

## 6 - Fájkezelés

### Fájlok létrehozása

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

```
touch fajlneve.txt
```

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

```
echo "alma" > gyumolcsok.txt
```

### Fájlok törlése

Egy fájl törlése:

```
rm fajlnev
```

Törlés rekurzívan:

```
rm -r fajlnev
```

Az rm parancs alapveten fájlok törlésére lett kitalálva, ezért könyvtárat csak úgy tudunk törölni vele, ha beállítjuk a rekurzívítást.

A könyvtár és tartalmának törlése:

```
rm -r könyvtárnév
```

Az egész könyvtárfa törlése a gyökértl kiindulva:

```
rm -r /
```

### Fájlok átnevezése

```
mv fajl001 fajl002
```

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

Adott állomány mozgatása adott könyvtárba:

```
mv fajl konyvtar
```

vagy:

```
mv fajl konyvtar/
```

A fájl nev állományt a konyvtar nev mappába mozgatjuk.

Több fájl is mozgatható egy könyvtárba, a lényeg, hogy a könyvtár neve legyen az utolsó:

```
mv fajl1 fajl2 fajl3 konyvtar
```

### Fájlok másolása

Az aktuális könyvtárban szeretnénk a file1 állományról egy file2 másolatot:

```
cp fajl01 fajl02
```

Adott egy „munka” nev könyvtár az aktuális könyvtáron belül. Ebbe szeretnénk másolni a file1 nev állományt, ugyanezen a néven:

```
cp fajl01 munka/
```

### Fájlok tartalmának megtekintése

Gyakran szükség van arra, hogy gyorsan belenézzünk egy fájlba. Erre használható például a cat parancs:

```
cat fajlnev
```

### Fájlok darabolása

```
split -b 1440000 fajlnev
```

Régebben ha floppy lemezekre akartunk másolni, pont ekkora méretre kellett darabolnunk a fájlt. A fenti parancs segítségével a fajlnev nev állományt 1440000 byte nagyságú részekre lett feldarabolva. A fájlokat automatikusan elnevezi ehhez hasonlóan:

- xaa
- xab
- xac
- stb

### Darabolt fájlok összefűzése

A darabolt fájlokat elbb-utóbb szeretnénk összeszerkeszteni. Ezt a cat paranccsal tehetjük meg az alábbi formában:

```
cat x* >> fajlnev
```

A csillag, azt jelenti, hogy minden könyvtár ami x-el kezdik.

### Fájlok szerkesztése

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

```
vi virag.txt
```

vagy

```
nano virag.txt
```

vagy

```
mcedit virag.txt
```

Az Midnight Commander beépített szövegszerkesztje. Valójában egy szimbolikus link az mc parancsra. Ebből következik, hogy a használatához az „mc” csomagnak telepítve kell lennie a gépünkön.

## Nano

### Billentyk

A nano szövegszerkeszt alapvet billentykombinációi Ctrl gombbal működnek. Ezt maga a szövegszerkeszt is tudatja velünk, de a Ctrl helyett egy „^” karaktert látunk mindig. A mentés például a Ctrl+O billentykombinációval történhet, amit így látunk:

```
^O
```

Vagy a kilépés Ctrl + X billentykombináció, amelyet így látunk:

```
^X
```

### Szerkesztés

Meg kell mondani, hogy most kijelölés jön. A kijelölés végét nem kell jelezni (ellentétben az mcedit-el). Ha kijelöltük azt amit vágólapra szeretnénk tenni, akkor el kell dönteni ezt hogyan szeretnénk. A kijelölt szöveg kivágásával, vagy másolásával. Ennek megfelel billentykombinációt nyomunk.

Alt + 6	Kijelölés kezdete
Ctrl + K	Kivágás
Ctrl + U	Beillesztés

A másolás tulajdonképpen úgy néz ki, hogy Ctrl + U billentykombinációt kétszer használjuk. Először helyben, ahonnan kivágja, majd a kívánt másolási helyre viszem a kurzort és ott is Ctrl + U.

A Ctrl + K billentykombinációt használhatjuk kijelölés nélkül is. Ekkor azt a sort vágja ki, amelyikben éppen állt a kurzor a Ctrl + K lenyomásakor.

### Egyéb

Alt + Q	Kijelölés kezdete
---------	-------------------

### Linkek a fájlrendszerben

#### Merevlink

Amikor létrehozunk egy állományt a merevlemezen lefoglalunk számára egy szektort. Ugyanakkor létrejön egy bejegyzés, amelyet inode néven említjük. Az inode nev egységben van leírva a fájl idbélyege, mérete, tulajdonosa, csoportja, jogok, stb. Az inodeban van feljegyezve, hogy hol helyezkedik el a merevlemezen a lefoglalt szektor, ezt a bejegyzést hívhatjuk tulajdonképpen linknek. A fájl létrehozásakor tehát egy linket hozunk létre, mégpedig merevlinket. Ilyen merevlinket többet létrehozhatunk egy fájlra. Ennek módja a következő:

```
ln fajlnev linkfajlra
```

Ha már két merevlink mutat egy fájlra, a fájl csak akkor törlődik, ha mindkét linket töröljük.

Vegyük a következő példát. Létrehozok egy fájlt:

```
touch erdo.txt
```

Most létrehozok egy rámutató linket:

```
ln erdo.txt fak.txt
```

Most törlöm a erdo.txt-t:

```
rm erdo.txt
```

A fájl maga nem kerül törlésre, mert a fak.txt nev link még az állományra mutat. Az állomány törlése csak ekkor történhet meg.

Az ls -l parancs kimenete megmutatja hány darab szimbolikus link mutat egy fájlra.

## Szimbolikus link

A szimbolikus link nem valódi link, ezért is nevezzük szimbolikusnak. Ha van egy állomány amelyre csak egyetlen merevlink mutat és egy szimbolikus link, amikor törlöm a merevlinket a fájl törlődni fog. A szimbolikus link ez után megmarad, de nem mutat majd sehova.

Szimbolikus link létrehozása

```
ln -s fajlnev linkfajlra
```

## Fájlok statisztikája

A stat parancs még több információt szolgáltat egy állományról vagy egy könyvtárról.

```
stat fajlnev
```

Például:

```
stat log.txt
```

```
File: "log.txt"
  Size: 0          Blocks: 0          IO Block: 4096   szabályos üres fájl
Device: 802h/2050d Inode: 1461005     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   andras)   Gid: ( 1000/   andras)
Access: 2012-01-15 23:00:17.365731988 +0100
Modify: 2011-08-09 22:35:50.584228887 +0200
Change: 2011-08-09 22:35:50.584228887 +0200
```

File	A fájl neve
Size	A fájl mérete
Blocks	A fájl által használt blokkok száma (512 bájtos blokkokkal számol!)
IO	IO blokkméret a fájl számára
fajltípus	szabályos fájl, szabályos üres fájl könyvtár szimbolikus link foglatat speciális karakterfájl speciális blokkfájl
Device	Az eszköz neve hexadecimális és decimális formában.
Inode	A fájl egyedi azonosítója a fájlrendszerben
Links	Linkek száma a fájlra

Access	Hozzáférés szám és karakterformában.
Uid	Tulajdonos
Gid	Csoport
Access	Utolsó fájlolvasási időpont
Modify	A fájl tartalmának utolsó változása
Change	A fájl inode leírójának utolsó változása

```
stat könyvtarnev
```

Magáról a fájlrendszerrel kaphatunk információt a -f kapcsoló megadásával:

```
stat -f fajlnev
```

Például:

```
stat -f log.txt
```

```
File: "log.txt"
  ID: 3f4e5580b0204b72 Namelen: 255      Type: ext2/ext3
Block size: 4096      Fundamental block size: 4096
Blocks: Total: 9612076   Free: 4608081   Available: 4119806
Inodes: Total: 2444624   Free: 2115881
```

Ha írunk a fájlba, akkor módosul a mtime (Modify time), de ezzel együtt a ctime (Change time) is. A ctime viszont módosul még néhány esetben. Ha például módosítom a fájl jogait, vagy tulajdonsát, akkor a ctime érték frissül.

A -c vagy a --format kapcsolóval kiírathatjuk (többek között) mit ért (hány bájt-ot) a stat blokkméret alatt:

```
stat --format=%B fajlnev
```

## Fájlok mérete

A fájlok lemezen foglalt helyét és méretét tudhatjuk meg a „du” parancs segítségével. A du alapértelmezésként a lemezen elfoglalt helyet mutatja meg blokkméretben. 1 blokkot 1 kibibájt-nak számít, azaz 1024 bájt-nak.

Ha bájt-ban szeretnénk megkapni pontosan mekkora az állomány mérete, akkor a -b vagy --bytes kapcsolót kell használnunk.

Hozzunk létre saját könyvtárunkban egy fájlt:

```
cd ~
echo "Lorem ipsum doloest amet" > lorem.txt
```

Nézzük meg a helyfoglalását:

```
du lorem.txt
```

Az eredmény:

```
4 lorem.txt
```

Nem négy bájt. Négy kilobájt (4 \* 1024 bájt). Ennyi helyet foglal a lemezen. A du parancs tehát alapértelmezésben 1024-es blokkokkal számol.

Ha be van állítva a POSIXLY\_CORRECT környezeti változó, akkor 512 bájtós blokkokkal kell számolnunk. Állítsuk be a változót és nézzük meg újra a parancs kimenetét:

```
set -o posix
du lorem.txt
```

A kimenet:

```
8 lorem.txt
```

A kimenetben most 8 blokk, vagyis 8 darab 512 bájtós helyet foglal az állományunk, ami 4096 bájt összesen.

A POSIXLY\_CORRECT környezeti változót azért szokták beállítani, hogy a programok úgy működjenek, hogy megfeleljenek a POSIX szabványoknak. A POSIXLY\_CORRECT kikapcsolásához:

```
set + o posix
```

A set Bash belső parancsa, több információért nézd meg a leírását:

```
help set
```

Most nézzük meg a valódi méretét:

```
du -b lorem.txt
```

vagy:

```
du --bytes lorem.txt
```

26 bájtot kapunk:

```
26 lorem.txt
```

Nézzük meg a stat paranccsal is a lorem.txt lefoglalt blokkméretét. Vegyük észre a különbséget. A stat azt mondja 8 blokk, mert az eleve 512 bájt mérettel számol.

Ebbl láthatjuk, hogy az adott fájlrendszer blokkmérete 4096 bájt.

A teszteléshez létrehozhatunk egy 4096 bájt nagyságú fájlt:

```
dd if=/dev/zero of=teszt.txt bs=1024 count=4
```

Esetleg adjunk hozzá még két bájtot:

```
echo a >> teszt.txt
```

## Fájlnevek

A fájlnevek tartalmazhatnak ékezetes fájlneveket, számokat, alulvonást, kötjelet, pontot, stb.

```
Maximális hosszuk: 256 bájt
```

Vegyük észre, hogy a 256 a karakterek száma helyett a szükséges bájtokat tartalmazza. A magyar ékezetes betűk például 2 bájtosak.

## Fájlok keresése

Keresés tartalom alapján:

```
find . -name "*.txt" -print | xargs grep "Nagy József"
```

Olyan .txt kiterjesztés fájlokat keresünk, amelyben szerepel a „Nagy József” név.

Egyszerbben:

```
grep -lir "Nagy József" /utvonal/könyvtár
```

Olyan fájlokat keresünk, adott könyvtárban, amelyben szerepel a „Nagy József” név. Az eredmény a fájl neve.

Harmadik verzió:

```
egrep -R "Nagy József" /útvonál/könyvtár
```

A find és grep parancsról később részletesen is szó esik.







A locate parancs:

```
locate minta
```

Létezik egy /var/lib/mlocate/mlocate.db nev adatbázis, amely segít megtalálni állományokat. Az adatbázis a következő paranccsal frissíthet:

```
updatedb
```

## Gyakorlat

- Milyen fájl szerkesztő parancsok vannak?
- Milyen paranccsal hozhatunk létre egy új üres állományt?
- Milyen paranccsal törölhetek egy állományt?
- Milyen paranccsal nevezhetek át egy állományt? Írjon példát.
- Milyen paranccsal mozgathatok egy saját könyvtárból egy alkönyvtárba egy fájlt. Írjon példát.
- Hozza létre újra a Proba könyvtárat, és másolja bele az /etc könyvtár passwd fájlját! 
- Nézze meg a Proba könyvtár passwd fájljának linkszámát! Hozzon létre egy hardlinket erre a fájlra jelszofajl néven! Jelenítse meg a jelszofajl tartalmát! Mennyi a linkszáma ennek és az eredeti fájlnek? 
- Törölje a passwd nev fájlt! Mennyi most a linkszám? Megjeleníthet a jelszofajl tartalma? 
- Nevezze át a jelszofajl-t passwd-re! Hozzon létre egy softlinket erre a fájlra, a neve legyen s\_p.txt! 
- Milyen hosszú ez a fájl? Miért ennyi? 
- Jelenítse meg az s\_p.txt tartalmát! 
- Jelenítse meg a tartalomjegyzékét! Hol látható, hogy ez a fájl valójában softlink