Linux alapozó – mérési segédlet

Összeállította: a BME Távközlési és Médiainformatikai Tanszéke 2016-ban.
A forrásdokumentáció Sonkoly Balázs munkája.
Vill. BSc-re adaptálta: Kulik Ivett.

Bevezetés: GNU/Linux rendszerek

A GNU egy kizárólag szabad szoftverekből álló operációs rendszer, melyet a GNU projekt keretein belül fejlesztenek. A rövidítés jelentése rekurzív módon: "GNU's Not Unix", ami arra utal, hogy a GNU egy Unix-szerű (Unix-like) operációs rendszer, viszont nem tartalmaz Unix-ból származó kódot és azzal ellentétben szabad szoftver. A GNU projekt 1983-ban indult Richard M. Stallman vezetésével azzal a céllal, hogy egy Unix-szerű, de szabad operációs rendszert fejlesszenek ki. A Free Software Foundation (FSF) 1985-ös alapítása szintén Stallman nevéhez fűződik. A GNU projekt által fejlesztett komponensekre GNU csomagok vagy GNU programok néven hivatkoznak, melyek közül a legfontosabbak a következők: GNU Compiler Collection (GCC), GNU Binary Utilities (binutils), bash shell, GNU C library (glibc), GNU Core Utilities (coreutils), Emacs szövegszerkesztő. A GNU operációs rendszer hivatalos kernelje (rendszermagja) a GNU Hurd komponens, ami az aktív fejlesztések ellenére sem teljes és nem működik együtt minden részegységgel. Ezért helyette széles körben használják a Linux kernelt mint külső magot. Azonban a Linux kernel – ellentétben olyan külső szoftverekkel, mint például az X.Org X Window rendszere vagy a ! TeX betűszedő rendszer – nem lett hivatalosan része a GNU operációs rendszernek.

A Linux elnevezés szigorú értelemben a Linux kernelre vonatkozik, melyet Linus Torvalds kezdett el fejleszteni 1991-ben. A tradicionális Unix kernelekhez hasonlóan a Linux kernel is egy monolitikus kernel, ami azt jelenti, hogy egyetlen nagy programból áll a rendszermag, nem pedig különálló, egymással különböző interfészeken keresztül kommunikáló programok összessége, mint napjaink mikrokernelei esetében (pl. Hurd). A Linux kernelen alapuló rendszerek szinte minden esetben a GNU projekt által fejlesztett rendszerkönyvtárakat és alapprogramokat használják. Ezért az FSF magára az operációs rendszerre "GNU/Linux" operációs rendszerként hivatkozik. (Ezzel szemben sokan, mint például Torvalds, nincsenek ezen a véleményen, és magát az operációs rendszert is Linux-nak nevezik.)

A Linux egy széles körben portolt kernel, ami azt jelenti, hogy mára már a legkülönfélébb számítógépes architektúrán működik. A beágyazott rendszerektől, mobiltelefonoktól kezdve a szuperszámítógépekig számos helyen alkalmazzák. Legelterjedtebben szervereken használják, de egyre nagyobb teret hódít az asztali PC-k és laptopok esetében is.

Számos Linux disztribúció létezik, ami egy adott készítőtől vagy gyártótól származó összeállítás, amely egy GNU/Linux alaprendszert, és ahhoz tartozó, bizonyos szempontok szerint válogatott és testreszabott programokat (és gyakran egyéni kernelt) tartalmaz. Néhány népszerű disztribúció például: Debian GNU/Linux, Ubuntu, openSUSE, Red Hat Linux, Knoppix, stb. A mérés során a Debian disztribúciót fogjuk használni.

A rendszer felépítése

A Linux egy többfelhasználós (multiuser), többfeladatos (multitasking) operációs rendszer a Unix-kompatibilis eszközök teljes készletével. Kódja a legtöbb Unix-implementációhoz hasonlóan három fő részből áll: \begin{itemize}

- **Kernel:** az operációs rendszer központi eleme, ami a belső funkciókért felelős. Szabályozza a rendszer működését, vezérli a perifériákat, a rendszererőforrásokhoz való hozzáférést és olyan alapfeladatokat lát el, mint például a processzek ütemezése vagy a virtuális memória kezelése. A Linux kernel mindig a processzor privilegizált végrehajtási módjában (*kernel mode*) hajtódik végre, amikor minden fizikai erőforráshoz hozzáfér. A kernel nem tartalmaz felhasználói módú (*user mode*) kódot. A kernel módba váltás a rendszerkönyvtárak rutinjaiban levő rendszerhívásokkal történik.
- **Rendszerkönyvtárak:** egyrészt azokat a standard funkciókat valósítják meg, melyeken keresztül az alkalmazások együtt tudnak működni a kernellel, másrészt olyan függvényeket biztosítanak, melyekre több alkalmazásban is szükség van, így azokat nem kell minden programmal együtt külön lefordítani és tárolni, hanem egy közös helyen tárolva bármely program elérheti őket.
- **Segédprogramok (felhasználói programok):** a felhasználói igények széles körét lefedő alkalmazások különféle egyedi, illetve speciális feladatokra, mint például: rendszer inicializálás, konfigurálás, állandóan futó programok a hálózati kapcsolatok kezelésére, belépési kezdeményezések kezelésére, naplózásra, stb.

Más Unix rendszerekhez hasonlóan van egy nagyon fontos felhasználói program, ami a felhasználó és az operációs rendszer között teremti meg a kapcsolatot. Ez a shell (héj), ami tulajdonképpen az operációs rendszer felhasználói felülete. Az a feladata, hogy beolvassa és értelmezze az operációs rendszer számára kiadott parancsokat, és például lehetővé tegye billentyűzet vagy egér segítségével felhasználói és egyéb programok indítását. A rendszerbe való sikeres belépés után normál esetben elindul a megfelelő (felhasználó által beállított) shell program.

A shell nem integráns része az operációs rendszernek és nem élvez speciális megkülönböztetéseket, így azt bárki lecserélheti az általa preferált programra. Különböző shell programok terjedtek el és vannak ma is használatban, ezek közül a legfontosabbak a következők:

- **sh:** Bourne shell, szinte minden Unix változatban megtalálható. Ez volt az első és de-facto szabvány lett.
- **bash:** Bourne Again Shell, a Bourne shell kibővített változata, ez a Linux default shellje.
- **csh:** C shell, szintaxisa a C programozási nyelvre épül.
- **ksh:** Korn shell, a Bourne shell és a C shell egyfajta ötvözete.
- **tcsh:** Tenex C shell, a C shell egy kibővített változata.
- **ash:** Almquist shell, erőforráshiányos környezetekben használják
- **zsh:** Zhong Shao-ról elnevezett shell.

Az egyes shell változatokban a különböző kényelmi szolgáltatásokban és a programozói felületükben van eltérés. Ki kell hangsúlyozni, hogy a Unix shellek nem csupán parancsértelmezők, hanem "programozási nyelvek" is egyben. Egyszerűen kialakíthatók ún.

shell scriptek vagy parancsállományok a parancskészlet bővítésére, melyek nagyon hatékony eszközt adnak a felhasználó kezébe. A mérés során a **bash** shellel fogunk részletesebben megismerkedni.

Állományrendszer

A GNU/Linux rendszerek állományrendszere a tradicionális hierarchikus Unix állományrendszert követi, amely egy fa struktúrába képezi a fájlokat. Egyetlen gyökér (root, "/") van a rendszerben, ez a fájlrendszer kezdőpontja. Az alap állományrendszerhez tetszőleges kötetek csatlakoztathatók (**mount**) a fa struktúra tetszőleges pontjainál. Az állományok három legfontosabb csoportja a következő: közönséges (plain), periféria (device), katalógus (directory). (Ezeken kívül vannak még egyéb típusok, mint például: socket, named pipe, symbolic link.) A katalógus további állományokat tartalmazhat (szülő csomópont). A fa struktúrában keresztkapcsolatok (link) is létrehozhatók.

A Linux többféle fájlrendszer kezelésére képes. Leggyakrabban az **ext2** vagy **ext3** fájlrendszert használják (mostmár van **ext4** is). Az újabb, naplózott fájlrendszerek (pl. **ext3**) nagyobb megbízhatóságot nyújtanak, mivel a lemezen végrehajtandó változások először egy bizonyos területre, a naplóba íródnak, mielőtt végrehajtódnának. Ha a leállás a változások kiírása előtt történik, akkor a naplóbejegyzés mutatja, hogy minek kellett volna megváltoznia, így újraindítás után újra végrehajtódik az adott naplóbejegyzés, és a módosítás befejeződik. Ha a leállás a változások kiírása után történt, akkor valószínűleg nincs hiba a fájlrendszerben.

Minden fájlhoz tartozik egy ún. inode, ami egy jól definiált formátumú információcsomag és a fájl összes jellemzőjét tartalmazza (fizikai elhelyezkedés, méret, tulajdonos, védelmi kód, módosítási/létrehozási idők, hivatkozás számláló). Egy inode több fizikai blokkra mutat, melyek az állomány megfelelő részeit tartalmazzák. Az inode-ok egy inode táblában vannak és rajtuk a műveleteket a rendszer végzi. A katalógusok neveket és inode-okat kapcsolnak össze (a könyvtárbejegyzések inode-okra mutatnak). Ha egy inode-ra több katalógusbejegyzés is mutat, akkor "hardlink"-ről beszélünk (ekkor az inode hivatkozás számlálója nagyobb 1-nél). A szimbolikus link (symlink) egy speciális fájl típus, ami egy másik fájlra mutat rá. Egy adott katalóguson belül az aktuális könyvtárra a **.** (pont), a szülő könyvtárra a **..** (pont-pont) segítségével hivatkozhatunk.

A UNIX-alapú rendszerek alapkoncepciója, hogy minden egy fájl. A rendszernek egyetlen (virtuális) fájlrendszere van, és minden valós fájlrendszer, hardvereszköz, processz abban helyezkedik el. Ez a programozó számára roppant kényelmes, mert ugyanazon parancsokkal (függvényekkel) férhet hozzá mindenhez a rendszerben. A fájlrendszer felépítése nem ad-hoc, az FSH (File System Hierarchy) szabvány szabályozza. Ez, bár idegen a Windows rendszerekben megszokottól, attól jóval logikusabb rendszerezést tesz lehetővé. Itt csak a fontosabb könyvtárakat említjük meg; azon pontokon, ahol az FSH implementációs szabadságot biztosít, a (laborban található) Debian rendszereket követjük:

- **/bin:** legfontosabb futtatható programok (boot folyamathoz, alapvető rendszeradminisztrációs feladatokhoz); pl. **ls, sh, rm, mv, mount, ...**
- **/boot:** tömörített kernel image-ek (vmlinuz*) és a kernellel kapcsolatos segédfájlok
- **/dev:** hardverekhez tartozó fájlbejegyzések
- **/etc:** konfigurációs fájlok helye. A konfig fájl általában a csomag nevével egyezik meg (néhol .conf végződéssel), vagy azzal megegyező nevű könyvtárban van. Pl. a rendszerindulási hálózati beállításokat az **/etc/network/interfaces** fájl tartalmazza, melynek leírása **man interfaces**paranccsal hozható elő.

    - **/etc/init.d/:** A rendszer szolgáltatásait kezelő scriptek. Minden itteni script azonos paraméterekkel hívható meg, melyek közül a legfontosabbak: **start** - indítás, **stop** - leállítás, **restart** - újraindítás. A scriptek rendszergazdaként futtathatók. Pl. a hálózati szolgáltatások az **/etc/init.d/networking restart** paranccsal indíthatók újra.
    - **/etc/fstab:** fájlrendszer beállításai.
    - **/etc/group:** felhasználói csoportok.
    - **/etc/hostname:** gép neve.
    - **/etc/inittab:** rendszerbetöltéshez kapcsolódó beállítások.
    - **/etc/modules/modules.conf:** betöltendő kernel modulok megadása.
    - **/etc/mtab:** felcsatolt fájlrendszerek.
    - **/etc/passwd:** jelszófájl.
    - **/etc/resolv.conf:** DNS szerverek jegyzéke.
    - **/etc/services:** TCP/UDP port hozzárendelések.
- **/home:** e könyvtár alatt van minden felhasználónak egy saját, ún. home könyvtára (pl. **/home/jozsi**). Erre a saját home könyvtárra a ~ karakterrel is lehet hivatkozni bash shell esetén.
- **/lib:** A **/bin** alatti programok által használt megosztott könyvtárak (shared library-k, kb. DLL-ek) gyűjtőhelye.
- **/mnt:** az itt levő alkönyvtárakba szokás becsatolni ("mountolni") a különböző fájlrendszereket.
- **/proc:** virtuális fájlrendszer; egyes fájlokból a rendszer aktuális állapota olvasható ki (pl. **/proc/cpuinfo**: cpu információ lekérdezése), másokba írva a kernel beállításait változtathatjuk meg (pl. **/proc/sys/net/ipv4/ip_forward**: IP forwarding engedélyezése vagy tiltása). Minden futó processznek van a **/proc** alatt egy könyvtára, amelynek neve a processz PIDje.
- **/root:** A rendszergazda home könyvtára; technikai okokból volt célszerű külön helyre tenni, lehetővé téve a külön partíción tárolást.
- **/sbin:** rendszergazda által használt binárisok; pl. **mke2fs, cfdisk, init, ...**
- **/tmp:** ideiglenes (temp) fájlok helye; mindenki számára írható, tartalma időnként (beállítástól függően) törlődik.
- **/usr:** Unix System Resources, a programok saját könyvtárai, fájljai vannak itt. Alatta majdnem a teljes FSH le van duplikálva, valamint – technikai okokból – létezik **/usr/local** a gépfüggő és **/usr/share** a gépre nézve nem egyedi fájloknak.
- **/var:** gyakran változó tartalmú adatfájlok vannak itt; pl. naplófájlok (logok), mailspool, print spool, ...

Védelmi rendszer

Mivel a Unix (és természetesen a Linux is) egy többfelhasználós rendszer, ezért szükség van az adatok védelmére. Egyrészt biztosítani kell a fájlok védelmét a háttértárolókon, másrészt védeni kell a processzeket a memóriában. A védelem kiterjed olvasásra, írásra és végrehajtásra. Minden felhasználónak van egy egyedi azonosítója (UID) és ezenkívül csoportok is definiálhatók (GID). Például a root privilegizált felhasználó azonosítója 0, a normál felhasználók azonosítói 1000-től indulnak. Egy felhasználó több csoporthoz is hozzárendelhető, az aktív csoportok a **groups** paranccsal kérdezhetőek le. A megfelelő azonosítók (UID, GID) a felhasználó azonosításakor, bejelentkezéskor kerülnek meghatározásra, és a felhasználó által indított összes processz ezeket örökli.

Minden fájlnak van egy tulajdonosa (user), általában az a felhasználó, aki létrehozta; van egy felhasználói csoportja; és van egy védelmi kódja. A védelmi kód meghatározza, hogy a tulajdonos, a csoporttársak és bárki más milyen műveleteket (olvasás, írás, végrehajtás) végezhetnek a fájlon. A védelmi kódokat például az **ls -l** parancs segítségével tudjuk lekérdezni:

-rwxr-xr-- 1 user group ...

Ez a fájl a tulajdonos által olvasható (r), írható (w) és végrehajtható (x); a csoporttagok csak olvashatják és végrehajthatják; míg a többi felhasználó csak olvashatja. Védelmi szempontból a katalógusok hasonlóan kezelhetőek, de ebben az esetben a végrehajtási jog helyett keresési jogról beszélünk (van-e joga a felhasználónak a katalógus bejegyzéseinek eléréséhez) és általában az olvasási joggal együtt használjuk. Fontos megjegyezni, hogy egy adott fájl védelme nem függ az őt tartalmazó katalógus védelmétől.

Egy fájl tulajdonosa a **chown** parancs segítségével modosítható:

chown user:group file

pl.: chown jozsi:hallgato text.txt

A jogosultságok beállítására a **chmod** parancs használható:

chmod [u|g|o][+|-|=][r|w|x|...] file

ahol az egyes felhasználói körökhöz (u/g/o, tulajdonos/csoport/mások) bizonyos jogosultságok (r/w/x) adhatók (+), illetve elvehetők (-). Például az alábbi paranccsal a tulajdonosnak és a csoportjának írási és végrehajtási jogot is adunk (az eddigi jogok mellé):

chmod ug+wx file

A jogosultságokat egy háromjegyű oktális számmal is megadhatjuk (ahol az egyes számjegyek a tulajdonoshoz, csoporthoz, illetve a többiekhez tartoznak, míg maga az érték az r/w/x jogosultságokat, mint bináris helyiértékeket tekintve kapható meg).

Alapvető parancsok

Az alábbiakban a leggyakrabban használt, alapvető parancsok kerülnek bemutatásra. A parancsok legtöbbször a következőképpen épülnek fel:

parancs kapcsolók argumentumok ...

pl.: ls -l *.txt (részletes lista a .txt végződésű fájlokról)

A kapcsolók legtöbbször a '-' jellel kezdődnek, míg ha egy fájlnév helyén áll a '-' karakter, akkor az a standard inputot vagy standard outputot jelenti.

Hasznos parancsok

man

A leghasznosabb parancs, minden UNIX-alapú rendszer részét képezi. Felhasználói kézikönyv, mely az összes parancs, függvény, API hívás leírását tartalmazza, valamint a főbb konfigurációs fájlokét. A *man* oldalakról a *q* billentyű lenyomásával lehet kilépni. A következőkben bemutatásra kerülő parancsoknak érdemes utánanézni ebben a kézikönyvben. Például a **man ls** parancs segítségével az **ls** listázó parancs használatáról kapunk részletes leírást.

mc

Midnight Commander – könnyen használható fájlkezelő program sok hasznos segédfunkcióval.

shutdown, halt, reboot, poweroff

Rendszer leállítása vagy újraindítása.

su, su username; sudo, sudo -u username

Superuser jogosultság megszerzése (**su**), illetve adott felhasználói jogosultság megszerzése. A **sudo** használatával egy parancs hajtható végre az adott jogosultsággal.

Fájlrendszerrel kapcsolatos műveletek

cd, pwd, mkdir, rmdir, ls, find, tar

Könyvtárműveletek: aktuális könyvtár megváltoztatása, aktuális könyvtár kiíratása, könyvtár létrehozása, üres könyvtár törlése, könyvtár tartalmának kiíratása, fájlok keresése a könyvtár hierarchiában nevük vagy tulajdonságaik alapján, könyvtárak archiválása vagy visszaállítása. Nem üres könyvtárak törlésére a fájltörlési parancs rekurzív változatát kell használni (**rm -r**).

Például az alábbi parancs az aktuális könyvtártól (**.**) rekurzívan keresi a .html fájlokat és a talált fájlokról részletes információt ad (mindegyik találatra végrehajtja az **ls -l** parancsot):

find . -name '*.html' -exec ls -l '{}' \;

touch, rm, cp, mv, ln

Fájlműveletek: fájl létrehozása vagy "megérintése" (dátumok aktuálisra állítása rajta), törlése, másolása, mozgatása, linkelése (szimbolikus link létrehozásához használjuk a **-s** opciót). Utóbbiak paraméterei mindig forrás – cél sorrendben követik egymást. A **-r** (vagy **-R**) kapcsolóval lehet rekurzívan (alkönyvtárakkal együtt) végeztetni fájlműveleteket.

Például az alábbi paranccsal szimbolikus linket hozhatunk létre a szülő könyvtárban elhelyezkedő **prog** fájlra **proglink** néven:

ln -s ../prog proglink

df, du, fsck, mount, umount

Adminisztratív műveletek: fájlrendszerek diszkhasználata, adott könyvtárak diszkhasználata (alkönyvtárakkal együtt), fájlrendszer ellenőrzése, fájlrendszer csatolása, illetve leválasztása.

Processzkezelés

ps, kill, top

Processzlista kiíratása, egyes processzek megölése. Leggyakoribb kombináció a **ps ax**. A **kill** parancs paramétere a **ps** által mutatott process ID. Ha a **kill** hatására nem hal meg a processz, úgy a **kill -9** kombináció még segíthet (ez a kötelező érvényű felfüggesztés jelzése). A **top** paranccsal monitorozhatjuk az éppen aktuális processzeket, kilépés **q** billentyűvel.

watch, sleep

A **watch** a paraméterében megadott parancsot adott időnként (alapértelmezésben ez 2 sec) lefuttatja, és a kimenetét megmutatja; míg a **sleep** adott ideig alszik (az időt másodpercekben kell megadni).

Hálózatkezeléshez kapcsolódó parancsok

ssh, scp, ftp

Kapcsolódás távoli gépekhez ssh, scp, illetve ftp protokoll segítségével. Ha távoli gépre jelentkezünk be és grafikus megjelenítést is szeretnénk a lokális gépen, használjuk az **ssh** parancsot a **-X** kapcsolóval ("enables X11 forwarding").

ifconfig, route

A hálózati interfészek beállítására, illetve a routing tábla elérésére szolgáló parancsok. A **man ifconfig**, ill. **man route** parancsok kiadásával pontos leírását és paramétereit vizsgálhatjuk meg.

dhclient, pump

Mindkét paranccsal IP-címet kérhetünk az adott hálózati interfészre. Szintaxisuk: **dhclient eth0**, illetve **pump -i eth0**.

mii-tool, arp

A felhúzott interfészek, illetve az arp tábla megmutatására szolgáló parancsok. Hasznos, ha az adatkapcsolati vagy fizikai szinten tapasztalunk problémát, illetve ha nem tudjuk eldönteni, hogy melyik hálókártya melyik eszköznek felel meg.

tcpdump

Hálózati forgalom monitorozására szolgáló parancs. Általában **-v** (bőbeszédű) és **-i any** (minden interfészen való figyelés) paraméterekkel hívjuk, és a kimenetét a **grep** paranccsal szűrjük. Hálózati hibák elhárításánál nagyon hasznos eszköz.

Szűrők

A Unix rendszerek nagyon hasznos eszközei a szűrők. Ezek olyan programok, melyek a standard bemenetüket a megfelelő művelet elvégzése után a standard kimenetükre másolják. A Unix rendszerekben számos egyébként önmagában nagyon egyszerű műveletet megvalósító szűrő van, és általában a segédprogramok képesek szűrőként is működni. Ezek a szűrők egymás után kapcsolhatók a pipe (csővezeték segítségével).

A shell által végrehajtott programok alapból három megnyitott állománnyal indulnak: standard input (0), standard output (1) és standard error (2). A bemenet/kimenet átirányítására a következő eszközökkel (*redirection metacharacters*) van lehetőség (Bourne Shell Family esetén):

- **prog < file**: standard input átirányítása (vagy hosszabban: **0<**)
- **prog > file**: standard output átirányítása (vagy hosszabban: **1>**)
- **prog 2> file**: standard error átirányítása
- **prog 2>&1**: standard error átirányítása standard outputba
- **prog 1>&2**: standard output átirányítása standard errorba
- **prog1 | prog2**: pipe, prog1 kimenetének prog2 bemenetére irányítása

- **prog >> file**: standard output hozzáírása (append) a megadott fájlhoz

echo, cat, tee

Paraméterként átadott szöveg kiíratása (**echo**), illetve fájlok kiíratása és összefűzése (**cat**). Gyakran használjuk a standard output átirányításával, vagy pipe-okkal együtt. A **tee** parancs a standard inputról másol a standard outputra, valamint a paraméterként megadott fájlokba is (adatfolyam elágaztatása).

Az alábbi példában a cat program kimenetét átirányítjuk az f1 fájlba, így a standard bemeneten bevitt sorok az adott fájlba íródnak egészen a fájlvége jel (ctrl-d) beviteléig:

cat >f1

more, less

Fájlok kiíratása úgy, hogy egyszerre egy képernyőnyi tartalom jelenik meg, illetve navigálási lehetőség biztosítása. A **less** a kifinomultabb változat.

head, tail

Fájlok első (**head**), illetve utolsó (**tail**) n sorának kiíratása.

tr

Alapértelmezésben karakterfordítást végez (translate): az első paraméterként megadott karaktereket cseréli a második paraméterben megadottakra. Tartomány is megadható, pl. **[0-9]** a számokat jelenti, **[a-z]** a kisbetűket. Ha az első paramétere **-d**, akkor törli a második paraméterben megadott karaktereket. Pipe részeként vagy átirányítással használjuk.

wc

Kiírja a sorok, szavak és karakterek számát ("word count").

cmp, diff, comm

Fájlok összehasonlítása: bájtról bájtra (**cmp**) vagy szöveges fájlokat sorról sorra (**diff**). A **comm** parancs két fájl közös sorainak kiíratására használható.

sort, uniq

A két parancsot általában együtt (egymásba pipe-olva) használjuk és ilyen sorrendben: a **sort** rendezi a bemenetet, míg a **uniq** a rendezett bemenet ismétlődő soraiból csak egyet-egyet hagy meg. A **uniq** paraméterezésével többféle működés is elérhető, pl. a sorok különféle számolása (pl. **uniq -c**), csak a többször szereplő (**uniq -d**), vagy az egyedi sorok kiíratása (**uniq -u**). A **sort** paraméterezésével számok és stringek rendezése is megoldható.

Például az alábbi egymás után kapcsolt szűrők a jelszófájlt rendezik a 3. oszlop szerint (**-k3**) numerikusan (**-n**) csökkenő sorrendben (**-r**) és az utolsó két sor lesz az eredmény (**tail -n 2**). A jelszófájlban a mezők közti szeparátor a **:**, ami a rendezésnél a **-t** kapcsolóval adható meg:

cat /etc/passwd | sort -t: -n -k3 -r | tail -n 2

grep, egrep, fgrep

Reguláris kifejezés-illesztő. A paraméterben (idézőjelben!) megadott reguláris kifejezésre (regexp) illeszti a bemenetet. Fontosabb paraméterei: **-A**, **-B**: melyekkel az illesztett sor környezetét (előző / következő, adott számú sorokat) is megmutatja, **-v**: mely fordított működést eredményez (nem illesztett sorokat mutatja), illetve **-q**: mely esetén nincs output, csak a visszatérési értéket állítja be. Ez utóbbit az **if** feltételeként szoktuk használni. A **grep** és **sed** parancsok reguláris kifejezéseiben az operátorokat "escape-elni" kell, különben karakternek tekinti őket a program. Tehát \| a vagy operátor, míg | a pipe karakter. A reguláris

kifejezésekről bővebben a **grep** parancs **man** oldalán olvashatunk, az alábbi példák csak a szemléltetést szolgálják (a sor eleji **$** szimbólum a shell promptját jelöli):

```
$ echo "bcd" | grep "a.*"
$ echo "bcacb" | grep "a.*b"
bcacb
$ echo "baaa" | grep "a*"
baaa
$ echo "baaa" | grep "^a*$"
```

sed

Teljes funkcionalitását tekintve sorszerkesztő, mi reguláris fordítóként fogjuk használni. Erre az **s** parancsa szolgál: **s/kif1/kif2/** a **kif1** reguláris kifejezést fordítja **kif2** kifejezésre mindazon sorokon, amelyekre **kif1** illeszkedik. A(z escape-elt) zárójelbe tett kifejezésrészletekre vissza lehet hivatkozni **kif2** -ben a **\1, \2, ...** referenciákkal. Ha az **s** parancs záró / -je után még egy **g** paramétert írunk, akkor soronként többször is végez illesztést.

Példák:

```
$ echo 'xxxaaaxxx' | sed 's/aaa/bbb/'
xxxbbbxxx
$ echo "a0001b" | sed 's/a\([0-9]*\)b/x\1y/'
x0001y
```

Reguláris kifejezések

c	Maga a c karakter, ha az nem speciális karakter.
\c	Kikapcsolja a c karakter speciális jelentését. Pl: **\[**: zárójel kezdődik
^	Sor eleje.
$	Sor vége.
.	Egy darab bármilyen karakter.
	(Az újsor kivételével minden karakter illeszkedik rá.)
[abc]	Bármelyik karakter a halmazból.
[^abc]	Bármelyik karakter, amelyik nincs a halmazban.
[a-z]	Bármelyik karakter a megadott tartományból.
r*	r reguláris kifejezés tetszőlegesen sokszor (akár 0-szor).
r+	r reguláris kifejezés 1-szer vagy sokszor. (extended regexp)
r?	r reguláris kifejezés 0-szor vagy 1-szer. (extended regexp)
r1r2	r1 és r2 egymás után úgy, hogy r1 a lehető leghosszabban illeszkedjen.
r1¦r2	r1 vagy r2. (extended regexp)
(...)	egymásba ágyazott kifejezések. (extended regexp)
r{n}	r reguláris kifejezés n-szer megismétlődik. (extended regexp)
r{n,}	r legalább n-szer megismétlődik. (extended regexp)
r{n,m}	r legalább n-szer, legfeljebb m-szer megismétlődik. (extended regexp)
\(r\)	r reguláris kifejezés önmaga, amire később hivatkozni lehet **\n** alakban.
\n	hivatkozás az n-edik **\(r\)** reguláris kifejezésre.

Példák reguláris kifejezésekkel

Az **/etc/passwd** fájlból írassuk ki az összes olyan sort, amelyben az 'r' és a 't' karakterek között tetszőleges számú 'o' szerepel:

$ cat /etc/passwd | grep 'ro*t'

Írassuk ki az aktuális könyvtár összes olyan könyvtárát, amihez mindenkinek írási joga van:

$ ls -l | grep '^d.......w.'

Írassunk ki minden olyan sort, amiben egymás után szerepel ugyanaz a betű:

$ cat /etc/passwd | egrep '(.)\1'

Cseréljünk le minden .conf fájlnév részletet .CONFIG-ra a kimeneten:

$ ls | sed s/.conf/.CONFIG/

Első és második karakter felcserélése egy fájlban:

$ cat file1 | sed 's/\(.\)\(.\)/\2\1/'

Jelszófájl első két mezőjének felcserélése (mező szeparátor a kettőspont):

$ cat /etc/passwd | sed 's/^\([^:]*\):\([^:]*\):/\2:\1:/'

Számoljuk meg melyik login shell hányszor szerepel az **/etc/passwd** fájlban (utolsó oszlop), majd ezt rendezzük csökkenő sorrendbe és írjuk ki a két legelsőt:

$ cat /etc/passwd | sed 's/.*:\([^:]*\)/\1/' | sort | uniq -c | sort -n -r | head -n2

Bash programozási alapok

A shell – ahogy korábban szó volt róla – parancsértelmező és egy programozási nyelv is egyben. Segítségével egyszerűen kialakíthatók ún. shell scriptek, melyekkel a parancskészlet tetszőlegesen bővíthető. Ezek paraméterezhetősége hasonló a normál programokéhoz. A bash a Linux rendszerek egyik alapértelmezett shellje, segítségével egyszerű adminisztrációs programokat tudunk írni. A korábban bemutatott bemenet/kimenet átirányítás, illetve a pipe alkalmazásával nagyon hatékony eszközt biztosít a shell akár komplex feladatok megoldására is.

Bash-ben a sorok lezárhatók enterrel vagy pontosvesszővel. Ha egy parancs helyére hosszabb kódrészletet szeretnénk írni, azt egy **do ... done** blokkba tehetjük, ez a C programok **{ ... }** blokkjaira hasonlít. Ahhoz, hogy egy script futtatható legyen, két dolgot kell megtennünk:

1. Az első sorának hashbang-gel kell kezdődnie, amit a bash elérési útja követ (**#!/bin/bash**).
2. A **chmod a+x script** paranccsal futtathatóvá kell tennünk. (Ha ezt nem tesszük meg, a futtatás a **bash script** paranccsal történhet.)

Mivel az aktuális könyvtár alapértelmezésben nincs benne a **PATH** környezeti változóban, így **./script** módon indíthatjuk a programunkat. (Vagy hozzáadjuk a **PATH** -hoz az aktuális könyvtárat a **PATH=$PATH:.** paranccsal.)

Hasznos shell funkciók

Az alábbi billentyűkombinációkkal hasznos funkciók vehetők igénybe:

ctrl-F1,F2,...	szöveges terminálok közti váltás
ctrl-alt-F1,...}	másik terminálra váltás grafikus terminálról
⇑ és ⇓	history, korábbi parancsok behívása
ctrl-r	history, parancs illesztése az első megfelelőre (reverse search)
TAB	állománynév kiegészítés
shift-PgUP és !PgDown	képernyő tartalmának léptetése
ctrl-a	sor elejére ugrás
ctrl-e	sor végére ugrás
ctrl-l	képernyő újrarajzolása

Változók, idézőjelek

Bash-ben a változókat nem kell deklarálni, név szerint lehet rájuk hivatkozni. Az értékadás szintaxisa **VAR="valami"**, ügyelve arra, hogy az egyenlőségjel baloldalán ne maradjon szóköz. A változó értékét **$VAR** formában lehet lekérni. Standard inputról változóba a **read** paranccsal olvashatunk.

A bash háromféle idézőjelet különböztet meg:

- szimpla idézőjel: **'echo $VAR'**. Az idézett szöveg változtatás nélkül kerül feldolgozásra. A fenti paramétert például egy C programnak átadva az **argv[1]** értéke **echo $VAR** lenne.
- dupla idézőjel: **"echo $VAR"**. Az idézett szövegen lefut a változóhelyettesítés, majd változtatás nélkül kerül feldolgozásra. Például egy C program esetén az **argv[1]** értéke **echo valami** lenne.
- vissza idézőjel: **`echo $VAR`**. Az idézett szöveget parancsként értelmezi, és az eredményét adja át. Most a C programunk **argv[1]** -ben a **valami** szöveget kapná.

Állománynév-helyettesítés

A bash ún. "wildcard"-okat (joker karaktereket) biztosít arra, hogy állományok bizonyos csoportját tudjuk kényelmesen kezelni és ne kelljen állandóan a **grep** parancsot használni az eredmények szűrésére.

*****	tetszőleges számú tetszőleges karakter
?	pontosan egy tetszőleges karakter
[abc]	bármelyik karakter a halmazból
[a-z]	bármelyik karakter az adott intervallumból

Például a következő parancs három képfájlról ad részletes listát, ha az aktuális könyvtárban létezik image1.jpg, image2.jpg és image3.jpg:

$ ls -l image[1-3].jpg

Ezeken kívül használható a szélesebb körű "brace expansion" mechanizmus is, amivel tetszőleges string sorozatok készíthetők. Ilyenkor a kapcsos zárójelen belül vesszővel szeparált sorozat minden eleméhez generálódik egy string oly módon, hogy a kapcsos zárójel

előtti és utáni rész minden elemhez hozzáfűződik. A következő példa az előbbi parancs kicsit módosított változata, most a három képfájl .jpg és .bmp változata is megjelenítésre kerül:

$ ls -l image[1-3].{jpg,bmp}

Tulajdonképpen az előző parancsot a shell a következőre fordítja le:

$ ls -l image1.jpg image1.bmp image2.jpg image2.bmp image3.jpg image3.bmp

Paraméter- vagy változóhelyettesítés

Ahogyan korábban láthattuk, a shellben egy **v** változóra egyszerűen a **$v** segítségével hivatkozhatunk és a megfelelő környezetben – ha nem szimpla idézőjelek között szerepel – megtörténik a változóhelyettesítés, amikor a **$v** helyére a **v** változó értéke kerül. A változóra hivatkozhatunk a **${v}** kifejezéssel is, amire akkor lehet szükség, ha az adott környezetben nem lenne egyértelmű, hogy hol ér véget a változónév. Pédául ha a **v1** változó értéke "x.htm", akkor a **v2=${v1}l** hatására **v2** változó értéke "x.html" lesz.

Az alábbi speciális változók mindig rendelkezésre állnak a bash scriptben:

$0	scriptfájl neve
$\#	bemeneti paraméterek száma
$i	az i-edik bemeneti paraméter
$?	utolsó paraméter
$@** és **$*	az összes paraméter
$\$	processz ID
$HOME	home könyvtár
$HOSTNAME	gép hosztneve
$PATH	elérési utak az állományokhoz
$UID	aktuális user ID
$PS1	aktuális prompt
$IFS	input mező szeparáló karakter

Parancshelyettesítés

A parancshelyettesítés segítségével lehetőség van arra, hogy bizonyos stringeket a shell parancsként értelmezzen és helyettesítse a lefuttatott parancs eredményével. Korábban láthattuk, hogy erre az egyik lehetőséget a vissza idézőjel nyújtja. Ezenkívül a következő, sima zárójeles formátum is használható: **$(parancs)**. Például a következő parancsok bármelyikének hatására a **current_dir** változó az aktuális könyvtár elérési útjára lesz beállítva:

```
current_dir=$(pwd)
current_dir=`pwd`
```

Aritmetikai helyettesítés

A bash lehetőséget biztosít az egészértékű (integer) aritmetikai kifejezések kiértékelésére és helyettesítésére is. Ezt a **$((kifejezés))** formátum használatával érhetjük el. Például, az alábbi shell script hatására a **c** változó az első két bementi argumentum értékének összegét veszi fel, és ezt kiírja a standard kimenetre:

```
#!/bin/bash
c=$(($1 + $2))
echo $c
```

Érdemes megjegyezni, hogy a dupla zárójelen belül a változóhelyettesítés a kiértékelés előtt megtörténik. Itt bármilyen shell változót használhatunk és a **$** jel használata nem kötelező. Ha shell változó szerepel értékadás műveletben, akkor értelemszerűen nem szabad a változóhelyettesítést kérő **$** szimbólumot használni. Például a **$((c=$var1+$var2))** értékadási műveletben a c változót nem szabad helyettesíteni, ezért nem használjuk előtte a **$** -t.

Ezenkívül használhatjuk aritmetikai kifejezések kiértékelésére (nem helyettesítésre) a beépített **let**, illetve a külső **expr** parancsot is.

Beépített parancsok

Az alábbiakban néhány hasznos beépített parancs kerül bemutatásra:

:	nem csinál semmit
. vagy **source**	más fájlok include-olása
alias/unalias	alias beállítása / eltávolítása
break/continue	ciklus elhagyása / következő iteráció
cd	aktuális könyvtár megváltoztatása
echo	argumentumok kiírása
eval	argumentum mint parancs végrehajtása
exec	argumentum végrehajtása, de nem indul új shell
exit	kilépés
export	shell változó exportálása
bg/fg	job háttérbe / előtérbe helyezése
let	aritmetikai kifejezés kiértékelése
pwd	aktuális könyvtár lekérdezése
read/readonly	változóba olvasás standard inputról
return	visszatérés függvényből
set/unset	változók lekérdezése / beállítása
shift	pozicionális paraméterek léptetése
test vagy **[ ]**	feltétel kiértékelése
times	futási idők

Vezérlési szerkezetek

A bash támogatja a szokásos strukturált vezérlési szerkezeteket, mint az **if, for, while, case**. Ezek közül a **for** érdemel külön említést, mivel az inkább foreach-nek felel meg. Az alábbi példák szemléltetik a szerkezeteket, további segítséget a man bash paranccsal kaphatunk.

if

```
if [ "$VAR" == "valami" ];  then
 echo "be van állítva"
else
 echo "nincs"
fi



if grep -q "hello"; then
 echo "szia"
fi
```

for

```
for i in a b c; do
 echo $i
done
```

```
for i in *; do
 echo $i
done
```

while

```
while read v; do
 echo "Új sor jött: $v"
done
```

case

```

case "$VAR" in
hello)
 echo "szia"
;;
bye)
 echo "bye-bye"
;;
*)
 echo "nem értem"
;;
esac
```

Feltétel kiértékelése

A feltételes kifejezések kiértékelésére – ahogyan az előbbi példákban is látható – a **test** parancs vagy a szögletes zárójel **[...]** használható. A parancs a következő fontosabb operátorokkal használható:

test s	igaz, ha s nem null string
test -z s	igaz, ha s nulla hosszúságú string
test -n s	igaz, ha s nem nulla hosszúságú string
test s1=s2	igaz, ha s1 string megegyezik s2-vel
test s1!=s2	igaz, ha s1 string nem egyezik meg s2-vel
test n1 -eq n2	igaz, ha n1 aritmetikailag egyenlő n2-vel
-ne, -gt, -ge	nem egyenlő, nagyobb, nagyobb vagy egyenlő
-lt, -le	kisebb, kisebb vagy egyenlő
test -f	igaz, ha f file létezik és nem könyvtár
test -r	igaz, ha f file létezik és olvasható
test -w	igaz, ha f file létezik és írható

Ezek a műveletek természetesen kombinálhatók a következő logikai műveletekkel: **!** (tagadás), **-o** (vagy), **-a** (és).

Egyszerű példaprogramok

Az alábbi egyszerű shell script kiírja az egész számokat 1-től 100-ig:

```
#!/bin/bash
c=1
while [ $c -le 100 ]
do
 echo $c
 c=$(($c+1))
done
```

A következő példaprogram kiírja, hogy hány fájl van az aktuális könyvtárban:

```
#!/bin/bash
n=0
for i in *; do
 if [ -f $i ]; then
 n=$(($n+1))
 fi
done
echo "$n fájl van a könyvtárban"
```

Rendszerkonfigurálás, szolgáltatások installálása, beállítása

Innen rendszergazda jogosultságra van szükség, ami **sudo –i** paranccsal állítható be.

FIGYELEM! A root filesystem nfs-en keresztül van csatolva az alaplapi interfészen. Tehát az alaplapi interfészt nem szabad lekapcsolni vagy átkonfigurálni, mert az a rendszer leállásához vezethet. A session adatai alapesetben a memóriába írodnak, így a gépet nem célszerű kikapcsolni.
Viszont a ~/mydata könyvtár a lokális diszkre linkel, ezért az ott tárolt adatok megmaradnak.
Célszer? a jegyz?könyvet itt tárolni.

Debian disztribúció

Számos Linux disztribúció létezik, ami egy adott készítőtől vagy gyártótól származó összeállítás, amely egy GNU/Linux alaprendszert, és ahhoz tartozó, bizonyos szempontok szerint válogatott és testreszabott programokat (és gyakran egyéni kernelt) tartalmaz. Néhány népszerű disztribúció például: Debian GNU/Linux, Ubuntu, openSUSE, Red Hat Linux, Knoppix, stb. A mérés során a Debian disztribúciót fogjuk használni.

A Debian előnyei:

- teljesen ingyenes
- számtalan résztvevő tartja karban
- fejlett csomagkezelő rendszerrel rendelkezik, ami könnyűvé teszi a programok frissítését, telepítését és eltávolítását.

Név eredete: A Debian a Debra (Ian Murdock felesége) és a Ian szavakból áll össze. Először: 1993. augusztus 27-én született meg egy levél, amelyben Ian Murdock írta le a Debian szót.

Három verzió létezik:

- A "stable" az aktuális ajánlott verzió, legtöbbször csak biztonsági frissitések jelennek meg hozzá. Mivel az új Debian verziók megjelenése jelenleg hosszas ellenőrzési procedúrát igényel, ezért a stabil változat gyakran régebbi programokat tartalmaz.
- A "testing" a váróterem a következő "stable" előtt, először ide kerülnek azok a csomagok az "unstable"-ből, melyeknél egy adott idő alatt nem került elő hiba; végül ebből fejlődik majd ki a következő stabil kiadás.
- A "unstable" változat a legfrissebb programokat tartalmazza, azonban mivel ezeket még a közösség nem tesztelte alaposan, semmi garancia nincs arra, hogy a csomagok működnek egyáltalán, nemhogy helyesen. Csak bátraknak és tapasztaltaknak ajánlott, hiszen előfordulhat, hogy egy hibás csomag után a rendszerben kézzel kell javításokat végezni ahhoz, hogy működőképes maradjon.

Kiadások (ezek a kódnevek a Pixar nagysikerű animációs filmjéből, a Toy Story-ból valók):

- Debian 3.1: kiadás 2005. június 6., kódneve "sarge" (obsolete stable release)
- Debian 4.0: kiadás 2007. április 8., kódneve "etch" (obsolete stable release)
- Debian 5.0: kiadás 2009. február 14., kódneve "lenny" (obsolete stable release)
- Debian 6.0: kiadás 2011. február 6., kódneve "squeeze" (**current stable release**)
- Debian 7.0: kódneve "wheezy" (**current testing release**)

- "unstable" kiadás: "sid"

Installálás

A Debian Linux rendszer installálására több lehetőség is kínálkozik. Ha a számítógépünk rendelkezik Internet kapcsolattal, akkor a legegyszerűbb hálózati installálást (*netinst*) végezni. Ilyenkor egy kisebb méretű install image fájlt kell letölteni (például innen), amit CD/DVD-re, pendrive-re, stb. másolás után lehet indítani a telepítési folyamatot. A rendszerhez szükséges összetevők a megadott debian mirror szerverekről kerülnek majd letöltésre.

A telepítési lépések során pontos üzeneteket kapunk arról, hogy mi történik és lehetőségünk van a tetszőleges testreszabásra. Ezen lépések során értelemszerű válaszokat kell adni a telepítőnek, melynek részletezésére jelen dokumentum nem tér ki. A telepítés kipróbálásának legegyszerűbb módja egy virtuális gépre történő installálás (pl. VirtualBox, VMWare stb.), ilyenkor a meglévő rendszerünk épségét nem kockáztatjuk egy esetleges hibás konfigurálással. Mindenkit bátorítunk ennek kipróbálására!

Installáláshoz kapcsolódó programok

A Linux rendszerekben nagyon sok segédprogram áll rendelkezésre, melyek az installálás előkészítése, illetve az installálás során használhatók. Ezekre az első fázisban sokszor nincs is szükség, mivel a telepítő elrejti előlünk ezeket a feladatokat. A későbbiekben azonban, például egy rendszer átkonfigurálása, új hardverrel történő bővítése során, szükség lehet rájuk. A következőkben a legfontosabb kapcsolódó programokat tekintjük át.

Partíciók kezelése

A Linux rendszerekben a hardverekhez tartozó fájlbejegyzések a **/dev** könyvtár alatt vannak. Korábban láttuk, hogy a UNIX-alapú rendszerek alapkoncepciója szerint *minden* egy fájl, ezzel biztosítva az egységes kezelést. Így a hardvereszközökhöz is tartoznak megfelelő fájlbejegyzések. Például az első SATA merevlemezhez a **/dev/sda**, míg a másodikhoz a **/dev/sdb** fájl tartozik. Régebbi IDE HDD-k esetén a fájlnevek: **/dev/hda**, **/dev/hdb**, stb. Az első SATA merevlemez partícióira a következő bejegyzések vonatkoznak: **/dev/sda1**, **/dev/sda2**, stb.

A partíciók kezelésére (létrehozás, törlés, lekérdezés, stb.) például az **fdisk** vagy a **cfdisk** programok használhatók. Például a *secondary slave* IDE vezérlőn lévő HDD partíciós táblájának lekérdezése a következő paranccsal történhet:

fdisk -l

A legegyszerűbb telepítés során két partíciót érdemes létrehozni a Linux rendszerünk számára. Az egyiket használhatjuk a rendszer és adataink tárolására, a másikat pedig ún. **swap** területnek, melyet a rendszer virtuális memória kezelője használ. Nagyobb rendszereknél természetesen célszerű több partíciót is létrehozni, melyeken különböző komponensek tárolhatók. Például érdemes különválasztani az operációs rendszert és a felhasználói adatokat (**/home** könyvtár), esetleg szükség lehet egy külön boot partícióra (**/boot**), stb. A partíciók menedzselésére a telepítés során egy felhasználóbarát felületen van lehetőség, de vigyázzunk, mert a meglévő rendszerünket könnyen elérhetetlenné tudjuk tenni!

Fájlrendszerek kezelése

A Linux rendszerek állományrendszeréről az első mérés segédletében találhatók információk: itt.

A létrehozott partíciók különböző célra használhatók, azokon különböző fájlrendszereket tudunk létrehozni. A következő részben azt feltételezzük, hogy az első SATA HDD-n három

partíciót hoztunk létre: **sda1,sda2,sda3**. Az elsőre az operációs rendszer kerül, a másodikat használjuk swap területnek, a harmadikon pedig az adataink lesznek tárolva.

A swap partíció létrehozására az **mkswap** parancs használható, ami a konkrét példánál a következő módon néz ki:

mkswap /dev/sda2

A swap fájlrendszer létrehozás után a **swapon /dev/sda2**, illetve a **swapoff /dev/sda2** parancsok segítségével vehető használatba, illetve kapcsolható ki. Az aktuális memória és virtuális memória használat lekérdezése a **free** paranccsal történhet.

A partíciók használatba vétele előtt azokon fájlrendszert kell létrehozni. A Linux rendszerekben több fájlrendszer is támogatott, a példákban az egyik legelterjedtebben használt **ext3** fájlrendszert fogjuk használni. Például **ext3** típusú állományrendszert az alábbi parancsokkal hozhatunk létre:

mkfs.ext3 /dev/sda1

mkfs.ext3 /dev/sda3

Amint láttuk, a GNU/Linux rendszerek állományrendszere a tradicionális hierarchikus Unix állományrendszert követi, amely egy fa struktúrába képezi a fájlokat. Egyetlen gyökér (root, "/") van a rendszerben, ez a fájlrendszer kezdőpontja. Az alap állományrendszerhez tetszőleges kötetek csatlakoztathatók a fa struktúra tetszőleges pontjainál. Ehhez a **mount** parancsot használhatjuk. Feltételezzük, hogy a telepítés során a **/dev/sda1** partíciót választottuk, hogy az itt létrehozott fájlrendszer legyen a gyökér pontba csatolva., viszont az adatainkat az **sda3** partíción szeretnénk tárolni. Ekkor az **sda3** partíción létrehozott fájlrendszert csatlakoztathatjuk például a **/home** könyvtárhoz az alábbi paranccsal:

mount /dev/sda3 /home

Így a **/home** könyvtár tartalma egy külön partíción lesz tárolva egy külön fájlrendszerben. A **mount** parancs segítségével különböző típusú fájlrendszerek is csatlakoztathatók, nézzünk egy-két példát!

A **mount** parancs leggyakrabban használt formája:

mount -t type device dir

ahol első paraméterként a fájlrendszer típusát (pl. ext3, vfat, stb.) adhatjuk meg (ha nem adjuk meg, a program megpróbálja kitalálni), majd a hardvereszközt és a csatlakoztatási pontot.

mount -t ntfs /dev/sda7 /srv/windows-data	ntfs kötet csatolása a megfelelő könyvtárba
mount -t iso9660 /dev/cdrom /media/cdrom	cd csatolása
mount -t nfs -o proto=tcp 192.168.1.100:/srv/server-data /srv/nfs-server	nfs szerver becsatolása

A **mount** parancs paraméterek nélküli futatásával arról kapunk képet, hogy jelenleg milyen kötetek milyen csatlakozási pontokon és milyen paraméterekkel vannak csatolva.

A fájlrendszerek fontosabb paraméterei, mint például a foglaltság, szabad kapacitás paraméterek a **df -h** paranccsal kérdezhetők le.

Fontos tudni, hogy alapesetben csak a root felhasználónak van jogosultsága kötetek csatolására, azonban az **/etc/fstab** fájlba előre beállíthatók különböző kötetek és azok

csatlakoztatási paraméterei. Ilyen paraméter lehet például, hogy a kötetet indulásnál automatikusan csatlakoztassa a rendszer vagy sem (noauto), vagy az, hogy a kötet csatolására joga van a normál felhasználóknak is (user). Ezenkívül természetesen több paraméter is beállítható. Egy tipikus **fstab** fájl valahogy így néz ki:

# /etc/fstab: static file system information.					
#					
# file system	mount point	type	options	dump	pass
proc	/proc	proc	defaults	0	0
/dev/sda1	/	ext3	errors=remount-ro	0	1
/dev/sda3	/srv	ext3	defaults	0	2
/dev/sda2	none	swap	sw	0	0
/dev/scd0	/media/cdrom0	udf,iso9660	user,noauto	0	0
/dev/fd0	/media/floppy0	auto	rw,user,noauto	0	0

Bővebb információ a **man fstab** és **man mount** parancsokkal kérhető.

Bootloader

Ahhoz, hogy egy adott partícióról indulni tudjon a rendszer, szükséges valamilyen bootloader installálása is. A Linux-os környezetben az egyik legelterjedtebben használt bootloader a **grub**. (Jelenlegi aktuális verziója a **grub2**, ami jelentős változásokat hozott a korábbi verzióhoz képest!) Telepítéskor választani lehet, hogy az MBR-be (Master Boot Record) vagy az installált partícióra telepítsük a grubot. Az MBR-be való telepítéskor figyeljünk arra, hogy a gépünkön lévő esetleges egyéb operációs rendszerek indíthatók maradjanak! (Ehhez a grub konfigurációs fájlját kell megfelelően beállítani.)

Ha pédául már telepítve van az MBR-be egy boot manager program (ami indítani tudja a többi partíciót) és a **/dev/sda1** partíciót szeretnénk bootolhatóvá tenni, akkor azt a következő módon tudjuk megtenni:

- először csatolnunk kell a fájlrendszert (pl. **mount /dev/sda1 /mnt/valami**)
- majd installálni kell a grubot a megfelelő boot szektorba: **grub-install --root-directory=/mnt/valami /dev/sda1**
- végül a grub konfigurációs fájl (**/boot/grub/menu.lst** a példában: **/mnt/valami/boot/grub/menu.lst**) szerkesztésével tudjuk testreszabni a boot menüt és az egyes menüpontokra indítandó beállításokat

A konfigurációs fájl pontos felépítéséről és a beállítási lehetőségekről itt található bővebb információ. (A **grub2** esetében ezek természetesen máshogy történnek.)

Programok installálása

Az egyes Linux disztribúciók az alap rendszeren kívül számos egyéb program, programcsomag telepítését is lehetővé teszik. A különböző disztribúciók különböző csomagkezelő rendszerekkel rendelkezenek, melyeknek az az alapja, hogy távoli szerveren (és azt tükröző mirror szervereken) állandóan elérhetők a programok és ezeket igény szerint egyszerű parancsokkal vagy grafikus interfészen keresztül telepíteni tudjuk meglévő rendszerünkre. A Red Hat-alapú Linux-ok csomagkezelője például az RPM (Red Hat Package Manager), míg a Gentoo Linux a Portage rendszert használja. A Debian és a Debian-alapú (pl. Ubuntu) Linux rendszerek csomagkezelője a **dpkg** (Debian Package Management System). Ez egy parancssorból kontrollálható rendszer, de számos frontend létezik hozzá, melyekkel jóval kényelmesebb módon kezelhetők a csomagok (például grafikus interfészen).

A Debian csomagok **.deb** fájlokban tárolódnak, ezen fájlok elnevezési konvenciója a következő:

zsh_3.1.6.pwsz1-1.deb

Ahol:

- zsh a csomag neve
- 3.1.6.pwsz1: a program verziószáma
- 1: Debian csomagváltozat
- .deb: a fájl kiterjesztése

A csomagok kezeléséhez a következő parancsok használhatók:

Csomag telepítése:

- **dpkg -i fájlnév.deb**: Csomag telepítése:
- **dpkg -i -R /tmp/ezaz**: az adott könyvtár összes deb csomagjának telepítése

Csomag konfigurációja:

- **dpkg --configure csomagnév**: befejezi a megadott csomag beállítását
- **dpkg --configure --pending**: minden függőben lévő csomagot beállít
- **dpkg --reconfigure csomagnév**: adott csomag újrakonfigurálása

Csomag törlése beállítások meghagyásával: (konfigurációs fájljaink nem törlődnek)

- **dpkg -r csomagnév** vagy **dpkg --remove csomagnév**

Csomag törlése beállításokkal együtt:

- **dpkg --purge csomagnév**

Lekérdezések:

- **dpkg -l kifejezés**: telepített csomagok listázása, kifejezés opcionális
- **dpkg -L csomagnév**: milyen fájlokat telepített egy bizonyos csomag
- **dpkg -s fájlnév**: csomag információk (pl status)
- **dpkg -S fájlnév**: melyik csomaghoz tartozik egy bizonyos fájl
- **dpkg -I ezaz.deb**: részletes listát ír ki a debről

Ennél jóval kényelmesebb kezelést biztosít az egyik legelterjedtebben használt frontend, az **apt** (Advanced Package Tool).

Az apt rendszere három fő részből áll:

- konfigurációs fájl a források eléréséhez: **/etc/apt/sources.list**
- **apt-get**: telepítés
- **apt-cache**: keresés

A **sources.list** olyan ftp és http címeket tartalmaz, ahonnan az apt adatok kérhetőek le. Egy bejegyzés formátuma a következő:

deb http://debian.mirrors.crysys.hu/debian/ lenny main contrib non-free

1. mező: Típus

- **deb**, ha befordított csomagokat keresünk
- **deb-src**, ha a forráscsomagokat nézzük

2. mező: URI, ez a Debian könyvtárszerkezet gyökere.

3. mező: Debian változata, pl:

- Stable
- Frozen
- Unstable
- Disztribúció neve (pl. lenny, sid)

4. és következő mezők: Könyvtárak, pl:

- main
- contrib
- non-free

A csomagok telepítése, menedzselése a következő parancsokkal történhet.

apt-get

Automatikusan kezeli a csomagok közötti függőségeket és ellentmondásokat. Például, ha szeretnénk telepíteni egy csomagot, ami más, még nem telepített csomagokat is igényel, akkor a program megoldja ezek installálását is.

- **apt-get update**: csomaglista frissítése a **sources.list** alapján (például új forrás hozzáadásánál szükséges)
- **apt-get install csomagnév**: csomag telepítése
- **apt-get -f install**: csomagok és függőségeik helyrehozása
- **apt-get source csomagnév**: forráscsomag telepítése, ha van **deb-src** sor
- **apt-get remove csomagnév**: csomag eltávolítása, de konfigurációs fájlok maradnak
- **apt-get purge csomagnév**: csomag eltávolítása konfigurációs fájlokkal együtt
- **apt-get upgrade**: az összes csomag automatikus frissítése
- **apt-get dist-upgrade**: teljes Linux-változat újratelepítése céljából készült, és automatikusan újraszervezi az összes megváltozott nevű csomaghoz tartozó függőségeket

apt-cache

- **apt-cache show csomagnév**: csomagokról szóló adat keresésére használható

- **apt-cache depends csomagnév**: listát ad arról, hogy mely más csomagok telepítésére van szükség a *csomag* telepítéséhez és helyes működéséhez
- **apt-cache search karaktersorozat**: keresés a csomagok között

Más frontendek is léteznek, melyekkel kényelmesen tudjuk menedzselni a program-csomagokat, mint például az **aptitude**, a **dselect** vagy az X Window alatt működő grafikus csomagkezelő frontendek, mint például a **synaptic**, a **gnome-apt** vagy a **kpackage**.

Felhasználók kezelése

A Linux - a UNIX alapokból fakadóan - egy többfelhasználós operációs rendszer. Egy kitüntett felhasználóval bír, amely neve általában **root**. Ő az a felhasználó aki korlátlan jogosultsággal bír a rendszer felett. Egy adott számítógépen definiált felhasználók listáját megtekinthetjük, ha kilistázzuk pl. cat parancs segítségével az **/etc/passwd** állományt. A kezdeti disztribúciók ebben az állományban tárolták az egyes felhasználókhoz tartozó kódolt jelszavakat is, ám ez lehetőséget biztosított a jelszavak offline törésére, így mára ezek átkerültek az **/etc/shadow** állományba, (olvasására vagy írására csak a root jogosult).

Felhasználó jelszavának megváltoztatására a **passwd** parancsot használhatjuk. A root tetszőleges felhasználó jelszavát módosíthatja, ha a passwd után írja a kívánt felhasználó nevét is.

Felhasználó hozzáadása **adduser** parancs segítségével történhet. A **--home** kapcsoló után megadható az új felhasználó home könyvtárának elérési útja, valamint a **--shell** kapcsoló után definiálható az új felhasználó alapértelmezett parancsértelmezője. Egy felhasználó törlését **deluser** paranccsal végezhetjük el. Itt két fontosabb kapcsolóra érdemes felhívni a figyelmet: a **--remove-home** segítségével a felhasználó home könyvtára is törlésre kerül, a **--remove-all-files** kapcsolóval pedig az összes a törlendő felhasználó tulajdonában lévő állományt törli.

A Linux lehetőséget nyújt felhasználói csoportok kialakítására is. Az **id** parancs kiadásával tekinthetjük meg, hogy az a felhasználó, akinek a nevében éppen bejelentkeztünk, mely csoportoknak a tagja. A hozzáférési jogosultságok tárgyalásánál láthattuk, hogy a felhasználói csoportokhoz éppúgy rendelhetünk jogosultságokat, mint az egyes felhasználókhoz.

Csoport létrehozása a **groupadd [csoportneve]** parancs használtával történhet. Míg egy adott felhasználót az **usermod -a -G [csoport_név] [felhasználó_név]** paranccsal adhatunk hozzá egy kiválasztott csoporthoz.

Hálózat konfigurálása

A linux esetében minden hálókártyához egy - egy hálózati interfész tartozik, ezek elnevezése általában **eth0** -tól kezdődik, ám ettől eltérő is lehet. Valamely hálózati interfész IP címének beállítására az **ifconfig** parancsot használhatjuk, melynek leggyakrabban használt szintaktikája a következő:

ifconfig [interfész_neve] [IP_cím] netmask [netmask] up

ifconfig	az aktív interfészek kilistázása
ifconfig -a	a számítógépben lévő összes interfész kilistázása
ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up	az eth0 interfész IP címének beállítása és annak "felhúzása"
ifconfig eth0 down	az eth0 hálózati interfész "letiltása"

Az útvonalválasztó (routing) tábla kezelésére a **route** parancs szolgál. Paraméter nélküli meghívása esetén kilistázza a routing tábla éppen aktuális bejegyzéseit (megjegyzés: **-n** kapcsolóval kikapcsolhatjuk a DNS feloldást ez sok esetben meggyorsítja a parancs végrehajtását). Például:

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.0.0	*	255.255.255.0	U	1	0	0	eth0
link-local	*	255.255.0.0	U	1000	0	0	eth0
default	192.168.0.1	0.0.0.0	UG	0	0	0	eth0

Az útvonalválasztó tábla módosításához a parancs általában használt szintaktikája a következő:

route add|del [-net|-host] target [netmask Nm] [gw Gw] [[dev] If]

route -n	az útvonalválasztó tábla DNS feloldás nélküli kilistázása, így csak IP címek szerepelnek benne
route add default gw 192.168.0.1	192.168.0.1 IP című állomás megjelölése mint alapértelmezett útvonalválasztó
route del -net 169.254.0.0 netmask 255.255.0.0	169.254.0.0-ás tartomány törlése a routing táblából

Amennyiben az általunk használt számítógép dinamikus IP cím hozzárendelést használ (pl.: DHCP) **dhclient** illetve **pump** programok használatával kérhetünk a hálózattól egy éppen nem használt IP címet. Amennyiben egy interfész nevét is megadjuk (**pump** esetében **-i** kapcsoló után), akkor csak az megadott interfészén, ellenkező esetben viszont a számítógép valamennyi interfészén próbál IP címet kérni. (A programok a routing tábla módosítását is elvégzik).

További, a hálózatkezeléshez kapcsolódó programok itt találhatók.

Az eddig bemutatott parancsok segítségével elvégzett beállítások azonban a számítógép újraindításával elvesznek. Annak érdekében, hogy a szükséges beállításokat a rendszer automatikusan minden indítás alkalmával végrehajtsa, azokat az **/etc/network/interfaces** állományban kell rögzítenünk, melynek felépítése alább látható:

/etc/network/interfaces felépítése

```
auto lo
iface lo inet loopback

//Interfész konfigurálása dinamikus IP címmel (DHCP)

auto eth0
iface eth0 inet dhcp

//Interfész konfigurálása statikus IP cím hozzárendeléssel

auto eth1
iface eth1 inet static
address 192.168.1.3
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
up route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.1.1
```

Hálózati programok

Hálózati programok és alkalmazásaik

ping	ICMP echo request és echo reply
traceroute	UDP csomagokat küld ki, TTL-t pedig növeli és nézi az ICMP "time exceeded" üzenetet
ssh	secure-shell: biztonságosan lehet kapcsolódni távoli számítógépekhez
scp	ssh kapcsolaton keresztül fájl másolása
route	csomagok útvonalának meghatározása
ifconfig	linux interfészek (minden hálózati kártyát egy interfész azonosít)
tcpdum	bejövő és kimenő csomagok figyelése adott interfészen

p	
wireshark	tcpdumphoz hasonló, de sokkal komolyabb: protokoll analizátor
tshark	ugyanaz cask text verzió
iptables	tűzfal, hálózati csomagok kezelése

Szolgáltatások

Linux boot folyamata

A Linux rendszer indulása az alábbi (kicsit egyszerűsített) lépések szerint történik (Debian esetében).

1. Boot loaderrel kiválasztjuk, hogy mit szeretnénk betölteni
2. Bebootol a kernel a megadott initramdisk-kel (ami egy kisméretű fájlrendszert tartalmaz)
3. Elindul az initram **init** processze, betölti a szükséges drivereket, befordított modulokat, ... stb., majd felcsatolja a (a boot menüben megadott) root fájlrendszert, innentől ez lesz a gyökér
4. Elindul a root fájlrendszer **/sbin/init** processze az **/etc/inittab**fájl alapján
    1. Leellenőrzi a fájlrendszereket és felcsatolja az **/etc/fstab** alapján
    2. Lefuttat előre megadott scripteket. Az **/sbin/init**működése:
        1. beolvassa az /etc/inittab fájlt
        2. lefuttat egy inicializáló szkriptet (**/etc/init.d/rcS**), amiben ez van: **exec /etc/init.d/rc S**, aminek hatására lefuttatja a /etc/init.d/rc scriptet
        3. Ezt követően a default run level alapján (**/etc/inittab** fájlban megadott paraméter) elkezdi lefuttatni az **/etc/rcX.d/** könyvtárban lévő scriptek közül azokat, amelyek **S** betűvel kezdődnek (start). A default run level debian alatt a 2, így az **/etc/rc2.d/** könyvtár tartalmát kezdi el lefuttatni. A sorrendet pedig a fájlok sorrendje határozza meg, ezért először az **S10syslogkd** indul el. Ez lefut, majd jön az **S11klogd**, stb. Innen indul a grafikus felület is (pl. kdm, gdm3, stb.)
        1. Megjegyzés: ezek a fájlok szimbolikus linkek az **/etc/init.d** könyvtárban található vezérlő fájlokra. Tehát ha új scriptet akarunk írni, akkor azt először be kell tenni az **/etc/init.d** -be, majd szimbolikus linket kell csinálni rá az **rc2.d** könyvtárban. Korrektebb megoldás, ha az **update-rc.d** parancsot használjuk ezen linkek manipulálására (upgrade esetén nem íródnak vissza a módosítások). Az **rc** (Run Control) könyvtárakban **K**-val kezdődő linkek is vannak, ezeket az adott level elhagyásakor indítja el (kill). Ehhez nem kell újraindítás.
    2. Az egyes run level (szintek):

0	System Halt	rendszer leáll, ha eléri

1	Single user	szolgáltatások (démonok) nem indulnak el, javításra használható (általában *single* szót kell a kernel comand line után beírni)
2	Full multi-user mode	default
3-5	Same as 2	általában: 3: text, 5: grafikus
6	System Reboot	mint 0, csak a végén reboot van

        - Az aktuális runlevel megnézése a **runlevel** paranccsal történik. Erre a válasz például: **N 2**, ami azt jelzi, hogy nem volt runlevel módosítás indítás óta (**N**) és most a második fut (**2**).
        - Runlevel váltás a **telinit X** paranccsal lehetséges. Például: **telinit 0** - halt, **telinit 6** - reboot
    - Indítja a **getty** folyamatokat, amik kirakják a bejelentkező (login) promptot a a virtuális terminálra. Ezek *respawned* állapotúak, ami azt jelenti, hogy ha leállnak, akkor az **init** egyszerűen újraindítja őket. A Debian disztribució hat virtuális terminált indít.
    - Az inittab fájl leírja még:
        - Mi történjen billentyű lenyomása után
        - Mi van ha elmegy/visszajön az áram

Szolgáltatások (démonok) kezelése

Amint láthattuk, a Debian Linux rendszerben a szolgáltatások (démonok) vezérlő scriptjei az **/etc/init.d** könyvtár alatt találhatók és az egyes runlevelekhez tartozó **rcX** alkönyvtárakban pedig ide mutató szimbolikus linkek vannak elhelyezve. Az **init.d** könyvtárban természetesen nemcsak olyan scriptek lehetnek, melyek automatikusan indulnak és nemcsak démonokat vezérlő scriptek lehetnek. Ezek a vezérlő scriptek adott paraméterekkel hívhatók meg, melyekkel a szolgáltatásokat elindíthatjuk/leállíthatjuk/újraindíthatjuk. Az ezekhez tartozó bemeneti paramétereket az itt található, démonokat vezérlő scriptek mindegyike "megérti". Így például ha a **mysql** adatbázisszerver telepítve van a gépen, akkor azt a következő paranccsal indíthatjuk el (ha nem indult automatikusan, mivel nem helyeztük el a megfelelő linket az **/etc/rc2.d/** könyvtárban):

/etc/init.d/mysql start

A leállítás és az újraindítás hasonlóan, a **stop** és a **restart** paraméterekkel történik.

Ezen és további szolgáltatások (démonok) vezérléséhez természetesen rendszergazdai jogosultság szükséges.