Linux

Azonosítás

Belépés

A Linuxos rendszer indulása után egy „login” várakozási jel jelenik meg egy kettősponttal. A login nem egy hibaüzenet, a felhasználónév beírását várja a rendszer:

```
login: geza
```

Ha a rendszerben a mi felhasználónevünk a geza, akkor azt a login: után beírjuk. A felhasználónév beírása után a password: várakozási jelet látjuk, amely a jelszavunk beírásra szólít fel:

```
password:
```

A jelszó beírása esetén alapértelmezetten még csillagok sem jelentek meg.

Jelszó megváltoztatása

```
$ passwd
```

A felhasználó a passwd paranccsal megváltoztathatja saját jelszavát.

Mások jelszavát csak a rendszergazda azaz a root felhasználó képes megváltoztatni. Ekkor meg kell adni az adott felhasználónevet a passwd parancs paramétereként.

A geza felhasználó jelszavának megváltoztatása

```
# passwd geza
```

Kijelentkezés

```
$ logout
```

A kijelentkezés parancsa még az **exit**, vagy a **Ctrl+D** billentyűkombináció.

A Linux terjesztések lehetővé teszik grafikus bejelentkezést is. Erre a célra XDM, KDM vagy a GDM csomag valamelyikének telepítése ad lehetőséget. Mi azonban most nem érintjük a grafikus felület használatát.

Megjegyzés

Ha valaki mégis feltelepítette a grafikus felületet és szeretne parancssorra átkapcsolni, a következőket kell tudnia. Egy linuxos rendszer alapértelmezetten 6 azaz hat darab karakteres (vagy konzolos) felületen ad belépési lehetőségeket. Ezeket terminálnak hívjuk. Vagyis van hat darab terminálunk. Indulás után az 1-s terminálon vagyunk. A többi terminált az F2, F3, stb. funkcióbillentyűkkel érhetjük el, egészen F6-ig, a lenyomott „Alt” billentyű mellett. A második terminál tehát ezzel a billentyűkombinációval érhető el: Alt+F2. A harmadik: Alt+F3, stb. A Linuxos rendszerek lehetővé teszik ugyancsak hat darab grafikus felülete elindítását is. A grafikus felület és egy XDM, KDM vagy GDM telepítése után a hat grafikus felületből egy automatikusan elindul. Ha

karakteres terminálban vagyunk ezt az Alt+F7 billentyűkombinációval érhetjük el. Grafikus felületről karakteresre kapcsolás esetén kell a Ctrl billentyű is. Egyes karakteres terminálra grafikus felületről tehát így kapcsolunk: Ctrl+Alt+F1. Ugyanígy érjük el a többit is. Persze ha már karakteres felületen vagyunk, akkor elég az Alt billentyű.

Grafikus felületen egy kis ablakban is indíthatók úgynevezett virtuális terminálok. Ebből végtelen sokat indíthatunk. Ha GNOME grafikus környezetet telepítettünk, akkor a virtuális terminálunk lehet rendszergazdai és felhasználói. Elérésük:

- *Alkalmazások → Kellékek → Terminál*
- *Alkalmazások → Kellékek → Rendszergazda terminál*

Várakozási jel

Ha bejutottunk a rendszerbe egy várakozási jel illetve karaktersorozat jelenik meg, idegen szóval: prompt. A várakozási jel más lehet a rendszergazda és felhasználó számára. Felhasználók esetén a szokásos alak tartalmazza a felhasználó nevét, a Linuxos rendszer nevét, melyik könyvtárban áll és végül a parancsértelmezőre jellemző karakter.

Felhasználó várakozási jele

```
geza@debian:~$
```

A fenti példa Debian GNU/Linux operációs rendszerek alapértelmezett várakozás jele felhasználók esetén. A geza felhasználónév után egy ”@” karakter majd utána szerver neve debian. A ”@” karakter elválasztó szerepet tölt be. A szerver neve után újabb elválasztó következik, egy kettőspont (”:”). A kettőspontot az aktuális könyvtár követi, amely esetünkben egy ”~” karakter. Ez mindig felhasználó saját könyvtárát jelöli. Alapértelmezésként geza saját könyvtára a /home/geza. A dollárjel („$”) a bash nevű parancsértelmezőt jelképezi.

A root felhasználónak az alapértelmezett várakozási jele eltérő. A várakozási jelben nem szerepel a felhasználónév és az utolsó karakter egy kettőskereszt (”#”).

Root felhasználó promptja

```
debian:~#
```

Hogy mi szerepeljen a várakozási jelben egy környezeti változó határozza meg, amelynek a neve PS1. Egy környezeti változóra úgy hivatkozhatunk, ha egy dollár jelet teszünk a neve elé: $PS1. Az echo paranccsal megtekinthetjük a tartalmát.

A PS1 környezeti változó tartalma

A PS1 tartalma

```
$ echo $PS1
```

A rendszer mielőtt megjeleníti a várakozási jelet beolvassa a PS1 környezeti változó tartalmát. A várakozási jelet ez alapján állítja be. Debian GNU/Linux 5.x felhasználók esetén alapértelmezésként ez így néz ki:

```
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\u@\h:\w\$
```

Ettől a beállítástól látjuk a fentiek szerint a várakozási jelet.

X terminál színezése

Természetesen megadhatunk más beállításokat is. A környezeti változókat mindig az export paranccsal szokás beállítani, mert ez teszi lehetővé, hogy más folyamatok indítása után is érvényes legyen a beállítás. Az alábbi beállítás a terminál színezését mutatja be:

```
export PS1='${debian_chroot:+($debian_chroot)}\[\033[01;34m\]\u@\h\
[\033[00m\]:\w\$ '
```

Vegyük észre, hogy az export parancs után írt PS1 környezeti változó elé nem írtunk „$” karaktert. Fentebb említettük, hogy erre csak akkor van szükség ha hivatkozunk rá, tehát példának okárét kiíratjuk azt.

Egyszerű prompt

A fenti prompt nagyon hosszú tud lenni ha van egy nagyon hosszú könyvtárútvonalunk. Ilyenkor be szoktam állítani egy egyszerű várakozási jelet, amely csak egy dollárjelet meg egy szóközt jelenít meg.

```
$ export PS1="$ "
```

Akkor használjuk, ha az útvonal nagyon hosszú szeretnénk egy egyszerű várakozási jelet.

Ezek a beállítások persze nem véglegesek, újraindítás után a környezeti változó tartalma eredetire visszaáll.

Ha a fentieket szeretnénk véglegesíteni akkor a /home/geza/.bashrc állományban írjuk a fenti sorokat: **Például**

```
$ echo "export PS1='$ '" >> /home/geza/.bashrc
```

vagy

```
$ echo "export PS1='$ '" >> ~/.bashrc
```

Az utóbbi példából azt látjuk hogy a /home/geza helyett használhatom a hullámjel karaktert ”~” is, a saját könyvtárunk meghatározására.

Könyvtárkezelés

Az első érdekes információ lehet számunkra, hogy melyik könyvtárban állunk jelenleg. Ezt a **pwd** parancs begépelésével tehetjük meg.

Aktuális könyvtár

```
$ pwd
```

Ha már tudjuk melyik könyvtárban állunk, az is fontos lehet, hogy az aktuális könyvtárban milyen újabb könyvtárak vagy állományok vannak. Ezt az ls paranccsal tudjuk megnézni. Az ls az angol list szóból származik, magyarul listázásnak fordítható.

A könyvtár tartalmának listázása

```
$ ls
```

Könyvtárak létrehozására az mkdir parancsot használjuk. A parancs első része az „mk” az angol make szóból származik. Magyarul elkészít, csinál szavakkal fordítható. A „dir” az angol directory szóból ered, amely magyarul könyvtár, vagy másik elterjedt nevén mappa néven fordítható. A Parancs tehát egy könyvtár elkészítésére utal. Paraméterkén több könyvtár is megadható.

Ügyeljünk arra, hogy a Linuxos rendszerek a kisbetű nagybetű érzékenyek. Például a „Bertabla” és a „bertalba” nevű könyvtár két különböző könyvtár.

Könyvtár létrehozása

```
$ mkdir konyvtarnev
```

A könyvtár nevekben lehetnek ékezetes betűk, azonban ennek használata nem ajánlott. Ha könyvtárakat más rendszerbe szeretnénk mozgatni, akkor azok esetleg olvashatatlanok lesznek.

Könyvtár törlése

```
rmdir konyvtarnev
```

A könyvtár átnevezését az mv paranccsal hajtjuk végre, amely az angol move szóból származik. Magyarul mozgatás jelent. A parancsot valójában arra találták ki, hogy egy állományt egy másik helyre mozgassunk. Viszont ha a mozgatás helye a kiindulási pont, a parancs akkor is működik vagyis átnevezés történik.

Könyvtár átnevezése

```
$ mv konyvtar1 konyvtar2
```

A konyvtar1 nevű könyvtárat konyvtar2-re nevezzük át.

Könyvtár átmozgatása

```
$ mv konyvtar1 konyvtar2/
```

A konyvtar1 nevű könyvtárat a konyvtar2 nevű könyvtárba mozgatja.

Ha már vannak könyvtáraink szeretnénk azokban mozogni is, azt is mondhatnánk szeretnénk könyvtárat cserélni. Erre a célra a cd parancs szolgál. A cd parancs a Change Directory szavak kezdőbetűi. A Directory már ismerős, a Change magyar fordítása cserélni. Ha egy könyvtárba szeretnénk belépni, akkor a cd parancsnak egyszerűen meg kell adni a könyvtár nevét paraméterként:

```
$ cd konyvtar1
```

Ezzel a konyvtar1 nevű könyvtárba kerülünk.

Könyvtár szerkezet

A Linuxos könyvtárszerkezet felépítése egy fához hasonlítható. Van egy gyökér a kiinduláspont, és abból újabb könyvtárak, alkönyvtárak nyílnak. A kiindulópontot azaz a gyökeret egyetlen karakterrel jelezzük: ”/”. A perjel tehát a gyökér. Ha a merevlemez több részre ún. partícióra van felosztva, gyökér akkor is csak egy van, így nem kell különféle jelöléssekkel ellátni a gyökér könyvtárat. Jó kérdés lehet, hogy akkor a többi partíciót hol és hogyan érjük el. A plusz partíciók egy-egy könyvtár alkönyvtáraiban érhetők el. A rendszergazda azokat bárhova felcsatolhatja, igaz van mindegyiknek egy megszokott helye.

A könyvtárak leírása

A felhasználók saját könyvtárai a „home” nevű könyvtár. Hogy ez rögtön a gyökér könyvtárból nyílik ezt így szoktuk leírni:

```
/home
```

A /home könyvtáron belül vannak a felhasználók könyvtárai. Legyen például egy joska, egy mari és egy janos nevű felhasználó. A joska felhasználói könyvtárát ekkor így írhatjuk le:

```
/home/joska
```

Mari felhasználó könyvtára:

```
/home/mari
```

János felhasználó könyvtára:

```
/home/janos
```

Fa formájú szerkeztben ezt így írhatjuk le:

```
/
|--home
      |--janos
      |--joska
      |--mari
```

Ha Jánosnak van a saját könyvtárában egy Dokumentumok alkönyvtár, akkor akkor a faszerkezet így néz ki:

```
/
|--home
      |--janos
      |     |--Dokumentumok
      |--joska
      |--mari
```

Itt szeretném megjegyezni, hogy a Linuxos fájlrendszer kis és nagybetű érzékeny. Tehát egy „dokumentumok” és egy „Dokumentumok” nevű könyvtár két különböző könyvtár.

További példa kedvéért Jánosnak legyen a Dokumentumok mellett egy Tervek és egy a tervekből nyíló Programozas könyvtára:

```
/
|--home
      |--janos
      |     |--Dokumentumok
      |     |--Tervek
      |           |--Programozas
      |--joska
      |--mari
```

A gyökérkönyvtárból is több könyvtár nyílik, a következő példában legyen egy „var” könyvtár amelyben a rendszer a változó adatokat tárolja:

```
/
|--home
|     |--janos
|     |     |--Dokumentumok
|     |     |--Tervek
|     |           |--Programozas
|     |--joska
|     |--mari
|--var
```

A „var” könyvtárból nyíljon egy „log” nevű könyvtár amelyben a naplófájlok vannak:

```
/
|--home
|     |--janos
```

```
|    |    |--Dokumentumok
|    |    |--Tervek
|    |          |--Programozas
|    |--joska
|    |--mari
|--var
    |--log
```

Ezzel a faszerkezettel tehát leírható az egész könyvtár hierarchia.

A „log” könyvtár útvonalát egyetlen sorba ezek után így ábrázolhatjuk:

`/var/log`

A János nevű felhasználó Dokumentumok könyvtára egyetlen sorban:

`/home/janos/Dokumentumok`

A János nevű felhasználó „Programozas” könyvtára így írható le:

`/home/janos/Tervek/Programozas`

A János nevű felhasználó „Tervek” könyvtára így írható le:

`/home/janos/Tervek`

Esetleg az egyértelműbb jelzés miatt a könyvtár nevek végére tehetünk egy ”/” karaktert:

```
/var/log/
/home/janos/Dokumentumok/
/home/janos/Tervek/Programozas/
/home/janos/Tervek/
```

Így tudjuk, hogy nem egy fájlról van szó.

A gyökér könyvtár könyvtárai

Most nézzük milyen könyvtárak találhatók a gyökérkönyvtárban.

- /bin A alapparancsok programjai itt vannak elhelyezve
- /boot A rendszerindításhoz szükséges fájlok helye
- /cdrom Ha kézzel csatolunk egy CD-ROM-t a fájlrendszerbe, akkor azt ide szokás
- /dev A számítógép minden eszköze (hardvere) ebben a könyvtárban egy fájl formájában leképezésre kerül
- /etc A rendszer beállításait tároljuk itt
- /home A felhasználók saját
- /lib Osztott programkönyvtárak, elsősorban a kernel, a /bin és az /sbin könyvtárak számára
- /lost+found Fájlrendszer sérülése után a helyreállító program ide helyezi azokat a fájlokat amit nem képes helyre állítani.
- /media Más fájlrendszerek automatikusan ide kerülnek felcsatolásra.
- /mnt Kézzel csatlakoztatott más fájlrendszerek
- /proc A memória leképezése fájlokra
- /root A root (rendszergazda) felhasználó saját könyvtára
- /sbin A rendszergazda parancsainak programjai itt találhatók
- /selinux A selinux szolgáltatás beállításai
- /srv Néhány program specifikus fájl, amelyek a rendszert szolgálják
- /sys Néhány kernlobjektum itt van leképezve
- /tmp Ideiglenes fájlok helye
- /usr A programok könyvtárai

        - /usr/bin/ : Same as for top-level hierarchy
        - /usr/include/ : Az programozáshoz szükséges „include” fájlok helye
        - /usr/lib/ : Osztott programkönyvtárak
        - /usr/sbin/ : A root felhasználók programjai
        - /usr/share/ : Architektúra független adatok
        - /usr/src/ : Forráskódok. (Debian csomagok fordításához; Lásd még a /usr/local/src/ könyvtárat)
        - /usr/X11R6/ : Az „X Window System” könyvtára
        - /usr/local/ : A rendszer-adminisztrátor által telepített adatok telepítési helye
            - /usr/local/bin : helyi bináris programok és scriptek helye, stb.
            - /usr/local/src : Forráskódok helye (Nem debianos eszközök telepítési helye)
    - /var Gyakran változó fájlok helye

Link: http://wiki.debian.org/FilesystemHierarchyStandard

Fájlkezelés

Fájlok létrehozása

Egy fájl létrehozásához a legegyszerűbb parancs a touch. Igaz eredeti célja, hogy egy fájl időbélyegét megváltoztassuk, de ha egy állomány nem létezik amelynek időbélyegét aktualizálni szeretnénk, akkor azt létrehozza. Természetesen a fájl tartalma üres lesz. A touch parancs nem kérdez semmit, csak elkészíti az állományt:

```
$ touch filenev.txt
```

Az echo paranccsal is létrehozhatunk fájlokat. Ekkor az echo után írt „Szöveg” a fájlba kerül.

```
$ echo "Szöveg" > filenev.txt
```

Fájlok törlése

```
$ rm filenev
```

Fájlok átnevezése

```
$ mv file1 file2
```

Az mv parancs valójában a fájlok mozgatására lett kitalálva, de ha nem adunk meg könyvtárat, vagyis ugyanabba a könyvtárba mozgatunk, akkor tulajdonképpen átnevezés történik.

Fájlok mozgatása

```
$ mv file konyvtar/
```

A file nevű állományt a konyvtar nevű mappába mozgatjuk.

Fájlok tartalmának megtekintése

```
$ cat filenev
```

Fájlok darabolása

```
$ split -b 1440000 filenev
```

A filenev nevű fájlt 1440000 byte méretű fájlokra darabolja. A fájlok rendre xaa, xab, xac, stb. lesznek.

Darabolt fájlok összefűzése

```
$ cat x* >> filenev
```

Fájlok szerkesztése

Fájlok szerkesztésére nagyon sok eszköz áll rendelkezésre. Lehet parancssorból a sed paranccsal szerkeszteni. Vagy lehet erre a célra kitalált szövegszerkesztőket használni. Legyen például a virag.txt nevű állomány amit szeretnénk szerkeszteni:

```
vi virag.txt
```

vagy

```
nano virag.txt
```

vagy

```
mcedit virag.txt
```

Billentyűzet használat

Terminal

Egy billentyűzet és egy monitor, amely egy számítógéphez kapcsolódik.

Eredetileg a nagygépes Unix rendszerekhez több terminál is csatlakozott. Esetleg egy PC-t is lehetett terminálként használni. A Linuxot ma leggyakrabban nem nagygépre, hanem egy PC-re telepítjük. A PC-hez pedig egyetlen monitor és billentyűzet kapcsolódik alapesetben. Mi ezt tekintjük most terminálnak.

Host

Az a nagyszámítógép, amelyhez a terminálok kapcsolódnak. Host számítógép alatt mi most PC-t fogjuk érteni, amelyen dolgozunk.

Teletype ASR33

Az első Unix terminálok. Rövidítve TTY.

Terminál nevének kiíratása:

```
tty
```

Terminál paramétereinek beállítása:

```
stty
```

Paraméter nélkül kiírja a Linux milyen beállításokkal használja a terminált.

A kisbetűk nagybetűvé alakítása:

```
stty olcuc
```

A kisbetűk nagybetűvé alakításának kikapcsolása:

```
stty -olcuc
```

További információkért lásd a man stty parancsot.

Milyen terminált használunk:

```
echo $TERM
```

Parancsok megtalálása

Jó néhány parancsot megismertünk, de általában szeretnénk a parancsokról többet megtudni. Esetleg azt is szeretnénk tudni, hogy a parancsokat megvalósító programok valójában hol találhatók. Esetleg szükségünk van egy parancsra de azt sem tudjuk hol, milyen néven keressük. A következő fejezet ezeket a problémákat járja végig.

Keresünk egy parancsot

Valamihez szeretnék parancsot találni. Például könyvtárkezeléshez keresek egy parancsot. Ha telepített rendszer magyar, akkor kereshetek magyar szavakra is. A parancsok kereséséhez egyik lehetséges program az apropos. Használata a könyvtárkezeléssel kapcsolatos parancsok keresése esetén például:

```
apropos könyvtár
```

A program kiírja azoknak a programoknak a neveit amelyeknek a leírása tartalmazza a könyvtár szót.

```
apropos könyvtár
basename (1)         - levágja a könyvtárat és a végződést a fájlneve...
chroot (1)           - parancs futtatása eltérő gyökérkönyvtárral
chroot (8)           - megváltoztatja a gyökérkönyvtárat és végrehajt ...
cp (1)               - fájlok és könyvtárak másolása
find (1)             - fájlokat keres egy könyvtárstruktúrában
ldd (1)              - kiírja a program által használt megosztott könyvt...
ls (1)               - könyvtárak tartalmának listázása
mkdir (1)            - könyvtár létrehozása
pwd (1)              - kiírja az aktuális (munka-) könyvtárat
rmdir (1)            - törli az üres könyvtárakat
whois (1)            - Internet user név könyvtár szolgáltatás
```

Persze nem kapjuk meg azoknak a parancsoknak a nevét amelyeknek a leírása nem lett magyarra fordítva. Ezért az angol nyelven érdemes rákeresni a könyvtár szóra:

```
apropos directory
```

Figyeljük meg, hogy a leírásból egy rövid kivonatot is közöl a parancsról, ami leírja mire való a prancs.

```
$ apropos directory
alphasort (3)        - scan a directory for matching entries
basename (1)         - strip directory and suffix from filenames
bf_compact (1)       - shell script to compact a bogofilter directory
bf_compact-bdb (1)   - shell script to compact a bogofilter directory
bf_copy (1)          - shell script to copy a bogofilter working directory
bf_copy-bdb (1)      - shell script to copy a bogofilter working directory
bf_tar (1)           - shell script to write a tar file of a bogofilter direc...
bf_tar-bdb (1)       - shell script to write a tar file of a bogofilter direc...
bindtextdomain (3)   - set directory containing message catalogs
chdir (2)            - change working directory
chroot (2)           - change root directory
chroot (8)           - run command or interactive shell with special root dir...
closedir (3)         - close a directory
dbus-cleanup-sockets (1) - clean up leftover sockets in a directory
depmod.conf (5)      - Configuration file/directory for depmod
desktop-file-install (8) - install a desktop file to the applications directory
```

```
dh_installdeb (1)    - install files into the DEBIAN directory
dh_testdir (1)       - test directory before building debian package
dir (1)              - list directory contents
dirfd (3)            - get directory stream file descriptor
dirsplit (1)         - splits directory into multiple with equal size

...

pwd (1)              - print name of current/working directory
pwdx (1)             - report current working directory of a process
readdir (2)          - read directory entry
readdir (3)          - read a directory
readdir_r (3)        - read a directory
readlinkat (2)       - read value of a symbolic link relative to a directory ...
remove (3)           - remove a file or directory
renameat (2)         - rename a file relative to directory file descriptors
rewinddir (3)        - reset directory stream
rmdir (2)            - delete a directory
...
```

A válasz most egész hosszúra sikeredett, így csak egy kivonatot másoltam ide.

Valamelyik parancs felkeltette az érdeklődésünket és többet szeretnénk róla tudni. Erre való a „man” parancs. A „man” a manual szóból van, vagyis kézikönyv. Minden parancsnak általában van kézikönyve, amely leírja annak használatát és persze mindent amit a program dokumentálója jónak látott. A listából például kiválasztjuk a ls parancsot, mert a könyvtárak tartalmának listázásáról többet szeretnénk tudni. A következő parancsot használhatjuk:

```
man ls
```

Ha elindítottuk a „man ls” parancsot a következő problémánk lehet, hogyan lépünk ki a kézikönyvből? Ehhez a „Q” billentyűt kell mindössze megnyomni és a kilépés megtörténik.

Ha tudjuk egy parancsnak a nevét, és egy rövid leírást szeretnék mire is jó, akkor használhatjuk a whatis parancsot:

```
whatis ls
```

A parancsnak érdemes kipróbálni help paraméterét is:

```
ls --help
```

Néhány parancsnak több karakterből álló paramétere csak egy kötőjellel van bevezetve: -help. De az is lehet, hogy csak egy karakter kell: -h Ez mindig az adott program írójától függ. A szabvány szerint több karakteres opciókat mindig két kötőjellel vezetjük, az egy karaktereseket egy kötőjellel.

Hol található maga a program?

Előfordulhat, hogy szeretnénk tudni hol található egy program. Például egy scriptet akarunk írni, amelyben az egész útvonalat szeretnénk felvenni. Nekiállhatnánk keresni is, de van egy parancs amelyik megmondja hol van:

```
which ls
```

A which parancsot pontosan erre a célra találták ki. A kimenet általában ez:

```
/bin/ls
```

Próbáljuk meg más parancsokkal is.

Útvonal

Parancsok útvonaláról lesz szó. Egy parancsot úgy tudunk végrehajtani, ha tudjuk melyik könyvtárban van, akkor beírjuk az útvonalát és egy <Enter>-t nyomunk. Például gcc paranccsal szeretnénk egy programot lefordítani, akkor ezt írhatjuk:

```
/usr/bin/gcc -o main main.c
```

Ehhez tudnunk kell, hogy a gcc parancs az /usr/bin könyvtárban van. Szerencsére nem kell ezt nekünk megjegyezni. A rendszer biztosít számunkra egy PATH nevű környezeti változót. Ha ez tartalmazza az /usr/bin karaktersorozatot akkor a gcc parancs kiadásánál a parancsértelmező megnézi, hogy van-e az /usr/bin/ könyvtárban gcc parancs. Ha van akkor végrehajtja. Persze a PATH változóban egyszerre több útvonal is megadható, egymástól kettősponttal elválasztva. Felhasználóként az útvonal nekem például így néz ki:

```
/home/andras/bin:/usr/local/bin:/usr/bin:/bin:/usr/games
```

Ez az le tudjuk kérdezni például az „echo” paranccsal, mint azt már fentebb tettük a PS1 környezeti változóval:

```
echo $PATH
```

Próbáljuk ki a parancsot. A parancsértelmező ezekben a könyvtárakban keresi az általunk kiadott parancsot. Ha a parancs útvonala nincs a PATH változóban, akkor kénytelen vagyunk beírni az útvonalat. Ilyen parancs lehet például az ifconfig, amely a hálózati kártyák beállítását kérdezi le. Az ifconfig parancs a /sbin könyvtárban van, ami az átlag felhasználónak nincs útvonalban. Igaz a felhasználó nem végezhet vele beállításokat, de a kártya beállításait lekérdezheti. Ehhez viszont meg kell adnia a teljes elérési utat:

```
/sbin/ifconfig
```

Felmerülhet a kérdés, mi van akkor ha egy parancs több könyvtárban is szerepel, amelyik útvonalban van. Ekkor az a parancs hajtódik végre, amelyik a PATH változóban hamarabb szerepel. Ha a másik parancsot szeretnénk, akkor meg kell adjuk a teljes elérési utat.

Archiválás

Archiválásra a **tar** parancsot használjuk. Maga a tar program nem képes tömörítésre, csak az archiválásra. De ez így is van rendjén. Az archiválás nem egyenlő a tömörítéssel. A tar parancsot nem is tömörítésre találták ki. Néhány kapcsolóval azonban a tar rávehető, hogy más programokat felhasználva tömörítést is végezzen.

Archiválás

```
$ tar -cf archivnev.tar konyvtarnev
```

Archiv csomagok kibontása

```
$ tar -xf archivnev.tar
```

Archiválás tömörítéssel

```
$ tar -czf archive.tar.gz konyvtar
```

Folyamatkezelés

Folyamatok indítása

Programok indításához ismerni kell azok helyét. A rendszerbe, annak részeként telepített programokat a következő helyeken találhatunk:

- /bin
- /sbin
- /usr/bin
- /usr/sbin
- /usr/local
- /usr/games

Az ls parancs futtatása például /bin/ls. Ha egy parancs útvonalban van, akkor nem kell megadnunk a teljes útvonalat. Elég az ls parancs kiadása.

A felhasználók is rendelkezhetnek saját futtatható programokkal vagy secriptekkel. Ezeket általában a saját bin könyvtárában tartja. A „jozsi” nevű felhasználó bin könyvtárát akkor így érjük el: /usr/jozsi/bin.

Előfordul, hogy egy parancs nincs útvonalban viszont nem szeretnénk a teljes útvonalat megadni, de a program helyi könyvtárban van. Ekkor egy ./ jelezzük, hogy a keresett program vagy script az aktuális könyvtárban van.

Folyamatok listázása

Folyamatok listázására a ps parancsot használjuk. Alapesetben azonban nem fogjuk látni az össze folyamatot.

```
$ ps
PID TTY          TIME CMD
6714 pts/2    00:00:00 bash
6895 pts/2    00:00:00 ps
```

Az összes folyamat megtekintéséhez használjuk a ps ax kapcsolóit:

```
$ ps ax
```

Vegyük észre, hogy a kapcsolóknak nem szükséges kötőjel.

Ezek után ehhez hasonló kimenetet láthatunk (részlet másolata):

```
PID TTY      STAT   TIME COMMAND
...
6379 ?        Sl     4:26 /usr/lib/iceweasel/firefox-bin -a iceweasel
6704 ?        Rl     0:01 gnome-terminal
6713 ?        S      0:00 gnome-pty-helper
6714 pts/2    Ss     0:00 bash
6894 pts/2    R+     0:00 ps x
```

Az első oszlopban a látjuk a pidszámokat (PID). A pid a „process identity” szavak rövidítése, ami folyamat azonosítónak fordítható. Folyamat alatt pedig a futó programot értjük. Az utolsó oszlopban a futtatott parancsot látjuk kapcsolókkal együtt.

Folyamatok leállítása

Folyamat leállítása erőszakosan, de biztosan

```
$ kill -9 pidszam
```

Például a ps ax parancs után látjuk hogy a leállítandó parancs pid száma 8340, akkor a kill parancsot így alkalmazzuk:

```
$ kill -9 8340
```

Folyamatoknak küldhető jelzések

man 7 signal kézikönyv alapján:

Jel	Azonosító	Típus	Leírás
SIGHUP	1	Term	Akadály detektálása kontroll terminálon vagy a halott folyamaton.
SIGINT	2	Term	Megszakítás a billentyűzetről
SIGQUIT	3	Core	Kilépés a billentyűzetről
SIGILL	4	Core	Illegális utasítás
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Lebegőpontos kivétel
SIGKILL	9	Term	Semlegesítő jel
SIGSEGV	11	Core	Érvénytelen memóriahivatkozás
SIGPIPE	13	Term	Eltört cső (pájp): írás vagy olvasás nem létező csőbe
SIGALRM	14	Term	Időzítőjel az alarm(2)-től
SIGTERM	15	Term	Megszakításjel
SIGUSR1	30,10,16	Term	Felhasználó által definiált jelzés 1
SIGUSR2	31,12,17	Term	Felhasználó által definiált jelzés 2
SIGCHLD	20,17,18	Ign	A gyermek leállt vagy megszakadt
SIGCONT	19,18,25	Cont	Folytatás, ha meg lett állítva
SIGSTOP	17,19,23	Stop	A folyamat megállítása
SIGTSTP	18,20,24	Stop	Megálljt gépeltek egy tty eszközön
SIGTTIN	21,21,26	Stop	tty bevitel egy háttérfolyamat számára
SIGTTOU	22,22,27	Stop	tty kivitel egy háttérfolyamat számára

Több folyamat kezelése egyetlen terminálban

Folyamat háttérbe helyezése

A folyamat elindítása után nyomjunk Ctrl+Z billentyűpárost.

A példa kedvéért indítsuk el a nano szövegszerkesztőt:

```
nano
```

Ha elindult dolgozhatunk vele, majd tegyük a Ctrl+Z paranccsal háttérbe. Ezek után indítsunk egy másik programot, például a mutt levelező programot. Itt is dolgozhatunk majd a Ctrl+Z paranccsal ezt is háttérbe tehetjük. Hogy milyen programok vannak a háttérben a jobs paranccsal tudhatjuk meg:

```
jobs
[1]-  Stopped                 nano
[2]+  Stopped                 mutt
```

Láthatjuk, hogy a nano és a mutt programok vannak a háttérben. Vegyük észre, hogy mind a két program sorszámot kapott. A sorszámok segítségével vissza, azaz előtérbe hozhatjuk a nano vagy a mutt programot. Hozzuk előtérbe a most a nano programot a fg paranccsal:

```
fg 1
```

Az fg programnak egyetlen paraméter szükséges, annak a programnak a sorszáma amit előtérbe szeretnénk hozni.