## Név nélküli csővezeték (pipe)

A pipe (csővezeték) egy egyszerű kommunikációs eszköz, amelynek segítségével egyszerűen tudjuk az információ átadást megvalósítani "családtagok" között (szülő-gyerek, gyerek-gyerek, közös őssel rendelkező folyamatok között). A csővezeték a felhasználó számára egy speciális név nélküli állomány formájában jelenik meg, amelyet a pipe függvényhívás hatására a rendszer hoz létre, és a végén a rendszer törli le. Úgy működik, mint egy sor (FIFO), azaz a beírás sorrendjében tudjuk kiolvasni az adatokat. A csővezeték két végét, két fájlmutatóval lehet elérni, az egyikkel az író, a másikkal az olvasó végét. A kiolvasott adatok törlődnek a csővezetékből. Ha kiürül a csővezeték, akkor az olvasó folyamat, ha megtelik, akkor az író folyamat várakozik. Ha a csővezeték írható végeit bezárjuk, akkor az olvasó végére fájlvége jel megy, ha az olvasó végeket zárjuk be, nem lehet bele írni.

```
#include <unistd.h>
int pipe(int filedes[2]);
```

#### Paraméterek:

- filedes[2]: egy olyan memóriacím, ahova 2 darab fájlleírót beírhatja a rendszer (tehát legyen ekkora memória lefoglalva)
  - filedes[0]: fájlleíró olvasásrafiledes[1]: fájlleíró írásra

## Visszatérési érték:

- siker esetén 0
- hiba esetén -1

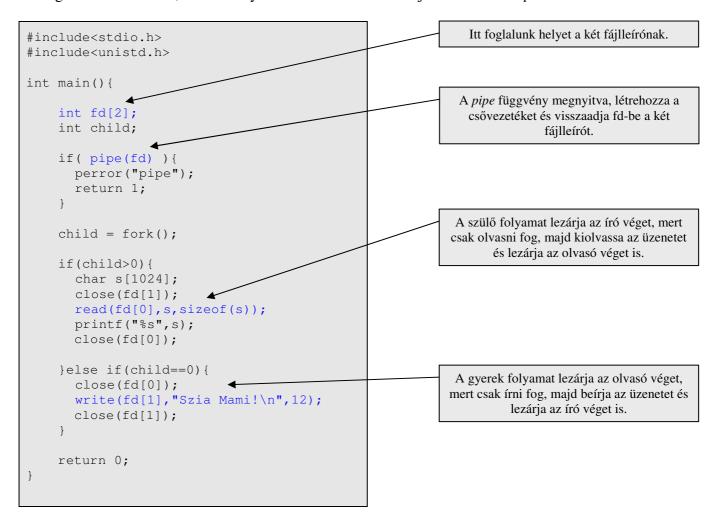
### Hibák:

- EFAULT: érvénytelen filedes
- EMFILE: túl sok fájlleírót használ a folyamat
- ENFILE: túl sok megnyitott fájl van a rendszerben

A csővezetéket a szülő folyamat szokta létrehozni, majd a *fork* függvényhívás után a létrejövő gyerek folyamatban a csővezeték két fájl leírójának a másolata lesz, azaz a fájlleírók megduplázódnak, a csővezetéknek "két" eleje és "két" vége lesz. A folyamatok általában bezárják a csővezetéknek azt a végét, amelyiket nem használják.

Több író esetén is a beírás sorrendjében jelenik meg az olvasóknál az adat (FIFO). Több olvasó esetén, figyelembe kell venni, hogy egy adatot csak egyszer lehet kiolvasni. Ha az egyik olvasónál kiolvasásra került egy adat, akkor nem lehet még egyszer kiolvasni. Több olvasó véget akkor használhatunk, ha párhuzamos folyamatok dolgozzák fel a csővezeték adatait, és az olvasó folyamatoknak a bent lévő adatok közül a legrégebbit kell kiolvasni. Több írható véget, pedig akkor, ha párhuzamos folyamatok állítják elő az adatokat.

Nézzünk egy rövid példát. A szülő folyamat létrehoz egy csővezetéket, a gyerek folyamat beleír egy szöveget a csővezetékbe, a szülő folyamat ezt kiolvassa és kiírja a standard *output*-ra.



## Javasolt feladat:

Egy "termelő" folyamat állítson elő egyszámjegyű véletlen számokat (*rand()* függvénnyel) az első nulláig. Minden szám előállítása után küldje el a "fogyasztó" folyamatnak csővezetéken. A fogyasztó a csővezetékből kiolvassa a kapott számokat, és online összegzi (menet közben összegzi és nem egy tömbbe rakja). Miután nem kap több számot, az összeget kiírja. A termelő megvárja, míg véget ér a fogyasztó folyamat.

# Összefoglalás:

- A név nélküli csővezeték egy sor (FIFO) alapú kommunikációs csatorna, a felhasználó számára egy speciális névnélküli állomány, amelyet létrehozás után a fájlleíróin keresztül érhet el (egyik az írásra, másik az olvasásra).
- Csővezeték létrehozása: *pipe(fildes[2])*, ahol
  - filedes[0]: fájlleíró olvasásra
  - *filedes[1]*: fájlleíró írásra
- A kiolvasott adatok törlődnek a csővezetékből. Ha kiürül a csővezeték, akkor az olvasó folyamat, ha megtelik, akkor az író folyamat várakozik.
- Ha a csővezeték írható végeit bezárjuk, akkor az olvasó végére fájlvége jel megy, ha az olvasó végeket zárjuk be, nem lehet bele írni (*write* hibával ér véget).
- Fork után a gyerek "örökli" a két fájlleírót (duplikált) -> a csővezetéknek két eleje és vége lesz, általában bezárja a szülő és gyermek azt a végét amit nem használ.
- Több olvasó vég esetén is csak egyszer lehet egy adatot kiolvasni. Több írható vég esetén is a beírás sorrendjében jelenik meg az olvasóknál az adat (FIFO).

## Nevesített csővezeték (FIFO)

A nevesített csővezeték szintén egyszerű kommunikációs eszköz, amelynek segítségével egyszerűen tudjuk megvalósítani az információ átadást tetszőleges (nem csak "családtag") folyamatok között. A csővezeték a felhasználó számára egy speciális névvel rendelkező állomány formájában jelenik meg. Tudunk a fájl nevére hivatkozni, nem kell a fájlleírókat továbbítani a kommunikációban résztvevő folyamatok között, ezért könnyen használhatják nem csak leszármazott folyamatok, hanem más "külső" folyamatok is. Mindegyik folyamat használhatja, amelyik ismeri a fájl nevét és rendelkezik a megfelelő jogosultságokkal. Az adatokat a beírás sorrendjében tudjuk kiolvasni. Egy FIFO típusú fájlt az *mknod* vagy az *mkfifo* függvénnyel tudunk létrehozni, majd az *open* függvénnyel megnyitva a név nélküli csővezetékhez hasonló használhatjuk. Hasonlóan a név nélküli csővezetékhez, ha kiürül a csővezeték, akkor az olvasó folyamat, ha megtelik, akkor az író folyamat várakozik. A folyamatoknak maguknak kell megnyitni a fájlt, a megnyitásnál várakoznak, ha a fájl nincs megnyitva "minkét irányba" (read, write). A kommunikáció végén a fájl töröléséről a folyamatoknak kell gondoskodni.

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
```

### Paraméterek:

- pathname: a FIFO fájl neve (elérési útvonala)
- mode: hozzáférési jogosultságok S\_IXY alakban, ahol
  - o X jogosultság a következők egyike:
    - R: olvasás
    - W: írás
    - X: végrehajtás
  - o Y jogosult a következők egyike:
    - USR: felhasználó
    - GRP: felhasználócsoport
    - OTH: egyéb felhasználó

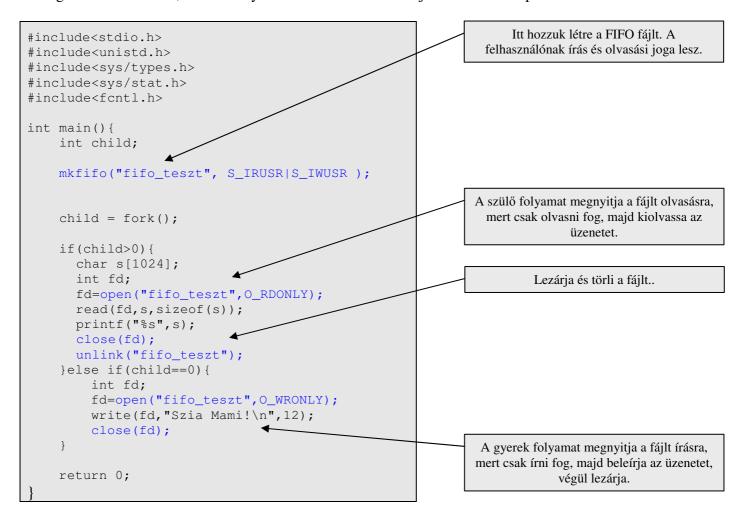
### Visszatérési érték:

- siker esetén 0
- hiba esetén -1

#### Hibák:

- EACCES: a megadott útvonalon (pathname) nincs végrehajtási jogosultságunk
- EEXIST: a megadott név (pathname) már létezik
- ENOENT: a megadott útvonal (pathname) egyik könyvtára nem létezik
- ENOTDIR: a megadott útvonal (pathname) egyik könyvtára nem könyvtár
- ENAMETOOLONG: a megadott név, útvonal (pathname) túl hosszú
- ENOSPC: nincs elég hely
- EROFS: a megadott útvonal (pathname) csak olvasható fájlrendszeren van

Nézzünk egy példát. A szülő folyamat létrehoz egy nevesített csővezetéket, a gyerek folyamat beleír egy szöveget a csővezetékbe, a szülő folyamat ezt kiolvassa és kiírja a standard output-ra.



## Javasolt feladat:

Írjon egy egyszerű kétszemélyes csevegő (chat) programot. A program parancssori argumentuma egy állomány elérési útvonala legyen. Ha létezik az állomány, akkor és nyissuk meg. Ha nem, akkor hozzuk létre FIFO-ként, és szintén nyissuk meg. A program egy példánya kérjen be egy sort a felhasználótól, írja a FIFO-ba, majd olvassa ki onnan a választ és írja ki a felhasználó képernyőjére. Ezt felváltva tegye a két folyamat, az egyik írással kezdje, a másik olvasással, attól függően, hogy melyik hozta létre a FIFO-t.

# Összefoglalás:

- A nevesített csővezeték egy sor (FIFO) alapú kommunikációs csatorna, a felhasználó számára egy speciális állomány, amelyet létrehozás után a közönséges fájloknál megszokott műveletekkel használhat.
- Csővezeték létrehozása: *mkfifo*
- A folyamatok a megnyitásnál várakoznak, ha a fájl nincs megnyitva "minkét irányba" (read, write).
- A kiolvasott adatok törlődnek a csővezetékből. Ha kiürül a csővezeték, akkor az olvasó folyamat, ha megtelik, akkor az író folyamat várakozik.
- Ha a csővezeték írható végeit bezárjuk, akkor az olvasó végére fájlvége jel megy, ha az olvasó végeket zárjuk be, nem lehet bele írni.