

Név nélküli csővezeték (pipe)

A *pipe* (csővezeték) egy egyszerű kommunikációs eszköz, amelynek segítségével egyszerűen tudjuk az információ átvitelt 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 végét bezárjuk, akkor az olvasó végére fájlvége jel megy, ha az olvasó végeket zárjuk be, nem lehet bele írni.

```
#include <unistd.h>
```

```
int pipe(int fildes[2]);
```

Paraméterek:

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

Visszatérési érték:

- siker esetén 0
- hiba esetén -1

Hibák:

- EFAULT: érvénytelen `fildes`
- EMFILE: túl sok fájlleíró használ a folyamat
- ENFILE: túl sok megnyitott fájl van a rendszerben

A csővezeték 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 csővezetékot, a gyerek folyamat beleír egy szöveget a csővezetékbe, a szülő folyamat ezt kiolvassa és kiírja a standard *output*-ra.

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

int main(){

    int fd[2];
    int child;

    if( pipe(fd) ){
        perror("pipe");
        return 1;
    }

    child = fork();

    if(child>0){
        char s[1024];
        close(fd[1]);
        read(fd[0],s,sizeof(s));
        printf("%s",s);
        close(fd[0]);

    }else if(child==0){
        close(fd[0]);
        write(fd[1],"Szia Mami!\n",12);
        close(fd[1]);

    }

    return 0;
}
```

Itt foglalunk helyet a két fájlleírónak.

A *pipe* függvény megnyitva, létrehozza a csővezetékot és visszaadja fd-be a két fájlleíró.

A szülő folyamat lezárja az író véget, mert csak olvasni fog, majd kiolvassa az üzenetet és lezárja az olvasó véget is.

A gyerek folyamat lezárja az olvasó véget, mert csak írni fog, majd beírja az üzenetet és lezárja az író véget is.

Javasolt feladat:

Egy "termelő" folyamat állítson elő egyszámjegyű véletlen számokat (*rand()* függvénnyel) az első nulláig. Minden szám előállítás után küldje el a "fogyasztó" folyamatnak csővezetéken. A fogyasztó a csővezetékéből 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 név nélküli állomány, amelyet létrehozás után a fájlleíróin keresztül érhet el (egyik az írásra, másik az olvasásra).
- Csővezeték létrehozása: *pipe(fildes[2])*, ahol
 - *fildes[0]*: fájlleíró olvasásra
 - *fildes[1]*: fájlleíró írásra
- A kiolvasott adatok törlődnek a csővezetékéből. 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 fájlleíró (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 törléséről 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
 - **X** jogosultság a következők egyike:
 - **R**: olvasás
 - **W**: írás
 - **X**: végrehajtás
 - **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 csővezeték, a gyerek folyamat beleír egy szöveget a csővezetékbe, a szülő folyamat ezt kiolvassa és kiírja a standard output-ra.

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>

int main(){
    int child;

    mkfifo("fifo_teszt", S_IRUSR|S_IWUSR );

    child = fork();

    if(child>0){
        char s[1024];
        int fd;
        fd=open("fifo_teszt",O_RDONLY);
        read(fd,s,sizeof(s));
        printf("%s",s);
        close(fd);
        unlink("fifo_teszt");
    }else if(child==0){
        int fd;
        fd=open("fifo_teszt",O_WRONLY);
        write(fd,"Szia Mami!\n",12);
        close(fd);
    }

    return 0;
}
```

Itt hozzuk létre a FIFO fájlt. A felhasználónak írás és olvasási joga lesz.

A szülő folyamat megnyitja a fájlt olvasásra, mert csak olvasni fog, majd kiolvassa az üzenetet.

Lezárja és törli a fájlt..

A gyerek folyamat megnyitja a fájlt írásra, mert csak írni fog, majd beleírja az üzenetet, végül lezárja.

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 fájlloknál megszokott műveletekkel használhat.
- Csővezeték létrehozása: *mkfifo*
- A folyamatok a megnyitásnál várnak, ha a fájl nincs megnyitva „minkét irányba” (read, write).
- A kiolvasott adatok törölődnek 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.