Tartalomjegyzék

0.0.1. A `telnet` parancs

A `telnet` paranccsal jelentkezhetünk be interaktívan egy távoli gépre. Például a

`$` *telnet rs1*

belép az `rs1` nevű gépbe, míg a

`$` *telnet 193.6.30.1*

belép a `193.6.30.1` Internet számú gépre. Ilyenkor pl. a következő jelenik meg a képernyőn:

```
Trying . . .
Connected to rs1
Escape character is '^T'.

AIX telnet (rs1)
IBM AIX Operating System
Version 3.2 (C) Copyright IBM Corp. 1992
(/dev/pts0)
login:
```

A parancs kiadása után a helyi gépünk megpróbálja a távoli gépet elérni, és sikeres kapcsolatfelvétel esetén megjelenik a távoli gép promptja, mintha terminálról jelentkeznénk be.

A `telnet` parancs képes arra, hogy a távoli géppel a kapcsolat kiépítésekor megállapodjon a használt terminál típusában (bizonyos esetekben azonban ezt kézzel kell beállítani a rendszer `/etc/profile` vagy a felhasználó `$HOME/.profile` állományaiban). Amennyiben a terminháltípust sikerült megállapítania a parancsnak, a távoli gépen automatikusan beállítja erre a `TERM` változó értékét. A beállítandó terminháltípust felülbírálhatjuk a `telnet` parancs `-e` kapcsolójával. Ismeretlen terminháltípus esetén a `vt100` típus beállítása az esetek nagy részében működik.

0.0.2. Az `ftp` utasítás

Az `ftp` paranccsal állományokat vihetünk át a helyi gép és egy távoli gép között. A parancs ezt az TCP/IP File Transfer Protocol (FTP) használatával valósítja meg. A parancs formátuma a következő:

`$` *ftp* GépNév

ahol a *GépNév* a távoli gép Internet neve vagy címe. A kapcsolat kiépülése után az `ftp` megkérdezi a távoli gépen használandó azonosítót és kulcsszót.

Az `ftp>` prompt után az `ftp` programnak különböző utasításokat adhatunk, amikkel pl. könyvtárakat listázhatunk ki, állományokat vihetünk át az egyik gépről a másikra, stb.

A parancs ugyan nagymértékben tudja követni a felhasználói igényeket, de az FTP szabvány megkötései miatt nem tud bizonyos file paramétereket átvinni (pl. tulajdonos neve, védelem, dátum, stb.). Ha ilyen típusú átvitelt szeretnénk megvalósítani (pl. alkönyvtárak kijelölése, stb.), akkor a `tar` archiváló parancsot (l. 12. oldal) célszerű használnunk. Futtatható, archív és más bináris állományok átvitelénél az FTP `binary` üzemmódját kell használni.

Az `ftp` kapcsolatot automatikusan építi ki a program a megadott gépre, ha a felhasználó `$HOME/.netrc` állományában a *GépNév*hez beírjuk az azon a gépen használandó felhasználói azonosító adatokat. Például ha a `.netrc` állományban a következő bejegyzés van:

```
machine rs1 login tas password huha
machine rs2 login kond
```

akkor az `ftp rs1` parancs automatikusan belép a `tas` felhasználói azonosítóra `huha` jelszóval. Az `ftp rs2` parancs elvégzi a belépést, de megkérdezi a `kond` felhasználó jelszavát, tekintve, hogy ez nincs benne a `.netrc`-ben. Ha el akarjuk kerülni, hogy az `ftp` használja a `.netrc` adatait, akkor `ftp -n` *`GépNév`* formátumot használjuk.

A `$HOME/.netrc` állomány fontos személyes információkat tartalmaz, éppen ezért a UNIX rendszer nem használja ezt, amennyiben a file elérhető a csoport vagy a külvilág számára (megfelelő biztosíték pl. a 600-as védelmi kulcs beállítása a

`$` *chmod 600 $HOME/.netrc*

utasítással).

Egy interaktívan futó `ftp` használatot a `quit` vagy `bye` utasításokkal, vagy az *End of File* (`Ctrl-D`) billentyűvel szakíthatunk meg az `ftp>` promptnál. Egy file átvitelt az *Interrupt* billentyűvel (`Ctrl-C`) szakíthatunk félbe.

Az `ftp` utasításai

Az `ftp` értelmezi az `ftp>` prompt után megadott parancsokat. Ha a parancsok használata során állománynévként a kötőjelet (`-`) adjuk meg, akkor a standard inputot (általában a billentyűzetet) használja a program bemenetként, a standard outputot (általában a képernyőt) pedig kimenetként.

A leggyakrabban használt `ftp` utasítások a következők (a gyakorlott felhasználók a felsoroltakon kívül több más parancsot is használhatnak, pl. makrózáshoz):

`user` *`Felhasználó`*
: a helyi felhasználót a megadott *Felhasználó* paraméterekkel azonosítja a távoli gépen.

`cd` *`TávoliKönyvtár`*
: belép a megadott *TávoliKönyvtár*ba a távoli gépen.

`lcd` *`HelyiKönyvtár`*
: belép a megadott *HelyiKönyvtár*ba a helyi gépen.

`pwd`
: kiírja az aktuális könyvtárat a távoli gépen.

`dir` *`[TávoliKönyvtár]`*
: Kiírja az aktuális könyvtár (vagy a megadott *TávoliKönyvtár*) állománylistáját.

`ascii`
: átkapcsolja az átviteli módot ASCII formátumba (alapértelmezés).

`binary`
: átkapcsolja az átviteli módot bináris formátumba.

`get` *`TávoliFile [HelyiFile]`*
: átmásolja a *TávoliFile*-t a helyi gépre. Ha megadjuk a *HelyiFile* paramétert, az új állomány neve ez lesz.

`put` *`HelyiFile [TávoliFile]`*
: átmásolja a *HelyiFile*-t a távoli hostra (az esetlegesen megadott *TávoliFile* néven).

`prompt`
: bekapcsolja/kikapcsolja az interaktív promptot. Kikapcsolása általában az `mput/mget` parancsoknál szokásos.

`mdelete` *`TávoliFile`*
: kifejti a *TávoliFile* kifejezést, és kitörli a távoli gépen ezeket az állományokat.

`mget` *`TávoliFile`*
: kifejti a *TávoliFile* kifejezést, és átmásolja a távoli gépről ezeket az állományokat a helyi gépre.

`mput` *`[HelyiFile]`*
: kifejti a *HelyiFile* kifejezést a helyi gépen, és a megadott állományokat átmásolja a távoli gépre.

`!`*`Parancs [Paraméter]`*
: meghívja a shellt a helyi gépen az esetleges *Parancs*csal és *Paraméter*ekkel.

`quit`
: lezárja az `ftp` bejelentkezést és kilép az `ftp` programból.

Az `ftp`-t például a következőképpen használhatjuk:

```
$ ftp rs1                          rácsatlakozunk az rs1 gépre
Connected to rs1.
220 rs1 FTP server (Version 4.1 Fri Aug 28 11:37:57 GDT 1987) ready.
ftp> user kond                     felhasználói nevünk pl. kond
331 Password required for kond.
Password:                  megadjuk jelszavunkat, de ez nem jelenik meg
230 User kond logged in.
ftp> dir                   kérünk egy file listát a távoli (rs2) gépen
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 60
drwx------   2 kond    boss        512 Oct 19 23:45 Mail
-rwxr-xr-x   1 kond    boss       1564 Oct 11 10:27 drawzdata
-rw-r-----   1 kond    boss       9546 Sep 30 16:37 dwhetc.c
-rw-r--r--   1 kond    boss       3051 Dec 06 03:44 smit.log
drwxr-xr-x   2 kond    boss        512 Dec 10 18:03 tmp
-rw-r-----   1 kond    boss       9820 Oct 11 10:28 z05001.gpl

226 Transfer complete.
827 bytes received in 0.19 seconds (4.2 Kbytes/s)
ftp> get dwhet.c           áthozzuk a dwhet.c állományt, ASCII (default)
                           üzemmódban a távoli gépről a helyi gépre
200 PORT command successful.
150 Opening ASCII mode data connection for dwhetc.c (9546 bytes).
226 Transfer complete.
local: dwhetc.c remote: dwhetc.c
9903 bytes received in 0.031 seconds (3.1e+02 Kbytes/s)
                           a két méret a sorvégi konverzió miatt nem egyezik
ftp> !ls                   a helyi gépen megnézzük, megérkezett-e
```

```
config.tel      dwhetc.c        gro             tmp
ftp             plsr            wave
ftp> put config.tel          átvisszuk a távoli gépre a config.tel állományt
200 PORT command successful.
150 Opening ASCII mode data connection for config.tel. (6122 bytes)
226 Transfer complete.
local: config.tel remote: config.tel
6043 bytes sent in 0.039 seconds (1.5e+02 Kbytes/s)
ftp> binary  átkapcsolunk bináris üzembe
200 Type set to I, binary transfer mode
ftp> mget z*                 áthozzuk a távoli gépről az összes z-vel kezdődő file-
                             t.
mget z05001.gpl? y           az ftp egyesével végigkérdezi, hogy át akarjuk-e
                             hozni az adott állományt
200 PORT command successful.
150 Opening BINARY mode data connection for z05001.gpl (9820 bytes).
226 Transfer complete.
local: z05001.gpl remote: z05001.gpl
9820 bytes received in 0.036 seconds (272 Kbytes/s)
ftp> quit                                        kilépünk a programból
221 Goodbye.
```

0.0.3. A `finger` parancs

A `finger` paranccsal egy adott gépen található felhasználó(k)ról kaphatunk információt. A paranccsal egy adott gépen (alapesetben a helyi gépen) létező felhasználóról, vagy egy adott gépet éppen használó felhasználókról kaphatunk információt.

Példák:

Az `rs2` gépen interaktívan bejelentkezett felhasználókat a

```
$ finger @@rs2   Enter
```

parancs listázza ki:

```
kond    console Mar 15  13:19
tas     pts0    Mar 15  13:01
huba    tty0    Mar 15  13:01
```

A `kond` felhasználóról az `rs2` gépen a

`$` *finger kond@@rs2* `Enter`
parancs a következőt írja ki:

```
Login name: kond
Directory: /u/kond Shell: /bin/ksh
On since May 8 07:13:49 on console
No Plan.
```

A `finger` program ilyenkor kiírja a felhasználó `HOME` könyvtárát, a login shellt, és a `$HOME/.plan`, valamint a `$HOME/.project` file tartalmát is.

Bizonyos gépeknél a fenti információk módosulhatnak, megjelenhet pl. a felhasználó telefonszáma, van-e új levele, stb.

1. fejezet

Archiválás és archív állományok használata

A UNIX használata során előbb-utóbb szükségünk lesz nem csak egy file, hanem egész könyvtárak ill. könyvtárrendszerek archiválására (elmentésére). Erre — több más, hasonló parancs mellett (pl. `cpio`, `backup/restore`) — a `tar` parancs biztosít kényelmes lehetőséget. Gyakran az állományok — különösen az archív állományok — sok, ismétlődő részt tartalmaznak, s ezért jól tömöríthetők: erre a `pack` vagy a `compress` utasítást használva kb. felére csökkenthető az állományok mérete. Ezek az utasítások minden UNIX rendszerben ugyanúgy működnek, ezért jól használhatók különböző típusú rendszerek közötti adatcserére is: pl. az egyik gépen létrehozott `tar` állományt egy másik típusú gép is szét tud csomagolni.

A programok mellett több, elsősorban DOS-os környezetből származó public domain archíváló program is megjelent a UNIX világban (pl. `arc`, `zoo`, `zip` stb.). Ezek használata azonban csak a DOS felé történő adatátvitelnél és csak előzetes kipróbálás után ajánlott: az ilyen programokkal szinte mindig verzióbeli (inkompatibilitási) problémák lépnek fel, gyakran a DOS-ban írt tömörített állományt a UNIX nem tudja olvasni és viszont. Jó megoldás lehet viszont a `tar` és a `compress` IBM PC-n való használata.

A különböző bináris állományokat (amik lehetnek pl. archív állományok vagy fordítóprogrammal és linkerrel létrehozott programok) elektronikus levélben csak ASCII formátumba való átalakítás után tudjuk átvinni. A két leggyakoribb ilyen kódolási eljárást a tradicionális `uuencode/uudecode`, illetve a hatékonyabb `atob/btoa` parancspárral használhatjuk.

A nagy állományokat az E-levélben történő továbbításhoz célszerű feldarabolni. Erre a `split` parancsot használjuk.

1.1. Könyvtárak archiválása

1.1.1. A `tar` parancs

A `tar` parancs könyvtárak ill. könyvtárrendszerek archiválására szolgál. Az utasítás archív állományként nemcsak a filerendszer egyik állományát tudja kezelni, hanem közvetlenül is írhatja/olvashatja pl. a külső szalagos eszközt (tape).

A `tar` parancs általános alakja a következő:

`$` *tar* [Opciók] -f ArchívÁllomány Állomány(ok)

A `tar` parancsban az *Állomány(ok)* nevének mérete nem haladhatja meg a 100 karaktert.

Leggyakoribb opcióként a következő kapcsolók használatosak[1]:

`-c` Egy új *ArchívÁllomány*t hoz létre a `-f` kapcsolóval megadott néven, majd beleírja a megadott *Állomány*(ok)at.

`-r` A megadott *ArchívÁllomány* végéhez írja az *Állomány*(ok)at. Ez az opció a szalagos egységnél nem használható.

`-u` A megadott *ArchívÁllomány* végéhez írja az *Állomány*(ok)at, amennyiben azok nem szerepelnek az *ArchívÁllomány*ban, vagy az archiválás óta megváltoztak. Az opció a szalagos egységnél nem használható.

`-t` Kilistázza az *ArchívÁllomány* tartalmát.

`-v` Kiírja az egyes állományok nevét, ahogy azokra sor kerül a feldolgozás során.

`-x` Kiolvassa az *ArchívÁllomány*ból a megadott állományokat. Ha az *Állomány*(ok)at nem adjuk meg, akkor az összes állományt kiolvassa az archív állományból. Ha a megadott állomány ugyanazon a néven többször is szerepel, akkor csak a legutolsó változatot olvassa el (l. a `-u` és `-r` kapcsolókat). Vigyázat! A helyettesítő karakterek nem a megszokott módon működnek (a 17. oldalon egy jól működő változat látható).

`-f` *`ArchívÁllomány`*
Megadja a használandó *ArchívÁllomány* nevét. Amennyiben ennek a - (mínusz jel) értéket adjuk, a `tar` a standard inputról olvas, ill. a standard outputra ír.

[1] Megjegyzendő, hogy a kapcsolókat egyben szoktuk megadni, azaz `-xvf` és nem `-x -v -f`. Erra azért van szükség, mivel bizonyos UNIX rendszerek másképpen nem tudják helyesen értelmezni.

`-b` *`Blokkméret`*
: A rekordok mértetét adja meg 512 byte méretű blokkokban. A *Blokkméret* alapértelmezésben 20, ami egyben a maximum is egyes UNIX rendszerekben. Ez az érték megfelel hálózaton keresztüli mentésre is, mivel a rekordok ilyenkor még nem nagyon szegmentálódnak a TCP/IP átvitelben. Ha nagyobb *Blokkméret*et választunk, az adathordozó több adatot tud tárolni. Amennyiben nem a 20-as értéket használjuk, erősen ajánlott ezt az adathordozón feltüntetni a későbbi frusztrációk elkerülésére.

`-B` Hálózati mentés vagy visszaállítás során használni kell ezt a kapcsolót. Hatására a `tar` addig vár, amíg a kommunikációs csatornában nem gyűlik össze egy rekordnyi adat.

`-h` Követi a szimbolikus linkeket, azaz a linkekkel megadott állományokat és könyvtárakat is elmenti. A `tar` alapértelmezésben csak a szimbolikus link információt írja be az archív állományba.

`-p` A kiolvasott állományokat az eredeti, az archív állományban tárolt védelmi kódokkal, felhasználói és csoport azonosítókkal látja el.

`-C` *`Könyvtár`* Amennyiben a felsorolt állományok előtt a `-C` *`Könyvtár`* kifejezés van, a `tar` egy `cd` utasítást hajt végre a *Könyvtár*ban. A paranccsal csak abszolút könyvtárneveket adhatunk meg, és az elmentendő állományokat is egyenként kell felsorolnunk.

`-d` A `tar` a kapcsoló hatására a különleges (üres könyvtár, hardver file, FIFO, stb.) állományokat is helyesen kezeli el. Ezt az opciót a `tar` nem minden UNIX rendszerben ismeri.

Példákat a `tar` parancs használatáról a 1.1.4 fejezetben, a 16. oldalon találhatunk.

☞ `tar` példák a 16. oldalon

1.1.2. A `dd` adatkonvertáló utasítás

A `dd` parancs adatok formátumának átalakítására használatos a
`dd` *`[Opció=Érték]`*
alakban. Méret megadásakor használhatjuk az érték után a `b` jelzést, ami 512 byte-os egységet, és a `k` jelzést, ami 1024 byte-os egységet jelöl. A leggyakrabban használt *Opció*k a következők:

`if=`*`Állománynév`*
: A bemeneti file neve. Alapértelmezésben ez a standard input.

`of=`*`Állománynév`*
: A kimeneti file neve. Alapértelmezésben ez a standard output.

`bs=`*`Méret`*
: A blokkok mérete byte-ban. Mérete alapesetben gépfüggő, általában 512 byte, de van, amikor 1024 byte. Felülbírálja az `ibs` és `obs` kapcsolókat.

`ibs=`*`Méret`*
: A bemeneti blokkok mérete byte-ban. Amennyiben a *Méret* után egy `b`-t írunk, akkor a `tar`-nál megismert 512 byte méretű egységeket használ (pl. `ibs=20b`).

`obs=`*`Méret`*
: A kimeneti blokkok mérete byte-ban. Amennyiben a *Méret* után egy `b`-t írunk, akkor a `tar`-nál megismert 512 byte méretű egységeket használ (pl. `obs=20b`).

`skip=`*`Szám`*
: A másolás megkezdése előtt a `dd` átugorja *Szám*mal megadott blokkot.

`conv=`*`ascii`*
: Az EBCDIC-ről ASCII-ra alakít.

`conv=`*`ebcdic`*
: Az ASCII-ról EBCDIC-re alakít.

`conv=`*`block`*
: A változó hosszúságú rekordokat fix hosszúságúvá alakítja.

`conv=`*`unblock`*
: A fix hosszúságú rekordokat változó hosszúságúvá alakítja.

`conv=`*`lcase`*
: A betűket kisbetűvé alakítja.

`conv=`*`ucase`*
: A betűket nagybetűvé alakítja.

`cbs=`*`Méret`*
: A konverziós blokkok méretét adja meg byte-ban. Csak a `conv` opcióval együtt használjuk. A *Méret*tel megadott számú karakter kerül egyszerre konverzióra. Az `ascii` és az `unblock` konverziónál a vezető szóközöket a `dd` így levágja, és egy újsor karakter kerül a sor végére. Az `ebcdic`, `ibm` és `block` átalakításnál a *Méret*tel megadott számú karakter kerül a bufferebe, majd a `dd` üres karakterekkel tölti fel a kimeneti blokkméretre a blokkot.

Például egy `gyakorlas` ASCII állományt a következő paranccsal konvertálhatunk át az `ejnye` EBCDIC állományba:

```
$ dd if=gyakorlas of=ejnye conv=ebcdic
```

Egy, a /dev/rmt0h eszközben lévő EBCDIC szalagot, amely tíz 80 byte méretű virtuális EBCDIC lyukkártyát tartalmaz rekordonként, a `ejnye` ASCII állományba a

```
$ dd if=/dev/rmt0h of=ejnye ibs=800 cbs=80 conv=ascii,lcase
```

parancs alakít.

A `dd` parancs használatáról további példákat a 1.1.4 fejezetben, a 16. oldalon találhatunk.

☞ `dd` példák a 16. oldalon

1.1.3. Mágnesszalagos egységek

A szalagos egységeket az `mt` parancson keresztül vezérelhetjük:
`mt [-f Eszköz] Parancs [Szám]`

Az `Eszköz`re a `/dev` könyvtárban található speciális állomány(ok)on keresztül hivatkozhatunk, melyek neve gépenként más és más lehet. Emellett a szalagos egység különböző üzemmódjaihoz (írássűrűség, stb.) is különböző állományok tartoznak. Az éppen használható eszközökről a rendszergazda adhat felvilágosítást: ennek hiányában SUN rendszereknél a `/dev/rst0`, IBM AIX rendszereknél a `/dev/rmt0`, míg DEC Ultrix rendszereknél a `/dev/rmt0h` eszközöket próbálhatjuk meg használni. A felsorolt eszközök írás vagy olvasás után azonnal visszatekerik a szalagot, s ezért általában létezik ún. nem visszacsévélő (no rewind) változat is: ehhez az egység neve elé egy `n` betűt kell illeszteni. Például ha a `/dev/rmt0h` eszköz helyett a `/dev/nrmt0h` eszközre hivatkozunk, akkor a szalagos egység nem fog automatikusan visszacsévélni.

Amennyiben nem adunk meg eszközt a `-f` kapcsolóval, akkor az `mt` megpróbálja a `TAPE` környezeti változóban definiált eszközt használni.

A leggyakrabban használatos *Parancs*ok a következők (a *Szám* alapértelmezésben egy):

`bsf`
: A megadott *Szám*ú file vége (EOF, End of File) karaktert lép vissza a szalagon. A szalag az EOF jel szalagkezdeti oldalára áll rá.

`bsr`
: A megadott *Szám*ú rekordot lép vissza.

`fsf`
: A megadott *Szám*ú állományt lép előre a szalagon.

`fsr`
: A megadott *Szám*ú rekordot lép előre.

`rewind`
: Visszacsévéli a szalagot.

`retension`
: Teljesen előretekeri a szalagot, majd visszacsévéli. Hatására a szalag feszültségmentes lesz.

`status`
: Kijelzi a szalagos egység állapotát.

`eof`
: Az adott szalagpozícióra állományvég (EOF, End of File) karaktert ír.

`erase`
: Letörli a szalagot.

A következő példákban a `/dev/nrmt0h` mágnesszalagos eszközt kezeljük:

✎ Kérdezzük le a szalagos egység állapotát:

```
$ mt -f /dev/nrmt0h status
```

✎ A következő parancs visszatekeri a szalagot:

```
$ mt -f /dev/nrmt0h rewind
```

✎ Három file-t a következő paranccsal léphetünk vissza:

```
$ mt -f /dev/nrmt0h bsf 3
```

✎ Két EOF karatert a

```
$ mt -f /dev/nrmt0h eof 2
```

utasítás ír a szalagra.

1.1.4. Példák archiválásra

A példákban a `/dev/nrmt0h` mágnesszalagos eszközt használjuk.

Az aktuális könyvtárról és annak alkönyvtárairól a következő utasítással készíthetünk `tar` mentést:

✎

```
$ tar -cvf /dev/rmt0
```

Az `allomany1` és `allomany2` állományokat a következő utasítással írhatjuk szalagra:

```
$ tar -cvf /dev/rmt0 allomany1 allomany2
```

Az összes állományt a szalagról a

```
$ tar -xvf /dev/rmt0
```

parancs állítja vissza. A visszaállítás a szalagon megadott névre történik, azaz pl. a `/tmp/test.1` nevű állományt a `/tmp/test.1` állományba, míg a `tmp/test.1` nevű állományt az aktuális könyvtár alatti `./tmp/test.1` állományba írja.

A `/usr/local/doc` könyvtárat a

```
$ tar -cvf /dev/rmt0 -C /usr/local doc
```

utasításssal menthetjük el. Ennek hatására a szalagon keletkező állományok nevei a `/usr/local/doc/` helyett csak a `doc/` részt fogják tartalmazni.

Az összes állományt a szalagon a

```
$ tar -tvf /dev/rmt0
```

parancs listázza ki.

Az `innen` könyvtárat a következő parancs másolja át az `oda` könyvtárba az alkönyvtárakkal együtt

```
$ cd innen; tar -cvf - . || (cd oda; tar -xBpf -)
```

A `compress` utasítással (l. a 1.2.1 fejezetet a 18. oldalon) tömörített `test.tar.Z` állományt a

```
$ zcat test.tar.Z || tar -tvf -
```

vagy az ezzel egyenértékű

```
$ cat test.tar.Z || uncompress | tar -tvf -
```

parancs listázza ki.

Mivel a kifejezéseket a `tar` nem a megszokott formában kezeli, ezért pl. az összes, a `gon` mintát tartalmazó nevű állományt a

```
$ tar -xvf... /dev/rmt0 \ tar -tf /dev/rmt0 || grep 'gon'\
```

utasítással állíthatunk vissza. Megjegyezzük, hogy a parancs futása meglehetősen lassú, valamint elég könnyen túlléphetjük a parancssor maximális hosszát (ami általában néhány ezer karakter).

Az `allomany1` és `allomany2` állományokat a következő utasítással írhatjuk szalagra 20 blokkos méretben a `dd` parancs használatával:

```
$ tar -cvf - allomany1 allomany2 || dd of=/dev/rmt0 obs=20b
```

Tegyük fel, hogy a szalagon egymás után két archívum van rögzítve. Az elolvasáshoz először tekerjük vissza a szalagot (figyeljük meg, hogy itt már minden-

hol a vissza nem csévélő `/dev/nrmt0` állományt használjuk):

```
$ mt -f /dev/nrmt0 rewind
```

Ezután olvassuk ki az első állományt:

```
$ tar -xvf /dev/nrmt0
```

A szalag az olvasás végén nem a következő állományon áll, hanem az EOF (állomány vége) jelen. Ez lépjük át:

```
$ mt -f /dev/nrmt0 fsf 1
```

Ezután kiolvashatjuk az következő állományt:

```
$ tar -xvf /dev/nrmt0
```

A hálózaton keresztül is használhatjuk a más gépen elhelyezett szalagos egységeket: ehhez a célgépre `rsh` utasítással be kell tudnunk lépni (ennek beállítását l. a `rlogin` és `rsh` parancsokat a 30. oldalon). Például az `anglia` távoli gépre a következő utasítás készít archív állományt:

```
$ tar -b 20 -cvf - * || rsh anglia dd of=/dev/rmt0 obs=20b
```

A távoli gépen elhelyezett archív állományt a

```
$ rsh -n anglia dd if=/dev/rmt0 bs=20b || tar -xvBb 20 -f -
```

paranccsal szedhetjük szét.

1.2. Tömörítő programok

1.2.1. A `compress` eljárás

A `compress` parancs az ún. Lempel-Ziv eljárással tömöríti az állományt. Az eljárás az ismétlődő részeket rövidebb kifejezésekkel helyettesíti, s így tipikusan kb. felére csökkenti egy állomány méretét. Az eljárás dinamikusan működik, nincs szükség a teljes állomány ismeretére, ezért pipe-ban is használható. A tömörített állomány a `.Z` végződést kapja meg.

A tömörített állományokat az `uncompress` parancs segítségével állíthatjuk visszsza. Sikeres visszaállítás után az `uncompress` kitörli a tömörített file-t.

A `zcat` parancs a standard outputra írja ki az eredeti file tartalmát.

Például a `gyakorlas.o` állományt a következő paranccsal tömöríthetjük össze:

```
$ compress gyakorlas.o
```

A `gyakorlas.o` állomány visszaállítása a következő paranccsal történik:

```
$ uncompress gyakorlas.o.Z
```

Egy `tar` mentést a következő utasítással tömöríthetünk és írhatunk szalagra :

```
$ tar -cvf - allomany1 allomany2 || compress | dd of=/dev/rmt0 obs=20b
```

Az elmentett archívumot a

```
$ dd if=/dev/rmt0 ibs=20b || uncompress | tar -xvBf -
```

parancs állítja vissza.

1.2.2. A `pack` eljárás

A `pack` parancs az ún. Huffmann eljárással tömöríti az állományt. Az eljárás a karaktereket gyakoriságuk alapján rövidebb bitmintákkal helyettesíti, s így kb. kétharmadára (esetleg felére) csökkenti egy állomány méretét. Az eljáráshoz a teljes állomány ismerete szükséges, azaz nem működik dinamikusan ezért pipeban nem használható. Az eljárás lassú, és csak ritka, különleges esetekben tömörít jobban a `compress`nél. A tömörített állomány a `.z` végződést kapja meg.

A tömörített állományokat az `unpack` parancs segítségével állíthatjuk vissza. A sikeres visszaállítás után a tömörített file kitörlődik.

A `pcat` parancs a standard outputra írja ki az eredeti file tartalmát.

1.3. Kódolási eljárások

1.3.1. Az `uuencode/uudecode` parancs

Az `uuencode` parancs segítségével egy bináris állományt (pl. `tar` archívum, futtatható file, magyar ékezetes állomány) az ASCII karaktertartományba kódolhatunk át. Erre pl. levél küldésénél van szükségünk: a UNIX elektronikus leveleket továbbító programja csak 7 bitet (azaz az ASCII tartományt) viszi át (l. a 27. oldalt), s ezért bináris állományok közvetlen küldésére nem alkalmas.

A kódolási eljárás 3 byte-ot 4 ASCII karakterré (+ a kontroll információ) alakít, azaz kb. 35%-kal megnő a file mérete. Ennek ellenére lassú vonalakon érdemes egy szöveges állományt először tömöríteni (tipikusan 50%-ra), majd ezt az állományt `uuencode`-olni: a file mérete így kb. 30%-kal csökken.

A parancs általános alakja a következő:

`uuencode` *`[Állomány] TávoliÁllomány`*

Az utasítás a megadott *TávoliÁllomány* néven kódolja a bemeneti *Állomány*t, ill.

annak hiányában a standard inputot. A végeredmény a standard outputon jelenik meg.

✎ Például a `gyakorlas.tar` állományt a

```
$ uuencode gyakorlas.tar gyakorlas.tar >gyakorlas.uue
```

utasítással kódolhatjuk.

A kódolt állományból a `uudecode` parancs állítja vissza az eredeti file-t a `uuencode`ban megadott *TávoliÁllomány* néven.

✎ Az előző `gyakorlas.uue` állományból az eredetit a

```
$ uudecode gyakorlas.uue
```

utasítás bontja ki.

1.3.2. A `btoa/atob` eljárás

A `btoa` szűrőutasítás bináris állományt olvas be a standard inputon, majd ezt ASCII karakterekkel kódolva kiírja a standard outputon. A kódból a szintén szűrőként működő `atob` utasítás állítja vissza az eredeti állományt.

A `btoa` eljárás egy kompakt, 85-ös számrendszerű eljárást használ, így 4 byteból 5 byte lesz. Az eljárás gazdaságosabb a tradicionális `uuencode` módszernél, de nem minden UNIX rendszer része.

1.3.3. Állományok szétdarabolása

Néhány elektronikus hálózat (pl. a BITNET) a nagyon nagy (kb. 100 kbyte feletti) elektronikus leveleket nem képes továbbítani. Az Interneten ugyan nincs ilyen korlát, mégis érdemes a nagyon nagy (2-300 kbyte feletti) állományokat feldarabolni, ha azokat valamilyen okból levélben akarjuk elküldeni. Egy nagyobb állományt több kisebb file-ra szövegszerkesztővel is feldarabolhatunk, ettől a fáradságos (mi több, könnyen elhibázható) művelettől a `split` parancs kímél meg minket. A parancs alakja a következő:

```
split [-Szám] [Állomány [Prefix] ]
```

Az utasítás a megadott *Állomány*t (alapesetben a standard inputot) *Szám*nyi (alapesetben 1000) sorral rendelkező állományokra bontja. Az állományok neve a *Prefix*-szel (alapesetben ez `x`) kezdődik, és végződésük először az `aa`, majd az `ab` stb. értékeket veszi fel, a legutolsó a `zz` lesz. Ilymódon legfeljebb 676 állományra bonthatjuk a bemeneti *Állomány*t.

A `split` soronként képes darabolni az állományokat. Bináris állományok szétbontásához használhatjuk a hasonló `bsplit` parancsot, amely — tekintve,

hogy közprogram — nem található meg minden UNIX rendszeren.

Például a `hosszufile` állományt a következő utasítással bonthatjuk fel `rovid` kezdetű, 100 sort tartalmazó darabokra:

```
$ split -100 hosszufile rovid
```

Az utasítás létrehozza a `rovidaa`, `rovidab`, stb. állományokat. Tegyük fel, hogy nincsen más `rovid` kezdetű állomány a könyvtárban. Ekkkor a

```
$ cat rovid* > ujhosszufile
```

parancs visszaállítja a `hosszufile`-t `ujhosszufile` néven.

2. fejezet

Hálózati alapismeretek és egyéb programok

2.1. TCP/IP alapok

A TCP/IP a UNIX rendszerek által leggyakrabban használt kommunikációs protokoll, melynek segítségével a számítógépek erőforrásaikat megoszthatják egymással hálózaton keresztül.

A leggyakrabban a következő TCP/IP szolgáltatásokat használják a számítógépek között:

- File átvitel, amire a File Transfer Protocol (`ftp`) szolgál.
- Távoli bejelentkezés, amit a Network Terminal Protocol (`telnet`) tesz lehetővé.
- Elektronikus levelezés, ami lehetővé teszi üzenetek küldését más számítógépekre is.
- Network file rendszerek (NFS,RFS,AFS), amelyek lehetővé teszik egész file rendszerek elérését más gépek számára. A file rendszert a távoli gépek saját operációs rendszerükbe beillesztve használhatják.
- Távoli nyomtatás, ami lehetővé teszi más gépekhez kötött nyomtatók használatát. Segítségével egy adott felhasználói csoport relatíve kevés nyomtatóval is kiszolgálható.
- Távoli programfuttatás (RPC). Segítségével lehetővé válik programok futtatása a távoli gépen. Akkor hasznos, ha csak egy programot akarunk más gépen futtatni, vagy ha a másik géphez kötött eszközöket szeretnénk közvetlenül a gépünkről elérni (pl. backup készítéséhez).

- Name szerverek, amelyek az egyedi gépek neveit és címeit tartalmazzák egy intézményen belül. Segítségével nem szükséges minden host címét a helyi gépen tárolni, az a központi szerverről elérhető.
- Terminál szerverek, amelyek speciális célszámítógépek, ahonnan távoli gépekre jelentkezhetünk be. A terminálok ezekhez az (általában Ethernet) hálózatra kötött eszközökhöz tartoznak, így nem szükséges egy hosthoz bejelentkezni csak azért, hogy egy másik gépet `telnet`tel elérjünk.
- Network-alapú window rendszerek, mint pl. az X11 rendszer.

2.1.1. A TCP/IP protokollok általános felépítése

A TCP/IP protokollt több egymásra épülő protokoll alkotja. Általában egy TCP/IP alkalmazás 4 fő rétegből áll:

- Applikációs protokoll, mint pl. az `ftp`.
- Egy olyan protokoll, mint pl. a TCP, amely az applikáció számára különböző szolgáltatásokat biztosít.
- Ip, amelyik az alapvető szolgáltatás az ún. *datagram*mok átviteléhez.
- A fizikai hordozót kezelő felület, amelyik pl. az Ethernetet vagy a soros vonalat hajtja meg.

A TCP/IP az ún. catenet modellt használja. Ez feltételezi, hogy sok egymástól független hálózat van összekötve egymással gateway-ekkel. A csomagok ekkor több különböző hálózaton is keresztülmehetnek, mielőtt célhoz érnének. Ez a folyamat a routing, ami a felhasználó számára teljesen láthatatlan. A csomag a célgép Internet címével van ellátva, ami egyértelműen megadja a végcélt. Az Internet cím egy 32-bites szám, amit gyakran 4 decimális számmal jelölnek, mint pl. a `146.113.42.42`. A felhasználók általában nem a címet, hanem a hozzárendelt szimbolikus címet (pl. `vogon.shi.edu`) használják. Az egyes gépeken futó eljárások külön-külön is kiépíthetnek TCP/IP kapcsolatot, ezeket egy Internet címen belül az ún. portszámmal különböztetjük meg. Az Internet cím és a portszám együtt egyedi kombinációt alkot. Például:

	Internet cím		TCP portok	
1. kapcsolat	`128.6.4.194`	`128.6.4.7`	`1234`	`21`
2. kapcsolat	`128.6.4.194`	`128.6.4.7`	`1235`	`21`

A TCP/IP az ún. "összeköttetésmentes" technológián alapul, amelyben az információ ún. *datagram*ban terjed. A datagram (csomag) az egy üzenetben elküldött adatok összessége. Minden datagram a hálózaton egyedi módon terjed:

a hálózati összeköttetés függvényében a futási idők különbözők lehetnek, azaz egy később elküldött datagram megelőzheti egy korábban elküldöttet. Ugyanígy lehetséges, hogy valamilyen hiba folytán egy datagram elveszik, amit újra kell küldeni.

2.1.2. A TCP szint

A TCP/IP csomagok továbbítására két protokoll, a TCP és az IP szolgál. A TCP (Transmission Control Protocol) végzi el az üzenetek feldarabolását csomagokra az egyik oldalon, míg a másik oldalon a beérkező datagramokból összerakja az eredeti üzenetet. Ez a szint kezeli a esetlegesen elvesző csomagok újrakérését és a sorrendváltozást.

Az IP (Internet Protocol) az egyedi datagramok továbbításáért felelős. Ugyan úgy tűnhet, hogy a TCP végzi el a feladat nagy részét, de ez csak kis hálózatok esetén van így. Az Internet egy nagy hálózat, és a datagramok célhoz juttatása nagyon bonyolult lehet, amennyiben több hálózaton kell azt keresztülküldeni. Az összes kapcsolat és a különböző vonalak (esetlegesen különböző fizikai hordozókon) kezelése komplex feladat, amit az IP szintnek kell elvégeznie.

Példa: Egy `ftp` processz a következő adatfolyamot kívánja a távoli gépen futó eljárásnak átküldeni (ez lehet pl. egy 1 Mbyte-os file):

. . . . . .

A TCP ezt az adatsort kezelhető darabokra szaggatja (ez illeszkedik a hálózat és a szemközti program által kezelhető maximális méretű csomaghoz):

. . . .

A TCP minden datagram elejére egy header részt illeszt, ami tartalmazza a forrás- és célprocessz port címét és az ún. sorozatszámot. A szemközti TCP program ezek ismeretében állítja vissza az adatfolyamot, a sorozatszám alapján eldönti, hogy veszett-e el csomag, stb. A sorozatszám nem a csomagokat, hanem a csomagokban lévő oktetteket számozza. A TCP ezek mellé még egy ún. checksumot adva hozza létre a TCP headert (`T`). Az adatok tehát már így néznek ki:

T. . . . T. . . . T. . . . T. . . . T. . . . T. . . . T. . . .

A TCP az IP szintnek adja át az adatokat a cél Internet címével együtt. Az IP ebből és a helyi Internet címből (`I`) képez egy csomagot, amit egy újabb checksummal ellátva ad tovább. Az adatok ekkor már így néznek ki:

```
IT....   IT....   IT....   IT....   IT....   IT....
```

ahol az I jelenti az IP headert. A fizikai szint különböző lehet, lehet soros vonal, X25, de a leggyakoribb az Ethernet. Az Ethernet saját 6 byte-os címekkel rendelkező fizikai eszköz, amely több protokollt is képes (akár egyszerre is) használni (pl. TCP/IP, DECnet, Novell, stb.). Az Ethernet saját `E` headerjét (benne a két Ethernet címmel) és `C` trailerjét (checksummal) illeszti az előző IP csomagokhoz:

```
EIT....C   EIT....C   EIT....C   EIT....C   EIT....C
```

Természetesen a fogadó oldal ezeket a konverziókat visszafelé végzi el. Az Ethernet interface levágja az Ethernet headert és a checksumot. Amennyiben a csomag IP típusú, akkor az Ethernet interface azt az IP-nek adja tovább, ami a protokoll mezőt figyeli. Ha protokoll mező TCP típusú, akkor az IP az IP headert kitörölve átadja a TCP-nek, ami a sorozatszám alapján visszaállítja az eredeti adatfolyamot.

Az Ethernet hálózat a csomagokat egy közös (általában koax) hordozóra helyezi. Mivel ez busz rendszer, előfordulhat csomagok ütközése, amikoris a rendszer bizonyos véletlenszerűen megadott idő múlva újra megkísérli a csomag elküldését. Az időzítések miatt a rendszer nagyon érzékeny a maximális futási időre. A hasznos sávszélesség ugyan csak egy része a teljes fizikai sávszélességnek, de az egyszerűség és flexibilitás kompenzálja ezt a hátrányt.

2.1.3. Socketek és applikációk

A felhasználói applikációk a TCP szint felett futnak. Ezek a programok a TCP/IP-t mint összeköttetési felületet használják, és - a felhasználókhoz hasonlóan - nem kell tudniuk a mélyebben fekvő rétegekről (IP, Ethernet).

Az egyes programok egységes használata érdekében ún. socket számokat használnak a különböző gépek az Internet hálózaton. Ezt azt jelenti, hogy míg a hívó gép (amely pl. egy `ftp` programot akar futtatni egy távoli gépen) a saját `ftp` client programjához egy véletlen, de egyedi portszámot rendel, a szemközti gépen a 21-es portot (az `ftp` socketjét) adja meg célként. A kapcsolat így egyértelműen

meghatározott (két különböző kapcsolatnál a két Internet és a két portszámból legalább egy különböző).

Az egyes applikációk a hibamentes TCP/IP csatornán további saját protokollokat használhatnak, mint pl. az SMTP levelezés:

```
RED    220 RED.VOGON.EDU SMTP Service at 29 Jun 92 05:17:18 EDT
TOPAZ  HELO topaz.vogon.edu
RED    250 RED.VOGON.EDU - Hello, TOPAZ.VOGON.EDU
TOPAZ  mail From:<zaphod@topaz.vogon.edu>
RED    250 mail accepted
TOPAZ  RCPT To:<civita@red.vogon.edu>
RED    250 Recipient accepted
TOPAZ  DATA
RED    354 Start mail input; end with <CRLF>.<CRLF>
TOPAZ  Date: Sat, 27 Jun 92 13:26:31 EDT
TOPAZ  From: zaphod@topaz.vogon.edu
TOPAZ  To: civita@red.vogon.edu
TOPAZ  Subject: meeting
TOPAZ
TOPAZ  Talalkozo holnap 1-kor a Vilag Vegen. Egy Pangalaktikus
TOPAZ  gegepukkasztot mar rendeltem...
TOPAZ  .
RED    250 OK
TOPAZ  QUIT
RED    221 RED.VOGON.EDU Service closing transmission channel
```

Figyeljük meg, hogy a parancsok csak "normális" szöveget tartalmaznak, ami tipikus az Internet szabványokban (ezek az ún. RFC ajánlások, amik *de facto* szabványok). A parancsokat általában egy szám jelenti, ami után a szöveg csak az esetleges debuggolást segíti. A számokat a különböző Internet szabványok rögzítik, általában a 2-vel kezdődőek sikeres lefutást jeleznek, a 3 további adatot kér, a 4 és 5 pedig "ideiglenes" ill. végzetes hibát jelent.

2.1.4. Egyéb protokollok: UDP, ICMP és ARP

Bizonyos alkalmazások esetén nem szükséges a TCP szintet használni, mivel az üzenet belefér egy datagramba. Ilyen lehet pl. a név szerinti Internet cím keresés. Ilyen típusú célokra az UDP ("User Datagram Protocol") használatos. Az UDP a TCP-hez hasonlóan saját port címekkel rendelkezik, és a saját headerjével ellátott csomagot adja át az IP-nek továbbításra. Az IP a távoli gépen nem a TCP, hanem az UDP protokoll kezelő programnak továbbítja a csomagot.

Egy másik protokoll az ICMP ("Internet Control Message Protocol"), amit maga a TCP/IP software használ hiba- és egyéb saját üzenetek továbbítására. Az ICMP még egyszerűbb, mint az UDP, mivel nincsenek saját portcímei.

A harmadik említésre méltó protokoll az Ethernet által használt ARP ("Address Resolution Protocol"), amely szigorúan véve nem IP protokoll. A kapcsolat kiépítésekor feltétlenül szükség van az Internet cím alapján a távoli gép Ethernet címére (ez kell a kapcsolat kiépítéséhez). Amennyiben a helyi ARP táblában ez az információ nincs meg, akkor ez lesz az, amit az ARP elvégez: egy speciális csomagban szétküldi az Internet címet a hálózaton. Minden gép figyeli az ARP csomagokat, és ha látja, hogy őt szólították meg, válaszol. Például az `ARP 128.6.4.7` kérésre az `128.6.4.7` Internet című gép visszaküldi saját Ethernet címét (pl. `8:0:20:1:56:34`). Ez az információ a helyi gép ARP cache tárolójába kerül, elkerülendő a gyakori ARP kéréseket. Az ARP kérések ún. broadcast kérések, amiket az `ff:ff:ff:ff:ff:ff` Ethernet címre küld az ARP.

2.1.5. Repeaterek, bridge-ek és routerek

Az Ethernet hálózat fizikai mérete három ok miatt maximált: a jel csillapodása és a futási idők limitálják a maximális távolságot, a véletlenszerű csomagküldés pedig a forgalom növekedése miatt a maximálisan üzemeltethető berendezések számát limitálja. A jel csillapításán ún. repeaterek használatával segíthetünk, amelyek egyszerű jelerősítők. Nagyobb távolságoknál a futási idők túlságosan nagyok lesznek, ezért ilyenkor a csomagok ütközését tárolás/továbbítás módszerrel kell elkerülni két távoli szegmens között. Erre a célra szolgálnak az ún. bridge-ek, amelyek ezen túlmenően csak a két szegmens közötti forgalmat engedik át, s így a helyi forgalom nem zavar más szegmenseket.

Az IP szint feladata a megadott datagram eljuttatása a cél címre. Ezt a feladatot routingnak hívjuk. A routing teljes egészében az Internet címen alapul. Éppen ezért a helyes routing megkönnyítése érdekében egy speciális struktúrát, az ún. domain struktrúrát alakították ki az Interneten.

A domain struktúra az jelenti, hogy különböző Internet címtartományokhoz különböző maximális címtartomány kapcsolódik. Az A osztályú címek az első byte-ban a 1-127-es számot tartalmazzák, és $256 \times 256 \times 256$ gépre kiterjedő hálózatot tennének lehetővé. Általában a B osztályú címeket használják a nagyobb intézmények, ahol 256×256 gépet lehet így hálózatba kötni. A B osztálynak a `128.1 - 191.254`-es Internet prefix tartomány felel meg. Kisebb hálózatok esetén elegendő a 256 címmel ellátott C osztályú Internet címtartomány a `192.1.1 - 223.254.254`-es tartományban. Megjegyzendő, hogy az oktettekben a 0 és 255 értéket nem használják.

A 0 és a 255 különleges jelentést hordoz: a 0 azoknak a gépeknek a címét jelenti, amelyek nem tudják saját Internet címüket. A `255`-as értéket ún. broad-

cast célokra használják, amikoris a (helyi) hálózaton minden gép figyeli az így megcímzett üzenetet. Pl. a `128.6.4` hálózaton a `128.6.4.255` értéket kell broadcast célokra használni. Általában lehet a `255.255.255.255` értéket is használni, a gateway feladata a broadcast helyi szinten tartása. A broadcastot a netmask megadásával is helyi szinten tarthatjuk. Egy adott osztály esetén a netmask értékét egyszerűen a fix Internet címtartomány 1-esre, míg a változó címtartomány 0-ra állításával kaphatjuk meg. Pl. a `194.43.42` hálózaton `255.255.255.0` lesz a netmask.

A legalacsonyabb szint a helyi hálózat: amennyiben az IP látja, hogy a datagramot a helyi hálózat egyik gépének címezték, akkor közvetlenül kiküldi. Amennyiben a cím nem érhető el közvetlenül, akkor egy, a helyi hálózathoz kapcsolt ún. gateway gépnek küldi el a datagramot. Általában a helyi hálózatokhoz csak egy router van illesztve, de a routerek képesek - a cím ismeretében - egymás között továbbítani a datagramot.

2.2. Az `rlogin` és `rsh` utasítások

Néha a `telnet` parancs használatával átjelentkezni egy távoli gépre nem elég kényelmes: például ha több gépen is van felhasználói azonosítónk, és gyakran átlépünk az egyikről a másikra (pl. segédprogramok futtatására), akkor célszerű lehet megadni a célgépen az ún. „megbízható” gépek listáját. Ezekről a gépekről ilyenkor kulcsszó használata nélkül is átléphetünk az adott gépre.

A megbízható gépek listáját kétféleképpen lehet megadni:

- A rendszergazda felsorolhatja a `/etc/hosts.equiv` állományban azokat a gépeket, amelyekről a felhaszálók automatikusan használhatják az `rlogin` és `rsh` parancsokat. Ekkor a felhasználói azonosítónak meg kell egyeznie a két gépen.

- Saját címeket is hozzáadhatunk a listához, ha a `$HOME/.rhosts` állományba soronként beírjuk az engedélyezendő gép nevét és a távoli felhasználói azonosítót (szóközzel elválasztva). Ezt az állományt csak a felhasználó írhatja, máskülönben a rendszer biztonsági okokból nem fogja használni.

Vigyázat! Ha egy felhasználói azonosítónk kulcsszavát valaki megtudja, akkor **minden**, a fenti módszerrel megadott gépünkhöz is hozzáfér!

Megjegyezzük, hogy a NIS (Network Information System, a néhai Yellow Pages) rendszerrel kényelmesebben engedélyeztethető az ilyen típusú hozzáférés sok

(kb. >5) gép esetén. A NIS-t elsősorban SUN operációs rendszerek használják, de több BSD alapú rendszerbe is beépítették.

2.2.1. Az `rlogin` utasítás

✎ Interaktív átjelentkezésre az `rlogin` utasítás szolgál. A legegyszerűbben a
$ *rlogin rs1*
utasítással léphetünk át az `rs1` nevű gépbe. Természetesen az `rs1` gépben a kiindulási gép és felhasználó azonosítójának szerepelnie kell a fent említett módon.

Amennyiben a két gépen nem egyezik meg az azonosítónk (pl. `zaphod` az `rs1` gépen), akkor az `rs1`-en a `$HOME/.rhosts` állományt kell használnunk, valamint
✎ a kiindulási gépen az
$ *rlogin rs1 -l zaphod*
utasítást kell használni.

A parancs kiadása után a helyi gép megpróbálja a távoli gépet elérni. Sikeres kapcsolatfelvétel esetén megjelenik a távoli gép promptja, mintha terminálról jelentkeznénk be.

2.2.2. Az `rsh` utasítás

Nem mindig van szükségünk interaktív használatra, sőt pl. távoli eszközök használatánál célszerűbb egy parancs távoli futtatása. Erre a célra az `rsh` utasítás
✎ szolgál. Pl. a `df` parancsot az `rs1` nevű gépen a
$ *rsh rs1 df*
utasítással futathatjuk le. Természetesen az `rs1` gépben a kiindulási gép és felhasználói azonosítójának szerepelnie kell a bevezetőben említett módon.

Ha a két gépen nem egyezik meg az azonosítónk (pl. `zaphod` az `rs1` gépen), akkor — az `rs1`-en a `$HOME/.rhosts` állományt használva) — a kiindulási gépen
✎ az
$ *rsh rs1 -l zaphod df*
utasítást kell használni.

A parancs kiadása után a helyi gépünk megpróbálja a távoli gépet elérni, majd a távoli gépen az utasítást lefuttatni. Az utasítás különleges értéke, hogy a távoli
✎ gépen futó utasítás standard be- és kimenete a helyi gépen van. Így például a
$ *rsh rs1 cat gyakorlas > > helyiproba*
utasítás az `rs1` gépen található `gyakorlas` állományt a helyi gépen található `helyiproba` állományhoz fűzi.

Ha az rs1 gépen található gyakorlas állományt az rs1 gépen található tavoliproba állományhoz akarjuk fűzni, akkor a shell metakaraktereit macskakörmökkel le kell védenünk:

```
$ rsh rs1 cat gyakorlas ">>" tavoliproba
```

Megjegyezzük, hogy C-shell (csh) használatakor az rsh utasítás néha hibaüzenettel tér vissza.

2.3. Anonymous ftp használata

A UNIX terjedésével egyre több és több olyan programot írnak, melyeket forráskóddal együtt, díjmentesen terjesztenek. Ezeket az ún. *public domain* programokat ugyan nem lehet árulni, de bárki szabadon használhatja őket. Ilyenek többek között pl. a GNU fejlesztés, amely egy teljes, néha a „professzionális” fejlesztőrendszereknél is jobb programokat (C, C++, Fortran-C fordító, debugger, UNIX segédprogramok, grafikus programok, stb.) tartalmazó szoftverrendszer.

Az Internet ragyogó lehetőséget biztosít az ilyen típusú programterjesztésre az ún. anonymous ftp segítségével.

Példaként vegyük a gnuplot programcsomagot: ennek megszerzéséhez először meg kell tudnunk azoknak a gépeknek a nevét, amelyek szerverként tárolják ezt a programcsomagot (a programcsomagok általában compress-el tömörített tar állományok, így az archívállomány neve .tar.Z végződik).

A szervergépek nevét az ún. archie szolgáltatáson keresztül kérdezhetjük le: pl. a bécsi egyetemen egy jól használható archie-szerver működik. Eléréséhez a

```
$ telnet 131.130.1.23
```

parancs kiadása után az archie felhasználói azonosítóval léphetünk be. Ezek után már magába az archie programba kerülünk be. A teljes parancskészlet nem egységes, az aktuális gépen ezt általában a help utasítással kérdezhetjük le.

A fenti példát folytatva először válasszuk ki az első 50 találatot (ez időt takarít meg):

```
archie> set maxhits 50
```

majd keressünk rá a gnuplot szóra:

```
archie> prog gnuplot
```

A program ezek után végigkeresi az adatbázist, és kiírja azoknak a gépeknek az Internet címét, amelyek a gnuplot programcsomagot tárolják. A címek mellett a teljes elérési útvonalak is megjelennek.

Egy adott cím kiválasztása után a `quit` (esetleg az `exit`) utasítással léphetünk ki az `archie`-ból. Ezek után az `ftp` segítségével lépjünk be a kiválasztott gépre: felhasználói azonosítóként `anononymous`-t, kulcsszóként saját E-levél címünket használjuk.

Ezek után lépjünk be az `archie` által megadott könyvtárba (ez általában a `/pub` könyvtárból nyílik), és az `ftp` parancsait használva (l. 4. oldal) hozzuk át a programcsomagot. Vigyázzunk, az általában `.tar, .Z, .gz` végződésű állományok bináris átvitelt (azaz először a `binary` utasítást) igényelnek!

Az átvitel után a szokásos `quit` `ftp` utasítással zárhatjuk a kapcsolatot.

Az áthozott programcsomagot a 1. fejezetben, a 11. oldalon leírt módokon csomagolhatjuk ki. Ezek után követnünk kell a programcsomag dokumentációjában leírt utasításokat.

Megjegyezzük, hogy az ELTE az `ftp.elte.hu` gépen üzemeltet anonymous `ftp` szolgáltatást.

2.3.1. A mail levelező program

A `mail` utasítás segítségével elektronikus leveleket küldhetünk más felhasználóknak. A levél elküldése után a címzett vagy azonnal értesítést kap erről (ha éppen be van jelentkezve a rendszerbe), vagy a következő bejelentkezéskor a UNIX rendszer figyelmeztetni fogja az új levélre.

A UNIX `mail` programja és az `elm`[1] csak két különböző, az operációs rendszer alap levelezőszolgáltatását használó program, így ugyanonnan veszik az új leveleket. Ezáltal elképzelhető, hogy valaki felváltva használja őket.

A UNIX `sendmail` levéltovábbító programja — történeti okok miatt — csak 7 bitet visz át. Magyar ékezetes szövegek elküldéséhez ezért a levelet kódolnunk kell, amire pl. a `uuencode` parancsot használhatjuk, vagy szöveget a TEX (esetleg az `etex`) formátumba is átalakíthatjuk.

☞ uuencode leírása a 19. oldalon.

A UNIX rendszerek különböző `mail` programokat használnak. A következőkben egy széles körben elterjedt programot, az ún. UCB mail programot ismertetjük. Ez a program fut alapértelmezésben a `mail` meghívásakor pl. az IBM AIX és a DEC Ultrix rendszerekben. A SUNOS/Solaris rendszerben a `mail` alapesetben ugyan a `/bin/mail` nagyon egyszerű programra vonatkozik, de a `/usr/ucb/mail` parancs itt is az UCB mail programot futtatja[2].

[1] Az `elm` nem része az eredeti UNIX rendszernek, hanem u.n. közprogram.

[2] Rendszeres használat esetén célszerű az `alias mail="/usr/ucb/mail"` parancssort a `$HOME/.profile` állományba beírni. Ekkor a `mail` automatikusan az UCB mail-re vonatkozik.

Az UCB mail helyett — amennyiben lehetőség van rá — célszerűbb a menüvel ellátott `elm` levelező programot használni, melynek leírását a 35. oldalon találhatjuk meg.

☞ `elm` leírása a 35. oldalon

Levél küldése

Egy egyszerű, rövid üzenetet a `mail` program segítségével küldhetünk el. Ha levelünk hosszú, célszerű lehet először a pl. `vi` szerkesztő segítségével megírni, és utána elküldeni.

A `mail` utasítás sorában meg kell adni a címzett(ek) felhasználói nevét. Ezután a program először a rövid tartalmat kérdezi meg (`Subject:`), majd soronként fogadja az üzenetet. Az üzenetet egy, csak egy `.`-ot tartalmazó sorral zárjuk le (használhatjuk a `Ctrl-D` kombinációt is). A legvégén - beállítástól függően - a `mail` program felajánlja "másolatok" küldésének lehetőségét (`Cc:`) is más felhasználóknak. Például:

`$` *mail galahad* Enter
Subject: delutani talalkozo Enter
Delutan 3 orakor megbeszeles lesz Enter
a kerekasztalnal Enter
Sir Robin Enter
. Enter
`Cc:` Enter

Ha file formájában már rendelkezésre áll az elküldendő információ (pl. egy program futásának végeredménye, vagy egy előre megírt levél, stb.), akkor azt egyszerűen a standard input átirányításával küldhetjük el. Például:

`$` *mail galahad <fontos.level* Enter

A program a `fontos.level` file-t elküldi a `galahad` felhasználónak.

Egyszerre több felhasználónak is elküldhetjük ugyanazt a levelet, ha a `mail` parancs után egyszerűen felsoroljuk a neveiket szóközzel elválasztva.

Ha a címzett nem ugyanazon a gépen dolgozik, ahonnan mi szeretnénk küldeni a levelet, akkor nemcsak a felhasználói nevet, hanem a másik gép nevét is ismernünk kell. A címbe ilyenkor a felhasználói név és a gép neve közé egy `@@` jelet kell tennünk.

Példa:

`$` *mail sirrobin arthur@@anglia.elte.hu galahad < fontos.level* Enter

A program a `fontos.level` file-t elküldi a az `arthur` nevű felhasználónak, aki a `anglia.elte.hu` elnevezésű gépen dolgozik, valamint `sirrobin` és `galahad` felhasználónak a helyi gépen. Természetesen mindenki ugyanazt a levelet kapja

meg.

Levél fogadása

Egy új levél érkezésekor a `"YOU HAVE MAIL"` (levele érkezett) üzenetet írja ki általában az operációs rendszer a képernyőre. Ha éppen dolgozunk a gépen akkor ez a levél érkezésekor megtörténik, máskülönben pedig a következő bejelentkezésünkkor kapjuk az üzenetet. Például az új levelet a

✎ `$` *mail* Enter

utasítással olvashatjuk el. A UNIX rendszer ekkor a következőképpen válaszol:

```
Mail version 2.18 5/19/83.  Type ? for help.
"/usr/spool/mail/lancelot": 3 messages 3 unread
R  1 arthur Tue Sep 23 11:28 24/725 "Gyules"
N  2 arthur Thu Sep 26 13:32 10/315 "Hova tetted a Gralt?"
U  3 arthur Thu Sep 26 14:35 11/305 "A fecske sebessege..."
&
```

A legelső sorban, rögtön a program neve után jelzi a `mail`, hogy `?`-et gépelve segítséget kaphatunk.

A következő sorban a levelesláda file nevét írja ki a rendszer, majd ezután következik a levelek felsorolása. Minden levélre kiíródik az állapota (`R` - már olvasott, `U` - még olvasatlan, `N` - új levél), a levél sorszáma, a küldő neve, a küldés ideje, a levél hossza sorban/karakterben, és végül a levél `Subject:` sora.

Az `&` karakter után a kívánt sorszámot beütve a levél tartalma megjelenik a

✎ képernyőn. Például:

```
& 2
Message  2:
From anglia Tue Sep  3 11:28:16 1991
Date: Tue, 3 Sep 91 11:26:57 -2300
From: arthur (Gral Knight)
To: lancelot
Subject: Hova tetted a Gralt?

Lancelot,
mar napok ota keresem a Gralt. Nem lattad valahol?
                          Arthur
```

A hosszabb levelek kiírását a `Ctrl-S`-`Ctrl-Q` gombokkal szabályozhatjuk.

A `mail` parancs `&` promptjánál különböző utasításokat adhatunk meg az éppen olvasott levélre vonatkozóan.

Példák:

Az

& *s levelem.doc* Enter

paranccsal kiírhatjuk a levelet egy `levelem.doc` nevű file-ba.

A

& *d* Enter

utasítás kitörli az adott levelet a levelesládából.

Az

& *f** Enter

utasítás kilistázza az adott levelesládában található leveleket, ugyanúgy, mint a `mail` programba való belépéskor.

A programból a `q` (quit) vagy az `x` (exit) parancsokkal léphetünk ki. A `q` utasítás kilép a `mail`ből, és a levelesládát megváltoztatja az esetleges törléseknek megfelelően.

Az `x` parancs NEM változtatja meg a `mail`be való belépés előtti állapotot, azaz pl. nem törli ki a `mail`ben a `d` paranccsal törölt leveleket.

Figyelem! Az `q` és az `x` parancs pontosan "fordítva" működik, mint ahogy azt esetleg megszoktuk a VAX/VMS operációs rendszerben!

A `mail` több más utasítást is ismer, a **??**. oldalon egy részletes leírás található a programról.

☞ `mail` részletes leírása a **??**. oldalon

2.3.2. Az elm levelező program

Az `elm` elektronikus levelezőrendszert a

$ *elm*

paranccsal hívjuk be. Amennyiben az első alkalommal hívjuk be a gépen az `elm`et, feltétlenül olvassuk el a **??**. oldalon található részletes magyarázatot a program beindításáról.

☞ `elm` első elindítása a **??**. oldalon

A UNIX `mail` programja és az `elm` csak két különböző, a UNIX rendszer levelezőszolgáltatását használó program, így ugyanonnan veszik az új leveleket. Ezáltal elképzelhető, hogy valaki felváltva használja őket.

Az `elm` főmenüje egy teljes képernyőt betöltő rendszer. A képernyő tetején a program kiírja a levelesláda helyét és azt, hogy hány levél található benne. Többféle levelesláda (folder) lehet; a bejövő új leveleken, valamint az eltárolt régi leveleken kívül létrehozhatjuk magunknak még például az elküldött levelek másolatát, vagy egy adott témával kapcsolatos levelek gyűjtőjét is. Az `elm` az új posta vizsgálatával jelentkezik be:

```
       Folder is '/usr/mail/guest' with 4 messages [elm 2.3 PL2]
N       1   Aug 17 Bokor Gyorgy       (25)   targyalas
O       2   Aug 19 J.F. Kenedi        (22)   Re: a tiedet
NU      3   Sep 28 jozsi@ludens.elt   (37)   Mi ez!!!
        7   Oct 21 POSTMASTER         (76)   Undeliverable mail

 You can use any of the following commands by pressing the first character
d)elete or u)ndelete mail, m)ail a message, r)eply or or f)orward mail, q)uit
   To read a message, press <return>. j=move down, k=move up, ?=help

Command:
```

Minden levélről egy egysoros bejegyzést láthatunk (amennyiben van levelünk). A UNIX `mail`től eltérően az `elm` olyankor is elindul, ha nincs új levelünk. A bejegyzés elején a levél státusza áll (új, még nem olvasott, törölt, stb.), azt követi egy sorszám, a levél beérkezésének dátuma, a feladó, a levél hossza sorokban, és a levél tárgya.

A képernyő alján rövid segítséget ad a program a legfőbb, az adott menüben használható utasításokról. Kilépni a `q` gomb lenyomásával tudunk a levelezőprogramból. Ha volt levelünk, akkor az `elm` a kilépés során megkérdezi, hogy a törlésre kijelölt leveleket kitörölje-e, az olvasott leveleket áttegye-e az olvasott levelek gyűjtőjébe. Minden kérdésre felajánl egy alapmegoldást, amit az `Enter`-re elvégez.

Levelet úgy tudunk elolvasni, hogy rámegyünk a kurzorral (a kiválasztott levél fordított színekkel vagy a `->` jellel van megjelölve a képernyőn). Mozogni a kurzormozgató nyilakkal tudunk, de beállítástól függően a `vi` szövegszerkesztő utasításai is működnek: a `j` lefelé, a `k` felfelé mozgat. Ha a kiválasztott levélen vagyunk, az `Enter` lenyomásával kezdhetjük el azt olvasni. Az `elm` a `more` programot használja a levél tartalmának kiírására, így a `more` parancsait is használhatjuk.

Levelet a főmenüben az `m` (mail) gomb lenyomásával küldhetünk. A beépített `mail` levelezőrendszerhez hasonlóan ilyenkor az `elm` megkérdezi, kinek akarjuk küldeni a levelet a `Send the message to:` kiírásával. Ide kell beütni a címzett E-mail címét (vagy amennyiben létrehoztuk, a becenevét). Ezek után az `elm` a `Subject of the message:` kiírásával a levél tárgyát kérdezi meg (igen rövid leírása annak, hogy milyen ügyben írtuk a levelet). Amennyiben ezt a mezőt nem töltjük ki, az `elm` megkérdezi, hogy folytatni akarjuk-e a levél elküldését. Ha igen, válaszoljunk `y`-nal. Az `elm` beállításától függően ezek után vagy mindjárt írhatjuk a levelet, vagy még előbb megkérdezi a program, hogy küldjön-e másolatot valakinek a `Cc:` (Carbon Copy) kiírásával (ha nem akarunk senkinek másolatot küldeni, nyomjuk le az `Enter` gombot, különben írjuk be a címe(ke)t).

Amikor ezeken túljutottunk nagy valószínűséggel a `vi` szövegszerkesztőben találjuk magunkat. Ebben megírhatjuk a levelet, és a `vi`-ból kilépve az `elm` megkérdezi, hogy elküldje-e a levelet. Ezt az opciót ajánlja fel ([s] – send), így [Enter]-re a levelet elküldi. Ha nem akarjuk a levelet elküldeni, akkor a [f] (forget) lenyomásával állíthatjuk le a levél elküldését.

Az `elm` részletesebb leírását lásd a **??**. fejezetben, a **??**. oldalon.

☞ `elm` részletes leírása a **??**. oldalon