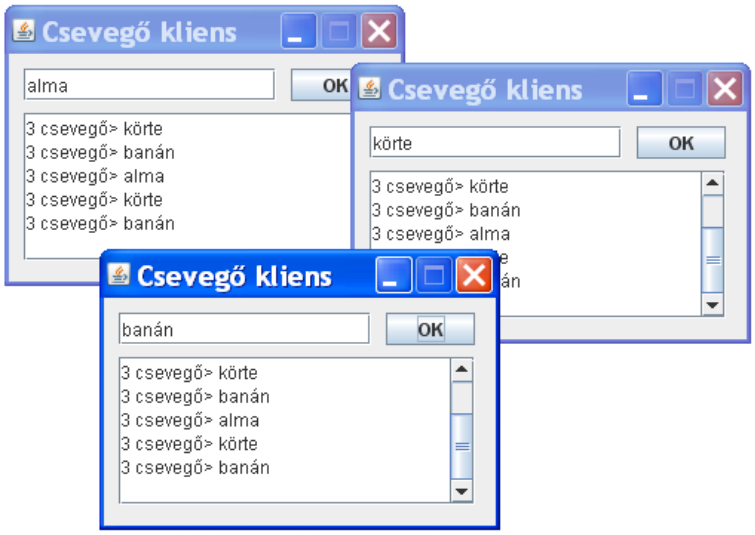
Induljunk ki a *131*. oldal, a *Multiplexelt, nem blokkolódó Java szerver* című pont programjából!

```
public class MultiplexeltCsevegő {
    public static void main(String [] args) {
```

```
        java.util.List<java.nio.channels.SocketChannel>
csevegőSzoba
                = new
java.util.ArrayList<java.nio.channels.SocketChannel>();

        try {
            java.nio.channels.ServerSocketChannel
szerverSocketCsatorna
                    =
java.nio.channels.ServerSocketChannel.open();
            szerverSocketCsatorna.configureBlocking(false);
            java.net.ServerSocket szerverSocket
                    = szerverSocketCsatorna.socket();
            szerverSocket.bind(new
java.net.InetSocketAddress(2006));
            java.nio.channels.Selector kliensSzelektor
                    = java.nio.channels.Selector.open();
            szerverSocketCsatorna.register(kliensSzelektor,
                    java.nio.channels.SelectionKey.OP_ACCEPT);

            while(true) {

                kliensSzelektor.select();

                java.util.Iterator it =
kliensSzelektor.selectedKeys().iterator();
                while(it.hasNext()) {
                    java.nio.channels.SelectionKey kliens
                            = (java.nio.channels.SelectionKey)
it.next();
                    it.remove();

                    if(kliens.isAcceptable()) {

                        java.nio.channels.ServerSocketChannel
serverSocketCsatorna
                                =
(java.nio.channels.ServerSocketChannel)kliens.channel();
                        java.nio.channels.SocketChannel
socketCsatorna
                                = serverSocketCsatorna.accept();
                        socketCsatorna.configureBlocking(false);
                        socketCsatorna.register(kliensSzelektor,
                                java.nio.channels.SelectionKey.O
P_READ);

                       csevegőSzoba.add(socketCsatorna);

                    } else if(kliens.isReadable()) {

                        java.nio.channels.SocketChannel
socketCsatorna
                                =
(java.nio.channels.SocketChannel)kliens.channel();

                        byte [] buffer = (csevegőSzoba.size()+"
csevegő> ").getBytes();
```

```
                        java.nio.ByteBuffer nioBuffer
                                =
java.nio.ByteBuffer.allocate(512);
                        nioBuffer.clear();
                        nioBuffer.put(buffer);

                        try {

                            if(socketCsatorna.read(nioBuffer) ==
-1) {

                                csevegőSzoba.remove(socketCsator
na);
                                socketCsatorna.close();

                            } else {

                                nioBuffer.flip();
                                for(java.nio.channels.SocketChan
nel k: csevegőSzoba)
                                    k.write(nioBuffer.duplicate(
));
                            }

                        } catch(java.io.IOException e) {
                            socketCsatorna.close();
                            System.out.println(e);
                        }
                    }
                }
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Egy kliens a multiplexelt, nem blokkolódó csevegő szerverhez



15. ábra: A multiplexelt, nem blokkolódó csevegő szerver Swinges kliensei

```
class HálózatiSzál {

    SwingKliens swingKliens;
    java.net.Socket socket;
    java.io.BufferedReader bejövőcsatorna;
    java.io.PrintWriter kimenoCsatorna;

    public HálózatiSzál(SwingKliens swingKliens) {

        this.swingKliens = swingKliens;
        try {

            socket = new java.net.Socket("127.0.0.1", 2006);
            bejövőcsatorna =
                    new java.io.BufferedReader(
                    new
java.io.InputStreamReader(socket.getInputStream()));

            kimenoCsatorna =
                    new java.io.PrintWriter(socket.
                    getOutputStream());

            olvas();

        } catch (Exception e) {
            System.out.println(e);
        }

    }

    public void kilép() {

        try {
```

```
            socket.close();
        } catch (Exception e) {
            System.out.println(e);
        }

    }

    public void olvas() {
        javax.swing.SwingWorker worker =
                new javax.swing.SwingWorker<String, Object>() {
            public String doInBackground() {
                String sor = null;
                try {
                    sor = bejövőcsatorna.readLine();
                } catch (Exception e) {
                    sor = e.getMessage();
                }
                return sor;
            }
            public void done() {
                try {
                    swingKliens.válasz(get());
                } catch(Exception e) {}

                olvas();
            }
        };
        worker.execute();

    }

    public void ír(String üzenet) {
        kimenoCsatorna.println(üzenet);
        kimenoCsatorna.flush();
    }

}

public class SwingKliens extends javax.swing.JFrame {

    javax.swing.JTextArea valaszTextArea;
    HálózatiSzál hálózatiSzál;

    public SwingKliens() {
        super("Csevegő kliens");
        setResizable(false);
        setDefaultCloseOperation(javax.swing.WindowConstants.EXI
T_ON_CLOSE);
        addWindowListener(
                new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent
w) {
                hálózatiSzál.kilép();
            }
        });
        getContentPane().setLayout(null);
        java.awt.Dimension kepernyo
```

```
                =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((int)kepernyo.getWidth()/2-270/2,
                (int)kepernyo.getHeight()/2-190/2, 270, 190);
        final javax.swing.JTextField hosztTextField
                = new javax.swing.JTextField();
        hosztTextField.setText("felhasználói input");
        hosztTextField.setBounds(10, 10, 170, 22);
        hosztTextField.setToolTipText("felhasználói input");
        getContentPane().add(hosztTextField);
        javax.swing.JButton okButton
                = new javax.swing.JButton();
        okButton.setText("OK");
        okButton.setBounds(190, 10, 60, 22);
        okButton.addActionListener(
                new  java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent
                    e) {
                hálózatiSzál.ír(hosztTextField.getText());
            }
        });
        getContentPane().add(okButton);
        valaszTextArea
                = new javax.swing.JTextArea();
        javax.swing.text.DefaultCaret dc =
                new javax.swing.text.DefaultCaret();
        dc.setUpdatePolicy(
                javax.swing.text.DefaultCaret.ALWAYS_UPDATE);
        valaszTextArea.setCaret(dc);
        javax.swing.JScrollPane valaszScrollPane
                = new javax.swing.JScrollPane();
        valaszScrollPane.setViewportView(valaszTextArea);
        valaszScrollPane.setBounds(10, 40, 240, 100);
        getContentPane().add(valaszScrollPane);
        setVisible(true);

        hálózatiSzál = new HálózatiSzál(this);

    }

    public void válasz(String válasz) {
        valaszTextArea.append(válasz+"\n");
    }

    public static void main(String [] args) {
        new SwingKliens();
    }
}
```

VI.1.2.5.4 Csevegő C-ben

VI.1.2.6 UDP szerver, kliens

```
$ man 7 udp
```

A szerver megírásánál a *102*. oldal *Soros, azaz iteratív* című pontjának programjából indulunk ki, azt alakítjuk át:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 1024
int
main (void)
{
  int kapu_figyelo, kliensm;
  char buffer[BUFFER_MERET];
  time_t t;
  struct sockaddr_in szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP))
       == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
          sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr),
          ntohs (szerver.sin_port));
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));

      if (recvfrom (kapu_figyelo, buffer, BUFFER_MERET, 0,
              (struct sockaddr *) &kliens, (socklen_t *)
               & kliensm) < 0)
      {
       perror ("recvfrom");
       exit (EXIT_FAILURE);
      }
      printf ("     <-> %s:%d\n",
           inet_ntoa (kliens.sin_addr),
```

```
            ntohs (kliens.sin_port));
      t = time (NULL);
      ctime_r (&t, buffer);
      if (sendto (kapu_figyelo, buffer, strlen (buffer), 0,
            (struct sockaddr *) &kliens, (socklen_t) kliensm)
          < 0)
      {
       perror ("sendto");
       exit (EXIT_FAILURE);
      }
    }
}
```

A kliens elkészítésénél a *132*. oldal *C socket kliens* című pontjának programjából indulunk ki, azt alakítjuk át:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 1024
int
main (void)
{
  int kapu, olvasva, szerverm;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET] = "Mennyi az ido?";
  memset ((void *) &szerver, 0, (szerverm = sizeof (szerver)));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (sendto (kapu, buffer, strlen (buffer), 0,
            (struct sockaddr *) &szerver, (socklen_t) szerverm)
      < 0)
    {
      perror ("sendto");
      exit (EXIT_FAILURE);
    }
  if ((olvasva = recvfrom (kapu, buffer, BUFFER_MERET, 0,
                  (struct sockaddr *) &szerver,
                  (socklen_t *) & szerverm)) < 0)
    {
      perror ("recvfrom");
      exit (EXIT_FAILURE);
    }
  printf("%s", buffer);
```

```
  exit (EXIT_SUCCESS);
}
```

Tesztelés közben figyeljük a statisztikát:

```
$ netstat -suc
.
.
.
Udp:
    550 packets received
    3 packets to unknown port received.
    0 packet receive errors
    553 packets sent
.
.
.
```

azaz miközben futtatjuk a szervert:

```
$ ./szerver
127.0.0.1:2005
     <-> 127.0.0.1:32773
```

és a klienst:

```
$ ./kliens
Sun Nov 27 23:19:15 2005
```

Egy ebből a példából származtatott UDP IPC-s példát a *49*. oldalon, a *Feladat – ugyanez UDP-vel* című pontban találunk.

VI.1.2.6.1 Feladat - játék az UDP szerverrel-klienssel

Módosítsuk úgy az előző példát, hogy a kliens sorszámozza meg a csomagokat, a szerver pedig vizsgálja, hogy milyen sorrendben kapja meg azokat, és mindez viszont: a szerver számozza a válaszait, a kliens ezeket vizsgálja!

Néhány apró kiegészítéssel előáll a szerver:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 64
int
main (void)
{
  int kapu_figyelo, kliensm, elozo=-1, sorszam;
  char buffer[BUFFER_MERET];
  struct sockaddr_in szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  szerver.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP))
       == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }

  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
          sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr),
          ntohs (szerver.sin_port));
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));

      if (recvfrom (kapu_figyelo, buffer, BUFFER_MERET, 0,
              (struct sockaddr *) &kliens, (socklen_t *)
          & kliensm) < 0)
      {
       perror ("recvfrom");
       exit (EXIT_FAILURE);
      }
      sorszam = atoi(buffer);
      if(sorszam != elozo+1)
        printf("sorszam nem stimmel: %d", sorszam);
      elozo = sorszam;
      if (sendto (kapu_figyelo, buffer, strlen (buffer), 0,
            (struct sockaddr *) &kliens, (socklen_t) kliensm)
          < 0)
      {
       perror ("sendto");
       exit (EXIT_FAILURE);
      }
    }
}
```

Hasonlóan a kliens is:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 64
int
main (void)
{
  int kapu, olvasva, szerverm, i, sorszam;
```

```
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET] = "Mennyi az ido?";
  memset ((void *) &szerver, 0, (szerverm = sizeof (szerver)));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  for (i = 0; i < 100; ++i)
    {
      sprintf (buffer, "%d\n", i);
      if (sendto (kapu, buffer, strlen (buffer), 0,
            (struct sockaddr *) &szerver, (socklen_t) szerverm)
          < 0)
      {
       perror ("sendto");
       exit (EXIT_FAILURE);
      }
      if ((olvasva = recvfrom (kapu, buffer, BUFFER_MERET, 0,
                      (struct sockaddr *) &szerver,
                      (socklen_t *) & szerverm)) < 0)
      {
       perror ("recvfrom");
       exit (EXIT_FAILURE);
      }
      if((sorszam = atoi(buffer)) != i)
                    printf("sorszam nem stimmel: %d", sorszam);
    }
  exit (EXIT_SUCCESS);
}
```

Mivel csak akkor íratunk ki, ha borul a sorrend, de mégis szeretnénk informálódni a program futásáról, használjuk a

```
$ netstat -us
```

parancsot. Teszteljük a kliens-szervert a `localhost`-on és két különböző gépen is! Az eredmények értékelése kapcsán olvassuk el a [*Hálózati*/TB] „Vezeték nélküli TCP és UDP” fejezetét, annak is a végét.

VI.1.2.6.1.a A nem megbízhatóság szimulálása

Szimuláljuk most egy szerveroldali „válasz csomag” eltünését! Veszítsük el a 18-asal sorszámozott csomagunkra a választ, azaz az alábbihoz hasonlóan módosítsuk a szerver forrását:

```
     if (sorszam != 18)
        if (sendto (kapu_figyelo, buffer, strlen (buffer), 0,
                    (struct sockaddr *) &kliens, (socklen_t)
                      kliensm) < 0)
```

Mi történik? Elemezzük!

Módosítsuk úgy a klienst, hogy csak egy időkorlát átlépéséig blokkolódjon a `recvfrom()` rendszerhívásnál! Azaz szereljük fel az iménti klienst a *43*. oldalon a *A riasztás alkalmazása* című pontban megismert funkcionalitással:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <signal.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 64
void
idotullepes (int sig)
{
  printf ("\nAz adott idokorlaton belul nem jott meg a
csomag!\n");
}
int
main (void)
{
  int kapu, olvasva, szerverm, i, sorszam;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET] = "Mennyi az ido?";
  struct sigaction sa;
  sa.sa_handler = idotullepes;
  sigemptyset (&sa.sa_mask);
  memset ((void *) &szerver, 0, (szerverm = sizeof (szerver)));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  sigaction (SIGALRM, &sa, NULL);
  for (i = 0; i < 100; ++i)
    {
      sprintf (buffer, "%d\n", i);
      if (sendto (kapu, buffer, strlen (buffer), 0,
            (struct sockaddr *) &szerver, (socklen_t) szerverm)
          < 0)
      {
       perror ("sendto");
       exit (EXIT_FAILURE);
      }
      alarm (3);
      if ((olvasva = recvfrom (kapu, buffer, BUFFER_MERET, 0,
                      (struct sockaddr *) &szerver,
                      (socklen_t *) & szerverm)) < 0)
      {
       perror ("recvfrom");
```

```
      }
      alarm (0);
      if ((sorszam = atoi (buffer)) != i)
         printf ("sorszam nem stimmel: %d", sorszam);
      printf ("%d ", atoi (buffer));
      fflush(stdout);
    }
  exit (EXIT_SUCCESS);
}
```

Ekkor a 18. as csomag elvesztése már nem blokkolja örökre a kommunikációt:

```
$ ./kliens
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Az adott idokorlaton belul nem jott meg a csomag!
recvfrom: Interrupted system call
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
86 87 88 89 90 91 92 93 94 95 96 97 98 99
```

Hasonló „szimulációval” próbáljuk ki a példa program viselkedését a csomagok megkettőződésekor.

A szállítás megbízhatóságának intuitív képéhez közelebb áll a programok alábbi egyszerűsítése:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 64
int
main (void)
{
  int kapu_figyelo, kliensm, elozo = -1, sorszam;
  char buffer[BUFFER_MERET];
  struct sockaddr_in szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  szerver.sin_addr.s_addr = htonl (INADDR_ANY);
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP))
       == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }

  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
          sizeof (szerver)) == -1)
    {
```

```
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr),
          ntohs (szerver.sin_port));
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));

      if (recvfrom (kapu_figyelo, buffer, BUFFER_MERET, 0,
              (struct sockaddr *) &kliens, (socklen_t *) &
           kliensm) < 0)
      {
       perror ("recvfrom");
       exit (EXIT_FAILURE);
      }
      sorszam = atoi (buffer);
      printf ("%d\n", sorszam);
      if (sorszam != elozo + 1)
          printf ("sorszam nem stimmel: %d", sorszam);
      elozo = sorszam;
    }
}
```

ennek megfelelően a kliens:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 64
int
main (void)
{
  int kapu, szerverm, i;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET];
  memset ((void *) &szerver, 0, (szerverm = sizeof (szerver)));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu = socket (PF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  for (i = 0; i < 100; ++i)
    {
      sprintf (buffer, "%d\n", i);
      if (sendto (kapu, buffer, strlen (buffer), 0,
            (struct sockaddr *) &szerver, (socklen_t) szerverm)
          < 0)
      {
```

```
       perror ("sendto");
       exit (EXIT_FAILURE);
      }
    }
  exit (EXIT_SUCCESS);
}
```

VI.1.2.6.2 UDP szerver, kliens Java-ban

Készítsük el az iménti pontokban mutatott C-beli szerver és kliens Java változatait:

```
public class Szerver {
  public static final int BUFFER_MERET = 1024;
  public static void main(String []args) {
    try{
        java.net.DatagramSocket kapu
              = new java.net.DatagramSocket(2005);
        while(true) {
          byte[] buffer = new byte[BUFFER_MERET];
          java.net.DatagramPacket fogadottCsomag
              = new java.net.DatagramPacket(buffer, buffer.length);
          kapu.receive(fogadottCsomag);
          int port = fogadottCsomag.getPort();
          java.net.InetAddress hoszt
              = fogadottCsomag.getAddress();
          System.out.println(new String(fogadottCsomag.getData(),
                        0, fogadottCsomag.getLength()));
          System.out.println(hoszt.getHostAddress()+":"+port);
          String ido = new java.util.Date().toString();
          byte[] puffer = ido.getBytes();
          System.arraycopy(puffer, 0, buffer, 0, puffer.length);
          java.net.DatagramPacket kuldendoCsomag
             = new java.net.DatagramPacket(buffer, puffer.length,
                        hoszt, port);
          kapu.send(kuldendoCsomag);
        }
    } catch(Exception e) {
            e.printStackTrace();
    }
  }
}
```

A kliens oldal pedig:

```
public class Kliens {
  public static final int BUFFER_MERET = 1024;
  public static void main(String []args) {
    try{
        java.net.DatagramSocket kapu
             = new java.net.DatagramSocket();
        byte[] buffer = new byte[BUFFER_MERET];
        String ido = "MennyiAzIdo?";
        byte[] puffer = ido.getBytes();
        System.arraycopy(puffer, 0, buffer, 0, puffer.length);
```

```
        java.net.InetAddress hoszt
             = java.net.InetAddress.getByName("localhost");
        java.net.DatagramPacket kuldendoCsomag
             = new java.net.DatagramPacket(buffer, puffer.length,
                  hoszt, 2005);
        kapu.send(kuldendoCsomag);
        java.net.DatagramPacket fogadandoCsomag
             = new java.net.DatagramPacket(buffer, buffer.length);
        kapu.receive(fogadandoCsomag);
        int port = fogadandoCsomag.getPort();
        hoszt = fogadandoCsomag.getAddress();
        System.out.println(hoszt.getHostAddress()+":"+port);
        System.out.println(new String(fogadandoCsomag.getData(),
                  0, fogadandoCsomag.getLength()));
    } catch(Exception e) {
            e.printStackTrace();
    }
  }
}
```

Fordítás után futtatva a szervert:

```
$ javac Szerver.java Kliens.java
$ java Szerver
MennyiAzIdo?
127.0.0.1:32772
MennyiAzIdo?
127.0.0.1:32773
```

és a klienst:

```
$ java Kliens
127.0.0.1:2005
Mon Nov 28 16:32:04 CET 2005
```

VI.2 Raw socketek

```
$ man 7 raw
```

VI.2.1 ping C-ben

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip_icmp.h>
#include <netinet/ip.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
int
main ()
{
  int kapu_figyelo;
  struct icmp icmp_echo;
  struct icmp *icmp_echoreply;
  struct ip *ip;
  int icmp_seqc = 0;
  int pid = getpid ();
  char buffer[2048];
  int v, len, gyermekem_pid;
  struct sockaddr_in szerver, kliens;
  memset ((void *) &szerver, 0, sizeof (szerver));
  icmp_echo.icmp_type = ICMP_ECHO;
  icmp_echo.icmp_code = 0;
  icmp_echo.icmp_cksum = 0;
  icmp_echo.icmp_id = pid;
  icmp_echo.icmp_seq = icmp_seqc++;
  if ((kapu_figyelo = socket (PF_INET, SOCK_RAW,
          IPPROTO_ICMP)) == -1)
    {
      perror ("socket");
      exit (-1);
    }
  if ((gyermekem_pid = fork ()) == 0)
    {
      for (;;)
     {
       if ((v = sendto (kapu_figyelo, (void *) &icmp_echo,
                   sizeof (icmp_echo), 0,
                   (struct sockaddr *) &szerver,
                   sizeof (szerver))) == -1)
         {
           perror ("sendto");
           exit (-1);
         }
       sleep (1);
       icmp_echo.icmp_seq = icmp_seqc++;
```

```
     }
    }
  else if (gyermekem_pid > 0)
    {
      for (;;)
     {
       if ((v = recvfrom (kapu_figyelo, (void *) buffer,
                    sizeof (buffer), 0,
                    (struct sockaddr *)&kliens,
                     (socklen_t *) & len)) == -1)
         {
           perror ("recvfrom");
           exit (-1);
         }
       ip = (struct ip *) buffer;
       icmp_echoreply = (struct icmp *)
                              (buffer + (ip->ip_hl << 2));
       if (icmp_echoreply->icmp_type == ICMP_ECHOREPLY
           && icmp_echoreply->icmp_id == pid)
         {
           printf ("%d bajt ICMP_ECHOREPLY a %s-rol seq: %d ttl:
%d\n",
                   v - (ip->ip_hl << 2),
                   inet_ntoa (kliens.sin_addr),
                   icmp_echoreply->icmp_seq, ip->ip_ttl);
         }
     }
    }
  else
    {
      perror ("fork");
      exit (-1);
    }
}
```

```
# ./ping
28 bajt ICMP_ECHOREPLY a 127.0.0.1-rol seq: 0 ttl: 64
28 bajt ICMP_ECHOREPLY a 127.0.0.1-rol seq: 1 ttl: 64
28 bajt ICMP_ECHOREPLY a 127.0.0.1-rol seq: 2 ttl: 64
28 bajt ICMP_ECHOREPLY a 127.0.0.1-rol seq: 3 ttl: 64
28 bajt ICMP_ECHOREPLY a 127.0.0.1-rol seq: 4 ttl: 64
28 bajt ICMP_ECHOREPLY a 127.0.0.1-rol seq: 5 ttl: 64
28 bajt ICMP_ECHOREPLY a 127.0.0.1-rol seq: 6 ttl: 64
.
.
.
```

Építsünk be egy egy másodperces időtúllépés figyelést az `ICMP_ECHOREPLY`-re várakozásba!

Parancsok:
`ping, ifconfig, traceroute, route`

VI.2.1.1.1.a ping C#-ban

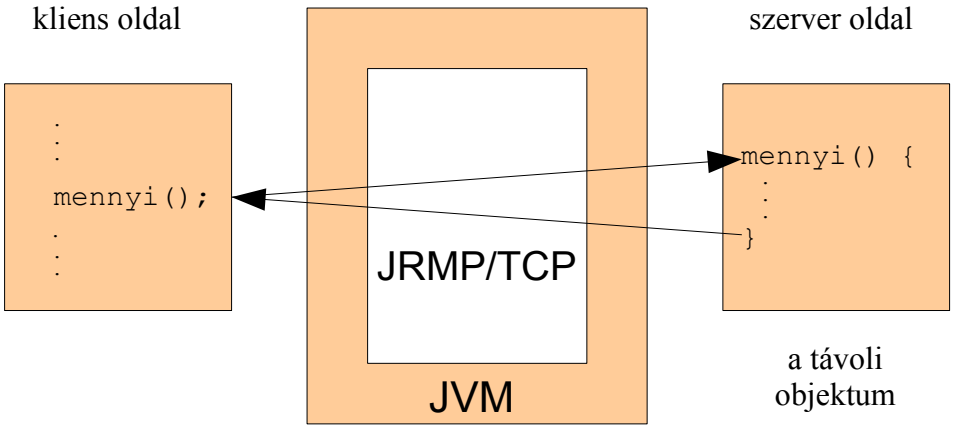
VI.3 Távoli eljáráshívás

VI.3.1 RPC

VI.3.2 JAX-RPC

VI.3.3 Java RMI

Szokásos állatorvosi példánkat, a daytime jellegű szerverünket – ami nem csinál mást, mint küldi a rendszeridőt a jelentkező klienseknek – készítsük most el Java RMI környezetben!



Ábra 16: Java távoli metódushívás

A távoli objektumot a következő interfészen keresztül látjuk:

```
public interface Ido extends java.rmi.Remote {
  public String mennyi() throws java.rmi.RemoteException;
}
```

megvalósítása: némi – a kipróbálgatását segítő – logolás és az idő visszaadása.

```
public class IdoImpl implements Ido {
  public String mennyi() {
    try {
          System.out.println(java.rmi.server.RemoteServer.getClient
Host());
    } catch(java.rmi.server.ServerNotActiveException e) {
        e.printStackTrace();
    }
    return new java.util.Date().toString();
  }
}
```

A távoli objektumot az alábbi szerver hozza létre és `MennyiAzIdo` néven bejegyzi az RMI registry-be:

```
public class IdoSzerver {
  public IdoSzerver() {
    try {
          IdoImpl idoImpl = new IdoImpl();
```

```
            Ido ido = (Ido)
java.rmi.server.UnicastRemoteObject.exportObject(idoImpl, 0);
            java.rmi.registry.Registry registry =
                        java.rmi.registry.LocateRegistry.getRegistry();
            registry.bind("MennyiAzIdo", ido);
        } catch (java.rmi.AlreadyBoundException be) {
            be.printStackTrace();
        } catch (java.rmi.RemoteException re) {
            re.printStackTrace();
        }
    }
    public static void main(String args[]) {
      new IdoSzerver();
    }
}
```

A következő klienssel vesszük igénybe a távoli `Ido` objektum `mennyi()` szolgáltatását:

```
public class IdoKliens {
  public static void main(String args[]) {
    try {
          java.rmi.registry.Registry registry =
                  java.rmi.registry.LocateRegistry.getRegistry();
          Ido ido = (Ido) registry.lookup("MennyiAzIdo");
          System.out.println(ido.mennyi());
    } catch (java.rmi.NotBoundException be) {
        be.printStackTrace();
    } catch (java.rmi.RemoteException re) {
        re.printStackTrace();
    }
  }
}
```

Fordítsuk le a szerver oldalt:

```
$ javac Ido.java IdoImpl.java IdoSzerver.java
```

Futtassuk a névszolgáltatót:

```
$ $ rmiregistry
```

a szervert:

```
$ java IdoSzerver
```

teszteljünk a klienssel:

```
$ javac IdoKliens.java
$ java IdoKliens
Sat Nov 26 13:35:57 CET 2005
```

Másoljuk most át egy másik gépre, mondjuk a `kalapacs.eurosmobil.hu` gépre az `Ido.java` és az `IdoKliens.java` forrásokat, utóbbiban módosítsuk a szerver nevét:

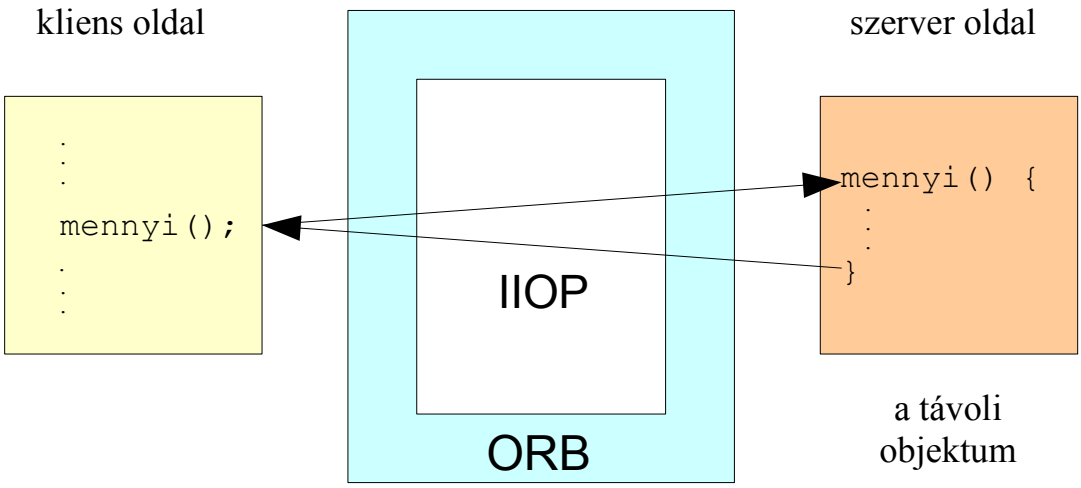
```
.
.
.
LocateRegistry.getRegistry("niobe.eurosmobil.hu");
.
.
.
```

Most a `kalapacs.eurosmobil.hu` gépről teszteljünk:

```
$ javac IdoKliens.java
$ java IdoKliens
Sat Nov 26 13:42:52 CET 2005
```

Közben figyeljük, mit logolt a szerver.

VI.3.4 CORBA



Ábra 17: CORBA távoli metódushívás

VI.3.4.1 ORBit

VI.3.4.1.1 Névszolgáltatás

```
#include <stdio.h>
#include <orb/orbit.h>
#include <ORBitservices/CosNaming.h>
int main(int argc, char **argv)
{
  CORBA_ORB orb;
  CORBA_Environment env;
  CosNaming_NamingContext nevszolgaltato_gyokere;

  char * gyokerIOR =
"IOR:010000002800000049444c3a6f6d672e6f72672f436f734e616d696e672
f4e616d696e67436f6e746578743a312e300001000000caaedfba50000000010
10000290000002f746d702f6f726269742d6e6f7262692f6f72622d333032303
739333334313930313837323131390000000018000000000000000cea82e05564
d120e010000007818b0abce930163";

  orb = CORBA_ORB_init (&argc, argv, "orbit-local-orb", &env);

  // ezt tennenk szokasosan, de futtassuk az orbit-name-server-t
  // nevszolgaltato_gyokere =
  //    CORBA_ORB_resolve_initial_service(orb,
  //                                      "NameService", &env);
  // es az igy megszerzett IOR-el ferjunk hozza a
```

```
  // nevszolgaltatohoz:
  nevszolgaltato_gyokere = CORBA_ORB_string_to_object(orb,
                                                gyokerIOR, &env);
  printf("A nevszolgaltato gyokere: %s\n",
      CORBA_ORB_object_to_string(orb, nevszolgaltato_gyokere,
                                 &env));

  return 0;
}
```

Fordítsuk így:

```
$ gcc -Wall `orbit-config --cflags server --libs` -o nevszolg_ior
nevszolg_ior.c
```

Futtassuk:

```
$ ./nevszolg_ior
A nevszolgaltato gyokere:
IOR:010000002800000049444c3a6f6d672e6f72672f436f734e616d696e672f4e6
16d696e67436f6e746578743a312e300001000000caaedfba500000000101000029
0000002f746d702f6f726269742d6e6f7262692f6f72622d3330323037393333343
139303138373231313900000000180000000000000cea82e05564d120e01000000
7818b0abce930163
```

VI.3.4.2 Gnorba

VI.3.4.2.1 Névszolgáltatás

```
#include <stdio.h>
#include <libgnorba/gnorba.h>
#include <ORBitservices/CosNaming.h>

int main(int argc, char **argv)
{
  CORBA_ORB orb;
  CORBA_Environment env;
  CosNaming_NamingContext nevszolgaltato_gyokere;

  orb = gnome_CORBA_init ("CORBA objektumok", "0.0.1", &argc,
                          argv, GNORBA_INIT_SERVER_FUNC, &env);

  nevszolgaltato_gyokere = gnome_name_service_get ();

  printf("A nevszolgaltato gyokere: %s\n",
      CORBA_ORB_object_to_string(orb, nevszolgaltato_gyokere,
                                 &env));
  return 0;
}
```

Fordítsuk így:

```
$ gcc `gnome-config --cflags gnorba gnomeui --libs` -o nevszolg_ior
nevszolg_ior.c
```

Futtassuk:

```
$ ./nevszolg_ior
A nevszolgaltato gyokere:
IOR:010000002800000049444c3a6f6d672e6f72672f436f734e616d696e672f4e6
16d696e67436f6e746578743a312e300001000000caaedfba50000000010100002a
0000002f746d702f6f726269742d6e6f7262692f6f72622d3139363732373432333
4313535323938353037340000001800000000000000e673c5f8d630464b01000000
05adb1fb3395796e
```

VI.3.4.3 Java IDL

VI.4 Web programozás

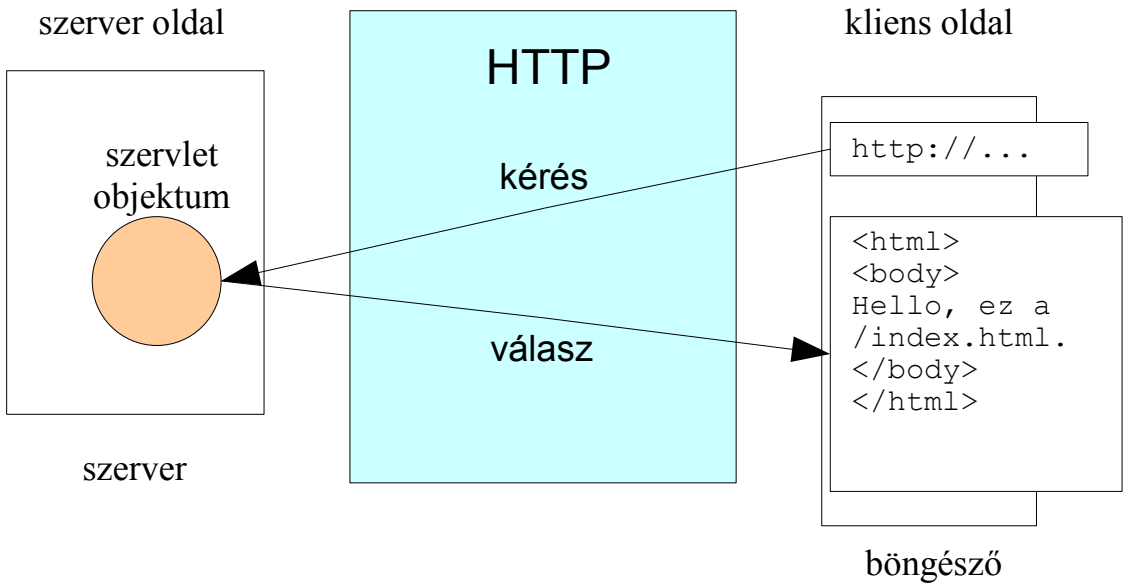
VI.4.1 A HTTP protokoll

```
1.   $ telnet niobe.eurosmobil.hu 80
2.   Trying 192.168.1.1...
3.   Connected to niobe.eurosmobil.hu (192.168.1.1).
4.   Escape character is '^]'.
5.   GET /index.html HTTP/1.0
6.
7.   HTTP/1.1 200 OK
8.   Date: Sun, 11 Dec 2005 09:11:17 GMT
9.   Server: Apache/2.0.54 (Fedora)
10.  Last-Modified: Sun, 11 Dec 2005 09:11:04 GMT
11.  ETag: "1bdb34-37-407a1ff779a00"
12.  Accept-Ranges: bytes
13.  Content-Length: 55
14.  Connection: close
15.  Content-Type: text/html; charset=UTF-8
16.
17.  <html>
18.  <body>
19.  Hello, ez a /index.html.
20.  </body>
21.  </html>
22.  Connection closed by foreign host.
```

(a)	5.	A HTTP kérés fejének **kérés sor**a
(b)	6.	A HTTP kérés fejének **kulcs:érték pár**jai (most nincsenek)
(c)		A HTTP kérés **test**e (most üres)
(d)		Elválasztó sor
(e)	7.	A HTTP válasz fejének **válasz sor**a
(f)	8-15.	A HTTP válasz fejének **kulcs:érték pár**jai
(g)	16.	Elválasztó sor
(h)	17-21.	A HTTP válasz **test**e

További olvasmányként ajánljuk: [*Hálózati*/RFC2616].

VI.4.2 Java szervletek



Ábra 18: HTTP szervletek

A Fedora Core 4 telepítő lemezen is megtalálható Tomcat-el dolgozva:

```
# cd /usr/share/tomcat5/webapps/
# mkdir prog-pater
# chown norbi:tomcat prog-pater
```

Itt fogunk felhasználóként dolgozni:

```
$ cd /usr/share/tomcat5/webapps/prog-pater
$ mkdir WEB-INF
$ mkdir WEB-INF/classes
$ joe  WEB-INF/web.xml
```

A [*Java*/TC] dokumentáció alapján a webalkalmazásunkat így írjuk le, azaz a `web.xml` fájl az alábbi:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
    <display-name>ProgPater pelda</display-name>
    <description>
        ProgPater pelda: visszajelzesek listazasa, felvitele.
    </description>
    <servlet>
      <servlet-name>visszajelzesek</servlet-name>
```

```
      <description>
        Visszajelzesek listazasa
      </description>
      <servlet-class>VisszajelzesekSzervlet</servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>visszajelzesek</servlet-name>
      <url-pattern>/lista</url-pattern>
    </servlet-mapping>
</web-app>
```

Készítsük most el a tervezett szervlet objektumot:

```
public class VisszajelzesekSzervlet
    extends javax.servlet.http.HttpServlet {

    public void doGet (javax.servlet.http.HttpServletRequest keres,
                       javax.servlet.http.HttpServletResponse valasz)
       throws java.io.IOException, javax.servlet.ServletException {
        log ("VisszajelzesekSzervlet doGet() elindult\n");
        valasz.setContentType ("text/html");
        java.io.PrintWriter szovegesCsatorna = valasz.getWriter ();
        szovegesCsatorna.println ("<html>");
        szovegesCsatorna.println ("<body>");
        szovegesCsatorna.println ("Ide listzzuk majd a
visszajelzeket.");
        szovegesCsatorna.println ("</body>");
        szovegesCsatorna.println ("</html>");
        szovegesCsatorna.close ();
    }
}
```

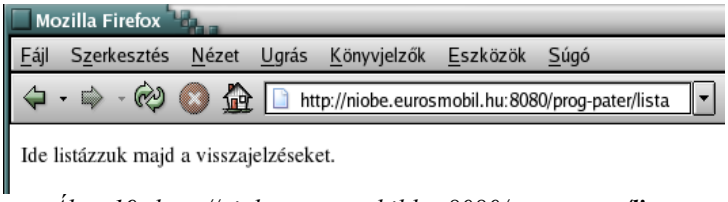
Fordítsuk le, ehhez „látni kell” a használt szervelt API implementációját, azaz:

```
$ export CLASSPATH=$CLASSPATH:/usr/share/java/servletapi5.jar
$ javac WEB-INF/classes/VisszajelzesekSzervlet.java
```

Ha még nem futna, akkor indítsuk el a Tomcat-et:

```
# /etc/rc.d/init.d/tomcat5 restart
```

ezután nézhetjük meg alkotásunkat böngészőből:



*Ábra 19: http://niobe.eurosmobil.hu:8080/prog-pater/**lista***

Ezt a szervletet fejlesztjük tovább a *204*. oldalon, az *Adatbázis programozás* című fejezet pontjaiban: mindenféle adatbázisokban/ból tároljuk/olvassuk majd az olvasói visszajelzéseket.

VI.5 Bluetooth

VI.5.1 javax.bluetooth

VII. C kernelszint

VII.1 Linux 2.6

VII.1.1 Kernelfordítás

Az [*OS*/KO] csomagban található (mindenkori) README alapján járunk el, most:

```
$ uname -r
2.6.11-1.1369_FC4
$ bzip2 -cd linux-2.6.13.4.tar.bz2 |tar xvf -
$ cd linux-2.6.13.4
$ make mrproper
$ cp /usr/src/kernels/2.6.11-1.1369_FC4-x86_64/.config .
$ make gconfig
```

A `make mrproper` takarít, törli a `.config`-ot és az esetleges korábbi fordítások eredményeit, például a `.o` fájlokat. Kiindulhatunk egy korábbi konfigurációs fájlból is, ekkor a `make oldconfig` csak az újdonságokra kérdez rá. A `make gconfig` esetén grafikusan kattintgathatunk a lehetőségek közül. Például a „*Processor type and features”* kategóriában a „*Generic-x86-64”* processzort a saját procimnak megfelelően „*AMD-Opteron/Athlon64”*-re állítom, vagy a „*Kernel hacking”* témában beállítom a „*Show timing information on printk”* opciót stb.

```
$ make
$ su
Password:
# make modules_install install
$ uname -r
2.6.13.4
```

VII.1.1.1 Kernel blog

Ebben a pontban – ahogy időm engedi – próbálok majd pár sort írni az éppen aktuális kernelfordításról.

```
$ uname -r
2.6.14
```

Többek között a *„General setup/Kernel .config support”*-ot próbáltam ki, így immár a kerneltől is el tudom kérni a konfigurációs fájlját:

```
$ ls -l /proc/config.gz
-r--r--r--  1 root root 14263 Nov  2 20:28 /proc/config.gz
```

VII.1.1.2 Linux kernel verziók

Fő verzió szám . másodlagos verzió szám . kiadás . karbantartás

Stabil a kernel, ha a másodlagos verzió szám páros, egyébként fejlesztői. A kernelt itt találjuk: [*OS*/KO]

VII.1.2 Kernelmodulok

VII.1.2.1 Az első kernelmodulok

```
#include <linux/module.h>
#include <linux/init.h>
MODULE_DESCRIPTION("Ez az elso kernel modulom");
MODULE_AUTHOR("Bátfai Norbert (norbi@javacska.hu)");
MODULE_LICENSE("GPL");
static int elso_init_module(void)
{
     printk("Elso modul init\n" );
     return 0;
}
static void elso_exit_module(void)
{
     printk("Elso modul exit\n" );
}
module_init(elso_init_module);
module_exit(elso_exit_module);
```

A `Makefile`:

```
obj-m += elso.o
all:
     make -C /lib/modules/`uname -r`/build M=`pwd` modules
clean:
     make -C /lib/modules/`uname -r`/build M=`pwd` clean
     rm *~
```

Készítsük el a modult:

```
$ make
make -C /lib/modules/`uname -r`/build M=`pwd` modules
make[1]: Entering directory `/home/norbi/k14/linux-2.6.14'
  CC [M]  /home/norbi/OS/LKM/1/elso.o
  Building modules, stage 2.
  MODPOST
  CC      /home/norbi/OS/LKM/1/elso.mod.o
  LD [M]  /home/norbi/OS/LKM/1/elso.ko
make[1]: Leaving directory `/home/norbi/k14/linux-2.6.14'
```

Kipróbálni root-ként tudjuk:

```
# /sbin/insmod elso.ko
Elso modul init
# /sbin/rmmod elso.ko
Elso modul exit
```

Ha nem a konzolon dolgoztunk, hanem például távolról, akkor a dmesg paranccsal láthatjuk az üzeneteket:

```
# dmesg
.
.
.
[ 1807.008923] Elso modul init
```

Nézzük meg a modulunk infóit:

```
$ /sbin/modinfo elso.ko
filename:       elso.ko
license:        GPL
author:         Bátfai Norbert (norbi@javacska.hu)
description:    Ez az elso kernel modulom
srcversion:     3E2AFABD180FEC1040AA456
depends:
vermagic:       2.6.14 gcc-4.0
```

Parancsok:
`lsmod, insmod, rmmod, modinfo, modprobe`

Állományok:
`/proc/modules, /var/log/messages`

VII.1.2.1.1 printk

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
MODULE_DESCRIPTION("Ez a masodik kernel modulom");
MODULE_AUTHOR("Bátfai Norbert (norbi@javacska.hu)");
MODULE_LICENSE("GPL");
static int masodik_init_module(void)
{
        printk(KERN_NOTICE "Masodik modul init\n" );
        return 0;
}
static void masodik_exit_module(void)
{
        printk(KERN_NOTICE "Masodik modul exit\n" );
}
module_init(masodik_init_module);
module_exit(masodik_exit_module);
```

Próbáljuk ki:

```
# /sbin/insmod masodik.ko
Masodik modul init
# /sbin/rmmod masodik.ko
Masodik modul exit
```

Nézzük meg a `linux/kernel.h`-ban [*OS*/KO] a `printk()`-ban használható logolási szinteket!

VII.1.2.1.1.a console_loglevel

```
$ more /proc/sys/kernel/printk
6       4       1       7
```

VII.1.2.1.2 struct task_struct

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/list.h>
MODULE_DESCRIPTION ("Ez a harmadik kernel modulom");
MODULE_AUTHOR ("Bátfai Norbert (norbi@javacska.hu)");
MODULE_LICENSE ("GPL");
static int
harmadik_init_module (void)
{
 struct task_struct *kezdet, *task;
 kezdet = list_entry (current->tasks.next, struct task_struct,
                      tasks);
 for (task = kezdet;
 (task =
 list_entry (task->tasks.next, struct task_struct,
 tasks)) != kezdet;)
 printk (KERN_NOTICE "%s %i %lX %li %lu %lu\n", task->comm,
         task->pid, task->flags, task->state, task->sleep_avg,
         task->policy);
 return 0;
}
static void
harmadik_exit_module (void)
{
 printk (KERN_NOTICE "Harmadik modul exit\n");
}
module_init (harmadik_init_module);
module_exit (harmadik_exit_module);
```

Próbáljuk ki:

```
# /sbin/insmod harmadik.ko
.
.
.
init 1 800100 1 899998863 0
.
.
.
szerver 4466 800000 1 499980368 0
szerver 4467 800040 1 799992682 0
szerver 4468 800040 1 699992298 0
szerver 4469 800040 1 599991069 0
szerver 4470 800040 1 499989599 0
szerver 4471 800040 1 399986759 0
.
```

```
.
.
```

Irjuk meg ugyanezt a `list_for_each()` makróval:

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/sched.h>
#include <inux/list.h>
MODULE_DESCRIPTION ("Ez a harmadik kernel modulom");
MODULE_AUTHOR ("Bátfai Norbert (norbi@javacska.hu)");
MODULE_LICENSE ("GPL");
static int
harmadik_init_module (void)
{
 struct task_struct *task;
 struct list_head *p;
 list_for_each (p, current->tasks.next) {
   task = list_entry (p, struct task_struct, tasks);
   printk (KERN_NOTICE "%s %i %lX %li %lu %lu\n", task->comm,
           task->pid, task->flags, task->state, task->sleep_avg,
           task->policy);
 }
 return 0;
}
static void
harmadik_exit_module (void)
{
 printk (KERN_NOTICE "Harmadik modul exit\n");
}
module_init (harmadik_init_module);
module_exit (harmadik_exit_module);
```

Nézzük meg a `linux/sched.h`-ban [*OS*/KO], hogy milyen értékei lehetnek például a következő tagoknak: `task->state`, `task->flags`, `task->policy`.

VII.1.2.2 A Proc fájlrendszer

VII.1.2.2.1 A seq_file interfész egyszerűen

Oldjuk meg az előző feladatot, mint állatorvosi kernelpéldát, ezt az interfészt használva:

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/list.h>
#include <linux/proc_fs.h>
#include <linux/stat.h>
#include <linux/seq_file.h>
MODULE_DESCRIPTION ("Ez a negyedik kernel modulom");
MODULE_AUTHOR ("Bátfai Norbert (norbi@javacska.hu)");
```

```
MODULE_LICENSE ("GPL");
static int
taszk_lista_show (struct seq_file *m, void *v)
{
  int i = 0;
  struct task_struct *task;
  struct list_head *p;
  seq_puts (m, "sajat taszk lista (negyedik kernel modul)\n"
        "SZAMOZAS  PARANCSSOR       PID    FLAGS    STATE
POLICY\n");
  list_for_each (p, current->tasks.next)
  {
    task = list_entry (p, struct task_struct, tasks);
    seq_printf (m, "%-9i %-16s %-6i %.8lX %-6li %-6lu\n", ++i,
           task->comm, task->pid, task->flags, task->state,
           task->policy);
  }
  return 0;
}
static int
sajat_open (struct inode *inode, struct file *file)
{
  return single_open (file, taszk_lista_show, NULL);
}
static struct file_operations sajat_fajl_muveletek = {
  .owner = THIS_MODULE,
  .open = sajat_open,
  .read = seq_read,
  .llseek = seq_lseek,
  .release = single_release
};
static struct proc_dir_entry *sajat_proc;
static int
negyedik_init_module (void)
{
  struct proc_dir_entry *sajat_proc_fajl;
  if (!(sajat_proc = proc_mkdir ("sajat", &proc_root)))
    {
      remove_proc_entry ("sajat", &proc_root);
      printk (KERN_NOTICE "/proc/sajat/ letrehozas
sikertelen\n");
      return -1;
    }
  printk (KERN_NOTICE "/proc/sajat/ letrehozva\n");
  if ((sajat_proc_fajl =
       create_proc_entry ("taszk_stat", S_IFREG | S_IRUGO,
                          sajat_proc)))
    {
      sajat_proc_fajl->proc_fops = &sajat_fajl_muveletek;
      sajat_proc_fajl->owner = THIS_MODULE;
      printk (KERN_NOTICE "/proc/sajat/taszk_stat
letrehozva\n");
      return 0;
    }
  else
    {
      remove_proc_entry ("taszk_stat", sajat_proc);
      remove_proc_entry ("sajat", &proc_root);
```

```
      printk (KERN_NOTICE "/proc/sajat/taszk_stat letrehozas
sikertelen\n");
      return -1;
    }
}
static void
negyedik_exit_module (void)
{
  remove_proc_entry ("taszk_stat", sajat_proc);
  printk (KERN_NOTICE "/proc/sajat/taszk_stat torolve\n");
  remove_proc_entry ("sajat", &proc_root);
  printk (KERN_NOTICE "/proc/sajat torolve\n");
}
module_init (negyedik_init_module);
module_exit (negyedik_exit_module);
```

Próbáljuk ki:

```
# /sbin/insmod negyedik.ko
[  844.744663] /proc/sajat/ letrehozva
[  844.744682] /proc/sajat/taszk_stat letrehozva
$ more /proc/sajat/taszk_stat
sajat taszk lista (negyedik kernel modul)
SZAMOZAS  PARANCSSOR       PID    FLAGS    STATE  POLICY
1         init             1      00800100 1      0
2         ksoftirqd/0      2      00008040 1      0
3         events/0         3      00008040 1      0
4         khelper          4      00008040 1      0
5         kthread          5      00008040 1      0
.
.
.
# /sbin/rmmod negyedik.ko
[  846.595437] /proc/sajat/taszk_stat torolve
[  846.595455] /proc/sajat torolve
```

Hol láthatunk további példákat az interfész egyszerű használatra, [*OS*/KO]:
`linux/kernel/cpuset.c, linux/kernel/sched.c, linux/kernel/dma.c`

VII.1.2.2.2 A seq_file interfész kevésbé egyszerűen

VII.1.2.2.3 A `linux/fs/proc/generic.c`-ben hivatkozott hekkelés alapján

VII.1.2.2.4 A lapméretnél kevesebb adat a kernelből

VII.1.3 Rendszerhívások

Nézzük meg például a `pause()` rendszerhívás kapcsán a [*OS*/KO]-ban a `sys_pause ()` implementációját a `kernel/signal.c` forrásban, majd az `/usr/include/asm-`

`x86_64/unistd.h` headerben a `__NR_pause` deklarációját.

VII.1.3.1 A `norbi()` rendszerhívás

Bővítsük a `include/asm-x86_64/unistd.h` headert a saját rendszerhívásunkkal:

```
#define __NR_norbi      256
__SYSCALL(__NR_norbi, sys_norbi)
```

Implementáljuk a `kernel/norbi.c` forrásban a saját rendszerhívásunkat:

```
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/list.h>
asmlinkage long
sys_norbi ()
{
  struct list_head *p;
  int i = 0;
  list_for_each (p, current->tasks.next)++ i;
  printk (KERN_NOTICE "norbi a kernelben: %d folyamatot
szamoltam.\n", i);
  return i;
}
```

A `kernel/Makefile` fájlban az obj-y célok közé vegyük be a `norbi.o` tárgyat is. Majd következhet a *173*. oldalon, a *Kernelfordítás* pontba megismert kernelfordítás. A rendszer újraindítása után a felhasználói szinten is elérhetővé tesszük a kernel szinti munkánkat: a /usr/include/asm-x86_64/unistd.h headert bővítsük a saját rendszerhívásunk számával:

```
#define __NR_norbi              256
```

Hogy a felhasználók (programozók) kényelmesen tudjanak dolgozni, hozzuk létre az alábbi új header fájlt `/usr/include/sys/norbi.h`

```
#include <linux/unistd.h>
_syscall0 (int, norbi);
```

Ezek után kényelmesen használható a rendszerben az új rendszerhívásunk, először egy óvatosabb, majd egy bátrabb példában próbáljuk is ki!

```
#include <stdio.h>
#include <sys/syscall.h>
#include <errno.h>
int
main ()
{
  int s;
  printf ("Norbi mondja: %d folyamatot szamolt.\n",
```

```
            (s = syscall (256)));
  if (s == -1)
    {
      int e = errno;
      printf ("errno=%d\n", e);
      switch (e)
     {
     case EBADF:
       printf ("Elirtuk a szamat!\n");
       break;
     case ENOSYS:
       printf ("Nem implementaltuk!\n");
       break;
     }
      return -1;
    }
  return 0;
}
```

Fordítsuk, futtassuk!

```
$ gcc -o norbi norbi.c
$ ./norbi
[  755.655086] norbi a kernelben: 85 folyamatot szamoltam.
Norbi mondja: 85 folyamatot szamolt.
```

Saját rendszerhívásunk tipikus hívása így fest:

```
#include <stdio.h>
#include <sys/norbi.h>
int
main ()
{
  long n = norbi ();
  printf ("Norbi mondja: %ld folyamatot szamolt.\n", n);
  return 0;
}
```

Fordítsuk, futtassuk!

```
$ gcc -o norbi norbi2.c
$ ./norbi
[  768.235026] norbi a kernelben: 85 folyamatot szamoltam.
Norbi mondja: 85 folyamatot szamolt.
```

VIII. GUI programozás

A következő pontokban grafikus klienst készítünk a *102*. oldal, *Hálózati programozás* című fejezetének szervereihez. Mire érdemes figyelni a GUI kialakításánál? Figyeljünk arra, hogy mi történik, ha időigényes dolgokat indítunk el a felületről? Például teszteljük úgy a grafikus klienst, hogy a szerver kiszolgáló függvényébe beteszünk egy 10 másodperces alvást! Majd vizsgáljuk, hogy viselkedik ekkor a grafikus kliensünk? Az nyilván nem kívánatos, hogy e – jelen pillanatban – szándékosan beépített várakozáskor a felület lefagyjon...

A következő Swing és a GTK+ részekben lépésekben építjük fel a kliens (A Swing részben nagy lépésekben, a GTK+ részben kis lépésekben) programot, az AWT, GNOME, Java-GNOME részekben csak a felület felépítésére mutatunk példát.

VIII.1 Java

Grafikus felületű Java alkalmazások építéséhez a netBeans IDE-t [Java/NB] ajánljuk.

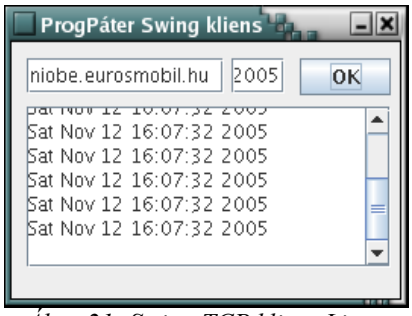
VIII.1.1 Swing

Célunk az alábbi ábrákon látható grafikus kliensek elkészítése:

VIII.1.1.1 Windows alatt futtatva

Ábra 20: Swing TCP kliens Windows alatt, a hálózati rész szervereihez.

VIII.1.1.2 Linux alatt futtatva



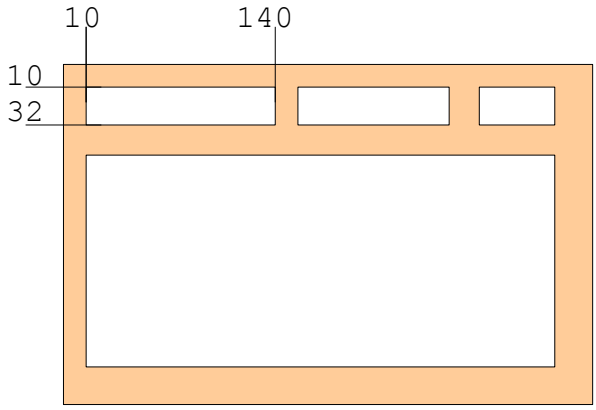
Ábra 21: Swing TCP kliens Linux alatt, a hálózati rész szervereihez.

A két képen jól láthatóa felület azonossága! A továbbiakban lássunk hozzá a program elkészítéséhez!

VIII.1.1.3 ProgPáter TCP kliens

A legtöbb Java GUI-s példában mindenféle elhelyezési stratégiák alkalmazásával ismerkedhetünk meg, ezért ebben a példában a klasszikus utat követjük: kockás lapon leskicceljük és direltben, pixelben határozzuk meg a komponensek kívánt helyét, méretét!

Építsük fel a felületet:



Ábra 22: A felület skicce

Most forrásban:

```
public class SwingKliens extends javax.swing.JFrame {
    public static void main(String [] args) {
        new SwingKliens();
    }
    public SwingKliens() {
        super("ProgPáter Swing kliens");
        setResizable(false);
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_O
N_CLOSE);
        getContentPane().setLayout(null);
        java.awt.Dimension kepernyo
            = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((int)kepernyo.getWidth()/2-270/2,
                (int)kepernyo.getHeight()/2-190/2, 270, 190);
        javax.swing.JTextField hosztTextField
                = new javax.swing.JTextField();
        hosztTextField.setText("localhost");
        hosztTextField.setBounds(10, 10, 130, 22);
        hosztTextField.setToolTipText("hoszt név, IP szám");
        getContentPane().add(hosztTextField);
        javax.swing.JTextField portTextField
                = new javax.swing.JTextField();
        portTextField.setText("2005");
        portTextField.setBounds(145, 10, 35, 22);
        portTextField.setToolTipText("port szám");
        getContentPane().add(portTextField);
        javax.swing.JButton okButton
                = new javax.swing.JButton();
        okButton.setText("OK");
        okButton.setBounds(190, 10, 60, 22);
```

```
        okButton.addActionListener(
          new  java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
 e) {
                kattint(e);
            }
        });
        getContentPane().add(okButton);
        javax.swing.JTextArea valaszTextArea
                = new javax.swing.JTextArea();
        javax.swing.JScrollPane valaszScrollPane
                = new javax.swing.JScrollPane();
        valaszScrollPane.setViewportView(valaszTextArea);
        valaszScrollPane.setBounds(10, 40, 240, 100);
        getContentPane().add(valaszScrollPane);
        setVisible(true);
    }
    private void kattint(java.awt.event.ActionEvent e) {
        System.out.println("Kattintás!");
    }
}
```

Implementáljuk az eseménykezelőt, a *134*.oldalon, a *Java socket kliens* című pontban már használt kóddal:

```
private void kattint(java.awt.event.ActionEvent esem) {
    try {
        int port = Integer.parseInt(portTextField.getText());
        java.net.Socket socket =
           new java.net.Socket(hosztTextField.getText(), port);
        java.io.BufferedReader br =
           new java.io.BufferedReader(
           new java.io.InputStreamReader(socket.getInputStream()));
        valaszTextArea.append(br.readLine()+"\n");
        socket.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

(A megjelölt komponenseket felvehetjük például példánytagoknak, hogy a `kattint()` metódusból is lássuk őket.)

Teszteljük is ezt az aktuális változatát a példánknak! Majd a szerverbe tegyünk be a kiszolgálás elé egy 10 másodperces alvást! Ekkor a felület „lefagy”, azaz nem frissül. Ezt a gomb nyomva maradt állapota is jelzi, de vonszoljuk csak ki az egérrel az ablakot a képernyő alján „ütközésig”, majd vissza! Mit tapasztalunk? Ezt:

Ábra 23: Mivel blokkoltuk az eseménykezelő szálat, így a felület átmenetileg "lefagy".

Finomítsuk úgy a programot, hogy el tudjuk kerülni a felületnek ezt a nem kívánatos viselkedését! Használjuk a `SwingWorker` osztályt [Java/SW1, SW2, SW3]. Az eseménykezelőben indítjuk a `SwingWorker` szálunkat:

A 6-os Java-tól a SwingWorker osztály már része a sztenderd Java API-nak! De a metódusnevek és a szignatúrák is megváltoztak némiképpen, ennek megfelelően a következő kód megérett az átalakításra.

```
    private void kattint(java.awt.event.ActionEvent esem) {
        int port = 2005;
        try {
            port = Integer.parseInt(portTextField.getText());
        } catch (NumberFormatException e) {
            port = 2005;
            e.printStackTrace();
        }
        new HalozatiSzal(hosztTextField.getText(),
                port, valaszTextArea).start();
    }
```

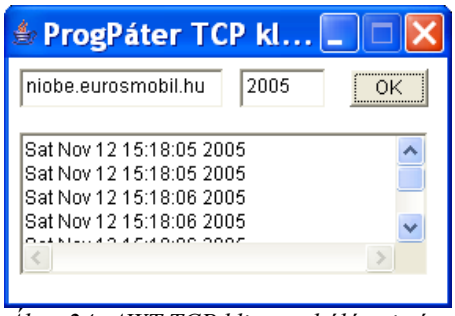
Magát a `SwingWorker` szálunkat pedig így készítjük el:

```
class HalozatiSzal extends SwingWorker {
  String hoszt;
  int port;
  javax.swing.JTextArea valaszTextArea;
  public HalozatiSzal(String hoszt, int port,
            javax.swing.JTextArea valaszTextArea) {
     this.hoszt = hoszt;
     this.port = port;
     this.valaszTextArea = valaszTextArea;
  }
  public Object construct() {
    StringBuffer valasz = new StringBuffer();
    try {
         java.net.Socket socket =
           new java.net.Socket(hoszt, port);
         java.io.BufferedReader br =
           new java.io.BufferedReader(
```

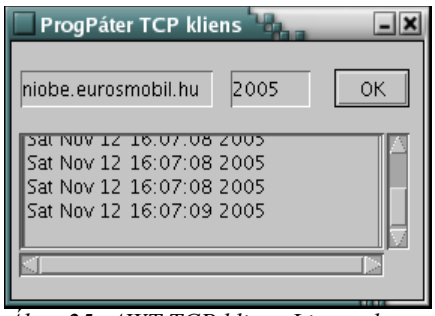
```
            new java.io.InputStreamReader(socket.getInputStream()));
          valasz.append(br.readLine());
          socket.close();
      } catch (Exception e) {
          valasz.append(e.getMessage());
      }
      return valasz.toString();
  }
  public void finished() {
      valaszTextArea.append(get()+"\n");
  }
}
```

VIII.1.2 AWT

VIII.1.2.1 ProgPáter TCP kliens



Ábra 24: AWT TCP kliens a hálózati rész szervereihez.



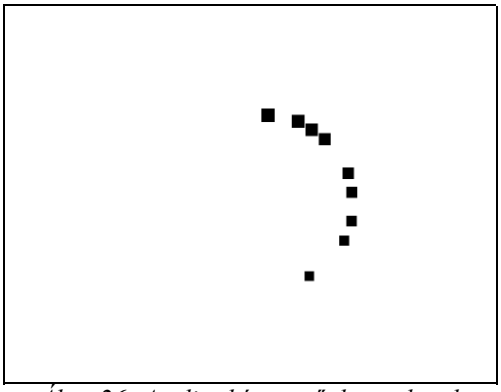
Ábra 25: AWT TCP kliens Linux alatt a hálózati rész szervereihez.

VIII.1.3 Teljes képernyő

Azaz a Java Full Screen Exclusive Mode API használata.

```
public class TeljesKepernyo extends java.awt.Frame {
    java.awt.GraphicsDevice graphicsDevice;
    java.awt.image.BufferStrategy bufferStrategy;
    int szelesseg;
    int magassag;
    public TeljesKepernyo(
            java.awt.GraphicsDevice graphicsDevice) {
        this.graphicsDevice = graphicsDevice;
        boolean fullScreenTamogatott
                = graphicsDevice.isFullScreenSupported();
        setUndecorated(true);
        setIgnoreRepaint(true);
        setResizable(true);
        if(fullScreenTamogatott)
            graphicsDevice.setFullScreenWindow(this);
        validate();
        setVisible(true);
        java.awt.DisplayMode displayMode
                = graphicsDevice.getDisplayMode();
        szelesseg = displayMode.getWidth();
        magassag = displayMode.getHeight();
        if(!fullScreenTamogatott)
            setSize(szelesseg, magassag);
        try {
            createBufferStrategy
                    (2,
                    new java.awt.BufferCapabilities
                    (new java.awt.ImageCapabilities(true),
                    new java.awt.ImageCapabilities(true),
                    java.awt.BufferCapabilities.FlipContents.BACKGR
OUND)
                    );
        } catch(java.awt.AWTException e) {
            e.printStackTrace();
        }
        bufferStrategy = getBufferStrategy();
        addMouseListener(new java.awt.event.MouseAdapter(){
            public void mousePressed(java.awt.event.MouseEvent e) {
                setVisible(false);
                TeljesKepernyo.this.graphicsDevice.setFullScreenWin
dow(null);
                System.exit(0);
            }
        });
        addMouseMotionListener(new
java.awt.event.MouseMotionAdapter(){
            public void mouseMoved(java.awt.event.MouseEvent e) {
                int ex = e.getX();
                int ey = e.getY();
                dobozok(ex, ey);
                setCursor(java.awt.Cursor.getPredefinedCursor(
                        java.awt.Cursor.HAND_CURSOR));
            }
```

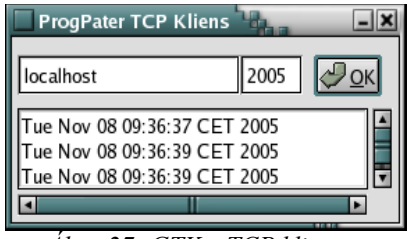
```
        });
    }
    int dobozok [] = {
        10, 10,
                20, 20,
                30, 30,
                40, 40,
                50, 50,
                60, 60,
                70, 70,
                80, 80,
                90, 90
    };
    public void dobozok(int x, int y) {
        java.awt.Graphics g = bufferStrategy.getDrawGraphics();
        if (!bufferStrategy.contentsLost()) {
            for(int i = 0; i<dobozok.length/2; ++i)
                g.fillRect(dobozok[i*2], dobozok[i*2+1], 20+i,
20+i);
            for(int i = 2; i<dobozok.length; ++i)
                dobozok[i-2] = dobozok[i];
            dobozok[dobozok.length-2] = x;
            dobozok[dobozok.length-1] = y;
            g.dispose();
            bufferStrategy.show();
        }
    }
    public static void main(String[] args) {
        java.awt.GraphicsEnvironment graphicsEnvironment
                =
java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment();
        java.awt.GraphicsDevice graphicsDevice
                = graphicsEnvironment.getDefaultScreenDevice();
        new TeljesKepernyo(graphicsDevice);
    }
}
```



Ábra 26: A teljes képernyőt közvetlenül kezeljük

VIII.2 GTK+

VIII.2.1 ProgPáter TCP kliens



Ábra 27: GTK+ TCP kliens a hálózati rész szervereihez.

(Aki a grafikus felhasználói felület kézzel való építgetését nehézkesnek találja, azoknak ajánljuk például a Glade Interface Designer programot.)

A következőkben lépésenként építsünk fel kézzel egy hasonló felületet:

```
#include <gtk/gtk.h>
#include <string.h>
gint
prog_pater_ablak_becsuk (GtkWidget * widget, GdkEvent * event,
gpointer data)
{
  gtk_main_quit ();
  return FALSE;
}
int
main (int argc, char **argv)
{
  GtkWidget *prog_pater_ablak;
  gtk_init (&argc, &argv);
  prog_pater_ablak = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title (GTK_WINDOW (prog_pater_ablak),
                "ProgPater TCP Kliens");
  gtk_window_set_position (GTK_WINDOW (prog_pater_ablak),
GTK_WIN_POS_CENTER);
  g_signal_connect (G_OBJECT (prog_pater_ablak), "delete_event",
              G_CALLBACK (prog_pater_ablak_becsuk), NULL);
  gtk_widget_show (prog_pater_ablak);
  gtk_main ();
  return 0;
}
```

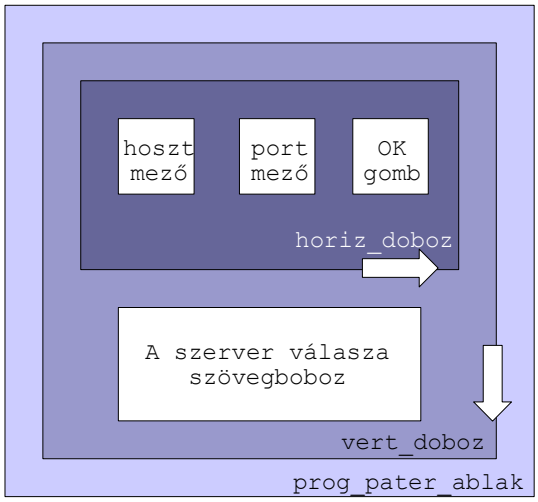
Így fordítsuk:

```
gcc -Wall `pkg-config --cflags --libs gtk+-2.0`
prog_pater_kliens.c -o prog_pater_kliens
```

Lássuk az eredményt:

Ábra 28: ProgPáter ablak első lépés

Az alábbi vázlat alapján építgessük a felületet:



Ábra 29: A ProgPáter ablak tervezése

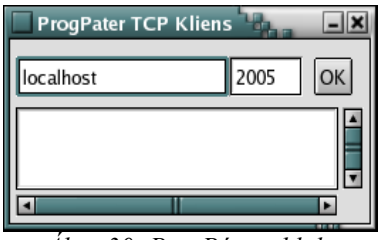
forrásban:

```
#include <gtk/gtk.h>
#include <string.h>
gint
prog_pater_ablak_becsuk (GtkWidget * widget, GdkEvent * event,
gpointer data)
{
  gtk_main_quit ();
  return FALSE;
}
int
main (int argc, char **argv)
{
```

```
  GtkWidget *prog_pater_ablak;
  GtkWidget *hoszt_textfield;
  GtkWidget *port_textfield;
  GtkWidget *valasz_szovegdoboz;
  GtkWidget *vert_doboz;
  GtkWidget *horiz_doboz;
  GtkWidget *elkuld_gomb;
  GtkWidget *gorgetosav;
  gtk_init (&argc, &argv);
  prog_pater_ablak = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title (GTK_WINDOW (prog_pater_ablak),
              "ProgPater TCP Kliens");
  gtk_window_set_resizable (GTK_WINDOW (prog_pater_ablak), FALSE);
  gtk_window_set_position (GTK_WINDOW (prog_pater_ablak),
    GTK_WIN_POS_CENTER);
  gtk_container_set_border_width (
    GTK_CONTAINER (prog_pater_ablak), 4);
  g_signal_connect (G_OBJECT (prog_pater_ablak), "delete_event",
              G_CALLBACK (prog_pater_ablak_becsuk), NULL);
  vert_doboz = gtk_vbox_new (FALSE, 0);
  gtk_container_add (GTK_CONTAINER (prog_pater_ablak),
    vert_doboz);
  horiz_doboz = gtk_hbox_new (FALSE, 0);
  gtk_container_add (GTK_CONTAINER (vert_doboz), horiz_doboz);
  hoszt_textfield = gtk_entry_new ();
  gtk_entry_set_max_length (GTK_ENTRY (hoszt_textfield), 64);
  gtk_entry_set_width_chars (GTK_ENTRY (hoszt_textfield), 20);
  gtk_entry_set_text (GTK_ENTRY (hoszt_textfield), "localhost");
  gtk_box_pack_start (GTK_BOX (horiz_doboz), hoszt_textfield,
    TRUE, TRUE, 0);
  port_textfield = gtk_entry_new ();
  gtk_entry_set_max_length (GTK_ENTRY (port_textfield), 5);
  gtk_entry_set_width_chars (GTK_ENTRY (port_textfield), 6);
  gtk_entry_set_text (GTK_ENTRY (port_textfield), "2005");
  gtk_box_pack_start (GTK_BOX (horiz_doboz), port_textfield,
    TRUE, TRUE, 0);
  elkuld_gomb = gtk_button_new_with_label ("OK");
  gtk_container_set_border_width (GTK_CONTAINER (elkuld_gomb), 8);
  gtk_box_pack_start (GTK_BOX (horiz_doboz), elkuld_gomb,
    TRUE, TRUE, 0);
  gorgetosav = gtk_scrolled_window_new (NULL, NULL);
  gtk_scrolled_window_set_shadow_type (
    GTK_SCROLLED_WINDOW (gorgetosav), GTK_SHADOW_IN);
  gtk_box_pack_start (GTK_BOX (vert_doboz), gorgetosav,
    TRUE, TRUE, 0);

  valasz_szovegdoboz = gtk_text_view_new ();
  gtk_text_view_set_editable (GTK_TEXT_VIEW (valasz_szovegdoboz),
     FALSE);
  gtk_container_add (GTK_CONTAINER (gorgetosav),
     valasz_szovegdoboz);
  gtk_widget_show (vert_doboz);
  gtk_widget_show (horiz_doboz);
  gtk_widget_show (hoszt_textfield);
  gtk_widget_show (port_textfield);
  gtk_widget_show (elkuld_gomb);
  gtk_widget_show (gorgetosav);
  gtk_widget_show (valasz_szovegdoboz);
```

```
  gtk_widget_show (prog_pater_ablak);
  gtk_main ();
  return 0;
}
```

Az előzővel megegyező módon fordítva, ezt az ablakot kapjuk:



Ábra 30: ProgPáter ablak, második lépés

Most dolgozzuk fel az `OK.` gombon való kattintást! Egyelőre visszaírjuk a hoszt nevét és a portszámot a szövegdobozbe, illetve kiírjuk a konzolra.

```
#include <gtk/gtk.h>
#include <string.h>
#define BUFFER_MERET 256
GtkWidget *hoszt_textfield;
GtkWidget *port_textfield;
GtkWidget *valasz_szovegdoboz;
gint
prog_pater_ablak_becsuk (GtkWidget * widget, GdkEvent * event,
gpointer data)
{
  gtk_main_quit ();
  return FALSE;
}
void
elkuld_gomb_benyomva (GtkWidget * widget, gpointer data)
{
  char buffer[BUFFER_MERET];
  const gchar *h = gtk_entry_get_text (GTK_ENTRY
(hoszt_textfield));
  const gchar *p = gtk_entry_get_text (GTK_ENTRY
(port_textfield));
  snprintf(buffer, BUFFER_MERET, "%s:%s\n", h, p);
  g_print(buffer);
  gtk_text_buffer_insert_at_cursor (gtk_text_view_get_buffer
                          (GTK_TEXT_VIEW (valasz_szovegdoboz)),
                         buffer, strlen (buffer));
}
int
main (int argc, char **argv)
{
  GtkWidget *prog_pater_ablak;
  GtkWidget *vert_doboz;
  GtkWidget *horiz_doboz;
  GtkWidget *elkuld_gomb;
  GtkWidget *gorgetosav;
```

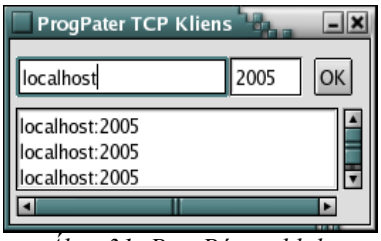
```
  gtk_init (&argc, &argv);

  prog_pater_ablak = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title (GTK_WINDOW (prog_pater_ablak),
                "ProgPater TCP Kliens");
  gtk_window_set_resizable (GTK_WINDOW (prog_pater_ablak), FALSE);
  gtk_window_set_position (GTK_WINDOW (prog_pater_ablak),
GTK_WIN_POS_CENTER);
  gtk_container_set_border_width (GTK_CONTAINER
(prog_pater_ablak), 4);

  g_signal_connect (G_OBJECT (prog_pater_ablak), "delete_event",
              G_CALLBACK (prog_pater_ablak_becsuk), NULL);

  vert_doboz = gtk_vbox_new (FALSE, 0);
  gtk_container_add (GTK_CONTAINER (prog_pater_ablak),
vert_doboz);

  horiz_doboz = gtk_hbox_new (FALSE, 0);
  gtk_container_add (GTK_CONTAINER (vert_doboz), horiz_doboz);

  hoszt_textfield = gtk_entry_new ();
  gtk_entry_set_max_length (GTK_ENTRY (hoszt_textfield), 64);
  gtk_entry_set_width_chars (GTK_ENTRY (hoszt_textfield), 20);
  gtk_entry_set_text (GTK_ENTRY (hoszt_textfield), "localhost");
  gtk_box_pack_start (GTK_BOX (horiz_doboz), hoszt_textfield,
TRUE, TRUE, 0);

  port_textfield = gtk_entry_new ();
  gtk_entry_set_max_length (GTK_ENTRY (port_textfield), 5);
  gtk_entry_set_width_chars (GTK_ENTRY (port_textfield), 6);
  gtk_entry_set_text (GTK_ENTRY (port_textfield), "2005");
  gtk_box_pack_start (GTK_BOX (horiz_doboz), port_textfield, TRUE,
TRUE, 0);

  elkuld_gomb = gtk_button_new_with_label ("OK");
  gtk_container_set_border_width (GTK_CONTAINER (elkuld_gomb), 8);
  gtk_box_pack_start (GTK_BOX (horiz_doboz), elkuld_gomb, TRUE,
TRUE, 0);

  g_signal_connect (G_OBJECT (elkuld_gomb), "clicked",
              G_CALLBACK (elkuld_gomb_benyomva), NULL);

  gorgetosav = gtk_scrolled_window_new (NULL, NULL);
  gtk_scrolled_window_set_shadow_type (GTK_SCROLLED_WINDOW
(gorgetosav),
                               GTK_SHADOW_IN);
  gtk_box_pack_start (GTK_BOX (vert_doboz), gorgetosav, TRUE,
TRUE, 0);

  valasz_szovegdoboz = gtk_text_view_new ();
  gtk_text_view_set_editable (GTK_TEXT_VIEW (valasz_szovegdoboz),
FALSE);
  gtk_container_add (GTK_CONTAINER (gorgetosav),
valasz_szovegdoboz);

  gtk_widget_show_all (prog_pater_ablak);
```

```
  gtk_main ();

  return 0;
}
```

A fordításhoz a következő make fájlt használva:

```
CC = gcc
CFLAGS = -Wall `pkg-config --cflags gtk+-2.0`
LDFLAGS = `pkg-config --libs gtk+-2.0`

prog_pater_kliens : prog_pater_kliens.c
     $(CC) $(CFLAGS) $(LDFLAGS) $^ -o $@

clean:
     rm *~ prog_pater_kliens
```

ezt kapjuk:



Ábra 31: ProgPáter ablak, harmadik lépés

Most pedig építsük bele a hálózati funkcionalitást! A *132*. oldal *C socket kliens* programját így módosítjuk, azaz részlet a most megírandó `tcp_kliens.h` headerből:

```
char *
tcp_kliens (char *hoszt, char *port, char *buffer, int
buffer_meret);
```

Az ehhez tartozó, módosított C forrás:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
char *
tcp_kliens (char *h, char *p, char *buffer, int buffer_meret)
{
  int kapu, olvasva;
  struct sockaddr_in szerver;
  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton (h, &(szerver.sin_addr));
  szerver.sin_port = htons (atoi(p));
```

```
  if ((kapu = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
    {
      perror ("socket");
      strncpy(buffer, "Hiba a kapcsolodasnal!\n", buffer_meret);
      return buffer;
    }
  if (connect (kapu, (struct sockaddr *) &szerver, sizeof
(szerver)) == -1)
    {
      perror ("connect");
      strncpy(buffer, "Nem megy a szerver!\n", buffer_meret);
      return buffer;
    }
  olvasva = read (kapu, buffer, buffer_meret);
  buffer[olvasva]='\0';
  return buffer;
}
```

A felülettel kapcsolatos kód, a hálózati funkcionalitást egy pthread szálban valósítjuk meg:

```
#include <gtk/gtk.h>
#include <string.h>
#include <pthread.h>
#include "tcp_kliens.h"
#define BUFFER_MERET 256
GtkWidget *hoszt_textfield;
GtkWidget *port_textfield;
GtkWidget *valasz_szovegdoboz;
char *hoszt;
char *port;
void *
kliens ()
{
  char buffer[BUFFER_MERET];
  char *valasz = tcp_kliens (hoszt, port, buffer, BUFFER_MERET);
  gtk_text_buffer_insert_at_cursor (gtk_text_view_get_buffer
                        (GTK_TEXT_VIEW (valasz_szovegdoboz)),
                       valasz, strlen (valasz));
  while(gtk_events_pending())
    gtk_main_iteration();
  pthread_exit (NULL);
}
gint
prog_pater_ablak_becsuk (GtkWidget * widget, GdkEvent * event,
gpointer data)
{
  gtk_main_quit ();
  return FALSE;
}
void
elkuld_gomb_benyomva (GtkWidget * widget, gpointer data)
{
  pthread_t nemkell_id;
  const gchar *h = gtk_entry_get_text (GTK_ENTRY
(hoszt_textfield));
  const gchar *p = gtk_entry_get_text (GTK_ENTRY
(port_textfield));
```

```
  g_print ("%s:%s\n", h, p);
  hoszt = (char *) h;
  port = (char *) p;
  if (pthread_create (&nemkell_id, NULL, kliens, NULL))
    {
      char *hiba = "pthread_create hiba";
      perror (hiba);
      gtk_text_buffer_insert_at_cursor (gtk_text_view_get_buffer
                          (GTK_TEXT_VIEW (valasz_szovegdoboz)),
                          hiba, strlen (hiba));
    }
}
int
main (int argc, char **argv)
{
  GtkWidget *prog_pater_ablak;
  GtkWidget *vert_doboz;
  GtkWidget *horiz_doboz;
  GtkWidget *elkuld_gomb;
  GtkWidget *gorgetosav;

  gtk_init (&argc, &argv);

  prog_pater_ablak = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title (GTK_WINDOW (prog_pater_ablak),
                "ProgPater TCP Kliens");
  gtk_window_set_resizable (GTK_WINDOW (prog_pater_ablak), FALSE);
  gtk_window_set_position (GTK_WINDOW (prog_pater_ablak),
GTK_WIN_POS_CENTER);
  gtk_container_set_border_width (GTK_CONTAINER
(prog_pater_ablak), 4);

  g_signal_connect (G_OBJECT (prog_pater_ablak), "delete_event",
              G_CALLBACK (prog_pater_ablak_becsuk), NULL);

  vert_doboz = gtk_vbox_new (FALSE, 0);
  gtk_container_add (GTK_CONTAINER (prog_pater_ablak),
vert_doboz);

  horiz_doboz = gtk_hbox_new (FALSE, 0);
  gtk_container_add (GTK_CONTAINER (vert_doboz), horiz_doboz);

  hoszt_textfield = gtk_entry_new ();
  gtk_entry_set_max_length (GTK_ENTRY (hoszt_textfield), 64);
  gtk_entry_set_width_chars (GTK_ENTRY (hoszt_textfield), 20);
  gtk_entry_set_text (GTK_ENTRY (hoszt_textfield), "localhost");
  gtk_box_pack_start (GTK_BOX (horiz_doboz), hoszt_textfield,
TRUE, TRUE, 0);

  port_textfield = gtk_entry_new ();
  gtk_entry_set_max_length (GTK_ENTRY (port_textfield), 5);
  gtk_entry_set_width_chars (GTK_ENTRY (port_textfield), 6);
  gtk_entry_set_text (GTK_ENTRY (port_textfield), "2005");
  gtk_box_pack_start (GTK_BOX (horiz_doboz), port_textfield, TRUE,
TRUE, 0);

  elkuld_gomb = gtk_button_new_with_label ("OK");
  gtk_container_set_border_width (GTK_CONTAINER (elkuld_gomb), 8);
```

```
  gtk_box_pack_start (GTK_BOX (horiz_doboz), elkuld_gomb, TRUE,
TRUE, 0);

  g_signal_connect (G_OBJECT (elkuld_gomb), "clicked",
              G_CALLBACK (elkuld_gomb_benyomva), NULL);

  gorgetosav = gtk_scrolled_window_new (NULL, NULL);
  gtk_scrolled_window_set_shadow_type (GTK_SCROLLED_WINDOW
(gorgetosav),
                                GTK_SHADOW_IN);
  gtk_box_pack_start (GTK_BOX (vert_doboz), gorgetosav, TRUE,
TRUE, 0);

  valasz_szovegdoboz = gtk_text_view_new ();
  gtk_text_view_set_editable (GTK_TEXT_VIEW (valasz_szovegdoboz),
FALSE);
  gtk_container_add (GTK_CONTAINER (gorgetosav),
valasz_szovegdoboz);

  gtk_widget_show_all (prog_pater_ablak);

  gtk_main ();

  return 0;
}
```

A fordításhoz a következő make fájlt használva, feltéve, hogy a külön könyvtárban dolgozunk a `prog_pater_kliens.c` és a `tcp_kliens.h`fájlokkal:

```
CC = gcc
CFLAGS = -Wall `pkg-config --cflags gtk+-2.0`
LDFLAGS = -lpthread `pkg-config --libs gtk+-2.0`

objects := $(patsubst %c, %o, $(wildcard *.c))

prog_pater_kliens : $(objects)
     $(CC) $(LDFLAGS) $^ -o $@

clean:
     rm *.o *~ prog_pater_kliens
```

Közben egy szervert is futtatva kapjuk, hogy:

Ábra 32: ProgPáter ablak, negyedik lépés

VIII.2.2 GNOME

VIII.2.3 Java-GNOME

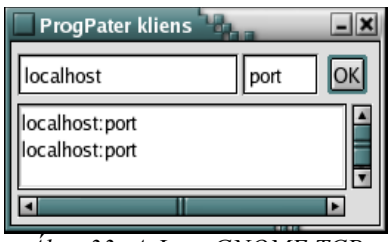
```
public class ProgPaterKliens {
    public static void main (String args[]) {

     org.gnu.gnome.Program.initGnomeUI ("ProgPater TCP kliens",
                               "0.0.1", args);

     org.gnu.gtk.Window progPaterAblak
       = new org.gnu.gtk.Window(org.gnu.gtk.WindowType.TOPLEVEL);

     progPaterAblak.setBorderWidth(4);
     progPaterAblak.setResizable(false);
     progPaterAblak.setTitle("ProgPater kliens");
     progPaterAblak.setPosition(org.gnu.gtk.WindowPosition.CENTER_A
LWAYS);
     progPaterAblak.addListener (
        new org.gnu.gtk.event.LifeCycleListener () {
          public boolean lifeCycleQuery (org.gnu.gtk
                                .event.LifeCycleEvent esemeny) {

              if (esemeny.isOfType (org.gnu.gtk
                            .event.LifeCycleEvent.Type.DELETE)) {
                org.gnu.gtk.Gtk.mainQuit ();
              }

              return false;
          }
          public void lifeCycleEvent (org.gnu.gtk
                               .event.LifeCycleEvent esemeny) {
          }
         });

     org.gnu.gtk.VBox vertDoboz = new org.gnu.gtk.VBox(false, 2);
     progPaterAblak.add(vertDoboz);

     org.gnu.gtk.HBox horizDoboz = new org.gnu.gtk.HBox(false, 2);
     vertDoboz.add(horizDoboz);

     final org.gnu.gtk.Entry hosztTextfield
       = new org.gnu.gtk.Entry();
     hosztTextfield.setMaxLength(64);
     hosztTextfield.setWidth(20);
     hosztTextfield.setText("localhost");
     horizDoboz.packStart(hosztTextfield, true, true, 0);

     final org.gnu.gtk.Entry portTextfield =
       new org.gnu.gtk.Entry();
     portTextfield.setMaxLength(5);
     portTextfield.setWidth(6);
     portTextfield.setText("port");
     horizDoboz.packStart(portTextfield, true, true, 0);

     org.gnu.gtk.Button okGomb
```

```
      = new org.gnu.gtk.Button("OK", false);
    okGomb.setBorderWidth(4);
    horizDoboz.packStart(okGomb, true, true, 0);

    org.gnu.gtk.ScrolledWindow gorgetosav
      = new org.gnu.gtk.ScrolledWindow();
    gorgetosav.setShadowType(org.gnu.gtk.ShadowType.IN);
    vertDoboz.packStart(gorgetosav, true, true, 0);

    final org.gnu.gtk.TextView valaszSzovegdoboz
      = new org.gnu.gtk.TextView();
    valaszSzovegdoboz.setEditable(false);
    gorgetosav.add(valaszSzovegdoboz);

    okGomb.addListener (new org.gnu.gtk.event.ButtonListener () {
         public void buttonEvent(org.gnu.gtk.event.ButtonEvent
esemeny) {
              if (esemeny.isOfType (org.gnu.gtk
                            .event.ButtonEvent.Type.CLICK)) {
               String hosztPort = hosztTextfield.getText()
                   +":"+portTextfield.getText();
               System.out.println(hosztPort);
               org.gnu.gtk.TextBuffer buffer
                   = valaszSzovegdoboz.getBuffer();
               buffer.insertText(hosztPort+"\n");
               valaszSzovegdoboz.setBuffer(buffer);
              }
          }
        });

    progPaterAblak.showAll ();
    org.gnu.gtk.Gtk.main ();
   }
}
```

Így fordítsuk és futtassuk:

```
$ javac -classpath
/usr/share/java/gtk2.6.jar:/usr/share/java/gnome2.10.jar
ProgPaterKliens.java
$ java -Djava.library.path=/usr/lib64 -classpath
.:/usr/share/java/gtk2.6.jar:/usr/share/java/gnome2.10.jar
ProgPaterKliens
```

Az eredmény:



Ábra 33: A Java-GNOME TCP kliens felülete.

A *268*. oldalon példát láthatunk, hogyan készítünk ebből a forrásból futtathatót a `gcj` fordítóval.

IX. Adatbázis programozás

Bevezető feladatként fejlesszük most tovább a *169*. oldal *Java szervletek* című pontjában megismert olvasói visszajelzéseket kezelő szervletet!

IX.1 MySQL

Létrehozzuk a majd az alábbi egyetlen táblát tartalmazó MySQL adatbázisunkat. Tervünk, hogy a könyvel kapcsolatos esetleges visszajelzéseket tároljuk benne a következő tábla formájában:

sorszám	email	visszajelzés ideje	szövege

```
# mysql
.
.
.
mysql> CREATE DATABASE prog_pater;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL ON prog_pater.* TO
'norbi'@'localhost.localdomain'
          IDENTIFIED BY 'jelszo';
Query OK, 0 rows affected (0.00 sec)
mysql> quit
Bye
```

Majd felhasználóként – kis ismerkedés – után létrehozzuk a táblát:

```
$ mysql -h localhost.localdomain -u norbi -pjelszo prog_pater
.
.
.
mysql> show databases;
+------------+
| Database   |
+------------+
| prog_pater |
| test       |
+------------+
2 rows in set (0.00 sec)
mysql> source visszajelzes.sql;
Database changed
ERROR 1051 (42S02): Unknown table 'visszajelzes'
Query OK, 0 rows affected (0.01 sec)
Query OK, 2 rows affected (0.00 sec)
Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
Query OK, 1 row affected (0.00 sec)
+---------+----------------------+- ...
| sorszam | email                | mikor ...
+---------------------------------+ ...
|       1 | nbatfai@inf.unideb.hu | 2005-12-07 20:32:04 ...
|       2 | javacska@javacska.hu  | 2005-12-07 20:32:04 | ...
...
3 rows in set (0.00 sec)
```

```
mysql> quit
Bye
```

Láthatóan, a kívánt iniciális tesztadatokkal létrejött a tábla. A `visszajelzes.sql` fájt hajtottuk végre, ebben fogalmaztuk meg a táblánkat:

```
USE prog_pater;
DROP TABLE visszajelzes;
CREATE TABLE visszajelzes (
     sorszam    INT UNSIGNED NOT NULL AUTO_INCREMENT
                      PRIMARY KEY,
     email            VARCHAR(40),
     mikor            TIMESTAMP DEFAULT CURRENT_TIMESTAMP
                      ON UPDATE CURRENT_TIMESTAMP,
     megjegyzes       TEXT
);
LOAD DATA LOCAL INFILE './visszajelzesek.txt' INTO TABLE
visszajelzes;
INSERT INTO visszajelzes VALUES
(\N, 'nbatfai@inf.unideb.hu', \N, 'Valóban? Ennek örülök :)');
SELECT * FROM visszajelzes;
```

A harmadik sort „kézzel szúrtuk” be, az első kettőt a `visszajelzesek.txt` fájlból:

```
\N nbatfai@inf.unideb.hu \N Örömmel vesszük a visszejelzéseket!
\N javacska@javacska.hu  \N Nagyon tetszik :)
```

(az oszlopokat tabulátorok határolják).

A szervlet előtt írjunk egy olyan JDBC-s programot, ami kiírja a tábla tartalmát:

```
public class Visszajelzesek {
  public static final void main(String args[]) {
    java.sql.Statement sqlUtasitas = null;
    java.sql.ResultSet eredmeny = null;
    try {
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      java.sql.Connection conn
        = java.sql.DriverManager
          .getConnection("jdbc:mysql://localhost/prog_pater?user=no
rbi&password=jelszo");
          sqlUtasitas = conn.createStatement();
          eredmeny = sqlUtasitas
            .executeQuery("SELECT * FROM visszajelzes");
          while (eredmeny.next())
            System.out.printf("%s|%-30s|%-20s\n%s\n",
              eredmeny.getString(1), eredmeny.getString(2),
              eredmeny.getString(3), eredmeny.getString(4));
          eredmeny.close();
          eredmeny = null;
          sqlUtasitas.close();
          sqlUtasitas = null;
      } catch (java.sql.SQLException se) {
          se.printStackTrace();
```

```
      } catch (Exception e) {
          e.printStackTrace();
          System.err.println("Benne van a classpathban a MySQl JDBC
meghajto jar?");
      } finally {
        if(eredmeny != null) {
          try {
            eredmeny.close();
          } catch(Exception e) {}
          eredmeny = null;
        }
        if(sqlUtasitas != null) {
          try {
            sqlUtasitas.close();
          } catch(Exception e) {}
          sqlUtasitas = null;
        }
      }
  }
}
```

Ha a fordítás után, a futtatáskor a classpath-ban nincs benne a `com.mysql.jdbc.Driver` osztályt is tartalmazó JDBC meghajtóprogram, akkor ezt kapjuk:

```
$ java Visszajelzesek
java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
         .
         .
         .
        at Visszajelzesek.main(Visszajelzesek.java:6)
Benne van a classpathban a MySQl JDBC meghajto jar?
```

Egyébként, az alábbi beállítás

```
$ export CLASSPATH=$CLASSPATH:/usr/share/tomcat5/common/lib/mysql-
connector-java-3.1.12-bin.jar
```

után

```
$ java Visszajelzesek
1|nbatfai@inf.unideb.hu         |2005-12-11 20:35:06.0
Örömmel vesszük a visszejelzéseket!
2|javacska@javacska.hu          |2005-12-11 20:35:06.0
Nagyon tetszik :)
3|nbatfai@inf.unideb.hu         |2005-12-11 20:35:06.0
Valóban? Ennek örülök :)
```

IX.1.1 Feladat – sor beszúrása

Az előző példa mintájára készítsük el azt a programot, ami a parancssorából átvett emaillel és visszajelzéssel bővíti a táblát!

```
public class Visszajelzes {
  public static final void main(String args[]) {
    if(args.length != 2) {
      System.err.println("java Visszajelzes email visszajelzes");
```

```
      System.exit(-1);
    }
    java.sql.PreparedStatement sqlUtasitas = null;
    try {
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      java.sql.Connection conn
        = java.sql.DriverManager
          .getConnection("jdbc:mysql://localhost/prog_pater?user=no
rbi&password=jelszo");
       sqlUtasitas = conn
              .prepareStatement("INSERT INTO  visszajelzes (email,
megjegyzes) VALUES(?, ?)");
       sqlUtasitas.setString(1, args[0]);
       sqlUtasitas.setString(2, args[1]);
       int m = sqlUtasitas.executeUpdate();
       sqlUtasitas.close();
       sqlUtasitas = null;
      } catch (java.sql.SQLException se) {
          se.printStackTrace();
      } catch (Exception e) {
          e.printStackTrace();
          System.err.println("Benne van a classpathban a MySQl JDBC
meghajto jar?");
      } finally {
        if(sqlUtasitas != null) {
          try {
            sqlUtasitas.close();
          } catch(Exception e) {}
          sqlUtasitas = null;
        }
      }
  }
}
```

IX.1.2 Az adatbázis elérése Java szervletből

Csináljuk ugyanezt – azaz a listázást és új sor bevitelét – de most szervletből! Fejlesszük tovább a *169*. oldal *Java szervletek* című pontjában megismert olvasói visszajelzéseket kezelő szervletet!

Azaz böngészőbe listázunk:

```
public class VisszajelzesekSzervlet
    extends javax.servlet.http.HttpServlet {
    javax.sql.DataSource progPaterAdatforras;
    public void init () throws javax.servlet.ServletException {
     try {
         javax.naming.Context initialContext
          = new javax.naming.InitialContext ();
         progPaterAdatforras =
          (javax.sql.DataSource) initialContext
          .lookup ("java:/comp/env/jdbc/progPater");

     } catch (Exception e) {
         log("Az adatforras nem elerheto!");
         throw new javax.servlet
          .UnavailableException ("Az adatforras nem elerheto!");
```

```
    }
    log ("Az adatforras beallitva.");
   }
   public void doGet (javax.servlet.http.HttpServletRequest keres,
               javax.servlet.http.HttpServletResponse valasz)
    throws java.io.IOException, javax.servlet.ServletException {
    log ("VisszajelzesSzervlet doGet() elindult\n");
    valasz.setContentType ("text/html");
    java.io.PrintWriter szovegesCsatorna = valasz.getWriter ();
    szovegesCsatorna.println ("<html>");
    szovegesCsatorna.println ("<body>");
    szovegesCsatorna
        .println ("<table border=\"1\" cellpadding=\"3\">");
    java.sql.Connection dbKapcsolat = null;
    java.sql.Statement sqlSelect = null;
    java.sql.ResultSet eredmeny = null;
    int oszlopokSzama = 0;
    try {
        dbKapcsolat = progPaterAdatforras.getConnection ();
        sqlSelect = dbKapcsolat.createStatement ();
        eredmeny = sqlSelect
          .executeQuery ("SELECT * FROM visszajelzes");
        java.sql.ResultSetMetaData eredmenyAdat
           = eredmeny.getMetaData();
        oszlopokSzama = eredmenyAdat.getColumnCount();
        szovegesCsatorna.println ("<tr>");
        for(int i=1; i<=oszlopokSzama; ++i) {
         szovegesCsatorna.println ("<td><b>");
         szovegesCsatorna
           .println (eredmenyAdat.getColumnLabel(i));
         szovegesCsatorna.println ("</td></b>");
        }
        szovegesCsatorna.println ("</tr>");
        while (eredmeny.next ()) {
         szovegesCsatorna.println ("<tr>");
         for(int i=1; i<=oszlopokSzama; ++i) {
             szovegesCsatorna.println ("<td>");
             szovegesCsatorna.println (eredmeny.getString (i));
             szovegesCsatorna.println ("</td>");
         }
         szovegesCsatorna.println ("</tr>");
        }
        eredmeny.close ();
        eredmeny = null;
        sqlSelect.close ();
        sqlSelect = null;
        dbKapcsolat.close();
        dbKapcsolat = null;
    } catch (java.sql.SQLException e) {
        e.printStackTrace ();
        szovegesCsatorna.println (e.getMessage());
    } finally {
        if(eredmeny != null) {
         try {eredmeny.close();} catch(Exception e) {}
         eredmeny = null;
        }
        if(sqlSelect != null) {
         try {sqlSelect.close();} catch(Exception e) {}
```

```
            sqlSelect = null;
          }
          if(dbKapcsolat != null) {

            try {dbKapcsolat.close();} catch(Exception e) {}
            dbKapcsolat = null;
          }
      }
      szovegesCsatorna.println ("</table>");
      szovegesCsatorna.println ("</body>");
      szovegesCsatorna.println ("</html>");
      szovegesCsatorna.close ();
     }
}
```

Ezt a szervletet a megfelelő – a *169*. oldali – *web.xml* alapján a `http://niobe.eurosmobil.hu:8080/prog-pater/lista` címen érjük el a böngészőből.

Böngészőből, HTML formról viszünk fel visszajelzést:

```
public class VisszajelzesSzervlet
    extends javax.servlet.http.HttpServlet {
    javax.sql.DataSource progPaterAdatforras;
    public void init () throws javax.servlet.ServletException {
     try {
         javax.naming.Context initialContext
           = new javax.naming.InitialContext ();
         progPaterAdatforras =
           (javax.sql.DataSource) initialContext
           .lookup ("java:/comp/env/jdbc/progPater");
     } catch (Exception e) {
         log("Az adatforras nem elerheto!");
         throw new javax.servlet
           .UnavailableException ("Az adatforras nem elerheto!");
     }
     log ("Az adatforras beallitva.");
    }
    public void doPost (
                 javax.servlet.http.HttpServletRequest keres,
                 javax.servlet.http.HttpServletResponse valasz)
     throws java.io.IOException, javax.servlet.ServletException {
     log ("VisszajelzesSzervlet doPost() elindult\n");
     valasz.setContentType ("text/html");
     java.io.PrintWriter szovegesCsatorna = valasz.getWriter ();
     String email = keres.getParameter("email");
     if(email == null || email.length()==0)
         email = "nincs megadva";
     if(email.length()>40)
         email = email.substring(0, 39);
     String megjegyzes = keres.getParameter("megjegyzes");
     if(megjegyzes == null || megjegyzes.length()==0)
         megjegyzes = "nincs megadva";
     if(megjegyzes.length()>1000)
         megjegyzes = megjegyzes.substring(0, 999);
     szovegesCsatorna.println ("<html>");
     szovegesCsatorna.println ("<body>");
```

```
     szovegesCsatorna
         .println ("<table border=\"1\" cellpadding=\"3\">");
     java.sql.Connection dbKapcsolat = null;
     java.sql.PreparedStatement sqlUpdate = null;
     java.sql.Statement sqlSelect = null;
     java.sql.ResultSet eredmeny = null;
     int oszlopokSzama = 0;
     try {
         dbKapcsolat = progPaterAdatforras.getConnection ();
         sqlUpdate = dbKapcsolat
          .prepareStatement("INSERT INTO  visszajelzes (email,
megjegyzes) VALUES(?, ?)");
         sqlUpdate.setString(1, email);
         sqlUpdate.setString(2, megjegyzes);
         int m = sqlUpdate.executeUpdate();
         sqlUpdate.close();
         sqlUpdate = null;
         sqlSelect = dbKapcsolat.createStatement ();
         eredmeny = sqlSelect
           .executeQuery ("SELECT * FROM visszajelzes");
         java.sql.ResultSetMetaData eredmenyAdat
           = eredmeny.getMetaData();
         oszlopokSzama = eredmenyAdat.getColumnCount();
         szovegesCsatorna.println ("<tr>");
         for(int i=1; i<=oszlopokSzama; ++i) {
          szovegesCsatorna.println ("<td><b>");
          szovegesCsatorna
            .println (eredmenyAdat.getColumnLabel(i));
          szovegesCsatorna.println ("</td></b>");
         }
         szovegesCsatorna.println ("</tr>");

         while (eredmeny.next ()) {
          szovegesCsatorna.println ("<tr>");
          for(int i=1; i<=oszlopokSzama; ++i) {
              szovegesCsatorna.println ("<td>");
              szovegesCsatorna.println (eredmeny.getString (i));
              szovegesCsatorna.println ("</td>");
          }
          szovegesCsatorna.println ("</tr>");
         }
         eredmeny.close ();
         eredmeny = null;
         sqlSelect.close ();
         sqlSelect = null;
         dbKapcsolat.close();
         dbKapcsolat = null;
     } catch (java.sql.SQLException e) {
         e.printStackTrace ();
         szovegesCsatorna.println (e.getMessage());
     } finally {
         if(sqlUpdate != null) {
          try {sqlUpdate.close();} catch(Exception e) {}
          sqlUpdate = null;
         }
         if(eredmeny != null) {
          try {eredmeny.close();} catch(Exception e) {}
          eredmeny = null;
```

```
        }
        if(sqlSelect != null) {
          try {sqlSelect.close();} catch(Exception e) {}
          sqlSelect = null;
        }
        if(dbKapcsolat != null) {
          try {dbKapcsolat.close();} catch(Exception e) {}
          dbKapcsolat = null;
        }
     }
     szovegesCsatorna.println ("</table>");
     szovegesCsatorna.println ("</body>");
     szovegesCsatorna.println ("</html>");
     szovegesCsatorna.close ();
    }
}
```

A szervletet az alábbi egyszerű HTML formról (`http://niobe.eurosmobil.hu/uj.html`) hívhatjuk például:

```
<html>
        <form action="http://niobe.eurosmobil.hu:8080/prog-
pater/uj" method="post">
                Email:
                <br>
                <input type="text" size="40" name="email">
                <br>
                Megjegyzés:
                <br>
                <textarea name="megjegyzes" cols="40"
                   rows="5"></textarea>
                <br>
                <input type="submit" value="Elküld">
        </form>
</html>
```

Ne felejtsük el az új szervletet is felvenni a *169*. oldalon elkezdett *web.xml* fájlba, nem is beszélve az adatforrás megadásáról:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
    <resource-ref>
      <description>
        A ProgPater-el kapcsolatos visszajelzesek adatforrasa.
      </description>
      <res-ref-name>
        jdbc/progPater
      </res-ref-name>
      <res-type>
        javax.sql.DataSource
      </res-type>
      <res-auth>
        Container
      </res-auth>
```

```
    </resource-ref>
    <display-name>ProgPater pelda</display-name>
    <description>
        ProgPater pelda: visszajelzesek listazasa, felvitele.
    </description>
    <servlet>
      <servlet-name>visszajelzesek</servlet-name>
      <description>
        Visszajelzesek listazasa
      </description>
      <servlet-class>VisszajelzesekSzervlet</servlet-class>
    </servlet>
    <servlet>
      <servlet-name>visszajelzes</servlet-name>
      <description>
        Uj visszajelzes felvitele
      </description>
      <servlet-class>VisszajelzesSzervlet</servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>visszajelzesek</servlet-name>
      <url-pattern>/lista</url-pattern>
    </servlet-mapping>
   <servlet-mapping>
      <servlet-name>visszajelzes</servlet-name>
      <url-pattern>/uj</url-pattern>
    </servlet-mapping>
</web-app>
```

Erre az új szervletre – ahogy a korábbi `uj.html` fájlban láttuk – POST módszerrel, a `http://niobe.eurosmobil.hu:8080/prog-pater/uj` címmel hivatkozhatunk.

Még az adatforrást kell elkészítenünk, esetünkben, azaz a *169*. oldalon, a *Java szervletek* pontban elkezdett példa esetén a `/usr/share/tomcat5/conf/server.xml` fájlt egészítjük ki az alábbi webalkalmazás leírással:

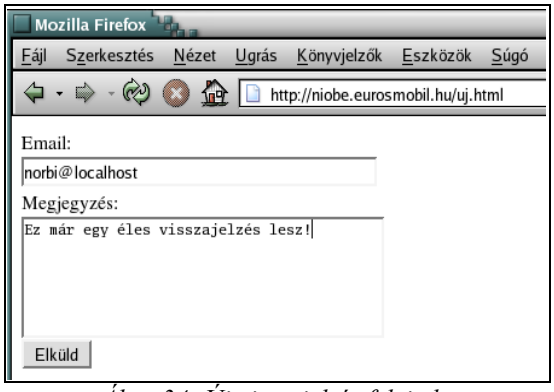
```
<Context path="/prog-pater" reloadable="true">
  <Resource name="jdbc/progPater" auth="Container"
            type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/progPater">
    <parameter>
      <name>username</name>
      <value>norbi</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>jelszo</value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>com.mysql.jdbc.Driver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:mysql://localhost:3306/prog_pater</value>
    </parameter>
```

```
    <parameter>
      <name>maxActive</name>
      <value>8</value>
    </parameter>
    <parameter>
      <name>maxIdle</name>
      <value>4</value>
    </parameter>
  </ResourceParams>
</Context>
```

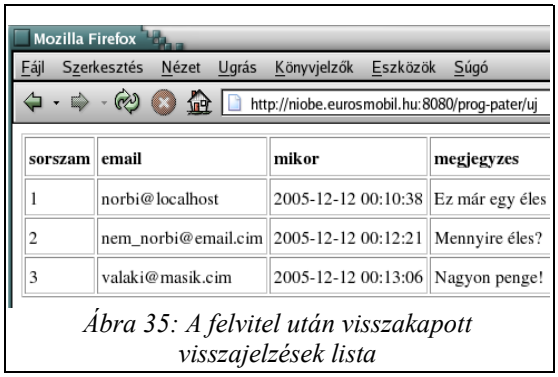
Nincs más hátra, mint a szerver újraindítása:

```
# /etc/rc.d/init.d/tomcat5 restart
```

Hogy végül lássuk munkánk gyümölcsét:



Ábra 34: Új visszajelzés felvitele



Ábra 35: A felvitel után visszakapott visszajelzések lista

IX.1.3 Feladat – 2 in 1

Dolgozzuk össze az iménti két szervletet egybe!

IX.2 PostgreSQL

Most ugyanazt játsszuk végig, amit az iménti fejezetben csináltunk, de a PostgreSQL adatbáziskezelőt használjuk! Figyeljük meg, hogy az előző fejezetben megírt Java

szervletekkel semmi dolgunk nem lesz, csupán a `server.xml` fájlban adunk meg egy új adatforrást:

```
<Context path="/prog-pater" reloadable="true">
  <Resource name="jdbc/progPater" auth="Container"
            type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/progPater">
    <parameter>
      <name>username</name>
      <value>norbi</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>jelszo</value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>org.postgresql.Driver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:postgresql://localhost/prog_pater</value>
    </parameter>
    <parameter>
      <name>maxActive</name>
      <value>8</value>
    </parameter>
    <parameter>
      <name>maxIdle</name>
      <value>4</value>
    </parameter>
  </ResourceParams>
</Context>
```

De természetesen az adatforrás alá fel kell építenünk a megfelelő PostgreSQL adatbázist:

```
# su - postgres
```

```
$ createuser norbi -P
Enter password for new user:
Enter it again:
Shall the new user be allowed to create databases? (y/n) n
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER

$ createdb prog_pater --encoding LATIN2 -O norbi
CREATE DATABASE
```

Majd – ugyancsak postgres felhasználóként – szerkesszük meg a `$PGDATA/pg_hba.conf` fájt, az elejére szúrjuk be:

```
local   prog_pater      norbi                   md5
host    prog_pater      norbi   127.0.0.1/32    md5
```

Majd indítsuk újra az adatbáziskezelőt:

```
# /etc/rc.d/init.d/postgresql restart
```

Végül felhasználóként rakjuk össze az előző fejezetben megtervezett táblánkat:

```
$ psql prog_pater -U norbi
.
.
.
prog_pater=> \i visszajelzes.sql
DROP TABLE
.
.
.
CREATE TABLE
INSERT 17456 1
INSERT 17457 1
INSERT 17458 1
 sorszam |         email          |           mikor
---------+------------------------+---------------------- ...
       1 | nbatfai@inf.unideb.hu | 2005-12-14 15:53:14.6 ...
       2 | javacska@javacska.hu  | 2005-12-14 15:53:14.63 ...
       3 | nbatfai@inf.unideb.hu | 2005-12-14 15:53:14.633 ...
(3 sor)
prog_pater=> \q
```

A `visszajelzes.sql` fájt hajtottuk végre – az előző fejezethez hasonlóan – ebben fogalmaztuk meg a táblánkat:

```
DROP TABLE visszajelzes;
CREATE TABLE visszajelzes (
        sorszam         SERIAL PRIMARY KEY,
        email           VARCHAR(40),
        mikor           TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        megjegyzes      TEXT
);
INSERT INTO visszajelzes (email, megjegyzes)
        VALUES ('nbatfai@inf.unideb.hu', 'Örömmel vesszük a
visszejelzéseket!')
INSERT INTO visszajelzes (email, megjegyzes)
        VALUES ('javacska@javacska.hu', 'Nagyon tetszik :)');
INSERT INTO visszajelzes (email, megjegyzes)
        VALUES ('nbatfai@inf.unideb.hu', 'Valóban? Ennek örülök
:)');
SELECT * FROM visszajelzes;
```

Ezzel készen is vagyunk, ne felejtsük el a a `server.xml` fájl módosítása miatt a Tomcat-et újraindítani, aztán máris jöhet a tesztelés.

X. Mobil programozás

X.1 Symbian OS

Különleges követelmények.

X.1.1 Kódolási konvenciók

X.1.1.1 C++

X.1.1.1.1 Symbian C++ „Helló Erika!” a Series 60 Developer Platform 2.0 telefonokra

Ezt a példát a gyakorlaton nem tudjuk kipróbálni, ezért képekben is bemutatjuk, illetve a Nokia 6600 készüléken tudjuk megnézni. (Magunk a Series 60 2nd Edition SDK for Symbian OS Supporting Feature Pack 2, For C++, Supporting Microsoft Visual C++ and Borland C++BuilderX [FN]-el készítettük.)

Parancssorból dolgozva:

```
bldmake bldfiles
```

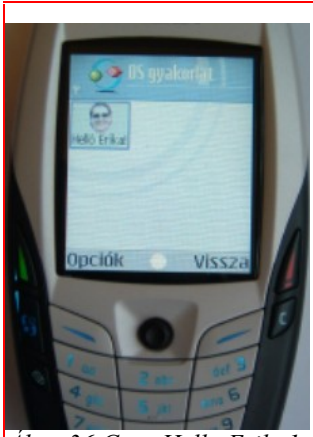
összeállítottuk a felépítéshez szükséges abld-t

```
abld build armi urel
```

a telefonra készítettük

```
makesis HelloErika.pkg
```

és elkészítettük a `sis` fájt, amit Bluetooth-on keresztüli sikeres feltöltés után most az alábbi képekben és egy valódi Nokia 6600 telefonon tudunk bemutatni:



Ábra 36 C++ Hello Erika1, Nokia 6600

Ábra 37 C++ Hello Erika2, Nokia 6600

Ábra 38 C++ Hello Erika3, Nokia 6600

X.2 Java ME

A Java platform. Konfiguráció, profil. JTWI.

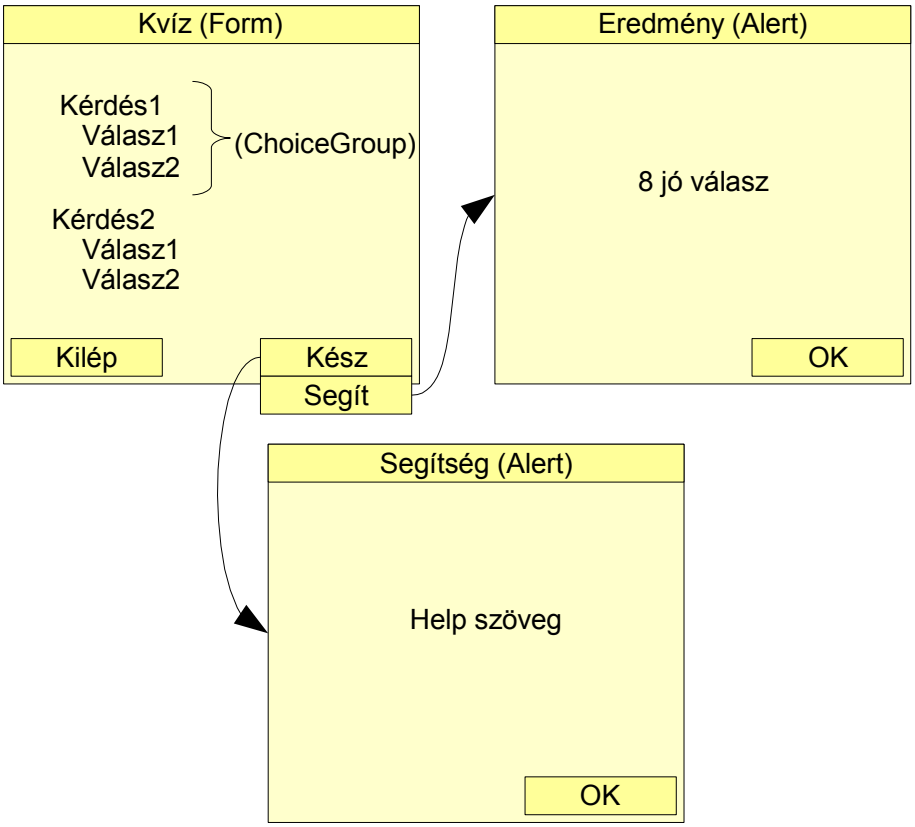
A fejlesztéshez a NetBeans Mobility Pack-ot használjuk [*Mobil*/NB MOBILITY].

X.2.1 A Java ME alkalmazások életciklusa

```
public class MIDletEletciklus
        extends javax.microedition.midlet.MIDlet
        implements javax.microedition.lcdui.CommandListener{

    private javax.microedition.lcdui.Display kijelző;
    private javax.microedition.lcdui.Form elsőForm;
    private javax.microedition.lcdui.Command kilépParancs;

    public MIDletEletciklus() {

        kijelző =
           javax.microedition.lcdui.Display.getDisplay(this);
        kilépParancs =
           new javax.microedition.lcdui.Command("Kilépek",
                javax.microedition.lcdui.Command.EXIT, 10);

    }

    public void startApp() {

        elsőForm = new javax.microedition.lcdui.Form("Első form");
        elsőForm.addCommand(kilépParancs);
        elsőForm.setCommandListener(this);

        elsőForm.append(new
                javax.microedition.lcdui.StringItem("Helló",
                                                    "Világ!"));

        kijelző.setCurrent(elsőForm);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(javax.microedition.lcdui.Command
        command,javax.microedition.lcdui.Displayable displayable) {

        if (command == kilépParancs) {
            kilép();
        }
    }
```

```
    public void kilép() {
        kijelző.setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

}
```

A Java ME API dokumentációjával (`netbeans-5.5beta2 / mobility7.3 / emulators-inst / wtk22_win.zip / emulator / wtk22 / docs / api / midp / index.html`) való további ismerkedésképpen az iménti életciklus példát fejlesszük tovább egy egyszerű kvízzé!



39. ábra: A kvíz bevezető példa szerkezete

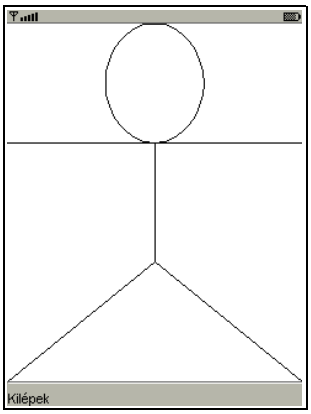
X.2.1.1 Pálcikaember a vásznon

Fejlesszük tovább `MIDletEletciklus` osztályunkat, `Form` helyett most egy vásznat készítsünk, s rajzoljunk erre a vászonra egy pálcikaembert!

```
public class VaszonMIDlet
        extends javax.microedition.midlet.MIDlet
        implements javax.microedition.lcdui.CommandListener{
```

```
    private javax.microedition.lcdui.Display kijelző;
    private javax.microedition.lcdui.Canvas elsőVászon;
    private javax.microedition.lcdui.Command kilépParancs;

    public VaszonMIDlet() {

        kijelző =
javax.microedition.lcdui.Display.getDisplay(this);
        kilépParancs = new
javax.microedition.lcdui.Command("Kilépek",
                javax.microedition.lcdui.Command.EXIT, 10);
    }

    public void startApp() {

        elsőVászon = new javax.microedition.lcdui.Canvas(){

            public void paint(javax.microedition.lcdui.Graphics g)
{

                g.setColor(0x00ffffff);
                g.fillRect(0, 0, getWidth(), getHeight());
                g.setColor(0x00000000);
                g.drawArc(getWidth()/3, 0, getWidth()/3,
getHeight()/3, 0, 360);
                g.drawLine(0, getHeight()/3, getWidth(),
getHeight()/3);
                g.drawLine(getWidth()/2, getHeight()/3,
getWidth()/2, 2*getHeight()/3);
                g.drawLine(0, getHeight(), getWidth()/2,
2*getHeight()/3);
                g.drawLine(getWidth(), getHeight(), getWidth()/2,
2*getHeight()/3);

            }

        };
        elsőVászon.addCommand(kilépParancs);
        elsőVászon.setCommandListener(this);

        kijelző.setCurrent(elsőVászon);
    }

    public void commandAction(javax.microedition.lcdui.Command
command,
            javax.microedition.lcdui.Displayable displayable) {
        if (command == kilépParancs) {
            kilép();
        }
    }

    public void kilép() {
        kijelző.setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    public void pauseApp() {
```

```
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

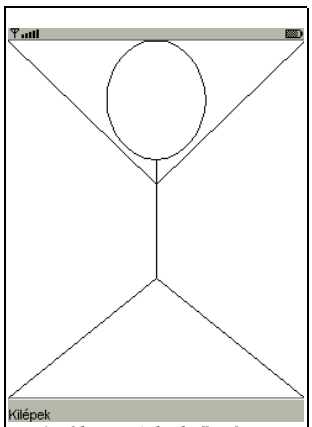


40. ábra: A VaszonMIDlet belső vászna

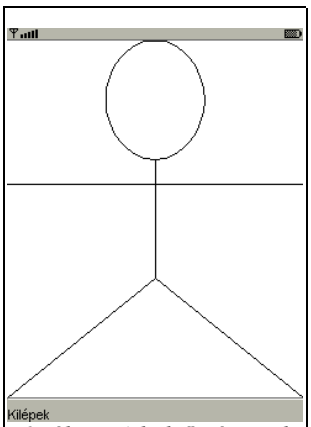
Fejlesszük tovább a `VaszonMIDlet` osztályt, először a telefon kurzor fel/le gombjának nyomására „lengesse” fel/le a karjait, majd ezt is fejlesszük tovább úgy, hogy ne csak erre a felhasználói inputra, hanem folyamatosan és periodikusan integessen (lengessen)!

```
public class VaszonMIDlet
        extends javax.microedition.midlet.MIDlet
        implements javax.microedition.lcdui.CommandListener{

    private javax.microedition.lcdui.Display kijelző;
    private javax.microedition.lcdui.Canvas elsőVászon;
    private javax.microedition.lcdui.Command kilépParancs;

    public VaszonMIDlet() {

        kijelző =
javax.microedition.lcdui.Display.getDisplay(this);
        kilépParancs = new
javax.microedition.lcdui.Command("Kilépek",
                javax.microedition.lcdui.Command.EXIT, 10);
    }

    public void startApp() {

        elsőVászon = new javax.microedition.lcdui.Canvas(){

            boolean föl = true;

            public void keyPressed(int billenytű) {

                if(getGameAction(billenytű) ==
javax.microedition.lcdui.Canvas.UP)
                    föl = true;
                else
                    föl = false;

                repaint();
```

```
            }

            public void paint(javax.microedition.lcdui.Graphics g)
{

                g.setColor(0x00ffffff);
                g.fillRect(0, 0, getWidth(), getHeight());
                g.setColor(0x00000000);
                g.drawArc(getWidth()/3, 0, getWidth()/3,
getHeight()/3, 0, 360);
                if(föl) {
                    g.drawLine(0, 0, getWidth()/2,
getHeight()/3+20);
                    g.drawLine(getWidth(), 0, getWidth()/2,
getHeight()/3+20);
                } else
                    g.drawLine(0, getHeight()/3+20, getWidth(),
getHeight()/3+20);
                g.drawLine(getWidth()/2, getHeight()/3,
getWidth()/2, 2*getHeight()/3);
                g.drawLine(0, getHeight(), getWidth()/2,
2*getHeight()/3);
                g.drawLine(getWidth(), getHeight(), getWidth()/2,
2*getHeight()/3);

            }

        };
        elsőVászon.addCommand(kilépParancs);
        elsőVászon.setCommandListener(this);

        kijelző.setCurrent(elsőVászon);
    }

    public void commandAction(javax.microedition.lcdui.Command
command,
            javax.microedition.lcdui.Displayable displayable) {
        if (command == kilépParancs) {
            kilép();
        }
    }

    public void kilép() {
        kijelző.setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```



41. ábra: A belső vászon fel állapota



42. ábra: A belső vászon le állapota

Az „integetéshez” egy szálat készítünk. Programunkat két, a `LengetMIDlet` és a `LengetVaszon` osztályra bontjuk. A MIDlet osztályt alig változtatjuk: `elsőVászon = new LengetVaszon();` vásznunkat már nem belső osztályként valósítjuk meg:

```
public class LengetMIDlet
        extends javax.microedition.midlet.MIDlet
        implements javax.microedition.lcdui.CommandListener{

    private javax.microedition.lcdui.Display kijelző;
    private javax.microedition.lcdui.Command kilépParancs;
    private LengetVaszon elsőVászon;

    public LengetMIDlet() {

        kijelző =
javax.microedition.lcdui.Display.getDisplay(this);
        kilépParancs = new
javax.microedition.lcdui.Command("Kilépek",
                javax.microedition.lcdui.Command.EXIT, 10);

    }

    public void startApp() {

        elsőVászon = new LengetVaszon();
        elsőVászon.addCommand(kilépParancs);
        elsőVászon.setCommandListener(this);

        kijelző.setCurrent(elsőVászon);
    }

    public void commandAction(javax.microedition.lcdui.Command
command,
            javax.microedition.lcdui.Displayable displayable) {
        if (command == kilépParancs) {
            kilép();
        }
    }

    public void kilép() {
        elsőVászon.fut = false;
```

```
        kijelző.setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

A szálat a vászon osztályban implementáljuk: 100 ms késleltetéssel változtatjuk a kar vízszintes pozícióját jellemző `sorKoord` változót.

```
public class LengetVaszon extends javax.microedition.lcdui.Canvas
        implements Runnable {

    int sorKoord = 0;
    boolean fut = true;

    public LengetVaszon() {

        sorKoord = 0;
        fut = true;
        new Thread(this).start();
    }

    public void run() {

        while(fut) {

            sorKoord = (sorKoord+2) % (getHeight()/3+20);
            repaint();

            try {
                Thread.sleep(100);
            } catch(InterruptedException e) {}

        }
    }

    public void keyPressed(int billenytű) {

        if(getGameAction(billenytű) ==
javax.microedition.lcdui.Canvas.UP)
            sorKoord = 0;
        else
            sorKoord = getHeight()/3+20;

        repaint();
    }

    public void paint(javax.microedition.lcdui.Graphics g) {

        g.setColor(0x00ffffff);
        g.fillRect(0, 0, getWidth(), getHeight());
```

```
        g.setColor(0x00000000);
        g.drawArc(getWidth()/3, 0, getWidth()/3, getHeight()/3, 0,
360);

        g.drawLine(0, sorKoord, getWidth()/2, getHeight()/3+20);
        g.drawLine(getWidth(), sorKoord, getWidth()/2,
getHeight()/3+20);

        g.drawLine(getWidth()/2, getHeight()/3, getWidth()/2,
2*getHeight()/3);
        g.drawLine(0, getHeight(), getWidth()/2, 2*getHeight()/3);
        g.drawLine(getWidth(), getHeight(), getWidth()/2,
2*getHeight()/3);

    }
}
```

Fontos megjegyeznünk, hogy a példában a `boolean fut;` változó bevezetésével csak jelezni akartuk és érdemben nem foglalkoztunk azzal, hogy a szálak megszüntetését mindig különösen figyelnie kell a mobil fejlesztőnek.

A következő példában a `GameCanvas` osztályt mutatjuk be, abból a szempontból, hogy használata megkönnyíti a játék állapotváltozásait vezérlő szál és a megjelenítés összehangolását.

```
public class LengetVaszon extends
javax.microedition.lcdui.game.GameCanvas
        implements Runnable {

    int sorKoord = 0;
    boolean fut = true;

    public LengetVaszon() {
        super(true);
        sorKoord = 0;
        fut = true;
        new Thread(this).start();
    }

    public void run() {

        javax.microedition.lcdui.Graphics g = getGraphics();

        while(fut) {

            int billenytű = getKeyStates();

            if((billenytű &
javax.microedition.lcdui.game.GameCanvas.UP_PRESSED) != 0)
                sorKoord = 0;
            else if((billenytű &
javax.microedition.lcdui.game.GameCanvas.DOWN_PRESSED) != 0)
                sorKoord = getHeight()/3+20;

            sorKoord = (sorKoord+2) % (getHeight()/3+20);

            g.setColor(0x00ffffff);
```

```
            g.fillRect(0, 0, getWidth(), getHeight());
            g.setColor(0x00000000);
            g.drawArc(getWidth()/3, 0, getWidth()/3, getHeight()/3,
0, 360);

            g.drawLine(0, sorKoord, getWidth()/2,
getHeight()/3+20);
            g.drawLine(getWidth(), sorKoord, getWidth()/2,
getHeight()/3+20);

            g.drawLine(getWidth()/2, getHeight()/3, getWidth()/2,
2*getHeight()/3);
            g.drawLine(0, getHeight(), getWidth()/2,
2*getHeight()/3);
            g.drawLine(getWidth(), getHeight(), getWidth()/2,
2*getHeight()/3);

            flushGraphics();

            try {
                Thread.sleep(100);
            } catch(InterruptedException e) {}

        }
    }
}
```

Fejlesszük tovább úgy a programot, hogy pálcikaemberünk legyen jóval kisebb, folyamatosan integessen és a kurzorral mozgathassuk mindeközben a képernyőn. S ha a felhasználótól jövő mozgatásra a képernyő egyik oldalán kimegy, akkor a szemben lévő oldalon jöjjön be.

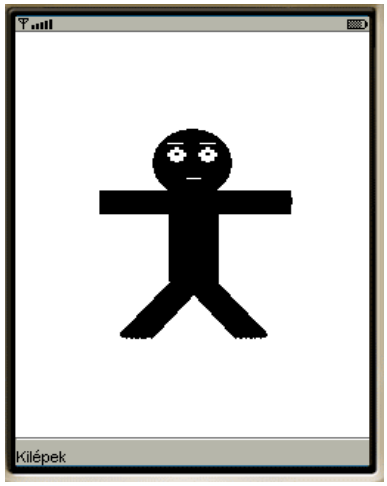
Majd fejlesszük még tovább a programot úgy, hogy a pálcikaember animációt ne mi rajzoljuk a vászonra, hanem külön – például a gimp programmal – készítsük el a néhány frame-ből álló animációt, majd ezeket a `.png` képeket az `Image` osztály statikus `createImage()` módszerével töltsük be és animáljuk! A projekt `.png` képerőforrásainak helyét a **NetBeans IDE** projektek fülében az aktuális projekt nevén egy jobb gomb nyomásával a `Properties` menüpontot kiválasztva a `Libraries & Resources` részben állíthatjuk be.

X.2.1.2 Pálcikaember sprite-ok

Fejlesszük most úgy tovább a programot, hogy az animációt a `Sprite` osztály segítségével végezzük el! Az animációt a `pislog.png` képfájlban készítjük el:

```
public class LengetVaszon extends
javax.microedition.lcdui.game.GameCanvas
        implements Runnable {

    javax.microedition.lcdui.game.Sprite fickó;
    javax.microedition.lcdui.Image animáció;

    boolean fut = true;
```

```
    public LengetVaszon() {
        super(true);
        fut = true;

        try {
            animáció =
                    javax.microedition.lcdui.Image.createImage("/pi
slog.png");
        } catch(java.io.IOException e) {}

        fickó = new javax.microedition.lcdui.game.Sprite(animáció,
150, 150);

        fickó.setFrameSequence(new int[]{0, 1, 2, 3, 2, 1, 0});
        fickó.defineReferencePixel(75, 75);
        fickó.setRefPixelPosition(getWidth()/2, getHeight()/2);

        new Thread(this).start();
    }

    public void run() {

        javax.microedition.lcdui.Graphics g = getGraphics();

        while(fut) {

            int billenytű = getKeyStates();

            fickó.paint(g);
            fickó.nextFrame();

            flushGraphics();

            try {
                Thread.sleep(100);
            } catch(InterruptedException e) {}
        }
    }
}
```



43. ábra: Animáció a Sprite osztállyal.

Fejlesszük úgy tovább a programot, hogy a sprite objektumot lehessen irányítani a telefon kurzor gombjaival!

Illetve készítsünk egy olyan továbbfejlesztést, amelyben két vagy több sprite objektum van a vásznon és érzékelik, ha összeütköznek! (Az egyik például véletlenszerűen mozog, a másikat a játékos irányítja.)

XI. Tudományos számítások

XI.1 A Pi

Dr. Arroway Elenor, Carl Sagan **Kapcsolat** című sci-fi regényének főhősnője, miután SETI programjában észlelte az idegen civilizációk rádió üzenetét és részt vett a sikeres kapcsolat kialakításában, most a Pi jegyeinek kifejtésében keres egy esetleges üzenetet.

Hasonló élményt élhetünk át az alig 10 éve [PI] publikált BBP [BBP ALG] algoritmussal mi magunk is!

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
 * pi_bbp.c, nbatfai@inf.unideb.hu
 *
 * A BBP (Bailey-Borwein-Plouffe) algoritmus a Pi hexa
 * jegyeinek meghatározására.
 * A program a David H. Bailey: The BBP Algorithm for Pi.
 * http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-alg.pdf
 * cikkben ismertetett BBP algoritmus megvalósítása.
 *
 * Fordítása, futtatása:
 * $ gcc pi_bbp.c -o pi_bbp -lm -std=c99
 * $ ./pi_bbp 0 80 1
 *
243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C8945
2821E638D01377
 * $ ./pi_bbp 1000000 1000001 20
 * 6C65E52CB858
 */

/*
 * Bináris hatványozás mod k,
 * a 16^n mod k értékének kiszámítása.
 *
 * n a kitevő.
 * k a modulus.
 */
long long int
binhatmod (long long int n, long long int k)
{
  long long int r = 1;

  long long int t = 1;
  while (t <= n)
    t *= 2;

  for (;;)
    {

      if (n >= t)
     {
```

```
       r = (16 * r) % k;
       n = n - t;
     }

      t = t / 2;

      if (t < 1)
     break;

      r = (r * r) % k;

    }

  return r;
}

/*
 * A hivatkozott David H. Bailey: The BBP Algorithm for Pi. cikk
 * alapján a {16^d Sj} részletösszeg kiszámítása, a {} a törtrészt
jelöli.
 *
 * A d+1. hexa jegytől számoljuk a hexa jegyeket.
 * A j az  Sj indexe.
 */
long double
Sj (long long int d, int j)
{

  long double sj = 0.0;
  long long int k;

  for (k = 0; k <= d; ++k)
    sj +=
      (long double) binhatmod (d - k, 8 * k + j) / (long double)
(8 * k + j);


  for (k = d + 1; k <= 2 * d; ++k)
    sj += powl (16.0, d - k) / (long double) (8 * k + j);


  return sj - floorl (sj);
}

/*
 * A hivatkozott David H. Bailey: The BBP Algorithm for Pi. cikk
 * alapján a {16^d Pi} = {4*{16^d S1} - 2*{16^d S4} - {16^d S5} -
{16^d S6}}
 * kiszámítása, a {} a törtrészt jelöli.
 *
 * A Pi hexa kifejtésének a d+1. hexa jegytől néhány jegy
előállítása.
 */
int
main (int argc, char **argv)
{

  long double pi_hkif = 0.0;
```

```
  long double s1 = 0.0;
  long double s4 = 0.0;
  long double s5 = 0.0;
  long double s6 = 0.0;

  long long int d, d_kezdo, d_befejezo;

  int jegy, jegyh, jegysz = 2;

  if (argc != 4)
    {
      printf
        ("Használat: ./pi_bbp mettől meddig max_jegy\nHasználati
példák: az első 1000 jegy egyesével ./pi_bbp 0 80 1\na 0-tól
számított 1000000. jegytől max húsz jegy ./pi_bbp 1000000 1000001
20\n");
      return -1;

    }

  d_kezdo = atoll (argv[1]);
  d_befejezo = atoll (argv[2]);
  jegysz = atoi (argv[3]);

  for (d = d_kezdo; d < d_befejezo; d += jegysz)
    {

      pi_hkif = 0.0;

      s1 = Sj (d, 1);
      s4 = Sj (d, 4);
      s5 = Sj (d, 5);
      s6 = Sj (d, 6);

      pi_hkif = 4.0 * s1 - 2.0 * s4 - s5 - s6;

      pi_hkif = pi_hkif - floorl (pi_hkif);

      for (jegyh = 0; jegyh < jegysz && pi_hkif != 0.0; ++jegyh)
     {
       jegy = (int) floorl (16.0 * pi_hkif);
       pi_hkif = 16.0 * pi_hkif - floorl (16.0 * pi_hkif);

       if (jegy < 10)
         printf ("%d", jegy);
       else
         printf ("%c", 'A' + jegy - 10);

       fflush (stdout);
     }
    }
     return 0;
}
```

Fordítsuk és futtassuk! Először a Pi hexa kifejtésének 1000 jegyét generáljuk le, majd a második futtatásban (a 0.-tól számítva) az egymilliomodik jegytől néhány jegyet.

```
$ gcc pi_bbp.c -o pi_bbp -lm -std=c99
$ ./pi_bbp 0 1000 1
243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89
452821E638D01377BE5466CF34E90C6CC0AC29B7C97C50DD3F84D5B5B5470917
9216D5D98979FB1BD1310BA698DFB5AC2FFD72DBD01ADFB7B8E1AFED6A267E96
BA7C9045F12C7F9924A19947B3916CF70801F2E2858EFC16636920D871574E69
A458FEA3F4933D7E0D95748F728EB658718BCD5882154AEE7B54A41DC25A59B5
9C30D5392AF26013C5D1B023286085F0CA417918B8DB38EF8E79DCB0603A180E
6C9E0E8BB01E8A3ED71577C1BD314B2778AF2FDA55605C60E65525F3AA55AB94
5748986263E8144055CA396A2AAB10B6B4CC5C341141E8CEA15486AF7C72E993
B3EE1411636FBC2A2BA9C55D741831F6CE5C3E169B87931EAFD6BA336C24CF5C
7A325381289586773B8F48986B4BB9AFC4BFE81B6628219361D809CCFB21A991
487CAC605DEC8032EF845D5DE98575B1DC262302EB651B8823893E81D396ACC5
0F6D6FF383F442392E0B4482A484200469C8F04A9E1F9B5E21C66842F6E96C9A
670C9C61ABD388F06A51A0D2D8542F68960FA728AB5133A36EEF0B6C137A3BE4
BA3BF0507EFB2A98A1F1651D39AF017666CA593E82430E888CEE8619456F9FB4
7D84A5C33B8B5EBEE06F75D885C12073401A449F56C16AA64ED3AA62363F7706
1BFEDF72429B023D37D0D724D00A1248DB0FEAD3
$ ./pi_bbp 1000000 1000001 20
6C65E52CB858
```

A BBP algoritmus a Pi hexadecimális kifejtésének tetszőleges jegyétől állítja elő a jegyeket, ezért kiválóan alkalmas párhuzamos vagy elosztott felhasználásra.

Ennek megfelelően az Operációs rendszerek laboron azzal a feladattal foglalkozunk, hogy egy több processzoros gépre az algoritmus számítását több folyamatban, illetve szálban osszuk szét, a Hálózatok laboron pedig azzal, hogy az algoritmus számítását több gép között osszuk szét.

XI.1.1 Pi több folyamattal, párhuzamosan

Dolgozzuk össze az iménti Pi-s programunkat (IPC tekintetében például) az *55*. oldal *Üzenetsorok* című pontjának a kernel üzenetsorát használó programjával! Úgy, hogy a Pi jegyeinek számítási részfeladatai legyenek az üzenetek, s az összes aktuális ilyen részfeladatot tegyük be ebbe a sorba, majd készítsünk annyi folyamatot, ahány processzorral rendelkezünk a gépben. A folyamatok a sorból folyamatosan kivéve a feladatokat végezzék a számításokat!

Két c és a két megfelelő fejléc állományt készítjük el. A `pi_bbp` az előző pontban megírt BBP algoritmus, a `pih_proc` a szóban forgó folyamatokat vezérli.

A `pi_bbp.h` fejléc állomány:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
 * A Pi hexa kifejtésének a d+1. hexa jegytől néhány jegy
előállítása és
 * kiírása a d_kezdo-d_befejezo.pih nevű fájlba. Az algoritmus
 * jegysz jegyenként lép.
 *
 * A BBP (Bailey-Borwein-Plouffe) algoritmus a Pi hexa
 * jegyeinek meghatározására.
 * A program a David H. Bailey: The BBP Algorithm for Pi.
```

```
 * http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-alg.pdf
 * cikkben ismertetett BBP algoritmus megvalósítása.
 */

int pi_bbp (long long d_kezdo, long long d_befejezo, int jegysz);
```

A `pi_bbp.c` állomány:

```
#include "pi_bbp.h"

/*
 * pi_bbp.c, nbatfai@inf.unideb.hu
 *
 * A BBP (Bailey-Borwein-Plouffe) algoritmus a Pi hexa
 * jegyeinek meghatározására.
 * A program a David H. Bailey: The BBP Algorithm for Pi.
 * http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-alg.pdf
 * cikkben ismertetett BBP algoritmus megvalósítása.
 *
 * int
 * pi_bbp (long long d_kezdo, long long d_befejezo, int jegysz)
 *
 * A Pi hexa kifejtésének a d+1. hexa jegytől néhány jegy
előállítása és
 * kiírása a d_kezdo-d_befejezo.pih nevű fájlba. Az algoritmus
 * jegysz jegyenként lép.
 *
 */

/*
 * Bináris hatványozás mod k,
 * a 16^n mod k értékének kiszámítása.
 *
 * n a kitevő.
 * k a modulus.
 */
long long int
binhatmod (long long int n, long long int k)
{
  long long int r = 1;

  long long int t = 1;
  while (t <= n)
    t *= 2;

  for (;;)
    {

      if (n >= t)
     {
       r = (16 * r) % k;
       n = n - t;
     }

      t = t / 2;

      if (t < 1)
```

```
     break;

      r = (r * r) % k;

    }

  return r;
}

/*
 * A hivatkozott David H. Bailey: The BBP Algorithm for Pi. cikk
 * alapján a {16^d Sj} részletösszeg kiszámítása, a {} a törtrészt
jelöli.
 *
 * A d+1. hexa jegytől számoljuk a hexa jegyeket.
 * A j az  Sj indexe.
 */
long double
Sj (long long int d, int j)
{

  long double sj = 0.0;
  long long int k;

  for (k = 0; k <= d; ++k)
    sj +=
      (long double) binhatmod (d - k, 8 * k + j) / (long double)
(8 * k + j);


  for (k = d + 1; k <= 2 * d; ++k)
    sj += powl (16.0, d - k) / (long double) (8 * k + j);


  return sj - floorl (sj);
}

/*
 * A hivatkozott David H. Bailey: The BBP Algorithm for Pi. cikk
 * alapján a {16^d Pi} = {4*{16^d S1} - 2*{16^d S4} - {16^d S5} -
{16^d S6}}
 * kiszámítása, a {} a törtrészt jelöli.
 *
 * A Pi hexa kifejtésének a d+1. hexa jegytől néhány jegy
előállítása és
 * kiírása a d_kezdo-d_befejezo.pih nevű fájlba.
 */
int
pi_bbp (long long d_kezdo, long long d_befejezo, int jegysz)
{

  long double pi_hkif = 0.0;

  long double s1 = 0.0;
  long double s4 = 0.0;
  long double s5 = 0.0;
  long double s6 = 0.0;
```

```
  long long int d;

  int jegy, jegyh;

  FILE *fp;
  char buffer[1024];

  snprintf (buffer, 1024, "%lld-%lld.pih", d_kezdo, d_befejezo);

  if ((fp = fopen (buffer, "w")) == NULL)
    return -1;

  for (d = d_kezdo; d < d_befejezo; d += jegysz)
    {

      pi_hkif = 0.0;

      s1 = Sj (d, 1);
      s4 = Sj (d, 4);
      s5 = Sj (d, 5);
      s6 = Sj (d, 6);

      pi_hkif = 4.0 * s1 - 2.0 * s4 - s5 - s6;

      pi_hkif = pi_hkif - floorl (pi_hkif);

      for (jegyh = 0; jegyh < jegysz && pi_hkif != 0.0; ++jegyh)
     {
       jegy = (int) floorl (16.0 * pi_hkif);
       pi_hkif = 16.0 * pi_hkif - floorl (16.0 * pi_hkif);

       if (jegy < 10)
         fprintf (fp, "%d", jegy);
       else
         fprintf (fp, "%c", 'A' + jegy - 10);

       fflush (stdout);
     }
    }
  fclose (fp);
  return 0;
}
```

A `pih_proc.h` fejléc állomány:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <errno.h>
#include "pi_bbp.h"

/*
 * Egy folyamat számára előkészített számolási részfeladat.
```

```
 * Tartalmazza, hogy mettől meddig és hány jegyenként
 * dolgozzon a BBP algoritmus.
 */
struct reszfeladat
{
  long mtype;
  long long mettol;
  int jegysz;
  long long meddig;
};
```

A `pih_proc.c` állomány:

```
#include "pih_proc.h"

int
main (int argc, char **argv)
{
  int gyermekem_pid;
  int uzenetsor;
  int jegysz, procsz, psz;
  int *proc_pid;
  long long int d, d_kezdo, d_befejezo;
  long mennyit = 100;

  if (argc != 5)
    {
      printf
     ("Hasznalat: ./pi_bbp_proc mettol meddig max_jegy
proc_szam\nHasznalati peldak: az elso 1000 jegy egyesevel 5
folyamattal./pi_bbp 0 80 1 5\n");
      return -1;

    }
  // Parancssor argumentumok átvétele
  d_kezdo = atoll (argv[1]);
  d_befejezo = atoll (argv[2]);
  jegysz = atoi (argv[3]);
  procsz = atoi (argv[4]);

  if ((uzenetsor =
       msgget (ftok (".", 43), IPC_CREAT | S_IRUSR | S_IWUSR)) ==
-1)
    {
      perror ("msgget");
      exit (EXIT_FAILURE);
    }
  // Számolási részfeladatok létrehozása valamilyen
  // politikával, most a d tekintetében egyforma darabokra vágás
  d = d_kezdo;
  while (d < d_befejezo)
    {
      struct reszfeladat rf;
      rf.mtype = 42;
      rf.mettol = d;
      rf.jegysz = jegysz;
      rf.meddig = d + mennyit - 1;
```

```
      if (msgsnd (uzenetsor, &rf, sizeof (struct reszfeladat) -
            sizeof (long), 0))
     {
       perror ("msgsnd");
       exit (EXIT_FAILURE);
     }

      d += mennyit;
    }
  // A számolást végző adott számú gyermekfolyamat létrehozása
  proc_pid = (int *) malloc (procsz * sizeof (int));
  psz = procsz;
  while (psz--)
    {
      printf ("%d szamolo folyamat letrehozasa\n", procsz - psz);
      fflush (stdout);
      if ((gyermekem_pid = fork ()) == 0)
     {
       for (;;)
         {
           struct reszfeladat rf;
           if (msgrcv (uzenetsor, &rf, sizeof (struct reszfeladat)
-
                  sizeof (long), 42, IPC_NOWAIT) < 0)
          {
            if (errno == ENOMSG)
              {
                printf ("%d> nincs tobb reszfeladat, kilepek.\n",
                      getpid ());
                exit (EXIT_SUCCESS);
              }
            perror ("msgrcv");
            exit (EXIT_FAILURE);
          }

           printf ("%d> %d-%d szamolasa indul\n", getpid (),
rf.mettol,
                rf.meddig);
           if (pi_bbp (rf.mettol, rf.meddig, rf.jegysz) != -1)
          printf ("%d> %d-%d szamolasa kesz\n", getpid (),
rf.mettol,
                rf.meddig);
           else
          printf ("%d> %d-%d szamolasa sikertelen\n", getpid (),
                rf.mettol, rf.meddig);

         }

     }
      else if (gyermekem_pid > 0)
     {
       *(proc_pid + psz) = gyermekem_pid;
     }
      else
     {
       perror ("fork");
       exit (EXIT_FAILURE);
     }
```

```
    }
  // Várakozás a számítások befejezésére
  while (procsz--)
    waitpid (*(proc_pid + procsz), NULL, 0);

  free (proc_pid);

  if (msgctl (uzenetsor, IPC_RMID, NULL))
    {
      perror ("msgctl");
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

A programot fordítva és futtatva:

```
$ gcc pi_bbp.c pih_proc.c -o pih -lm
$ ./pih 1 1000 1 5
1 szamolo folyamat letrehozasa
24565> 1-100 szamolasa indul
24565> 1-100 szamolasa kesz
24565> 101-200 szamolasa indul
2 szamolo folyamat letrehozasa
24566> 201-300 szamolasa indul
3 szamolo folyamat letrehozasa
24565> 101-200 szamolasa kesz
24565> 301-400 szamolasa indul
24567> 401-500 szamolasa indul
4 szamolo folyamat letrehozasa
24568> 501-600 szamolasa indul
5 szamolo folyamat letrehozasa
24569> 601-700 szamolasa indul
24566> 201-300 szamolasa kesz
24566> 701-800 szamolasa indul
24565> 301-400 szamolasa kesz
24565> 801-900 szamolasa indul
24568> 501-600 szamolasa kesz
24568> 901-1000 szamolasa indul
24566> 701-800 szamolasa kesz
24566> nincs tobb reszfeladat, kilepek.
24567> 401-500 szamolasa kesz
24567> nincs tobb reszfeladat, kilepek.
24569> 601-700 szamolasa kesz
24569> nincs tobb reszfeladat, kilepek.
24565> 801-900 szamolasa kesz
24565> nincs tobb reszfeladat, kilepek.
24568> 901-1000 szamolasa kesz
24568> nincs tobb reszfeladat, kilepek.
```

Nincs más dolgunk, mint összefésülni a létrejött `.pih` fájlokat:

```
$ ls *.pih
101-200.pih  201-300.pih  401-500.pih  601-700.pih  801-900.pih
1-100.pih    301-400.pih  501-600.pih  701-800.pih  901-1000.pih
```

Emeljük meg a kiszámítandó feladat (és ezzel együtt a forrásban a részfeladatok, `mennyit = 1000;`) méretét és egy másik ablakban futtassuk az `ipcs` parancsot a program üzeme alatt:

```
$ ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x2b0089e6 0          neuro      600        2232         93
```

s kis idő múlva újra adjuk ki a parancsot, jól látszik, ahogyan a folyamatok fogyasztják a részfeladatokat a sorból:

```
$ ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x2b0089e6 0          neuro      600        2112         88
```

XI.1.2 Pi több géppel, elosztottan

A korábbi *106*. oldal *Párhuzamos, azaz konkurens, folyamatokkal* című pontjának szerveréből indulunk ki, ezt alakítjuk át az elosztott számításunkat adminisztráló szerverré. A kliensek jelentkezését végző folyamatok közötti IPC eszközéül a *59*. oldal *Osztott memória* című példában bevezetett osztott memóriát használjuk. Ezen a közös memóriaterületen tartjuk a számolási részfeladatokat. A program jelen, első verziójában a kölcsönös kizárással nem foglalkozunk, majd a második változatban védjük ezt a közös területet szemaforokkal.

Az alábbi forrásokat fejlesztjük ki: `dpih.h`, `dpih.c`, `dpih_szerver.c`, `dpih_kliens.c` és persze felhasználjuk az előző pont `pi_bbp.c`, `pi_bbp.h` állományait, amit minimálisan (a kimentett hexa jegyek fájlnevének tekintetében) módosítunk is.

A szervert és a klienst így fordítsuk:

```
$ gcc dpih_szerver.c dpih.c -o dpih_szerver -lnsl
$ gcc dpih_kliens.c pi_bbp.c dpih.c -o dpih_kliens -lnsl -lm
```

Futtassuk a szervert:

```
$ ./dpih_szerver 10000 10500 2 42
127.0.0.1:2006
```

Majd egy másik ablakban egy klienst

```
$ ./dpih_kliens localhost 2006
42-es szamitast megkezdem, 0. reszf.: 10000 10099 2
42-es szamitast megkezdem, 2. reszf.: 10200 10299 2
42-es szamitast megkezdem, 4. reszf.: 10400 10499 2
Nincs szamitas
```

Egy újabb másik ablakban egy másik klienst:

```
$  ./dpih_kliens localhost 2006
42-es szamitast megkezdem, 1. reszf.: 10100 10199 2
42-es szamitast megkezdem, 3. reszf.: 10300 10399 2
42-es szamitast megkezdem, 4. reszf.: 10400 10499 2
Nincs szamitas
```

Közben mit logolt a szerverünk:

```
$ ./dpih_szerver 10000 10500 2 42
127.0.0.1:2006
     <-> 127.0.0.1:60485 VISZ keresre valasz: 42 0 10000 2 10099
     <-> 127.0.0.1:60486 VISZ keresre valasz: 42 1 10100 2 10199
     <-> 127.0.0.1:60487 HOZ: 42-es szamitas 0. reszf.
     <-> 127.0.0.1:60488 VISZ keresre valasz: 42 2 10200 2 10299
     <-> 127.0.0.1:60489 HOZ: 42-es szamitas 1. reszf.
     <-> 127.0.0.1:60490 VISZ keresre valasz: 42 3 10300 2 10399
     <-> 127.0.0.1:60491 HOZ: 42-es szamitas 2. reszf.
     <-> 127.0.0.1:60492 VISZ keresre valasz: 42 4 10400 2 10499
     <-> 127.0.0.1:60493 HOZ: 42-es szamitas 3. reszf.
     <-> 127.0.0.1:60494 VISZ keresre valasz: 42 4 10400 2 10499
     <-> 127.0.0.1:60495 HOZ: 42-es szamitas 4. reszf.
     <-> 127.0.0.1:60496 VISZ keresre valasz: -1 Nincs tobb
reszfeladat.
     <-> 127.0.0.1:60497 HOZ: 42-es szamitas 4. reszf.
     <-> 127.0.0.1:60498 VISZ keresre valasz: -1 Nincs tobb
reszfeladat.
```

Közben kiadhatunk egy `ipcs` parancsot, hogy lássuk az osztott memória használatát.

Az eredmény:

```
$ ls *.pih
10000-10099.pih  kliens_szamitas_10000-10099.pih
10100-10199.pih  kliens_szamitas_10100-10199.pih
10200-10299.pih  kliens_szamitas_10200-10299.pih
10300-10399.pih  kliens_szamitas_10300-10399.pih
10400-10499.pih  kliens_szamitas_10400-10499.pih
```

A `kliens_szamitas_` előtagú fájlok a számításokat végző kliensek átmeneti fájljai, az e nélküliek a szerverre felküldött és ott kimentett fájlok.

`dpih.h`

```
#define SZERVER_PORT 2006
#define SZERVER_SOR_MERET 10
```

```
#define UJ 0
#define KIADOTT 1
#define KESZ 2
#define BUFFER_MERET 1024

typedef struct reszfeladat
{
  long long mettol;
  int jegysz;
  long long meddig;
  int allapot;

} RESZFELADAT, *RESZFELADAT_MUTATO;

int szam_beolv (int kliens);
int szamll_beolv (int kliens);
```

dpih.c

```
#include "dpih.h"

int
szam_beolv (int kliens)
{
  char buffer[BUFFER_MERET];
  int i = 0;

  while (read (kliens, buffer + i, 1) > 0)
    {
      if (*(buffer + i) == ' ')
     break;
      ++i;
    }
  *(buffer + i) = '\0';

  return atoi (buffer);
}

int
szamll_beolv (int kliens)
{
  char buffer[BUFFER_MERET];
  int i = 0;

  while (read (kliens, buffer + i, 1) > 0)
    {
      if (*(buffer + i) == ' ')
     break;
      ++i;
    }
  *(buffer + i) = '\0';

  return atoll (buffer);
}
```

dpih_szerver.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/shm.h>
#include "dpih.h"

int szamitas = 42, reszfsz;
RESZFELADAT_MUTATO reszfeladatok;
void *osztott_memoria_terulet;

RESZFELADAT_MUTATO
vanFeladat (int *sorszam)
{
  int i;

  for (i = 0; i < reszfsz; ++i)
    if (reszfeladatok[i].allapot == UJ)
      {
     reszfeladatok[i].allapot = KIADOTT;
     *sorszam = i;
     return reszfeladatok + i;
      }
  for (i = 0; i < reszfsz; ++i)
    if (reszfeladatok[i].allapot == KIADOTT)
      {
     *sorszam = i;
     return reszfeladatok + i;
      }

  return NULL;

}

void
kiment (int kliens, int sorszam)
{
  int fd;
  char buffer[1024];
  int n;

  snprintf (buffer, 1024, "%lld-%lld.pih",
reszfeladatok[sorszam].mettol,
         reszfeladatok[sorszam].meddig);

  fd = open (buffer, O_CREAT | O_WRONLY,
          S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);
```

```
  while ((n = read (kliens, buffer, 1024)) > 0)
    write (fd, buffer, n);

  close (fd);

}

/* A hasznalt kliens-szerver modellbeli protokollunk:
 * a kliens kerese:
 *  VISZ
 *    egy reszfeladatot ker a kliens
 *  HOZ SZAMITAS_KOD RESZFELADAT_SORSZAM HOZOTT_HEXA_JEGYEK
 *    egy kiszamolt reszfeladatot hoz a kliens
 * a szerver valasza a kliens VISZ keresere:
 *  SZAMITAS_KOD RESZFELADAT_SORSZAM METTOL JEGYSZ MEDDIG
 *    negativ szamitas kod eseten SZAMITAS_KOD UZENET
 */
int
kiszolgal (int kliens)
{
  char buffer[BUFFER_MERET];
  int sorszam, sz;
  RESZFELADAT_MUTATO reszfeladat;

  if (read (kliens, buffer, 1) < 0)
    {
      perror ("read");
      exit (EXIT_FAILURE);
    }

  if (buffer[0] == 'H')         // HOZ
    {
      while (read (kliens, buffer, 1) > 0)
     {
       if (*buffer == ' ')
         break;
     }
      sz = szam_beolv (kliens);
      sorszam = szam_beolv (kliens);

      printf (" HOZ: %d-es szamitas %d. reszf.\n", sz, sorszam);
      fflush (stdout);

      if (szamitas == sz)
     {
       kiment (kliens, sorszam);
       reszfeladatok[sorszam].allapot = KESZ;
     }
    }
  else                    // VISZ
    {
      if ((reszfeladat = vanFeladat (&sorszam)) != NULL)
     snprintf (buffer, BUFFER_MERET, "%d %d %lld %d %lld",
             szamitas, sorszam, reszfeladat->mettol, reszfeladat-
>jegysz,
             reszfeladat->meddig);
      else
```

```
     snprintf (buffer, BUFFER_MERET, "%d Nincs tobb reszfeladat.",
-1);

      printf (" VISZ keresre valasz: %s\n", buffer);
      fflush (stdout);
      write (kliens, buffer, strlen (buffer));
    }
  return 0;
}

void
zombi_elharito (int sig)
{
  signal (SIGCHLD, zombi_elharito);
  while (wait (NULL) > 0)
    ;
}

void
szerver ()
{
  int kapu_figyelo, kapcsolat, kliensm, gyermekem_pid, sockoptval
= 1;
  struct sockaddr_in szerver, kliens;

  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton ("127.0.0.1", &(szerver.sin_addr));
  szerver.sin_port = htons (SZERVER_PORT);
  if ((kapu_figyelo = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP))
== -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  setsockopt (kapu_figyelo, SOL_SOCKET, SO_REUSEADDR,
          (void *) &sockoptval, sizeof (sockoptval));
  if (bind (kapu_figyelo, (struct sockaddr *) &szerver,
        sizeof (szerver)) == -1)
    {
      perror ("bind");
      exit (EXIT_FAILURE);
    }
  if (listen (kapu_figyelo, SZERVER_SOR_MERET) == -1)
    {
      perror ("listen");
      exit (EXIT_FAILURE);
    }
  printf ("%s:%d\n", inet_ntoa (szerver.sin_addr), ntohs
(szerver.sin_port));
  signal (SIGCHLD, zombi_elharito);
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));
      if ((kapcsolat = accept (kapu_figyelo,
                       (struct sockaddr *) &kliens,
                       (socklen_t *) & kliensm)) == -1)
     {
```

```
       perror ("accept");
       exit (EXIT_FAILURE);
     }
      printf ("     <-> %s:%d",
           inet_ntoa (kliens.sin_addr), ntohs (kliens.sin_port));
      fflush (stdout);

      if ((gyermekem_pid = fork ()) == 0)
     {
       close (kapu_figyelo);
       if (kiszolgal (kapcsolat) == -1)
         {
           perror ("kiszolgal");
         }
       close (kapcsolat);
       if (shmdt (osztott_memoria_terulet) == -1)
         {
           perror ("shmdt");
           exit (EXIT_FAILURE);
         }
       exit (EXIT_SUCCESS);
     }
      else if (gyermekem_pid > 0)
     {
       // wait(&statusz); e miatt kezeljuk a SIGCHLD jelet,
       // l. a Zombik fejezetet!
       close (kapcsolat);
     }
      else
     {
       close (kapcsolat);
       perror ("fork");
       exit (EXIT_FAILURE);
     }
    }
}

int
main (int argc, char **argv)
{
  int i, jegysz, szamsz;
  long long int d_kezdo, d_befejezo;
  long mennyit = 100;
  int osztott_memoria;

  if (argc != 5)
    {
      printf
     ("Hasznalat: ./dpih_szerver mettol meddig
szamitas_szama\nHasznalati peldak:\n");
      return -1;

    }

  d_kezdo = atoll (argv[1]);
  d_befejezo = atoll (argv[2]);
  jegysz = atoi (argv[3]);
  szamsz = atoi (argv[4]);
```

```
  reszfsz = (d_befejezo - d_kezdo) / mennyit;

  if ((osztott_memoria =
       shmget (ftok (".", 44), reszfsz * sizeof (RESZFELADAT),
            IPC_CREAT | S_IRUSR | S_IWUSR)) == -1)
    {
      perror ("shmget");
      exit (EXIT_FAILURE);
    }
  if ((osztott_memoria_terulet = shmat (osztott_memoria, NULL, 0))
< 0)
    {
      perror ("shmat");
      exit (EXIT_FAILURE);
    }

  reszfeladatok = (RESZFELADAT_MUTATO) osztott_memoria_terulet;

  for (i = 0; i < reszfsz; ++i)
    {
      reszfeladatok[i].allapot = UJ;
      reszfeladatok[i].mettol = d_kezdo + i * mennyit;
      reszfeladatok[i].jegysz = jegysz;
      reszfeladatok[i].meddig = d_kezdo + (i + 1) * mennyit - 1;
    }

  szerver ();
}
```

dpih_kliens.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

#include "dpih.h"

int szamitas;

int
hoz (char *hoszt, int port, RESZFELADAT_MUTATO reszfeladat)
{

  int kapu, olvasva;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET];

  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton (hoszt, &(szerver.sin_addr));
```

```
  szerver.sin_port = htons (port);
  if ((kapu = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (connect (kapu, (struct sockaddr *) &szerver, sizeof
(szerver)) == -1)
    {
      perror ("connect");
      exit (EXIT_FAILURE);
    }

  snprintf (buffer, BUFFER_MERET, "VISZ");
  write (kapu, buffer, strlen (buffer));

  szamitas = szam_beolv (kapu);
  if (szamitas < 0)
    {
      printf ("Nincs szamitas");
      return szamitas;
    }

  reszfeladat->allapot = szam_beolv (kapu);
  reszfeladat->mettol = szamll_beolv (kapu);
  reszfeladat->jegysz = szam_beolv (kapu);
  reszfeladat->meddig = szamll_beolv (kapu);

  close (kapu);

  return szamitas;
}

int
visz (char *hoszt, int port, RESZFELADAT_MUTATO reszfeladat)
{

  int kapu, olvasva;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET];
  int fd, n;

  memset ((void *) &szerver, 0, sizeof (szerver));
  szerver.sin_family = AF_INET;
  inet_aton (hoszt, &(szerver.sin_addr));
  szerver.sin_port = htons (port);
  if ((kapu = socket (PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
    {
      perror ("socket");
      exit (EXIT_FAILURE);
    }
  if (connect (kapu, (struct sockaddr *) &szerver, sizeof
(szerver)) == -1)
    {
      perror ("connect");
      exit (EXIT_FAILURE);
    }
```

```
  snprintf (buffer, BUFFER_MERET, "HOZ %d %d ", szamitas,
         reszfeladat->allapot);
  write (kapu, buffer, strlen (buffer));

  snprintf (buffer, BUFFER_MERET, "kliens_szamitas_%lld-%lld.pih",
         reszfeladat->mettol, reszfeladat->meddig);

  fd = open (buffer, O_RDONLY);

  while ((n = read (fd, buffer, BUFFER_MERET)) > 0)
    write (kapu, buffer, n);

  close (fd);
  close (kapu);

  return szamitas;
}


int
main (int argc, char **argv)
{
  int port;
  RESZFELADAT reszfeladat;

  if (argc != 3)
    {
      printf ("Hasznalat: ./dpih_kliens hoszt port\n");
      return -1;
    }

  port = atoi (argv[2]);

  while ((szamitas = hoz (argv[1], port, &reszfeladat)) > 0)
    {

      printf ("%d-es szamitast megkezdem, %d. reszf.: %lld %lld
%d\n",
           szamitas, reszfeladat.allapot, reszfeladat.mettol,
           reszfeladat.meddig, reszfeladat.jegysz);
      fflush (stdout);

      pi_bbp (reszfeladat.mettol, reszfeladat.meddig,
reszfeladat.jegysz);

      visz (argv[1], port, &reszfeladat);

    }

  exit (EXIT_SUCCESS);
}
```

pi_bbp.c

```
#include "pi_bbp.h"

/*
 * pi_bbp.c, nbatfai@inf.unideb.hu
```

```
 *
 * A BBP (Bailey-Borwein-Plouffe) algoritmus a Pi hexa
 * jegyeinek meghatározására.
 * A program a David H. Bailey: The BBP Algorithm for Pi.
 * http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-alg.pdf
 * cikkben ismertetett BBP algoritmus megvalósítása.
 *
 * int
 * pi_bbp (long long d_kezdo, long long d_befejezo, int jegysz)
 *
 * A Pi hexa kifejtésének a d+1. hexa jegytől néhány jegy
előállítása és
 * kiírása a d_kezdo-d_befejezo.pih nevű fájlba. Az algoritmus
 * jegysz jegyenként lép.
 *
 */

/*
 * Bináris hatványozás mod k,
 * a 16^n mod k értékének kiszámítása.
 *
 * n a kitevő.
 * k a modulus.
 */
long long int
binhatmod (long long int n, long long int k)
{
  long long int r = 1;

  long long int t = 1;
  while (t <= n)
    t *= 2;

  for (;;)
    {

      if (n >= t)
     {
       r = (16 * r) % k;
       n = n - t;
     }

      t = t / 2;

      if (t < 1)
     break;

      r = (r * r) % k;

    }

  return r;
}

/*
 * A hivatkozott David H. Bailey: The BBP Algorithm for Pi. cikk
 * alapján a {16^d Sj} részletösszeg kiszámítása, a {} a törtrészt
jelöli.
```

```
 *
 * A d+1. hexa jegytől számoljuk a hexa jegyeket.
 * A j az  Sj indexe.
 */
long double
Sj (long long int d, int j)
{

  long double sj = 0.0;
  long long int k;

  for (k = 0; k <= d; ++k)
    sj +=
      (long double) binhatmod (d - k, 8 * k + j) / (long double)
(8 * k + j);


  for (k = d + 1; k <= 2 * d; ++k)
    sj += powl (16.0, d - k) / (long double) (8 * k + j);


  return sj - floorl (sj);
}

/*
 * A hivatkozott David H. Bailey: The BBP Algorithm for Pi. cikk
 * alapján a {16^d Pi} = {4*{16^d S1} - 2*{16^d S4} - {16^d S5} -
{16^d S6}}
 * kiszámítása, a {} a törtrészt jelöli.
 *
 * A Pi hexa kifejtésének a d+1. hexa jegytől néhány jegy
előállítása és
 * kiírása a d_kezdo-d_befejezo.pih nevű fájlba.
 */
int
pi_bbp (long long d_kezdo, long long d_befejezo, int jegysz)
{

  long double pi_hkif = 0.0;

  long double s1 = 0.0;
  long double s4 = 0.0;
  long double s5 = 0.0;
  long double s6 = 0.0;

  long long int d;

  int jegy, jegyh;

  FILE *fp;
  char buffer[1024];

  snprintf (buffer, 1024, "kliens_szamitas_%lld-%lld.pih",
d_kezdo,
          d_befejezo);

  if ((fp = fopen (buffer, "w")) == NULL)
    return -1;
```

```
  for (d = d_kezdo; d < d_befejezo; d += jegysz)
    {

      pi_hkif = 0.0;

      s1 = Sj (d, 1);
      s4 = Sj (d, 4);
      s5 = Sj (d, 5);
      s6 = Sj (d, 6);

      pi_hkif = 4.0 * s1 - 2.0 * s4 - s5 - s6;

      pi_hkif = pi_hkif - floorl (pi_hkif);

      for (jegyh = 0; jegyh < jegysz && pi_hkif != 0.0; ++jegyh)
     {
       jegy = (int) floorl (16.0 * pi_hkif);
       pi_hkif = 16.0 * pi_hkif - floorl (16.0 * pi_hkif);

       if (jegy < 10)
         fprintf (fp, "%d", jegy);
       else
         fprintf (fp, "%c", 'A' + jegy - 10);

       fflush (stdout);
     }
    }
  fclose (fp);
  return 0;
}
```

pi_bbp.h

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/*
 * A Pi hexa kifejtésének a d+1. hexa jegytől néhány jegy
előállítása és
 * kiírása a d_kezdo-d_befejezo.pih nevű fájlba. Az algoritmus
 * jegysz jegyenként lép.
 *
 * A BBP (Bailey-Borwein-Plouffe) algoritmus a Pi hexa
 * jegyeinek meghatározására.
 * A program a David H. Bailey: The BBP Algorithm for Pi.
 * http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-alg.pdf
 * cikkben ismertetett BBP algoritmus megvalósítása.
 */

int pi_bbp (long long d_kezdo, long long d_befejezo, int jegysz);
```

A példa következő továbbfejlesztésben kölcsönös kizárással érjük el a megosztott memórián elhelyezett adatszerkezetet, a *46*. oldalon, a *Szemaforok* című pontban bevezetett szemaforokkal dolgozunk. A példán több, más apróbb javítást is végzünk.

XII. Kvantum számítások

XII.1 Bevezető kísérletek

XII.1.1 Kétrés kísérletek

A klasszikus változatot Young 1801-ben végezte, azóta elvégezték elektronokkal, atomokkal, molekulákkal is, [Kvantum/DS]-ben például a hélium atommal végzett kísérletet ismerhetjük meg.
A rejtélyekbe bevezető kísérlet:
(Wheeler késletetett választásos kísérlete)

XII.1.2 Egyrészecskés interferencia a Mach-Zehnder interferométerben

A kísérlet:

XII.1.3 Foton polarizációs kísérlet

A kísérlet:

XII.1.4 Einstein-Podolsky-Rosen paradoxon

Avagy a gondolatkísérlet visszaüt :) A kísérlet:

(Bell egyenlőtlenség, lokalitás, rejtett paraméterek, Aspect kísérlet.)

XII.2 Bevezető elmélet

Az absztrakt Hilbert tér, önadjungált, unitér, kommutátor.
Hullámfüggvény, állapot vektor, a fizikai mennyiségek és értékeik..
A hullámfüggvény időfejlődése, a Schrödinger egyenlet.
Dirac **bracket**.

XII.2.1 Kvantum bit, kvantum regiszter

Qubit (quantum bit), tenzor szorzat. Párhuzamosság, mérés. Kapcsolat (entanglement), lokalitás.

Mutassuk meg, hogy $|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$.

XII.2.1.1 Feladat – qubit nem másolható

Mutassuk meg, hogy tetszőleges qubit nem másolható, azaz

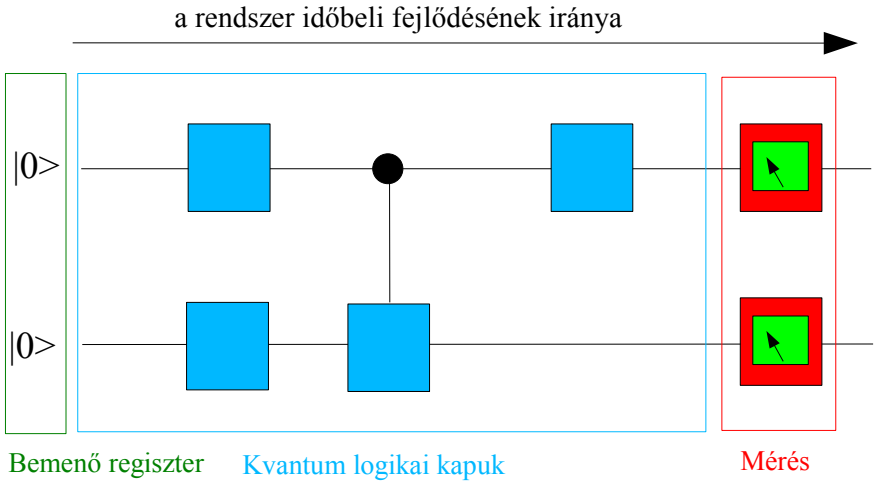
$!\exists U$ *unitér, hogy* $U(|q0\rangle) = |qq\rangle, |q\rangle \in H.$

Majd vessük össze e tapasztalatunkat a másoló logikai hálózattal, külön ellenőrizzük, mit ad például $|1\rangle \otimes |0\rangle$ -ra!

XII.2.2 Kvantum számítási modellek

XII.2.2.1 Kvantum logikai hálózatok

(Quantum circuits)



Ábra 44: Kvantum logikai hálózatok ábrázolása

XII.2.2.1.1 Kvantum logikai kapuk

(Quantum gates)

Egy bemenetű kapuk: NOT, fázis eltolás, Hadamard (H), Pauli mátrixok

Két bemenetű kapuk: vezérelt U (CU), vezérelt NOT (CNOT)

Három bemenetű kapuk: Toffoli

XII.2.2.1.2 Feladat – Mach-Zehnder egyrészecskés interferencia

Ábra 45: Egyrészecskés interferencia

Számítsuk ki mit ad a következő hálózat:

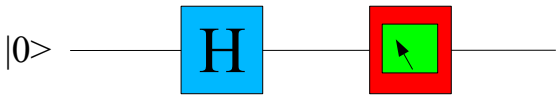
$\frac{1}{2}((1+e^{i\varphi})|0>+(1-e^{i\varphi})|1>)$ ezt kaptuk? Mit jelent ez a $\varphi=0$ és a $\varphi=\pi$ esetekben?
Hogyan kapcsolódik az egyrészecskés interferencia a Mach-Zehnder interferométerben című kísérlethez. (A további részletek iránt érdeklődők figyelmébe: [AE2].)

XII.2.2.2 Kvantum Turing gép

Turing gép, nem determinisztikus Turing gép, randomizált Turing gép.
Kolmogorov bonyolultság, a véletlen sorozat fogalma.

XII.2.2.2.1 Mit tud a kvantum számítógép, amit a Turing gép nem?

Feldobni egy érmét:

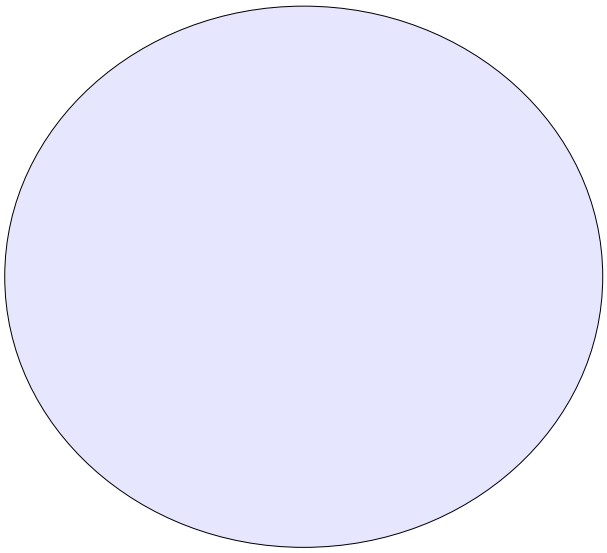


Ábra 46: Egy szabályos érme feldobása

XII.2.2.2.2 Kvantum Kolmogorov bonyolultság

XII.2.3 Bonyolultsági osztályok

P, **QP**, **BPP**, **BQP**



Ábra 47: Bonyolultsági osztályok

XII.2.4 Kvantum algoritmusok

XII.2.4.1 Az első – Deutsch konstans orákuluma

A Deutschtól származó [Kvantum/DD] feladatot ma már a legtöbb bevezető munkában megtaláljuk: [Kvantum/AE1,2/DE/TQ/DL].

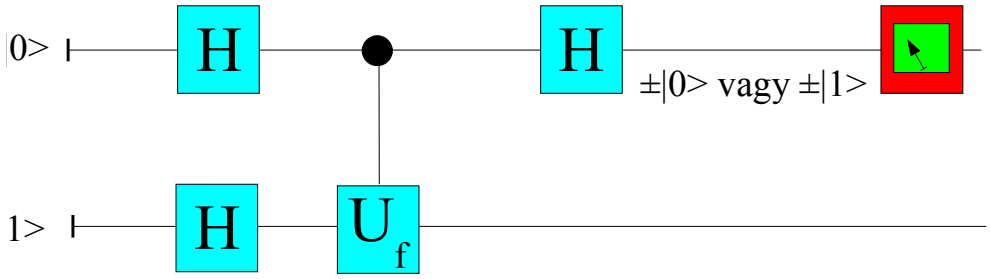
Legyen $f:\{0,1\}\rightarrow\{0,1\}$, ekkor f az alábbiak egyike lehet:

x	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

$f = f_{1,}\, f_4$ esetén f konstans.

- Hány kérdésből tudjuk megmondani, hogy f konstans-e?
- Illetve hány mérés kell, ha egy kvantum számítógépet kérdezünk?

$$\mp\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad \text{ha f nem konstans}$$
$$\mp\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{ha f konstans}$$



Ábra 48: Konstans orákulum

XII.2.4.1.1 Feladat – orákulum

Végezzük el a számítás részleteit!

$$H|0>=\frac{1}{\sqrt{2}}(|0>+|1>)$$

$$H|1>=\frac{1}{\sqrt{2}}(|0>-|1>)$$

$$U_f(\frac{1}{\sqrt{2}}(|0>+|1>)\otimes\frac{1}{\sqrt{2}}(|0>-|1>))=$$

$$\frac{1}{2}U_f(|0>\otimes|0>-|0>\otimes|1>+|1>\otimes|0>-|1>\otimes|1>),$$

$$mivel\ U_f(|x>\otimes|y>)=|x>\otimes|y\oplus f(x)>,\ így$$

$$=\frac{1}{2}(|0>\otimes|0\oplus f(0)>-|0>\otimes|1\oplus f(0)>+|1>\otimes|0\oplus f(1)>-|1>\otimes|1\oplus f(1)>)$$

$$=\frac{1}{2}(|0>\otimes|f(0)>-|0>\otimes|1-f(0)>+|1>\otimes|f(1)>-|1>\otimes|1-f(1)>)$$

$$=\frac{1}{2}(|0>\otimes(|f(0)>-|1-f(0)>)+|1>\otimes(|f(1)>-|1-f(1)>))$$

Ábra 49: Konstans orákulum számítás részletei

Ha f nem konstans, akkor

$$\frac{1}{2}(|0>\otimes(|1>-|0>)+|1>\otimes(|0>-|1>))=\frac{1}{2}((|1>-|0>)\otimes(|0>-|1>))$$

$$\frac{1}{2}(|0>\otimes(|0>-|1>)+|1>\otimes(|1>-|0>))=\frac{1}{2}((|0>-|1>)\otimes(|0>-|1>))$$

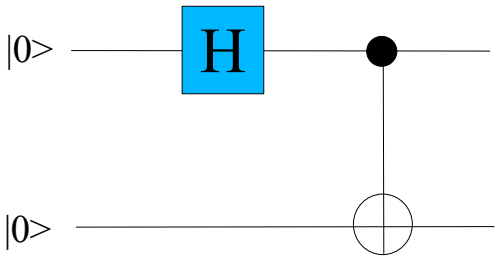
Ha f konstans, akkor

$$\frac{1}{2}(|0\rangle \otimes (|0\rangle - |1\rangle) + |1\rangle \otimes (|0\rangle - |1\rangle)) = \frac{1}{2}((|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle))$$

$$\frac{1}{2}(|0\rangle \otimes (|1\rangle - |0\rangle) + |1\rangle \otimes (|1\rangle - |0\rangle)) = \frac{1}{2}((-|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle))$$

XII.2.4.2 Sűrűségi kódolás

Közösen a Földön csinálunk egy EPR párt, majd Te a pár egyik qubitjével a zsebedben elutazol a Holdra.



$$\frac{1}{\sqrt{2}}(|00>+|11>)$$

Táblázat 12: EPR pár készítése

XII.2.4.2.1 Kódolás

A Földön I, X, Y vagy Z-t csinálok az qubitemmel:

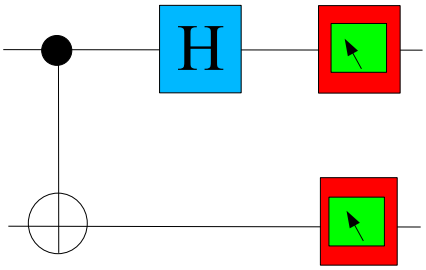
Mit akarok kódolni?	***Hogyan?***
00	I⊗I
01	X⊗I
10	Y⊗I
11	Z⊗I

$$\frac{1}{\sqrt{2}}(|00>+|11>)$$

Táblázat 13: Sűrűségi kódolás

XII.2.4.2.2 Dekódolás

Elküldöm az eredmény qubitet a Holdra, ahol Te az alábbi áramkörrel tudod dekódolni:



Ábra 50: Dekódolás a Holdon

XII.2.4.2.3 Feladat – mit mérünk a Holdon?

Mutassuk meg, hogy mit mérünk a Holdon ezzel az áramkörrel, ha a Földön például 01-et kódolunk?

XII.2.4.2.4 Kvantum teleportálás

Közösen a Földön csinálunk egy EPR párt $\frac{1}{\sqrt{2}}(|00>+|11>)$, majd Te a pár egyik qubitjével a zsebedben elutazol a Holdra. A Földön veszek egy tetszőleges $\alpha|0>+\beta|1>$ qubitet, ezt fogom teleportálni neked a Holdra.

A Földön CNOT, H-t csinálok az alábbiakban előkészített regisztertartalmon:

$$(\alpha|0>+\beta|1>)(\frac{1}{\sqrt{(2)}}(|00>+\beta|11>))=\frac{1}{\sqrt{(2)}}(\alpha|000>+\alpha|011>+\beta|100>+\beta|111>)$$

$$(CNOT\otimes I)(\frac{1}{\sqrt{(2)}}(\alpha|000>+\alpha|011>+\beta|100>+\beta|111>))=$$

$$\frac{1}{\sqrt{(2)}}(\alpha|000>+\alpha|011>+\beta|110>+\beta|101>)$$

$$(H\otimes I\otimes I)(\frac{1}{\sqrt{(2)}}(\alpha|000>+\alpha|011>+\beta|110>+\beta|101>))=$$

$$\frac{1}{2}(\alpha(|0>+|1>)|00>+\alpha(|0>+|1>)|11>+\beta(|0>-|1>)|10>+\beta(|0>-|1>)|01>)=$$

$$\frac{1}{2}(\alpha|000>+\alpha|100>+\alpha|011>+\alpha|111>+\beta|010>-\beta|110>+\beta|001>-\beta|101>)=$$

$$\frac{1}{2}(|00>(\alpha|0>+\beta|1>)+|01>(\alpha|1>+\beta|0>)+|10>(\alpha|0>-\beta|1>)+|11>(\alpha|1>+\beta|0>))$$

Megmérjük az első két bitet (ezt a tetszőleges és a pár földi része alkotja). Tegyük fel, hogy $|01>$ -et mérünk az első két qubitre, ezt az eredményt klasszikus 01-ként közöljük a Holddal, ahol Te a megfelelő (általában lásd az előző sűrűségi kódolás táblázatát, hogy mikor I, X, Y, Z) esetünkben az X leképezést alkalmazod a pár holdi részére, amivel így előállítod a korábbi földi tetszőleges $\alpha|0>+\beta|1>$ qubitet.

XII.2.4.3 Adatbázis kvantum keresése

A Grover algoritmus.

XII.2.5 Kvantum számítógépek

XII.3 Tudatmodellek

XII.3.1 A tudattal kapcsolatos interpretációk

Neumann,Wheeler, Wigner, Penrose, Everett, Deutsch.

XII.3.2 Kvantum tudat

Hameroff-Penrose: **Orch OR** (orchestrated objective reduction) model.

XII.4 Bioinformatika

Szilicium alapú eszközök, Moore törvény.

XII.4.1 Bioqubit?

Sejtváz, mikrotubulusok (MT), tubulin, MAP.
Biológiai kvantum teleportáció.

XII.4.1.1 Mikrotubulusok

XII.4.2 Fehérjék tekeredése

Elsődleges, másodlagos szerkezet.

XII.4.3 Sejtautomata

XIII. Appendixek

XIII.1 Mindenféle hasznos dolog, amit jó, ha tud egy felhasználó, pláne egy programozó

XIII.1.1 Az alapvető parancsok tipikus használata

XIII.1.1.1 Tömörítés

```
$ man gzip
```

Jó bevezetést [T, 46. oldal] és részletes leírást [R, 106. oldal] olvashatunk a Lempel-Ziv-Welch módszerről.

```
$ man bzip2
```

Parancsok:
`tar, gzip, bzip2`

Példák a használatra a következő pontban:

XIII.1.1.2 tar – archívumok készítése

Az `OS` könyvtár tartalmát betömörítjük az `OS.tgz` fájlba.

```
$ tar cvzf OS.tgz OS
```

Kitömörítése:

```
$ tar xvzf OS.tgz
```

Mi történik, ha `z` helyett `j`-t használunk?

```
$ man tar
```

XIII.1.1.3 ln

Nem akarjuk tárolni, hogy milyen parancsokat adtunk ki:

```
$ ln -s /dev/null .bash_history
[norbi@niobe ~]$ ls -l .bash_history
lrwxrwxrwx  1 norbi norbi 9 nov 12 11:43 .bash_history -> /dev/null
```

Bosszanthatja a rendszergazdát, de minket is, mert nem fog menni a jól megszokott „kurzor föl”.

XIII.1.1.4 netstat

Futtassuk például a *47*. oldal *Socketek* című pontjának Socketes lokális IPC-s szerverét, ekkor

```
$ netstat -xpa
.
.
.
unix  2      [ ACC ]     STREAM     LISTENING     80241
18965/unix_szerver  szerver.socket
.
.
.
```

vagy a *102*. oldal *Soros, azaz iteratív* című pontjának szerverét, ekkor:

```
$ netstat -tap
.
.
.
tcp        0      0 localhost.localdomain:2005  *:*
LISTEN      19438/soros_szerver
.
.
.
tcp        0      0 niobe.eurosmobil.hu:ftp
kalapacs.eurosmobil.hu:1044 ESTABLISHED -
tcp        0      0 niobe.eurosmobil.hu:ssh
kalapacs.eurosmobil.hu:1034 ESTABLISHED -
tcp        0     52 niobe.eurosmobil.hu:ssh
kalapacs.eurosmobil.hu:1041 ESTABLISHED -
.
.
.
```

Tegyünk be a szerver kiszolgáló függvényébe például egy 10 másodperces `sleep(10)` várakozást és egy másik ablakban folyamatosan tanulmányozzuk a program kimenetét:

```
$ netstat -tapc
.
.
.
tcp        0      0 localhost.localdomain:2005  *:*
LISTEN      3032/soros_szerver
```

Közben csatlakozzunk egy klienssel, például a szokásos `telnet` programmal, közben:

```
.
.
.
tcp        0      0 localhost.localdomain:56804
localhost.localdomain:2005  ESTABLISHED 3070/telnet
tcp        0      0 localhost.localdomain:2005
localhost.localdomain:56804 ESTABLISHED 3032/soros_szerver
.
.
.
```

XIII.1.1.5 mount

Hogyan telepíthetjük fel például a `lynx` nevű programot a Fedora Core 4 telepítő dvd lemezéről?

```
# mount /dev/hdc /media/cdrecorder/ -t iso9660
```

```
# rpm -ivh /media/cdrecorder/Fedora/RPMS/lynx-2.8.5-23.x86_64.rpm
warning: /media/cdrecorder/Fedora/RPMS/lynx-2.8.5-23.x86_64.rpm:
Header V3
DSA signature: NOKEY, key ID 4f2a6fd2
Preparing...                ########################################
[100%]
   1:lynx                    ########################################
[100%]
# umount /media/cdrecorder/
```

De akár egy iso fájlt is bemountolhatunk:

```
# mount Matyi.iso /mnt -t iso9660 -o loop=/dev/loop0
$ xine /mnt/video_ts/vts_01_1.vob
# umount /mnt
```

XIII.2 Fordítás

```
$ man gcc
```

XIII.2.1 C

GNU **C** **C**ompiler.

XIII.2.1.1 gcc

Lássuk, milyen a `gcc` fordítónk?

```
$ gcc –version
gcc (GCC) 4.0.0 20050519 (Red Hat 4.0.0-8)Copyright (C) 2005
Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.
$ gcc -v
Using built-in specs.
Target: x86_64-redhat-linux
Configured with: ../configure --prefix=/usr --
mandir=/usr/share/man --infodir=/usr/share/info --enable-shared
--enable-threads=posix --enable-checking=release --with-system-
zlib --enable-__cxa_atexit --disable-libunwind-exceptions --
enable-libgcj-multifile --enable-
languages=c,c++,objc,java,f95,ada --enable-java-awt=gtk --with-
java-home=/usr/lib/jvm/java-1.4.2-gcj-1.4.2.0/jre --
host=x86_64-redhat-linux
Thread model: posix
gcc version 4.0.0 20050519 (Red Hat 4.0.0-8)
```

XIII.2.1.1.1 Tipikus fordítások a jegyzetben

A jegyzet legtöbb példájánál megadjuk a fordítást végző parancssort, például az *102*. oldalon kezdődő program végén, a hálózati programozás témában:

```
$ gcc -lnsl -o szerver szerver.c
```

A `szerver.c` forrást fordítjuk, a fordítás kimenetét (**`-o`**) a `szerver` nevű fájlba helyezzük, a `-lnsl` linker opció azt mondja meg, hogy az `nsl` könyvtárat linkeljük a programunkhoz.

Az anyagban a következő `-llibrary` linker opciókkal találkozunk:

`-lm`	*34*., *82*. oldal
`-lnsl`	*103*.oldal
`-lpthread`	*64*.oldal
`-lcurses`	*89*. oldal
`-lgtkjava` `-lgnomejava`	*268*. oldal

Nézzük meg például a *116*.oldal, *Párhuzamos, POSIX szálakkal* című pont programjának fordítását:

```
$ gcc -lpthread -lnsl -o szerver szerver.c
```

Írassuk ki, hogy ekkor milyen könyvtárakat használ a `szerver`:

```
$ ldd szerver
        libpthread.so.0 => /lib64/libpthread.so.0
(0x0000003841e00000)
        libnsl.so.1 => /lib64/libnsl.so.1 (0x0000003849500000)
        libc.so.6 => /lib64/libc.so.6 (0x0000003841100000)
        /lib64/ld-linux-x86-64.so.2 (0x0000003840f00000)
```

XIII.2.1.1.2 Üzenetek

Fordítsuk a 109. oldal, *Párhuzamos, folyamatok sorával* című pontjának programját a `-Wall` opcióval, illetve anélkül:

```
$ gcc -lnsl -o szerver szerver.c
$ gcc -Wall -lnsl -o szerver szerver.c
szerver.c: In function 'main':
szerver.c:48: warning: unused variable 'kliens'
szerver.c:46: warning: unused variable 'kliensm'
```

XIII.2.1.1.3 Feltételes fordítás

```
#include <stdio.h>
#include <unistd.h>
#include <limits.h>
int
main (void)
{
#ifdef ARG_MAX
#ifdef DEBUG
  printf ("ARG_MAX definialt\n");
#endif
  printf ("ARG_MAX=%ld\n", ARG_MAX);
#else
#ifdef DEBUG
  printf ("ARG_MAX nem definialt\n");
#endif
  printf ("ARG_MAX=%ld\n", sysconf (_SC_ARG_MAX));
#endif
  return 0;
}
```

A példa jól mutatja a feltételes fordítás skatulyázhatóságát is.

```
$ gcc -o argmax argmax.c
$ ./argmax
ARG_MAX=131072
$ gcc -DDEBUG -o argmax argmax.c
$ ./argmax
ARG_MAX definialt
ARG_MAX=131072
```

XIII.2.1.1.4 Nomkövetés

XIII.2.1.2 make

XIII.2.2 Java

A következő pontokban a `gcj`-t és a Sun JDK csomagját mutatjuk be, mi tipikusan ez utóbbit használjuk.

XIII.2.2.1 gcj

Példaképpen a *130*. oldal *Java nyelven a szerveroldal* című pontjának `EgyszeruSzerver`-ét próbáljuk ki:

```
$ gcj -o szerver --main=EgyszeruSzerver EgyszeruSzerver.java
```

Nézzük meg a fordítás eredményét:

```
$ file szerver
szerver: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV),
for GNU/Linux 2.4.0, dynamically linked (uses shared libs), not
stripped
```

Futtasuk is a szervert:

```
$ ./szerver
```

majd közben teszteljük:

```
$ telnet localhost 2005
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Thu Nov 03 12:42:16 GMT+01:00 2005
Connection closed by foreign host.
```

Vagy készítsünk futtathatót a *201*. oldal *Java-GNOME* TCP kliensének felületéből:

```
$ gcj --classpath
/usr/share/java/gtk2.6.jar:/usr/share/java/gnome2.10.jar
-lgtkjava-2.6 -lgnomejava-2.10 --main=ProgPaterKliens -o
prog_pater_kliens  ProgPaterKliens.java
```

XIII.2.2.2 A Java fejlesztői csomag telepítése

Azaz a Java™ 2 Platform Standard Edition Development Kit telepítése. Tipikusan ezt szoktuk használni, ha Java-ban dolgozunk, ezért kicsit részletesebben írunk róla: kitérve a telepítésre is: a Java fejlesztői csomagot a [*Java*/SE] címről tölthetjük le.

Mi mondjuk most éppen a *„JDK 5.0 Update 5”*-t választottuk, azon belül is a *„Linux AMD64 Platform - J2SE(TM) Development Kit 5.0 Update 5”*-öt, mert egy Opteron procis gépre lesz. (Aki szereti az integrált fejlesztői környezeteket, az választhatja a *„NetBeans IDE + JDK 5.0 Update 5”* linket is.) Egyszerű felhasználóként így telepíthetjük:

```
$ chmod +x jdk-1_5_0_05-linux-amd64.bin
$ ./jdk-1_5_0_05-linux-amd64.bin
```

```
.
.
$ export PATH=$HOME/Java/jdk1.5.0_05/bin:$PATH
$ java -version
java version "1.5.0_05"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_05-
b05)
Java HotSpot(TM) 64-Bit Server VM (build 1.5.0_05-b05, mixed mode)
```

Innentől megy a legújabb `java`, `javac`. Persze a `PATH`-ot nem érdemes mindig kézzel állítgatni, ezt a sort egyszerűen tegyük be a `~/.bash_profile` fájlba, ennek kapcsán lásd a *24*. oldal *Az elérési út beállítása* című pontot.

Az előző ponthoz hasonlóan avassuk is fel a frissen feltett JDK-t! A *130*. oldal *Java nyelven a szerveroldal* című pontjának `EgyszeruSzerver`-ét próbáljuk ki vele:

Fordítunk:

```
$ javac EgyszeruSzerver.java
```

Nézzük meg a fordítás eredményét:

```
$ file EgyszeruSzerver.class
EgyszeruSzerver.class: compiled Java class data, version 49.0
```

Futtasuk a szervert:

```
$ java EgyszeruSzerver
```

Teszteljük:

```
$ telnet localhost 2005
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Thu Nov 03 19:53:48 CET 2005
Connection closed by foreign host.
```

XIII.3 Mindenféle hasznos dolog, amit jó, ha tud egy programozó, pláne egy rendazergazda

XIII.3.1 SysV init

Módosítsuk az Apache konfig fájlját, hogy tudjunk CGI programokat használni:

```
ScriptAlias /norbi-cgi/ "/home/norbi/public_html/cgi-bin/"

<Directory "/home/norbi/public_html/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

Indítsuk újra a webszervert:

```
# /etc/rc.d/init.d/httpd restart
httpd leállítása:                          [  OK  ]
A(z) httpd indítása:                       [  OK  ]
```

Ekkor a „Helló, világ!” alábbi, kis módosításával már futni is fog a „Helló, világ!” CGI változata:

```
#include <stdio.h>
int
main(void)
{
  printf ("Content-type: text/plain\n\n");
  printf("Hello, Vilag!\n");
  return 0;
}
$ gcc -o hello hello.c
$ cp hello public_html/cgi-bin
```

A `http://niobe.eurosmobil.hu/norbi-cgi/hello` címen.

Témánkhoz visszatérve, a SysV init szkripteknél tipikusan a `start`, `stop`, `status`, `restart` paraméterek használhatóak.

XIII.3.1.1 chkconfig

Ha nemcsak „most” akarjuk futtatni a szervert, hanem – ellentétben az előző pontban – mindig, azaz például minden induláskor, akkor:

```
# chkconfig httpd on
```

a

```
# chkconfig --list httpd
httpd           0:ki    1:ki    2:be    3:be    4:be    5:be    6:ki
```

megmutatja, hogy a webszerver mely futási szinteken fog indulni. (Persze grafikus felületen is kattintgathatunk a témában, ha futtatjuk a system-config-services programot...)

XIII.4 Bash programozás

```
$ more vissza
#!/bin/bash
exit 5
$ ./vissza
$ echo $?
5
```

Értékadás alapértelmezéssel: `vált1=${vált2-alapértelmezett érték}`

```
$ more kiir
#!/bin/bash
p=${1-alma}                          # Ha van $1, akkor p=$1
                                     # különben legyen p=alma
echo "A p értéke: $p"
echo "Hogy hivatkozok a p értékére:" '$p' # hogy írom ki
a $-t?
exit 0
$ ./kiir
A p értéke: alma
Hogy hivatkozok a p értékére: $p
```

XIII.4.1 `getenv()`

Bemenet a környezetből:

```
$ more korny
#!/bin/bash
#előtte: export VALTOZO=ertek
echo $VALTOZO
exit 0
$ ./korny

$ export VALTOZO=ertek
$ ./korny
ertek
$ more valtozo.c
#include <stdio.h>
int main(int argc, char *argv[])
{
  printf("%s\n", getenv("VALTOZO"));
  return 0;
}
$ gcc -o v valtozo.c
$ ./v
ertek
```

XIII.4.2 `char **environ`

A szokatlan harmadik paraméter:

```
$ more kornyezet.c
```

```
#include <stdio.h>
int main(int argc, char *argv[], char *korny[])
{
 int i=0;
 while (korny[i] != NULL)
   printf("%s\n", korny[i++]);
 return 0;
}
$ gcc -o k kornyezet.c
$ ./k
TERM=vt100
SHELL=/bin/bash
...
```

Mi a `korny`? Nézzünk utána:

```
man environ
```

XIII.4.3 A burok változói

```
$ more burok1
#!/bin/bash
echo "$0"
echo "$1"
echo "$2"
echo "$3"
echo "$#"
echo "$*"
echo "$@"
exit 0
$ ./burok1 alma korte banan
./burok1
alma
korte
banan
3
alma korte banan
alma korte banan
```

Bemenet a parancssorból:

```
$ more burok2
#!/bin/bash
for i;
do
        echo "$i"
done
echo
for i in $*;
do
        echo "$i"
done
echo
for i in $@;
do
        echo "$i"
```

```
done
exit 0
$ ./burok2 alma korte banan
alma
korte
banan

alma
korte
banan

alma
korte
banan
```

Mi a különbség a `$*` és a `$@` között:

```
$ more burok3
#!/bin/bash
s=0
for i in "$*"
do
        echo "$i"
        let s+=1
done
echo $s
s=0
for i in "$@";
do
        echo "$i"
        let s+=1
done
echo $s
exit 0
$ ./burok3 alma korte banan
alma korte banan
1
alma
korte
banan
3
```

Bemenet a:

```
$ more beolv
#!/bin/bash

echo "írj be egy szót:"
read p
echo "A p ertéke: $p"
exit 0
$ ./beolv
írj be egy szót:
alma
A p ertéke: alma
```

Gyakorlásképpen az alábbi ciklusok felhasználásával írjunk variánsokat egy olyan kis szkriptre, ami parancssorból átvett számút `Ctrl+g`-zik, illetve ha nincs megadva a `$1`, akkor alapértelmezésben hármat! (vi-ban a `Ctrl+g`-t `Ctrl+v Ctrl+g` formában tudjuk bevinni: `echo "^G"`)

```
for i in `seq 1 $p`;
```

```
for ((i=0;i<$p;++i))
```

```
while test $p -gt 0
```

Fájlok átnevezése:

```
$ more html
#!/bin/bash
for i in *.htm
do
        echo "$i -> ${i}l"
        mv $i ${i}l
done
$ ls *.htm
a.htm  b.htm  c.htm
$ ./html
a.htm -> a.html
b.htm -> b.html
c.htm -> c.html
$ ls *.htm
ls: *.htm: No such file or directory
$ ls *.html
a.html  b.html  c.html
```

XIII.4.4 sed

```
$ ls *.txt
a.txt  b.txt  c.txt
$ more a.txt
alma korte banan korte
alma korte banan korte
$ more korte2dio
#!/bin/bash
for i in *.txt
do
        echo "$i -> ${i}.cserelve"
        sed 's/korte/dio/g' $i > ${i}.cserelve
done
$ ./korte2dio
a.txt -> a.txt.cserelve
b.txt -> b.txt.cserelve
c.txt -> c.txt.cserelve
$ more a.txt.cserelve
alma dio banan dio
alma dio banan dio
```

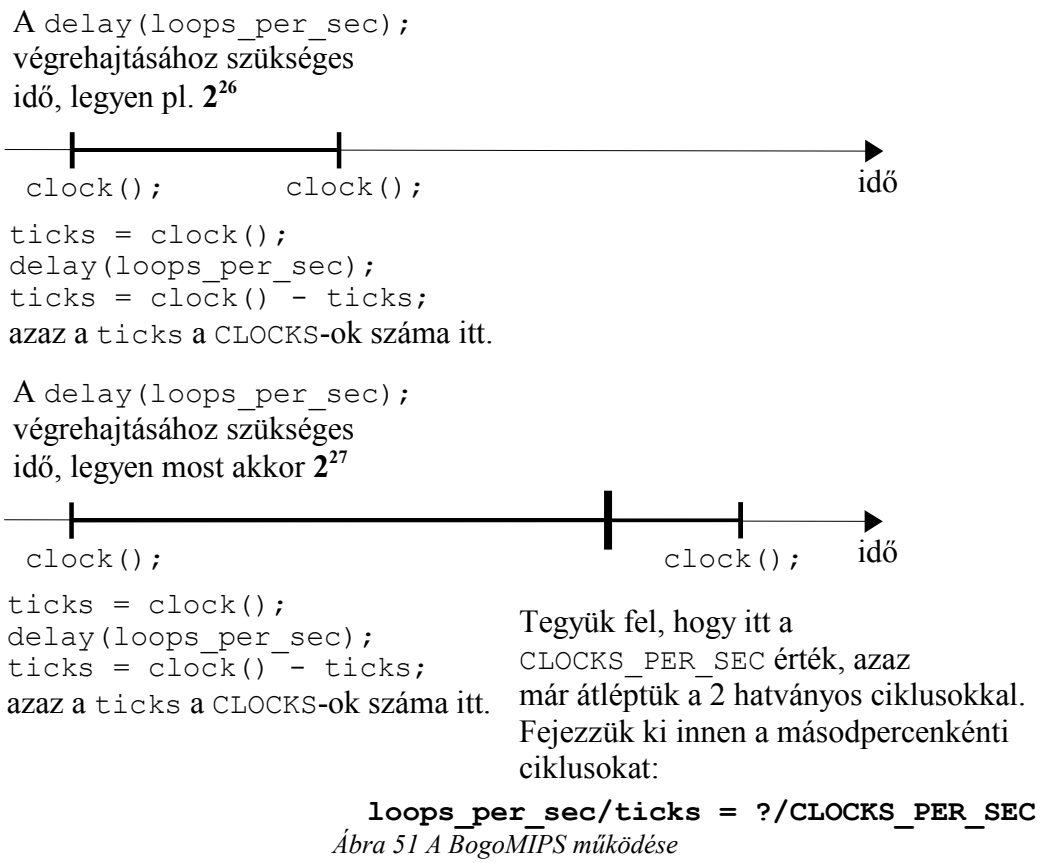
Parancsok:
`find, grep, sed`

XIII.5 További érdekességek

XIII.5.1 BogoMIPS

A Jeff Tranter által (Linus Torvalds Linux kernel-beli kódja alapján) írt standalone BogoMIPS izgalmas mérésekre ad lehetőséget.
Letöltés:
http://packages.debian.org/stable/utils/sysutils.html

A működés vázlata:



Ábra 51 A BogoMIPS működése

XIII.5.2 Számábrázolás

Miért digitális most az informatika?

XIII.5.2.1 Számrendszerek

XIII.5.2.1.1 komplex alap

XIII.5.3 LEGO® programozás

XIII.5.3.1 leJOS

XIV. GYIK – FAQ

XIV.1 Java

Linux alatt nem fordulnak le a jegyzet Java forrásai, a karekterkódolás miatt. Fordítsuk például így:

```
javac -encoding ISO8859_2 SwingKliens.java
```

XV. Gondolkodtató, programoztató kérdések

XV.1 Általános kérdések: C programozás, manuál lapok olvasása stb.

1. Mire utal a `const` a paraméterátadásnál, például az alábbi manuál lapon?

```
$ man 3 strcpy
```

   - Konstans sztringet kell átadnunk!
   - Nem változik meg az `strcpy()` működése során.

XV.2 Folyamatokkal kapcsolatos kérdések

1. Mit nyomtat ki az alábbi program?

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
int kulso = 0;
static int statikus_kulso = 0;
void
valtozok (void)
{
  int belso = 0;
  static int statikus_belso = 0;
  int gyermekem_pid;
  int statusz;
  if ((gyermekem_pid = fork ()) == 0)
    {
      printf ("GY: %d %d %d %d\n", ++kulso, ++statikus_kulso,
              ++belso, ++statikus_belso);
      exit (0);
    }
  else if (gyermekem_pid > 0)
    {
      wait (&statusz);
    }
  else
    {
      exit (-1);
    }
  printf ("SZ: %d %d %d %d\n", kulso, statikus_kulso,
          belso, statikus_belso);
}
int
main (void)
{
  valtozok ();
  valtozok ();
  return 0;
}
```

```
GY: 1 1 1 1
SZ: 0 0 0 0
GY: 1 1 1 1
SZ: 0 0 0 0

GY: 1 1 1 1
SZ: 1 1 0 0
GY: 1 1 1 1
SZ: 1 1 0 0

GY: 1 1 1 1
```

```
SZ: 1 1 0 0
GY: 2 2 1 1
SZ: 2 2 0 0
```

Tipp: próbáljuk ki!

2. Mit nyomtat ki az előző program alábbi módosítása?

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
int kulso = 0;
static int statikus_kulso = 0;
int masik_kulso = 0;
int *masik_kulso_p = &masik_kulso;
void
valtozok (void)
{
  int belso = 0;
  static int statikus_belso = 0;
  int gyermekem_pid;
  int statusz;
  if ((gyermekem_pid = fork ()) == 0)
    {
      printf ("GY: %d %d %d %d %d\n", ++kulso, ++statikus_kulso,
              ++belso, ++statikus_belso, ++*masik_kulso_p);
      exit (0);
    }
  else if (gyermekem_pid > 0)
    {
      wait (&statusz);
    }
  else
    {
      exit (-1);
    }
  printf ("SZ: %d %d %d %d %d\n", kulso, statikus_kulso,
          belso, statikus_belso, *masik_kulso_p);
}
int
main (void)
{
  valtozok ();
  valtozok ();
  return 0;
}
```

```
GY: 1 1 1 1 1
SZ: 0 0 0 0 0
GY: 1 1 1 1 1
SZ: 0 0 0 0 0
```

```
■  GY: 1 1 1 1 1
   SZ: 1 1 0 0 1
   GY: 1 1 1 1 1
   SZ: 1 1 0 0 1

■  GY: 1 1 1 1 1
   SZ: 1 1 0 0 1
   GY: 2 2 1 1 1
   SZ: 2 2 0 0 1
```

Tipp: próbáljuk ki!

3. Feltámadhat-e egy zombi folyamat?
   - Igen, csak a megfelelő jelzőket kell visszaállítani.
   - Nem, mert nincs miket visszaállítani.

Tipp: Készítsünk egy zombi folyamatot (a 108. oldal *Kérdések – a párhuzamos szerverről* című pontjában találunk egy „zombi szervert") és nézzünk szét a `/proc/[megfelelő_PID]` fájlokban!

4. ?
   - Igen, .
   - Nem, .

Tipp:

XV.3 IPC-vel kapcsolatos kérdések

1. Mennyi a felcsatolások száma a – a 59. oldal, *Osztott memória* című pontja programjának kis módosításával készített – alábbi program futásának első 10 másodpercében?

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#define IDO_MERET 48
void
ebreszto ()
{
  printf ("Ebreszto, Neo!\n");
}
int
main ()
{
  int i, gyermekem_pid;
  int osztott_memoria;
  char *osztott_memoria_terulet;
  if ((osztott_memoria =
       shmget (ftok (".", 44), 64, IPC_CREAT | S_IRUSR | S_IWUSR))
        == -1)
    {
      perror ("shmget");
      exit (EXIT_FAILURE);
    }
  if ((osztott_memoria_terulet = shmat (osztott_memoria, NULL, 0))
       < 0)
    {
      perror ("shmat");
      exit (EXIT_FAILURE);
    }
  for (i = 0; i < 3; ++i)
    if ((gyermekem_pid = fork ()) == 0)
      {
     char buffer[IDO_MERET];
     time_t t = time (NULL);
     char *p = ctime_r (&t, buffer);
     strncpy (osztott_memoria_terulet, p, IDO_MERET);
     sleep (10);
     kill (getppid (), SIGALRM);
     if (shmdt (osztott_memoria_terulet) == -1)
       {
         perror ("shmdt");
         exit (EXIT_FAILURE);
       }
```

```
     exit (EXIT_SUCCESS);
       }
     else if (gyermekem_pid > 0)
       {
      signal (SIGALRM, ebreszto);
       }
     else
       {
      exit (-1);
       }
  pause ();
  printf ("%s", osztott_memoria_terulet);
  if (shmdt (osztott_memoria_terulet) == -1)
    {
      perror ("shmdt");
      exit (EXIT_FAILURE);
    }

  if (shmctl (osztott_memoria, IPC_RMID, NULL))
    {
      perror ("shmctl");
      exit (EXIT_FAILURE);
    }
  return 0;
}
```

- Egy, hiszen csupán egyetlen `shmat()` felcsatolás van..
- Három, mert a szülő átadta a a gyerekeknek a felcsatolást.
- Négy, mert a gyerekek is megkapták a felcsatolást.

Tipp: próbáljuk ki!

XV.4 Linux kernel

1. Ki a kernel szálak szülője?
    - A PID=0 folyamat.
    - A PID=1 folyamat, az init.

Tipp: írassuk ki!

XVI. Válaszok a programoztató kérdésekre

XVI.1 Általános kérdések: C programozás, manuál lapok olvasása stb.

XVI.2 Folyamatok

1. Mit nyomtat ki az alábbi program?

```
✔ GY: 1 1 1 1
  SZ: 0 0 0 0
  GY: 1 1 1 1
  SZ: 0 0 0 0
```

2. Mit nyomtat ki az alábbi program?

```
✔ GY: 1 1 1 1 1
  SZ: 0 0 0 0 0
  GY: 1 1 1 1 1
  SZ: 0 0 0 0 0
```

3. Feltámadhat-e egy zombi folyamat?
   - ✗ Igen, csak a megfelelő jelzőket kell visszaállítani.
   - ✔ Nem, mert nincs miket visszaállítani.

Tipp: például a `/proc/[megfelelő_PID]/stat`, `/proc/[megfelelő_PID]/statm` fájlokat érdemes megnézni. A kapcsolódó mezők leírását a

```
$ man proc
```

lapon, vagy a kernel források [*OS*/KO] között az `fs/proc/array.c` forrásban találjuk.

XVI.3 IPC

1. Mennyi a felcsatolások száma a program futásának első 10 másodpercében?

```
$ ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch
status
0x2c05582b 2457606    norbi     600        64         4

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
```

   - ✗ Egy, hiszen csupán egyetlen `shmat()` felcsatolás van..
   - ✗ Három, mert a szülő átadta a a gyerekeknek a felcsatolást.
   - ✔ Négy, mert a gyerekek is megkapták a felcsatolást.

XVI.4 Linux kernel

1. Ki a kernel szálak szülője?
   - ✗ A PID=0 folyamat.
   - ✔ A PID=1 folyamat, az init.

XVII. Az anyag fejlődése

0.0.1, 2005-03-28, kezdeti dokumentum, bash programozás elkezd.
0.0.2, 2005-04-04, C rendszerszint elkezd.
0.0.3, 2005-04-14, Hálózati programozás elkezd.
0.0.4, 2005-04-18, Mobil programozás elkezd.
0.0.5, 2005-04-18, LEGO® programozás elkezd.
0.0.8, 2005-05-08, Java, C++, C# példákkal bővítés elkezd.
0.0.9-0.0.15, javítások, finomítások.
0.0.16, 2005-05-12, ebédelő filoszok C-ben elkezd.
0.0.17, 2005-05-13, Programok és platformokban az oldalszámok berakása elkezd.
0.0.18, 2005-05-13, javítások, finomítások, socket szerverek kiegészítés.
0.0.19, 2005-05-15, ebédelő filoszok C folyt.
0.0.20, 2005-05-16, javítások, finomítások.
0.0.21, 2005-05-18, ebédelő filoszok C kieg., ábrajegyzék.
0.0.22, 2005-05-24, ebédelő filoszok és egyszerű TCP kliens Java, C#.
0.0.23, 2005-05-25, Feldolgozási javaslatok, raw socketek, ping elkezd.
0.0.24, 2005-05-25, magyarázó ábrák készít.
0.0.25, 2005-05-27, a Bátfai Erika (a feleségem, bme@javacska.hu) javasolta új cím: **Programozó Páternoszter**, javítások, finomítások.
0.0.26, 2005-05-28, Hivatkozott és ajánlott irodalom frissít.
0.0.27, 2005-05-28, javítások, finomítások.
0.0.28, 2005-06-17, Kvantum számítások elkezd.
0.0.29, 2005-06-20, javítások, finomítások.
0.0.30, 2005-06-21, kvantum teleportáció.
0.0.31-0.0.43, 2005-08-12, irodalom felkutatása, feldolgozása (biológiai vonatkozások, pl. kvantumos tudat modellek stb.)
0.0.44, 2005-08-14, javítások, finomítások.
0.0.45, 2005-08-15, Bioinformatika elkezd.
0.0.46, 2005-08-21, Mikrotubulusok elkezd.
0.0.47, 2005-09-15, Szerver és kliensoldali socket programozás kieg.
0.0.48, 2005-09-25, javítások, finomítások.
0.0.48-0.0.55, 2005-10-18, javítások, finomítások.
0.0.56, 2005-10-19, átszervezés.
0.0.57, 2005-10-20, kölcsönös kizárás az accept()-ekre, kernel modulok elkezd.
0.0.58, 2005-10-21, kölcsönös kizárás az accept()-ekre a folyamatoknál, C kliens.
0.0.59, 2005-10-22, kernel modulok folyt., hálózati rész kieg.
0.0.60, 2005-10-23, Proc fájlrendszer elkezd.
0.0.61, 2005-10-24, javítások, finomítások.
0.0.62, 2005-10-25, javítások, Proc fájlrendszer folytat, a seq_file interfész elkezd.
0.0.63, 2005-10-28, javítások, finomítások, socket szerverek kiegészítése.
0.0.64-0.0.76, 2005-10-31, javítások, finomítások, kiegészítések, színek, az anyag átszervezésének megkezdése és **Belépés a gépek mesés birodalmába** mottó.
0.0.77, 2005-10-31, lokális socket IPC.
0.0.78-0.0.79, 2005-11-1, elgondolkodtató, programoztató kérdések-válaszok, anonim socket IPC.
0.0.80, 2005-11-1, fordítás: gcc.
0.0.81-0.0.83, 2005-11-2, javítások, finomítások, csővezetékek. Hálózati résznél a 2.4, 2.6 szálkezelést összehasonlító rész javítása, kernel blog :)

0.0.84-0.0.86, 2005-11-3, kereszthivatkozások javítása, make elkezdése, gcj, JDK, zombi ábra.
0.0.87-0.0.89, 2005-11-4, javítások, finomítások, Gép melletti használatra! Még több csővezeték.
0.0.90, 2005-11-5, C kernelszint javítás, üzenetsorok elkezd, osztott memória elkezd.
0.0.91, 2005-11-6, IPC kérdések elkezd.
0.0.92, 2005-11-7, javítások, finomítások.
0.0.93, 2005-11-8, GUI elkezd.
0.0.94, 2005-11-12, jegyzékek a végére, üzenetsorok kieg., GUI folyt., AWT, Swing TCP kliens elkezd.
0.0.95, 2005-11-12, javítások, finomítások.
0.0.96-0.0.114, 2005-11-13, , javítások, finomítások, üzenetsorok ábra, GUI kliensek folyt., fs.
0.0.115, 2005-11-14, javítások, finomítások.
0.0.116, 2005-11-14, shm ábra, feladat.
0.0.117, 2005-11-14, javítások, finomítások.
0.0.118, 2005-11-18, rendszerhívások, norbi().
0.0.119, 2005-11-19, multiplexelt Java szerver.
0.0.120, 2005-11-20, egyszerű C# szerver, riasztások javítása, parancsok bővít, Helló átirányítás javítása :)
0.0.121-0.0.122, 2005-11-21, SMTP feladat: C, Java, kieg.
0.0.123, 2005-11-22, netstat kieg.
0.0.124-0.0.126, 2005-11-24, GTK+ folyt., javítások, finomítások, CORBA elkezd.
0.0.127, 2005-11-25, javítások, finomítások.
0.0.128-0.0.140, 2005-11-25, 26, Java RMI elkezd., Java-GNOME ProgPáter kliens felület, gcj kieg. IPv6 kieg. elkezd.
0.0.141-0.0.145, 2005-11-27, tartalmi könyvjelzők stb., RMI folyt., UDP elkezd.
0.0.146-0.0.147, 2005-11-28, UDP folyt.
0.0.148, 2005-11-29, javítások, finomítások, tech. Könyvjelzők, UDP folyt.
0.0.149-0.0.152, 2005-11-30, GNU FDL, UDP folyt., IPC kieg.
0.0.153-0.0.163, 2005-12-3, , javítások, finomítások, borítólap, távoli metódushívás ábrák.
0.0.164-0.0.167, 2005-12-4, , javítások, finomítások. Borítólap, Swing folyt., feldolg. jav. kieg.
0.0.168, 2005-12-5, UDP kieg., Swing folyt.
0.0.169-0.0.172, 2005-12-6, javítások, finomítások, GUI átszerv., Swing folyt.
0.0.173-0.0.175, 2005-12-7-10, javítások, finomítások.
0.0.176-0.0.187, 2005-12-11, HTTP protokoll, Szervletek, MySQL.
0.0.188, 2005-12-12, áttekintő ábrák elkezd.
0.0.189-0.0.197, 2005-12-14, bevezetés átszerv., kieg. PostgreSQL folyt.
0.0.198-0.0.215, 2005-12-15, javítások, finomítások, teljes képernyő, $, #, IPv6 kieg. Web prog. kieg.
0.0.216-0.0.218, 2006-02-08, Gantt diagramm az oprendszerek 1. szervezési részhez és kieg., tanári megjegyzések kékkel az elejéhez.
0.0.219, 2006-02-09, javítások.
0.0.220, 2006-04-15, a millió programozó országa.
0.0.221, 2006-08-25, teljes képernyő javít.
0.0.222, 2006-09-18, rendszer statisztika példa a fájlkezeléshez.
0.0.223, 2006-09-23, rendszer statisztika példa ábra, Java ME példák elkezd.
0.0.225, 2006-09-24, a Java ME alkalmazások életciklusa, általános jelölések.
0.0.226, 2006-10-02, LOKI link, Mobil programozás, J2SE hálózatok laborok tematikái.
0.0.227, 2006-10-04, a rendszer statisztika példa köré épülő feladatok.
0.0.228, 2006-10-07, TCP/IP csevegő Java-ban és C-ben példa elkezd.

0.0.229, 2006-10-08, TCP/IP multiplexelt, nem blokkolódó csevegő Java-ban.
0.0.232, 2006-10-14, Java ME, pidcs folytat.
0.0.234, 2006-10-29, Java ME szálak, GameCanvas, BBP algoritmus és párhuzamos, illetve elosztott számítások elkezd.
0.0.236, 2006-10-30, Java ME folyt., párhuzamos, illetve elosztott számítások folyt.
0.0.238, 2006-11-02, Pi jegyei több folyamattal.
0.0.241, 2006-11-11, Pi párhuzamos számítások folyt.
0.0.247, 2006-11-12, Java ME folyt.: sprite-ok, Pi elosztott számítások folyt.

XVIII. Felhasznált és ajánlott irodalom

Témák szerint csoportosítunk.

XVIII.1 OS

[FG] Fazekas Gábor: *Operációs rendszerek.* http://www.inf.unideb.hu/~fazekasg/oktatas/sajat/index1.html (az előadás anyaga)

[RH] Richard Petersen, Ibrahim Haddad: *RedHat Enterprise Linux és Fedora*, Panem 2005.

[KO] Kernel.org: *Kernel források*, http://www.kernel.org (2.6.13.4, 2.6.14)

[CRL] Cross-Referencing Linux, http://lxr.linux.no

[OS] Andrew S. Tanenbaum, Albert S. Woodhull: *Operációs rendszerek*. Panem 1999.

[LKMPG] Peter Jay Salmaz: *The Linux Kernel Module Programming Guide*, Linux Documentation Project.

Driver porting: *The seq_file interface*, http://lwn.net/Articles/22355

Steven Goldt et al.: *The Linux Programmer's Guide,* http://www.ibiblio.org/pub/Linux/docs/linux-doc-project/programmers-guide/lpg-0.4.pdf

Bányász Gábor, Levendovszky Tihamér: *Linux programozás*. Szak kiadó, 2003.

[LK] kernel-doc-2.6.11/Documentation/kbuild

[JR] John the Ripper: http://directory.fsf.org/security/Encryption/JohntheRipper.html

Jean Bacon, Tim Harris: *Operating Systems, Concurrent and Distributed Software Design*. Addison-Wesley 2003.

M. Beck et. al.: *Linux Kernel Internals*, Addison-Wesley 1998.

[NX] Brian W. Kernighan, Rob Pike: *A UNIX operációs rendszer.* Műszaki Könyvkiadó, 1987.

[T] Éric Lévénez-féle idővonalak: http://www.levenez.com/unix, http://www.levenez.com/windows

Stuart Sechrest: *An Introductory 4.4BSD Interprocedd Communication Tutorial*, http://docs.freebsd.org

Samuel J. Leffler et al.: *An Advanced 4.4BSD Interprocedd Communication Tutorial*, http://docs.freebsd.org

The Quintessential Linux Benchmark: http://www.linuxjournal.com/article/1120 (2005)

Christian Poellabauer: Process Management:
http://www.cse.nd.edu/~cpoellab/teaching/cse354/sp17.pdf

Introduction To Unix Signals Programming:
http://mia.ece.uic.edu/~papers/WWW/signals/signals-programming.html (2005)

Lusheng Wang's Home Page, CS3271 fork program:
http://www.cs.cityu.edu.hk/~lwang/fork (2005)

Randal E. Bryant and David R. O'Hallaron: *Computer Systems: A Programmer's Perspective*
http://csapp.cs.cmu.edu/
http://csapp.cs.cmu.edu/public/ch13-preview.pdf

Mark Hays: POSIX Threads Tutorial:
http://math.arizona.edu/~swig/documentation/pthreads

Alex Vernios: *Linux Cluster Architecture*, SAMS, 2002.

XVIII.2 C, C++

[KR] Brian W. Kernighan, Dennis M. Ritchie: *A C programozási nyelv.* Műszaki Könyvkiadó, 1993.

Bjarne Stroustrup: *A C++ programozási nyelv.* Kiskapu, 2001.

The GNU C Library: http://www.gnu.org/software/libc/manual/html_node (2005)

[TU] GNU Text Utilities:http://www.gnu.org/software/textutils/textutils.html

XVIII.3 Java

Java Technology: http://java.sun.com

[SE] J2SE letöltése: http://java.sun.com/j2se/1.5.0/download.jsp

[NB] netBeans IDE 5.5, http://www.netbeans.org

JDK™ 5.0 Documentation, http://java.sun.com/j2se/1.5.0/download.jsp#docs

Thomas Christopher, George K. Thiruvathukal: *High-Performance Java Platform Computing: Multithreaded and Networked Programming.*

David Reilly, Michael Reilly: *Java Network Programming and Distributed Computing.* Addison-Wesley, 2002.

Andrew Davison: *Killer game Programming in Java*, O'Reilly, 2005.

[WS] A Simple Multithreaded Web Server:
http://java.sun.com/developer/technicalArticles/Networking/Webserver

Jeffrey Morgan: *Java-GNOME 2.10 GNOME Tutorial*, http://java-gnome.sourceforge.net

[SW1] Hans Muller, Kathy Walrath: *Threads and Swing,*
http://java.sun.com/products/jfc/tsc/articles/threads/threads1.html

[SW2] *Using a Swing Worker Thread,*
http://java.sun.com/products/jfc/tsc/articles/threads/threads2.html

[SW3] Joseph Bowbeer: *The last Word in Swing Threads,*
http://java.sun.com/products/jfc/tsc/articles/threads/threads3.html

[TC] *Apache Tomcat:* http://tomcat.apache.org

XVIII.4 C#

Illés Zoltán: *Programozás C# nyelven.* Jedlik Oktatási Stúdió, 2004.

[MS] Microsoft .NET Framework SDK: http://msdn.microsoft.com/netframework

Joseph Mayo: *C# Unleased*, SAMS, 2002.

XVIII.5 GUI

[GTK] Tony gale, Ian Main & the GTK team: *GTK+ 2.0 Tutorial*: http://www.gtk.org/tutorial

GTK+ Reference manual

XVIII.6 Mobil

Java 2 Platform, Micro Edition: http://java.sun.com/j2me

[NB MOBILITY] NetBeans IDE 5.5, http://www.netbeans.org és ugyanitt Mobility Pack.

[WT] J2ME Wireless Toolkit 2.5: http://java.sun.com/products/j2mewtoolkit

The CLDC HotSpot™ Implementation Virtual Machine:
http://java.sun.com/products/cldc/wp/CLDC_HI_WhitePaper.pdf

Richard Harrison: *Symbian OS C++ for Mobile Phones.* Wiley, 2003.

Jo Stichbury: *Symbian OS Explained.* Wiley, 2005.

Charaf Hassan, Csúcs Gergely, Forstner Bertalan, Marossy Kálmán: *Symbian alapú szoftverfejlesztés*. Szak kiadó, 2004.

Martin de Jode: Programming *Java 2 Micro Edition on Symbian OS*. John Wiley & Sons, 2004.

Symbian OS – the mobile operating system: http://www.symbian.com

[FN] Forum Nokia – Developer Resources: http://www.forum.nokia.com

XVIII.7 Hálózati

[UNP] W. Richard Stevens: *UNIX Network Programming.* Prentice Hall PTR, 1998.

Diego Sevilla Ruiz, Mathieu Lacage, Dirk-Jan C. Binnema: *GNOME & CORBA*, http://developer.gnome.org

C Language Mapping Specification, http://www.omg.org

[TB] Andrew S. Tanenbaum: *Számítógéphálózatok*. Panem 2004.

[RFC2133] Robert E. Gilligan et al.: *RFC 2133: Basic Socket Interface Extension for Ipv6*, http://www.ietf.org/rfc

[RFC2553] RFC 2553: Basic Socket Interface Extension for Ipv6, http://www.ietf.org/rfc

[RFC2616] RFC 2616: Hypertext Transfer Protocol - HTTP/1.1, http://www.ietf.org/rfc

XVIII.8 Adatbázis

[MY] *MySQL*: http://www.mysql.com

[PG] *PostgreSQL*: http://www.postgresql.org

XVIII.9 Egyéb: elméleti, matekos stb.

[IK] Debreceni Egyetem Informatikai Kar, http://www.inf.unideb.hu

[EM] EUROSMOBIL Játék- és Alkalmazásfejlesztő Bt., http://www.eurosmobil.hu

[DR] A Diák-Robot Barátság lapok: *http://www.javacska.hu*

[JV] A Jávácska Vortál lapjai: *http://javacska.lib.unideb.hu*

[KN] D. E. Knuth: *A számítógép programozás művészete 2.* Műszaki, 1987.

GNU make: http://www.gnu.org/software/make/manual

[VG] Deák István: *Véletlenszám generátorok és alkalmazásaik.* Akadémiai Kiadó, 1986.

Robert W. Sebesta: *Concepts of Programming Languages*. Addison Wesley, 2003.

[R] Rónyai Lajos et al.: Algoritmusok. Typotex, 1998.

[T] Tusnády Gábor: Sztochasztikus számítástechnika. KLTE, 1996.

BARNES, David J.: *Teaching introductory Java through LEGO MINDSTORMS models*. ACM SIGCSE Bulletin , Proceedings of the 33rd SIGCSE technical symposium on Computer science education. February 2002. Volume 34 Issue 1

[PI] David H. Bailey, Peter B. Borwein, Simon Plouffe: *On The Rapid Computation of Various Polylogarithmic Constants*. http://citeseer.ist.psu.edu/bailey96rapid.html

[BBP ALG] David H. Bailey: *The BBP Algorithm for Pi.* http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-alg.pdf

[SETI] Seti@HOME: http://setiathome.berkeley.edu

[K] Carl Sagan: Kapcsolat.

XVIII.10 Kvantum

[DD] David Deutsch: *Quantum theory, the Church-Turing principle and the universal quantum computer.* Proceedings of the Royal Society of London A 400, pp. 97-117., (1985).

[DS] O. Carnal, J. Mlynek: *Young's Double-Slit Experiment with Atoms: A Simple Atom Interferometer.* Physical Reviews Letters, Vol. 66, NO. 21., (1991).

Neumann János: *A kvantummechanika matematikai alapjai*. Akadémiai Kiadó, (1980).

[QC] Mika Hirvensalo: *Quantum Computing.* Springer, (2001).

Gerard J. Milburn: The Feynman Processor. Perseus Books, (1998).

Samuel J. Lomonaco, JR.: *A Rosetta Stone for Quantum Mechanics with an introduction to quantum computation.* quant-ph/0007045

Paul M. B. Vitányi: *Quantum Kolmogorov Complexity Based on Classical Descriptions.* IEEE Trans. on Information Theory, Vol. 47., NO. 6., (2001)

A. Berthiaume, Wim van Dam, S. Laplante: *Quantum Kolmogorov Complexit.* quant-ph/0005018

E. Bernstein, U. Vazirani: *Quantum complexity theory.*

D. Aharonov, T. Naveh: Q*uantum NP – A Survey.* quant-ph/0210077.

[DE] David Deutsch, Artur Ekert: M*achines, Logic and Quantum Physics*. quant-ph/9911150.

[TQ] Riley T. Perry: *The Temple of Quantum Computing*.

[DL] Diósi Lajos: *Kvantum-információ elmélet*. ELTE előadás anyag.

[AE1] Artur Ekert et. al.: *On quantum algorithm*. quant-ph/9903061.

[AE2] Artur Ekert et. al.: *Basic concepts in quantum computation*.
Paul Vitányi: *The Quantum Computing Challenge*.

Eleanor Rieffel, Wolfgang Polak: *An Introduction to Quantum Computing for Non-Physicists*. quant-ph/9809016.
Arthur O. Pittenger: *An Introduction to Quantum Computing Algorithms*. Birkhauser, 2000.

XVIII.11 Bio

Fred H. Thaheld: *Does consciousness really collapse the wave function? A possible objective biophysical resolution of the measurement problem*. BioSystems 81 (2005) 113-124.

P.C.W. Davies: *Does quantum mechanics play a non-trivial role in life?* BioSystems 78 (2004) 69-79.

Andreas Mershin et al.: Towards Experimental Tests of Quantum Effects in Cytoskeletal Proteins.

Világi Ildikó: *Neurokémia*. Dialóg Campus, 2003.

A. Mershin et al.: *Tubulin dipole moment, dielectric constant and quantum behavior: computer simulations, experimental results and suggestion.* BioSystems 77 (2004) 73-85.

Stuart R. Hameroff: *A new theory of the origin of cancer: quantum coherent entanglement, centrioles, mitosis, and differentation.* BioSystems 77 (2004) 119-136.

Luiz Pinguelli Rosa, Jean Faber: *Quantum models of the brain: Are they compatible with environment decoherence?* Physical Review E 70, 031902 (2004).

Stuart Hameroff, Roger Penrose: *Orchestrated Objective Reduction of Quantum Coherence in Brain Microtubules: The „Orch OR" Model for Consciousness*. Mathematics and Computer Simulation 40:453-480, 1996.

Nick E. Mavromatos et al.: *QED-Cavity model of microtubules implies dissipationless energy transfer and biological quantum teleportation*. quant-ph/0204021.

Stuart Hameroff et al.: *Conduction pathways in microtubules, biological quantum computation, and consciousness.* BioSystems 64 (2002) 149-168.

Stuart Hameroff, Roger Penrose: *Conscious Events as Orchestrated Space-Time Selections.* NeuroQuantology 2003; I: 10-35.

Szabó Gábor: *Sejtbiológia.* Medicina, Budapest 2004.

Tartalomjegyzék

Ábrajegyzék

Táblázatjegyzék

Betűrendes tárgymutató