5. Gyakorlat

Nyitott fájlok listázása:

- A rendszerben az összes nyitott fájl kilistázható az **lsof** (List Open Files) program segítségével.
- Kapcsolókért és részletekért, lásd: **man lsof**
- A program kimenete:
  - COMMAND: a processzhez tartozó parancsnév,
  - PID: a processz azonosítója,
  - TID: thread ID, a szál azonosítója,
  - USER: a processz tulajdonosa,
  - FD: File Descriptor azonosító (nem mindenhol értelmezhető),
  - TYPE: milyen típusú a fájl
    - REG = hagyományos,
    - CHR = karakteres eszköz,
    - DIR = jegyzék,
    - FIFO = csővezeték,
    - stb.
  - DEVICE: a fájlt tároló eszköz major és minor azonosítója, illetve dev fájl esetén annak az azonosítója,
  - SIZE/OFF: fájl mérete (nem mindenhol értelmezhető),
  - NODE: a fájlhoz tartozó inode száma,
  - NAME: a fájl neve.

Hálózati kapcsolatok listázása:

- A **netstat** program segítségével kilistázhatók a hálózati kapcsolatok, routing táblák, hálózati interfészek adatai, statisztikák, stb.
- Fontosabb kapcsolók:
  - netstat **-r** – routing táblák megjelenítése (gyakorlatilag a **route** parancs megfelelője),
  - netstat **-i** – hálózati interfészek megjelenítése (kb. **ifconfig** rövidített változata),
  - netstat **-s** – protokollonkénti statisztikák megjelenítése,
  - netstat **-n** – numerikusan jeleníti meg a címeket, nem próbál névfeloldást végezni,
  - netstat **-p** – megjeleníti annak a processznek a PID-jét és nevét, amelyikhez az adott kapcsolat tartozik,
  - netstat **-a** – minden kapcsolat megjelenítése,
  - Egyéb kapcsolókért és részletekért, lásd: **man netstat**
- A **netstat -anp** kimenete:
  - Active Internet connections:
    - Proto: a socket által használt protokoll,
    - Recv-Q: fogadó oldali várakozósorban mennyi adat van,
    - Send-Q: küldő oldali várakozósorban mennyi adat van,
    - Local Address: a socket "lokális végéhez" tartozó cím és port,
    - Foreign Address: a socket "távoli végéhez" tartozó cím és port,
    - State: a socket állapota (pl.: ESTABLISHED, LISTEN, SYN_SENT, stb.)
    - PID/Program name: annak a processznek a PID-je és neve, amelyikhez a socket tartozik.

- Active UNIX domain Sockets:
  - Proto: a socket által használt protokoll (általában unix),
  - RefCnt: hány darab processz kapcsolódik a sockethez,
  - Flags: leggyakrabban SO_ACCEPTION (ACC), akkor használják, ha a processzek kapcsolódásra várnak,
  - Type: a socket típusa (pl.: STREAM, RAW, PACKET, DGRAM, stb.)
  - State: a socket állapota (pl.: LISTENING, CONNECTED, stb.),
  - I-Node:
  - PID/Program name: annak a processznek a PID-je és neve, amelyikhez a socket tartozik.
  - Path: az útvonal, ahonnan a processz kapcsolódott a sockethez.

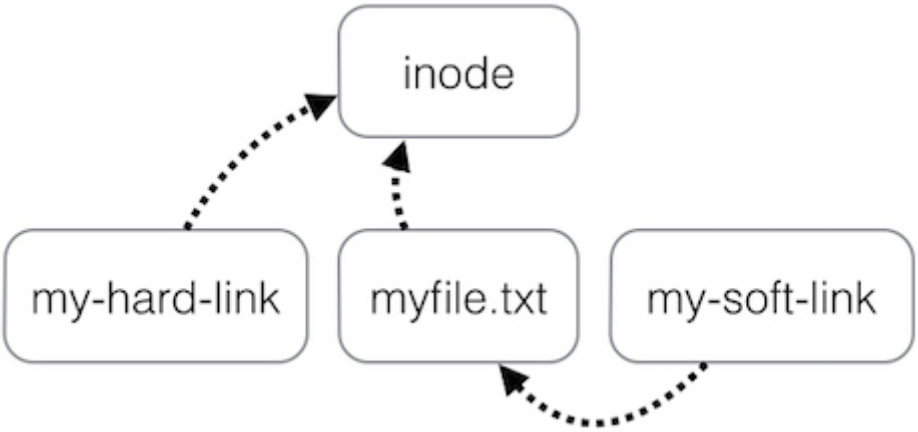
UNIX API, C függvény-, illetve rendszerhívások:

- **open:** Az open() rendszerhívás segítségével megnyithatunk és létrehozhatunk fájlokat a megadott paraméterektől függően különböző módokon. A visszatérési érték egy file descriptor (FD), aminek a felhasználásával az adott fájlt kezelhetjük. (Lásd: **man 2 open**)
- **close:** A close() rendszerhívással „lezárhatunk fájlokat”, paraméterként egy FD-t vár, visszatérési értéke hiba esetén -1, egyébként 0. (Lásd: **man 2 close**)
- **lseek:** Segítségével „a kurzort pozícionálhatjuk” az FD-hez kapcsolódó fájlban. Pozícionáláshoz paraméterként az FD és egy offset mellett a pozícionálás módját kell megadni, ami lehet a SEEK_SET, SEEK_CUR, illetve SEEK_END valamelyike. Visszatérési értéke megmondja, hogy milyen offset-re állt be a kurzor. (Lásd: **man lseek**)
- **read:** A read() segítségével az FD-hez kapcsolódó fájlból kiolvashatunk egy bufferbe megadott mennyiségű byte-ot (de nem többet, mint a fájl mérete). Visszatérési értéke a beolvasott byte-ok száma. (Lásd: **man 2 read**)
- **write:** A write() segítségével az FD-hez kapcsolódó fájlba írhatunk ki egy buffer tartalmából megadott számú byte-nyit. Visszatérési értéke a kiírt byte-ok száma. (Lásd: **man 2 write**)
- **flock:** Az flock() segítségével korlátozható, hogy miközben egy processz dolgozik az FD-hez kapcsolódó fájllal, más processzek milyen mértékben tudjanak hozzáférni. Tulajdonképpen zárolhatjuk a fájlt megadott műveletekre. A megadható lehetőségek LOCK_SH (osztott), LOCK_EX (kizárólagos) és LOCK_UN (zárolás feloldása). (Lásd: **man flock**)
- **link:** Létrehozhatunk hivatkozásokat, a link() segítségével ún. hard linkeket. Paraméterként a hivatkozott fájl nevét (igazából elérési útvonal, path) és a link nevét kell megadni. (Lásd: **man 2 link**)
- **unlink:** Segítségével megszüntethetünk hivatkozásokat, azaz lényegében törölhetjük a linkeket. (Lásd: **man 2 unlink**)
- **stat:** A stat() rendszerhívás segítségével beolvashatjuk egy fájl adatait egy **struct stat** típusú bufferbe. A **stat** hívás esetén névvel, **fstat** esetén FD-vel hivatkozhatunk a fájlra. (Lásd: **man 2 stat**)

Soft Link vs. Hard Link:

- **Hard Link létrehozása:** ln myfile.txt my-hard-link
- **Soft Link (vagy Symbolic Link) létrehozása:** ln -s myfile.txt my-soft-link
- Amikor létrehozunk egy fájlt, létrejön egy inode, ami a fájl „metaadatait” (pl.: a fájl blokkjainak helye, hivatkozások száma, stb.) tárolja. Hard link létrehozásakor a már meglévő inode-hoz jön létre egy újabb hivatkozás, a link a „tartalomra mutat”, ezzel szemben soft link létrehozásakor új inode készül, a link a „fájlnévre mutat”, nem pedig a

tartalomra. Ebből kifolyólag, ha töröljük a fájlt, a soft link egy nem létező fájlnévre mutat, azaz a tartalom már nem elérhető, míg a hard linkre nézve csupán az inode-ra való hivatkozások száma csökkent, a tartalom továbbra is elérhető a linken keresztül.



- Forrás: https://askubuntu.com/questions/108771/what-is-the-difference-between-a-hard-link-and-a-symbolic-link
- Link létrehozásról bővebben: **man ln**

Feladat 1:

Eddigi ismereteinket felhasználva írjunk programot, amelyben 3 gyerek processz „felváltva” (lock!) ír 1-1 byte-ot egy fájlba, addig még annak a mérete el nem éri az 1000 byte-ot. Miután a gyerekek elvégezték a feladatukat, a szülő 50 byte-onként, új sorba, kiírja a fájl tartalmat a képernyőre.

Feladat 2:

A már jól ismert tippelgetős programunkat írjuk át úgy, hogy a gyerek processz a tippet ezúttal nem visszatérési értékként továbbítja a szülőnek, hanem beleírja egy fájlba, a szülő pedig onnan kiolvassa, és megmondja, hogy kisebb, nagyobb, vagy éppen ugyanaz, mint a „gondolt” szám.