Linux bevezetés, fájl és könyvtárkezelő- és egyéb parancsok, felhasználók és csoportok kezelése

Összeállította: Anda Zoltán

Rövid Unix/Linux történet

- Ken Thomson és Dennis Ritchie az AT&T Bell Laboratóriumában 1969-ben készítette el a Unix első változatát egy PDP-7 számítógépre.
- Az AT&T a Unix forráskódját amerikai egyetemek rendelkezésére bocsátotta, rövidesen sok Unix rendszer jött létre.
- A rendszermagot 1973-ban átírják C nyelvre.
- A gyors terjedés hátránya volt a nem ellenőrzött sokféle forráskód. Két verzió vált a legjelentősebbé, a Berkeley egyetemen kifejlesztett BSD Unix, illetve az AT&T System V (SYSV) változata.
- Elindulnak a szabványosítási törekvések, az IEEE kidolgozta a „POSIX" (Portable Operating System Interface (x)) ajánlást, amely igyekszik egyesíteni a két jelentősebbé vált szabványt.
- A PC-k megjelenésével létrejönnek az Intel-PC-alapú Unixok, először oktatási célokra, majd létrejönnek a komoly munkákra is képes PC-s Unix verziók: Unixware Unix (Novell), Solaris Unix (Sun Microsystems), AIX Unix (IBM), …
- Az egyetemeken Unixra sokféle program készült C nyelven, ezek forrását publikussá is tették, amelyeket mások továbbfejlesztettek.
- Létrejön a **Free Software Foundation** (FSF, Richard Stallman irányításával) alapítvány, melynek célja egy szabadon és ingyenesen hozzáférhető szoftverkörnyezet biztosítása. Ennek a munkának a részeként jön létre a **GNU** (GNU is Not Unix) projekt, amelynek célja egy teljesebb Unix környezet létrehozása.
- A GNU ötletének jogi formába öntése a **GPL** (GNU General Public Licence).
- A GPL védelme alá eső szoftvert bárki készíthet, szerzői jogi védelmet az FSF biztosít. GPL szoftvert bárki használhat, megkapja a forrását, azt módosíthatja, de pénzt nem kaphat érte, a módosítás után is a szoftver GPL termék marad. Továbbadása ingyenes.
- E munkák során rendelkezésre állt a GNU környezet (fordítók, segédprogramok, grafikus felület). Mindehhez szükség lett volna egy olyan operációs-rendszer magra, ami nem tartalmazza a BSD vagy SYSV védettség alá eső kódjait.
- Ezt a munkát kezdi el **Linus Torvalds** (helsinki egyetemista) aki megírja a **Linux** kernelt, akinek a munkáját később sokan segítik. Így jön létre a Unix egy klónja, a Linux.
- A Linux kernelre sok fejlesztő cég építi a saját Linux operációs rendszerét (segédprogramokkal, alkalmazásokkal), így jönnek létre a Linux disztribúciók (pl.: Debian, Mandriva, Slackware, OpenSuse, Redhat, Ubuntu, Kubuntu, Fedora, Uhu, Zenwalk, Frugalware, ….).

A Linux főbb jellemzői

- **Multitask és multiuser** rendszer.
- Eredetileg az I-386 processzor biztosította a modern tár és feladatkezelési módokat, így a valódi időosztásos környezetet ki lehetett alakítani. A Linux kernel a 386-os processzor „védett” üzemmódját használja.
- Ennél a védett üzemmódnál a kernel hozzáfér a gép összes fizikai erőforrásához (és csak a kernel). A folyamatok (processzek) ún, „user” módban futnak. A processzor több ilyen „user” módú processzt tud kezelni. Ezek a folyamatok nem érik el sem egymás-, sem a kernel memóriaterületét (védelem). A folyamatok a gép fizikai erőforrásaihoz sem férnek hozzá közvetlenül, csak a kernelen keresztül (biztonságos fájlrendszer).
- **Lapozásos virtuális memóriakezelés** (swap használata).
- **Lapozásnál (paging)** a rendszer arra ügyel, hogy a folyamatokhoz szükséges lapok mindig a fizikai memóriában legyenek. Ha azok netán lemezen vannak, akkor gondoskodik memóriába olvasásukról, illetve ha a memória megtelt, akkor a ritkábban használt lapokat a lemezre írja.
- **Tárcserénél (swapping)** pedig a rendszer figyeli az egyes folyamatok aktivitását, és ha szabad memóriára van szükség, egy inaktív folyamat egészét háttértárra írja. Ezután felszabadítja a folyamat által használt összes fizikai memóriát.
- A Linux használja az ún. **buffer cache módszer**t, amellyel a lemezműveleteket gyorsítja. Ekkor memória mérete dinamikusan (terheléstől függően) változik, mindig az éppen szabad fizikai memória egészét erre a célra használja. A lemezre írások is a buffer cachen keresztül történnek. Minden írás először a cache memóriába kerül, és egy megadott idő után íródik ki lemezre (vagy akkor, ha megfelelő mennyiségű adat összegyűlt).
- **Demand paging (igény szerinti lapozás)**: egy fájl futtatásakor nem az egész fájl töltődik a memóriába, hanem csak azok a lapok, amelyekre éppen szükség van.
- **Osztott kódkönyvtárak használata**: a programok C nyelven íródnak, így sokban szerepel olyan függvény, amely más programokban is előfordul. Emiatt ezek a függvények csak egyszer töltődnek be a memória egy bizonyos helyére. A **dinamikus linker** segít a programoknak megtalálni a memóriában a függvényeket, illetve a memóriába töltésükről gondoskodik, amennyiben még nem lennének betöltve.
- **Copy-on-write mechanizmus**: multitaszkos operációs rendszerben gyakran lehet arra szükség, hogy bizonyos memórialapokat több folyamat között megosszunk. Mivel a memóriamásolás meglehetősen időigényes dolog, nem kell azt a lapot lemásolni, csak elhelyezni az ugyanarra a lapra mutató hivatkozásokat a megfelelő helyeken. Most már csak arra kell vigyáznunk, hogy ha az ugyanarra a lapra hivatkozó több folyamat közül valamelyik módosítani akarja a lapot, (mivel mindegyik folyamat azt hiszi, a memórialap csak az övé) akkor készítsünk csak annak a folyamatnak a számára másolatot, amelyet aztán módosíthat kedve szerint.
- **Preemptív időosztásos ütemezés** : az operációs rendszernek egy processzoron kell több feladatot végrehajtania, ezért valamilyen formában meg kell osztania a rendelkezésreálló CPU időt az egyes folyamatok között. A Unix rendszerek (így a Linux is) a preemptív időosztásos ütemezés módszerét alkalmazzák, ami azt jelenti, hogy a rendelkezésre álló időt felosztja egyenlő részekre, és ezekből az egyenlő időszeletekből juttat –a folyamat prioritásának megfelelően – többet vagy kevesebbet az adott folyamatnak. Az egyes folyamatok prioritása természetesen állítható. Amikor az adott folyamat számára kijelölt időszelet letelt, a kernel

megszakítja a folyamat futását és más folyamatnak adja át a vezérlést. Linuxban az ütemezés alapegysége az 1/100 másodperc.

Linux fájlrendszere

- A Linux „mindent” fájlként kezel. Fájlként kezeli a billentyűzetet (csak olvasható), a szöveges képernyőt (csak írható), a nyomtatót, a CD írót, de még a fizikai memória tartalmát is. A gépben lévő winchester szektorait is egy speciális file olvasásával illetve írásával érhetjük el, Ugyanígy a hálózati eszközöket is a filekezelés szabályainak megfelelően használhatjuk.
- Legalább háromféle fájltípus van a Linuxban:
    - **Egyszerű fájl:** bármilyen állomány, ami valamilyen adatot, szöveget tartalmaz. Ez lehet egy program forráskódja, egy futtatható állomány, egy szöveges dokumentum vagy egy adatbázisunk.
    - **Jegyzék fájlok:** lehetővé teszik, hogy a fájljainkat (mindhárom típust) valamilyen logikai rendszerbe szervezzük.
    - **Különleges fájlok:** ezek rendszerint olyan fájlok, amelyek valamilyen eszközt képviselnek, például diszket, a terminálunkat vagy a hangkártyánkat.
- A Linux operációs rendszer egyetlen fájlrendszerrel dolgozik. Ez alapvetően egy fa szerkezetű struktúra, melynek van egy gyökere, a **/** nevű jegyzék. Ebben, mint minden jegyzékben a korábban tárgyalt háromféle típusú bejegyzések találhatók. Minden egyes jegyzék további bejegyzéseket és így további jegyzékeket tartalmazhat, továbbá biztosan tartalmazza a saját magára mutató . és a szülőjére mutató .. bejegyzést. A gyökérben a .. is saját magára mutat. A jegyzékben minden bejegyzésnek külön névvel kell rendelkeznie. Bejegyzés neve az ASCII 0-ás és a / jel kivételével tetszőleges karaktert tartalmazhat.
- Az **inode-**ok az egyes fájlokkal kapcsolatos információt tartalmazzák: a méretet, a láncok számát, a mutatókat azon lemezblokkokra, amelyek a fájltartalmat ténylegesen tárolják, valamint a létrehozás, módosítás és hozzáférés dátumát és idejét.
- **Egy standard jegyzékszerkezet:**

/bin	Itt találhatók a létfontosságú bináris programok.
/boot	A rendszer bootolásakor használatos fájljait tartalmazó könyvtár. Itt található általában a rendszermag (kernel, vmlinuz), illetve GRUB rendszerbetöltő esetén annak konfigurációs állománya is.
/dev	A speciális eszközök lelőhelye.
/etc	Mindenféle vegyes dolgot tartalmaz, például egyes programok konfigurációs fájljait, a jelszóállomány(oka)t, stb.
/etc/rc.d	A rendszer futási szintjeihez tartozó irányító szkriptek vannak itt.
/etc/skel	Az itt lévő fájlokat kapja meg minden új felhasználó a home jegyzékébe.
/home	Rendszerint itt vannak a felhasználók saját jegyzékei.
/lib	A legtöbb program futásához nélkülözhetetlen dinamikusan linkelhető könyvtárak vannak itt.
/proc	Linuxokra jellemző, a processzekkel kapcsolatos információkat hordozó virtuális fájlrendszer.
/root	A rendszergazda home jegyzéke.
/sbin	A rendszergazda számára alapvető fontosságú bináris programok jegyzéke.
/tmp	Ideiglenes fájlok tárolására szolgáló jegyzék. Mindenki írhat bele.

/usr	Ez a könyvtár teszi ki a használt lemezterület nagyságrendileg 80-90%-át. Olyan fájlok és könyvtárak vannak benne, amelyek parancsokat (bin), rendszerparancsokat (sbin), függvénykönyvtárakat (lib), dokumentációkat (doc), kézikönyveket (man), forrásokat (src), ideiglenes fájlokat (spool) tartalmaznak.
/var	Sűrűn változó dolgok könyvtára. Olyan fájlokat tartalmazó könyvtár, amelyek állandóan változnak (pl. log/napló fájlok). Itt találhatók még egyes programok átmeneti, de hosszabb ideig tárolt fájljai is (/var/cache/apt/archives), és alapértelmezett esetben a felhasználói levél fiókok (/var/mail/user).

- **A Linux fájlrendszere (ext4)**
  - Az **ext4** a legtöbbet használt linuxos fájlrendszernek, az ext3-nak a továbbfejlesztése. Az ext4 mélyebb átalakulást jelent, mint az ext2 ext3-ra fejlesztése, mivel az ext2-ből jóformán csak a naplózás hozzáadásával lett ext3. Az ext4-ben viszont fontos, adatstruktúrát érintő változtatások vannak, amik az adatok tárolását befolyásolják. Ezeknek a változtatásoknak és az optimalizált tervezésnek hála a fájlrendszernek jobb a teljesítménye, megbízhatóbb lett és újabb funkciókkal gazdagodott.
  - Az Ext4-et kifejezetten úgy fejlesztették ki, hogy az ext3-mal a lehető leginkább visszamenőleg kompatibilis legyen. Ez lehetővé teszi az ext3 fájlrendszerek frissítését az ext4 fájlokra.
  - Az ext4 48-bites blokkcímzést használ, így 1 EB (Exabyte) a fájlrendszer maximális mérete, amelyen legfeljebb 16 TB méretű lehet egy fájl. 1 EB = 1 048 576 TB (1 EB = 1024 PB, 1 PB = 1024 TB, 1 TB = 1024 GB).

Az Ubuntu disztribúció

- Az Ubuntu egy Debian alapú Linux disztribúció 6 hónapos kiadási ciklusokkal, a legfrissebb szoftverekkel, hibajavításokkal.
- Az Ubuntu szó afrikai eredetű, jelentése „emberiesség másokkal szemben".
- Az Ubuntu jelenleg a **GNOME** felületet használja alapértelmezetten.
- Egyéb felhasználói felületekkel is elérhető, melyek fejlesztése kisebb-nagyobb mértékben különálló projektként fut, mint például az Ubuntu KDE-s testvére, a Kubuntu vagy az LXQt-re alapozott Lubuntu disztribúció.
- Az Ubuntu projekt mögött a Canonical Ltd áll. Ez többek között a legfontosabb fejlesztők támogatását, az infrastruktúra fenntartását, illetve üzleti tevékenységet (céges támogatás) takar.
- Az Ubuntuban a szoftvereket a licencük, és támogatottságuk szerint négy osztályba (komponensek) sorolják:
  - **Main:** hivatalosan támogatott szabad szoftver.
  - **Restricted:** hivatalosan támogatott nem szabad szoftver.
  - **Universe:** hivatalosan nem támogatott szabad szoftver.
  - **Multiverse:** hivatalosan nem támogatott nem szabad szoftver.
- **LTS** (Long Term Support - hosszú ideig támogatott) kiadás, kétévente adják ki, 5 évig támogatott.

Parancssoros fájlkezelés

sudo: (Superuser do) Lehetővé teszi, hogy rendszergazdaként vagy más felhasználó nevében hajtsunk végre parancsokat.
gksu: A sudo grafikus megfelelője; grafikus felületű programokat ezzel kell indítani, amennyiben rendszergazdai jogok szükségesek

Segítség

man: Egy parancs manuálját adja, kilépés: **q**

```
$ man man Man parancs manuálja.
$ man -k IPv4 Megadja az IPv4et tartalmazó parancsokat.
```

parancs – help: A parancsról ad segítséget.

```
$ pwd --help Segítséget ad a pwd parancsról.
```

info parancs: Ugyancsak a parancsról ad információt.

```
$ info pwd Információt ad a pwd parancsról.
```

whatis parancs: Parancsról ad egysoros tájékoztatót.

whereis parancs: Parancs és manuáljának a helye.

which parancs: Parancs futtatható formájának a helye.

Fájl- és könyvtárkezelő parancsok

pwd: Kiírja az aktuális munkakönyvtárat.
ls: Kilistázza az aktuális könyvtár tartalmát

```
$ ls könyvtár1 könyvtár2 Kilistázza a 2 könyvtár tartalmát.
$ ls -A Kilistázza az aktuális könyvtár tartalmát , a rejtetteket is.
$ ls -l Több információ.
$ ls -li Mint az előző, csak az inode számokat is kiírja.
$ ls -A -l Kapcsolók sorrendje tetszőleges a legtöbb parancsnál.
$ ls -lA Kapcsolók egymásba ágyazhatók.
$ ls -l /etc/*.conf Az /etc mappa conf kiterjesztésű fájljait adja.
$ ls -l /etc/u??.conf Az /etc mappa u karakterrel kezdődő és 3 karakter
névhosszúságú conf kiterjesztésű fájljait adja.
$ ls -l /etc/[au]??.conf Az /etc mappa a vagy u karakterrel kezdődő és 3
karakter névhosszúságú conf kiterjesztésű fájljait adja.
$ ls -l /etc/[!a-g]??.conf Az /etc mappa azon 3 karakter névhosszúságú conf
kiterjesztésű fájljait adja, amelyek kezdőkaraktere nem az a-g intervallumból
való.
$ ls -l /etc/{ker,lib}*.conf Az /etc mappa ker vagy lib szavakkal kezdődő conf
kiterjesztésű fájljait adja.
```

Helyettesítő (joker vagy wildcard) karakterek: Lehet használni joker karaktereket. Hatékonyan lehet sok fájllal dolgozni, ezek a karakterek használhatók a parancsok többségénél. A joker karakterek:

*	0 vagy több bármilyen karakter
?	Egy bármilyen karakter
[abcde]	Egy a felsorolt karakterekből

[a-e] Egy karakter az intervallumból

[!a-e] Bármilyen karakter, ami nincs a listában

cd : Segítségével mozoghatunk a könyvtárstruktúrában az alábbi módon:

```
$ cd /etc/apt/ Megadhatjuk a teljes elérési utat.
$ cd ./apt.conf.d/ A ./ használatával nem kell újra és újra begépelnünk ugyanazt (jelen esetben ez a /etc/apt/ elérési utat), ezt egyszerűen kiváltjuk a ./ segítségével.
$ cd ~ A home mappába ugrunk.
$ cd .. Fölfelé ugrunk a könyvtárstruktúrában (gyökér irányába).
$ cd - Visszalépés az előző könyvtárba.
```

cp : Ezzel a paranccsal tudunk másolni.

```
$ cp -r /home/Dir1 /home/Dir2 Rekurzívan másol.
```

mv : Ezzel adott fájlt vagy könyvtárat tudunk mozgatni (áthelyezni) vagy átnevezni.

```
$ mv /home/file1 /home/file2 file1 fájlt nevezi át file2 névre.
```

rm : Fájlok törlése.

```
$ mv /home/file1 Törli a file1 fájlt.
$ mv -r /home/Dir1 Törli a Dir1 mappa teljes tartalmát.
```

rmdir: Egy üres könyvtár törlése.

mkdir : Könyvtár létrehozása.

mount : Partíció, képfájl felcsatolása.

```
$ sudo mount /dev/sdb1 /media/AA sdb1 partíció csatolása az AA mappába.
$ sudo mount 192.168.1.100:/home/user1 /media/AA Másik gép egy felhasználói könyvtárának mount-olása (Majd később tudjuk használni!).
```

umount: Partíció, képfájl leválasztása.

```
$ sudo umount /media/AA Előző mount-olás megszüntetése.
```

A Linux archiváló programja a tar: A **tar** parancs fájlok, könyvtárstruktúrák archiválására, mentésére szolgál. A megadott fájlokat illetve könyvtárstuktúrát egy közönséges fájlba (az ún. *tarfile*-ba vagy archívumba) csomagolja be. A **tar** parancs után általában szükséges legalább egy *kulcs* használata, ami azt adja meg, mit is várunk éppen a parancstól. A kulcs vagy kulcsok után a kapcsolók a szokásos szerepet töltik be, s használatuk nem kötelező. A *tarfile* nevét csak akkor adhatjuk meg, ha az f kapcsoló szerepel.

tar : Archiváló alkalmazás.

```
$ tar cwf tarfile.tar * Mindent archivál az adott helyről, rákérdez minden fájl esetén. Nem kerül be az archívumba a fájl elérési útja.
$ tar cf ~/Dir1/tar1.tar ~/Dir2/* Mindent archivál a Dir mappából és az archív fájl a Dir1 mappába kerül. Bekerül az archívumba a fájl elérési útja (abszolút).
$ tar czf ~/Dir1/tar1.tar.gz ~/Dir2/* Annyiban tér el az előzőtől, hogy gzip-el tömöríti is az archív fájlt.
$ tar cf Dir1/tar2.tar Dir2 Archiválja a Dir1 mappát és az archív fájl a Dir1 mappába kerül. Bekerül az archívumba a fájl elérési útja (relatív).
```

Ha az archiválandó fájlok előtt van útvonal (abszolút vagy relatív), akkor az archívumba került fájlok is ezen utakkal archiválódnak! Nyilván az archívumból való kicsomagolásnál ezen útvonalak alá kerülnek a kibontott fájlok (ha az útvonal nem létezik, a tar létrehozza).

```
$ tar tf tarfile.tar Kiírja a tarfile.tar fájl tartalmát.
$ tar tf ~/Dir1/tar1.tar Kiírja a tar1.tar fájl tartalmát.
$ tar rf tarfila.tar kesz.html Hozzáteszi a tarfila.tar archívhoz a kesz.html-t.
$ tar f tarfile.tar --delete file1 Törli az archivból a file1-et.
$ tar uf tarfile.tar * Csak azokat a fájlokat teszi az archív fájlba, amelyek tartalma változott, azaz eltérnek az archívban találttól.
```

```
$ tar xf tarfile.tar A tar file kicsomagolása helyben. Az x kulccsal a gzip-el tömörített archív fájlt is kicsomagolja.
$ tar xf tarfile.tar –C ~/IDE A tar file kicsomagolása az IDE mappába.
```

gzip: Egy fájl tömörítésére használható, általában több fájl esetén előtte archiváljuk a fájlokat, majd az archív fájlt tömörítjük, így lesz **.tar.gz** kiterjesztésű fájlunk.

```
$ gzip tarfile.tar A tarfile.tar archívot tömöríti, tarfile.tar.gz tömörített fájl jön létre.
$ gzip –d tarfile.tar.gz A tarfile.tar.gz tömörített fájl kicsomagolása.
$ gunzip tarfile.tar.gz Ugyanaz, mint az előző!
```

zip, unzip: Fájlok tömörítése és archiválása zip formátumba(ból).

```
$ zip ~/archiv ~/Dir1/* A ~/Dir1 mappa összes fájlját tömöríti egy archiv.zip fájlba.
$ unzip ~/archiv.zip –d ~/Dir2 Az ~/archiv.zip fájl kicsomagolása a ~/Dir2 mappába (útvonallal együtt csomagol ki).
```

cat: Összefúz és kiír fájlokat.

```
$ cat file1.txt Kiírja a file1.txt fájlt.
$ cat –n file1.txt file2.txt Kiírja a két fájlt sorszámozva.
$ cat file1.txt > file2.txt file1.txt fájlt beleírja file2.txt fájlba.
$ cat e*.txt ???.txt Az e kezdetű és a 3 kar. txt fájlok kiírása.
```

Parancsok összefűzése, átirányítása:

| Ennek az operátornak a segítségével egymás után lehet fűzni az utasításokat oly módon, hogy a következő utasítás bemenete a megelőző utasítás kimenete legyen.

> A parancs kimenetét nem a szabványos kimenetre, hanem egy fájlba irányítjuk át. A fájl korábbi tartalma elvész.

>> A parancs kimenetét nem a szabványos kimenetre, hanem egy fájlba irányítjuk át. A fájl korábbi tartalma nem vész el, az új tartalmat a fájl végére írjuk.

chattr: Fájlok, könyvtárak attribútumát változtatja meg.

Néhány attribútum: **A**: Nem változik az utolsó módosítási dátum (gyorsítást jelent). **a**: Csak hozzáfűzés módban lehet megnyitni írásra a fájlt. **c**: Automatikusan tömörít a kernel a lemezen. Ilyen fájlt olvasva a kitömörített adatot kapjuk vissza, íráskor a tömörített adat kerül a lemezre. **i**: A fájlt nem lehet módosítani. Nem lehet törölni, átnevezni, hozzáfűzni, benne adatot átírni és semmilyen kötést (link) rá létrehozni. Csak superuser (root) tud adni vagy elvenni ilyen attribútumot. **s.** A fájl törlésekor blokkjai kinullázódnak a lemezen. **S** A fájl módosításakor, a változások szinkronban lesznek a lemezen lévő adattal. **u**: A fájl törlésekor annak tartalma megőrződik. Ez lehetővé teszi, hogy később visszahozhassuk.

```
$ sudo chattr -R +i Dir1 A Dir1 mappa rekurzívan kapja az i attribútumot. A „+” hozzáad attribútumot, „-„ jellel elveszünk.
```

lsattr: Fájlok, könyvtárak attribútumát nézhetjük meg. A –R kapcsoló itt is működik, ami a rekurzióra utal.

cmp: Összehasonlítja két fájl tartalmát bájtonként.

```
$ cmp –lb file1 file2 file1 –et és file2-t hasonlítja össze. Nincs különbség nem ír  semmit, „-l” kapcsolónál kiírja az összes különbséget (nélküle csak az első eltérést), „-b” kapcsolónál nemcsak az eltérő bájtok pozícióját, hanem értékét is megjeleníti.
```

diff: Ez is összehasonlít szöveges fájlokat soronként, könyvtárakat is tud, ekkor „-r” kapcsolóval rekurzívan hasonlítja a fájlokat sorban.

```
$ $ diff -r Dir1 Dir2 > file1 file1-be írja a különbséget.
```

cut: Fájlok sorainak egy részét írja ki.

```
$ cut -f1 -d' 'file1 file1 fájl sorainak első szavát írja ki.
$ cut -c1-10 -d file1 file1 fájl sorainak első 10 karakterét adja.
$ cut -c-10 file1 file1 fájl sorainak tartalmát adja a 10. karaktertől.
$ cut -c1,3 file1 file1 fájl sorainak 1. és 3. karaktereit írja ki.
```

find: Fájlok és könyvtárak keresése a könyvtár-hierarchiában.

```
$ find /usr -name x-rgb.xml A /usr mappában és almappáiban keresi a fájlt.
$ find /etc -type f -name "ubuntu*" A /etc könyvtárban és alkönyvtáraiban keresi azokat a fájlokat, melyek neve „ubuntu” szóval kezdődik, nincs különbség kis- és nagybetűk között.
$ find /tmp -group user1 A /tmp-ben keresi a user1 csoporthoz tartozó bejegyzéseket.
$ find /tmp -type f -user user1 A /tmp-ben keresi a user1 felhasználó fájljait.
$ find /var -type f -size +50M A /var mappa (és almappái) 50 Mb-nál nagyobb fájljait adja.
$ find ~ -type f -mmin -5 A ~ mappa azon fájljai, amelyek tartalma változott az elmúlt 5 percben.
$ find ~ -type f -cmin -5 A ~ mappa azon fájljai, amelyek változtak az elmúlt 5 percben (pl. tartalma, neve, stb.).
```

head: Szűrő, mely egy szöveges fájl első sorait, karaktereit adja.

```
$ head -n5 file1 file1 első 5 sorát adja.
$ head -n-2 file1 file1 sorait adja az utolsó kettő kivételével.
$ head -c10 file1 file1 első 10 karakterét adja.
```

tail: Szűrő, mely egy szöveges fájl utolsó sorait, karaktereit adja.

```
$ tail -n3 file1 file1 utolsó 3 sorát adja.
$ tail -n+3 file1 file1 sorait adja a 3. sortól.
$ tail -c10 file1 file1 utolsó 10 karakterét adja.
$ head -n3 file1 | tail -n+3 file1 3. sorát adja.
```

less: Fájl tartalmát írja ki lapozva.

```
$ less -N file1 file1 sorait írja ki sorszámozva.
$ ls -l /etc|less Lapozva jeleníti meg az /etc  mappa tartalmát.
```

sort: Sorba rendezi a fájl sorait. „-r” kapcsolóval fordított sorrend.

touch: Fájl időbélyegének megváltoztatása. Néhány kapcsoló: **a**: Fájl utolsó elérésének a dátumát változtatjuk.. **m**: Fájl utolsó módosításának a dátumát változtatjuk. **t**: Idő megadása [[CC]YY]MMDDhhmm.[ss] formátumban.

```
$ touch -m -t 202003061546.10 file1 file1 utolsó módosítási dátumát állítjuk be.
$ touch ~/file1.txt Egy üres file1.txt fájlt hoz létre az aktuális idővel.
```

file: Megvizsgálja egy fájl típusát.

```
$ file file1 file1 típusát írja ki.
$ file * Adott könyvtár tartalmát írja típusok szerint.
```

which: A program futtatható állományának elérési útvonalát adja meg.

```
$ which nano /usr/bin/nano-t adja.
```

Softlink és hardlink létrehozása: A **szimbolikus** vagy a **softlink** az eredeti fájlhoz való tényleges link, míg a hardlink az eredeti fájl tükrös másolata. Ha töröljük az eredeti fájlt, a softlinknek ezután

nincs értéke, mert egy nem létező fájlra mutat. De a hardlink esetén ez más, ha töröljük is az eredeti fájlt, a hardlink fájl továbbra is tartalmazza az eredeti fájl adatait, mivel a hardlink az eredeti fájl tükrözött másolataként működik. A fájlhoz egy hardlink létrehozása más, mint a másolás. Ha másolunk egy fájlt, akkor az csak a tartalmat fogja lemásolni. Tehát, ha módosítjuk az egyik fájl tartalmát, akkor az a másolatra nincs hatással. Ha azonban létrehozunk egy hardlinket egy fájlhoz, és megváltoztatjuk bármelyik fájl tartalmát, akkor a változás mindkét oldalon látható lesz. Softlink esetén az inode értékek és a jogosultságok különböznek, míg hardlink esetén megegyeznek.

ln: Hardlink és softlink vagy szimbolikus link létrehozása.

```
$ ln –s /usr/bin/nano editor Softlink: editor is inítja a nanot-t.
$ ln ~/file1.txt ~/file2.txt Hardlink: file1.txt hardlinkje lesz a file2.txt.
```

Reguláris kifejezések (regular expression - RegEx): Több Linux eszköz használ mintaillesztést, s a megadott mintára illeszkedő adatokon folyhat további munka. Vannak olyan parancsok, ahol a felhasználó adja meg a keresendő mintát, ilyen a **grep** parancs, más programokban rejtve a minta megadása. Például mintaillesztési szabályokat használnak a különböző editorok (**ed, vi**). Reguláris kifejezések használatakor egy összetett mintát adunk meg (ez a reguláris kifejezés), s azt vizsgáljuk, hogy a feldolgozandó adatok (az esetek túlnyomó részében szöveges fájlok, tehát karakterláncok) melyik része illeszkedik a megadott mintára. A reguláris kifejezések karakterekből állnak, ezek közül néhány speciális jelentést hordoz, ezeket metakaraktereknek hívjuk (hasonlóan a joker karakterekhez).

A reguláris kifejezések alapvető szabályai:

- Egy egyedülálló karakter (kivéve ezeket: **. * [\ ] ^ $)** önmagára illeszkedik.
- A *c* páros, ahol c egy látható karakter, a c karakterre illeszkedik a karakter. Tehát a * illeszkedik a *-ra, a \\ pedig a \-re.
- A **.** (pont) karakter egy olyan reguláris kifejezés, amely bármelyik (nem újsor) karakterre illeszkedik. Így például az v**.**d minta illeszkedik az **vad**, **vád**, **véd**, stb. kifejezéskere.
- Ha *r* egy reguláris kifejezés, akkor *r****** egy olyan reguláris kifejezés, amely *r* reguláris kifejezés 0 vagy többszöri előfordulását jelenti. Így az **k*** reguláris kifejezés illeszkedni fog az üres stringre is (hiszen abban nullaszor szerepel az **k** karakter), valamint az **k**, **kk**, **kkk**, **kkkk**, ... stb. kifejezésekre.
- **[..]** a szögletes zárójelbe tett karaktersorozat illeszkedik az abban a pozícióban lévő bármely, a zárójelben felsorolt karakterre. Kódjukat tekintve egymás után következő karaktereket rövidíteni lehet a kötőjel használatával. Például **[0-9]** ill. **[a-z]** jelenti az összes számjegyet és az angol ábécé összes kisbetűjét.
- A nyitó zárójelet követő ^ jel, a felsorolt karakterek tagadása. Azaz **[^0-9]** jelenti bármely, nem szám karaktert.
- Két egymás után írt reguláris kifejezés szintén reguláris kifejezés. Például a **[^0-9][0-9]** kifejezés egy nem szám karaktert követő számkarakterre illeszkedik.
- Két egymástól | jellel vagy újsorral elválasztott reguláris kifejezés illeszkedik akár az egyik akár a másik kifejezésre. Így a **[a-z]|[0-9]** kifejezés az adott pozícióban csak kisbetűt vagy egy számjegyet fogad el.
- A **\(..\)** tett reguláris kifejezés illeszkedik a zárójelbe tett kifejezésre, és egyben megjelöli azt.
- A ^ jel a sor elejére a **$** jel a sor végére illeszti a mintát. Például a **^[^0-9]*$** kifejezés a számot nem tartalmazó sorra illeszkedik.
- A *n* kifejezés (ahol *n* egy szám) a zárójelezéssel kijelölt mintára hivatkozik, mégpedig a kijelölés sorrendjében. Így a ^\(.\)\(.\)\(.\).*\3\2\1$ minta olyan sorokra illeszkedik, ahol a sor első három karaktere tükörszimmetrikus az utolsó két karakterre (pl. '**abcxy123cba**').

grep: Szövegrészt keres fájlokban, kimenetben.

```
$ grep –r ”ubuntu” Dir1 Dir1 könyvtárban rekurzívan keresi azokat a fájlokat,
melyek tartalmazzák az „ubuntu” szót.
$ grep –r –l ”ubuntu” Dir1 Mint az előző, csak a fájlneveket adja.
```

```
$ grep "^a" file1 file1 „a" karakterrel kezdődő sorait adja.
$ grep "^[^0-9]*$" file1 file1 számokat nem tartalmazó sorait adja.
$ grep "^\(.\)\(.\)\(.\).*\1\2\3$" file1 file1 azon sorait adja, ahol a sor első
3 karaktere megegyezik az utolsó 3-mal.
$ man cp|grep "symbolic" A cp man-jában keresi a symbolic szót tartalmazó sorokat.
```

Jogok, felhasználók módosítására szolgáló parancsok

adduser: Új felhasználó és jelszó megadása.

```
$ sudo adduser user1 user1 felhasználó létrehozása.
$ sudo adduser user1 group1 user1-et beteszi a group1 csoportba.
$ cat /etc/passwd|less Felhasználók listázása.
```

passwd: Jelszó megváltoztatása.

```
$ sudo passwd user1 user1 jelszavának a megváltoztatása.
```

usermod: A felhasználó accountjának módosítása, csoporttagság megadása.

```
$ sudo usermod -L user1 user1 belépését letiltjuk.
$ sudo usermod -U user1 user1 belépését engedélyezzük.
```

addgroup: Csoport létrehozása

```
$ sudo addgroup suli suli csoport létrehozása.
$ cat /etc/group|less csoportok listázása.
```

deluser, delgroup: Felhasználó és csoport törlése.

login: Bejelentkezés.

logout: Kijelentkezés.

su: Switch user, felhasználó váltás. Lehetővé teszi a felhasználó számára, hogy másik felhasználóvá váljon anélkül, hogy ki/belépne.

```
$ su user2 user2 felhasználó lép be.
$ exit Előzőleg belépett user2 felhasználó kilép.
```

who: Bejelentkezett felhasználók.

w: Ki van bejelentkezve, mit csinál.

whoami: Milyen néven vagyunk bejelentkezve.

users: Rendszeren lévő felhasználók kiírása.

groups: Egy user csoportjait írja ki.

```
$ groups user1 user1 csoportjait írja ki.
```

history: Parancsok kiíratása.

```
$ history 10 Utolsó 10 parancsot adja.
$ history -c Törli a parancselőzményeket (~/bash_history tartalmát).
$ history | grep "bashrc" Kiírja azon parancsokat, amelyben szerepel a bashrc.
```

du: Adott könyvtár és fájljainak a méretét írja ki hierarchikusan.

```
$ du --si --max-depth=3 Aktuális könyvtárban ír ki a 3. mélységig, méret után
mértékegység is megjelenik.
$ sudo du -h /home A /home helyfoglalásait adja (megfelelő mértékegységgel).
```

stat: Fájlról, könyvtárról részletes ismertetés.

Hozzáférési jogosultságok (permissions): Egy többfelhasználós rendszerben lényeges, hogy az egyes felhasználók saját adatait meg tudjuk védeni a szándékos vagy véletlen károkozástól. Ezért a Linux egy tulajdonosi rendszert használ, s ennek alapján dönt az állományokhoz való hozzáférés engedélyezéséről, illetve megtagadásáról. A Linux minden felhasználót a felhasználói azonosítója (uid) azonosít a rendszerben, ezen felül minden felhasználó valamilyen csoportba is tartozik, így van egy csoportazonosítója (gid) is. Amikor szeretnénk hozzáférni egy fájlhoz vagy mappához, a rendszer a tulajdonosi viszony alapján dönt a hozzáférésről. Lehetünk **tulajdonosi- (u)**, **csoporttagsági- (g)**, vagy **egyéb (o)** viszonyban a bejegyzéssel. A bejegyzésekkel kapcsolatos tevékenységek

szempontjából három fő csoport van: az állomány **olvasása (r)**, **írása (w)**, illetve **végrehajtása (x),** ez utóbbi katalógus esetében keresést jelent. Minden fájlművelet előtt a Linux megvizsgálja, hogy melyik tulajdonosi kategóriába tartozunk, és megvizsgálja, hogy ebben a kategóriában engedélyezett-e vagy sem a végrehajtani kívánt művelet. A hozzáférési jogosultságok az **ls -l** és a **stat** parancsok alapján ellenőrizhetők a legkönnyebben. A következő két sor az ls –l egy kimenetét mutatja:

```
-rw-r--r-- 1 owner1 group1 14649 Sep 6 09:54 nevek.txt
drwxr-xr-x 2 owner2 group2 32 Nov 22 24:32 alkonyvtar
```

A baloldali 10 karakteres oszlop (sötétszürke) első karaktere a fájltípust mutatja, ezek a következők lehetnek:

- - reguláris fájl (teljesen egyszerű bináris vagy szöveges állomány)
- d könyvtár
- c karakteres típusú eszközfájl (konzol is ilyen például: /dev/tty)
- b blokk típusú eszközfájl (winchesterek: /dev/sda)
- l linktípusú fájl
- p speciális cső (pipe)
- s socket-ek - a hálózati kapcsolatokat oldják meg

A következő kilenc karakter tartalmazza, háromszor hármas bontásban, a hozzáférési jogosultságokat. Ha egy művelet engedélyezett, a neki megfelelő betű látszik a listán, ha nem, a '**-**' karakter jelzi a tiltást. Az első hármas csoport a tulajdonos (u), a második a csoport (g), végül a harmadik a többiek (o) jogosultságait mutatja.

chmod: Könyvtár és fájljogok beállítására szolgáló parancs.

```
$ chmod 755 file1 file1-nél u(ser) mindent, g(roup) és o(ther) r-t, x- et kap.
$ chmod a-w file1 file1-nél a(ll)-tól (mindenkitől) elvesszük a w-t.
$ chmod –R o-rwx Dir1 Dir1-nél az o-tól mindent elveszünk rekurzívan.
```

chown: Fájlok, könyvtárak tulajdonosának megváltoztatása.

```
$ sudo chown root file1 file1 új tulajdonosa root lesz.
$ sudo chown –R root:root Rir1 Dir1 új tulajdonosa és csoportja root lesz
rekurzívan.
```

chgrp: Fájlok, könyvtárak csoportjának megváltoztatása.

```
$ sudo chgrp root file1 file1 csoportja a root lesz.
```

A jogok kiegészítése: Három ilyen különleges bit van, ami az eddig említett jogosultságokat kiegészíti: *setUID* (Set User ID), *setGID* (Set Group ID) és *sticky* bit. Először arról ejtsünk szót, hogy milyen szerepet töltenek be ezek a bitek.

setUID bit: A programok futtatásánál előfordulnak olyan helyzetek, amikor az eddigi jogosultsági beállításokkal nem tud működni az operációs rendszer. Ehhez nézzünk egy nyilvánvaló példát. A jelszavak tárolása az */etc/shadow* fájlban történik. E fájl jogosultságai:

```
user2@user1-VirtualBox:/home/user1$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1395 febr   5 13:23 /etc/shadow
```

Ezt a fájlt csak a *root* tudja írni. Ennek ellenére bármely felhasználó a *passwd* paranccsal módosítani tudja a saját jelszavát, azaz írás történik a *shadow* fájlba. Ez hogyan lehetséges, amikor a *passwd* parancsot nem is a *root* indítja? Ebben az esetben a *passwd* parancs kap egy különleges jogosultságot, ez lesz a *setUID* jog, ami azt jelenti, hogy a *passwd* parancs a *root* jogkörével fog futni. Természetesen e jog megadása sok biztonsági problémát is fölvet. E jog így jelenik meg (*x* helyett *s* karakter a tulajdonosnál):

```
user2@user1-VirtualBox:/home/user1$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59640 márc  22  2019 /usr/bin/passwd
```

A *setUID* shell scripteknél nem működik!

setGID bit: Hasonlít a *setUID* joghoz. Azt jelenti, hogy ha *setGID* jogosultságú fájlt futtatunk, akkor a fájlt a rendszer a fájl csoportjának jogaival futtatja. E jog így jelenik meg (x helyett *s* karakter a csoportnál):

```
user1@user1-VirtualBox:~$ ls -l progi
-rwxrwsr-- 1 user1 user1 0 febr   6 14:56 progi
```

sticky bit: Ezt a bitet könyvtárak esetén használjuk. Ha egy könyvtárra engedélyezzük a *sticky* bitet, ez azt jelenti, hogy a könyvtárban mindenki csak a saját tulajdonában lévő fájlt tudja törölni.

```
user1@user1-VirtualBox:~$ ls -ld /tmp
drwxrwxrwt 15 root root 4096 febr   6 14:20 /tmp
```

A */tmp* mappára mindenkinek van írási joga, de a *sticky* bit beállítása miatt mindenki csak a saját fájlját törölheti. E jog úgy jelenik meg, hogy *x* helyett *t* karakter van az *other*-nél.

Az eddigi jogosultságok megadását egy oktális számhármassal meg tudtuk tenni, egy oktális jeggyel adtuk meg sorban a tulajdonos, csoport és mindenki más jogait. A *setuid, setgid* és *sticky* bitek beállítását egyrészt megtehetjük betűkkel (*s* és *t* karakterekkel). Például:

```
user1@user1-VirtualBox:~$ ls -l progi
-rwxr-xr-x 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod u+s progi
user1@user1-VirtualBox:~$ ls -l progi
-rwsr-xr-x 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod u-s progi
user1@user1-VirtualBox:~$ ls -l progi
-rwxr-xr-x 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod g+s progi
user1@user1-VirtualBox:~$ ls -l progi
-rwxr-sr-x 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod g-s progi
```

Másrészt egy újabb oktális számmal, amit az előző háromjegyű oktális szám elé teszünk. Így egy 4 jegyű oktális számmal megadhatjuk ezeket a speciális jogokat is. A 4 jelenti a setuid-, 2 jelenti a *setgid*-, és az 1 jelenti a *sticky* bit beállítását. Például:

```
user1@user1-VirtualBox:~$ ls -l progi
-rwxr-xr-x 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod 4755 progi
user1@user1-VirtualBox:~$ ls -l progi
-rwsr-xr-x 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod 2755 progi
user1@user1-VirtualBox:~$ ls -l progi
-rwxr-sr-x 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod 1755 progi
user1@user1-VirtualBox:~$ ls -l progi
-rwxr-xr-t 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod 7755 progi
user1@user1-VirtualBox:~$ ls -l progi
-rwsr-sr-t 1 user1 user1 0 febr   6 14:56 progi
user1@user1-VirtualBox:~$ chmod 0755 progi
user1@user1-VirtualBox:~$ ls -l progi
-rwxr-xr-x 1 user1 user1 0 febr   6 14:56 progi
```

umask: Ezzel a paranccsal lehet lekérdezni az aktuális umask értéket, illetve be lehet vele állítani az új umask-ot.

```
$ umask
```

```
$ umask 0002
```

Az umask szerepe: Könyvtárak esetében a *0777*, míg fájloknál a *0666* oktális számból kivonjuk az *umask* értékét, akkor megkapjuk azt a jogot, amit az újonnan létrehozott könyvtárak és fájlok kapnak. Egy általános felhasználónál az *umask* alapértelmezett értéke *0002*, míg az első felhasználónál *0022*.

Fájlhozzáférések szabályozása ACL-ekkel: Az Access Control Lists (ACL) egy kibővített engedélyezési mechanizmust biztosít a fájlrendszerek eléréséhez. Úgy tervezték, hogy segítse a Linux fájljogosultságait, mert azok nem biztosítanak arra lehetőséget, hogy a fájljogosultságokat megfelelően ki tudjuk terjeszteni. Az ACL lehetővé teszi, hogy engedélyeket adjon bármely felhasználónak vagy csoportnak bármilyen lemezes erőforráshoz. Az ACL akkor használható, ha telepítve van az acl csomag. Az ACL engedélyezéséhez a fájlrendszert az acl opcióval kell felszerelni (ext4-nél ez alapértelmezett). Ennek ellenőrzését a **tune2fs** paranccsal lehet megtenni (pl.: *sudo tune2fs –l /dev/sda1*). Tehát az ACL használatával újabb felhasználók és csoportok számára tudunk jogosultságokat beállítani. Amikor könyvtárakhoz vagy fájlokhoz jogosultságokat rendelünk egy ún. *ACE* (Access Control Entry) jön létre, ami a kiosztott jogosultságokat írja le. A jogosultságok:

Megnevezés	Jelölés	Jelentés
Tulajdonos	user::rwx	A fájl/könyvtár tulajdonosa.
Egyéb felhasználó*	user:<név>:rwx	ACL-ben megadott felhasználó.
Tulajdonoscsoport*	group::rwx	A fájl/könyvtár csoportja.
Egyéb csoport*	group:<név>:rwx	ACL-ben megadott újabb csoport.
Maszk	mask::rwx	Ezzel a *-gal jelöltekből maszkolhatunk ki biteket.
Mindenki más	other::rwx	A fentieken kívüli felhasználók.

Az ACL-ek lekérdezésére szolgál a *getfacl* parancs. Ha nem állítunk be ACL-el jogosultságot, akkor a következő jelenik meg:

```
user1@user1-VirtualBox:~$ ls -l hard.txt
-rw-r--r-- 2 user1 user1 10 febr   2 17:50 hard.txt
user1@user1-VirtualBox:~$ getfacl hard.txt
# file: hard.txt
# owner: user1
# group: user1
user::rw-
group::r--
other::r--
```

Az ábrán látható, hogy ugyanazt az eredményt adja az *ls* parancs, mint a *getfacl*. Látható, hogy a csoportnak csak olvasási joga van. Adjunk egy *user2* nevű felhasználónak olvasási és írási jogot is! Ezt a *setfacl* paranccsal tehetjük meg, majd ellenőrizzük az eredményt a *getfacl* és *ls* parancsokkal:

```
user1@user1-VirtualBox:~$ setfacl -m user:user2:rw hard.txt
user1@user1-VirtualBox:~$ getfacl hard.txt
# file: hard.txt
# owner: user1
# group: user1
user::rw-
user:user2:rw-
group::r--
mask::rw-
other::r--

user1@user1-VirtualBox:~$ ls -l hard.txt
-rw-rw-r--+ 2 user1 user1 10 febr   2 17:50 hard.txt
```

A *setfacl* parancs *-m* kapcsolója után adjuk meg a módosítást az előzőekben bemutatott jelölések felhasználásával. Látható a *getfacl* kimenetén, hogy a *user2* felhasználó megkapta az *rw* jogokat. Az *ls* parancs kimenetén a jogosultságok után egy „+” jelent meg, ami azt jelenti, hogy a fájl az alapjogosultságokon kívül ACL-lel beállított jogokkal is rendelkezik.

Adjuk meg az *rw* jogot a *hard.txt* esetén egy *student* csoportnak is:

```
user1@user1-VirtualBox:~$ setfacl -m group:student:rw hard.txt
user1@user1-VirtualBox:~$ getfacl hard.txt
# file: hard.txt
# owner: user1
# group: user1
user::rw-
user:user2:rw-
group::r--
group:student:rw-
mask::rw-
other::r--
```

Az ACL-lel megadott jogokat az *–x* kapcsolóval tudjuk törölni. Távolítsuk el az előzőleg a *student* csoportnak megadott jogokat:

```
user1@user1-VirtualBox:~$ setfacl -x group:student hard.txt
user1@user1-VirtualBox:~$ getfacl hard.txt
# file: hard.txt
# owner: user1
# group: user1
user::rw-
user:user2:rw-
group::r--
mask::rw-
other::r--
```

Az ACL-lel megadott összes jogot a –b kapcsoló használatával tudjuk törölni:

```
user1@user1-VirtualBox:~$ setfacl -b hard.txt
```

Könyvtárakra hasonlóan adjuk ACL-lel a jogokat, csak *user* helyett *group* kulcsszó lesz a *setfacl* parancsban.

Default ACL-ek: Könyvtárak esetén megadhatunk ún. alapértelmezett jogokat. Ekkor egy újabb könyvtár létrehozásakor a gyermek könyvtár örökli a szülő könyvtár ACL bejegyzéseit és az öröklődés beállításait is. Egy új fájl létrehozásakor a könyvtár ACL-jei érvényesek lesznek a létrehozott fájlra is. Az alapértelmezett ACL beállításához a *-m* mellett a *-d* kapcsolót használjuk.

A következő példában egy *ARC* mappa esetén adunk a *student* csoportnak alapértelmezett *rwx* jogosultságokat:

```
user1@user1-VirtualBox:~$ getfacl ARC
# file: ARC
# owner: user1
# group: user1
user::rwx
group::rwx
other::r-x

user1@user1-VirtualBox:~$ setfacl -d -m group:student:rwx ARC
user1@user1-VirtualBox:~$ getfacl ARC
# file: ARC
# owner: user1
# group: user1
user::rwx
group::rwx
group:student:rwx
mask::rwx
other::r-x
default:user::rwx
default:group::rwx
default:group:student:rwx
default:mask::rwx
default:other::r-x
```

Ezután ki kell próbálni az *ARC* mappában mappa és fájl létrehozását, majd megnézni az ACL jogosultságaikat!

A maszk szerepe: Az itt szereplő jogok az *Egyéb felhasználó* (nem a tulajdonos, hanem az a másik felhasználó, akinek jogot adunk az ACL-ben), *Tulajdonoscsoport* (az a csoport, amelyhez a bejegyzés tartozik) és az *Egyéb csoport* (nyilván a tulajdonoscsoporton kívüli másik csoport, amelynek jogot adunk az ACL-ben) esetén jelenthet változást a jogosultságokban. Ebben a három esetben megadott jogosultságok csak akkor érvényesek, ha azok a jogok a maszkban is szerepelnek.

```
user1@user1-VirtualBox:~$ getfacl Dir1
# file: Dir1
# owner: user1
# group: user1
user::rwx
group::r-x
other::r-x

user1@user1-VirtualBox:~$ setfacl -m other:--- Dir1
user1@user1-VirtualBox:~$ getfacl Dir1
# file: Dir1
# owner: user1
# group: user1
user::rwx
group::r-x
other::---
```

Létrehozunk egy *Dir1* mappát a *user1* felhasználó home könyvtárában. Az első jogosultsági lekérdezés azt mutatja, hogy alapesetben a tulajdonoson kívüli (*other*) felhasználók *r-x* jogosultságokat kapnak. Ezt megszüntetjük a *setfacl* paranccsal, majd újra lekérdezzük a jogosultságokat.

```
user1@user1-VirtualBox:~$ su user2
Jelszó:
user2@user1-VirtualBox:/home/user1$ ls -l Dir1
ls: nem lehet a következő könyvtárat megnyitni: 'Dir1': Engedély megtagadva
```

Belépünk egy másik felhasználóval (*user2*) és megpróbáljuk megjeleníteni a *Dir1* tartalmát. Nyilván nem sikerül, mert nincs rá jogosultságunk.

```
user1@user1-VirtualBox:~$ setfacl -m user:user2:r-x Dir1
user1@user1-VirtualBox:~$ getfacl Dir1
# file: Dir1
# owner: user1
# group: user1
user::rwx
user:user2:r-x
group::r-x
mask::r-x
other::---
```

Most beállítjuk a *setfacl* paranccsal a *user2* felhasználónak a *Dir1* mappára az *r-x* jogosultságokat, majd megjelenítjük a jogokat a *getfacl* paranccsal.

```
user2@user1-VirtualBox:/home/user1$ ls -l Dir1
összesen 0
```

A jogok megadása után a *user2* meg tudja jeleníteni a *Dir1* tartalmát (az most egy üres mappa), tehát a megkapott jogok érvényesülnek.

```
user1@user1-VirtualBox:~$ setfacl -m mask:--- Dir1
user1@user1-VirtualBox:~$ getfacl Dir1
# file: Dir1
# owner: user1
# group: user1
user::rwx
user:user2:r-x                  #effective:---
group::r-x                      #effective:---
mask::---
other::---

user1@user1-VirtualBox:~$ su user2
Jelszó:
user2@user1-VirtualBox:/home/user1$
user2@user1-VirtualBox:/home/user1$ ls -l Dir1
ls: nem lehet a következő könyvtárat megnyitni: 'Dir1': Engedély megtagadva
```

Ha elvesszük a maszkról a jogosultságokat és belépünk a *user2* felhasználóval, azt tapasztaljuk, hogy a felhasználó nem fér hozzá a *Dir1* mappához. Annak ellenére nem fér hozzá, hogy ő megkapta az *r-x* jogosultságokat. A maszkban beállított jogosultságok, azaz minden jog tiltása érvényesült az egyéb felhasználónál (*user2*) a maszk miatt. A maszk által érvényesített tényleges jogokat mutatja az effective kulcsszó után a *user2*-nél és a *group*-nál.

Egyéb rendszeradminisztrációs parancsok

shutdown: Leállítás.

```
$ sudo shutdown –h now Azonnali leállítás.
$ sudo shutdown –h 22:00 Leállítás adott időben.$ sudo shutdown –r now Azonnali
újraindítás.
$ sudo shutdown –h 22:00 Újraindul adott időben.
$ sudo shutdown –c Időzítés kikapcsolása.
```

reboot: Ugyancsak újra indítja az op. rendszert

echo: Kiírás parancsa.

```
$ echo „Hollo” Kiírja, hogy Hello..
$ echo $[12/4] Kiírja az eredményt.
$ echo $PATH Kiírja a PATH környezeti változó értékét.
```

lsbkl: A számítógépen lévő blokk eszközökről szolgáltat információkat.

dumpe2fs: Az ext2/ext3/ext4 fájlrendszerről ír ki információkat.

```
$ sudo dumpe2fs /dev/sda1 Az sda1 blokkezközről írja ki az információkat.
```

parted: Partíció módosítása, beállítások megjelenítése.

```
$ sudo parted Elindít egy menüs/párbeszédes lehetőséget.
$ sudo parted –l /dev/sda Az sda lemez partícióit írja ki.
```

fdisk: Lemezkezelő, partícionáló program.

```
$ sudo fdisk -l A partíciós táblát írja ki.
```

df: Szabad tárterületet adja partíciónként.

```
$ df --si
$ df –hT | sort –hrk 3 A 3 oszlop (méret) alapján rendezve adja, fájlrendszert
(T) is kiírja.
```

free: Memóriahasználat kilistázása. A „–b” „–k” „–m” „–g” kapcsolókkal byte-ban, kibibyte-ban (alapértelmezett), mebibyte-ban és gibibyte-ban adja az eredményt.

Információkat lehet még nyerni a **/proc** mappában található fájlokból is. Pl:

```
$ cat /proc/version Kernelről.
$ cat /proc/cpuinfo Processzorról.
$ cat /proc/meminfo Memóriáról.
$ cat /proc/partitions Partíciókról.
```

Aliasok: Amikor hosszabb parancsokat írunk be rendszeresen, nagy segítséget nyújthatnak az alias-ok, amelyekkel saját parancsokat készíthetünk. Aliast a következőképpen készíthetünk:

`Szintaxisa: alias [name=['command']]`

Egy példa: `alias sysup='sudo apt-get update'`

Ezzel a paranccsal készítettünk egy `sysup` nevű parancsot, amely a `sudo apt-get update` paranccsal egyenértékű. Ezzel a módszerrel további parancsok készíthetők. A termeniálból való kijelentkezéssel a létrehozott aliasok is elvesznek. Az aliasok mentését a **.bashrc** szerkesztésével tudjuk megtenni.

```
$ alias Kiírja az érvényes aliasokat.
$ alias -p Mint az előző.
$ unalias sysup Törli a aliast.
$ alias d1='du –h --max-depth=2 /etc | sort –hr | less'
```