Problémák

- A szignál érkezéséről nem tud az a folyamat, amihez a szignált küldik. Ahhoz, hogy a címzett folyamat tudomást szerezzen a neki küldött szignálról, fel kell függeszteni a folyamatot egy szignál érkezéséig.
- A szignállal küldött információ a handlerhez érkezik, nem ahhoz a folyamathoz, melynek a szignált címezték. Így ezt az információ nem jut el a címzett folyamathoz.
- A szignállal küldött információt akkor kell feldolgozni, amikor érkezik, vagyis a küldő folyamat határozza meg a feldolgozás idejét, nem a fogadó.

Szignál kezelése folyamatból

<signal.h>

- **sigtimedwait**(&{szignálkészlet}, &{info}, &{várakozás})

  Várakozik míg

  – lejár a {várakozás} által meghatározott idő, vagy

  – érkezik a folyamathoz egy a {szignálkészlet}-ben szereplő szignál. Ekkor {info} ugyanúgy kap értéket, ahogy az összetett handler info paramétere.

  Értéke: Az érkezett szignál száma, vagy
  -1, ha nem érkezett szignál az időtúllépésig.

Problémák

- A szignálhoz csatolt információ csak egy egész szám vagy egy mutató lehet.
  - Az egész számnak csekély az információmennyisége (ha pl. egy mondatot akarunk így átküldeni, 4 karakterenként 1 szignált küldve tehetjük meg).
  - A mutató tetszőlegesen összetett információra mutathat, ám ennek az információnak a közös kódban kell létrejönnie, hiszen a szétválasztás (fork) után ugyanaz a memóriacím a két folyamatban fizikailag teljesen más memóriaterületet jelent. Így mutatóval olyan információt nem tudunk átadni, ami a küldő folyamatban keletkezett.

Névtelen cső

- **pipe**(int pipefd[2])
  Létrehoz egy csövet és a fájlazonosítóit elhelyezi a paraméterként átadott két elemű tömbben.
  A tömb elemei int típusúak. A
  - [0] eleme az olvasásra, az
  - [1] eleme pedig az írásra használható fájlazonosítót kapja.

Névtelen cső

- **read(…)**
  Vár, míg megjön a várt adatmennyiség vagy hiba történik (pl. eof).
- **write(…)**
  Vár, míg ki tudja írni a teljes adatmennyiséget (pl. ha tele a fifo), vagy hiba történik.
- **close(…)**
  A cső végeit (R/W) külön-külön kell lezárni.

(Részletekért ld.: Fájlkezelés rendszerhívások használatával)

Névtelen cső hátrányai

- Mivel a csövet olyan közös kódban kell megnyitni, mely minden a csövet használó folyamat közös kódja, ezért esetlegesen olyan folyamatokban is nyitva lesz, melyekben nem kívánjuk használni.
- Csak olyan folyamatok tudnak rajta keresztül kommunikálni, melyek azonos kódból indultak.

Nevesített cső (named pipe)

- **mkfifo**({fájlnév}, {engedélyek})
  Létrehoz egy cső fájlt a megadott hozzáférési engedélyekkel.
  {engedélyek}: S_I[R|W][USR|GRP|OTH])

- **unlink**({fájlnév})
  Törli a cső fájlt.

Nevesített cső (named pipe)

- {csőazonosító} = **open**({fájlnév}, {mód})
  Cső megnyitása.
  {mód}:
  - O_RDONLY: olvasásra
  - O_WRONLY: írásra
  - O_RDWR: írásra és olvasásra

  Egyirányú megnyitás esetén mindaddig vár, míg a másik irányba is nyitás történik.
- **close**({csőazonosító})
  Cső lezárása.

Nevesített cső (named pipe)

- **read**({csőazonosító}, {kezdőcím},
  {olvasandó byte-ok száma})

  Olvasás csőből.

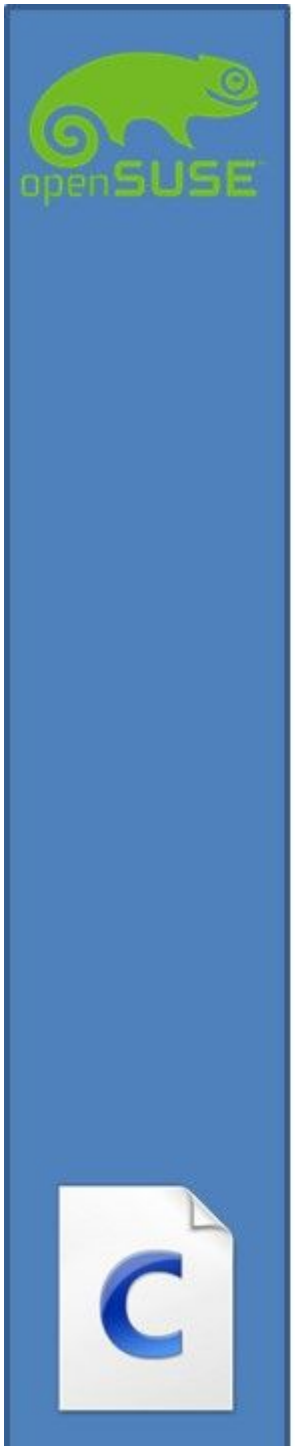
  Vár, míg megjön a várt adatmennyiség vagy hiba történik (pl. eof).
- **write**({csőazonosító}, {kezdőcím},
  {írandó byte-ok száma})

  Írás csőbe.

  Vár, míg ki tudja írni a teljes adatmennyiséget (pl. ha tele a fifo), vagy hiba történik.

Nevesített cső (named pipe)

- **A cső csak úgy nyitható meg, ha mindkét végét megnyitjuk. Az open vár, míg a cső másik véget is megnyitja egy folyamat. Tehát**
  - Mialatt meg van nyitva írásra, akárhány szálon nyitható olvasásra.
  - Mialatt meg van nyitva olvasásra, akárhány szálon nyitható írásra.
  - Ha nincs megnyitva írásra de van benne adat, az *olvasásra* nyitás vár, míg írásra nyitás is érkezik, majd elsőként a már bennlévő adatot olvassa ki (nem vész el a bennmaradt adat).
- **eof akkor következik be, ha minden írásra nyitó folyamat lezárta és kiürült.**
- **Külön programokban elhelyezett folyamatok is kommunikálhatnak rajta keresztül.**



Cső állapotának lekérdezése

<poll.h>

- **pollfd** struktúra
  - fd: file (cső) azonosító
  - events: figyelt esemény
  - revents: bekövetkezett esemény
- events és revents események:
  - POLLIN: van olvasható adat a fájlban,
  - POLLOUT: lehet írni adatot a fájlba,
  - …

poll

- int **poll**({figyelendők tömbje}, {figyelendők száma}, {időtúllépés})

  Blokkolja a folyamatot, míg
  - egyik figyelt fájlleíróban a figyelt esemény bekövetkezik,
  - szignál érkezik,
  - időtúllépés történik.

  Értéke:
  - pozitív: Figyelt esemény következett be az egyik figyelt fájlban. Ekkor a visszatérési érték a bekövetkezett események száma, és a bekövetkezett események a {figyelendők tömbje}[].revents adattagjaiban találhatók.
  - 0: letelt az {időtúllépés} által meghatározott idő.
  - negatív: hiba

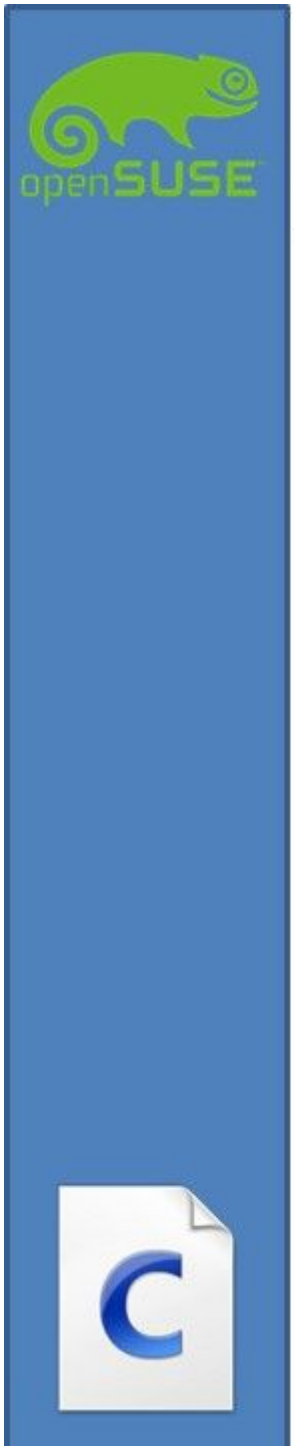
ppoll

- int **ppoll**({figyelendők tömbje}, {figyelendők száma}, {időtúllépés}, {szignálkészlet})

  Működése megegyezik a poll működésével, de
  - az időtúllépés timespec struktúra típusú, és
  - a várakozás ideje alatt a {szignálkészlet} által meghatározott szignálmaszk érvényes.

select

- **fd_set**: fájlleíró készlet típus
- **FD_ZERO**({készlet}):
  {készlet} kiürítése
- **FD_SET**({fájlazonosító}, {készet}):
  hozzáadja a {fájlazonosító}-t a {készet}-hez
- **FD_CLEAR**({fájlazonosító}, {készet}):
  eltávolítja a {fájlazonosító}-t a {készet}-ből
- **FD_ISSET**({fájlazonosító}, {készet}):
  értéke 0, ha nem tartalmazza a {készet} a {fájlazonosító}-t



select

- **select**({ndfs}, {readfds}, {writefds}, {exceptfds}, {időtúllépés})

  Figyeli a fájlleírókkal megadott fájlokat, és felfüggeszti a folyamat működését, míg
  - valamelyikük készen áll a figyelt műveletre,
  - vagy eltelik az {időtúllépés}-ben megadott idő,
  - vagy szignál érkezése szakítja meg a folyamatot.

  Értéke:

  >0: Az egyes fájlleíró készletekbe azon fájlazonosítók kerülnek, melyek a figyelt funkcióra készenállnak. A visszatérési érték ezen azonosítók együttes száma.

  0: Időtúllépés.

  -1: Hiba.

Problémák

A cső állapotának lekérdezése nem garantálja az adatérvényességet.

Például, ha két folyamat közel egyidőben lekérdezi, hogy van-e adat a csőben, amiben egyetlen adat van, akkor mindkettő azt az információt kapja, hogy van kiolvasható adat. Ha ezután mindkettő megpróbál olvasni a csőből, az egyik kiolvassa az adatot, a másik pedig nem talál adatot benne.