Prog1, C a gyakorlatban

Magasszintű programozási nyelvek

BSc előadás

Dr. Bátfai Norbert
egyetemi adjunktus
http://www.inf.unideb.hu/~nbatfai/
Debreceni Egyetem, Informatikai Kar,
Információ Technológia Tanszék
batfai.norbert@inf.unideb.hu
Skype: batfai.norbert

Prog1_4.ppt, v.: 0.0.2, 2011. 03. 09.
http://www.inf.unideb.hu/~nbatfai/#p1

Az óra blogja: http://progpater.blog.hu/
A Nokia Ovi store-ban is elérhető: http://store.ovi.com/content/100794

Felhasználási engedély

Bátfai Norbert
Debreceni Egyetem, Informatikai Kar, Információ Technológia Tanszék
<nbatfai@inf.unideb.hu, nbatfai gmail com>

Célok és tartalom

Előadás

a) IPC („azt mondja az egyik program a másiknak…”)
b) Klasszikus IPC problémák és megoldásuk
   Ebédelő filoszok
   Termelők és fogyasztó
   Olvasók és írók
c) Dijkstra-féle szemaforok

Labor

a) System V és POSIX szemaforok, üzenetsorok,osztott memória, Lokális, anonim és TCP/IP socketek, csővezetékek bevezetése

Laborkártyák

a) Példás kártyák

Otthoni opcionális feladat

a) A japán világbajnok HELIOS csapat szoftvereinek otthoni tanulmányozása.

Kapcsoldó videók, videómagyarázatok és blogok

http://progpater.blog.hu/2011/03/06/halozati_vegyertek

Az írásbeli és a szóbeli vizsgán bármi (jegyzet, könyv, forráskód, számítógép mobiltelefon stb.) használható! (Az írásbeli vizsgán beszélni viszont tilos.) Hiszen az én feladatom az lesz, hogy eldöntsem, jól felkészült programozóval, vagy mennyire felkészült programozóval állok szemben.

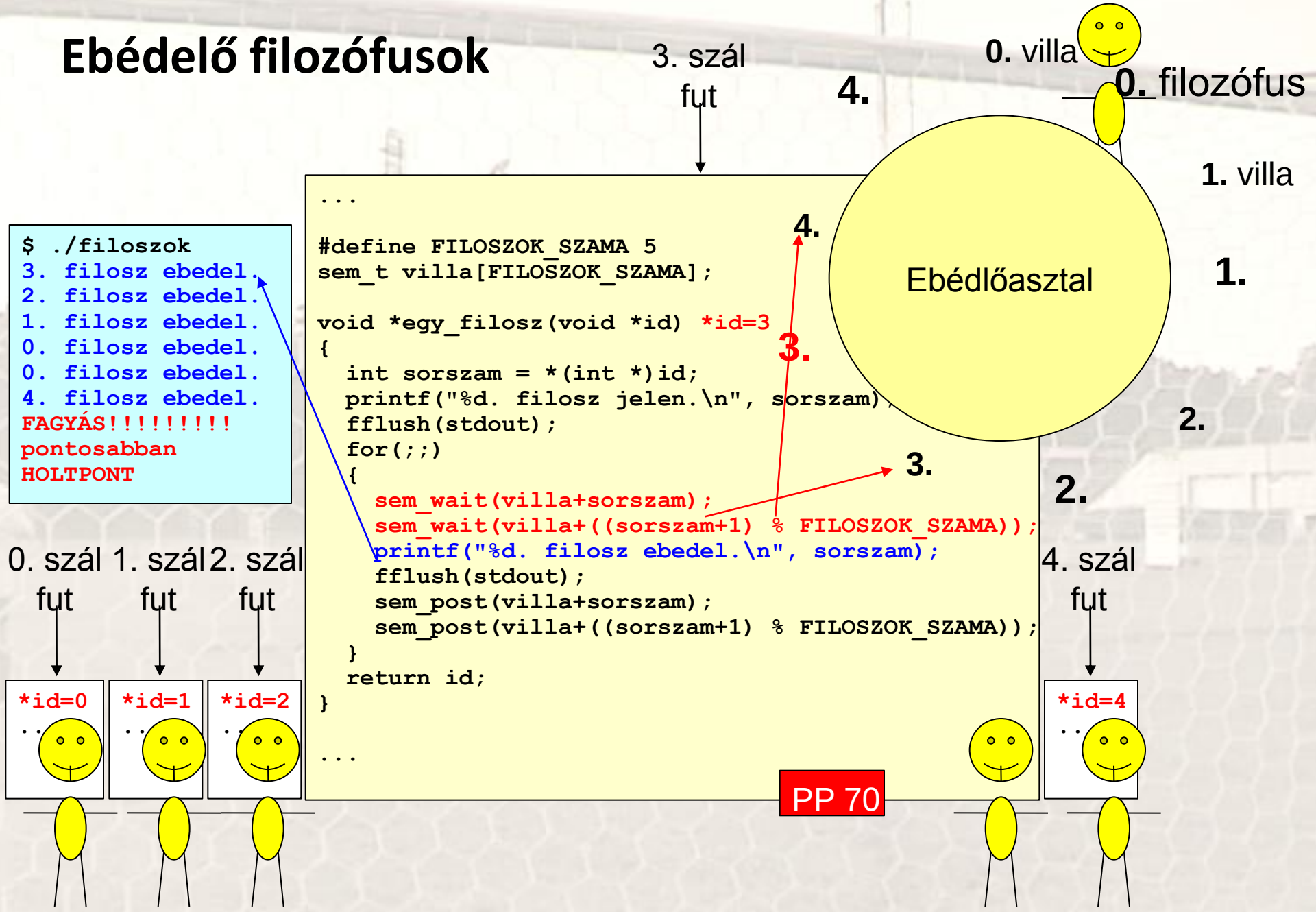
Minimális gyakorlati cél

A hallgató meg tudja írni (másolás alapján) a PP bármely rendszerhívásokat is tartalmazó (párhuzamos, socket interfészes) kódját.

Minimális elméleti cél

1) IPC, nevezetes problémák „elmesélése”
2) Legalább az egyik nevezetes IPC probléma megoldása (ez a fóliákon is látható „szemi-formális” (vagy C) nyelven bemutatott megoldás részletes interpretálását jelenti)

Ebédelő filozófusok



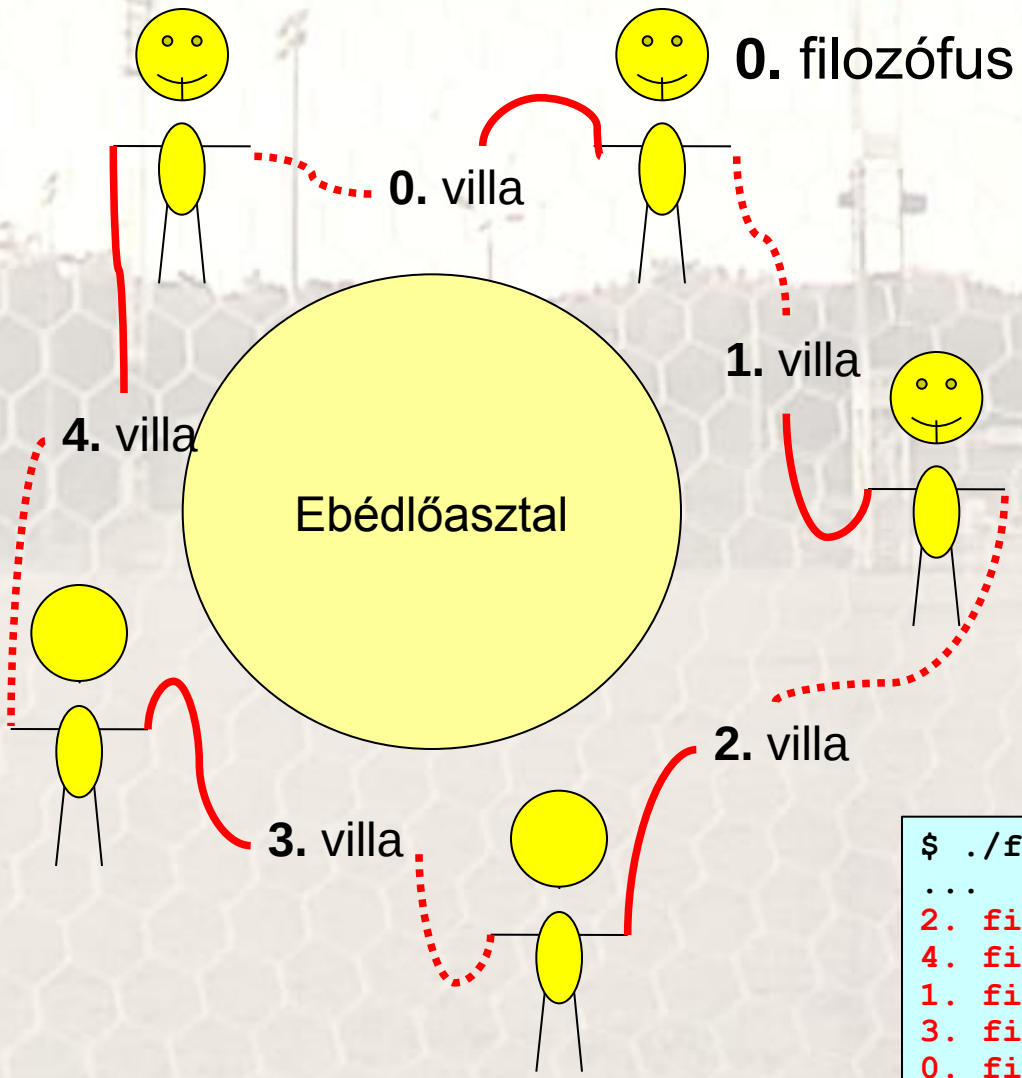
Ebédelő filozófusok

```
...

#define FILOSZOK_SZAMA 5
sem_t villa[FILOSZOK_SZAMA];

void *egy_filosz(void *id) *id=3
{
  int sorszam = *(int *)id;
  printf("%d. filosz jelen.\n", sorszam);
  fflush(stdout);
  for(;;)
  {
    sem_wait(villa+sorszam);
    printf("%d. filosz a 2. sem elott.\n", sorszam);
    fflush(stdout);
    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;
}

...
```

```
$ ./filoszok
...
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.
```

PP 72

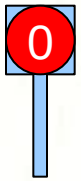
Ebédelő filozófusok



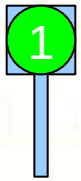

```
$ ./filoszok
...
2. filosz 2 sem elott.
4. filosz 2 sem elott.
1. filosz 2 sem elott.
3. filosz 2 sem elott.
0. filosz 2 sem elott.
```

1965, Dijkstra-féle szemaforok

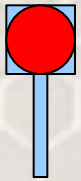
Az S szemafor egyfajta egész értékű változó két primitivvel (azaz oszthatatlan, „atomi” művelettel).



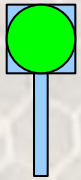
FOGLALT



SZABAD

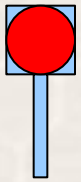


```
P(S) {
        S = S – 1;
        if(S < 0)
                // Várakozás S-en
                wait();
}
```

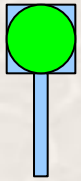


```
V(S) {
        S = S + 1;
        if(S <= 0)
                // Egy S-en
                // várakozó
                // ébresztése
                wakeup();
}
```

Vagy kevésbé szemléletesen:



```
P(S) {
        while(S <= 0) // Várakozás
                      //S-en
                ;
        S = S - 1;
}
```



```
V(S) {
        S = S + 1;
        // Ezzel S-en egy
        // várakozó
        // ébresztése
}
```

DOWN, **P**robeer (try), wait, pend, acquire

UP, **V**erhoog (inc), signal, post, release

Ismétlés: pthreads könyvtár, mutex zárak, pthreads_

```
PTHREAD_MUTEX_LOCK(P)      POSIX Programmer's Manual      PTHREAD_MUTEX_LOCK(P)

                                    ad_mutex_trylock, pthread_mutex_unlock - lock

       The mutex object
       pthread_mutex_loc
       thread shall bloc
       shall return wi
       state with the ca
```

```
...
#define SZALAK_SZAMA 100
int szamlalo = 0;

...

int
main(void)
{
   pthread_t sz[SZALAK_SZAMA];
...
```

```
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;
}
```

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

PP 67

```
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;
}
```

Kölcsönös kizárás

```
...
#define SZALAK_SZAMA 100
int szamlalo = 0;
pthread_mutex_t szamlalo_zar =

...

int
main(void)
{
   pthread_t sz[SZALAK_SZAMA];
...
```

```
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;
}
```

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

PP 69

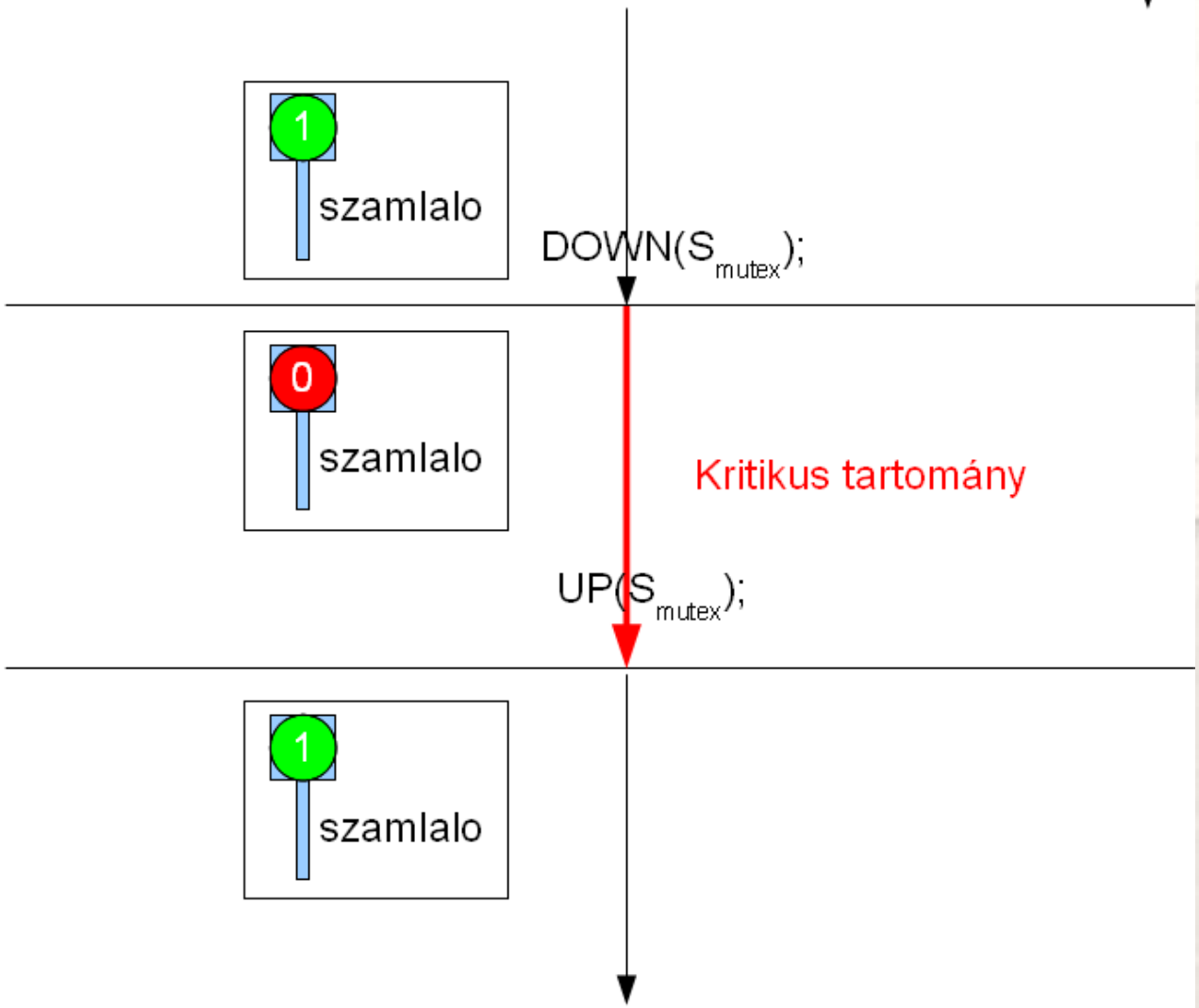
```
void *
csokkent_szal(void *id)
{
  int i;
  for(i=0; i<100; ++i) {
    printf("Szal: %d, %d\n", *(int *)id, pthread_self());
    fflush(stdout);
    var();
    pthread_mutex_lock(&szamlalo_zar);
    szamlalo = szamlalo - 1;
    pthread_mutex_unlock(&szamlalo_zar);
  }
  return id;
}
```

Kölcsönös kizárás bináris szemaforral

```
DOWN(S_mutex);

// kritikus szekció

UP(S_mutex);
```

```
P(S) {
      S = S – 1;
      if(S < 0)
            // Várakozás S-en
            wait();
}
```

```
V(S) {
      S = S + 1;
      if(S <= 0)
            // Egy S-en
            // várakozó
            // ébresztése
            wakeup();
}
```

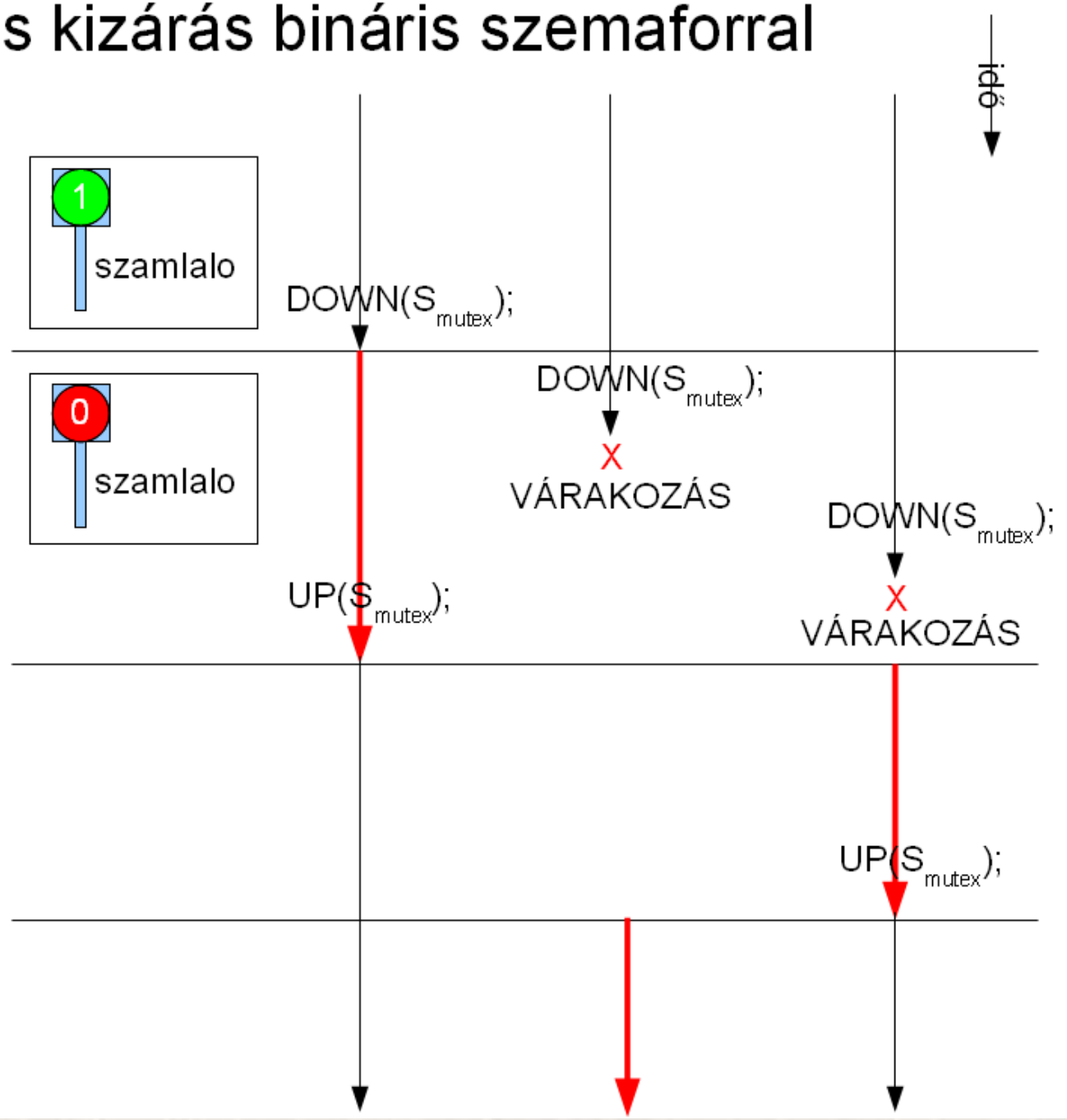


Kölcsönös kizárás bináris szemaforral

```
DOWN(S_mutex);

// kritikus szekció

UP(S_mutex);
```

```
P(S) {
    S = S – 1;
    if(S < 0)
        // Várakozás S-en
        wait();
}
```

```
V(S) {
    S = S + 1;
    if(S <= 0)
        // Egy S-en
        // várakozó
        // ébresztése
        wakeup();
}
```



Ebédelő filozófusok

OR 95, 2.18 ábra

```
FILOSZOK_SZAMA = 5;
int filosz_allapota[FILOSZOK_SZAMA];
Semaphore S_kritikus_szekcio_zar = 1;
Semaphore S_filosz_szemafor[FILOSZOK_SZAMA];
```

```
egy_filosz // a sorszam. filosz szála
{
  for (;;)
  {
    felveszi_villakat (sorszam);
    printf ("%d. filosz ebedel.\n", sorszam);
    fflush (stdout);
    leteszi_villakat (sorszam);
```

```
felveszi_villakat (int sorszam)
{
  DOWN (S_kritikus_szekcio_zar);
  filosz_allapota[sorszam] = FEL_AKARJA_VENNI;
  probal_villakat_felvenni (sorszam);
  UP (S_kritikus_szekcio_zar);
  DOWN(S_filosz_szemafor[sorszam]);
}
```

```
probal_villakat_felvenni (int sorszam)
{
  if (filosz_allapota[sorszam] == FEL_AKARJA_VENNI
      && filosz_allapota[(sorszam - 1) % FILOSZOK_SZAMA] != ESZIK
      && filosz_allapota[(sorszam + 1) % FILOSZOK_SZAMA] != ESZIK)
    {
      filosz_allapota[sorszam] = ESZIK;
      UP(S_filosz_szemafor[sorszam]);
```

Ebédelő filozófusok

OR 95, 2.18 ábra

```
FILOSZOK_SZAMA = 5;
int filosz_allapota[FILOSZOK_SZAMA];
Semaphore S_kritikus_szekcio_zar = 1;
Semaphore S_filosz_szemafor[FILOSZOK_SZAMA];
```

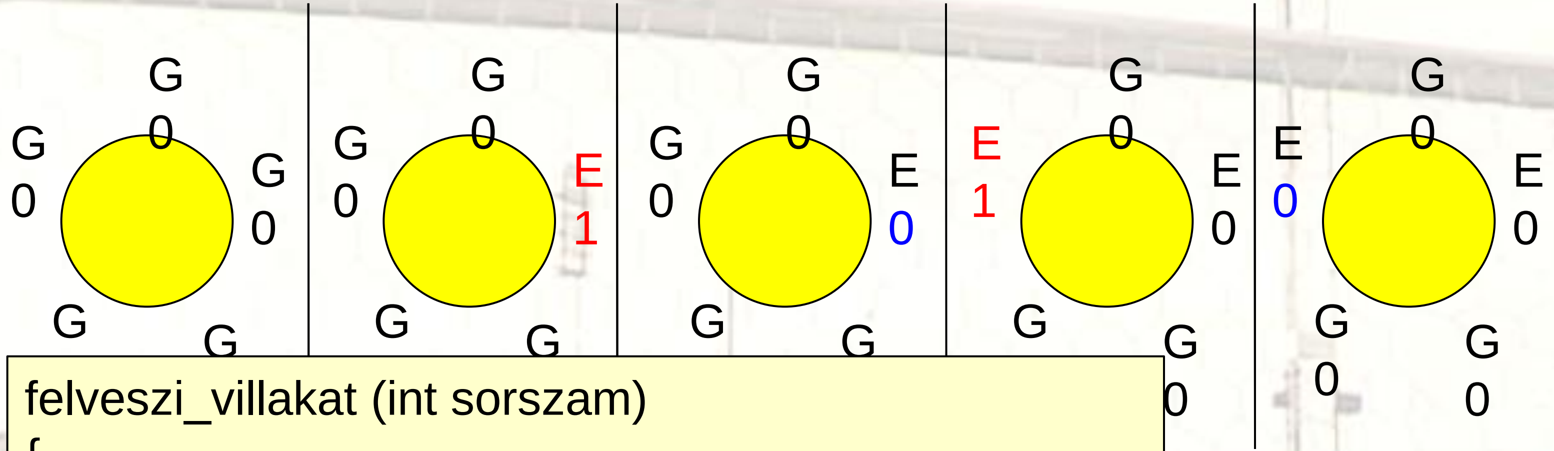
```
egy_filosz // a sorszam. filosz szála
{
  for (;;)
    {
      felveszi_villakat (sorszam);
      printf ("%d. filosz ebedel.\n", sorszam);
      fflush (stdout);
      leteszi_villakat (sorszam);
    }
}
```

Ebédelő filozófusok

OR 95, 2.18 ábra

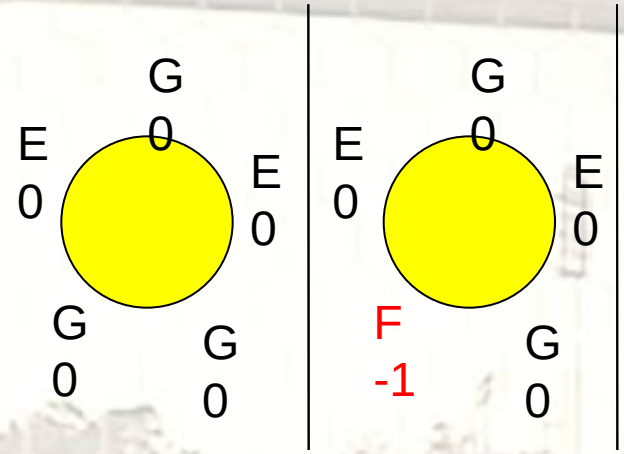
```
egy_filosz // a sorszam. filosz szála
{
  for (;;)
    {
      felveszi_villakat (sorszam);
      printf ("%d. filosz ebedel.\n", sorszam);
      fflush (stdout);
      leteszi_villakat (sorszam);
    }
}
```

```
leteszi_villakat (int sorszam)
{
  DOWN (S_kritikus_szekcio_zar);
  filosz_allapota[sorszam] = GONDOLKOZIK;
  probal_villakat_felvenni ((sorszam - 1) % FILOSZOK_SZAMA);
  probal_villakat_felvenni ((sorszam + 1) % FILOSZOK_SZAMA);
  UP (S_kritikus_szekcio_zar);
}
```



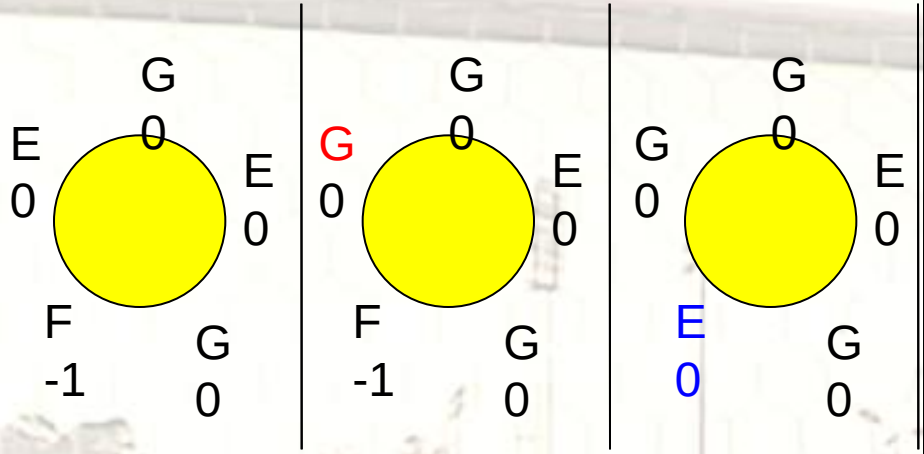

```
felveszi_villakat (int sorszam)
{
  DOWN (S_kritikus_szekcio_zar);
  filosz_allapota[sorszam] = FEL_AKARJA_VENNI;
  probal_villakat_felvenni (sorszam);
  UP (S_kritikus_szekcio_zar);
  DOWN(S_filosz_szemafor[sorszam]);
}
```

```
probal_villakat_felvenni (int sorszam)
{
  if (filosz_allapota[sorszam] == FEL_AKARJA_VENNI
      && filosz_allapota[(sorszam - 1) % FILOSZOK_SZAMA] != ESZIK
      && filosz_allapota[(sorszam + 1) % FILOSZOK_SZAMA] != ESZIK)
    {
      filosz_allapota[sorszam] = ESZIK;
      UP(S_filosz_szemafor[sorszam]);
    }
}
```




```
felveszi_villakat (int sorszam)
{
  DOWN (S_kritikus_szekcio_zar);
  filosz_allapota[sorszam] = FEL_AKARJA_VENNI;
  probal_villakat_felvenni (sorszam);
  UP (S_kritikus_szekcio_zar);
  DOWN(S_filosz_szemafor[sorszam]);
}
```

```
probal_villakat_felvenni (int sorszam)
{
  if (filosz_allapota[sorszam] == FEL_AKARJA_VENNI
      && filosz_allapota[(sorszam - 1) % FILOSZOK_SZAMA] != ESZIK
      && filosz_allapota[(sorszam + 1) % FILOSZOK_SZAMA] != ESZIK)
    {
      filosz_allapota[sorszam] = ESZIK;
      UP(S_filosz_szemafor[sorszam]);
    }
}
```




```
egy_filosz // a sorszam. filosz szála
{
  for (;;)
    {
      felveszi_villakat (sorszam);
      printf ("%d. filosz ebedel.\n", sorszam);
      fflush (stdout);
      leteszi_villakat (sorszam);
```

```
leteszi_villakat (int sorszam)
{
  DOWN (S_kritikus_szekcio_zar);
  filosz_allapota[sorszam] = GONDOLKOZIK;
  probal_villakat_felvenni ((sorszam - 1) % FILOSZOK_SZAMA);
  probal_villakat_felvenni ((sorszam + 1) % FILOSZOK_SZAMA);
  UP (S_kritikus_szekcio_zar);
```

```
probal_villakat_felvenni (int sorszam)
{
  if (filosz_allapota[sorszam] == FEL_AKARJA_VENNI
      && filosz_allapota[(sorszam - 1) % FILOSZOK_SZAMA] != ESZIK
      && filosz_allapota[(sorszam + 1) % FILOSZOK_SZAMA] != ESZIK)
    {
      filosz_allapota[sorszam] = ESZIK;
      UP(S_filosz_szemafor[sorszam]);
    }
```

Ebédelő filozófusok

```c
/*
 * OR 95. oldal 2.18. abrajanak C megvalositasa
 * POSIX mutex zarral es szemaforral.
 */

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

#define FILOSZOK_SZAMA 5
#define FEL_AKARJA_VENNI 0
#define CONDOLKOZIK 1

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 (filosz_szemafor + i, 0, 0);
  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 (filosz_szemafor + i);
  return 0;
}
```

Ebédelő filozófusok

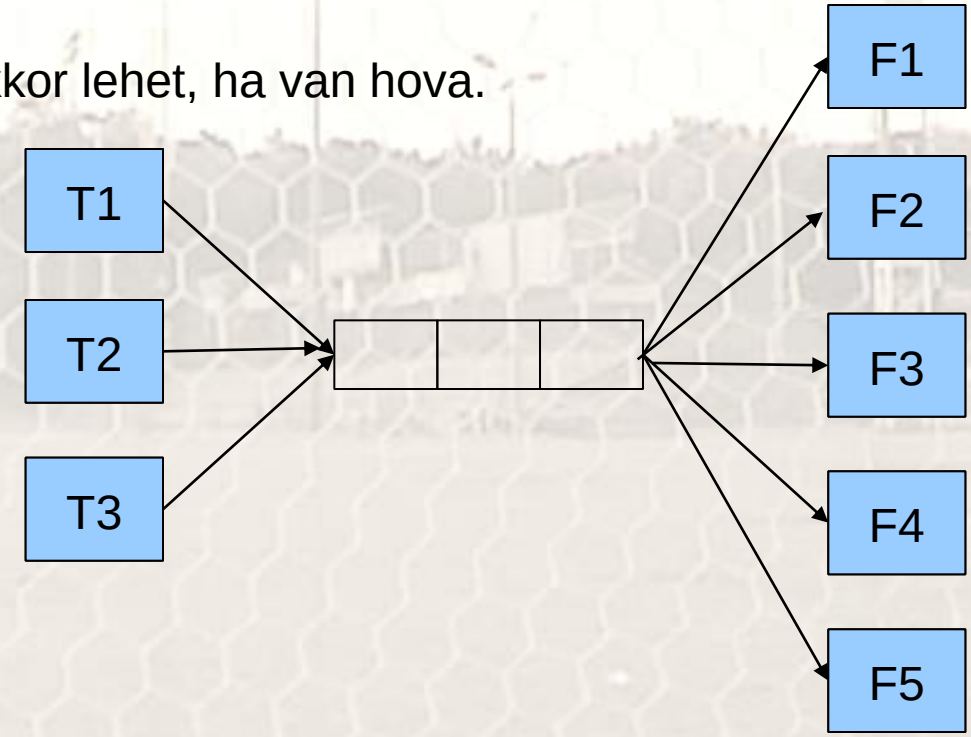
```
void *
egy_filosz (void *id)
{
  int sorszam = *(int *) id;
  for (;;)
    {
      felve
      print
      fflus
      letes
    }
  return id
}
```

```
void
felveszi_villakat (int sorszam)
{
  pthread_mutex_lock (&kritikus_szekcio_zar);
  filosz_allapota[sorszam] = FEL_AKARJA_VENNI;
  probal_villakat_felvenni (sorszam);
  pthread_mutex_unlock (&kritikus_szekcio_zar);
  sem_wait (filosz_szemafor + sorszam);

}

void
leteszi_villakat (int sorszam)
{
  pthread_mutex_lock (&kritikus_szekcio_zar);
  filosz_allapota[sorszam] = GONDOLKOZIK;
```

```
void
probal_villakat_felvenni (int sorszam)
{
  if (filosz_allapota[sorszam] == FEL_AKARJA_VENNI
      && filosz_allapota[(sorszam - 1) % FILOSZOK_SZAMA] != ESZIK
      && filosz_allapota[(sorszam + 1) % FILOSZOK_SZAMA] != ESZIK)
    {
      filosz_allapota[sorszam] = ESZIK;
      sem_post (filosz_szemafor + sorszam);
    }

}
```

Termelők és fogyasztók probléma

Betenni akkor lehet, ha van hova.



Kivenni akkor lehet, ha van mit.

Termelők és fogyasztók probléma

OR 85

Legyen N=3 a rekeszek száma!

N = 3;
Semaphore $S_{tömb\_mutex}$ = 1;
Semaphore S_{teli_helyek} = 0;
Semaphore $S_{üres\_helyek}$ = N;

S_{teli_helyek} = 0 $S_{üres\_helyek}$ = N

$S_{tömb\_mutex}$ = 1

OR 85, 2.12 ábra

```
Termelő() {
        for(;;) {
                // termel
                DOWN(S_üres_helyek);
                DOWN(S_tömb_mutex);
                // berak
                UP(S_tömb_mutex);
                UP(S_teli_helyek);
        }
}
```

```
Fogyasztó() {
        for(;;) {
                DOWN(S_teli_helyek);
                DOWN(S_tömb_mutex);
                // kivesz
                UP(S_tömb_mutex);
                UP(S_üres_helyek);
                // feldolgoz
        }
}
```

Futtatás

```
Termelő
    // termel
    P(S_üres_helyek);
    P(S_tömb_mutex);
    // berak
    V(S_tömb_mutex);
    V(S_teli_helyek);
```

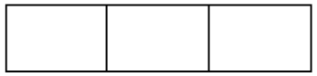
```
Fogyasztó
    P(S_teli_helyek);
    P(S_tömb_mutex);
    // kivesz
    V(S_tömb_mutex);
    V(S_üres_helyek);
    // feldolgoz
```

```
P(S)
    S = S – 1;
    if(S < 0)
        wait();
```

```
V(S)
    S = S + 1;
    if(S <= 0)
        wakeup();
```

Mennyit tudunk kivenni? betenni?

$S_{teli\_helyek} = 0$ $S_{üres\_helyek} = 3$

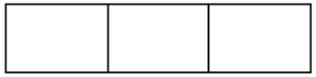


T1

termel 42

$P(S_{üres\_helyek})$;

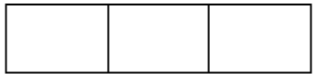
$S_{teli\_helyek} = 0$ $S_{üres\_helyek} = 2$



F1

$P(S_{teli\_helyek})$;

$S_{teli\_helyek} = -1$ $S_{üres\_helyek} = 2$



VÁRAKOZIK...

idő

Futtatás

```
Termelő
    // termel
    P(S_üres_helyek);
    P(S_tömb_mutex);
    // berak
    V(S_tömb_mutex);
    V(S_teli_helyek);
```

```
Fogyasztó
    P(S_teli_helyek);
    P(S_tömb_mutex);
    // kivesz
    V(S_tömb_mutex);
    V(S_üres_helyek);
    // feldolgoz
```

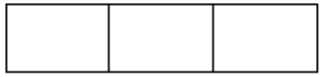
```
P(S)
    S = S – 1;
    if(S < 0)
        wait();
```

```
V(S)
    S = S + 1;
    if(S <= 0)
        wakeup();
```

Mennyit tudunk

kivenni? S_{teli_helyek}

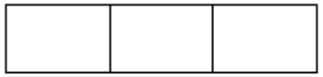
betenni? $S_{üres\_helyek}$



F2

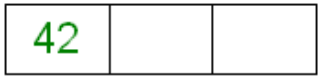
$P(S_{teli\_helyek})$;

$S_{teli\_helyek} = -2$ $S_{üres\_helyek} = 2$



VÁRAKOZIK...

$S_{teli\_helyek} = -2$ $S_{üres\_helyek} = 2$



$V(S_{teli\_helyek})$;

$S_{teli\_helyek} = -1$ $S_{üres\_helyek} = 2$

```
Termelő
    // termel
    P(S_üres_helyek);
    P(S_tömb_mutex);
    // berak
    V(S_tömb_mutex);
    V(S_teli_helyek);
```

```
Fogyasztó
    P(S_teli_helyek);
    P(S_tömb_mutex);
    // kivesz
    V(S_tömb_mutex);
    V(S_üres_helyek);
    // feldolgoz
```

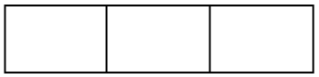
```
P(S)
    S = S – 1;
    if(S < 0)
        wait();
```

```
V(S)
    S = S + 1;
    if(S <= 0)
        wakeup();
```

Futtatás

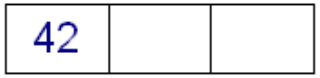
Mennyit tudunk

kivenni? S_{teli_helyek} betenni? $S_{üres\_helyek}$



FELÉBRED

$S_{teli\_helyek} = -1$ $S_{üres\_helyek} = 2$



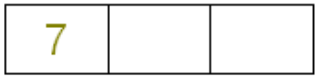
$V(S_{üres\_helyek})$;

$S_{teli\_helyek} = -1$ $S_{üres\_helyek} = 3$

termel 42

$P(S_{üres\_helyek})$;

$S_{teli\_helyek} = -1$ $S_{üres\_helyek} = 2$



$V(S_{teli\_helyek})$;

$S_{teli\_helyek} = 0$ $S_{üres\_helyek} = 2$

Futtatás

```
Termelő
    // termel
    P(S_üres_helyek);
    P(S_tömb_mutex);
    // berak
    V(S_tömb_mutex);
    V(S_teli_helyek);
```

```
Fogyasztó
    P(S_teli_helyek);
    P(S_tömb_mutex);
    // kivesz
    V(S_tömb_mutex);
    V(S_üres_helyek);
    // feldolgoz
```

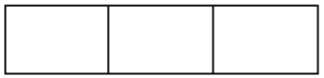
```
P(S)
    S = S – 1;
    if(S < 0)
        wait();
```

```
V(S)
    S = S + 1;
    if(S <= 0)
        wakeup();
```

Mennyit tudunk

kivenni? betenni?

S_{teli_helyek} $S_{üres\_helyek}$



FELÉBRED

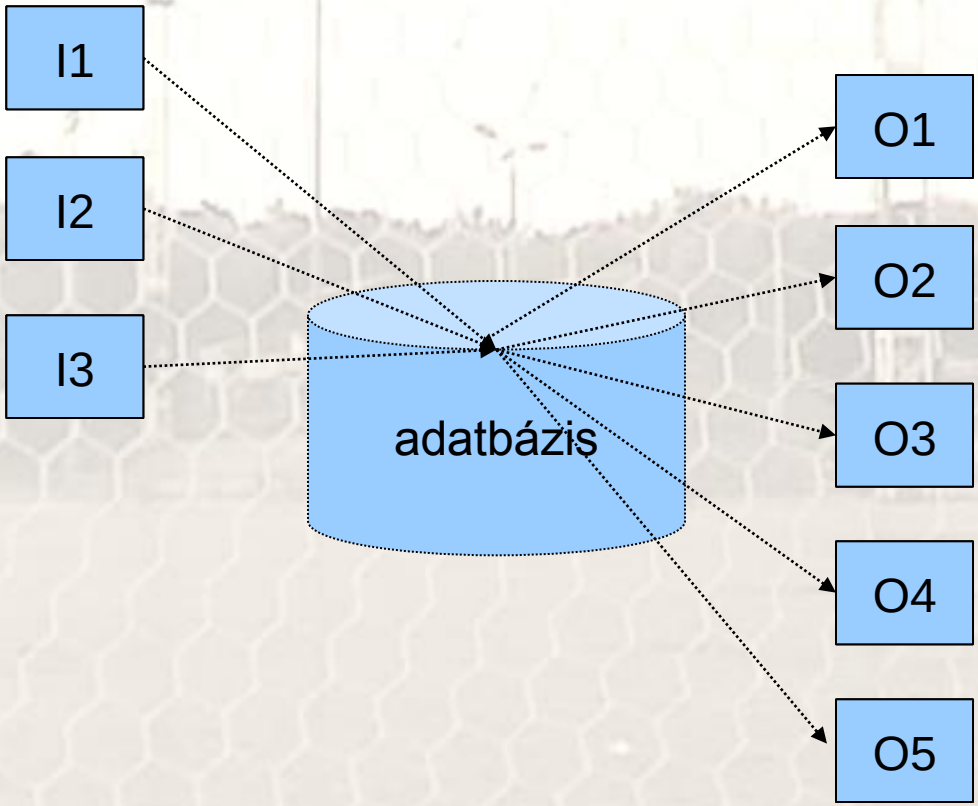
$S_{teli\_helyek} = 0$ $S_{üres\_helyek} = 2$

$V(S_{üres\_helyek})$;

$S_{teli\_helyek} = 0$ $S_{üres\_helyek} = 3$

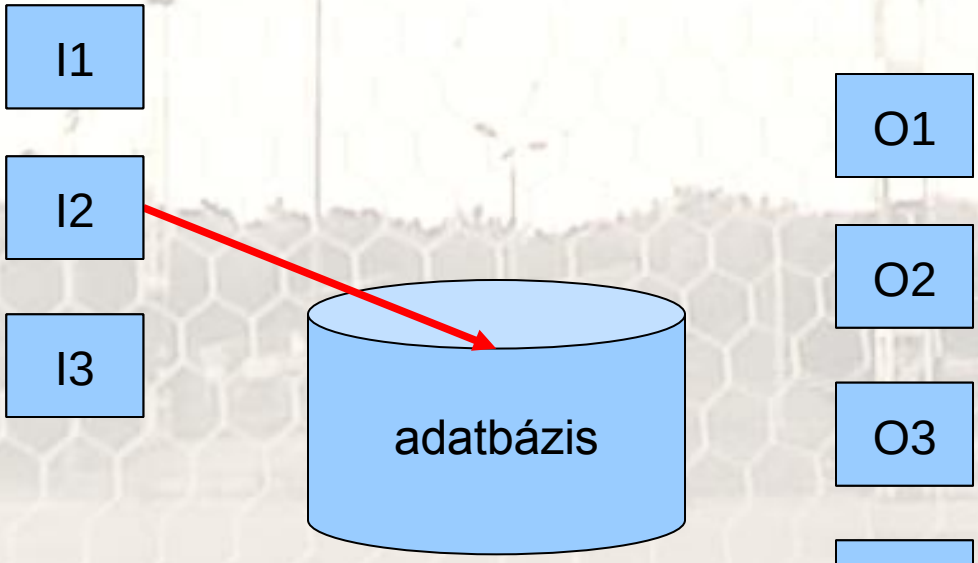
Olvasók és írók

OR 96



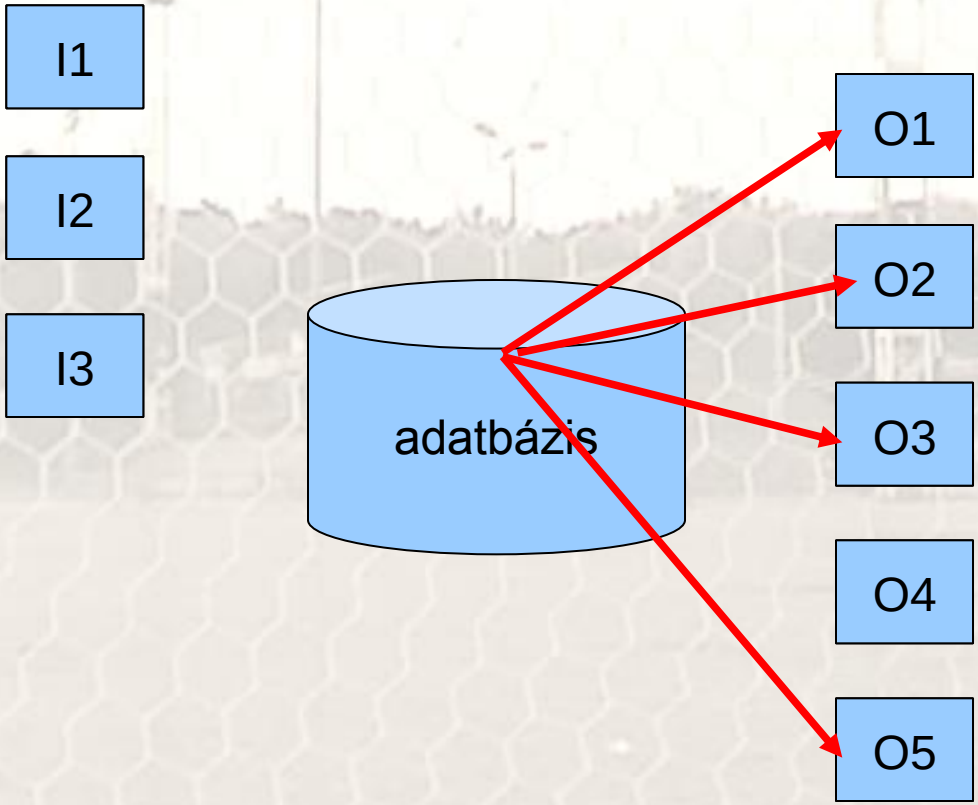
Az írók írni, az olvasók olvasni szeretnék.

Olvasók és írók



Vagy egy írhatja.

Olvasók és írók



Vagy bármennyi olvashatja.

Olvasók és írók

OR 97, 2.19 ábra

```
Író() {
        for(;;) {
                DOWN(S_adatb);
                // ír
                UP(S_adatb);
        }
}
```

```
int olvaso_szamlalo = 0;
Semaphore S_mutex = 1;
Semaphore S_adatb = 1;
```

```
Olvasó() {
        for(;;) {
                DOWN(S_mutex);
                ++olvaso_szamlalo;
                if(olvaso_szamlalo == 1)
                        DOWN(S_adatb);
                UP(S_mutex);
                // olvas
                DOWN(S_mutex);
                --olvaso_szamlalo;
                if(olvaso_szamlalo == 0)
                        UP(S_adatb);
                UP(S_mutex);
        }
}
```

Olvasók és írók

```c
/*
 * OR 97. oldal 2.19. abrajanak C megvalositasa
 * POSIX mutex zarral es szemaforral.
 */

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

#define S

int olvas
sem_t ada
pthread_m
```

```c
int
main (void)
{
  pthread_t sz[SZALAK_SZAMA];
  int s[SZALAK_SZAMA], *r, i;

  sem_init (&adatb, 0, 1);

  for (i = 0; i < SZALAK_SZAMA; ++i)
    {
      s[i] = i;
      if (pthread_create (&sz[i], NULL,
                          (i < SZALAK_SZAMA / 5) ? olvaso : iro,
                          (void *) &s[i]))
        {
          perror ("Hiba");
          exit (-1);
        }
    }
  for (i = 0; i < SZALAK_SZAMA; ++i)
    {
      pthread_join (sz[i], (void *) &r);
    }
  sem_destroy (&adatb);
}
```

OR 97

Olvasók és írók

```
void *
iro (void *id)
{
  int sorszam = *(int
  for (;;)
    {
      sem_wait (&adat
      printf ("%d. ir
      fflush (stdout)
      sem_post (&adat
    }
  return id;
}
```

```
void *
olvaso (void *id)
{
  int sorszam = *(int *) id;
  for (;;)
    {
      pthread_mutex_lock (&kritikus_szekcio_zar);
      ++olvasok_szama;
      if (olvasok_szama == 1)
        sem_wait (&adatb);
      pthread_mutex_unlock (&kritikus_szekcio_zar);
      printf ("%d. olvaso olvas.\n", sorszam);
      fflush (stdout);
      pthread_mutex_lock (&kritikus_szekcio_zar);
      --olvasok_szama;
      if (olvasok_szama == 0)
        sem_post (&adatb);
      pthread_mutex_unlock (&kritikus_szekcio_zar);
    }
  return id;
}
```

Nem könnyű velük programozni!

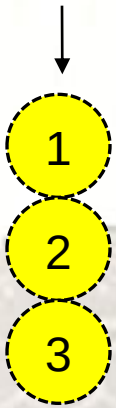
```
Író() {
        for(;;) {
                DOWN(S_adatb);
                DOWN(S_mutex);
                // „tesztelésként” kiíratjuk
                // az olvasók számát,
                // mert ennek itt nullának
                // kell(ene) lennie!
                UP(S_mutex);
                // ír
                UP();
        }
}
```

```
Olvasó() {
        for(;;) {
                DOWN(S_mutex);
                ++olvaso_szamlalo;
                if(olvaso_szamlalo == 1)
                        DOWN(S_adatb);
                UP(S_mutex);
                // olvas
                DOWN(S_mutex);
                --olvaso_szamlalo;
                if(olvaso_szamlalo == 0)
                        UP(S_adatb);
                UP(S_mutex);
        }
}
```

```
int olvaso_szamlalo = 0;
Semaphore S_mutex= 1;
Semaphore S_adatb= 1;
```

Tervünk, hogy mutex-el védve férünk hozzá a kritikus olvaso_szamlalo-hoz... Mi fog történni, ha a fent pirossal szedett módon ezt meg is tesszük???

Holtpont!



HOLTPONT!

```
Író() {
        for(;;) {                    1   1
                DOWN(S_adatb);
                DOWN(S_mutex);       3
                // „tesztelésként” kíratjuk
                // az olvasók számát,
                // mert ennek itt nullának
                // kell(ene) lennie!
                UP(S_mutex);
                // ír
                UP(S_adatb);
        }
}
```

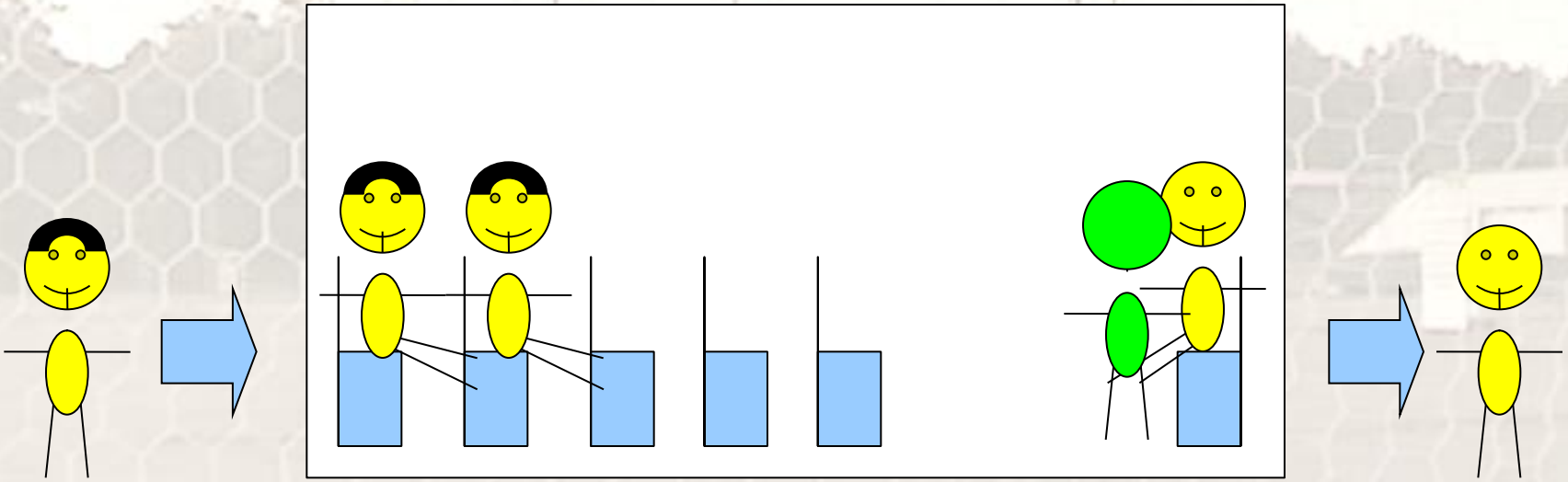
```
Olvasó() {
        for(;;) {                    2   2
                DOWN(S_mutex);
                ++olvaso_szamlalo;
        3       if(olvaso_szamlalo == 1)
                        DOWN(S_adatb);   4
                UP(S_mutex);
                // olvas
                DOWN(S_mutex);
                --olvaso_szamlalo;
                if(olvaso_szamlalo == 0)
                        UP(S_adatb);
                UP(S_mutex);
        }
}
```

```
int olvaso_szamlalo = 0;
Semaphore S_mutex = 1;
Semaphore S_adatb = 1;
```

3. VÁR 2. miatt, 4. VÁR 1. miatt!

Alvó borbély

OR 99



Alvó borbély

OR 99, 2.21 ábra

```
Borbély() {
        for(;;) {
                DOWN(S_vendég);
                DOWN(S_mutex);
                --varakozok_szama;
                UP(S_borbély);
                UP(S_mutex);
                // hajat vág
        }
}
```

```
Vendég() {
        for(;;) {
                DOWN(S_mutex);
                if(varakozok_szama < SZEKEK_SZAMA) {
                        ++varakozok_szama;
                        UP(S_vendég);
                        UP(S_mutex);
                        DOWN(S_borbély);
                        // haját vágják
                } else {
                        UP(S_mutex);
                }
        }
}
```

```
#define SZEKEK_SZAMA
int varakozok_szama = 0;
Semaphore S_mutex = 1;
Semaphore S_borbély = 0;
Semaphore S_vendég = 0;
```

Alvó borbély

```c
/*
 * OR 99. oldal 2.21. abrajanak C megvalositasa
 * POSIX mutex zarral es szemaforral.
 */

#include <stdio.h>
#include <stdlib.h>
#inc
#inc

#def
#def

int
sem_
pthr

int
main (void)
{
  pthread_t sz[SZALAK_SZAMA];
  int s[SZALAK_SZAMA], *r, i;

  sem_init (&borbelys, 0, 0);
  sem_init (&vendegs, 0, 0);

  for (i = 0; i < SZALAK_SZAMA; ++i)
    {
      s[i] = i;
      if (pthread_create (&sz[i], NULL,
                          (i == 0) ? borbely : vendeg, (void *) &s[i]))
        {
          perror ("Hiba");
          exit (-1);
        }
    }
  for (i = 0; i < SZALAK_SZAMA; ++i)
    {
      pthread_join (sz[i], (void *) &r);
    }
  sem_destroy (&borbelys);
  sem_destroy (&vendegs);
}
```

Alvó borbély

```
void *
borbely (void *id)
{
  int sorszam = *(int *) id;
  for (;;)
    {
      sem_wa
      pthrea
      --vara
      sem_po
      pthrea
      printi
      fflush
    }
  return id;
}
```

```
void *
vendeg (void *id)
{
  int sorszam = *(int *) id;
  for (;;)
    {
      pthread_mutex_lock (&kritikus_szekcio_zar);

      if (varakozok_szama < SZEKEK_SZAMA)
        {
          ++varakozok_szama;
          sem_post (&vendegs);
          pthread_mutex_unlock (&kritikus_szekcio_zar);
          sem_wait (&borbelys);
          printf ("Vendeg (%d szal) hajat vagjak.\n", sorszam);
          fflush (stdout);
        }
      else
        {
          pthread_mutex_unlock (&kritikus_szekcio_zar);

        }

    }
  return id;
}
```

System V és POSIX szemaforok, üzenetsorok, osztott memória

```
SVIPC(7)                    Linux Programmer's Manual                    SVIPC(7)

NAME
       svipc - System V interprocess communication mechanisms

SYNOPSIS
       # include
       # include
       # include
       # include
       # include

DESCRIPTION
       This  mar
       i
       a
       a
...
```

```
...
  int uzenetsor;
  if ((uzenetsor =
       msgget (ftok (".", 43), IPC_CREAT | S_IRUSR | S_IWUSR))
         == -1)
    {
      perror ("msgget");
      exit (EXIT_FAILURE);
    }
...
```

PP 56

```
$ 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
```

Üzenetsorok

```
MSGGET(2)                  Linux Programmer's Manual                  MSGGET(2)

NAME
SYNO
DESC
...
```

```
MSGOP(2)                   Linux Programmer's Manual                   MSGOP(2)

NAME
       msgop - message operations

SYNOPSIS
       #include <sys/types.h>
       #include <sys/ipc.h>
       #include <sys/msg.h>

       int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);

       ssize_t  msgrcv(int  msqid,  void *msgp, size_t msgsz, long msgtyp, int
       msgflg);

DESCRIPTION
       The msgsnd() and msgrcv() system calls are used, respectively, to  send
       messages  to,  and receive messages from, a message queue.  The calling
       process must have write permission on the message  queue  in  order  to
       send a message, and read permission to receive a message.

...
```

Üzenetsorok

```
...
  int uzenetsor;
  if ((uzenetsor =
       msgget (ftok (".", 43), IPC_CREAT | S_IRUSR | S_IWUSR))
         == -1)
    {
      perror ("msgget");
      exit (EXIT_FAILURE);
    }
...
```

PP 56

```
FTOK(3)                    Linux Programmer's Manual                    FTOK(3)

NAME
       ftok  -  convert  a pathname and a project identifier to a System V IPC
       key

SYNOPSIS
       # include <sys/types.h>
       # include <sys/ipc.h>

       key_t ftok(const char *pathname, int proj_id);

DESCRIPTION
       The ftok() function uses the identity of the file named  by  the  given
       pathname  (which  must  refer  to an existing, accessible file) and the
       least significant 8 bits of proj_id (which must be non-zero) to
       generate  a  key_t  type  System V IPC key, suitable for use with
       msgget(2), semget(2), or shmget(2).
...
```

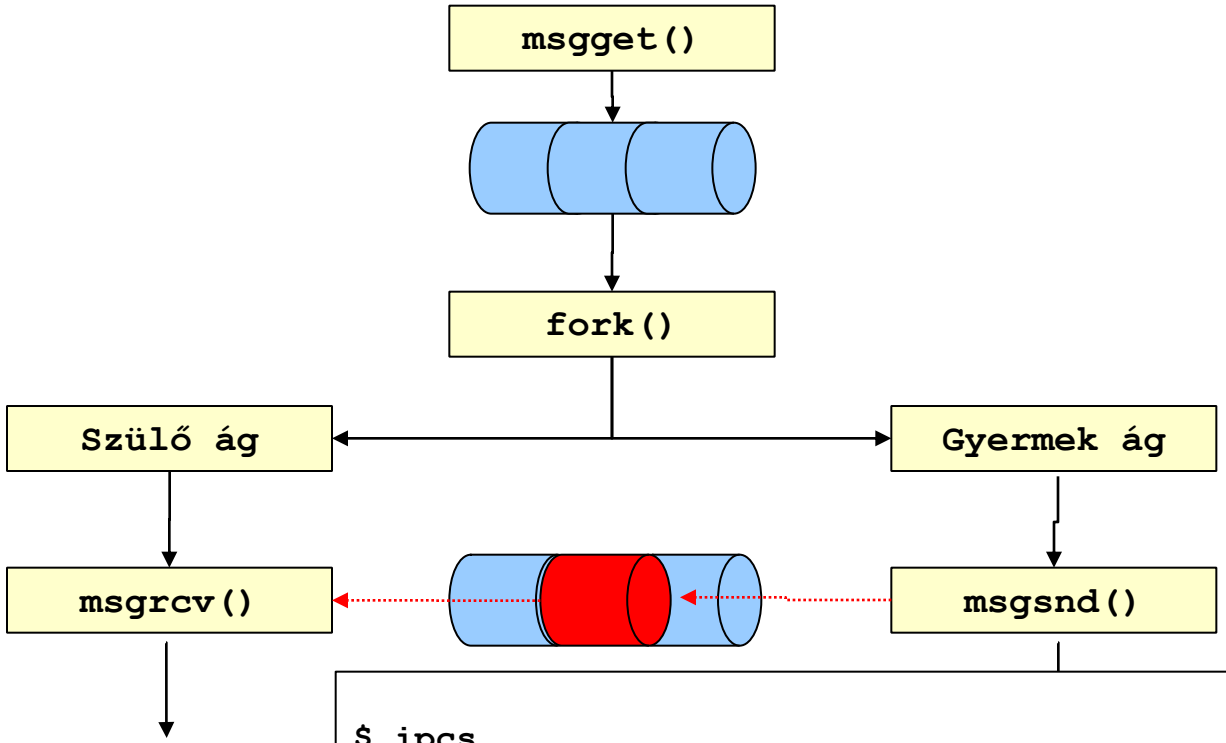
Üzenetsorok

```
/include/linux/msg.h
...
#define MSGMNI    16   /* <= IPCMNI */     /* max # of msg queue identifiers */
#define MSGMAX  8192   /* <= INT_MAX */   /* max size of message (bytes) */
#define MSGMNB 16384   /* <= INT_MAX */   /* default max size of a message queue */...
...
```

```
$ more /proc/sys/kernel/msgmni
16
$ more /proc/sys/kernel/msgmax
8192
$ more /proc/sys/kernel/msgmnb
16384
```

PP 55

Üzenetsorok
PP példa




```
$ 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
```

A Pi jegyei, több folyamattal párhuzamosan

```
$ 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 sza
24565> 101-200 s
2 szamolo folyar
24566> 201-300 s
3 szamolo folyar
24565> 101-200 s
24565> 301-400 s
24567> 401-500 s
4 szamolo folyar
24568> 501-600 s
5 szamolo folyar
24569> 601-700 s
24566> 201-300 s
24566> 701-800 s
24565> 301-400 szamolasa kesz
24565> 801-900 szamolasa indul
24568> 501-600 s
24568> 901-1000
24566> 701-800 s
24566> nincs tob
24567> 401-500 s
24567> nincs tob
24569> 601-700 s
24569> nincs tob
24565> 801-900 s
24565> nincs tob
24568> 901-1000
24568> nincs tob
```

PP 231

```
$ 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
```

```
$ 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
```

PP 237

POSIX üzenetsorok

```
MQ_OVERVIEW(7)              Linux Programmer's Manual              MQ_OVERVIEW(7)

NAME
```

```
...
LINUX SPECIFIC DETAILS
   Versions
       POSIX  message  queues have been supported on Linux since kernel 2.6.6.
       Glibc support has been provided since version 2.3.4.

   Kernel configuration
       Support  for  POSIX  message  queues  is  configurable  via  the  CON-
       FIG_POSIX_MQUEUE  kernel  configuration option.  This option is enabled
```

```
$ more /proc/sys/fs/mqueue/msgsize_max
8192
$ more /proc/sys/fs/mqueue/msg_max
10
$ more /proc/sys/fs/mqueue/queues_max
256
```

```
sgget(2), msgsnd(2), msgrcv(2), etc.) are  an
g  messages between processes.  POSIX message
designed  interface  than  System  V  message
d POSIX message queues are less widely avail-
       able (especially on older systems) than System V message queues.

arious message queue functions  is  shown  in
```

Az üzenetsor felmountolása Linuxon:

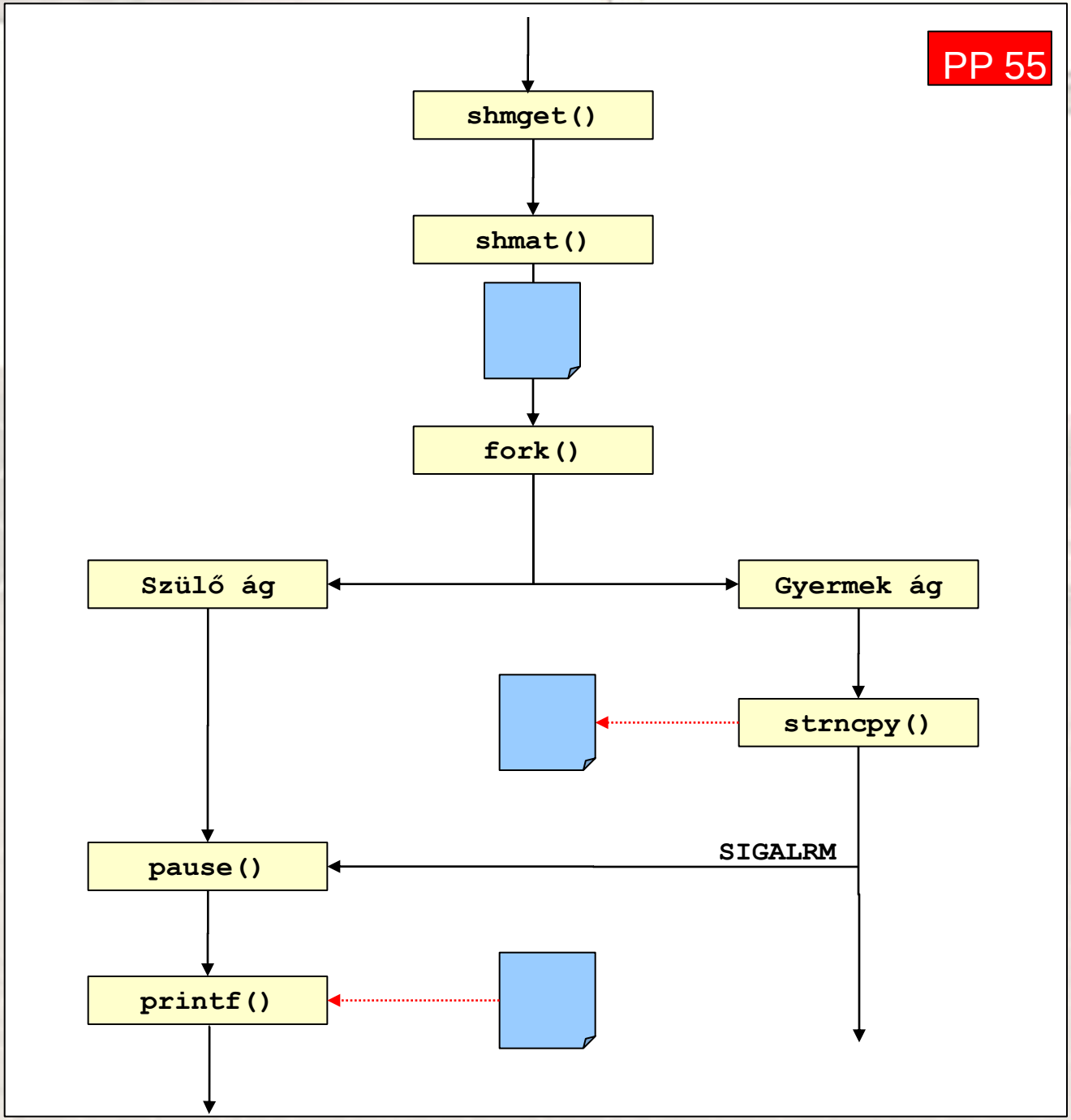
```
$ mkdir uzenetsor
# mount -t mqueue none uzenetsor
$ ls -l uzenetsor
```

Osztott memória

```
SHMGET(2)                  Linux Programmer's Manual                  SHMGET(2)

NAME

SYNOP

DESCR

...
```

```
SHMOP(2)                   Linux Programmer's Manual                   SHMOP(2)

NAME
       shmop - shared memory operations

SYNOPSIS
       #include <sys/types.h>
       #include <sys/shm.h>

       void *shmat(int shmid, const void *shmaddr, int shmflg);

       int shmdt(const void *shmaddr);

DESCRIPTION
       shmat()  attaches  the shared memory segment identified by shmid to the
       address space of the calling process.  The attaching address is  speci-
       fied by shmaddr with one of the following criteria:

       If  shmaddr  is NULL, the system chooses a suitable (unused) address at
       which to attach the segment.

...
```

Osztott memória PP példa



Osztott memória

```
/include/linux/shm.h
...
/*
 * SHMMAX, SHMMNI and SHMALL are upper limits are defaults which can
 * be increased by sysctl
 */

#define SHMMAX 0x2000000           /* max shared seg size (bytes) */
#define SHMMIN 1                   /* min shared seg size (bytes) */
#define SHMMNI 4096                /* max num of segs system wide */
#define SHMALL (SHMMAX/PAGE_SIZE*(SHMMNI/16)) /* max shm system wide (pages) */
#define SHMSEG SHMMNI                      /* max shared segs per process */
...
```

```
$ more /proc/sys/kernel/shmmni
4096
$ more /proc/sys/kernel/shmmax
33554432
$ more /proc/sys/kernel/shmall
2097152
```

Szemaforok PP példa

```
...
  struct sembuf zar, nyit;
  zar.sem_num = 0;
  zar.sem_op = -1;
  nyit.sem_num = 0;
  nyit.sem_op = 1;
...


...szemafor =
       semget (ftok (".", 42), 1, IPC_CREAT | S_IRUSR | S_IWUSR))...



...semctl (szemafor, 0, SETVAL, 1)...
```

PP 111

```
...semop (szemafor, &zar, 1)...

KRITIKUS SZAKASZ

...semop (szemafor, &nyit, 1)...
```

Blokkolódik, amíg a szemafor értéke < | zar.sem_op = -1|, majd csökkenti (sem_op < 0)

Növeli a szemafor értékét (sem_op > 0)

Szemafortömbök

```
SEMGET(2)                  Linux Programmer's Manual                  SEMGET(2)

NAME

SYNOP

DESCR

...
```

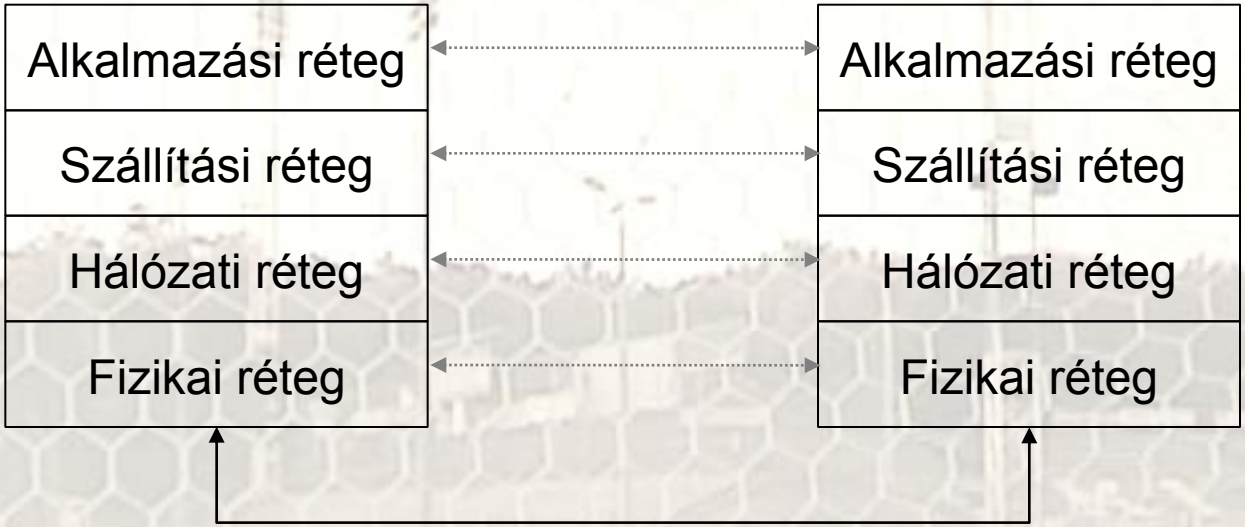
```
SEMOP(2)                   Linux Programmer's Manual                   SEMOP(2)

NAME
       semop, semtimedop - semaphore operations

SYNOPSIS
       #include <sys/types.h>
       #include <sys/ipc.h>
       #include <sys/sem.h>

       int semop(int semid, struct sembuf *sops, unsigned nsops);

       int  semtimedop(int  semid, struct sembuf *sops, unsigned nsops, struct
       timespec *timeout);

DESCRIPTION
       Each semaphore in a semaphore set has the following associated values:

           unsigned short  semval;   /* semaphore value */
           unsigned short  semzcnt;  /* # waiting for zero */
           unsigned short  semncnt;  /* # waiting for increase */
           pid_t           sempid;   /* process that did last op */
...
```

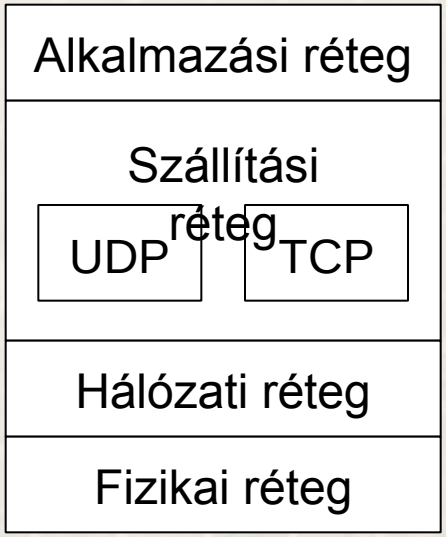
POSIX szemaforok

```
SEM_OVERVIEW(7)            Linux Programmer's Manual            SEM_OVERVIEW(7)

NAME
       sem_overview - Overview of POSIX semaphores

DESCRIPTION
       POSIX  semaphores  allow  processes  and  threads  to synchronise their
       actions.

       A semaphore is an integer whose value is never allowed  to  fall  below
       zero.   Two  operations  can  be performed on semaphores: increment the
       semaphore value by one (sem_post(3)); and decrement the semaphore value
       by  one  (sem_wait(3)).  If the value of a semaphore is currently zero,
       then a sem_wait(3) operation will block until the value becomes greater

...
LINUX
   Ver

...
```

```
...
CONFORMING TO
       POSIX.1-2001.

NOTES
       System V semaphores (semget(2), semop(2), etc.) are an older  semaphore
       API.  POSIX semaphores provide a simpler, and better designed interface
       than System V semaphores; on the other hand POSIX semaphores  are  less
       widely   available   (especially   on   older   systems)   than  System  V
       semaphores.

EXAMPLE
       An example of the use of various POSIX semaphore functions is shown  in
       sem_wait(3).
...
```

TCP/IP modell





Berkeley socket programozási interfész

Socket IPC

```
SOCKET(2)                  Linux Programmer's Manual                  SOCKET(2)

NAME
       socket - create an endpoint for communication

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>

       int socket(int domain, int type, int protocol);

DESCRIPTION
       socket()  creates  an endpoint for communication and returns a descrip-
       tor.

       The domain parameter specifies a communication domain; this selects the
       protocol  family  which will be used for communication.  These families
       are  defined  in  <sys/socket.h>.   The  currently  understood  formats
       include:

       Name                Purpose                                  Man page
       PF_UNIX, PF_LOCAL   Local communication                      unix(7)
       PF_INET             IPv4 Internet protocols                  ip(7)
...
```

- Unix domain socketek
- Internet domain socketek

```
kapuleiro = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)
```

BSD, SVR4

Socket IPC

The socket has the indicated type, which specifies the communication semantics. Currently defined types are:

SOCK_STREAM
Provides sequenced, reliable, two-way, connection-based byte streams. An out-of-band data transmission mechanism may be supported.

SOCK_DGRAM
Supports datagrams (connectionless, unreliable messages of a fixed maximum length).

SOCK_SEQPACKET
Provides a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer is required to read an entire packet with each read system call.

SOCK_RAW
Provides raw network protocol access.

SOCK_RDM
Provides a reliable datagram layer that does not guarantee

...

```
kapuleiro = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)
```

Lokális (Unix domain) socketek

```
UNIX(7)                    Linux Programmer's Manual                    UNIX(7)

NAME
       unix,  PF_UNIX,  AF_UNIX, PF_LOCAL, AF_LOCAL - Sockets for local inter-
       process communication

SYNOPSIS
       #include <sys/socket.h>
       #include <sys/un.h>

       unix_socket = socket(PF_UNIX, type, 0);
       error = socketpair(PF_UNIX, type, 0, int *sv);

DESCRIPTION
       The PF_UNIX (also known as PF_LOCAL) socket family is used to  communi-
       cate  between  processes  on the same machine efficiently. Unix sockets
       can be either anonymous (created by socketpair(2)) or associated with a
       file  of  type socket.  Linux also supports an abstract namespace which
       is independent of the file system.

       Valid  types  are:  SOCK_STREAM,  for  a  stream-oriented  socket  and
       SOCK_DGRAM,  for  a  datagram-oriented  socket  that  preserves message
       boundaries (as on most Unix implementations, Unix domain datagram sock-
...
```

Páternoszter socketes példák

Szerver oldal

```
#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);
  // ts -> Sun Mar  9 19:36:56 2008
  return write(kliens, ts, strlen(ts));
}
```

Kliens oldal

```
$ ./kliens
Sun Mar  9 19:36:56 2008
```

Lokális TCP socketek használata, szerver oldal

```
#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

int
main (void)
{
  int kapu_figyelo,
  struct sockaddr_u
  memset ((void *)
  szerver.sun_famil
  strncpy (szerver.
           sizeof (
  if ((kapu_figyelo
    {
      perror ("sock
      exit (EXIT_FA
    }
  if (bind (ka
          siz
    {
      perror (
      exit (EX
    }
  if (listen (
    {
      perror (
      exit (EX
    }
```

1 2 3 4

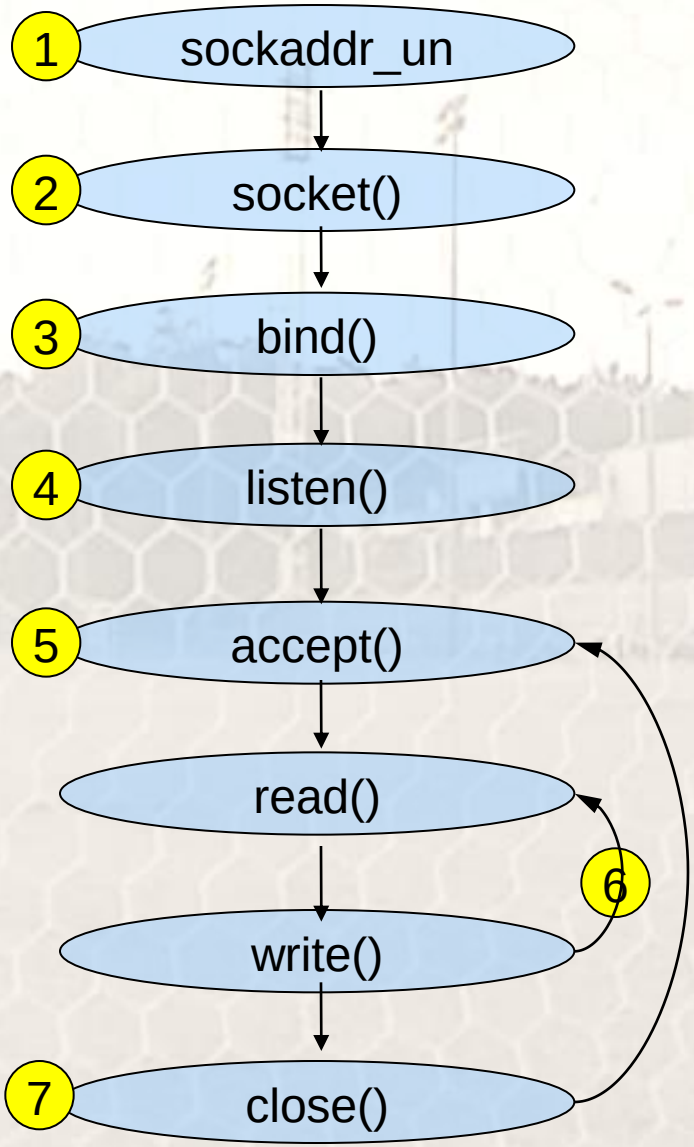
```
  for (;;)
    {
      memset ((void *) &kliens, 0, (kliensm = sizeof (kliens)));
      if ((kapcsolat = accept (kapu_figyelo,
                               (struct sockaddr *) &kliens,
                               (socklen_t *) & kliensm)) == -1)
        {
          perror ("accept");
          exit (EXIT_FAILURE);
        }
      if (kiszolgal (kapc
        {
          perror ("kiszolgal");
```

5 6

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

```
$ ls -l szerver.socket
srwxr-xr-x 1 norbi norbi 0 Mar 10 13:12 szerver.socket

$ ps -Ao pid,comm|grep szerver
24745 szerver
$ ls -l /proc/24745/fd/
total 0
lrwx------ 1 norbi norbi 64 Mar 10 13:15 0 -> /dev/pts/1
lrwx------ 1 norbi norbi 64 Mar 10 13:15 1 -> /dev/pts/1
lrwx------ 1 norbi norbi 64 Mar 10 13:15 2 -> /dev/pts/1
lrwx------ 1 norbi norbi 64 Mar 10 13:15 3 -> socket:[23426205]
```

Lokális TCP socketek használata, szerver oldal



Lokális TCP socketek használata, kliens oldal

```
#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_M
    write (1, buffer, olvasva);
  exit (EXIT_SUCCESS);
}
```

1

2

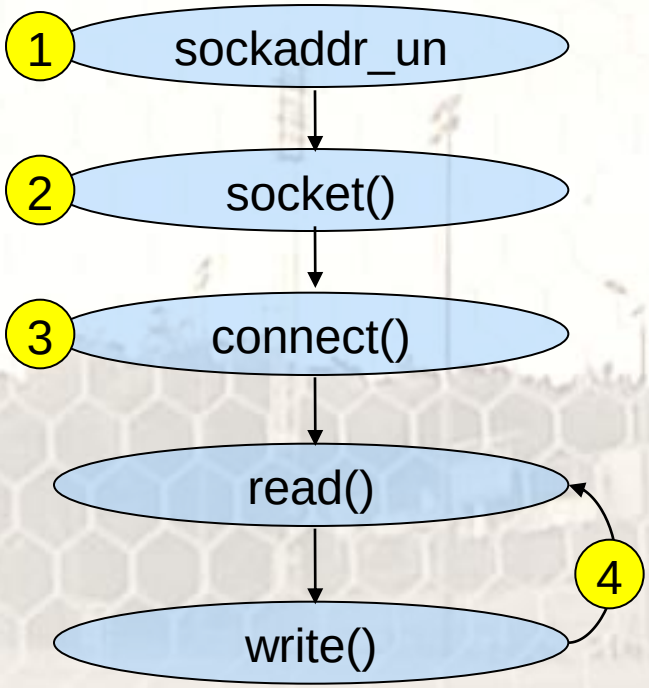
3

4

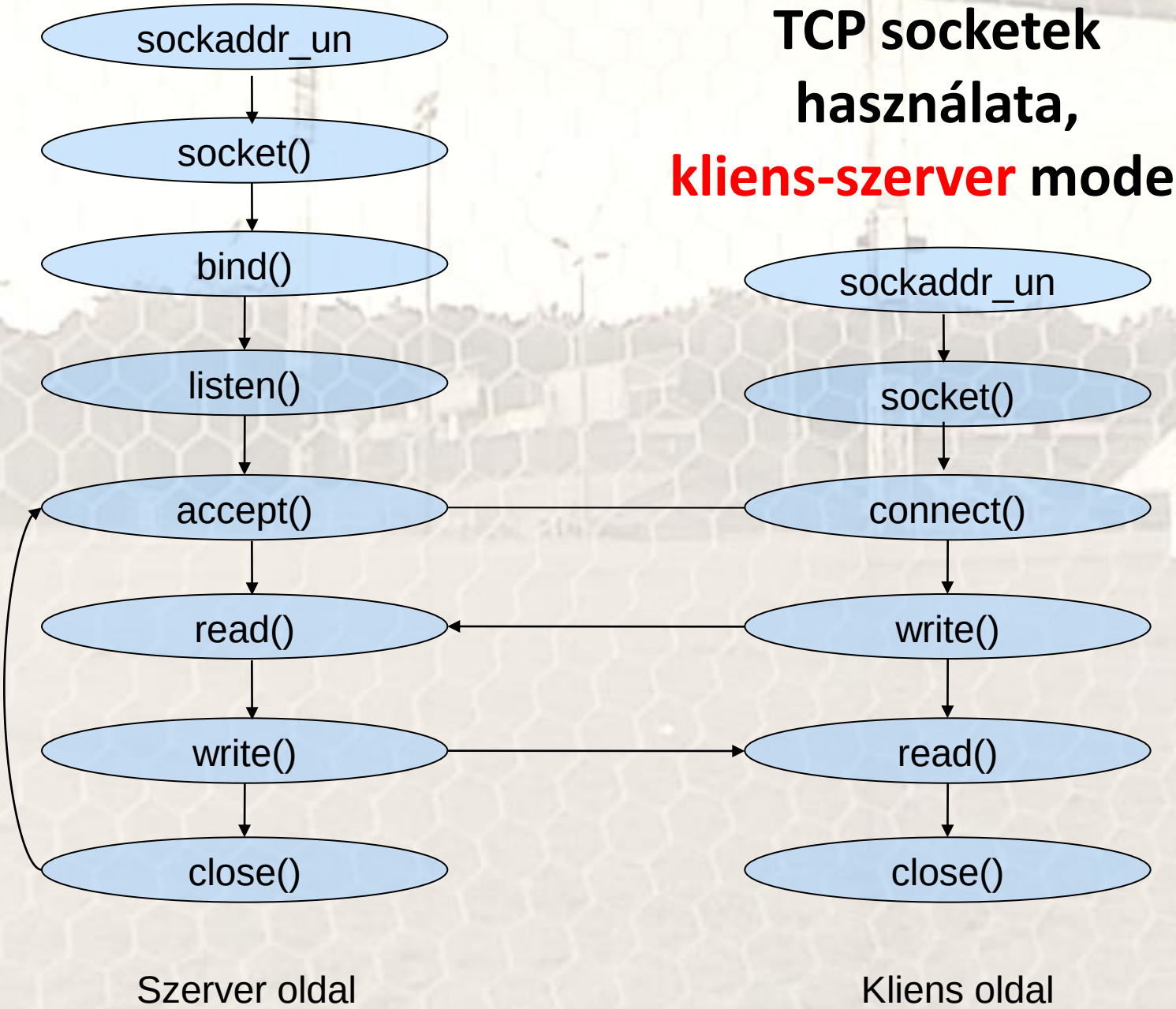
read/write

```
$ gcc -o kliens kliens.c
$ ./kliens
Mon Mar 10 13:14:18 2008
```

Lokális TCP socketek használata, kliens oldal



Lokális TCP socketek használata, kliens-szerver modell



BIND(2) Linux Programmer's Manual BIND(2)

LISTEN(2) Linux Programmer's Manual LISTEN(2)

ACCEPT(2) Linux Programmer's Manual ACCEPT(2)

CONNECT(2) Linux Programmer's Manual CONNECT(2)

NAME

connect - initiate a connection on a socket

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t
addrlen);
```

DESCRIPTION

The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by serv_addr. The addrlen argument specifies the size of serv_addr. The format of the address in serv_addr is determined by the address space of the socket sockfd; see socket(2) for further details.

If the socket sockfd is of type SOCK_DGRAM then serv_addr is the address to which datagrams are sent by default, and the only address from which datagrams are received. If the socket is of type SOCK_STREAM or SOCK_SEQPACKET, this call attempts to make a connection

...

Lokális UDP socketek használata, 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>
#include <sys/un.h>
#define SZERVER_PORT 2005
#define BUFFER_MEPET 64
int
main (void)
{
  int kapu_figyel
  char buffer[BUF
  time_t t;
  struct sockaddr
  memset ((void *
  szerver.sun_fam
  strncpy (szerve
           sizeof
  if ((kapu_figye
    {
      perror ("so
      exit (EXIT_
    }
  if (bind (kapu_
          sizeof
    {
      perror ("bi
      exit (EXIT_
    }
```

```
  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);
        }
    }
}
```

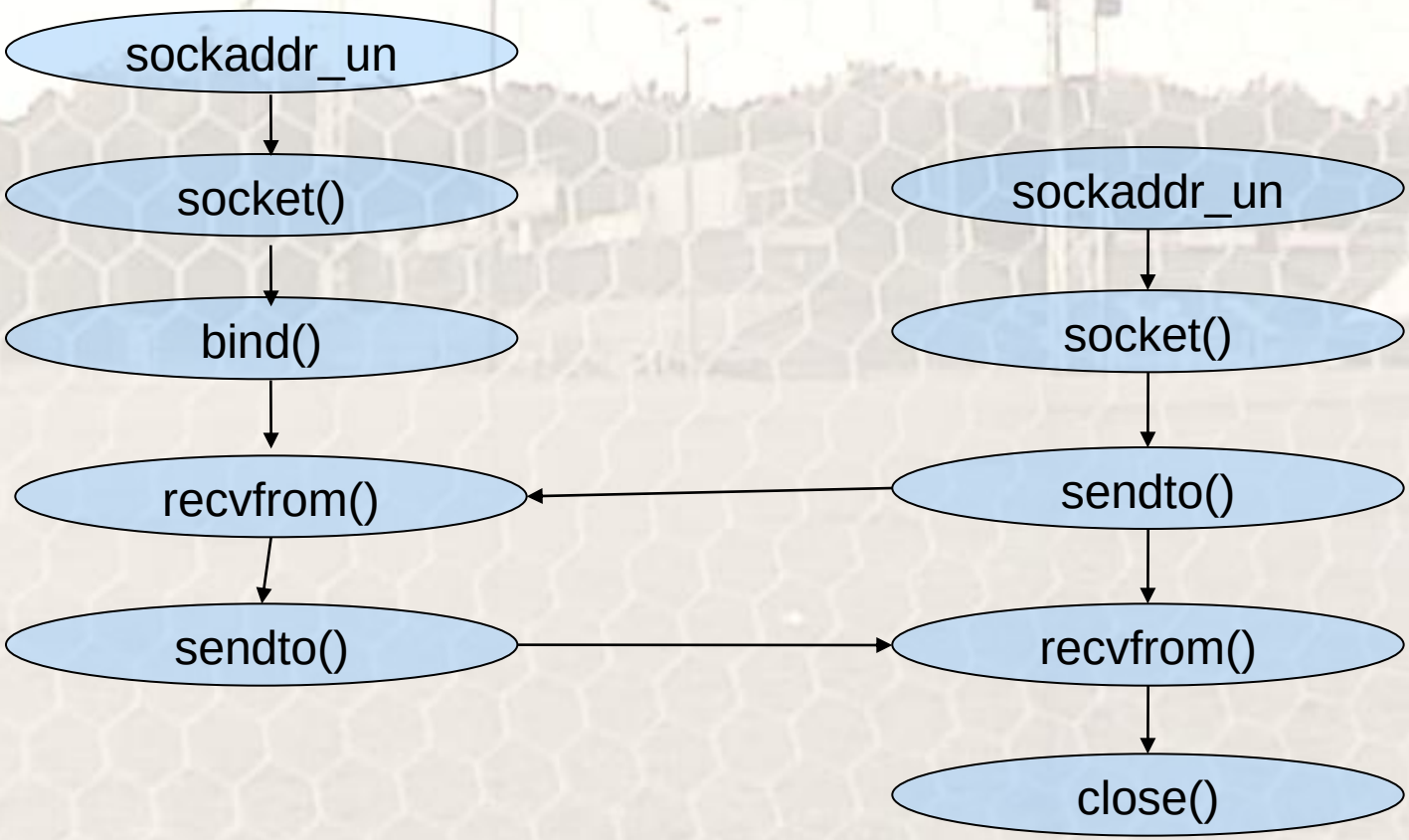
```
$ gcc szerver.c -o szerver
$ ./szerver
```

Lokális UDP socketek használata, kliens oldal

```
#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, olvas
  struct sockaddr
  char buffer[BUF
  memset ((void *
  szerver.sun_fam
  strncpy (szerve
           sizeof
  if ((kapu = soc
    {
      perror ("so
      exit (EXIT_
    }
  memset ((void *
  kliens.sun_fami
  strncpy (kliens
           sizeof
  if (bind (kapu,
      == -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);
  unlink("kliens.socket");
  exit (EXIT_SUCCESS);
}
```

```
$ gcc kliens.c -o kliens
$ ./kliens
Mon Mar 10 14:23:21 2008
```

Lokális UDP socketek használata, kliens-szerver modell



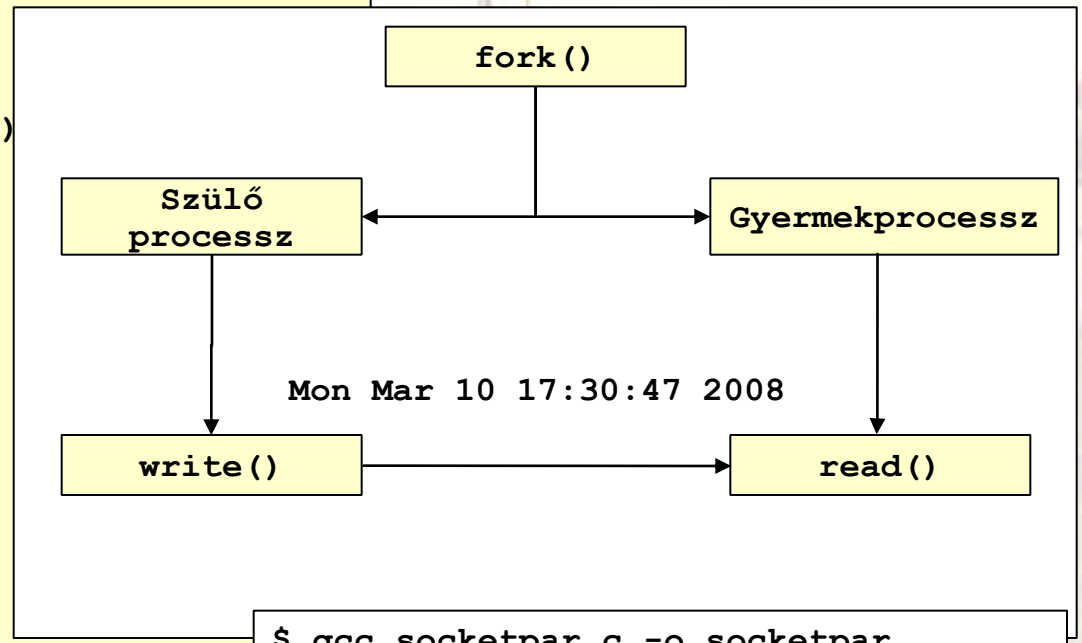
Anonim socketek

```
SOCKETPAIR(2)              Linux Programmer's Manual              SOCKETPAIR(2)

NAME
       socketpair - create a pair of connected sockets

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>

       int socketpair(int d, int type, int protocol, int sv[2]);

DESCRIPTION
       The  socketpair()  call creates an unnamed pair of connected sockets in
       the specified domain d, of the specified type, and using the optionally
       specified  protocol.  The descriptors used in referencing the new sock-
       ets are returned in sv[0] and sv[1].  The  two  sockets  are  indistin-
       guishable.

RETURN VALUE
       On  success,  zero is returned.  On error, -1 is returned, and errno is
       set appropriately.

...
```

4.2BSD

Anonim socketek

```
#include <unistd.h>
#include <sys/socket.h>
#include <time.h>
#include <string.h>
#define BUFFER_MERET 256
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,
      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;
}
```




```
$ gcc socketpar.c -o socketpar
$ ./socketpar
Mon Mar 10 17:30:47 2008
```

Csővezetékek

```
SOCKETPAIR(2)              Linux Programmer's Manual              SOCKETPAIR(2)

NAME
       socketpair - create a pair of connected sockets

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>

       int socketpair(int d, int type, int protocol, int sv[2]);

DESCRIPTION
       The  socketpair()  call creates an unnamed pair of connected sockets in
       the specified domain d, of the specified type, and using the optionally
       specified  protocol.  The descriptors used in referencing the new sock-
       ets are returned in sv[0] and sv[1].  The  two  sockets  are  indistin-
       guishable.

RETURN VALUE
       On  success,  zero is returned.  On error, -1 is returned, and errno is
       set appropriately.

...
```

4.2BSD

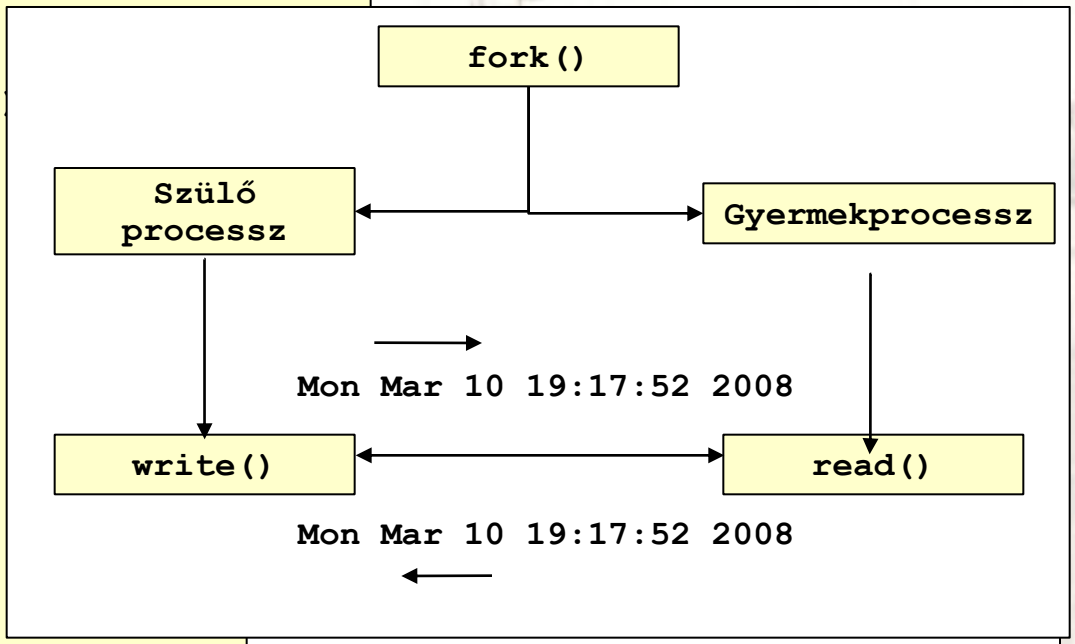
Csővezetékek

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#define BUFFER_MERET 256
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]);
    }
  ...
```

```
$ gcc pipe.c -o pipe
$ ./pipe
Mon Mar 10 19:15:33 2008
```

Anonim socketek vs. csővezetékek

```
...
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,
      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,
      write (1, buffer, olvasva);
      close (sv[1]);
    }
  ...
```

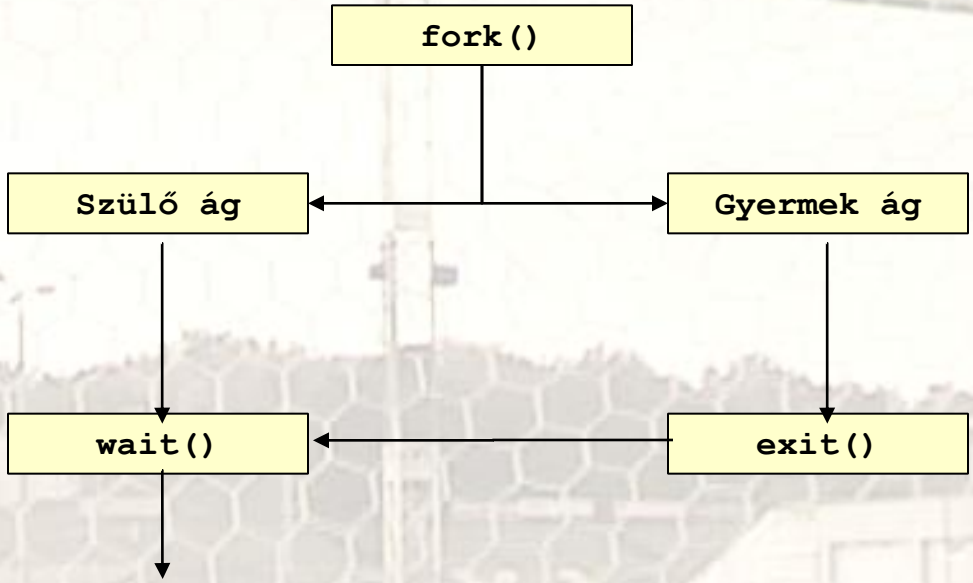



```
$ gcc socketpar2.c -o socketpar2
$ ./socketpar2
Mon Mar 10 19:17:52 2008
Mon Mar 10 19:17:52 2008
```

Ismétlés: a fork() tipikus használata

PP 40

```
#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;
}
```

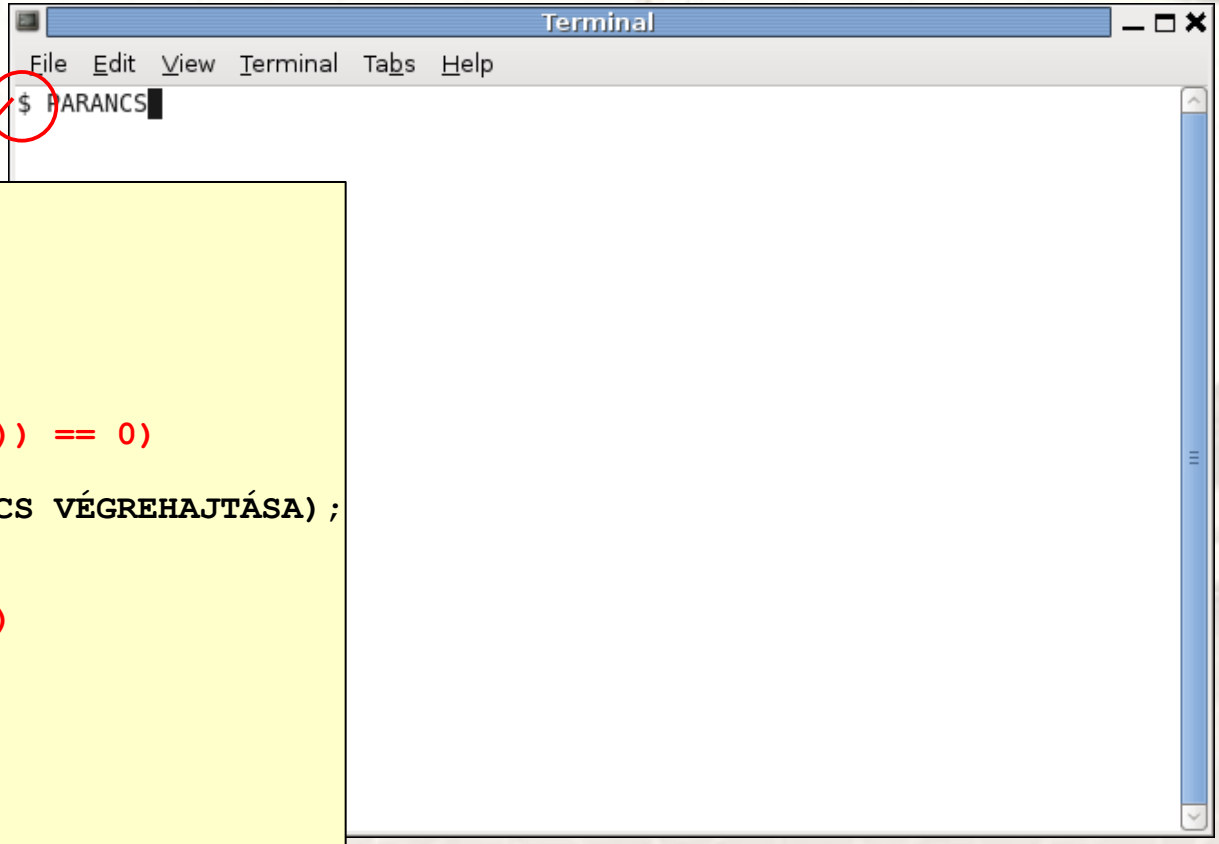


A gyermek folyamat elvégzi az ls parancs futtatását, majd kilép (befejeződik)

Közben a szülő folyamat várakozik a gyermek befejeződésére

Miért jó ez?

Ismétlés: a parancssor




```
...
for(;;)
{
  prompt_kiírása("$");
  parancssor_beolvasása();

  if((gyermekem_pid = fork()) == 0)
  {
    execve(BEOLVASOTT PARANCS VÉGREHAJTÁSA);
    exit(0);
  }
  else if(gyermekem_pid > 0)
  {
    wait(&statusz);
  }
  else
  {
   exit(-1);
  }
}
...
```

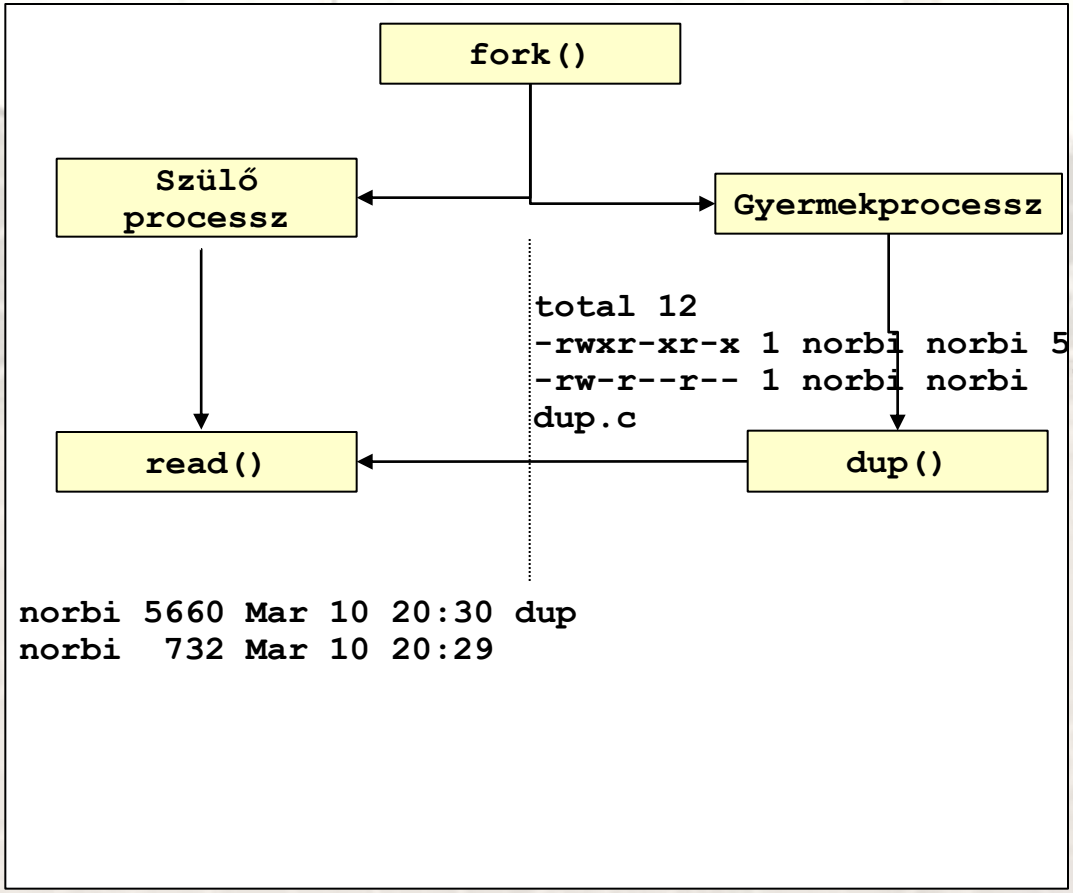
A parancsértelmező (shell) szervezése

```
DUP(2)                    Linux Programmer's Manual                    DUP(2)

NAME
       dup, dup2 - duplicate a file descriptor

SYNOPSIS
       #include <unistd.h>

       int dup(int oldfd);
       int dup2(int oldfd, int newfd);

DESCRIPTION
       dup() and dup2() create a copy of the file descriptor oldfd.

       After  a  successful  return from dup() or dup2(), the old and new file
       descriptors may be used interchangeably.  They refer to the  same  open
       file description (see open(2)) and thus share file offset and file sta-
       tus flags; for example,  if  the  file  offset  is  modified  by  using
       lseek(2)  on one of the descriptors, the offset is also changed for the
       other.

       The two descriptors do not share file descriptor flags  (the  close-on-
       exec  flag).  The close-on-exec flag (FD_CLOEXEC; see fcntl(2)) for the
...

       dup() uses the lowest-numbered unused descriptor for the  new  descrip-
       tor.

...
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#define BUFFER_MERET 256
int
main (void)
{
  int gyermekem_
  int sv[2];
  if (pipe (sv) =
    {
      perror ("pi
      exit (EXIT_
    }
```

```c
  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;
}
```

```
$ gcc dup.c -o dup
$ ./dup
total 12
-rwxr-xr-x 1 norbi norbi 5660 Mar 10 20:30 dup
-rw-r--r-- 1 norbi norbi  732 Mar 10 20:29 dup.c
```




```
total 12
-rwxr-xr-x 1 norbi norbi 5660 Mar 10 20:30 dup
-rw-r--r-- 1 norbi norbi  732 Mar 10 20:29
dup.c
```

Internet domain socketek

```
IP(7)                       Linux Programmer's Manual                       IP(7)

NAME
       ip - Linux IPv4 protocol implementation

SYNOPSIS
       #include <sys/socket.h>
       #include <netinet/in.h>
       #include <netinet/ip.h> /* superset of previous */

       tcp_socket = socket(PF_INET, SOCK_STREAM, 0);
       udp_socket = socket(PF_INET, SOCK_DGRAM, 0);
       raw_socket = socket(PF_INET, SOCK_RAW, protocol);

DESCRIPTION
       Linux implements the Internet Protocol, version 4, described in RFC 791
       and RFC 1122.  ip contains a level 2 multicasting  implementation  con-
       forming  to RFC 1112.  It also contains an IP router including a packet
       filter.

       The programming interface is BSD sockets compatible.  For more informa-
       tion on sockets, see socket(7).

...
```

TCP socketek, 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
#define SZERVER_SOR_M
int
main(void)
{
  int kapu_figyelo, k
  struct sockaddr_in
  memset((void *)&sze
  szerver.sin_family
  inet_aton("127.0.0.
  szerver.sin_port =
  if((kapu_figyelo =
    == -1 )
  {
    perror("socket");
    exit(EXIT_FAILURE
  }
  setsockopt(kapu_fig
    (void *)&sockoptv
  if(bind(kapu_figyel
    sizeof(szerver))
  {
    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);
  }
}
```

A szerver futtatása és tesztelése

```
$ gcc szerver.c -o szerver
$ ./szerver
127.0.0.1:2005
```

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

```
$ gcc szerver.c -o szerver
$ ./szerver
0.0.0.0:2005
```

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

```
$ gcc szerver.c -o szerver
127.0.0.1:2005
     <-> 127.0.0.1:48625
     <-> 127.0.0.1:48626
     <-> 127.0.0.1:48627
     <-> 127.0.0.1:48628
     <-> 127.0.0.1:48629
```

Szerver oldal

```
$ telnet localhost 2005
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Tue Mar 11 13:07:47 2008
Connection closed by foreign host.
$ telnet localhost 2005
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Tue Mar 11 13:07:47 2008
Connection closed by foreign host.
```

Kliens oldal

A szerver futtatása és tesztelése

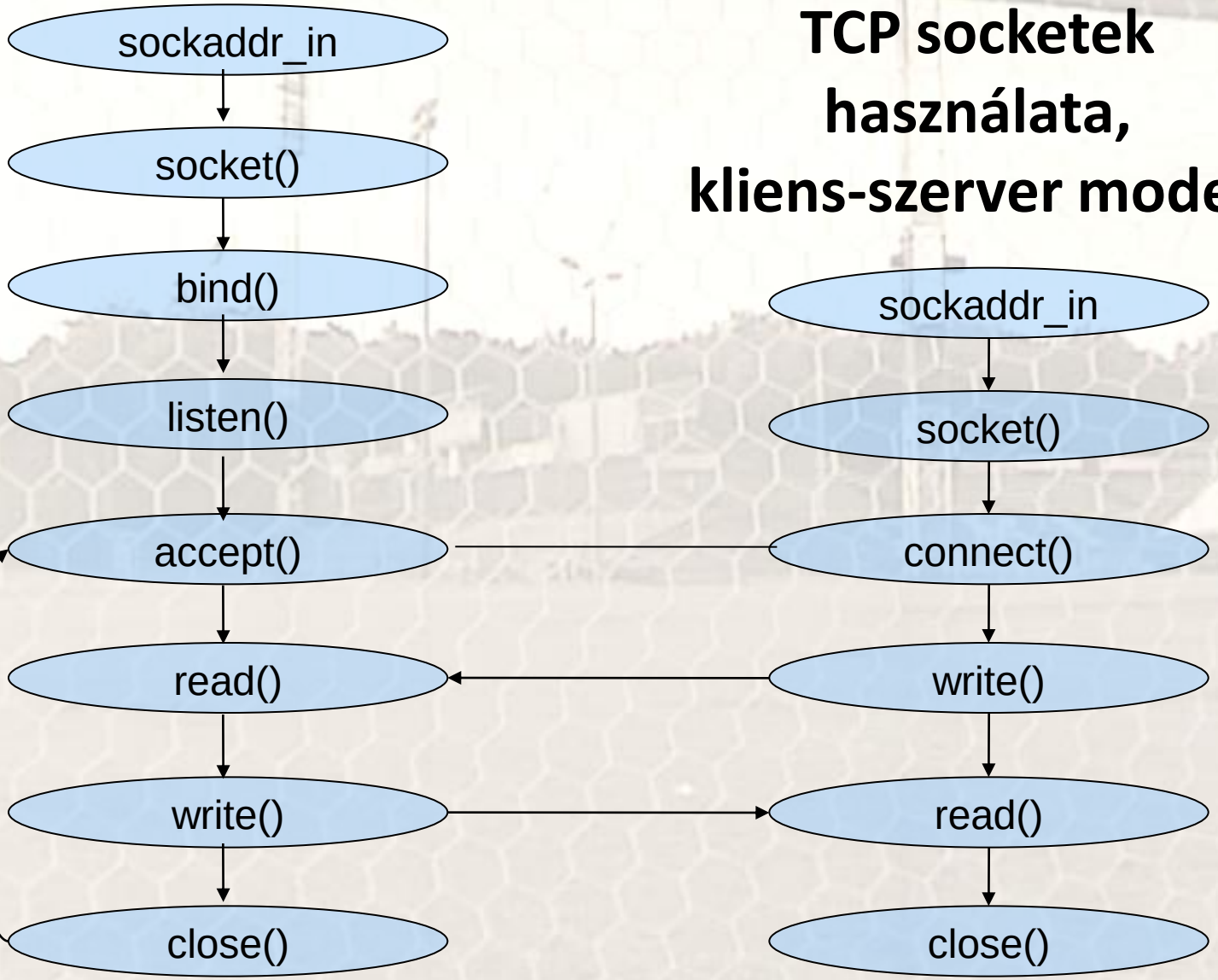
```
$ netstat -tapi
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address          State PID/Program name

...
tcp        0      0 *:2005                  *:*                      LISTEN      13206/szerver
...
tcp        0      0 *:2005                  *:*                      LISTEN      13206/szerver
tcp        0      0 szerver.inf.unideb.:2005 kliens.inf.unideb.:2662 ESTABLISHED 13206/szerver
...
```

```
$ ./szerver
0.0.0.0:2005
     <-> 172.17.135.156:2662
```

TCP socketek, kliens oldal

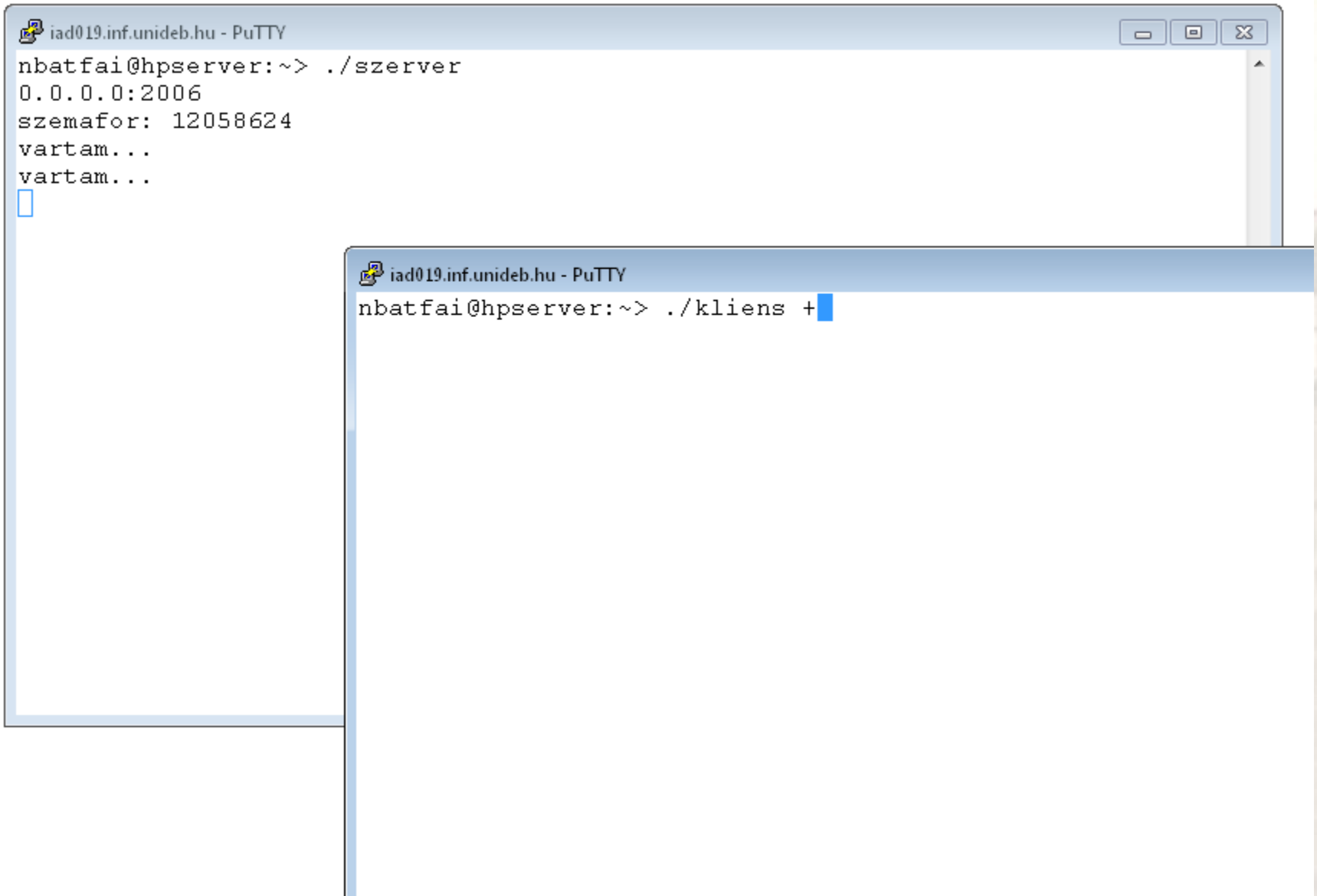
```
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define SZERVER_PORT 2005
#define BUFFER_MERET 256
int
main (void)
{
  int kapu, olvasva;
  struct sockaddr_in szerver;
  char buffer[BUFFER_MERET];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

Internet domain TCP socketek használata, kliens-szerver modell



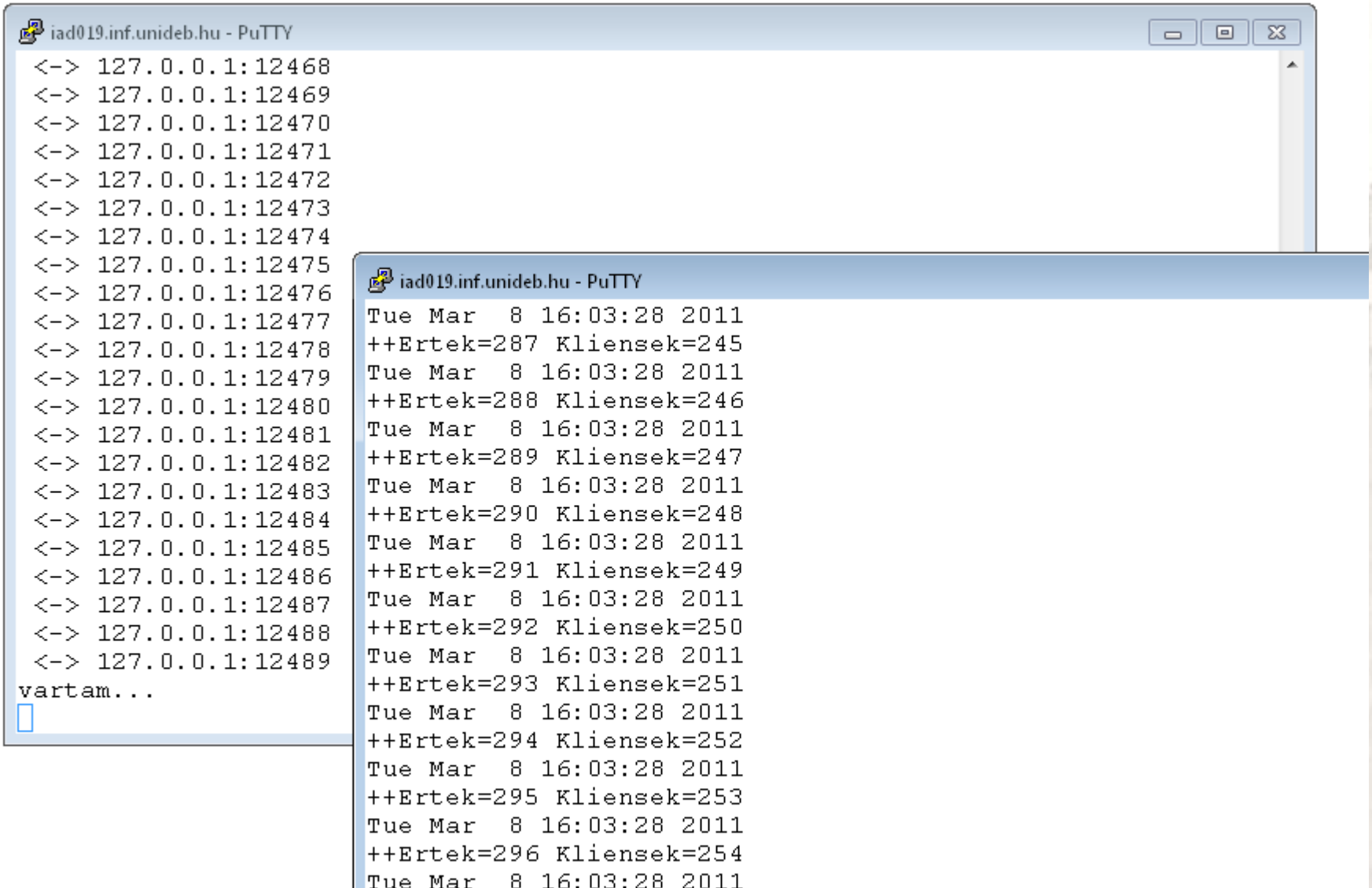
Labor

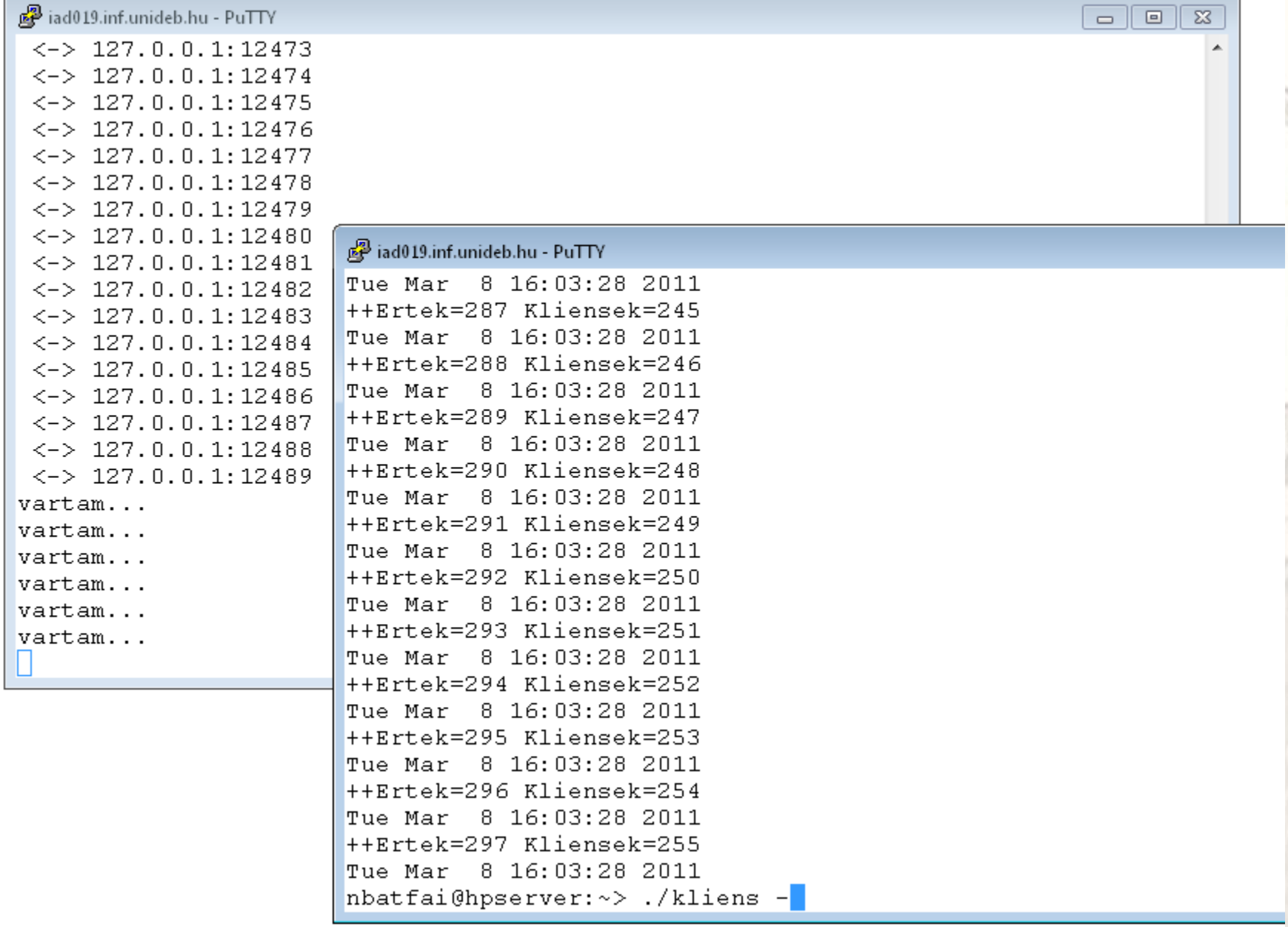
http://progpater.blog.hu/2011/03/06/halozati_vegyertek

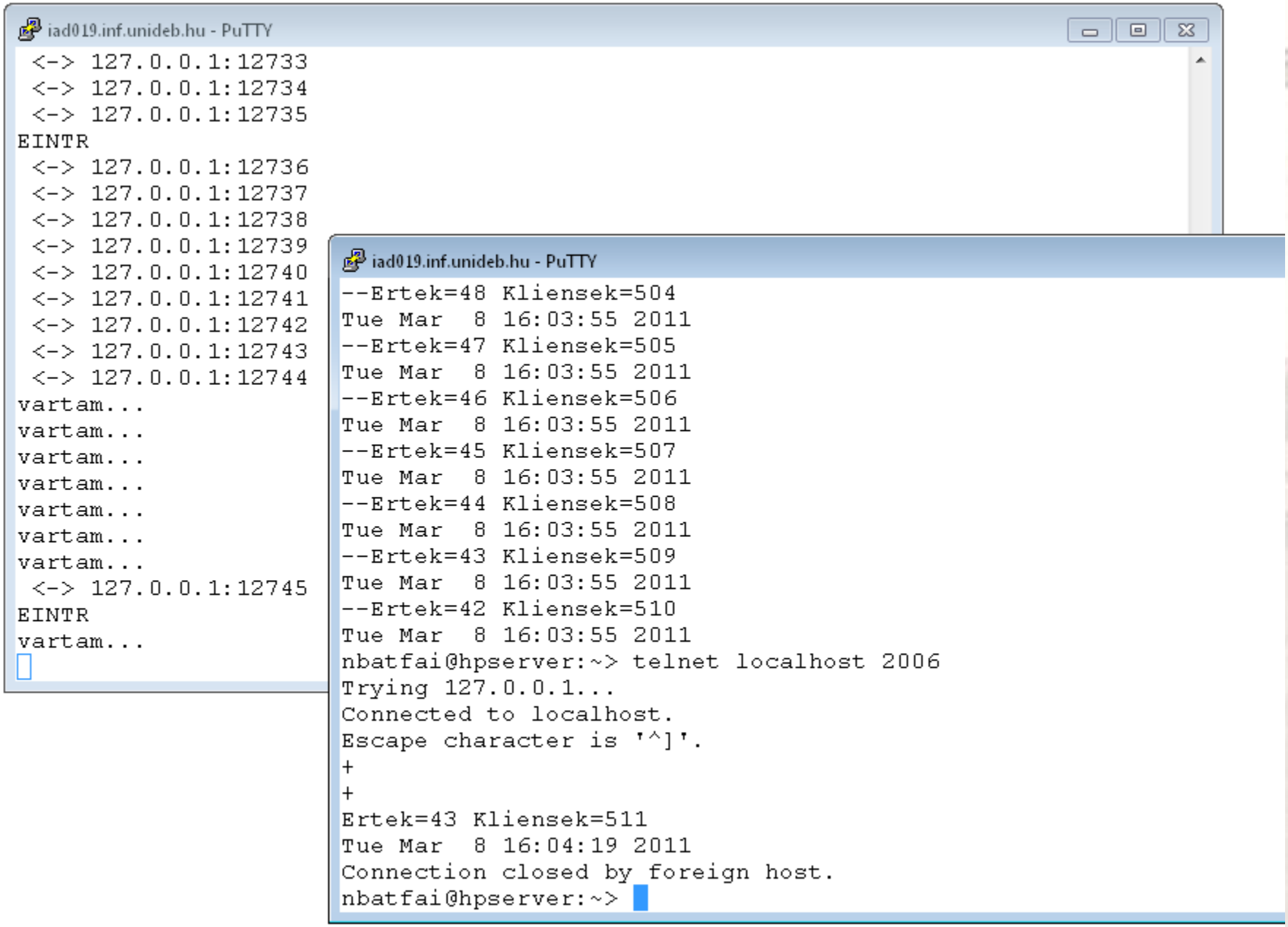


Labor

http://progpater.blog.hu/2011/03/06/halozati_vegyertek




iad019.inf.unideb.hu - PuTTY
<-> 127.0.0.1:12473
<-> 127.0.0.1:12474
<-> 127.0.0.1:12475
<-> 127.0.0.1:12476
<-> 127.0.0.1:12477
<-> 127.0.0.1:12478
<-> 127.0.0.1:12479
<-> 127.0.0.1:12480
<-> 127.0.0.1:12481
<-> 127.0.0.1:12482
<-> 127.0.0.1:12483
<-> 127.0.0.1:12484
<-> 127.0.0.1:12485
<-> 127.0.0.1:12486
<-> 127.0.0.1:12487
<-> 127.0.0.1:12488
<-> 127.0.0.1:12489
vartam...
vartam...
vartam...
vartam...
vartam...
vartam...
iad019.inf.unideb.hu - PuTTY
Tue Mar 8 16:03:28 2011
++Ertek=287 Kliensek=245
Tue Mar 8 16:03:28 2011
++Ertek=288 Kliensek=246
Tue Mar 8 16:03:28 2011
++Ertek=289 Kliensek=247
Tue Mar 8 16:03:28 2011
++Ertek=290 Kliensek=248
Tue Mar 8 16:03:28 2011
++Ertek=291 Kliensek=249
Tue Mar 8 16:03:28 2011
++Ertek=292 Kliensek=250
Tue Mar 8 16:03:28 2011
++Ertek=293 Kliensek=251
Tue Mar 8 16:03:28 2011
++Ertek=294 Kliensek=252
Tue Mar 8 16:03:28 2011
++Ertek=295 Kliensek=253
Tue Mar 8 16:03:28 2011
++Ertek=296 Kliensek=254
Tue Mar 8 16:03:28 2011
++Ertek=297 Kliensek=255
Tue Mar 8 16:03:28 2011
nbatfai@hpserver:~> ./kliens -


iad019.inf.unideb.hu - PuTTY
<-> 127.0.0.1:12733
<-> 127.0.0.1:12734
<-> 127.0.0.1:12735
EINTR
<-> 127.0.0.1:12736
<-> 127.0.0.1:12737
<-> 127.0.0.1:12738
<-> 127.0.0.1:12739
<-> 127.0.0.1:12740
<-> 127.0.0.1:12741
<-> 127.0.0.1:12742
<-> 127.0.0.1:12743
<-> 127.0.0.1:12744
vartam...
vartam...
vartam...
vartam...
vartam...
vartam...
vartam...
<-> 127.0.0.1:12745
EINTR
vartam...
iad019.inf.unideb.hu - PuTTY
--Ertek=48 Kliensek=504
Tue Mar 8 16:03:55 2011
--Ertek=47 Kliensek=505
Tue Mar 8 16:03:55 2011
--Ertek=46 Kliensek=506
Tue Mar 8 16:03:55 2011
--Ertek=45 Kliensek=507
Tue Mar 8 16:03:55 2011
--Ertek=44 Kliensek=508
Tue Mar 8 16:03:55 2011
--Ertek=43 Kliensek=509
Tue Mar 8 16:03:55 2011
--Ertek=42 Kliensek=510
Tue Mar 8 16:03:55 2011
nbatfai@hpserver:~> telnet localhost 2006
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+
+
Ertek=43 Kliensek=511
Tue Mar 8 16:04:19 2011
Connection closed by foreign host.
nbatfai@hpserver:~>

Laborkártyák

```
int
kiszolgal (int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time (NULL);
  char *ts = ctime_r (&t, buffer);
  char buffer2[100] = "";

  int olvasott = read(kliens, buffer2, 100);
  write(kliens, buffer2 , olvasott);

  return write (kliens, ts, strlen (ts));
}
```

Ha így módosítod a PP 126-on kezdődő multiplexelt párhuzamos szerverét, akkor mit ír ki, ha futtatod? (beugratós)

Hogy tudod egyáltalán tesztelni a kiszolgal() fgv-beli módosítást?

Laborkártyák

```
int
kiszolgal (int kliens)
{
  char buffer[CTIME_BUFFER_MERET] = "";
  time_t t = time (NULL);
  char *ts = ctime_r (&t, buffer);
  char buffer2[10] = "";

  int olvasott = read(kliens, buffer2, 10);
  write(kliens, buffer2 , olvasott);

  olvasott = read(kliens, buffer2, 10);
  write(kliens, buffer2 , olvasott);

  return write (kliens, ts, strlen (ts));
}
```

Ha így módosítod a PP 126-on kezdődő multiplexelt párhuzamos szerverét, akkor mit ír ki, ha egy TCP-s klienstől ezt kapja:
0123456789012345678 9

Laborkártyák

Mi a *char *ctime_r(const time_t *timep, char *buf);* 2. paramétere?

Mit jelent az, hogy egy fgv. *reentráns*?

C nyelven a hívott fgv. aktuális paraméterei milyen sorrendben kerülnek a verembe?

Mi az *errno*?

Laborkártyák

A PP 126-on kezdődő progiban, ha a szülő klózolja a kapcsolatot, az milyen hatással lehet a klienssel kommunikáló gyerekfolyamatra?

```
            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);
```

Laborkártyák

Mi az alábbi csipetben az *olvaso* és mi az *iro*?

```
int
main (void)
{
  pthread_t sz[SZALAK_SZAMA];
  int s[SZALAK_SZAMA], *r, i;

  sem_init (&adatb, 0, 1);

  for (i = 0; i < SZALAK_SZAMA; ++i)
    {
      s[i] = i;
      if (pthread_create (&sz[i], NULL,
                          (i < SZALAK_SZAMA / 5) ? olvaso : iro,
                          (void *) &s[i]))
        {
          perror ("Hiba");
          exit (-1);
        }
    }
  for (i = 0; i < SZALAK_SZAMA; ++i)
    {
      pthread_join (sz[i], (void *) &r);
    }
  sem_destroy (&adatb);
}
```

Laborkártyák

Mi történik, ha „futtatod”? **Tgyfh.: egy olvasó most éppen olvas!**

```
int olvaso_szamlalo = 0;
Semaphore S_mutex = 1;
Semaphore S_adatb = 1;
```

```
// 4 írónk van
Író() {
        for(;;) {

DOWN(S_adatb);
                // ír 2 mp-ig
                UP(S_adatb);
        }
}
```

```
// 8 olvasónk van
Olvasó() {
        // 4 másodpercenként jön egy
        // olvasó az alábbi kód szerint
        // olvasni
        for(;;) {
                DOWN(S_mutex);
                ++olvaso_szamlalo;
                if(olvaso_szamlalo == 1)
                        DOWN(S_adatb);
                UP(S_mutex);
                // olvasok 8 mp-ig
                DOWN(S_mutex);
                --olvaso_szamlalo;
                if(olvaso_szamlalo == 0)
                        UP(S_adatb);
                UP(S_mutex);
        }
}
```

Visszatekintő laborkártyák

```c
#include <stdio.h>
#include <unistd.h>
int
main (void)
{
  // hányadik betűn állok?
  int hanyadik_betu = -1;
  // azon a helyen mit olvastam?
  int elso = 0, masodik = 0, harmadik = 0, i = 0, jegy = 0;

  while ((i = getchar ()) != EOF)
    {

      switch (i)
        {

        case 'T':
          jegy = 0;
          break;
        case 'C':
          jegy = 1;
          break;
        case 'A':
          jegy = 2;
          break;
        case 'G':
          jegy = 3;
          break;
        }

      hanyadik_betu = (hanyadik_betu + 1) % 3;

      if (!hanyadik_betu)
        {
          elso = jegy;
          printf ("\na triplet elso betuje a %c volt\n", i);
        }
      else if (!(hanyadik_betu - 1))
        {
          masodik = jegy;
          printf ("a triplet masodik betuje a %c volt\n", i);
        }
      else
        {
          harmadik = jegy;
          printf ("a triplet harmadik betuje a %c volt\n", i);
        }
    }
}
```

Hogy működik?

http://progpater.blog.hu/2011/03/06/az_otodik_labor aminoosavas progijából átalakítva

Visszatekintő laborkártyák: feltételes fordítás

```
SYSCONF(3)                  Linux Programmer's Manual                  SYSCONF(3)

NAME
       sysconf - Get configuration information at runtime

SYNOPSIS
       #include <unistd.h>

       long sysconf(int name);

DESCRIPTION
       POSIX  allows  an  applic
       certain options are suppo
       urable constants or limit
```

```
#include <stdio.h>
#include <unistd.h>
#include <limits.h>
int
main (void)
{
#ifdef OPEN_MAX
#ifdef DEBUG
  printf ("OPEN_MAX definialt\n");
#endif
  printf ("OPEN_MAX=%d\n", OPEN_MAX);
#else
#ifdef DEBUG
  printf ("OPEN_MAX nem definialt\n");
#endif
  printf ("OPEN_MAX=%d\n", sysconf (_SC_OPEN_MAX));
#endif
  return 0;
}
```

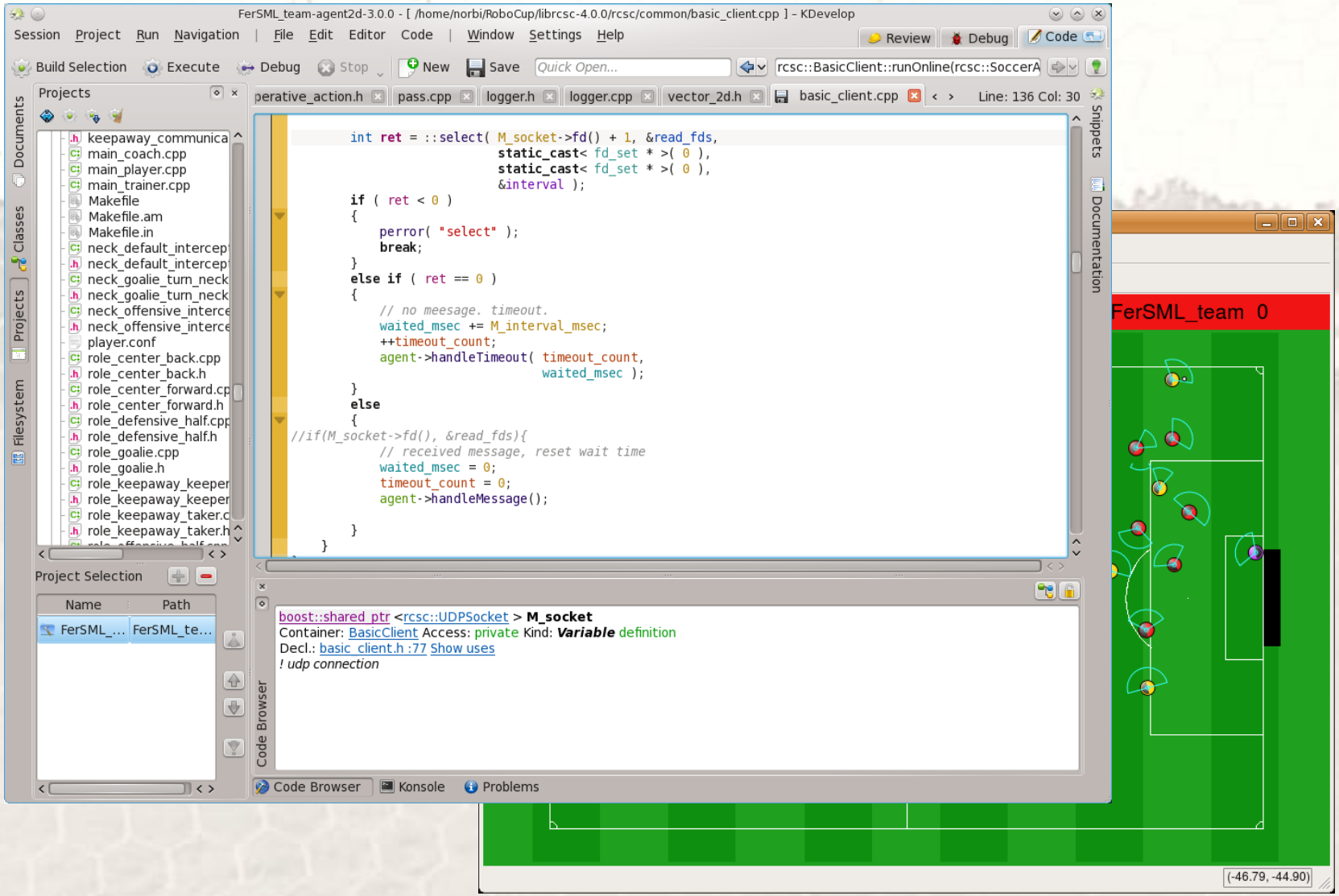
Melyik nyomta ki?

```
$ gcc openmax.c -o openmax
$ ./openmax
OPEN_MAX=1024
$ gcc -DDEBUG openmax.c -o openmax
$ ./openmax
OPEN_MAX nem definialt
OPEN_MAX=1024
```

```
       OPEN_MAX - _SC_OPEN_MAX
              The  maximum number of files that a process can have open at any
              time.  Must not be less than _POSIX_OPEN_MAX (20).
...
```

Otthoni opcionális feladat

A robotfoci japán szoftvereinek (librcsc, agent2d) tanulmányozása a KDevelop-ban.



Kötelező olvasmány

Az MR könyvből olvashatsz részletesebben a fóliákon megismert rendszerhívásokról (és persze a hivatkozott manuál lapokról).