Linux parancssor használata. Alapparancsok.

„…mert karakteres felület mindig van…"

A parancssor alapjai

Hogyan használjam?

A Command Line Interface (CLI)

- Szöveg alapú interfész, amelyben szöveges parancsok segítségével tudjuk vezérelni a számítógépet.
- A parancsok a futás előtt lefordításra kerülnek a parancsértelmező által a számítógép számára értelmezhető nyelvre.
- A parancsértelmezőt más néven ***shell***–nek is hívjuk.
- A ***terminál*** egy olyan alkalmazás amely segítségével a parancsértelmező számára tudunk parancsokat begépelni és futtatni.

Terminálok

- Szervereken általában nem szükséges a grafikus felület. Itt alapból egy terminált kapunk, amelybe bejelentkezve egy *prompt*-ot látunk, amely várja, hogy begépeljük a parancsot.
- A prompt a felhasználó igényei szerint beállítható információkat tartalmaz (lásd később…).
- Asztali gépeken illetve grafikus felületen az alkalmazások közt megtaláljuk a terminált amit el tudunk indítani.
- Szintén grafikus felületen léteznek un. virtuális terminálok. Ezek segítségével a grafikus felület futásával egy időben tudunk parancsokat futtatni. Minden virtuális terminálba be kell jelentkezni. Váltás az *Ctrl+Alt+F1 – F8* billentyűkombinációkkal lehetséges.

A prompt

- A prompt a terminálba való bejelentkezés után válik láthatóvá. Tulajdonképpen a prompt jelzi a felhasználónak, hogy parancsokat lehet begépelni.
- A prompt valójában karaktersorozat, amely a felhasználó igényei szerint változtatható, hogy milyen információkat tartalmazzon.
- Pl: **sysadmin@localhost:~$**

A $ jel jelzi, hogy normál felhasználói módban vagyunk.
Rendszeradminisztrátort a # jel jelzi.

Kinek a nevében vagyunk bejelentkezve?

A gép neve

Hol állunk jelenleg a könyvtárstruktúrában?
~ (tilde) a saját felhasználói fiókot jelöli.

A shell

- A *shell* vagy más néven parancsértelmező segítségével a felhasználó által begépelt parancsok lefordításra kerülnek a gép által értelmezhető és végrehajtható formára.
- A shell-nek több típusa is létezik. Talán a legismertebb a *BASH* (*B*ourne *A*gain *Sh*ell).
- A BASH által nyújtott néhány fontosabb szolgáltatás:
  - *Scriptek* futtatása: parancsok elhejezése egy fájlban,majd ezek együttes futtatása. Lehetőség van vezérlési szerkezetek alkalmazására is.
  - *Alias*: Lehetőség van parancsokhoz, szkriptekhez „álneveket" megadni.
  - *Változók*: belső tárolt információk, amik megváltoztathatók a rendszer működésének finomhangolása érdekében.

Parancsok formátuma

- Bizonyos Linux parancsok önállóan kiadva is működnek, mások használatához kötelezően meg kell adni bemenetet.
- A Linux parancsok általános szintaxisa:

  parancs [opciók] [paraméterek]
- Az opciók a parancs viselkedését módosítják, míg a paraméterek a működéshez szükséges információkat tartalmazzák.
- Pl. *ls /home* kilistázza a /home könyvtár tartalmát.
- *FONTOS!!!*

  A Linux parancsok különbséget tesznek a kis és a nagybetűk közt!!!

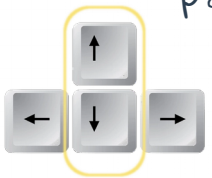
A parancsok opciói

- Az opciók a parancsok működését módosítják.
- Vannak egy betűs opciók, ezek elé egyszeres kötőjelet kell tenni. Pl. *ls -l*
- Vannak opciók melyek kulcsszavakból állnak, ezek elé dupla kötőjelet kell tenni. Pl.: *ls --human-readable*
- A kulcsszavas formátumú opcióknak van egybetűs változata is.
- Pl. *ls -h* és a *ls --human-readable* parancsok azonosak.
- Gyakran használunk Linux parancsokban együttesen opciókat is és paramétereket is.

  Pl. *ls -lh /home*

Parancselőzmények

- A kiadott parancsok egy parancspufferbe kerülnek eltárolásra hasonlóan a veremhez (LIFO).
- A puffer tartalmát a *history* parancs segítségéével lehet megjeleníteni.
- A kurzormozgató nyilakkal elő tudjuk hívni az előzetesen begépelt parancsokat:



  - Felfelé nyíl: visszafelé lépkadés a kiadott parancsok közt.
  - Lefelé nyíl: előrelépkedés a kiadott parancsok közt.
- A *history <szám>* parancs segítségével meg tudjuk jeleníteni az utolsó *szám* db kiadott parancsot, sorszámokkal együtt.
- Ha az előzményben található parancsok közül egy adott sorszámút szeretnénk lefuttatni akkor használjuk a felkiáltójelet „*!*”.

  Pl. *!27*:a parancspuffer 27. sorszámú parancsát fogja futtatni.

A BASH által használt változók

- A BASH tárolhat adatokat, amelyekkel a rendszer működését befolyásolhatjuk.
- Ezen változók megjelenítése az *echo* parancs segítségével történik.
  Pl. *echo $HISTSIZE*
- Amennyiben a változó értékére vagyunk kíváncsiak, akkor a változó neve elé a $ jelet kell beillesszük.
- Amennyiben egy változó értékét meg szeretnénk változtatni, akkor csak a változó nevét kell használjuk.
  Pl. *HISTSIZE=500*
- A BASH sok változót használ (lásd később…).

A *PATH* változó

- Egyike a legfontosabb változóknak.
- Könyvtárak elérési útvonalait tartalmazza, amelyekben a felhasználók által futtatható parancsok találhatók. Megjelenítése a *echo $PATH* paranccsal lehetséges.
- Amennyiben egy parancs nem található a PATH változó elérési útvonalai között, a *command not found* üzenetet kapjuk vissza.
- A PATH-hoz újabb útvonalakat tudunk adni amennyiben szükséges.
  Pl. *PATH=/usr/bin/custom:$PATH*

Az *export* parancs

- A Linux által használt változókat két típusba soroljuk:
  - *Környezeti változók*: ezek a változók a parancsok értelmezésénél illetve a feladatok végrehajtásánál válnak szükségessé.
  - *Helyi (lokális) változók*: ezek a változók inkább a felhasználó-alapú feladatoknál lehetnek szükségesek, felhasználókhoz kapcsolódó adatokat tartalmazhatnak.
- A rendszer által használt környezeti változókat az *env* parancs segítségével tudjuk megjeleníteni.
- Amennyiben létrehozunk tetszőleges névvel és értékkel egy változót, azt az *export* parancsal tudjuk a környezeti változók közé illeszteni.

  Pl. *export variable2='Else'*

Néhány egyéb parancs...

- *which*: megmutatja egy parancs elérési útvonalát
  Pl. *which ls* parancs kimenete: */bin/ls*
- *type*: információt ad egy parancsról, pl. milyen típusú (beépített vagy nem), elérési út, stb…. Jelzi ha egy parancs egy másik parancs álneve (alias). Hasonló a *which* parancshoz.
- *alias*: ezzel a paranccsal egy másik, esetleg sok kapcsolót használó, hosszú parancsot lehet lerövidíteni.
  Pl. *alias lh='ls -shl'*
  az ls -shl parancsot egy rövidebb paranccsal (*lh*) lehet használni.
  Az alias-ok csak a shell bezárásáig érvényesek.

Helyettesítő karakterek

- * karakter: tetszőleges számú és jelentésű karaktert helyettesíthet.
  Pl. echo /etc/t* kilistázza az összes állományt a /etc mappában amely „t" betűvel kezdődik.
- ? karakter: **egy** tetszőleges jelentésű karaktert helyettesít.
  Pl. echo /etc/t??????? kilistázza az összes „t" betűvel kezdődő és 8 karakter névhosszúságú állományt a /etc mappában.
- [] karakterek: a szögletes zárójelek közé beírt egy vagy több karakter **vagy**-al lesz értelmezve.
  Pl. echo /etc/[gu]* kilistáz minden olyan állományt a /etc mappában, amely „g" vagy „u" betűvel kezdődik.
  Tartományt is meg lehet adni, amelynek elemei szintén **vagy**-al vannak értelmezve
  Pl. [a-c] [1-9] stb.

Helyettesítő karakterek(folyt...)

- ! karakter: a [] található tartomány kizárására használjuk. Pl. *echo /etc/[!gu]** kilistázza az összes állományt a /etc mappában, amely ***NEM*** „g" vagy „u" betűvel kezdődik.
- Idézőjelek:
  - „ : a dupla idézőjelek közti szöveget csak részben értelmezi a BASH. A helyettesítő karaktereket nem, de a környezeti változókra hivatkozva annak tartalmát igen.
  - '': az egyszeres idézőjelek közti szövegben a BASH nem értelmez semmiféle speciális karaktert.
  - ``: fordított egyszeres idézőjelek (AltGr+7). Ezek közé a BASH által értelmezhető parancsokat kell írni és a beillesztés helyén a parancs végrehajtásának kimenete jelenik meg.
- \ karakter: feloldja a speciális karakterek jelenését.

Több parancs futtatása

- ; karakter: a pontosvesszővel elválasztott parancsok egymás után lefutnak, függetlenül attól, hogy mi az egyes parancsok kimenete.
- && karakterek: a „dupla és" karakterekkel elválasztott parancsok balról kezdve indulnak. Ha az első parancs hiba nélkül lefut, akkor a másik parancs is lefut. Ellenkező esetben nem. *Logikai ÉS.*
- || karakterek: a „dupla cső" karakterekkel elválasztott parancsok is balról-jobbra indulnak. Ha az első parancs lefut akkor a másodikra nincs szükség és nem fut le. Ha azonban az első nem fut le, akkor a második lefut. *Logikai VAGY.*

Ajánlott parancsok

- *whoami*: milyen felhasználónévvel vagyunk bejelentkezve?
- *uname*: információ a rendszerről. A *-a* kapcsolóval való használata az összes információt adja.
- *pwd*: az aktuális hely a könyvtárstruktúrában, ahol jelenleg állunk.
- *echo*: egy változó értékének megjelenítésére használható.
- *which*: a paraméterként megadott parancs helyét adja meg a $PATH változóban tárolt útvonalak közül.
- *history*: a parancselőzmény megjelenítése.
- *type*: információ a paraméterként megadott parancsról.
- *alias*: helyettesítő név megadása egy parancshoz.
- *export*: környezeti változó értékadása illetve létrehozása.

Segítség parancssoros felületen

A *man* parancs

- A *man* parancs megjeleníti a paraméterkén megadott parancs használatára vonatkozó leírást.
  Pl. *man cal* segítség a *cal* parancs használatához.
- A több képernyőnyi tartalmak közt a *more* vagy a *less* parancsok segítenek, hogy lapozni tudjunk.
- A manuálok (parancsok leírásai) szekciókra vannak osztva.
- Ha egy manuál nagyon bőséges (sok-sok oldal) a „/" jel és egy keresendő kulcsszó segítségével tudunk egy kifejezésre rákeresni. Lapozgatás a „n" vagy a „N" karakterek segítségével történik.

Manuál fejezetek

1. Futtatható programok vagy shell parancsok
2. Rendszerhívások (a kernel számára kiadható rendszerhívások)
3. Megosztott könyvtárak függvényei
4. Eszközkezelők (/dev könyvtárban megtalálható eszközök)
5. Állományformátumok, pl /etc/passwd
6. Játékok
7. Egyéb (beleértve a makró csomagokat és szabályokat)
8. Rendszer-adminisztrációs parancsok (általában csak a root felhasználó számára)
9. Kernel rutinok [Nem alapértelmezett]

Keresés a manuálokban

- Ha egy parancs esetleg több fejezetben is szerepel, akkor a *man -f <parancs>* paranccsal kapok erről egy listát. Kimenete megegyezik a *whatis <parancs>* paranccsal.
- A manuálokban kulcsszó alapján való keresésre a *man -k <kulcsszó>* parancs nyújt segítséget. Kimenete megegyezik az *apropos <kulcsszó>* paranccsal.

Az *info* parancs

- A *man*-hoz hasonló információt ad a paraméterként megadott parancsról, de más formában.
- Itt kapcsolódó oldalakat is lehet találni, melyekre a kurzorral ráállva át lehet ugrani az adott oldalra.
- Beépített „*help*" segítségével könnyen tudunk keresgélni is a kapcsolódó oldalak közt.
- Az *info* parancs paraméter nélküli használatával a legfelső szintre léphetünk.

Egyéb segítségek

- *<parancs> --help* : leírás a parancs használatára vonatkozólag.
- További dokumentációk találhatók un. README fájlokban is. Ezek elérési útvonala változó, de általában a *usr/share/doc* vagy a **usr/doc** könyvtárakban találkozhatunk ilyen állományokkal.
- A *whereis <parancs>* segítségével a parancs elérési útvonalát is és a manuáljának elérési útvonalát is megkapjuk.

Fájlok vagy könyvtárak keresése

- *locate <MINTA>*: különböző adatbázisokban keresve próbálja meghatározni a rendszeren található állományok elérési útvonalát a **MINTA** alapján. A mintában spec. karakterek használata megengedett (*,?,[],!).
- Az adatbázisok frissítése az *updatedb* paranccsal lehetséges.
- *locate -c <MINTA>*: kiírja a találatok számát.
- *locate -b <MINTA>*: pontos egyezést keres a **MINTA**-val és a tartalmazást elhagyja.
  Pl. *locate -b "\passwd"*

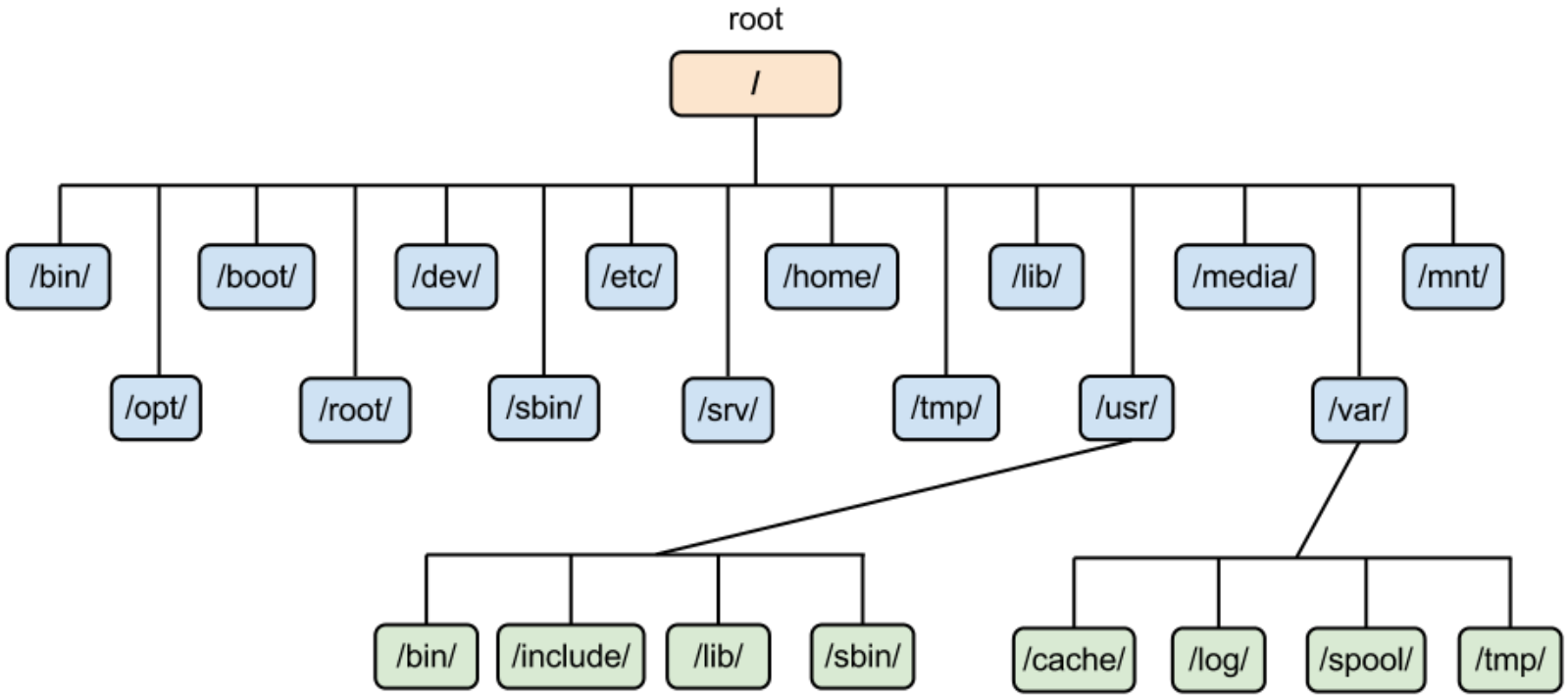
Ajánlott parancsok

- *man*: használati utasítás a paraméterként megadott parancshoz.
- *info*: a man-hoz hasonló segítség a parancs használatához.
- *more*: lapozó program hosszú szüvegekhez.
- *less*: lapozó program hosszú szövegekhez.
- *whatis*: a manuál mely fejezeteiben szerepel a paraméterként megadott parancs. Megegyezik a *man -f* paranccsal.
- *apropos*: kulcsszó alapján keresés. Megegyezik a *man -k* paranccsal.
- *whereis*: parancs elérési útvonalát adja vissza és a manuálban található fejezetét.
- *locate*: állomány elérési útvonalát adja vissza.

Munka fájlokkal és könyvtárakkal

A Linux könyvtárstruktúrája

- Különbözik a Windows könyvtárstruktúrájától.



A Linux könyvtárstruktúrája

- */bin* : Binárisokat, azaz futtatható programokat tartalmazó könyvtár. Itt tárolódnak a standard parancsok és segédprogramok.
- */boot* : A rendszer bootolásakor használatos fájljait tartalmazó könyvtár.
- */dev* : Ebben a könyvtár találhatók a különböző eszközök kezeléséhez szükséges fájlok.
- */etc* : Itt olyan fájlok találhatók, melyek a rendszer és az egyes alkalmazások alapvető beállításaival vannak kapcsolatban. Rendszerkonfigurációs fájlokat tartalmaz.
- */home* : Ebben a könyvtárban kapnak helyet a felhasználók saját, bejelentkező (login) könyvtárai. Ezeket a könyvtárakat csak a tulajdonosa olvashatja.
- */lib* : Itt olyan fájlok találhatók, melyek a különböző programok függvénykönyvtárait tartalmazzák.

A Linux könyvtárstruktúrája

- */mnt* : ez a könyvtár arra szolgál, hogy külső adathordozót (pl. pendrive, CD-ROM, külső winchaster, stb…) tartalmát lehessen a Linux könyvtárstruktúrájába beilleszteni (*mount*).
- */proc* : Ezt a könyvtárat csak az operációs rendszer használhatja!
- */root* : Ez a rendszergazda (root) bejelentkezési (login) könyvtára, amely más számára nem olvasható.
- */sbin* : A rendszerindításhoz és karbantartáshoz használt rendszeradminisztrációs parancsokat tartalmazó könyvtár.
- */tmp* : Ez az ideiglenes fájlok könyvtára, amelyet azok a programok használnak, amelyeknek szükségük van ilyen fájlokra működésükhöz. A könyvtár tartalmát rendszerindításkor az operációs rendszer kitörli.

A Linux könyvtárstruktúrája

- */usr* : Általában ez az egyik legnagyobb könyvtár. Olyan fájlok és könyvtárak vannak benne, amelyek parancsokat (bin), rendszerparancsokat (sbin), függvénykönyvtárakat (lib), dokumentációkat (doc), kézikönyveket (man), forrásokat (src), ideiglenes fájlokat (spool) tartalmaznak.
- */var* : Olyan fájlokat tartalmazó könyvtár, amelyek állandóan változnak (pl. log fájlok).

Állományok elérési útvonala

- Linux alatt egy állományra (könyvtár,fájl,stb…) a könyvtárstruktúrában elfoglalt helye alapján lehet hivatkozni. Ezt hívjuk elérési útvonalnak.
- A Linux könyvtárstruktúra kezdő (kiindulási) pontja a struktúra gyökere, amit a / jellel jelölünk.
- Az egyes állományokat vagy ehhez viszonyítva adjuk meg, vagy pedig a könyvtárstruktúrában elfoglalt jelenlegi helyzetünkhöz viszonyítva.
- Ennek megfelelően létezik abszolút és relatív elérési út (lásd később).

  Pl. */usr/share/doc*

A *home* könyvtár

- A *home* könyvtár különleges szerepet tölt be a Linux könyvtárai közt annyiban, hogy ebben a mappában találjuk a felhasználói fiókokat (mappákat).
- Egy felhasználó ha terminálon keresztül jelentkezik be a rendszerbe (nem használva grafikus felületet), akkor a saját mappájába kerül, ahol teljes joga van az ott található állományok módosítására, törlésére, átnevezésére, stb… .

  Pl. a gazda nevű felhasználó felhasználói fiókja a */home/gazda* elérési útvonalon lesz megtalálható.
- A „~" (tilde) jellel minden felhasználó a saját felhasználói fiókjára tud hivatkozni.

Váltás könyvtárak közt

- *cd <elérési útvonal>* : ezzel a paranccsal az elérési útvonallal megadott könyvtárba tudunk ugrani.
- Ha nem tudun elérni egy könyvtárat, annak több oka is lehet:
  - Nincs jogunk belépni.
  - Rossz útvonalat adtunk meg (kis és nagybetűk különböznek!!!).
- Visszatérni a saját mappánkba a *cd ~* paranccsal lehetséges.
- Egy szinttel feljebb lépni a könyvtárstruktúrában a *cd ..* paranccsal lehetséges.
- Több szintet is tudunk feljebb lépni egy paranccsal:

  Pl. *cd ../..* paranccsal két szintet lépünk feljebb.
  cd / paranccsal a kiindulási pontba (gyökér) jutunk

Abszolút és relatív elérési utak

- *Abszolút elérési útvonal*: az állomány elérési útvonalát a gyökérhez (/) képest adjuk meg. Ezek az útvonalak mindig a „/" jellel kezdődnek.

  Pl. */usr/share/doc*

- *Relatív elérési útvonal:* Mindig az aktuális helyünkhöz viszonyítva adja meg az állomány helyét.

  Pl. Ha a */usr/share* mappában állunk és be szeretnénk lépni a */usr/share/doc* mappába, akkor relatív útvonallal megadva ez így néz ki: *cd doc* . (nincs „/" jel…)

- Annak eldöntésében, hogy melyik változatot használjuk az elérési utaknak általában az dönt, hogy melyik a rövidebb, melyik igényel kevesebb gépelést.

Könyvtár tartalmának listázása

- Az alap parancs az *ls*. Természetesen jó néhány kapcsolóval rendelkezik, amelyek használatával hatékonyabban tudunk listázni:
  - *ls --color* : színekkel különbözteti meg az egyes állomány típusokat.
  - *ls -a* : kilistázza a rejtett állományokat is.
  - *ls -l* : részletes lista. Lásd később.
  - *ls -h* : a fájlok méreténél megjeleníti a mértékegységet is.
  - *ls -ld <könyvtár>* : a paraméterben szereplő könyvtár részletes tulajdonságait jeleníti meg.
  - *ls -R* : rekurzívan megjeleníti az alkönyvtárak tartalmát is.
  - *ls -ls* : méret szerint rendez.
  - *ls -lt* : utolsó módosítás szerint rendez.
  - *ls --full-time* : a teljes időbélyeget jeleníti meg.
  - Az előző rendező kapcsolók a *-r* kapcsolóval ellenétes rendezést eredményeznek.
  - Helyettesítő karakterek használata hasznos, de picit gondolkodj előtte!!

Könyvtár tartalmának részletes listázása

- *ls -l* : részletes listát ad. A parancs kimenetében szereplő sorok részei balról jobbra tekintve:

  `drwxr-xr-x 1 root root 0 Apr 11 21:58 upstart`

  - Első oszlop az állomány típusa:
    - *d* könyvtár.
    - *-* szabályos fájl.
    - *l* szimbolikus link. Lásd később.
    - *s* socket. Hálózati folyamatok tárolására.
    - *p* csővezeték. Processzek közti kommunikáció.
    - *b* blokkeszköz meghajtó. Olyan eszközök, amelyek az adatokat blokkonként kezelik. Pl. merevlemezek.
    - *c* karaktereszköz meghajtó. Olyan eszközök, amelyek karakterenként kezelik az adatokat. Pl. virtuális terminálok.

      `drwxr-xr-x 1 root root 0 Apr 11 21:58 upstart`

Könyvtár tartalmának részletes listázása folyt...

rwxr-xr-x

- A következő 9 karakter a jogosultságot mutatja, amelyekkel szabályozva van ki mit tehet az adott állománnyal. Részleteket később.

```
drwxr-xr-x 1 root root    0 Apr 11 21:58 upstart
```

- Következő számérték jelzi, hogy az adott állományra hány szimbolikus link mutat.

```
drwxr-xr-x 1 root root    0 Apr 11 21:58 upstart
```

- Ezt követi a tulajdonos felhasználói neve.

```
drwxr-xr-x 1 root root    0 Apr 11 21:58 upstart
```

- Ezután az állomány elsődleges csoportja következik.

```
drwxr-xr-x 1 root root    0 Apr 11 21:58 upstart
```

- Az állomány mérete a következő

```
drwxr-xr-x 1 root root    0 Apr 11 21:58 upstart
```

- Ezt az időbélyeg követi

```
drwxr-xr-x 1 root root    0 Apr 11 21:58 upstart
```

- És végül az állomány neve

```
drwxr-xr-x 1 root root    0 Apr 11 21:58 upstart
```

Állományok másolása

- *cp <forrás> <cél>*
- Néhány kapcsoló:
  - *-v* : „bőbeszédű” mód.
  - *-i* : ha a célterületen már létezik az adott névvel állomány, rákérdez a felülírásra.
  - *-n* : a felülírást alapértelmezetten elutasítja.
  - *-r* : könyvtár és teljes tartalmának másolása.

Fájlok áthelyezése

- *mv <forrás> <cél>*
- Az áthelyezés során az eredeti fájl törlődik. Ehhez speciális jog kell. Ennek hiányában az áthelyezés hibaüzenettel fog leállni és nem teljesül. (Permisson denied)
- Ha a cél esetén a forrásfájl nevét módosítva adjuk meg, akkor az áthelyezés során a fáj más néven kerül áthelyezésre.
- Ha a forrás és célkönyvtárak megegyeznek akkor egyszerű átnevezés történik.
- Kapcsolók:
  - *-v* : „bőbeszédű" mód.
  - *-i* : rákérdezés felülírás előtt.
  - *-n* : alapértelmezetten elutasítja a felülírást.

Állományok létrehozása, törlése

- *touch <fájlnév>* : üres fájl létrehozása. Nem feltétlenül kell ott álljunk ahová a fájlt létre szeretnénk hozni (elérési útvonalak…).
- *mkdir <könyvtár neve>* : könyvtár létrehozása. Nem feltétlenül kell ott álljunk ahol a mappát létre szeretnénk hozni (elérési útvonalak…).
- *rm <fájlnév>* : fájl törlése. *-i* kapcsolóval rákérdez a fájl törlése előtt. Tömeges törlés esetén hasznos lehet.
- *rmdir <könyvtár neve>* : *üres* könyvtár törlése.
- *rm -r <könyvtár neve>* : könyvtár törlése. Ha van benne tartalom, akkor azok is törlésre kerülnek. *-i* kapcsoló használata hasznos lehet ebben az esetben.

Ajánlott parancsok

- *cd <elérési útvonal>* : könyvtárváltás
- *ls* : könyvtár tartalmának listázása (és kapcsolói…).
- *cp <forrás> <cél>* : állományok másolása.
- *mv <forrás> <cél>* : állományok áthelyezése.
- *touch <fájlnév> :* új szöveges fájl létrehozása.
- *mkdir <könyvtár neve>* : könyvtár létrehozása.
- *rm <fájlnév>* : fájl törlése.
- *rm -r <könyvtár neve>* : nem üres könyvtár törlése.
- *rmdir <könyvtár neve>* : üres könyvtár törlése.

Archiválás és tömörítés

Tömörítés és archiválás

- *Archiválás*: több különböző fájlból készítünk egy összetett fájlt.
- *Tömörítés*: egy fájlból készítünk egy kisebb méretű állományt, az adatok megtartása mellett.
- Mikor hasznosak a fenti műveletek?
  - Több állományból egy állományt készítve a mozgatás könnyebbé válik.
  - Naplófájlok kezelése tömörítéssel.
  - Biztonsági mentések esetén.
  - Lassú hálózaton keresztül val továbbítás előtt célszerű lehet a tömörítés.

Állományok tömörítése

- Redundanciák megszüntetése az adathalmazban úgy, hogy megfelelő módon visszaállíthatók legyenek az eredeti formára.
- Két tömörítési típus:
  - Veszteségmentes: amint a tömörített állományt visszaállítjuk, az megegyezik az eredetivel.
  - Veszteséges: a kitömörített állomány nem egyezik meg az eredetivel (veszteségi hányados…).
- A „tömörítés újratömörítése" kérdése…
- Az egyik legismertebb tömörítő alkalmazás Linux alatt a *gzip* (*Lempel–Ziv algoritmus…*).

A gzip (és társainak) használata

- gzip <állomány neve> : állomány betömörítése.
- -l kapcsoló : információ a tömörítés arányáról.
- gunzip <tömörített állomány> : gzip-el tömörített állomány kitömörítése (megegyezik a gzip -d paranccsal…)
- bzip2 <állomány neve> : másik tömörítési eszköz. Esetenként jobb tömörítési arányt produkál a gzip-nél.
- bunzip2 -z <állomány neve> : az előzővel megegyező tömörítési eljárás. Mindkettő a *Burrows-Wheeler* eljárást használja.
- bzip2 -d <tömörített állomány>
  bunzip -d <tömörített állomány>

  } A kitömörítés…

Fájlok archiválása

- Az archiválás lényege, hogy több állományból készítünk egy állományt a könnyebb mozgathatóság érdekében. Az erchivum természetesen később visszaállítható.
- A legismertebb archiváló alkalmazás a *tar* (Tape Archiver).
- A *tar* parancs módjai:
  - Állományokból archivum készítése.
  - Létező archivum kibontása.
  - Létező archivum tartalmának megtekintése kibontás nélkül.

A *tar* parancs

- Több állományból archívum készítése:

  tar <kapcsolók> <archívum neve> <állomány$_1$,állomány$_2$,…állomány$_n$>

  -c : create, azaz archívum készítése.

  -f : a leendő archívum neve.

  -z : az elkészített állományt a *gzip* alkalmazással be is tömöríti.

  -j :az elkészített állományt a *bzip,bzip2* alkalmazással be is tömöríti.

- Egyéb kapcsolói:

  -t : az archívum tartalmának megtekintése.

  -x : Létező archívum kibontása.

  -v : „bőbeszédű” mód

Zip fájlok

- Windows-os környezetben a legismertebb tömörítési eljárás.
- Linux alatt is elérhető.
- *gzip* vs. *zip* :
  - A *gzip* hatékonyabban tömörít mint a *zip*.
  - A *zip* archivál **ÉS** tömörít, a *gzip* csak tömörít.
  - Egy *zip*-elt archívumbók **könnyedén** kivehető egy fájl megtekintésre, ellentétben egy *gzip* és *tar* által elkészített archívumból.
  - A *zip* inkább windows-os környezetben népszerű, míg a *gzip* inkább Linux alatt.

A zip parancs

- Tömörített archívum készítése zip-el:

  zip <archívum neve> <archiválandó és tömörítendő állományok>

- Kapcsolói:

  -r : a könyvtárakat is a tartalmukkal együtt betömöríti.

Csövek, átirányítások és reguláris kifejezések

Csővezeték

- Alapesetben egy parancs kimenete vagy a hibaüzenet általában a képernyőn jelenik meg. Ezt meg lehet változtatni.
- A „|" (AltGr+w)karakter segítségével egy parancs kimenetét át tudjuk adni egy másik parancsnak bemenetként.
- *head <szöveges állomány>* : megjeleníti a szöveges állomány **első** 10 sorát.

  head -5 <szöveges állomány> : csak 5 sort…

  head -n -5 <szöveges állomány> : az utolsó 5 sor kivételével az összeset…
- *tail <szöveges állomány>* : megjeleníti a szöveges állomány **utolsó** 10 sorát.

  tail -5 <szöveges állomány> : csak 5 sort…
- Példa a fentiek használatára:

  ls -l /etc | head

  ls -l /etc | head -5 : csak az első 5 sor jelenik meg.
- *nl* : szöveges kimenet esetén e sorok meg lesznek számozva.

Be és kimenet átirányítása

- Az I/O átirányítás azt jelenti, hogy a különböző parancssori kimenetek átkerülnek különböző stream-ekre.
- A stream-ek fajtái:
  - ***STDIN*** (stream #0) vagy standard input. Parancssori adatok bevitelére szolgál, ami alapesetben a billentyűzet de lehet pl. fájl is.
  - ***STDOUT*** (stream #1) vagy standard output. A parancsok kimenete. Alapesetben ezek a képernyőre kerülnek.
  - ***STDERR*** (stream #2) vagy standard error. A szabványos hibacsatorna, ami alapesetben a képernyőre íródik ki.
- A fenti stream-ek átirányíthatók. Így pl. megadható, hogy a szabványos kimenet ne a képernyőre hanem egy fájlba legyen irányítva. Ehhez a „<" és „>" karaktereket használjuk.

Az STDOUT átirányítása

- Példa:

  echo „szöveg" > proba.txt

- Fontos tudni, hogy a szimpla nyíl („>") alkalmazásával felülírás történik ha a fájl (próba.txt) nem üres.
- Példa:

  echo „szöveg" >> proba.txt

- A dupla nyíl („>>") segítségével hozzáfűzés történik a tartalomhoz, ha a fájl (próba.txt) nem üres.

Az STDERR átirányítása

- A szimpla nyíl használata esetén az STDOUT az alapértelmezett, így ha az STDERR-t szeretnénk átirányítani akkor arra hivatkozni kell:

  `ls /proba 2> error.txt`

- A fenti parancs nem a szabványos kimenetet irányítja át az error.txt-be, hanem a hibaüzenetet („2>") mert a /proba állomány nem létezik.

Több stream egyidejű átirányítása

- Ha az STDOUT-ot fájlba és az SRDERR-t képernyőre akarjuk irányítani:

```
ls /proba /etc/network/interfaces > interfaces.txt
```

- Ha az STDOUT-ot képernyőre és az SRDERR-t fájlba akarjuk irányítani:

```
ls /proba /etc/network/interfaces 2> error.txt
```

- Ha az STDOUT-ot is és az SRDERR-t is ugyanabba a fájlba akarjuk irányítani:

```
ls /proba /etc/network/interfaces &> all.txt
```

- Ha az STDOUT-ot is és az SRDERR-t is más-másfájlba akarjuk irányítani:

```
ls /proba /etc/network/interfaces > interfaces.txt 2> error.txt
```

Az STDIN átirányítása

- Használata meglehetősen ritka. Egy alkalmazására álljon itt egy példa:

  tr parancs: karakterek lecserélése, tömörítése és/vagy törlése. A tr parancs alapértelmezésben a szabványos bemenetről olvas. Argumentumként nem adható meg elérési út.

  A feladat az, hogy a proba.txt szöveges állományban lévő összes kisbetűt lecserélje nagybetűre.

  `tr 'a-z' 'A-Z' proba.txt` → Hibával tér vissza

  `tr 'a-z' 'A-Z' < proba.txt` → helyesen lefut

  A ki és bemenet átirányításának kombinálása:

  `tr 'a-z' 'A-Z' < proba.txt > proba2.txt`

Fájlok keresése

- A *find* parancs szintaktikája:

  find [starting directory] [search option] [search criteria] [result option]

- Magyarázat:
  - *[starting directory]* : a kezdő könyvtár, ahol az adott fájlt keresni szeretnénk („/" esetén az egész rendszerben keres…).
  - *[search option]* : a keresési opciókkal meghatározzuk, hogy mi alapján keresünk (fájlnév, méret, tulajdonos, stb…).
  - *[search criteria]* : keresési feltétel. Pl. ha fájlnév alapján keresünk itt adjuk meg a fájl nevét (helyettesítő karakterek…).
  - *[result option]* : mi történjen ha megvan a keresés eredménye. Alapértelmezésben az eredmény az STDIN-en jelenik meg (képernyő).

A *find* parancs használata

- Név alapján keresés:

  find /etc -name hosts

- A fenti példa hozhat találatokat és hibaüzeneteket vegyesen, mert bizonyos könyvtárakban való keresési jog nem engedélyezett az adott felhasználónak.
- Az esetleges hibaüzenetek kiszűrése:

  find /etc -name hosts 2> /dev/null

- A */dev/null* állomány a Linux rendszerben a „fekete lyuk”. Mindent képes elnyelni anélkül, hogy a mérete változna.
- Fájlok keresése során a találatokban szereplő állományok részleteinek megjelenítése:
- *find /etc -name hosts -ls 2> /dev/null*

A *find* parancs használata

- Méret alapján keresés:

  find /etc -size 10c -ls 2> /dev/null

- Pontosan 10 bájt nagyságú állományok keresése a /etc mappában.

  c bájt

  k kilobájt

  M megabájt

  G gigabájt

- Egy megadott méretnél nagyobb (+) vagy kisebb (-) állományok keresése:

  find /usr -size +100M -ls 2> /dev/null

A *find* parancs használata

- Egyéb hasznos keresési opciók:

  -maxdepth : milyen mélységben keressen. Pl. *-maxdepth 1* az adott könyvtárban és annak közvetlen alkönyvtáraiban keres.

  -group : csoportnév alapján keres.

  -iname : név alapján keres. A megadott fájlnév esetén nem tesz különbséget a kis és nagybetű közt.

  -mmin : az utolsó módosítás minimális ideje. Pl. *-mmin 10* jelentése, a fájl adatai utoljára 10 perce lettek módosítva.

  -type : fájltípus alapján keres. Pl. *-type f* az általános fájlokra keres.

  -user : tulajdonos alapján keres.

- A fenti opciókból egyszerre többet is lehet használni a hatékony keresés érdekében:

```
find /etc -size 10c -type f -ls 2> /dev/null
```

Lapozó parancsok

- Kisebb szöveges állományok tartalmának megtekintésére elegendő a *cat <fájlnév>* parancs.
- Ha a tartalom több oldalas akkor két „lapozó" parancs ajánlott:
  - *less <fájlnév>* : könnyű használat és fejlett funkciók jellemzik. Fejlett súgó funkciója van („*h*" billentyű).
  - *more <fájlnév>* : kevesebb funkció található benne mint a *less*-ben, viszont minden Linux disztribúcióban benne van a kezdetek óta. Súgó funkció itt is van („*h*" billentyű).

A *less* és a *more* beépített súgó funkciói

- Mindkét parancs esetén a súgó előhívása egyaránt a „*h*" billentyűvel történik.
- Az alábbi billentyű–parancsok mind a *more* mind a *less* parancs esetén egyaránt működnek:

  SPACE : egy ablakkal tovább lapoz

  b : egy ablakkal vissza lapoz

  ENTER : egy sorral tovább lép

  q : kilépés (quit)

  h : súgó menü

- A *less* parancs esetén lehetőség van szövegrész keresésére is. Ehhez használjuk a „*/*" billentyűt! (Reguláris kifejezések használata…)

A *head* és a *tail* parancsok

- Szöveges állomány *első 10* sorának megjelenítése:

  head <állomány neve>
- Szöveges állomány *utolsó 10* sorának megjelenítése:

  tail <állomány neve>
- Szöveges állomány *első 7* sorának megjelenítése:

  head -7 <állomány neve>
- Szöveges állomány *utolsó 7* sorának megjelenítése:

  tail -7 <állomány neve>
- Szöveges állomány összes sorának megjelenítése az *utolsó* 5 kivételével:
- *head -n -5 <állomány neve>*
- Szöveges állomány összes sorának megjelenítése az *első* 5 kivételével:
- *tail -n +5 <állomány neve>*
- *tail -f <állomány neve>* : ekkor nem adja vissza a promptot. Valós időben jelzi az esetleges módosításokat. (log fájlok)

A *sort* parancs

- Szöveges állomány sorainak rendezésére szolgál:
  sort <állomány neve>
- Ha a szöveges állomány soraiban az adatok oszlopos elrendezésben vannak tárolva, a határoló karaktert megadva lehetőség van kiválasztani,melyik oszlop szerint szeretnénk rendezni:
  sort -t: -n -k3 /etc/passwd

  -t kapcsoló után a határoló karaktert adjuk meg

  -n jelzi, hogy számokat tartalmazó adatokat kell rendezzünk

  -k3 jelzi, hogy a balról a 3. oszlop adatai szerint történjen a rendezés.

A *sort* parancs

- Fordított sorrendben rendezés:

  `sort -t: -n -r -k3 /etc/passwd`

  `-r` fordított sorrendben rendez

- Ha nem csak egy oszlop alapján szeretnénk rendezni:

  `sort -t: -k2 -k1 -k3n /etc/passwd`

A wc paranccsal

- Statisztikát ad egy (vagy több) szöveges állományról:

  wc /etc/passwd /etc/shadow

  - Sorok száma
  - Szavak száma
  - Karakterek (byte-ok) száma
- Kapcsolói:

  -l csak a sorok száma

  -w csak a szavak száma

  -c csak a karakterek (byte-ok) száma

A *cut* parancs

- Szöveges állomány tartalmának megjelenítése esetén oszlopok vagy karakterpozíciók alapján szűr.
- Pl:

  cut -d: -f1,5-7 /etc/passwd

- Karakterpozíciók alapján:

  ls -l / | cut -c1-11,47-

- Kapcsolók:

  -d : oszlophatároló karakter megadása

  -f : megjelenítendő oszlop(ok) megadása

  -c : karakterpozíció(k) megadása

A *grep* parancs

- Szöveges állomány tartalmának szűrése.

  grep bash /etc/passwd

  A */etc/passwd* fájl azon sorainak megjelenítése, amelyben szerepel a ***bash*** kifejezés.

- Kapcsolói:

  --color : színnel jelöli a keresett szöveget.

  -c : a találati sorok számét adja vissza.

  -n : a találati sorok előtt megjelenik a kiinduló fájlban elfoglalt helyük sorszám alapján.

  -v : azon sorok megjelenítése, amelyek ***NEM*** tartalmazzák a mintaszöveget.

  -l : azon fájlok listázása, amelyek tartalmazzák a megadott kifejezést.

  -i : a találati fájlok összes olyan sorának listázása, amelyben a keresett szöveg megtalálható és elhagyja a kis és nagybetűk megkülönböztetését.

  -w : csak azokat a sorokat adja meg, amelyekben az illeszkedés teljes szavakból származik.

Reguláris kifejezések

- Olyan normál és speciális karakterek kollekciója, amelyeket a karaktersorozatokból álló mintákban használunk.
- Néhány alapvető reguláris kifejezés:
  - `.` (pont) : bármelyik karakter (de csak egy…).
  - `[]` : karakter tartomány, amely egy karakterre illeszkedik.
  - `*` : az előtte lévő karakter 0 vagy többszöri ismétlődése.
  - `^` („kalap”) : az őt követő karaktereknek a sor elején kell megjelenniük.
  - `$` : az őt megelőző szövegrésznek a sor végén kell megjelenjen.

Reguláris kifejezések (példák)

- `grep --color 'a..' example.txt`

  Megjeleníti azokat a sorokat (színnel kiemelve), amelyek az „a" karakterrel kezdődnek. A kiemelés az „a" karakter utáni két karakterre is vonatkozik.

- `grep --color '[ab][a-d]' example.txt`

  Megjeleníti azokat a sorokat (színnel kiemelve), amelyek az „a" vagy „b" karakterrel kezdődnek és a második karakter „a", „b", „c", „d" karakterek valamelyike. A kiemelés az „a" karakter utáni második karakterre is vonatkozik.

  `man ascii` → ascii karaktertábla megjelenítése.

- `grep --color '[^abc]d' example.txt`

  Azokat a mintákat emeli ki, amelyek az „a", „b" vagy „c" egyikével *SEM* kezdődnek, de a második karakter a „d".

Reguláris kifejezések (példák)

- *grep --color 'd*' example.txt*

  Azokat a mintákat emeli ki, amelyek a „d" karakterrel kezdődnek, és utána a „d" karakter 0-szor vagy többször fordul elő.

- *grep --color "^a" example.txt*

  Mindazon sorokat jeleníti meg, amelyek az „a" karakterrel kezdődnek.

- *grep "c$" example.tx*

  Mindazon sorokat jeleníti meg, amelyek az „c" karakterrel végződnek.

- *grep --color "cd*" example.txt*

  A „\" („vissza per") karakter feloldja az őt követő reguláris kifejezés hatását és normál karakterként lesz értelmezve.

Reguláris kifejezések (példák)

- A *grep* parancs egyik változata az *egrep* (EXTENDED grep). Az *egrep* parancs létező és futtatható is de meg is egyezik a *grep -E* paranccsal.
- Az *egrep* (*grep -E*) által használt néhány reguláris kifejezés:

  ? : passzol az őt megelőző karakter 0 vagy egyszeri előfordulására.

  + : passzol az őt megelőző karakter egyszeri vagy többszöri előfordulására.

  / (Alt Gr+w) : logikai VAGY-ként funkcionál.

Az *xargs* parancs

- Ha egy parancs esetén túl hosszú a paraméter lista, a parancs a „Argument list too long” hibaüzenettel leáll.
- Az *xargs* parancs részekre bontja a paraméterek listáját így megakadályozva a fenti hibát.
- Másik nagy előnye, ha egy állomány nevében pl whitespace karakterek vannak és ezeket a normál parancsok nem képesek kezelni. Ezt a problémát is fel lehet oldani a fenti paranccsal.
- Pl:

  ls | xargs rm

  Törli annak a mappának a tartalmát amelyben a parancsot kiadtuk.

Vége