
Pere László

GNU/Linux rendszerek
üzemeltetése II.

Hálózatok

PÉCS, 2005

Pere László: GNU/Linux rendszerek üzemeltetése II.
Hálózatok
© 2005 Pere László

Felelős kiadó a Kiskapu Kft. ügyvezető igazgatója
© 2005 Kiskapu Kft.

1081 Budapest, Népszínház u. 31., I. em., 7.
<http://www.kiskapukiado.hu/>
e-mail: kiado@kiskapu.hu

Lektorálta: Harka Győző
Anyanyelvileg lektorálta: Rézműves László
Borítóterv: Bognár Tamás
Műszaki szerkesztő: Csutak Hoffmann Levente

ISBN szám: 963 9301 98 1

Készült a debreceni Kinizsi Nyomdában
Felelős vezető: Bördős János

Tartalomjegyzék

1. Bevezetés	11
1.1. A jelölések és elnevezések	11
1.2. A könyvben található példák	12
2. Alrendszerek	13
2.1. A naplázás	13
2.1.1. A naplóállományok	13
2.1.2. A naplázás indítása és leállítása	14
2.1.3. A rendszernapló beállítása	15
2.1.4. Bejegyzések elhelyezése a naplóban	20
2.2. Időzített feladatok	21
2.2.1. Késleltetett feladatok	22
Feladatok kiíratása	24
Feladatok eltávolítása	24
A használat korlátozása	24
2.2.2. Időszakosan végrehajtandó feladatok	25
Új feladatok létrehozása	25
Feladat törlése	27
A feladatok kiíratása	27
2.2.3. Időzítés munkaállomásokon	28
2.3. Kézikönyv	30
3. A rendszermag	37
3.1. A rendszermag újrafordítása	37
3.1.1. A fordítás előkészítése	38
3.1.2. A fordítás	41
3.1.3. A telepítés	41
3.2. Magmodulok kezelése	42
3.2.1. Betöltött magmodulok kiíratása	44
3.2.2. Magmodulok eltávolítása	44
3.2.3. Magmodulok leírása	44
3.2.4. A magmodulok betöltése	46

3.2.5. A modulok automatikus betöltése	48
4. A hálózati kapcsolat kiépítése	51
4.1. A számítógépek összekapcsolása	51
4.1.1. Bevezetés	52
Ethernet hálózatok építése	52
Az Ethernet hálózatok működése	60
Az IP protokoll	63
4.1.2. A hálózati kártya beállításai	69
Az ISA csatolófelület	71
A PCI csatolófelület	73
A PCMCIA csatolófelület	75
4.1.3. A meghajtóprogram betöltése	76
Az automatikus kártyafelderítés	78
Több hálózati kártya	80
4.1.4. A hálózati csatoló beállítása	82
4.1.5. Az útválasztó táblázat	87
4.1.6. Összefoglalás	92
4.2. A számítógépek nevei	93
4.2.1. A névfeloldás beállítása	95
4.2.2. Névfeloldás szöveges állományokból	96
4.2.3. A számítógép saját neve	97
4.2.4. A névkiszolgáló használata	98
4.3. A beállítások automatikus lekérdezése	100
4.4. A hálózat beállítóállományai	101
4.4.1. Red Hat alapú terjesztések	101
4.4.2. Debian alapú terjesztések	104
5. Hálózati szolgáltatások	107
5.1. A hálózat alapprotokolljai	107
5.1.1. Az UDP protokoll	109
5.1.2. A TCP rotokoll	111
5.1.3. Az ICMP protokoll	112
5.2. Hálózati kapuk és szolgáltatások	113
5.2.1. A szolgáltatások nyilvántartása	114
Szolgáltatás nyilvántartás nélkül (távoli eljáráshívás)	119
5.2.2. A szolgáltatások felderítése	120
5.3. Az Internet szuperkiszolgáló	122
5.3.1. Az inetd beállítása és használata	122
5.3.2. A xinetd beállítása és használata	125
5.3.3. Az Internet szuperkiszolgáló védelme	132
5.4. A névkiszolgáló üzembe helyezése és beállítása	136
5.4.1. A másodlagos névkiszolgáló	143
5.4.2. Feloldó gyorsítótáras névkiszolgáló	145

5.4.3. Biztonsági beállítások	146
5.5. A hálózati beállítások szolgáltatása	150
5.5.1. A DHCP rendszer működése	151
5.5.2. A DHCP kiszolgáló üzembe helyezése	153
5.5.3. A munkaállomások nyilvántartásba vétele	155
5.5.4. A DHCP kiszolgáló beállításai	157
5.5.5. Egyéb hálózati beállítások szolgáltatása	158
6. A hálózat lehallgatása	163
6.1. A program indítása	164
6.2. A lehallgatás	165
6.3. Az adatok elemzése	167
6.4. A kapcsolattartás módjai	169
6.4.1. A csomagküldés előkészítése	169
6.4.2. Csomag küldése	173
6.4.3. Folyamatos kapcsolat használata	178
6.5. A csomagok szűrése	184
6.5.1. Állandók és műveleti jelek	186
6.5.2. Mezőnevek	188
Bármely csomag	188
Ethernet csomagok	188
ARP csomagok	189
IP csomagok	190
UDP csomagok	191
TCP csomagok	191
SMTP csomagok	192
FTP csomagok	192
HTTP csomagok	193
A rendszernapló csomagjai	194
7. Hálózatok összekapcsolása	195
7.1. Az útválasztó bállítása	196
7.2. A tűzfalak	197
7.2.1. Egyszerű csomagszűrés	197
A szabályrendszerek lekérdezése	200
Az alapszabály megváltoztatása	202
A szabályrendszerek kiürítése	203
Új szabály elhelyezése	204
Szabály törlése	210
Összefoglalás	210
7.2.2. A csomag előtörténete	213
A modulok betöltése	213
Az előtörténetet kezelő modul	215
7.2.3. Saját szabályrendszerek készítése	218

Szabályrendszerek létrehozása	220
Szabályrendszerek törlése	220
Összefoglalás	220
7.2.4. A címfordítás	222
A táblák	222
A forráscím megváltoztatása	224
A célcím megváltoztatása	228
A naplózás	231
7.3. A biztonság ellenőrzése	232
7.3.1. A támadóprogram telepítése	233
7.3.2. A program indítása	234
7.3.3. A támadás előkészítése	235
7.3.4. Az eredmény kiértékelése	237
8. Hálózati alkalmazások	239
8.1. A hálózati állományrendszer (NFS)	239
8.1.1. Az NFS kiszolgáló	240
8.1.2. Az NFS ügyfél	241
8.2. Felhasználói adatbázis megosztása (NIS)	242
8.2.1. A NIS kiszolgáló üzembe helyezése	243
8.2.2. A NIS ügyfél	248
8.2.3. Jelszóváltoztatás a NIS rendszerben	249
8.3. Állományok szolgáltatása	250
8.3.1. Az FTP kiszolgáló indítása és leállítása	250
8.3.2. Az FTP kiszolgáló beállítása	250
8.4. Az elektronikus levelek fogadása	254
8.4.1. A levél fogadás indítása, leállítása és működése	254
8.4.2. Beállítóállományok	256
8.4.3. A szolgáltatások engedélyezése és tiltása	259
8.4.4. Idegen levelek fogadása	262
8.4.5. Látszólagos felhasználók	263
8.5. A webkiszolgáló	265
8.5.1. A webkiszolgáló indítása és leállítása	266
8.5.2. A webkiszolgáló alapbeállításai	266
8.5.3. Az Apache moduljai	269
8.5.4. Könyvtárak szolgáltatása	273
8.5.5. Biztonsági korlátozások	278
8.5.6. Programok futtatása a webkiszolgálón	283
Webkiszolgálón futtatható programok készítése	285
8.5.7. Felhasználók azonosítása	286

Táblázatok jegyzéke

2.1. A syslogd program	16
2.2. Az mkfifo program	19
2.3. A logger program	20
2.4. Az at program	22
2.5. Az atd program	23
2.6. A crond program	25
2.7. A crontab program	26
2.8. Az anacron program	29
2.9. A man program	30
2.10. Az apropos program	33
2.11. A makewhatis program	34
2.12. Kézikönyvoldal-formázóparancsok	35
3.1. Az lsmod program	43
3.2. Az rmmod program	45
3.3. A depmod program	46
3.4. A modprobe program	48
4.1. Az RJ-45 csatlakozó bekötésének színkódolása	60
4.2. Az IP címek osztályai	66
4.3. Magáncélra használható IP címtartományok	67
4.4. Az lspci program	74
4.5. Az ifconfig program	84
4.6. A route program	88
4.7. A hostname program	98
5.1. Az inetd program	123
5.2. A xinetd program	125
5.3. A named program	137
5.4. A dhcpcd program	153
6.1. A ping program	169

6.2. A traceroute program	176
6.3. Az ethereal szűrőkifejezéseinak összehasonlító műveletei	187
6.4. Az ethereal szűrőkifejezéseinak részláncműveletei	187
7.1. A küldhető ICMP csomagtípusok	205
7.2. A legfontosabb ICMP csomagtípusok	207
7.3. A TCP kapcsolók elnevezése	209
8.1. Az ypserv program	248
8.2. A httpd program	266
8.3. A htpasswd program	288

Ábrák jegyzéke

3.1. A Linux rendszermag telepítés előtti beállítása	39
4.1. Vékony Ethernet ellenállások és csatlakozók	53
4.2. Szerelt vékony Ethernet csatlakozás	54
4.3. Az RJ-45 csatlakozó vezetékeinek számozása (aljzat felől)	56
4.4. RJ-45 csatlakozóval szerelt kábel	57
4.5. A csavart érpáras Ethernet hálózat kábele	58
4.6. A keresztkábel bekötése	59
4.7. Az Ethernet cím jelzése	61
4.8. ISA hálózati csatoló	72
4.9. PCI hálózati csatoló	73
4.10. Vezeték nélküli PCMCIA hálózati csatolók	76
4.11. A hálózat beállítása Fedora Core 2 rendszeren	103
5.1. A BIND beállításai grafikus felületen	150
6.1. Az ethereal főablaka	165
6.2. A lehallgatás kezdete	166
6.3. A lehallgatás állása	167
6.4. A lehallgatott adatok	168
6.5. A szűrők használata a lehallgatott adatok elemzése közben	186
7.1. Az egyszerű csomagszűrés szabálylistái	199
7.2. A nessus indulóképernyője	233
7.3. A nessus támadóprogramjai	234
7.4. A nessus támadás tulajdonságai	235
7.5. A nessus támadás kiszemelt célpontjai	236
7.6. A nessus támadás közben	237
7.7. A nessus támadás eredménye	237
8.1. Az Apache által létrehozott állománylista	276
8.2. Egyszerű BASH CGI program eredménye	285
8.3. A webböngésző jelszókérése	287

1. fejezet

Bevezetés

E könyv a GNU/Linux rendszerek üzemeltetéséről szóló sorozat második kötete, amely a számítógép-hálózatok üzemeltetését mutatja be. A kötet az első részben[82] tárgyalt ismereteket felhasználva mutatja be a GNU/Linux hálózatos alkalmazásait.

Ez úton is szeretnék köszönetet mondani mindenöknek a kollégáknak és olvasóknak, akik az eddigi könyveim hiányosságaira felhívták a figyelmet, tanácsaikkal, kérdéseikkel segítették a munkámat. Köszönet illeti a Pécsi Tudományegyetem érdeklődő hallgatóit, akik az ehelyütt tárgyalt ismeretek tanítása során kérdéseikkel hasznosabbá, érthetőbbé tették mondanivalóját. Szeretnék külön is köszönetet mondani a következő kollégáknak: DR. HEGYI SÁNDOR, DR. KONIORCZYK MÁTYÁS, IMREK GYULA, KISS GÁBOR, RÉBAY VIKTOR.

A könyv megírása során Különösen sok segítséget jelentett HARKA GYŐZŐ (CARLOS) szakmai-, illetve CSIZMAZIA LÁSZLÓ nyelvi lektori tevékenysége.

1.1. A jelölések és elnevezések

A szövegben található könyvtárbejegyzések neveit a szövegtől elütő betűtípussal emeltük ki (például /etc/fstab). A könyvtár típusú könyvtárbejegyzések neve / jellel végződik (például /home/). Reményeink szerint ez megkönnyíti a szöveg olvasását.

A programok és parancsok a szövegben színtén ki annak emelve (például ls), épp úgy, ahogyan a programoknak adott paraméterek és kapcsolók (például -lahd). Hasonlóképpen kiemeltük a különféle beállítóállományokban elhelyezhető kulcsszavakat (például auto).

A könyvben a GNU/Linux és a Linux elnevezések nem véletlenszerűen követik egymást. A GNU/Linux elnevezés a teljes telepített rendszert, a rendszer- és felhasználói programok összességét takarja. Ide tartoznak a programok, alkalmazások, a különféle beállítóállományok, egy szóval a teljes programrendszer. A Li-

nux elnevezés a rendszermagot jelenti. A rendszermag egyetlen program, amely a számítógép hardverelemeinek közvetlen vezérlését végzi és amely a programrendszer alapjául szolgál.

Néha nem lényeges ugyan a GNU/Linux és Linux elnevezések közti különbség, nem fontos, hogy a rendszermagról vagy a teljes rendszerről beszélünk-e, a legtöbb esetben azonban hasznos lehet tudni, mik azok a feladatok, amelyeket a rendszermag végez, és mi hárul az egyéb elemekre. Kínosan ügyeltünk tehát arra, mikor melyik elnevezést használjuk, a szöveg azonban a legtöbb esetben megértható anélkül, hogy erre a különbségre egyáltalán felfigyelnénk.

Hasonlóképpen szigorúan ügyelünk a parancsok és a programok megkülönböztetésére. A héjnak parancsot adó felhasználónak nem feltétlenül kell tudnia, hogy az általa begépelt parancsot a héj értelmezi és hajtja végre, vagy egy külső program elindulását jelenti, de a rendszert üzemeltető rendszergazdának illik tudnia, hogy belső parancsról vagy külső programról van-e szó.

Ha tehát egy kulcsszó értelmezését és végrehajtását a program végzi, amelynek begépeltük, parancsról, ha viszont a kulcsszó egy állomány neve, amelynek programként való indítását végzi az alkalmazás, programról beszélünk.

A könyv számítógép-hálózatokról szóló részében meglehetősen sokszor esik szó szabványokról, ajánlásokról. Az Internet felépítését szabályozó dokumentumokat mindenütt szabványként emlegetjük, nem ragaszkodunk a dokumentumok közti finom különbségek részletezéséhez, ezen dokumentumok életciklusának bemutatásához. Az érdeklődő olvasó az irodalomjegyzékben felsorolt dokumentumokat az Interneten megkeresve tájékozódhat ezekben a kérdésekben is. A hivatkozott RFC dokumentumok az Internet számos pontjáról letölthetők.

1.2. A könyvben található példák

A könyvben található néhány példaprogram és példaként felhasználható beállítóállomány. Ezek letölthetők a kiadó internetes oldalairól a <http://www.kiskapukiado.hu/106> cím felhasználásával.

A példákkal kapcsolatban fel kell hívnunk a figyelmet az óvatosságra! A programokat és beállítóállományokat ugyan többféle GNU/Linux gyűjteményen kipróbtáltuk, ez azonban nem jelenti azt, hogy azok garantáltan működnek minden számítógépen. A programok és beállítóállományok szabadon felhasználhatók és módosíthatók, valamint továbbadhatók, a használatukra azonban nem érvényes semmiféle garancia.

2. fejezet

Alrendszerök

E fejezetben olyan programrendszerkről olvashatunk, amelyek működése nem elengedhetetlenül fontos feltétele a számítógép-hálózat beállításának és használatának. A GNU/Linux rendszerek ezen összetevői azonban nagy segítséget nyújtanak a hálózat telepítése, karbantartása közben, így mindenki által érdemes néhány szót ejtenünk róluk.

2.1. A naplázás

A rendszernapló (*system log*) a UNIX rendszerek szolgáltatása, amelyet a különféle programok vehetnek igénybe. A rendszernapló a számítógép által kezelt nyilvántartás, amely az egyes eseményeket és a bekövetkezésük időpontját rögzíti. Az üzenetek rögzítését legtöbbször a háttérben futó programok kérik, hiszen ezen programok az üzeneteiket nem írhatják a képernyőre.

A rendszer felügyelete szempontjából igen fontos a naplázás, hiszen lehetővé teszi a rendszergazda számára, hogy olyan történésekről is értesüljön, amelyek bekövetkeztek kor nem volt jelen.

2.1.1. A naplóállományok

A GNU/Linux rendszereken általában a `/var/log/` alkönyvtárban találhatók a naplóállományok (*log files*).



Red Hat rendszereken a `/var/log/` alkönyvtárban megtalálható a `messages` az általános üzenetek számára, a `maillog` az elektronikus levelek naplózására, a `secure` a rendszer biztonsága szempontjából fontos üzenetek számára. Ezeket az állományneveket a `/etc/syslog.conf` beállítóállományban adhatjuk meg.



A Debian alapú rendszereken szintén a `/var/log/` alkönyvtárban találhatjuk meg a naplóállományokat. A Debian rendszerek naplózásának beállítására is a `/etc/syslog.conf` állományban van módunk.

A naplóállományokban szabályos formátumú üzeneteket találhatunk, soronként egy bejegyzést:

```
1 Jan 24 20:37:01 localhost automount[1001]: expired /auto/floppy
2 Jan 24 20:39:55 localhost -- pipas[925]: LOGIN ON tty4 BY pipas
```

Amint látjuk, a bejegyzések formája kötött. A naplóbejegyzések a következő részekből épülnek fel:

1. A sorok elején olvasható a naplóbejegyzés születésének dátuma és pontos ideje.
2. A második részben látható a számítógép neve, amelyen az üzenet született. A rendszernapló képes távoli gépek üzeneteinek naplózására is, ezért fontos rögzíteni, hogy az adott naplóbejegyzés melyik számítógépről származik.
3. A következő oszlopban általában annak a programnak a neve található, amely az üzenetet küldte, utána pedig szörgyűjtők között a küldő folyamat folyamatazonosítóját (PID) olvashatjuk.
4. A naplóbejegyzés utolsó része a szöveges üzenet, amelyet az alkalmazás a naplóban elhelyezett.

Láthatjuk, hogy a naplóállományok formátuma kötött, szabályos, így igen könnyen tudjuk kezelni a jól ismert szűrőkkel és programokkal. Erre valószínűleg szükségünk is lesz, mivel egy jól működő GNU/Linux rendszer oly sok naplóbejegyzést „termel”, hogy abban eligazodni sokszor csak gépi segítséggel lehet.

2.1.2. A naplózás indítása és leállítása

A naplózást a `syslogd` nevű program végzi, a háttérben futva. A `syslogd` szolgáltatás létrehozásával a szokásos módon indítható és állítható le. A következő példa bemutatja, hogyan indíthatjuk el és állíthatjuk le a rendszernaplót kezelő programot.

1. példa. Állítsuk le a rendszernaplót a szolgáltatást indító és leállító héjprogram `stop` paraméterével! Ez a legtöbb GNU/Linux terjesztés esetében a következőképpen végezhető el:

```
$ /etc/init.d/syslog stop
A kernelnaplózó leállítása: [ OK ]
A rendszernaplózó leállítása: [ OK ]
$
```

Ha újra el akarjuk indítani a rendszernaplót, ugyanazt a héjprogramot kell indítanunk, de a start paraméterrel:

```
$ /etc/init.d/syslog start
A rendszernaplózó indítása: [ OK ]
A kernelnaplózó indítása: [ OK ]
$
```

Amint látjuk, a rendszernaplót a szolgáltatásoknál megszokott módon lehet indítani és leállítani.

Fontos tudnunk, hogy a syslogd képes a hálózaton keresztül fogadni az üzeneteket más számítógépekről. Ez igen hasznos szolgáltatás, hiszen így az üzeneteket biztos helyre menthetjük, és egy központi gépen egységesen kezelhetjük – de a központi naplózásnak veszélyei is vannak.



Ha egy behatoló sikeresen bejut egy számítógépre, akkor valószínűleg az lesz az első lépése, hogy megsemmisíti a naplóállományokat, hogy az eseményeket a későbbiekben ne lehessen rekonstruálni, a betörő személyére ne lehessen fényt deríteni.

A naplóállományok törlése ellen védekezhetünk oly módon, hogy a naplóbejegyzéseket egy távoli számítógépre küldjük és ott rögzítjük, de egy pillanatig sem reménykedhetünk abban, hogy a behatolónak nem áll szándékában a távoli számítógépet is megtámadni, ha látja, hogy a naplóbejegyzések kicsúsztak a kezei közül. Ráadásul minden szolgáltatás potenciális támadási felület is egyben, lehet, hogy éppen a hálózaton keresztül történő naplózást használja ki a behatoló a számítógép megtámadására.

Ha tehát a syslogd távoli gépekről, hálózaton keresztül is fogad naplóbejegyzéseket, akkor ezen a szolgáltatásán keresztül a számítógép támadható is¹. Ez az oka annak, hogy a syslogd program alapesetben nem fogad hálózaton keresztül naplóbejegyzéseket, csak akkor, ha induláskor a -r kapcsolót kapja. Ha nem tudjuk biztosan, hogy a rendszeren milyen a beállítás, akkor érdemes a syslogd indítását végző héjprogramot megvizsgálnunk.

2.1.3. A rendszernapló beállítása

A syslogd működését a /etc/syslog.conf beállítóállomány határozza meg. Az állomány felépítése igen egyszerű, csak a következő két oszlopot tartalmazza:

1. Az első oszlop határozza meg, hogy milyen jellegű üzenetekre vonatkozik az adott sor.

¹Ezért tanácsos a hálózaton keresztről történő naplózáshoz egy elkülönített, csak naplózással foglalkozó kiszolgálót üzemeltetni (a lektor).

2.1. tábla: syslogd

A rendszernaplózást végző program.

`syslogd [kapcsolók]`

A program fogadja, és a beállítóállományainak megfelelő módon tárolja a naplóbejegyzéseket. Naplóbejegyzések a Linux rendszermag és az alkalmazások is küldhetnek.

Kapcsoló	Jelentés
-d	Hibakereső üzemmód.
-r	Naplóbejegyzések hálózaton keresztül történő fogadásának engedélyezése.

2. A második oszlop azt határozza meg, hogy az első oszlop által leírt jellegű üzeneteket hova kell továbbítani.

A beállítóállományban a szokásos módon – a # karakter segítségével – helyezhetünk el megjegyzéseket.

A következő példa bemutatja a /etc/syslogd.conf állomány felépítését:

- 2. példa.** A következő két sor a rendszernaplót beállító /etc/syslogd.conf állományból származik:

```
1 # Az elektronikus levelezéssel kapcsolatos üzenetek
2 mail.*          /var/log/maillog
```

A példa első sora egy megjegyzést tartalmaz, amelynek segítségével a beállítást végző rendszergazda a bejegyzés szöveges leírását megadta. A második oszlop első része (mail.*) az üzenetek egy csoportját választotta ki, a második oszlopból pedig megadta annak az állománynak a nevét, ahol az ilyen üzeneteket el kell helyezni.

Fontos megjegyeznünk, hogy a rendszernapló több sora is kiválaszthatja ugyanazzt az üzenetcsoportot. Ekkor nem történik semmi különös, az üzenetek egy része több helyen is megőrződik.

A példában láthattuk, hogy az üzenetek csoportját leíró első oszlop két mezőre oszlik, amelyeket a pont karakter választ el egymástól. A pont karakter előtt az üzenet téma, a pont után pedig a fontossága áll.

Az első oszlop első mezője – amely az üzenet kategóráját határozza meg – nem kötelező jellegű az alkalmazások számára. A naplóbejegyzést küldő alkalmazások és alrendszerök szabadon választhatnak, hogy milyen témaiban kívánják elhelyezni az egyes üzeneteket. A syslogd a következő témakat különbözteti meg:

authpriv Be- és kijelentkezással, azonosítással kapcsolatos üzenetek.

cron Az időzített feladatokkal kapcsolatos üzenetek.

daemon Háttérben futó programok általános üzenetcsoporthoz.

kern A rendszermag üzenetei.

lpr A nyomtatással kapcsolatos üzenetek.

mail Az elektronikus levelezéssel kapcsolatos bejegyzések.

news Az internetes hírcsoportokkal kapcsolatos üzenetek.

syslog A naplózással kapcsolatos üzenetek.

user A felhasználók által elhelyezett üzenetek.

uucp Az uucp (*UNIX to UNIX copy*, másolás UNIXról UNIXra) rendszer üzenetei.

localn A local0 és local7 közt található 8 kategória a rendszergazda által szabadon felhasználható, a helyi viszonyoknak megfelelően rugalmasan kezelhető.

Az első oszlop második mezőjében – a „,” után – az üzenet fontossága áll. Itt a következő kulcsszavakat használhatjuk, növekvő fontossági sorrendben:

debug Hibakeresés esetén jelentősége lehet az ilyen üzeneteknek, egyébként azonban valószínűleg lényegtelenek.

info A rendszerről információt nyújtó üzenetek.

notice Megjegyzés jellegű üzenetek, amelyeknek valószínűleg akkor fogunk jelentőséget tulajdonítani, amikor már készű.

warning Figyelmeztetés jellegű üzenetek, amelyek időben felkelthetik érdeklődésünket.

err Hibát jelző üzenetek, amelyek nem jelentik azt, hogy bármi működésképtelen volna.

crit Kritikus hibát jelző üzenetek, amelyek valamely fontos elemmel kapcsolatban fellépő problémára próbálják meg felhívni a figyelmet.

alert Riasztások, amelyek kimondottan azért kerültek a naplóba, mert rendszergazdai közbeavatkozásra van szükség.

emerg Veszélyt jelző üzenetek, amelyek akkor kerülnek a naplóba, amikor már valamelyik fontos elem nem működik.

Tudunk kell, hogy ha a fontosság valamely felsőbb szintjét írjuk be a /etc/syslogd.conf állomány egy sorába, akkor az az alacsonyabb szintre is vonatkozni fog. Ha csak egyetlen fontossági szintre akarunk hivatkozni, akkor a = jelet használhatjuk. Ezt mutatja be a következő példa:

3. példa. El szeretnénk érni, hogy a /var/log/crit.log állományba a rendszermag üzenetei és a többi üzenet közül csak a legmagasabb fontossági szintűek kerüljenek. A következő sorokat használhatjuk:

1	*.=emerg	/var/log/crit.log
2	kern.*	/var/log/crit.log

A példa első sora minden kategóriára vonatkozik – a kategória helyén a * jel áll –, de csak az emerg fontossági szinten. A második sorban a kern kategória minden szintjére vonatkozó bejegyzést láthatunk.

A rendszernaplót beállító /etc/syslog.conf állomány első oszlopában több bejegyzést is felsorolhatunk a ; jellel elválasztva. Ezt mutatja be a következő példa.

4. példa. A 3. példa két sorát egy sorba zsúfolhatjuk a ; karakter segítségével:

1	*.=emerg;kern.*	/var/log/crit.log
---	-----------------	-------------------

Ügyeljünk arra, hogy ennél a módszernél nem lehetünk szóközt vagy tabulátor-karaktert az oszlopba, hiszen ezek az jelek az oszlopok elválasztására szolgálnak.

A bejegyzésekben használhatjuk a ! jelet, amely a tagadást jelzi. Ezt mutatja be a következő példa:

5. példa. Használjuk a ! karaktert az = jellel és anélkül a kivétel jelzésére!

1	lpr.*;lpr.!notice	/var/log/lpr.log
2	mail.*;mail.!notice	/var/log/mail.log

A példa első sorában a nyomtatással kapcsolatos üzenetek minden szintjét kijelöltük, kivéve a notice szintet. A második sorban a levelezés üzeneteinek minden szintjét kijelöltük, amely a notice szint felett van.

A rendszernapló beállítóállományának (/etc/syslog.conf) második oszlopában adjuk meg, hogy az üzenetek hová kerüljenek. Ebben az oszlopban az első betű határozza meg azt, miképpen kell értelmezni az ide írt szöveget. A következő az egyes karakterek jelentése:

2.2. tábla: mkfifo

Csővezetékek létrehozására használható program.

mkfifo [kapcsolók] állománynevek

A program segítségével névvel ellátott, azaz könyvtárbejegyzésként az állományrendszerben megjelenő csővezetékeket hozhatunk létre. A csővezetékek programok összekapcsolására használhatók.

Kapcsoló Jelentés

-m jogok A létrehozandó könyvtárbejegyzéshez tartozó jogok megadása.

/ Szöveges állományok.

Ezek a bejegyzések állományokat jeleznek, amelyek nevét abszolút állományleíróval kell megadnunk. A rendszer az üzenetet az állomány végéhez fűzi.

- Szöveges állományok lemezgyorsítótár használatával.

Az állományok írása után a syslogd fizikailag is a háttértárra írja az adatokat, hogy áramszünet esetén lesz az üzenet újraindítás után is elérhető maradjon.

Ha az abszolút állományleírót egy - jellel vezetjük be, ez a művelet elmarad. Így gyorsabb működést kapunk, de a napló probléma esetén esetleg hiányos lehet.

| Csővezetékek.

Ezek a bejegyzések névvel rendelkező csővezetékekre mutatnak, amelyeket előzőleg a mkfifo segítségével létre kell hoznunk.

© Naplóbejegyzések távoli számítógépre.

Ezt a jelet távoli naplózáskor használjuk, utána a naplózást végző gép nevét írjuk.

* Ha ezt a karaktert írjuk célként, akkor az üzenet minden belépett felhasználó képernyőjén megjelenik,

a-z Ha a cél nem a felsorolt jelek valamelyikével kezdődik, hanem egyszerű betűvel, a syslogd feltételezi, hogy felhasználó nevét adtuk meg, esetleg több felhasználó nevét soroltuk fel vesszővel elválasztva. Ilyenkor – ha az adott felhasználó vagy felhasználók be vannak jelentkezve – az üzenetek megjelennek a képernyőiken.

2.3. tábla: logger

Naplóbejegyzések létrehozása.

logger [kapcsolók] Üzenet...

A program segítségével naplóbejegyzéseket hozhatunk létre a rendszernaplóban. A program nagyon hasznos lehet a rendszergazda által készített és időzített feladatként futtatott héjprogramok üzeneteinek kezelése során.

Kapcsoló	Jelentés
-p fontosság	Az üzenet kategóriájának és fontosságának megadása (például mail.info)
-s	Az üzenet a szabványos hibacsatornán is megjelenik.
-t szöveg	Bevezetőszöveg megadása.

Amint láthatjuk a rendszernapló beállításai viszonylag könnyen megváltoztathatók a /etc/syslogd.conf állomány módosításával. Ha a beállításokat módosítottuk, a rendszernapló szolgáltatást újra kell indítanunk, vagy a SIGHUP üzenetet kell küldenünk a syslogd programnak. Ez utóbbit mutatja be a következő példa:

6. példa. A beállítóállomány megváltoztatása után szeretnénk, ha a syslogd program újra beolvasná azt, és a megfelelő módon megváltoztatná a működését. Keressük meg a futó rendszernapló PID-jét, és küldjük neki a SIGHUP üzenetet!

```
$ ps aux | grep syslogd
root  2229  0.0  0.2  2924  588 ?          Ss   10:52   0:00 syslogd -m 0
root  3721  0.0  0.2  3912  664 pts/3      R+   11:26   0:00 grep syslogd
$ kill -SIGHUP 2229
$ tail -n 1 /var/log/messages
Feb  1 11:26:14 localhost syslogd 1.4.1: restart.
$
```

Amint láttuk, magából a rendszernaplóból is kiderül, hogy a rendszer naplózását végző alrendszer újraindult, és újra beolvasta a beállítóállományt.

2.1.4. Bejegyzések elhelyezése a naplóban

A logger parancs segítségével a felhasználók is elhelyezhetnek naplóbejegyzéseket a rendszernaplóban, sőt könnyedén készíthetünk olyan héjprogramokat, amelyek a naplóban rögzítik a működésük során bekövetkező eseményeket. A logger program segítségével létrehozott naplóbejegyzések alapértelmezés szerint a user kategóriába fognak kerülni.

7. példa. Helyezzük el egyszerű üzenetet a rendszernaplóban, és vizsgáljuk meg, milyen formában jelenik meg a naplóállományban!

```
$ logger Most kezdtem dolgozni.  
$
```

A parancs -t kapcsolójával adhatunk meg egy bevezetőszöveget a naplóbejegyzéshez, a -p segítségével pedig egy fontossági szintet:

```
$ logger -t Pipás -p 8 Kezdek fáradni...  
$
```

A fenti parancs a következő naplóbejegyzést helyezi el:

```
1 JAN 24 23:19:36 localhost Pipás: Kezdek fáradni...
```

A rendszernaplóról szóló szakasz végén érdemesnek tartjuk megjegyezni, hogy minden napló annyit ér, amennyit azzal a rendszergazda foglalkozik. Ha a rendszergazda nem olvassa rendszeresen a rendszernaplót, azt akár ki is kapcsolhatjuk, és erre a számítógépre elhelyezett „rúgj belém” felirattal fel is hívhatjuk a világ figyelmét².

2.2. Időzített feladatok

A GNU/Linux rendszereket többféle időzítő rendszerrel is felszerelik. Ezek az alrendszerek lehetővé teszik, hogy a számítógép bizonyos feladatokat automatikusan végrehajtson akkor is, ha a rendszergazda nem tartózkodik a közelben. A jól képzett szakember tudja, hogy milyen fontosak ezek az időzített feladatvégrehajtások, hiszen ezek nélkül éjjel-nappal a számítógép közelében kellene tartózkodnia, és olyan napi rutinfeladatok tömkelegét kellene elvégeznie, amelyekre amúgy a megfelelő héjprogramokat használja.

A következő oldalakon három időzítőrendszt mutatunk be:

- A késleltetett feladatok kezelésére használható at rendszer.

Ez a rendszer egyszeri feladatvégzésre használható, a segítségével bármi-lyen program elindítható a pontos időpont megadásával. Ezt a rendszert akkor használjuk, ha az adott programot csak egyszer kívánjuk futtatni.
- A programok szabályos időközönként ismétlődő végrehajtására használható cron rendszer.

Ezzel a rendszерrel szintén tetszőleges programok indíthatók, az at rendszerrel szemben a cron azonban elsősorban ismétlődő programvégrehajtást tesz lehetővé. Ha tehát valamelyen programot hetente, naponta vagy éppen óránként végre szeretnénk hajtani, a cron rendszert használjuk.

²A rendszernapló elemzésénél figyelembe kell vennünk, hogy GNU/Linuxon a nem privilegizált felhasználók is képesek bármilyen üzenetet küldeni a naplófájlokba, így hamisítva azok tartalmát (a lektor).

2.4. tábla: at

Időzített feladatok készítése.

at [kapcsolók] időpont

A program segítségével egyszer végrehajtandó időzített feladatot hozhatunk létre. Az időzített módon végrehajtandó parancsokat a program a szabványos bemenetről olvassa. Az időzített feladat indításának időpontját a program szinte minden szokásos formában elfogadja (például 11:30, now + 7days, 10am Jul 31 stb.).

Ha a programot az atq parancssal indítjuk, az érvényben lévő időzített feladatokat sorolja fel, ha az atrm parancssal, akkor pedig az időzített programok közül törlőhetünk.

Kapcsoló Jelentés

-f fájl	A parancsokat az adott állományból olvassa.
-v	A végrehajtás pontos időpontját a program kiírja.

- A szintén ismétlődő, de nem szigorú időbeosztás szerint végrehajtandó feladatok elvégzésére az anacron rendszer használhatjuk .

Az anacron rendszer olyan számítógépeken lehet hasznos, amelyek nincsenek folyamatosan bekapsolva, hiszen ez a rendszer a kikapcsolás miatt elmaradt feladatokat képes bepótolni.

A következő oldalakon e három időzítőrendszer használatát mutatjuk be.

2.2.1. Késleltetett feladatok

A felhasználó, aki valamilyen feladatot késleltetve, meghatározott időpontban akar elindítani, valószínűleg az at parancsot használja. Ezt mutatja be a következő példa:

8. példa. Elhatározzuk, hogy háromnegyed óra múlva hazaindulunk. Küldjünk a 45 perccel öregebb önmagunknak egy elektronikus levelet figyelmeztetésül:

\$ at now+45minutes

```
warning: commands will be executed using (in order) a) $SHELL
b) login shell c) /bin/sh
at> echo "Menni kell hazá!" | mail pipas
at> EOT>
job 3 at 2002-01-07 18:47
$
```

2.5. tábla: atd

Az időzített feladatok végrehajtásáért felelős program.

atd [kapcsolók]

Az időzített feladatokat az atd program hajtja végre. Az atd programot általában a szolgáltatásoknál megsokott módon a /etc/init.d/ könyvtárban található héjprogrammal indítjuk.

Kapcsoló **Jelentés**

-1 szám Megadja, hogy a batch programmal létrehozott időzített feladatokat milyen terhelés elérésekor halasztjuk későbbre. Többprocesszoros rendszerek esetén ezt az értéket érdemes magasabbra állítani, mert a többprocesszoros rendszereknél a terhelés a processzorok számának megfelelő szorzóval számítható ki.

Amint látjuk, az **at >** parancskérő jelnél az időzített módon végrehajtandó parancsokat kell begépelnünk, majd a parancsok beírása után egy állományvégjelet kell küldenünk a **[Ctrl]+d** billentyűkombinációval.

Az at program helyett használhatjuk a batch programot is, melynek igen hasonló a viselkedése. A különbség csak annyi, hogy a batch az időzített parancsokat szükség esetén késlelteti addig, amíg a rendszer terhelése elég alacsony nem lesz. A batch által indított programok tehát nem akkor terhelik a számítógépet, amikor az amúgy is terhelt.

Az at program segítségével létrehozott időzített feladatokat az atd program indítja, az időzített feladatok végrehajtásához tehát futnia kell az atd szolgáltatásnak. A szolgáltatás a szokásos módon indítható és állítható le (lásd 1. példa a 14. oldalon).



Sokaknak problémát okoz, hogy egyes démonokat pontosan úgy hívunk, mint ahogyan az őket indító héjprogramokat.

Az at démon például a /usr/sbin/ alkönyvtárban található atd bináris program. Ennek a szolgáltatásnak az indítására általában a /etc/init.d/ alkönyvtárban található atd héjprogram szolgál. Ha a szolgáltatás indításakor nem adjuk meg a /etc/init.d/ könyvtár nevét, akkor a bináris program indul el, és nem a szándékunk szerint indítani kívánt héjprogram.

A helyzetet csak bonyolítja, hogy sokan elfeleddik, hogy GNU/Linux rendszereken a munkakönyvtár – aktuális könyvtár – nincs a keresési útban, ezért előbb a cd parancs segítségével a /etc/init.d/ munkakönyvtárat állítják be, majd kiadják az atd parancsot. Természetesen a munkakönyvtár aktuális állapotától függetlenül a keresési útban megtalálható /usr/sbin/atd program indul el, ha csak nem adjuk meg a parancsban a /etc/init.d/ alkönyvtárat.

Feladatok kiíratása

A rendszergazda bármikor megvizsgálhatja, hogy a rendszeren hány olyan feladat várakozik, amelyet az at vagy a batch parancs segítségével helyeztek el a várakozási sorban (a felhasználó csak a saját feladatait írathatja ki). Erre az atq program használható.

9. példa. Írassuk ki a késleltetett feladatokat a képernyőre!

```
$ atq
3      2002-01-07 18:47 a root
4      2002-01-08 16:00 a pipas
$
```

A program kimenetének első oszlopában a feladat azonosítószámát találhatjuk, amely egyértelműen azonosítja a késleltetett feladatot. A következő oszlopokban a dátum és az idő található, amely a feladat indításának időpontja. Az utolsó oszlopban a felhasználó neve található, aki a késleltetett feladatot kérte.

Feladatok eltávolítása

A felhasználó eltávolíthatja a saját feladatait, mielőtt azok még elindulnának, a rendszergazda pedig ezt bárki feladataival megteheti. Az eltávolításra az atrm program használható, amelynek a törlendő feladat számát kell megadnunk.

10. példa. Távolítsuk el az előző feladat 3. időzített feladatát!

```
$ atrm 3
$
```

Amint látjuk, a feladat igen egyszerűen eltávolítható.

A használat korlátozása

A rendszergazda korlátozhatja azon felhasználók körét, akik az at segítségével feladatokat késleltethetnek. Erre a /etc/at.allow és /etc/at.deny állományok szolgálnak. Mindkét állományban felhasználói neveket sorolhatunk fel, minden sorban egyet.

Ha az at.allow állomány létezik, akkor csak azok a felhasználók használhatják az at parancsot, akiknek a felhasználói neve itt szerepel.

Ha nem létezik az at.allow állomány, akkor a rendszer megvizsgálja, hogy létezik-e az at.deny. Ha ez az állomány létezik, akkor csak azok a felhasználók késleltethetik a feladataikat az at parancsal, akiknek a felhasználói neve nincs itt felsorolva.

A rendszergazda mindenkorban használhatja az at parancsot, függetlenül attól, hogy mi van az at.allow és az at.deny állományokban.

2.6. tábla: crond

A program az időszakosan ismétlődő időzített feladatok végrehajtásáért felelős.

crond [kapcsolók]

Ez a program indítja az időszakosan ismétlődő időzített feladatokat. A crond programot általában a szolgáltatásoknál megszokott módon a /etc/init.d/könyvtárban található héjprogrammal indítjuk.

Kapcsoló	Jelentés
-n	A program indulás után démonként a háttérben fut. Ezt a viselkedést kapcsolhatjuk ki e kapcsolóval (tesztelés céljára).
-p	A program alapértelmezés szerint nem hajlandó figyelembe venni azokat az állományokat, amelyeket nem csak a rendszergazdának van jog a írni. Ezt a viselkedést kapcsolhatjuk ki e kapcsolóval. (Nem jó ötlet kikapcsolni ezt a „paranoiás” viselkedést, mert a behatolók sok esetben az időzített feladatokat használják fel a visszatérés lehetőségek biztosítására.)

2.2.2. Időszakosan végrehajtandó feladatok

A rendszergazda számára igen fontos eszköz a crond, amely időszakos feladat-végrehajtást tesz lehetővé. A segítségével előre meghatározhatjuk, hogy egyes parancsok, héjprogramok végrehajtására mikor kerüljön sor, így az automatizált rutinfeladatok olyan időpontokban futtathatjuk, amikor azok a felhasználók munkáját nem zavarják.

Az időzített feladatok kezeléséhez a crond szolgáltatást kell elindítanunk a szolgáltatásoknál megszokott módon (lásd 1. példa a 14. oldalon).

Az attól démonnal ellentétben a crond nem csak egyszeri programfutásra képzült, képes arra, hogy időről időre futtasson feladatokat az előre beállított időpontokban.

Új feladatok létrehozása

Új feladatokat legegyszerűbben a crontab -e kapcsolójával hozhatunk létre. A -e kapcsoló a feladatok szerkesztését (edit, szerkeszt) jelenti. Ilyenkor a crontab elindítja az alapértelmezett szövegszerkesztőket – amelynek neve a \$EDITOR környezeti változóban található –, és átadja neki az időzített feladatainkat tartalmazó táblázatot. Az új feladatot ebben a táblázatban kell elhelyeznünk, amely a következő oszlopokból áll:

1	perc	óra	a_hónapban_napja	hónap	a_hét_napja	parancs
---	------	-----	------------------	-------	-------------	---------

2.7. tábla: crontab

Az időszakosan ismétlődő időzített feladatok kezelésére használható program.

`crontab [kapcsolók] [állomány]`

A program segítségével az időszakosan ismétlődő végrehajtásra kijelölt feladatokat megjeleníthetjük, módosíthatjuk és törölhetjük.

Kapcsoló	Jelentés
-l	Az időzített feladatok kiírása a szabványos kimenetre.
-r	Az időzített feladatok törlése.
-e	Az időzített feladatok szerkesztése.
-u felh	Az adott felhasználó személyes feladatainak szerkesztése (csak a rendszergazda számára).

Fontos tudnunk, hogy az egyes mezőket szóközökkel kell elválasztanunk, így a mezőkön belül nem lehetnek szóközök. Az utolsó mező – amely több oszloból is állhat – a végrehajtandó parancs és annak paraméterei.

A táblázat első 5 oszlopában határozhatjuk meg, hogy az időzített feladatot mikor kell végrehajtani. Itt az időpont pontos meghatározására a következő jeleket használhatjuk:

0-9 A legegyszerűbb eszköz a szám. Ha például az első oszlop értéke 15, az azt jelenti, hogy a feladatot az óra 15. percében kell végrehajtani.

, Több számot is felsorolhatunk vesszővel elválasztva, ügyelnünk kell azonban arra, hogy ilyenkor szóközt ne használunk. Ha a második oszlopban például a 8,10,12 szerepel, az azt jelenti, hogy a feladatot 8, 10 és 12 órákor kell végrehajtani.

- Számok segítségével intervallumokat is kifejezhetünk. Ha például a 5. oszlopban a 1-5 bejegyzést találjuk, az azt jelenti, hogy a feladatot munkanapokon kell végrehajtani (hétfő–péntek).

A vesszővel és kötőjellel írt számokat kombinálhatjuk is, például írhatjuk azt, hogy 1-4,8-10.

* A számok helyett elhelyezett * karakter a beírható összes lehetséges értéket helyettesíti. Ha tehát a 4. oszlopban a * karaktert találjuk, az azt jelenti, hogy minden hónapban végre kell hajtanunk az adott feladatot.

/ Használhatjuk a / jelet a lépésköz jelzésére, például a 12-20/2 jelentheti ezt: „déltől este nyolcig kétóránként”.

A crond démon minden percben beolvassa az időzített feladatok listáját, és végrehajtja azokat, amelyek megfeleltethetők a gép beépített órájának. Amikor

a démon egy feladatot végrehajt, minden annak a felhasználónak a nevében futtatja a feladatot, aki elhelyezte azt³, futtatás előtt pedig az adott felhasználó saját könyvtárába lép⁴.

Az időzített feladatok elvégzéséről a feladatot elhelyező felhasználó elektronikus levélben kap értesítést. A levél tartalmazza a feladat végrehajtásakor keletkezett üzeneteket, vagyis a parancs szabványos kimenetén és szabványos hibacsatornáján megjelenő szöveget.

Feladat törlése

A legegyszerűbben úgy törölhetünk egy időzített feladatot, ha újra kérjük a feladatok szerkesztését a crontab -e kapcsolójával és egyszerűen eltávolítjuk az adott sort. Ha csak ideiglenesen akarjuk szüneteltetni a feladatot, lehetünk egy # karaktert az adott feladatot tartalmazó sor elejére. A crond ugyanis nem veszi figyelembe azokat a sorokat, amelyeknek első karaktere egy #.

Az összes időzített feladatunkat törölhetjük a -r kapcsolóval, amely a teljes táblázat törlésére ad parancsot.

A feladatok kiíratása

A crontab -l kapcsolójával kiírathatjuk a feladatokat.

A rendszergazda bárkinek az időzített feladatait kezelheti – így ki is írathatja –, a -u kapcsolóval megadva a felhasználó nevét. Ezt mutatja be a következő példa.

11. példa. Rendszergazdaként kíváncsiak vagyunk arra, hogy egy adott felhasználó milyen időzített feladatok végrehajtását kérte. Használjuk a crontab -u kapcsolóját a felhasználó felhasználói nevének megadására:

```
$ crontab -l -u pipas
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.7893 installed on Sun Jan  6 20:46:48 2002)
# (Cron version -- $Id: rendszergazda-II.tex,v 1.4 2005/05/28
# 11:16:22 pipas Exp pipas $)
0 * * * * echo "Üra van!"
```

\$

Amint látjuk, a rendszergazda biztonsági okokból megfigyelheti, milyen feladatokat futtattak a felhasználók távollétében.

³Ez természetes, hiszen ha nem így volna, a rendszerrel igen könnyen vissza lehetne élni.

⁴Sok megvalósításban a crond nem használja a felhasználó számára beállított héjt, így kellő előre-látással lehetséges a korábbi, érvényes héjprogramot visszaállítani (a lektor).

A rendszergazda közvetlenül is betekintést nyerhet a crond által időzített feladatokba. Ezek a feladatok ugyanis a `/var/spool/crond/` könyvtárban találhatók, minden felhasználónak a felhasználói nevével megegyező nevű állományban. Innen a rendszergazda egyszerű állománykezelő parancsokkal kiolvashatja az összes feladatot. Ezt mutatja be a következő példa.

12. példa. Írassuk ki egy bizonyos felhasználó időzített feladatait a `crontab` programmal és általános állománykezelő programokkal is! Vessük össze a két megoldás eredményét!

\$ crontab -lu pipas

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.7893 installed on Sun Jan  6 20:46:48 2002)
# (Cron version -- $Id: rendszergazda-II.tex,v 1.4 2005/05/28
# 11:16:22 pipas Exp pipas $)
0 * * * * ls

$ cat /var/spool/cron/pipas
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.7893 installed on Sun Jan  6 20:46:48 2002)
# (Cron version -- $Id: rendszergazda-II.tex,v 1.4 2005/05/28
# 11:16:22 pipas Exp pipas $)
0 * * * * ls

$
```

Amint látjuk, a két módszer ugyanazt az eredményt adja.

2.2.3. Időzítés munkaállomásokon

A crond igen hasznos eszköz, de nincs felkészülve arra, hogy a számítógépet hosszabb időre kikapcsoljuk. Ha az időzített feladat végrehajtására előírt időpontban a számítógép nincs bekapcsolva, vagy nem fut a crond, a feladat végrehajtása elmarad.

Azokban az időkben, amikor a crond készült, a számítógépeket nem volt szokás kikapcsolni. Ha tehát a rendszergazda beállította, hogy minden éjszaka lefusson egy feladat, biztos lehetett benne, hogy az adott munkát a gép minden nap végrehajtja. Ma azonban általában kikapcsoljuk a számítógépet, ha hosszabb ideig nem használjuk, ezért az éjszakai óráakra időzített feladatok elmaradnak.

A probléma megoldását egy újfajta időzítőrendszer jelentheti, amelyet az anacron démon valósít meg. Az anacron használatával a feladatokat nem napszakhoz, órához vagy perchez kötjük, csak azt állítjuk be, hogy az egyes feladatokat hány naponta kell elvégezni. Így tulajdonképpen megkerüljük a számítógép kikapcsolásából adódó problémát, lemondunk viszont arról, hogy a rutinmunkát azokra az órákra időzíthessük, amikor a felhasználók nem használják a rendszert.

2.8. tábla: anacron

A program munkaállomásokon használható, időszakosan ismétlődő feladatok végrehajtását végezi.

anacron [kapcsolók]

A program munkaállomásokon – ahol a számítógépet rendszeresen kikapcsoljuk – használható időszakosan ismétlődő feladatok indítására. Az anacron programot általában a szolgáltatásoknál megszokott módon a /etc/init.d/könyvtárban található héjprogrammal indítjuk.

Kapcsoló Jelentés

-d	A program indulás után démonként, a háttérben fut. Ezt a viselkedést tiltja ez a kapcsoló (hibakeresésre használható).
----	--

Az anacron által időzített feladatokat a /etc/anacrontab nevű állomány tartalmazza. Az állományban található időzített feladatok meghatározása négy oszlopból áll, melyek a következők:

1. Az első oszlop adja meg, hogy hány naponta kell az adott feladatot futtatni.
2. A második oszlop adja meg, hogy az anacron indulása után hány perccel indítsa a feladatot – ha szükséges.
3. A következő oszlop a feladatot azonosító név. Ha az anacron ezt a nevet kapja paraméterként, akkor csak az adott feladatot fogja vizsgálni és esetleg futtatni.
4. A negyedik oszlop a futtatandó parancsot tartalmazza.

A következő példa jól mutatja az anacrontab felépítését⁵:

```

1 # /etc/anacrontab: configuration file for anacron
2 # See anacron(8) and anacrontab(5) for details.
3
4 SHELL=/bin/sh
5 PATH=/sbin:/bin:/usr/sbin:/usr/bin
6
7 # These entries are useful for a Red Hat Linux system.
8 1      5      cron.daily      run-parts /etc/cron.daily
9 7      10     cron.weekly    run-parts /etc/cron.weekly
10 30     15     cron.monthly   run-parts /etc/cron.monthly

```

⁵Forrás: RedHat 7.2 alapbeállítás.

2.9. tábla: man

A rendszer dokumentációjának olvasóprogramja.

`man [kapcsolók] [fejezet] program`

A programnak a megfelelő program vagy állomány nevét paraméterként megadva elolvashatjuk a rendszeren található dokumentáció vonatkozó részét. Ha az adott néven több fejezetben is találhatók részek, a fejezet számának megadásával választhatunk.

Kapcsoló Jelentés

- | | |
|-----------|---|
| -a | Az összes vonatkozó kézikönyvoldalt megmutatja egymás után. |
| -K szöveg | Az adott szövegrész teljes körű keresése (ami általában igen sokáig tart). |
| -w | Nem mutatja meg a kézikönyvoldalt, csak az állomány nevét, amelyben megtalálható. |

A példa 1., 2. és 7. sora megjegyzés, amely a szokásos módon a # karakterrel kezdődik. A 4. és 5. sor környezeti változók értékadását tartalmazza, amelyek minden feladatra érvényesek lesznek.

A tulajdonképpeni feladatok az 8–10. sorban találhatóak. A 8. sor naponta – az első oszlop értéke 1 –, a 9. sor hetente, míg a 10. sor havonta végrehajtandó feladatot tartalmaz. Annak érdekében, hogy a feladatokat ne egyszerre hajtsa végre a rendszer, a feladatokat a második oszlopból ötperces eltolódással soroljuk fel.

Az at rendszernél megismert módon használhatjuk a /etc/cron.allow és /etc/cron.deny állományokat a cron használatának szabályozására. A Red Hat és a Debian alapú rendszereken a cron a PAM segítségével is ellenőrzi, hogy a felhasználónak van-e jog a időzített feladatok létrehozására.

2.3. Kézikönyv

A kézikönyvoldalak karakteres felületen a `man` parancs, grafikus felületen a `tkman` vagy az `xman` segítségével tekinthetők meg. A kézikönyvoldalakat igényes formában kinyomtathatjuk, vagy akár a weben is közzétehetjük, ezért igen hasznosak lehetnek.

A magyar anyanyelvű Linux-felhasználók áldozatos munkával igen sok kézikönyvoldalt lefordítottak magyar nyelvre. Ez mindenkorban igen hasznos segítség a kezdő felhasználóknak, ezért az oldalakat mindenkorban érdemes letöltenünk az Internetről és a mellékelt útmutató alapján telepítenünk. Természetesen nem csak magyar fordítás található meg az Interneten a Linux kézikönyvoldalaihoz, egyéb nyelveken is elérhetők ezek az oldalak. Ha többszínű környezetben dol-

gozunk, valószínűleg igen hálásak lesznek a felhasználóink, ha az ō nyelvükön is elérhetők a legfontosabb kézikönyvoldalak. A magyar nyelvű kézikönyvoldalak olyan szolgáltatást jelentenek a Linux számára, amely még a meglehetősen drága és igényes, kereskedelmi forgalomban kapható UNIX-változatokból is hiányzik.

A programok leírásait tartalmazó kézikönyvoldalak hagyomány szerint a következő szakaszokból állnak:

NAME A program neve és a működésének egysoros leírása.

SYNOPSIS Gyors segítség a program indításához. Itt az elhagyható – nem kötelező – argumentumok szöglletes zárójelek között olvashatók, az olyan argumentumokat pedig, amelyeket többször is megismételhetünk, a „...” kiegészítés követi. Szintén különleges jelentése van itt a | karakternek, amely a választható elemek elválasztására szolgál.

DESCRIPTION Részletesebb leírás a program működéséről. Ez általában több bekezdésre tagolódó folyó szöveg, nem csak vázlat.

OPTIONS A program kapcsolói, felsorolásszerűen. Itt minden kapcsolóról olyan részletekkel leírást találunk, amely elegendő a szerepének tökéletes megértéséhez.

EXAMPLES Példák a program használatához. Ez a szakasz egyszerűbb programoknál elmaradhat.

FILES A programra hatással lévő beállító- és adatállományok.

BUGS A program ismert hibái, a hibabejelentés módja.

SEE ALSO A kapcsolódó kézikönyvoldalak.

AUTHOR A program, a kézikönyv szerzője (szerző), valamint az esetleges fordító.

UNIX rendszereken a kézikönyvoldalak formázását általában a troff dokumentumformázó rendszer végzi, ezért kézikönyveket a troff parancsok alapszintű ismeretével készíthetünk. A következő sorok bemutatják egy egyszerű kézikönyvoldal felépítését.

```
1 .TH START 1 "2002. január" "változat 1.0"
2
3 .SH NÉV
4 start \- Programok indítása
5
6 .SH HASZNÁLAT
7 .B start
8 [ \fIkapcsolók\fP ] <kategória>
9
```

```

10 .SH LEÍRÁS
11 A start programok indítását végzi az alkalmazások
12 nyilvántartása alapján.
13
14 .SH KAPCSOLÓK
15 .I A start kapcsolói a következők:
16 .TP
17 \fB--version\fP
18 A program a változatszámát a szabványos kimenetre írja, majd
19 kilép.
20
21 .SH ÁLLOMÁNYOK
22 .IP /etc/applications 20
23 alkalmazások adatbázisa
24 .IP $HOME/.userrc 20
25 a felhasználót leíró beállítóállomány
26
27 .SH HIBÁK
28 A programnak a legfőbb hibája az, hogy még nem írtam meg.
29 Megígérem, hogy meg fogom írni. :--)
30 .LP
31 Egyébként azért csináltam ezt a lapot, hogy legyen egy
32 koncepcióm, mielőtt nekiállnék programozni.
33
34 .SH SZERZŐ
35 Pere László (pipas@linux.pte.hu)

```

Példánkban elenyésző számú troff parancsot használunk a kézikönyvoldal formázására. A példában kétféle troff parancsot figyelhetünk meg, melyek a következő formát követik:

.parancs A pont karakterrel kezdődő parancsok minden sor elején találhatók, a pont a sor első karaktere. A troff különleges jelentőséget tulajdonít a sorok elején található pont karakternek.

Az ilyen parancsok hatása a sor végéig, sorvégig tart.

\parancs1 szöveg \parancs2 A \ karakterrel kezdődő parancsok sor közben is kezdődhettek. Az ilyen parancsoknak minden van egy párja is, amely a szöveg lezárására szolgál.

Az ilyen parancsok hatása a parancs szöveg lezárására szolgáló változatáig tart.

Az első sorban található .TH parancs a kézikönyvoldal fejlécét írja le. A .TH parancs után négy paramétert kell megadnunk, amelyek a következők:

2.10. tábla: apropos

A program a kézikönyvoldalakban való gyorskeresésre használható, a parancsok egysoros leírásában keresi a megadott kulcsszavakat.

1. A kézikönyv címe. Ez általában annak a programnak vagy állománynak a neve, amelynek a leírását a kézikönyvoldal tartalmazza.
2. A kötet száma; azt mutatja meg, hogy a dokumentáció melyik kötetébe kerül a kézikönyvoldal.
3. A következő paraméter általában a kézikönyvoldal utolsó módosításának dátuma.
4. Az utolsó paraméter a változat száma.

A 3. sorban található – és további sorokban is látható – .SH parancs a szakaszok (főcímek) jelölésére szolgál. A parancs után beírt szöveg a sor végéig a szakasz címe lesz.

A 4. sorban a parancs rövid, egysoros leírását találjuk. Fontos, hogy a kézikönyvoldalat pontosan ebben a formában kezdjük, mert a kézikönyvoldalakban kereső apropos program különben nem használható az általunk készített kézikönyvoldalon.

 Valójában a példa nem is felel meg az apropos segítségével való keresésre. Ez a rendszer ugyanis a NAME szakaszt keresi ki a kézikönyvoldalból, és az utána található sorból veszi a program egysoros leírását. Ha tehát azt akarjuk, hogy a gyorskereső parancsokkal meg lehessen találni az általunk készített kézikönyvoldalakat, akkor a NÉV szakaszcímét az angol NAME megfelelőjére kell kicserélnünk.

Ha a gyorskeresési adatbázisban el szeretnénk helyezni az általunk készített kézikönyvoldalt, akkor le kell futtatnunk a makewhatis programot, ahogyan ezt a következő példában is láthatjuk:

13. példa. Elkészítettük és a megfelelő könyvtárban elhelyeztük a start nevű program kézikönyvét. Szeretnénk, ha abban az apropos programmal kereshetnéink. A következő sorok bemutatják, hogy ezt a makewhatis program futtatásával érhetjük el:

```
$ whatis start
start: nothing appropriate
$ makewhatis
$ whatis start
start          (1) - Programok indítása
```

2.11. tábla: makewhatis

A program a kézikönyvoldalakból kigyűjti és tárolja a parancsok egysoros leírását.

makewhatis [kapcsolók]

A makewhatis program a kézikönyvoldalakból kigyűjti a parancsok egysoros leírását, hogy az elkészült adatbázisban az apropos program gyorskeresést végezhesse. A makewhatis a dokumentáció szerint helyesen kezeli az angol, cseh, olasz, finn, francia, német és spanyol nyelvű kézikönyvoldalakat, de sajnos nem kezeli helyesen a magyar nyelvűeket.

Kapcsoló	Jelentés
-u	Csak a megváltozott vagy új kézikönyvoldalakat dolgozza fel.
-s kötet	Csak a kézikönyv adott kötetével foglalkozik.

```
$ apropos Programok
start          (1) - Programok indítása
$
```

Amint látjuk, a makewhatis futtatása előtt a kézikönyvoldal nem szerepelt a gyorskereső nyilvántartásában.

A program használatának és szerepének egysoros megadása után, a kézikönyv további soráiban folyamatosan írhatjuk a szakaszcímeket és az egyes szakaszokhoz tartozó szöveget. A bekezdéseket kettő újsor karakterrel vagy a .LP parancssal választhatjuk el egymástól.

Említést érdemel a .TP parancssal készített felsorolás, amely a példában a 16. sorban olvasható. Itt az első sorban kell szerepelnie a felsorolás kiemelt szövegének, a további sorokban pedig a felsorolás szövegének.

A 23. sorban látható .IP parancssal megadhatjuk, hogy hányszor karakterrel kerüljön beljebb a felsorolás törzsszövege.

A sorok belsejében használható a \fI parancs a kurzív írásmód bekapsolásához, a \fB a félkövér írásmóddhoz és a \fP a normál írás visszakapsolásához. Ezekre példát a 8. és 17. sorban láthatunk.

Az elkészült kézikönyvoldalt a következő ábrán láthatjuk.

START(1)	START(1)
NÉV	
start - Programok indítása	
HASZNÁLAT	
start [kapcsolók] <kategória>	

<i>Parancs</i>	<i>Jelentés</i>
.TH	a kézikönyvoldal címe
.SH	szakaszcím a lapon
.TP	felsorolás
.IP	felsorolás adott behúzással
.LP	bekezdéshatároló jel
.B	kövér betű
.I	kurzív betű
.BI	kövér kurzív betű
\fI	kurzív betű
\fB	kövér betű
\fP	normál betű

2.12. táblázat. Kézikönyvoldal-formázóparancsok

LEÍRÁS

A start programok indítását végzi az alkalmazások nyilvántartása alapján.

KAPCSOLÓK

A start kapcsolói a következők:

--version

A program a változatszámát a szabványos kimenetre írja, majd kilép.

ÁLLOMÁNYOK

/etc/applications alkalmazások adatbázisa

\$HOME/.userrc a felhasználót leíró beállítóállomány

HIBÁK

A programnak a legfőbb hibája az, hogy még nem írtam meg. Megígérem, hogy meg fogom írni.

Egyébként azért csináltam ezt a lapot, hogy legyen egy koncepcióm, mielőtt nekiállnék programozni.

SZERZŐ

Pere László (pipas@linux.pte.hu)

3. fejezet

A rendszermag

3.1. A rendszermag újrafordítása

Néhány évvel ezelőtt a UNIX rendszermag fordítása több napos munka volt, amelyre a környék összes szakértője és az érdeklődők népes tábora futott össze. Ráadásul azok a rendszermagok messze egyszerűbbek és kisebbek voltak! Ma már bárki újrafordíthatja az otthoni gépén a Linux rendszermagot, és még csak nem is kell túl sokat várnia az eredményre.

Mielőtt azonban nekilátnánk a rendszermag újrafordításának, érdemes elgondolkodnunk egy pillanatra, hogy szükségünk van-e rá egyáltalán! A legtöbb GNU/Linux terjesztéshez az előre beállított és lefordított rendszermag a rendelkezésünkre áll, ráadásul a rendszermaghoz a meghajtóprogramok, kiegészítők modulként szintén telepítve vannak, azokat csak be kell tölteni. A Linux rendszermag az utóbbi időben a moduláris felépítés irányába fejlődött, szinte minden lehet modulként is használni, így a rendszermagot nem kell újrafordítanunk, elegendő a megfelelő modult betölteni.

Néha azonban valóban szükséges lehet a rendszermag újrafordítása. Ha például valamilyen újdonsággal akarunk kísérletezni, a rendszermagot nem hivatalos forrásból származó bővítméssel akarjuk ellátni, vagy olyan eszközöt akarunk használni, amely ritkaságszámba megy, és ezért kimaradt a terjesztéshez adott bináris rendszermagból, a magot újra kell fordítanunk. Természetesen az sem probléma, ha a rendszermagot szórakozásból fordítjuk le, hiszen sokat lehet tanulni az új változatokkal való babrálásból, de érdemes tudatosan végiggondolnunk, hogy szükségünk van-e a rendszermag újrafordítására, vagy csak egyszerűen *le akarjuk fordítani* a rendszermagot¹.

Maga a rendszermagfordítás egyáltalán nem veszélyes, és a telepítés is csak akkor okozhat kárt, ha nem vagyunk tisztában a rendszer betöltésének és indításá-

¹A rendszermag fordítása mindenképpen indokolt, ha a telepítetténél sokkal újabb forrás áll a rendelkezésünkre, tekintve, hogy nagy valószínűséggel több biztonsági és egyéb hibát is javítottak közben (a lektor).

nak folyamatával. Aki ismeri az adott gépen használt rendszertöltőt, és tisztában van a rendszerindítás folyamatával, az nyugodtan lefordíthatja és telepítheti a rendszermagot anélkül, hogy egyetlen állomány egy percre is veszélybe kerülne. A rendszertöltő hibás beállítása viszont lehetetlenné teheti a rendszer elindítását, és ez általában azért veszélyes, mert a rendszer helyreállítására tett riadt kísérletek iszonyú károkat képesek okozni. A lényeg tehát, hogy ne essünk pánikba!

A rendszertöltés folyamatáról a könyv első kötetében[82] olvashatunk részlethez.

3.1.1. A fordítás előkészítése

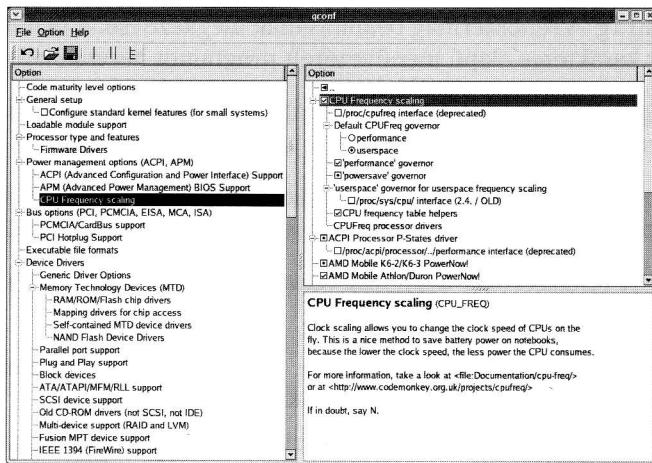
Amint azt már a programcsomagok fordításakor megszoktuk, a programokat fordítás előtt be kell állítanunk. Amíg az alkalmazások fordítás előtti beállítása általában automatikusan történik, addig a rendszermag fordítás előtti beállítása általában a rendszergazda feladata. A rendszermagot tehát fordítás előtt kézzel állítjuk be. (Tulajdonképpen éppen azért fordítjuk le a rendszermagot, hogy a fordítás előtt kézzel beállíthassuk.)

Mielőtt a rendszermag beállításának nekilátnánk, érdemes figyelmesen megvizsgálni a változat számát. A Linux rendszermag változatszámozása háromtagú, ahol az első tag változik legritkábban, az utolsó pedig a legsűrűbben. A változat számának második tagja különösen jelentős (az első túl lassan, az utolsó pedig túl gyorsan változik ahoz, hogy sokat elároljon). Ha a változat számának második tagja páros, kipróbált, ha páratlan, kísérleti változatról van szó. A kísérleti változattal igazából csak azoknak érdemes foglalkozniuk, akik szeretik a kihívásokat és csak olyan környezetben, ahol néhány leállás vagy elveszett állomány nem okoz problémát. Persze a kísérleti változat is elég stabil lehet, a stabil változat azonban *biztosan elég stabil*.

Fontos tudnunk azt is, hogy egyes GNU/Linux terjesztések különleges, többtagú változatszámmal ellátott Linux rendszermagokat is terjesztenek. A negyedik, ötödik stb. tag a változatszámban azt jelzi, hogy a hivatalos Linux rendszermaghoz képest módosításokat tartalmaz a csomag. Az, hogy miképpen vélekedünk az ilyen csomagokról, mára már vallási jellegeket öltött, ezért érdemes óvatosan, körültekintően nyilatkozni róla, vagy – ami még jobb – sehogyan sem.

A fordítás előtt a rendszermag forrását ki kell csomagolnunk valamelyik könyvtárban. A legtöbb esetben a `/usr/src/` könyvtárban szokás kicsomagolni a rendszermag forrását, így a kicsomagolás után ebben a könyvtárban jön létre a forrást tartalmazó könyvtár, melynek neve `linux-x.x.x`, az x-ek helyén a változatszámmal. A Linux rendszermag forrását tartalmazó könyvtár lehet például a `/usr/src/linux-2.6.5`. Régebbi rendszermagoknál szokás volt közvetett hivatkozást létrehozni erre a könyvtárra `/usr/src/linux` névvel, a mai változatok esetében azonban ez már nem szükséges, és nem is ajánlott.

A kicsomagolás után érdemes elolvasnunk a változásokat leíró állományt (`Documentation/Changes`), hogy lássuk, az általunk használt rendszermaghoz képest milyen változásokra kell számítanunk. Előfordulhat, hogy az új rendszermag-



3.1. ábra. A Linux rendszermag telepítés előtti beállítása

nak a programcsomagok olyan újabb változataira van szüksége, amelyek nincsenek telepítve, ezért a fordítás nem sikerülhet, vagy – ami még rosszabb – a telepítés után bizonyos dolgok nem működnek. A változások leírását átfutva felkészülhetünk az újdonságokra a megfelelő felhasználói programcsomagok letöltésével és telepítésével, így a rendszermag frissítése miatt nem válnak működésképtelennek az alkalmazásaink.

Ha gondoskodni akarunk arról, hogy a fordítás során létrejövő melléktermékek ne akadályozzák a munkánkat, használjuk a beállítás előtti tisztítást:

`make mrproper` A melléktermékként létrejött állományok törlésével gondoskodik róla, hogy a fordítás „tiszta lappal” induljon.

Nincs szükségünk erre a műveletre, ha a forrásprogramokat éppen kicsomagoltuk, vagy ha a fordítást nem akarjuk előlről kezdeni. Ha kismértékben módosítottuk a beállításokat, vagy szerkesztettük egy állományt, a parancsot nem érdemes kiadni, mert a fordítás megint az elejéről kezdődik, és az sok időt vesz igénybe.

Ellenben ha többször nekirugaszkodunk a fordításnak, és komoly változtatásokat hajtunk végre a beállítások között, szükség lehet a tisztításra, ellenkező esetben előfordulhat, hogy nem tudjuk lefordítani a rendszermagot.



Linus Torvalds saját bevallása szerint, amikor elkezdte írni a Linuxot, egy reklám ment a TV-ben, amely a Mr. Proper nevű háztartási tisztítószer ajánlotta a nézők figyelmébe...

Mielőtt lefodítanánk a Linux rendszermagot, el kell végeznünk a fordítás előtti tisztítást. Ezt a következő parancsok egyikével kérhetjük:

`make menuconfig` A rendszermag beállításának megkezdése. A beállításhoz az elindított karakteres alapú menüvezérelt programot használhatjuk.

`make xconfig` A rendszermag beállításának megkezdése. A beállításhoz a Qt programkönyvtáron alapuló grafikus programot használhatjuk (3.1. ábra).

`make gconfig` A rendszermag beállításainak megkezdése. A beállításhoz a GTK+ programkönyvtáron alapuló grafikus programot használhatjuk.

`make config` A rendszermag beállítása karakteres felületen. Ezt a módszert már ritkán használjuk.

`make oldconfig` A rendszermag telepítése a rendszermag forrását tartalmazó könyvtárban található `.config` állomány felhasználásával.

A telepítés végső lépéseként készül el a `.config` állomány, amely a beállításokat hordozható formában tartalmazza. Ez igen hasznos lehet, ha régebbi rendszermagot frissítünk, és nincs kedvünk az összes beállítást végigbogarászni. Az eljárás egyszerű: keressük meg a régebbi Linux forrásból a `.config` állományt, helyezzük el az új forrást tartalmazó könyvtárban, és indítsuk a beállítást a `make oldconfig` parancssal! A program csak azokra a beállításokra fog rákérdezni, amelyek a régi rendszermagban nem léteztek, a régi kérdéseket a régi beállításaink alapján automatikusan megválaszolja.

A fordítás előtti beállítás a Linux rendszermag minden változata esetében más és más, újabb és újabb kérdések jelennek meg, ahogyan a rendszermag bővül, fejlődik. Szerencsére minden egyes kérdéshez rövid, jól érthető magyarázat áll a rendelkezésünkre, így a beállítás némi nyelvtudást igényel ugyan, de nem reménytelen azok számára sem, akik nem járatosak a Linux rendszermag felépítésében.

A legtöbb kérdésnél arról döñthetünk, hogy szükségünk van-e egy eszközre vagy sem. Az ilyen kérdéseknek – ha lehet – érdemes a modulként való fordítást választani, hiszen így elérhető marad az eszköz anélkül, hogy a rendszermag kezelhetetlenül naggyá válna. A modulok betöltése ugyanis csak akkor történik meg, ha szükség van rájuk.

Az alapvető eszközöket, a merevlemezvezérlőt, a gyökérkönyvtár állományrendszerét, vagyis a rendszerindításhoz szükséges elemeket nem szerencsés modulként fordítani, mert gondjaink támadhatnak a rendszerindítás során, amikor a modulok még nem elérhetők.

Amikor végeztünk, a beállításokat menthetjük, és kiléphetünk a beállítóprogramból. Ez után a fordítás következik.

3.1.2. A fordítás

A rendszermag beállítása után következik a tulajdonképpeni fordítás. A fordítást a szokásos módon indíthatjuk²:

`make bzImage` A parancs hatására a Linux rendszermag fordítása elkezdődik. Ez általában kissé hosszadalmas, de nem elviselhetetlenül hosszú folyamat, amelynek eredményeként létrejön a lefordított, tömörített rendszermag.

Ha a fordítás elakad, érdemes végiggondolnunk a következő kérdéseket:

- Valamit rosszul állítottunk be a fordítás előtt? A Linux rendszermag fejlődésével egyre nehezebb olyan beállításokat találni, amelyek mellett a fordítás meghiúsul, de kis szerencsével és némi kitartással választhattunk olyan beállításokat, amelyek mellett a rendszermag nem fordítható le.

Érdemes megvizsgálnunk, hogy a fordítás a rendszermag melyik részénél fordításakor következett be, és a beállításainkat azon a környéken újra átvizsgálni.

- Lehetséges, hogy a fordítást egy félre lefordított rendszermagon kezdtük meg, és a fordítás azért állt le, mert az állományok egy részét a régi, egy másik része pedig az új beállítások mellett fordítottuk le.
Próbálkozzunk a fordítással a `make mrproper` parancs kiadása után.
- Lehetséges, hogy a fordításhoz szükséges programok valamelyike nem áll rendelkezésre? Ellenőrizzük a dokumentációban felsorolt programok változatszámát!
- Lehetséges, hogy kísérleti változatot telepítettünk – azaz a változat számának második tagja páratlan?

`make modules` A parancs hatására a magmodulok fordítása történik meg. A magmodulok fordítása és a rendszermag fordítása között igazából túl nagy különbség nincs, a rendszermag fordításakor elmondottak a modulok fordítására is igazak.

Szerencsés esetben mind a rendszermag, mind pedig a modulok fordítása hibaüzenet nélkül lezajlik. Ilyen esetben telepíthetjük a rendszermagot.

3.1.3. A telepítés

A telepítés első lépése a modulok felmásolása, amit a következő parancssal indíthatunk:

²A 2.6 változatnál régebbi rendszermagok fordítását a `make dep` parancs kiadásával kell kezdenünk amelyet az elhagyható `make clean` követ.

`make modules_install` A parancs hatására a modulok felmásolása kezdődik meg a `/lib/modules/` könyvtár megfelelő, a rendszermag változatszámmal megegyező nevű alkönyvtárába. Ha az alkönyvtár már létezik, a tartalma könyörtelenül felülíródik.

A következő lépés a rendszermag felmásolása.

A rendszermag a fordítása után az `arch/i386/boot/bzImage` tömörített állományba kerül. Ezt az állományt a szokásos módon a rendszertöltő segítségével tölthetjük be a számítógép bekapcsolásakor.

A telepítéshez tehát a megfelelő könyvtárba kell másolnunk az állományt (általában `/boot/`), a rendszertöltő beállítóállományát a megfelelő módon módosítanunk kell, és a számítógépet újra kell indítani. (Valójában az egyetlen ok, ami miatt a GNU/Linuxot futtató számítógépet újra kell indítani, az a rendszermag cseréje.)

Természetesen igen szerencsés, ha a régi rendszermagot és a régi rendszermag rendszertöltő beállításait megtartjuk, amíg az új rendszermag működőképességről meg nem győződtünk.

3.2. Magmodulok kezelése

A Linux rendszermag – eltérően a legtöbb más UNIX rendszeragtól – képes programrészeket, úgynevezett magmodulokat menet közben önmagába illeszteni, és képes ezeket a modulokat szükség esetén eltávolítani. A magmodulok kezelése, beillesztése és eltávolítása meglehetősen összetett és veszélyes folyamat, ezért a könyv előző részeiben – amennyire lehetséges volt – kerültük ezt a kényes témát, de a számítógép-hálózatok kezeléséhez szükségünk van ezekre az ismeretekre.

Tudnunk kell, hogy a UNIX rendszerek biztonsági szempontból alapjában véve kétszintű felépítésűek. minden processzor, amely képes UNIX rendszert futtatni, legalább két állapotú³. Az egyik állapotot akkor veszi fel a processzor, amikor a rendszermag utasításait futtatja, a másik állapotát pedig akkor, amikor más program utasításait hajtja végre. A rendszermag futtatása közben a processzor minden utasítást hajlandó elvégezni, más programok utasításainak végrehajtása közben azonban bizonyos műveleteket nem hajt végre, biztonsági okokból.

Programozói szempontból a processzor kétféle állapotának egyszerű a hatása: a rendszermag utasításai minden perifériát vezérelhetnek, minden memóriaterületet elérhetnek, a számítógép minden áramkörét elérhetik, míg a felhasználói programok (*user level programs*) csak a nekik kiosztott memóriaterületet használhatják, és kizárolag a rendszermag szolgáltatásainak közbeiktatásával érhetik el a perifériákat.

³A Linuxot portolták IBM XT-re is, ahol a processzornak nincsenek biztonsági szintjei (a lektor).

3.1. tábla: lsmod

A rendszermagba betöltött modulok listájának kiírása.

```
lsmod
```

A program kiírja a betöltött magmodulok nevét, méretét, és azt, hogy melyik magmodul van valójában használatban.



Meg kell említenünk, hogy a legtöbb processzor kettőnél több biztonsági állappal rendelkezik, és vannak olyan operációs rendszerek, amelyek ki is használják a több állapot adta lehetőségeket, fel is használják az állapotokat. A UNIX operációs rendszer az egyszerűség kedvéért használ két állapotot.

A programok minősége szempontjából a két biztonsági állapot fontos megkülönböztetést jelent a rendszermag és a felhasználói szinten futó programok közt. Egy hibás program felhasználói szinten futtatva megkíséríthet ugyan meg nem engedett műveletet végrehajtani, azt azonban a processzor egyszerűen nem végzi el. A hibás felhasználói programok tehát fennakadnak a védelmi rendszeren⁴, és általában nem okoznak túl nagy kárt. Egészen más azonban a helyzet a rendszer-maggal. Ha hiba van a rendszermagban, ha a rendszermag „rossz” utasításokat ad, a teljes rendszer könnyen működőképtelen válik.

A Linux rendszermag magmoduljai betöltéskor összeszerkesztődnek a rendszer-maggal, a magmodul utasításai tehát a rendszermagba épülve futnak. Ha a mag-modulban hiba van, a teljes rendszer működésképtelen válik.

A magmodulok kezelése ma már többnyire automatikus. A mag automatikusan tölti be a modulokat, ha szükség van rájuk, és automatikusan el is távolítja őket a memoriából, ha már nem kellenek. A rendszergazdának ennek ellenére ismernie kell a modulokat és a modulok kezelésére használt fontosabb parancsokat. Különösen fontosak ezek az ismeretek a különféle hardverelemek kezelését végző modulok esetében.

A UNIX rendszerek közül a GNU/Linux fut a legváltozatosabb hardverkörnyezetben, valószínűleg a Linux kezeli a legtöbb fajta eszközt. A hardvermeghajtó programokra csak akkor van szükség, ha az adott elem rendelkezésre áll a számítógépen, ezért a hardvermeghajtó programokat általában a magmodulként használjuk.

3.2.1. Betöltött magmodulok kiíratása

Az `lsmod` parancs segítségével kiírathatjuk a betöltött magmodulok listáját, megvizsgálhatjuk, mely modulok aktívak.

Az újabb Linux-változatok a magmodulok támogatásának igen fejlett módját valósítják meg. Amíg a régebbi változatok a magmodulok kezelését nagyrészt segédprogramokra bízták, addig ma a segédprogramok csak egyszerű feladatokat végeznek, a munka nagy részét a rendszermagra bízzák.

Nincs ez másképp a magmodulok kiírásakor sem. Az `lsmod` a magmodulok adatait a `/proc/modules` látszólagos állományból olvassa ki, ahogyan ezt a következő példából is láthatjuk.

14. példa. Vizsgáljuk meg a `lsmod` kimenetét és a `/proc/modules` állományt!

```
$ lsmod | head -n 3
Module           Size  Used by
snd_pcm_oss      40740  1
snd_mixer_oss    13824  3 snd_pcm_oss
$ head /proc/modules -n 2
snd_pcm_oss 40740 1 - Live 0x11e94000
snd_mixer_oss 13824 3 snd_pcm_oss, Live 0x11e58000
$
```

3.2.2. Magmodulok eltávolítása

Az `rmmod` program segítségével a használaton kívüli magmodulokat eltávolíthatjuk a memoriából. A program paramétereként az eltávolítandó modul nevét kell megadnunk.

Az `rmmod -f` kapcsolója akkor is megkísérli eltávolítani a modult, ha az éppen használatban van. Ez a művelet a dokumentáció szerint veszélyes, és ez valószínűleg a legenyhébb amit a használatban lévő magmodulok eltávolításáról elmondhatunk.

Az `rmmod` nagyon hasznos kapcsolója a `-w`, amelynek hatására a program addig várakozik, amíg a modul fel nem szabadul, és csak utána távolítja el. Mi több, a program a kapcsoló hatására tiltja a modulhoz való csatlakozást, így remélhetőleg az előbb-utóbb felszabadul.

3.2.3. Magmodulok leírása

A `modinfo` program segítségével információkat kérhetünk a különféle modulokról. A program neve után a modul nevét beírva az megkeresi a modult tároló állományt, majd kiolvassa és kiírja a legfontosabb információkat a modulról. Ezt mutatja be a következő példa.

⁴A legtöbb esetben, de vannak olyan programok, amelyek felhasználói jogosultsággal működés-képtelenné tesznek bizonyos Linux változatokat (a lektor).

3.2. tábla: rmmod

Magmodulok eltávolítása a memóriából.

rmmod [kapcsolók] modulnév

A program segítségével eltávolíthatjuk a rendszermagból a betöltött magmodulokat.

Kapcsoló Jelentés

-v	Részletes információk kiírása.
-f	A modul erőltetett eltávolítása (igen veszélyes).
-w	Várakozás, míg az erőforrás felszabadul, majd a modul eltávolítása.
-s	Az üzenetek küldése a rendszernaplóba.

15. példa. Tudomásunkra jutott, hogy egy ismerősünk számítógépében be van töltve a a3d modul. Meg szeretnénk tudni, hogy a modul a mi számítógépünkön is elérhető-e, és azt, hogy mire való. Használjuk a modinfo programot!

\$ modinfo a3d

```
author:      Vojtech Pavlik <vojtech@ucw.cz>
description: FP-Gaming Assassin 3D joystick driver
license:    GPL
vermagic:   2.6.5-1.358 686 REGPARM 4KSTACKS gcc-3.3
depends:    gameport
$
```

Amint látjuk, a legfontosabb információkat kiírta a program a szabványos kimenetre. Ebből tudhatjuk, hogy a modul a számítógépen elérhető, betölthető.

A modinfo program által kiírt sorok jelentése:

author: A modul készítője.

description: A modul leírása, amelyből megtudhatjuk, hogy mire használható a modul.

license: A modul szoftverfelhasználói engedélye.

parm: A modul betöltésekor megadható paraméterek leírása. Ez az információ igen fontos, hiszen leírja, hogy a modul paramétereivel hogyan befolyásolhatjuk a modul működését.

alias: A modul másodlagos neve vagy nevei.

vermagic: A modul fordításakor tárolt információ, amely pontosítja, milyen környezetben használható a modul.

3.3. tábla: depmod

Kigyűjti a magmodulok közti összefüggéseket.

`depmod [kapcsolók] [magváltozat_száma] [állománynevek]`

A program kigyűjti a magmodulok közti összefüggéseket, és elhelyezi a megfelelő állományokban, hogy a modprobe program a modulok betöltésekor figyelembe vehesse azokat.

Kapcsoló Jelentés

- | | |
|-----------------|---|
| <code>-n</code> | A program nem frissíti az állományokat, csak kiírja az adatokat a szabványos kimenetre. |
| <code>-a</code> | Minden beállított könyvtár összes magmodulját feldolgozza. |
| <code>-v</code> | Részletes információk kiírása. |

`depends`: A függőség, amely megadja, hogy az adott modulnak milyen más modulakra van szüksége.

3.2.4. A magmodulok betöltése

Régebben a magmodulok betöltésére az insmod programot használtuk, amelynek gyakorlatilag csak a betöltendő állományt kellett megadni paraméterként. Az insmod nem túl kifinomult módon kezelte a modulok betöltésének kérdését, ezért ma általában már kizárolag a modprobe programot használjuk, ha be akarunk tölteni egy magmodult.

A modprobe képes a magmodulok közti összefüggések kezelésére, és a modulok beállításait külön beállítóállományban kezeli. Egyéb szolgáltatásaival együtt ezek a tulajdonságai a modprobe programot a magmodulok kezelésére alkalmas hatékony eszközössé teszik.

A modprobe részletes bemutatása előtt érdemes szót ejtenünk a depmod programról, amely a modulok közti összefüggések kigyűjtésére használható. A depmod megvizsgálja a modulok tárolására használt könyvtárakban az állományokat, és beolvassa a modulokból a modulok közti összefüggéseket. A program az összefüggéseket elhelyezi a könyvtárakban a modules.dep állományban, ahonnan a depmod szükség esetén beolvassa azokat, hogy a modulok betöltésekor a szükséges támogató modulokat is betölthesse. A depmod beállítására a /etc/modules.conf állomány használható.

16. példa. Új magmodulokat telepítettünk, és szeretnénk a depmod segítségével nyilvántartásba venni azokat. Használjuk a depmod -a kapcsolóját!

```
$ depmod -a
$
```

A magmodulok közti összefüggések frissítése általában meglehetősen sokáig tart.

A modprobe programmal egyszerűen betölthetjük a depmod által nyilvántartott magmodulokat, anélkül, hogy tudnánk, melyik állományban találhatók. Ezt mutatja be a következő példa:

17. példa. Be szeretnénk tölteni az msdos modult minden olyan modullal, amelyre szüksége van. Használjuk a modprobe programot!

```
$ lsmod | grep msdos
$ modprobe msdos
$ lsmod | grep msdos
msdos                  7552  0
fat                    33984  1 msdos
$
```

Amint látjuk, a program betöltötte a fat nevű modult is, amelyre az msdos modulnak szüksége volt.

A régebbi GNU/Linux terjesztésekben a /etc/conf.modules vagy a /etc/modules.conf, esetleg a /etc/modules állomány szolgált a betöltendő magmodulok beállításainak tárolására. Az újabb GNU/Linux változatokban a /etc/modprobe.conf állományban állíthatjuk be, hogy az egyes modulok betöltésekor mi történjen. Az állományban a következő kifejezéseket használhatjuk:

alias név modul A kifejezés segítségével másodlagos nevet rendelhetünk a modulhoz.

A másodlagos névben szerepelhetnek állománynév-helyettesítő karakterek (* és ?), így a modul később több néven is betölthető.

options modulnév kapcsolók A kifejezés segítségével a modulhoz kapcsolókat rendelhetünk, amelyeket a modul a betöltés során megkap.

A kifejezéssel a modulok másodlagos nevéhez is rendelhetünk kapcsolókat, így ugyanaz a modul különféle neveken különféle kapcsolókkal tölthető be.

install modul parancs A kifejezés hatására a modul beillesztése helyett a modprobe a megadott héjparancsot hajtja végre.

remove modul parancs A kifejezés hatására a modprobe a modul eltávolítása helyett az adott parancsot hajtja végre.

include állománynév A kifejezéssel a /etc/modprobe.conf állomány értelmezése során más állományokat is beolvastathatunk.

18. példa. Egy számítógépben két hálózati csatoló található, amelyeket a 8139too és az ipw2100 magmodulokban található meghajtóprogramok kezelnek. Szeretnénk az egyes hálózati eszközökkhöz tartozó modulokat eth0 és eth1 néven betölteni. Használjuk az alias parancsot a /etc/modprobe.conf állományban!

3.4. tábla: modprobe

Magmodulok betöltésére használható program.

`modprobe [kapcsolók] modulnév [modulkapcsolók]`

A program segítségével betölthetünk és eltávolíthatunk magmodulokat a `/etc/modprobe.conf` beállítóállomány és a `depmod` által készített `modules.dep` állomány alapján.

Kapcsoló Jelentés

- | | |
|---------------------|---|
| <code>-v</code> | Részletes információk kiírása. |
| <code>-r</code> | A kapcsoló hatására a program nem betölti, hanem eltávolítja a modult. |
| <code>-o név</code> | A modul betöltésekor megváltoztatja a rendszermag által nyilvántartott nevet. |

1 `alias eth0 8139too`
 2 `alias eth1 ipw2100`

Ha a jövőben egyik vagy másik hálózati csatolót kicseréljük egy másik típusra, semmit sem kell átállítanunk csak az alias kulcsszó segítségével más modulnévhez kell rendelnünk az eth0 vagy eth1 másodlagos neveket.

3.2.5. A modulok automatikus betöltése

A Linux rendszermag mai változatait felkészítik arra, hogy a magmodulokat betöltsék, amikor azokra szükség van. Ez a szolgáltatás roppantul kényelmessé teszi a magmodulok használatát.

Az elgondolás igen egyszerű: amikor egy eszközre, állományrendszerre vagy meghajtóprogramra van szüksége a rendszermagnak, az elindítja a modprobe programot, és így gondoskodik a modul betöltéséről. Mivel a rendszermag által kezdeményezet magmodulbetöltés során ugyanaz a modprobe jut szerephez, mint amikor a rendszergazda kéri a modul betöltését, az automatikus betöltéssel betöltött modulok beállítása nem igényel külön figyelmet.

A rendszermag számára a `/proc/sys/kernel/modprobe` állományban állíthatjuk be, hogy hol találja a modprobe programot. A legtöbb GNU/Linux terjesztés esetében a telepítés után már be van állítva a modprobe pontos helye (általában `/sbin/modprobe`), így ezzel nem kell foglalkoznunk. Csak arra kell ügyelnünk, hogy a modprobe megtalálja a rendszermag által kért modulokat.

A modprobe beállításakor (régebbi rendszereken `/etc/modules.conf` vagy `/etc/conf.modules`, újabb rendszereken `/etc/modprobe.conf`) a következő dolgokra kell ügyelnünk, ha az automatikus betöltést használni szeretnénk:

- A legtöbb magmodult – például a hajlékonylemez meghajtójának floppy nevű modulját, az állományrendszerek moduljait – a rendszermag a megfelelő néven hívja, és probléma nélkül betölti.
- Hálózati kártyák. A rendszermag a hálózati kártyák meghajtóprogramjait a hálózati illesztők nevével próbálja meg betölteni, ez Ethernet hálózat esetében az eth0, eth1 stb. (lásd 76. oldal). A beállítás során a megfelelő meghajtóprogramot tartalmazó modulnak ilyen másodlagos nevet kell létrehoznunk.
- Párhuzamos csatoló. Ha a rendszermagnak a párhuzamos csatolóra van szüksége, a parport_lowlevel modult kísérli meg betölteni. A beállítás során egy ilyen másodlagos modulnevet kell készítenünk valamelyik meghajtóprogram számára.
- A blokkész köz-meghajtó állományok meghajtóprogramjait a rendszermag megkísérli betölteni block-major-n néven, ahol az n a blokkész közkezelő főeszközszerkezetének névvel kell ellátnunk azt a magmodult, amelyik az adott eszközök kezelését végzi.
- A karakteres köz-meghajtó állományok meghajtóprogramjait a rendszermag char-major-n néven keresi. Itt az n helyén szintén a főeszközszerkezet névvel kell ellátnunk azt a magmodult, amelyik az adott eszközök kezelését végzi.
- A hálózati szabványcsaládokat (*protocol family*) megvalósító modulokat a rendszermag net-pf-n néven keresi, ahol az n a linux/socket.h állományban a szabványcsaládra létrehozott állandó.
- A hangkártya meghajtóprogramját a rendszermag az snd-card-n néven próbálja betölteni, ahol n 0-ról indulva adja meg, hogy hányadik hangkártyáról van szó.
- Az USB eszközök meghajtóprogramját a rendszermag usb-controller, illetve usb-controller1, usb-controller2 stb. néven próbálja meg betölteni.

A legtöbb GNU/Linux terjesztés a telepítés során úgy készíti el a modprobe beállítóállományát, hogy a rendszermag be tudja tölteni a számára szükséges modulokat.

19. példa. Vizsgáljuk meg a következő állományt:

	/etc/modprobe.conf
1	# Hálózati kártyák:
2	alias eth0 8139too
3	alias eth1 ipw2100
4	
5	# Párhuzamos csatoló:

```
6 | alias parport_lowlevel parport_pc  
7 | options parport_pc io=0x378,0x278 irq=7,auto  
8 |  
9 | # USB eszközök:  
10 | alias usb-controller ehci-hcd  
11 | alias usb-controller1 uhci-hcd
```

Az állomány a hálózati kártyák, a párhuzamos csatoló és az USB eszközök moduljai számára hoz létre olyan másodlagos nevet, amelyet a rendszermag automatikusan be tud tölteni.

4. fejezet

A hálózati kapcsolat kiépítése

A számítógép-hálózatok ma már igen elterjedtek, és igen fontos szerepet töltnek be életünkben. A számítógépfelhasználók döntő többsége nem számításokat elvégző gépként, hanem kommunikációs eszközként használja a számítógépet, nem a számítógépet, hanem a számítógép-hálózatot használja.

A GNU/Linux nagyon fejlett hálózati támogatással rendelkezik. Azt is mondhatnánk, hogy amit egy számítógéppel el lehet végezni a számítógép-hálózaton, azt a GNU/Linux segítségével meg lehet valósítani.



Néhány évvel ezelőtt vállalkozó kedvű szakemberek postagalambokkal megvalósított hálózati kapcsolatot készítettek két GNU/Linux számítógép között[168]. Az elkészült megoldás az átlagosnál talán valamivel kevesebb gyakorlati haszonnal kecsegett, minden esetben jól jelzi a GNU/Linux sokszínűségét.

A következő oldalakon a számítógép-hálózatok felépítéséről, a GNU/Linux hálózati alrendszerének beállításáról és működéséről lesz szó.

4.1. A számítógépek összekapcsolása

A számítógép-hálózatokról szóló rész első szakaszában azt vizsgáljuk meg, hogy miképpen létesíthető kapcsolat két számítógép között, amelyek GNU/Linuxot futtatnak. Az itt tárgyalt ismeretek és eszközök lehetővé teszik, hogy a számítógépek között adatátvitelt valósítsunk meg, de csak az ehhez elengedhetetlen ismereteket tárgyaljuk. A fejezet további szakaszaiban ezekre az ismeretekre – és a kiépített kapcsolatra – építve a hálózat további elemeivel ismerkedhetünk meg.

4.1.1. Bevezetés

A számítógépek összekötésére használt szabványok közül a Xerox PARC (Xerox Palo Alto Research Center, Xerox Palo Alto-i kutatóközpont) által kifejlesztett Ethernet (éteri hálózat) a legelterjedtebb. Az Ethernet – amelyet több évtizede fejlődő szabványcsaládként mára már számítógépes eszközök tömegébe építettek be – leírja, miképpen kapcsolódhatnak egymáshoz a különféle gyártók által készített számítógépek. A szabványcsalád többfélé lehetőséget ad a gépek összekötésére, többfélé átviteli sebességet valósít meg, de működésének alapja minden változat esetében megegyezik[156].



Igen érdekes, hogy Xerox PARC nem csak az Ethernet hálózat megalkotásában jeleskedett, de ők fejlesztették ki a grafikus felhasználói felületet és a fénymásoló gépet is. Az viszont már egyenesen különös, hogy ezen találományok egyikét sem jegyezték be szabadalomként, így mindenki találományból vállalatok tucatjai profitálnak ma is. Nem is csoda, hogy a Xerox PARC munkatársait sokan úgy emlegetik, hogy „azok a zseniális fickók a Xeroxnál, akik más cégek számára találnak ki dolgokat”.

A következő néhány oldalon az Ethernet hálózatok kialakításához szükséges alapismereteket foglaljuk össze – igen vázlatosan – azok számára, akik a számítógép-hálózatok építésében nem rendelkeznek tapasztalatokkal.

Ethernet hálózatok építése

Napjainkban az Ethernet vezetékes adatátviteli módjai közül a 10 Mbit/s adatátviteli sebességet megvalósító árnyékolt kábeles és a 10 vagy 100 Mbit/s adatátviteli sebességet megvalósító árnyékolatlan csavart érpáras megoldás a legelterjedtebb. Az árnyékolt kábelre épülő Ethernet hálózatok elavultnak tekinthetők, de még sok helyen megtalálhatók, míg a csavart érpáras hálózatok nem csak modernek, de mára igen elterjedtek is.

A koaksiális árnyékolt vezetékre épülő Ethernet hálózatok – amint nevük is mutatja – árnyékolt vezetéket használnak az adatátvitelre. Az előírt árnyékolt vezeték 50Ω hullámimpedanciájú, szokásos „vékony ethernet kábel” néven is emlegetni (sejtetve ezzel, hogy a régebbi, mára kihalt Ethernet kábelek igen vastagok voltak).



Az 50Ω hullámimpedancia jelentésének pontos megértéséhez nem jön rosszul egy villamosmérnöki diploma, számunkra azonban nem is fontos, hogy pontosan megértsük ezt a fogalmat.

Amit viszont fontos tudnunk, az az, hogy a televíziózásban a vékony Ethernet kábelhez kísértetiesen hasonló, 75Ω hullámimpedanciájú kábeleket is használnak. Az eltérő hullámimpedanciájú kábeleket azonban nem használhatjuk, mert kísérteties és nehezen felderíthető hibákhoz vezet¹.

¹Figyeljünk a még fellehető 70Ω hullámellenállású kábelekre és hálózati csatolókra is (a lektor).



4.1. ábra. Vékony Ethernet ellenállások és csatlakozók

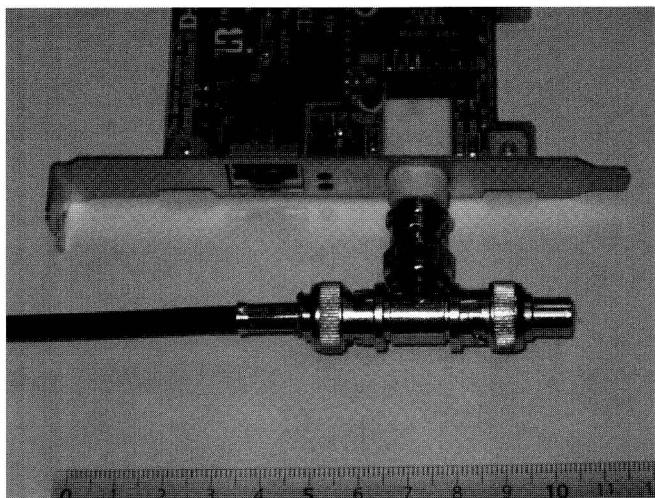
Nem szabad elfelejtenünk azt sem, hogy a hullámimpedancia és az egyenáramú elektromos ellenállás mértékegysége megegyezni látszik ugyan, de a két fizikai mennyiség egymástól lényesen különbözik. A hullámimpedancia tehát nem az az ellenállás, amit az általános iskolában is tanítanak.

A vékony Ethernet kábel kéterű (egy belső ér és egy külső árnyékolás), és balonettzáras csatlakozókkal kapcsolódik a számítógéphez. A számítógépek elektromos szempontból párhuzamosan vannak kapcsolva a kábelhez, amelynek minden végén 50Ω ellenállású úgynyevezett lezáró ellenállások zárnak rövidre. A vékony Ethernet hálózat végén található 50Ω ellenállású lezárás egyenáramú ellenállás, olcsó kézi ellenállásmérővel mérhető. Mivel a vezeték minden végén 50Ω ellenállású lezárás van, a vezeték egyenáramú ellenállása pedig elhanyagolható, a vezetéken egyenáramú ellenállásmérővel minden ponton 25Ω ellenállást mérhetünk.



A vezeték hullámimpedanciája és a lezáró ellenállása tehát pontosan megegyezik, ami természetesen nem véletlen. Ha nem megfelelő értékű lezáró ellenállást használunk, vagy teljesen elhagyjuk a lezáró ellenállást, akkor a vezetéken haladó jelek egyszerűen visszapattannak, és beleütköznek a mögöttük haladó jelekbe. Ezt természetesen nem akarhatjuk, mert olyan tololgás alakul ki, ami gyakorlatilag lehetetlenné teszi, hogy a számítógépek egymásnak üzeneteket küldjenek.

A vékony Ethernet kábelezés csatlakozót láthatjuk a 4.1 ábrán. A bal oldalon két lezáró ellenállás látható, amelyeket a vezeték két végén kell elhelyeznünk. Ezek a csatlakozók 50Ω ellenállású rövidrezáró elemeket tartalmaznak, egyszerű



4.2. ábra. Szerelt vékony Ethernet csatlakozás

ellenállásmérővel könnyedén ellenőrizhetők. Az ábra balról számított harmadik csatlakozója egy közdarab, amellyel a kábel toldható, az utolsó eleme pedig a vékony Ethernet számítógép csatlakoztatására használható T eleme. A T elem két egyforma oldalára csatlakozik a két kábel – amely a számítógéptől jobbra, illetve balra található más számítógépekhez vezet –, a középső részéhez pedig maga a számítógép. Ha a számítógép az utolsó vagy az első a sorban, a T alakú csatlakozó megfelelő helyén a lezáró ellenállást találjuk. Ezt mutatja be a 4.2. ábra, ahol a hálózati csatolóhoz egy T csatlakozó kapcsolódik, egyik végén kábelrel, másik végén pedig lezáró ellenállással.

A bajonettzáras elemek persze sokféléképpen egymáshoz kapcsolhatók (mint ahogyan jól példázza ezt az unatkozó rendszergazdák asztalán megfigyelhető, legőszerűen elemkre bontható és újra összeállítható vékony Ethernet szoborcsoport), de a sok kombináció közül csak az itt bemutatott kapcsolás eredményez működőképes hálózatot.

Ez persze többek közt azt jelenti, hogy *egy vékony Ethernet kábelből nem lehet hálózatot építeni*, azzal, hogy legalább egy kábel, két T csatlakozó és két lezáró ellenállás kell!

A vékony Ethernet hálózat kiépítése közben még egy fontos dolgot figyelembe kell vennünk, ez pedig a vezeték megengedett legnagyobb hossza. A vezeték két végén található lezáró ellenállás között legfeljebb 185 méter hosszú vezeték lehet. Ha a vezeték hossza ennél nagyobb, a hálózati kapcsolat akadozhat, esetlegessé válhat. Fontos tudnunk, hogy a vezeték hosszának korlátozására nem a vezeték ellenállása miatt, hanem a jel terjedésének sebessége miatt volt szükség. Ne bízzunk tehát abban, hogy a hálózat hosszabb vezetékkel is működni fog, ha jobb minőségű vezetéket sikerül beszereznünk.



Az Ethernet hálózatok szabványainak fejlődése során több koaxiális kábelre épülő változat is elterjedt, mi ezek közül csak a legelterjedtebbet tárgyalunk. Ennek 10BASE-2 a neve, a leíró szabvány pedig az IEEE 802.3, amely mára elavultnak tekinthető. A újabb szakirodalom sokszor már nem is foglalkozik ilyen hálózatokkal, de szakirodalom[60] és hálózati eszköz is található 10BASE-2 hálózatok készítéséhez.

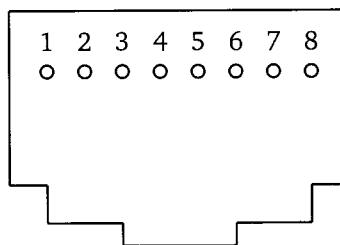
A vékony Ethernet kábelezéssel kialakított hálózat nagy előnye, hogy igen olcsón és egyszerűen elkészíthető. A régebbi Ethernet kártyákat a legtöbb számítógépes munkahelyen a fiókokban még fellehetjük, vagy a használtcikkkereskedésekben beszerezhetjük, a csatlakozók és a kábelezés pedig ma már filléres áron elérhető, így a vékony Ethernet kitűnően használható otthoni hálózatok készítésére.

A vékony Ethernet legnagyobb hátránya a sérülékenysége. Ha a csatlakozóval összekapcsolt elemeket megbontjuk egy ponton, a hálózat teljes szakasza működésképtelen válik, hiszen két szakaszra bomlik, amelyek egyik végén hiányzik a vezeték ellenállás. A számítógép ugyan büntetlenül eltávolítható a vékony Ethernet hálózatból, de a T csatlakozó nem! Mivel a legtöbb laikus ezt nem tudja, a nagyméretű, sok felhasználót kiszolgáló vékony Ethernet hálózatok általában csak életveszélyes fenyegetések árán tarthatók üzemben, ami a legtöbb esetben nem növeli jelentősen a rendszergazda népszerűségét.

Egészen más felépítésű az Ethernet hálózatok modernebb, úgynevezett árnyékolatlan csavart érpáras változata, amelyet általában UTP (*unshielded twisted pair*, árnyékolatlan csavart érpár) kábelezésű hálózatnak nevezünk. Ez a technológia 10 Mbit/s, 100 Mbit/s, sőt 1 Gbit/s adatátviteli sebességet is lehetővé tesz.

Az UTP kábel 8 egymástól elszigetelt vezetéket használ, amelyek páronként össze vannak egymással csavarva. Az összecsavart vezetékeknek köszönhetően a kábel viszonylagos védettséget élvez a külső mágneses hatásokkal szemben, ezért a legtöbb esetben külön árnyékolásra nincs is szükség. Különlegesen erős változó mágneses térben – ipari környezetben – a kábel csavart volta nem jelent elegendő védelmet. Ilyenkor a drágább STP (*shielded twisted pair*, árnyékolt csavart érpár) vezetéket használjuk, amely nagyon hasonlít az UTP kábelre, de azzal ellentétben árnyékolta. Az UTP és STP bábelezésre épülő Ethernet hálózatok ma modernnek és elterjedtnek tekinthetők, a szakma előszeretettel használja őket helyi számítógép-hálózatok kialakítására.

Az EIA/TIA-568 szabvány a használható átviteli sebesség alapján a csavart érpáras vezetékeket csoportosítja. A legalacsonyabb *Category 1*, vagy röviden *Cat1* csoportba tartozó kábelek hangfrekvenciás jel átvitelére készültek (ezért nevezhetjük őket telefonrőtnak is), a magasabb kategóriájú vezetékek pedig magasabb frekvenciájú jel átvitelére is képesek. Számítógép-hálózatok kiépítésére a *Cat1* vezetékek természetesen nem alkalmasak.



4.3. ábra. Az RJ-45 csatlakozó vezetékeinek számozása (aljzat felől)



Az egyes csoportokba tartozó csavart érpáras vezetékek a felületes szemlélő számára egyformának tűnnek, de a nagyfrekvenciás viselkedésük nagyon különbözik, ezért a kábelgyártók a csoport jelzését minden esetben rányomtatják a kábelre. Mielőtt használni kezdenénk, mindig ellenőrizzük, hogy a kábel a megfelelő minősítésű-e. Soha ne elégedjünk meg azzal, hogy a számítógépek „látják egymást” ez ugyanis nem zárja ki azt, hogy az adatcsomagok jelenős része elveszik, és lassú vagy akadozó lesz a hálózati forgalom a nem megfelelő kábeltípus miatt.

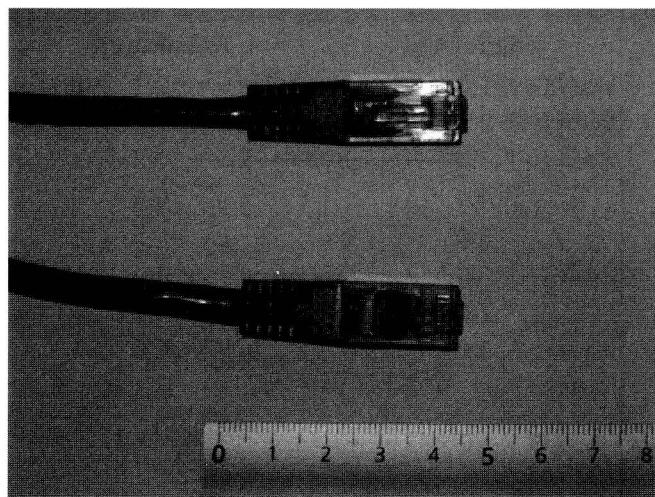
Csavart érpáras Ethernet hálózatokban ma még használjuk a 10BASE-T nevű változatot, amely 10 Mbit/s átviteli sebességet tesz lehetővé. Ez a változat Category 3 kábelezést tesz szükségesé, de ma már senki sem épít új hálózatot ilyen kábelezéssel.

A legelterjedtebb csavart érpáras Ethernet típus a 100BASE-T, amely 100 Mbit/s átviteli sebességet tesz lehetővé. Az ilyen hálózatokat általában Category 5 kábelezéssel építjük, így a legelterjedtebb kábelezési típus ma már a Category 5 árnyékolatlan, csavart érpáras kábelezés.

Szintén elérhető az 1 Gbit/s átviteli sebességet lehetővé tevő 1000BASE-T Ethernet típus, amelyet általában Category 5e vagy Category 6 kábelezéssel szereünk. Mivel maga a kábel olcsó ugyan, de a magas sebességet kihasználni képes számítógépek és egyéb aktív eszközök meglehetősen drágák, az ilyen eszközöket általában csak kiszolgálók, hálózatok összekötésére használjuk.

Az UTP és STP kábelek hosszát is korlátozza a szabvány, amely előírja, hogy két aktív eszköz között a vezeték hossza nem lehet nagyobb, mint 100 m. A fali vezeték legfeljebb 90 m, a kábel két végén található lengőkábel pedig összesen legfeljebb 10 méter hosszúságú lehet.

A csavart érpáras vezetékekhez az Ethernet hálózatokban általában RJ-45 (registered jack) csatlakozókat használunk. Ahogyan a kábelek, úgy a csatlakozók is csoportokba vannak osztva; nyilvánvaló, hogy a Category 5 kábelhez Category 5 RJ-45 csatlakozót kell használnunk. Fontos megjegyeznünk, hogy a kábeleket általában kétféle változatban, merevebb és olcsóbb fali, valamint hajlékonyabb



4.4. ábra. RJ-45 csatlakozóval szerelt kábel

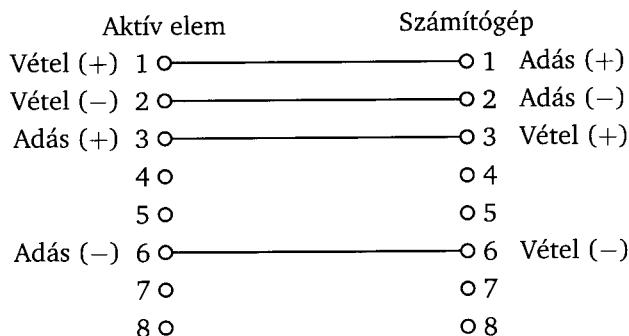
és drágább lengő kivitelben gyártják. Ügyelnünk kell, hogy a fali, illetve lengő kábelhez a megfelelő csatlakozót válasszuk, ellenkező esetben a használat során a csatlakozón belüli érintkezés meghibásodik, és bizonytalanságot okoz az adatátvitelben.



A számítógép-hálózatok kiépítésére használt RJ-45 csatlakozótípus nagyon hasonlít a telefoncsatlakozóra, sőt sok esetben a telephonhálózatot és az UTP számítógép-hálózatot közös kábelcsatornába, közös fali dobozba szerelik. Ez a legtöbb esetben nem jelenti azt, hogy a telefont és a számítógépet megcserélve mindenkorral működik, sokkal valószínűbb, hogy a csere után sem telefonálni, sem pedig internetezni nem lehet.

A 4.3. ábrán az RJ-45 csatlakozó vezetékeinek számozását, a 4.4 ábrán szerelt RJ-45 csatlakozót és UTP kábelt láthatunk. A fotón megfigyelhetjük, hogy az átlátszó csatlakozón át látni lehet az egyes érpárakat, a csatlakozó szerelését szemrevételezéssel ellenőrizni lehet. Ahhoz azonban, hogy megértsük, miképpen kell az egyes vezetékeket a csatlakozóba rendezni, meg kell vizsgálnunk a csavart érpáras hálózat felépítését.

A csavart érpáras Ethernet hálózat csillagpontos felépítésű, ami egyszerűen azt jelenti, hogy a számítógépeket egyenként kell egy központi eszközhez csatlakoztatnunk. Ennek az elrendezésnek a legnagyobb előnye, hogy a vezeték megsérülése vagy megbontása esetén csak egy számítógépen szűnik meg az összes hálózati szolgáltatás, azaz a felhasználók a saját számítógépük kiiktatásával csak saját maguknak árthatnak. Nincs tehát szükség életveszélyes fenyegéstekre a hálózat



4.5. ábra. A csavart érpáras Ethernet hálózat kábele

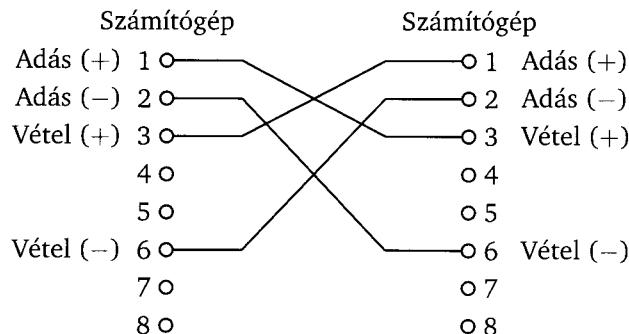
épségének megóvásához, így a rendszergazda nem feltétlenül szorul szociális el-szigeteltségebe. A csillagpontos elrendezés hátránya viszont az, hogy egy központi aktív eszköz is meg kell vásárolnunk a hálózat kiépítéséhez.

Egy esetben azonban megtakaríthatjuk UTP kábelezés esetén is a központi aktív elem árát: ha csak két számítógépet akarunk összekapcsolni. Két számítógép egyszerűen összekapcsolható egyetlen vezetékkel, nem kell központi aktív eszközt vásárolnunk. Ilyen esetben viszont különleges, úgynévezett keresztkábelt (*crossover cable*) kell használnunk. A keresztkábel csak a csatlakozók bekötésében különbözik az egyszerű UTP kábellel és RJ-45 csatlakozóval szerelt kábeltől.

Az 4.5. ábrán az aktív elemet a számítógéppel összekötő vezeték bekötését lát-hatjuk. Figyeljük meg, hogy az adásra és a vételre használt vezetékeket az aktív eszközben úgy rendezik el hogy a kábel egyenes vezetékelrendezés esetén éppen a megfelelő pontokat kapcsolja össze. Az RJ-45 csatlakozó és a legtöbb kábel nyolc eret tartalmaz, a kapcsolathoz azonban elegendő négyet felhasználni. Az ábrával kapcsolatban meg kell még jegyeznünk, hogy minden úgy kell kiválasztani az adás és vétel érpárokat, hogy azok +, illetve – sarkai egymással össze legyenek csavarva.



Érdemes megfigyelni a kábel bekötésénél, hogy az adásra és a vételre külön-külön elrendezést használunk a kábelben, így könnyen előfordulhat, hogy az adott kábel csak egy irányban működik, mert a másik érpár egyik vagy minden eret rosszul kötöttük be, esetleg egyik vagy másik érpár szakadt. Az olyan kábel, amelyik csak egy irányban működik, elég érdekes hibajelenségeket okozhat, amelyeket meglehetősen nehéz megtalálni, különösen azért, mert laikus gondolkodással nem számítunk rá, hogy a vezeték „csak az egyik irányba vezet”.



4.6. ábra. A keresztkábel bekötése

Egészen más elrendezést kell használnunk, ha két számítógépet kötünk össze csavart érpáras vezetékkel. Ilyen esetben a 4.6. ábrán látható keresztkábelbekötést kell használnunk. Figyeljük meg, hogy a kereszt alakú bekötés miatt az adásra, illetve vételre használt csatlakozási pontok éppen a megfelelő módon kapcsolódnak egymáshoz.

Amint láttuk, a csavart érpáras vezetékek bekötéséhez elegendő kiválasztani két összecsavart vezetéketet tartalmazó érpárat, és a bemutatott módon be kell kötnünk őket. Ha azonban tökéletes munkát akarunk végezni, minden a nyolc eret bekötjük, ráadásul éppen olyan színsorrendben, ahogyan azt a hagyomány előírja².

A kábel tehát 8 vezetékeret tartalmaz, amelyeket minden oldalon be akarunk kötni. Tudnunk kell, hogy a csatlakozó egyes pontjaihoz melyik vezetékeret kössük be. A legelterjedtebben használt UTP kábelek négy, egymással egyenértékű csavart érpárat tartalmaznak, amelyeket a szigetelés színe alapján tudunk egymástól megkülönböztetni. Mivel a négy érpár csereszabatos, több működő bekötési rendszer is elérhető, de általában ragaszkodunk a hagyományokhoz, mert így a kábel bekötése szemrevételezéssel könnyen ellenőrizhető.

A vezetékek azonosítására a telefontársaságok által kidolgozott színkódolást használjuk. Ez a színkódolás a fehér, vörös, fekete és sárga elsődleges színeket, valamint a kék, narancs, zöld, barna és pala másodlagos színeket használja. A színkódolás szabályai szerint az első ércsoportot az első alapszín (fehér), a második ércsoportot a második alapszín (vörös) jellemzi és így tovább.

Az ércsoportokon belül a vezetékeket a másodlagos színnel azonosítjuk. Az első ércsoport első vezetéke az a fehér vezeték, amelyik vékony kék (kereszttirányú vagy hosszanti) csíkokat tartalmaz, a második vezeték pedig az a kék vezeték, amely-

²A bekötetlen vezeték antennaként veszi a zavarjeleket, ráadásul a 1Gbit/s átvitel esetén minden a nyolc vezetéket használjuk (a lektor).

Ér	Szokásos	Választható
1	fehér/zöld	fehér/narancs
2	zöld/fehér (zöld)	narancs/fehér (narancs)
3	fehér/narancs	fehér/zöld
4	kék/fehér (kék)	kék/fehér (kék)
5	fehér/kék	fehér/kék
6	narancs/fehér (narancs)	zöld/fehér (zöld)
7	fehér/barna	fehér/barna
8	barna/fehér (barna)	barna/fehér (barna)

4.1. táblázat. Az RJ-45 csatlakozó bekötésének színkódolása

lyik vékony fehér csíkok tartalmaz. A további vezetékek a további másodlagos színeket használják, a harmadik vezeték fehér, narancsszínű sávval, a negyedik pedig narancsszínű, fehér sávval. Egyszerűsített írásmóddal azt mondhatjuk, hogy a vezetékek színe rendre fehér/kék, kék/fehér, fehér/narancs, narancs/fehér, fehér/zöld, zöld/fehér, fehér/barna, barna/fehér. Amint látható, a nyolc erű vezetékben csak a fehér főszínt használtuk, nem volt szükségünk a vörös, fekete és sárga alapszínekre, hogy a további ércsoportokat azonosítsuk.

A dolgot egy kicsit bonyolítja, hogy a gyártók sokszor kissé takarékoskodnak a festékkel, amikor az ilyen kevés érpárkból álló kábeleket gyártják, és a vékony fehér sávot egyszerűen elhagyják. Így a színek sorrendje fehér/kék, kék, fehér/narancs, narancs, fehér/zöld, zöld, fehér/barna, barna. Amint látható, a vezetékek azonosítása így is egyértelmű, hiszen csak egy főszínt használtunk.

Az RJ-45 csatlakozó színsorendjét a 4.1. táblázat mutatja be, ahol mindenjárt két-féle elfogadott elrendezést is találunk. A szakirodalom szerint Európában a szokásos, míg az Egyesült Államokban a választható elrendezés terjedt el, de a tapasztalat azt mutatja, hogy a kiterjedt nemzetközi kereskedelelmek köszönhetően hazánkban minden két színséma megtalálható.

Az Ethernet hálózatok működése

Az Ethernet csomagkapcsolt adatátvitel megvalósítására használható, azaz minden Ethernet hálózat adatcsomagok küldésére és fogadására alkalmas eszközöket tartalmaz. Az adatcsomagok méretét az Ethernet szabványcsalád az MTU (*maximal transfer unit*, legnagyobb adatátviteli egység) bevezetésével korlátozza. Az MTU értéke a legtöbb Ethernet hálózati eszköz esetében mintegy 1500 bájt, de az egyes gyakorlati megvalósítások esetében ettől különbözik.

Ha két számítógép között az Ethernet hálózaton az MTU értékénél nagyobb adatmennyiséget akarunk átvinni, azt csak a megfelelő méretű csomagokra darabolás után tudjuk megenni. Az Ethernet nem ad választ arra, hogy miképpen kell a csomagokat kialakítanunk, és hogyan állítható vissza a fogadó oldalon az eredeti információ a csomagok egymáshoz illesztésével, csak azt írja le, hogyan lehet a csomagokat egyik pontról a másikig eljuttatni.



4.7. ábra. Az Ethernet cím jelzése

Amint láttuk, a vékony Ethernet hálózaton a számítógépek elektromos szempontról párhuzamosan kapcsoltak. Ez azt eredményezi, hogy a vezetékre küldött adatcsomag minden, a vezeték által kialakított hálózatra kapcsolt számítógéphez eljut. Azt szokás mondani, hogy az Ethernet hálózat alapjában véve rádió üzemmódú, azaz a hálózatra küldött adatok a hálózat minden tagjához eljutnak, ahogyan a rádióadás is eljut minden vevőkészülékhez. (Ez természetesen nem jelenti azt, hogy nem készíthetünk olyan hálózatot, amely egymástól elválasztott részekre.)

Az Ethernet hálózatok csomagkapcsoltak, az adatokat csomagokra kell bontanunk, így, habár az adatátviteli közeget több számítógép is használhatja, egy időben több adatátvitel is folyhat. A csomagok a hálózatra egymás után kerülhetnek, így több adatátvitel fér meg egymás mellett.

Az Ethernet szabványnak így biztosítania kell, hogy a hálózaton található számítógépek minden tudják, hogy az adott csomag „nekit szól”, a csomaggal „foglalkozniuk kell”. A szabvány ezt az Ethernet cím (*Ethernet address*) segítségével oldja meg. Az Ethernet címet szokás MAC címként (*media access control*, adatátvitőközeg-vezérlő) is emlegetni. A MAC cím az Ethernet címnél kissé tárgabb jelentésű fogalom, úgy is fogalmazhatnánk, hogy az Ethernet hálózatok esetében az Ethernet cím tölti be a MAC cím szerepét. Fontos azonban, hogy megjegyezzük a MAC rövidítést is, mert sok program használja az üzeneteiben.

Az Ethernet cím szabvány szerint 6 bajt hosszúságú tetszőleges szám, amelyet a hagyomány szerint tizenhatos számrendszerben, kettőspontokkal tagoltan írunk (például 00:00:c0:e6:c9:9d). A címzési rendszer működéséhez elengedhetetlenül fontos, hogy a címek egyediek legyenek, így minden hálózati eszköz saját, beépített címmel rendelkezik.

A szabvány értelmében minden hálózati eszköz elhelyezi a csomag küldése előtt a csomag elején a saját címét és annak a hálózati eszköznek a címét is, amelynek a csomagot küldi. Beszélhetünk tehát küldő és címzett Ethernet címről, amelyeknek minden Ethernet adatcsomagban megtalálhatónak kell lenniük.

A hálózati eszközök automatikusan figyelik a beérkező csomagok címét, és minden csomag esetében összehasonlítják a saját címükkel. Ha a beérkezett csomag címzettjeként magukra ismernek, azaz, ha a beérkezett csomag címzettjének címe megegyezik a saját címükkel, a csomagot a számítógép felé továbbítják, más esetekben pedig figyelmen kívül hagyják. A másoknak szóló csomagok figyelmen kívül hagyása igen hasznos, hiszen így csak a számítógép számára érkező csomagokat kell feldolgozni, a többi csomag nem terheli a számítógépet.

Az Ethernet szabvány előírja azt is, hogy a különleges **ff : ff : ff : ff : ff : ff** címzettel ellátott csomagokat minden számítógépnek fel kell dolgoznia. Az ilyen csomagokkal mindenki számára fontos körüzenet (broadcast) küldhető, amelynek – ahogyan majd látni fogjuk – komoly szerepe van a kapcsolattartás megszervezésében. Fontos azonban, hogy a hálózaton minél ritkábban éljünk a körüzenet adta lehetőséggel, hiszen – amint láttuk – a körüzenet az összes számítógépet terhel.

Említettük, hogy az Ethernet szabvány szerint igen fontos, hogy minden hálózati eszköz egyedi Ethernet címmel rendelkezzen. A hálózati eszközöket gyártó vállalatok garantálják a hálózati eszközökben beállított címek egyediségét. Az Ethernet címek első három bájtja a gyártók számára az IEEE által kiosztott gyártókód, a második három bájtja pedig a gyártók által kiosztott azonosító. Az Ethernet csomagok címének első három bájtjából tehát könnyedén kideríthető, ki gyártotta az eszközt.

Az tehát, hogy az adatcsomagok minden számítógéphez eljutnak, az Ethernet címnek köszönhetően nem okoz problémát. A hálózati csatolók csak azokat az adatcsomagokat továbbítják a számítógép felé, amelyek nekik szólnak. Felmerül azonban egy másik probléma, mégpedig az, hogy a közös vezetéken egyszerre csak egy számítógép tud üzenetet küldeni. Ha több számítógép egy időben kísérel meg csomagot küldeni, nem lehet érteni, hogy „ki mit mond”. Ezt a problémát az Ethernet szabvány az ütközésvizsgállal oldja meg.

Ütközésről (*collision*) akkor beszélünk, amikor egy időben több hálózati csatoló kísérel meg üzenetet küldeni. minden Ethernet csatolót felszerelnek ütközésérzékelő áramkörrel, amely folyamatosan figyeli a hálózatot. Ha a csatoló üzenetet küld, és a hálózatot figyelve azt tapasztalja, hogy a küldött csomag nem olvasható vissza – mert valaki „belebeszélt” –, a csomag küldését azonnal megszakítja, és késsőbb automatikusan újra megismétli. Mivel minden hálózati csatoló véletlenszerű ideig várakozik a csomag megismétlése előtt, viszonylag kicsiny az esélye annak, hogy újra ütközés történik.

Az ütközések azonban komoly gondot okozhatnak, ha a hálózat túlterhelt. Minél több számítógép akar egy időben üzenetet küldeni, annál nagyobb a valósínlisége annak, hogy ütközés történik. Ez végső soron azt eredményezi, hogy a hálózat nem képes az elméleti átviteli sávszélességet teljesíteni. Minél jobban megközelítjük a lehetséges legnagyobb átviteli sávszélességet, annál magasabb lesz az ütközések száma. A túlterhelt hálózaton az ütközések száma olyan magas lehet, hogy a hálózat a sávszélességének csak töredékét képes teljesíteni, a hálózat „eldugul”.



Kisméretű hálózaton azonban megközelíthetjük az elméleti sávszélességet. Ha a hálózaton csak egyetlen számítógép küld adatokat, akkor nyilánvalóan nem következik be ütközés akkor sem, ha a csomagok gyorsan követik egymást. Nagyméretű hálózatoknál sokszor tapasztalhatunk olyan rövid időszakokat, amelyeket egy-egy számítógép erőteljes adatforgalma jelez.

Az Ethernet címzési rendszerének megismerése után könnyen megérthetjük, hogy milyen aktív elemeket szokás elhelyezni a csillagpontos kiépítésű csavart érpáras Ethernet hálózatok központjaiban. Ezeket az aktív elemeket két csoporthoz osztjuk.

Az egyszerűbb aktív elemek neve elosztó (*hub*). Az elosztó egyszerű jelerősítő áramkör, amely a beérkezett elektronikus jelcsomagokat erősíti, újraformázza, és minden lehetséges irányba továbbküldi. Az elosztó jól érhetően és hangosan szét-kiabálja minden vezetéken azt, amit az egyik vezetéken hall. Nyilvánvaló, hogy egy olyan csillagpontos hálózaton, ahol egy elosztó található a középpontban, egyszerre csak egy adatsomag haladhat, az viszont minden számítógéphez eljut.

Az összetettebb aktív elemek neve kapcsoló (*switch*). Ezek az eszközök már jóval intelligensebbek, képesek az Ethernet címek alapján a csomagot abba az irányba továbbküldeni, amelyik irányban a címzett található. Az olyan csillagpontos hálózatok esetében, amelyek középpontjában kapcsoló található, egyszerre több adatsomag is haladhat, hiszen a hálózat középpontjában „telefonközpontként” működő kapcsoló gondoskodik arról, hogy a csomagok a megfelelő irányba haladják.



Felmerülhet a kérdés, hogy honnan tudja a kapcsoló, hogy a címzett melyik irányban található. A válasz egyszerű: megjegyzi a küldő Ethernet címét, és ha legközelebb ő kap csomagot, már tudni fogja. Ha pedig a címzett nem szerepel a nyilvántartásában, egyszerűen elküldi a csomagot minden irányba. Ha a postai levelek kézbesítése is ilyen módon történne, nem kellene utcaneveket használnunk. Felmerül viszont az ismeretlen címzettű levelek néhány millió példányban való sokszorosításának problémája...

Ma már csak a nagyon kis méretű – néhány számítógépet összekötő – csavart érpáras Ethernet hálózatokban használunk elosztót az összekapcsolásra, a piacon pedig nagyon sokféle, különböző árú és minőségű kapcsoló található. Léteznek olyan kapcsolók, amelyeket akár otthoni felhasználásra is megvásárolhatunk, de nem ritka a brutális árú professzionális eszköz sem, amely nagysebességű átvitelt, nagy megbízhatóságot és rugalmas beállítási lehetőségeket biztosít.

Az IP protokoll

Amint láttuk, az Ethernet címet a gyártók helyezik el az Ethernet eszközökben, mégpedig központi irányítás mellett, hogy egyetlen Ethernet eszköz se kaphassa egy már létező másik eszköz címét. Az Ethernet címek kiosztásának módszere elmés és hatásos, de nem teszi lehetővé világméretű hálózat kialakítását.

Az Ethernet eszközök címét a gyártósor végén helyezik az eszközökbe, majd az eszközöket kamionokra, vasútra és hajókra rakják, majd széthordják a világ minden pontjára. Ha nagyobb tételt vásárolunk az eszközökből, valószínűleg egymáshoz közeli Ethernet címeket találunk, de még így sem lehet fogalmunk arról, hogy a világ mely pontjára került a következő Ethernet cím, amelyik hiányzik a gyűjteményünkben.

Megtehetnénk persze, hogy a birtokunkba jutott Ethernet eszközök címét megváltoztatjuk, de ekkor már nem lehetnénk biztosak abban, hogy az általunk használt új cím az egész világon egyedi, és különben is az egész világot körbe kellene telefonálnunk, hogy minden rendszergazdának elmeséljük, milyen címen érhetik el a vadonatúj hálózati eszközünket³.

A világmeretű hálózatok kialakítására is alkalmas címzési rendszer jól bevált módszereit és eszközeit írja le az IP (*internet protocol, internetprotokoll*)[127], amelyet az Interneten kötelező jelleggel használunk. Az IP leírja, hogy milyen címzési rendszerrel lehet az Interneten az egyes számítógépeket azonosítani.

Valójában jelenleg az IP protokollnak két változata is érvényben van. Az IPv4 (*internet protocol version 4*) az IP nagyon sikeres és elterjedt változata, az IPv6 (*internet protocol version 6.*)[34] pedig az újabb, de nehezen terjedő változata. Mivel az IPv6 nem túl elterjedt napjainkban, az IP protokoll alatt általában az IPv4 változatot értjük. És a változat számát csak akkor tesszük hozzá, ha kimondottan az IPv6-ról beszélünk. Habár napjainkra az IPv6 is rendelkezik szakirodalommal[56] és alkalmazásokkal, a legtöbb szakember ma is az IPv4 használatát javasolja[62, 63, 85].



Az IP-t alapprotokollnak is nevezhetjük, ami talán kissé idegen az elterjedt szaknyelvtől, de segít, hogy megkülönböztessük ezt a protokollt azoktól az alkalmazásprotokolloktól (például FTP, HTTP, SMTP stb.), amelyek szintén a hálózaton való kapcsolattartás mikéntjét írják le, de az IP-re épülve, mintegy magasabb szinten.

Magát az IP protokollt szinte sohasem használjuk önmagában, ezért általában hasonló protokollokkal (TCP, UDP, ICMP) együtt emlegetjük, együtt tárgyaljuk. A szakirodalom is együtt mutatja be ezeket, ami megmagyarázza, hogy miért szokás általában TCP/IP protokorról beszélni. Mivel remélhetőleg könnyebb részenként megérteni a hálózati eszközök alapvető működését, előbb csak az IP protokollal foglalkozunk, és az egyéb protokollokat csak később mutatjuk be.

Amint láttuk, az IP protokoll fontos eleme az IP cím, amely világmeretű hálózat felépítését is lehetővé teszi. Az IP cím egy négybájtos egyedi azonosító, amely a hálózati eszközt egyértelműen azonosítja. A négy bájtot a szokás szerint tízes számrendszerben, a bájtok között egy-egy ponttal írjuk, például 10.1.1.1 vagy 127.0.0.1. A négy bájton ábrázolható legkisebb szám IP címként leírva 0.0.0.0, a legnagyobb pedig 255.255.255.255, bár – amint látni fogjuk – ezek az értékek nem használhatók címként.



Az akciófilmekben a képernyőn ötbájtos IPv4 címeket vagy 255-nél nagyobbat használnak, hogy elejét vegyék a tömeghisztériának és a jogi problémáknak. Érdekes ez, hiszen azt sugallja, hogy az akciófilmek készítői értenek az IPv4 szabványhoz, és figyelembe is veszik, ami ellentében áll oly sok megfigyeléssel.

³Valójában csak szegmensen belül van szükség egyedi azonosítóra (a lektor).

Most vizsgáljuk meg, miképpen lehet egy címről kideríteni, hogy a világ mely részében, melyik hálózaton található!

Emlékezzünk vissza, hogyan osztottuk fel az Ethernet címeket két részre, hogy a címek egyes tartományait szétosztassuk a gyártók közt! Az IP címek esetében is ezt a jól bevált módszert követjük, azaz felosztjuk a címet két részre. A cím elején található számok jelzik, hogy melyik hálózatról van szó, a cím vége pedig az adott hálózati eszközt jelöli ki a hálózaton belül. Az Ethernet címekhez képest különbség, hogy nem minden középen választjuk szét a címet, mivel léteznek kissébb és nagyobb hálózatok is. Nyilvánvaló, hogy minél kevesebb helyet hagyunk a hálózati címnek, annál kevesebb hálózatot használhatunk, de azokban annál több hálózati eszközt helyezhetünk el. Ha kisebb helyet tartunk fenn a cím végén az eszközök számára, és nagyobb helyet biztosítunk a hálózatnak, akkor több, de kisebb hálózatot hozhatunk létre. Az IP protokoll szerint kisebb és nagyobb hálózatokat is képesek vagyunk kezelní, mert bizonyos címekben nagyobb, más címekben pedig kisebb helyet tartunk fenn hálózati címként.

Eredetileg, amikor még úgy tűnt, hogy jóval több címet alkothatunk, mint ahány hálózati eszköz valaha is lesz, a felosztást a következő szabályok szerint végezték el:

- Azokat az IP címeket, amelyek első bitje 0, „A” osztályú IP címeknek nevezték.

Ebben a tartományban az első 8 bit azonosítja a hálózatot (de az első bit minden 0 értékű!) és az utolsó 24 bit a hálózaton belüli eszközt. Az ilyen hálózatok több mint 16 millió hálózati eszközt tartalmazhatnak.

- Azokat az IP címeket, amelyek első két bitje 10, a „B” osztályba sorolták. (Nyilvánvaló, hogy egyetlen egy „B” osztályú cím sem „A” osztályú, hiszen az első bit értéke nem 0!)

Ebben a tartományban az első 16 bit azonosítja a hálózatot (de az első két bit minden 10) és az utolsó 16 bit a hálózaton belüli eszközt. Az ilyen IP címeket használó hálózatok több mint 65 ezer hálózati eszközt tartalmazhatnak.

- Azok az IP címek, amelyek első 3 bitje 110, „C” osztályú IP címek. (Megint csak egyértelmű, hogy egyetlen „C” osztályú cím sem lehet „B”, esetleg „A” osztályú is egyben.)

Ebben a tartományban az első 24 bit azonosítja a hálózatot (de az első 3 bit minden 110) és az utolsó 8 bit a hálózati eszközt. Az ilyen IP címeket tartalmazó hálózatok pontosan 254 darab hálózati eszközt tartalmazhatnak. (A 0 és a 255 nem használható címek, ahogyan azt majd látni fogjuk.)

- Az 1110 kezdetűek lettek az úgynevezett „D” osztályú, többeknek szóló üzenetek küldésére használható címek (*multicast address*).

- Az 1111 kezdetű címeket különleges célokra tartották fenn.

Első számjegy	Elnevezés	Hálózat/eszköz bitek
0 – 128	„A” osztály	8/24
128 – 191	„B” osztály	16/16
192 – 223	„C” osztály	24/8
224 – 239	„D” osztály	—
239 – 255	„E” osztály	—

4.2. táblázat. Az IP címek osztályai

E szerint a felosztás szerint tehát az IP cím elejéről eldönthető, hogy a cím melyik tartományba tartozik, azaz hány bitet tartottunk fenn a hálózat és hány bitet az eszközök azonosítására. A 4.2. táblázatban láthatjuk összefoglalva, hogyan állapítható meg az IP cím első bájtjáról – az első pont előtti számról –, hogy milyen osztályú a címtartomány.



Az IP címeknek a fentiekben bemutatott felosztását hagyománynak tekinthetjük, amelytől ma már sokszor eltérünk, és olyan hálózatokat is használunk, amelyek nem a bemutatott szabályok szerint osztják fel az IP címet hálózati és eszközcímre.

Minden hálózatban – álljon az „A”, „B” vagy „C” osztályú címekből – foglaltnak tekintjük a lehető legkisebb és a lehető legnagyobb eszközcímet a következő célokra:

- A lehető legkisebb eszközcímet a hálózat jelölésére használjuk, azt nem osztjuk ki egyetlen valódi hálózati eszköz számára sem.
- A lehető legnagyobb címet a körüzenetek – a hálózat minden tagjának elküldött üzenetek – számára foglaljuk le, azt nem osztjuk ki egyetlen eszköz számára sem.

Ha a két foglalt címet is figyelembe vesszük, könnyedén kiszámíthatjuk, hogy egy adott címet magába foglaló hálózat hány hálózati eszköz befogadására szolgál a hagyomány szerint. Ezt mutatja be a következő példa.

20. példa. Azt látjuk, hogy egy számítógép IP címe 172.16.12.99. Felmerül benne a kérdés, hogy az adott hálózatban hány eszköz számára adhatunk IP címet. A válasz egyszerű, ha a helyi rendszergazda betartotta a hagyományt!

A 4.2. táblázat alapján ez a cím egy „B” osztályú címeket tartalmazó tartomány tagja, hiszen az első számjegy 128 és 191 közé esik. Ebben a hálózatban tehát 16 bitet használunk a hálózat és 16 bitet az eszközök azonosítására.

A hálózatban tehát összesen 2^4 féle eszközcím képzelhető el, melyek közül a legkisebb (172.16.0.0) és a legnagyobb (172.16.255.255) nem használható. Összesen tehát 65534 hálózati eszköz számára van hely.

IP címek	Tartományok
10.0.0.0	„A” osztály
172.16.0.0 – 172.31.0.0	„B” osztály
192.168.0.0 – 192.168.255.0	„C” osztály

4.3. táblázat. Magáncélra használható IP címtartományok

A figyelmes olvasónak bizonyára feltűnt, hogy a példában konkrét IP címek találhatók, így a szerző nyilvánvalóan beleesett a hibába, amelynek kapcsán néhány sorral feljebb az akciófilmek készítőin élcelődött. Valójában azonban erről szó sincs, hiszen ezek magáncélra használható IP címek.

A 4.3. táblázatban látható IP címtartományokat bárki tetszőleges formában felhasználhatja a saját hálózatán, de – az ütközések elkerülésének érdekében – az Interneten ezeket a vonatkozó szabvány[138] alapján nem szabad használni. Nyilvánvaló, hogy éppen az teszi lehetővé a magáncélra való használatot, hogy az Interneten való használatról le kellett mondanunk.

Az IP címek „A”, „B” és „C” osztályú címtartományokba való sorolása roppant kényelmes és világos módszer, amelyet ma is használunk. Van azonban egy probléma az osztályokba sorolással, amelyre ki kell térnünk.

Tegyük fel, hogy egy vállalatnál 248 hálózati eszközt üzemeltetnek, de nem egy, hanem két hálózatban. Az egyszerűség érdekében tételezzük fel, hogy a két hálózat egyforma méretű, 124–124 hálózati eszközt tartalmaz. Hogyan tudunk a számukra IP címeket biztosítani?

Nyilvánvaló módon adódik a válasz: két „C” osztályú IP címtartományra van szükség. Ha minden hálózat számára lefoglalunk egy „C” osztályú tartományt, minden hálózati eszköznek adhatunk címet, sőt még marad is néhány cím tartalékba.

A problémát éppen a „tartalékba” maradt címek okozzák! Ha a két 124 számítógépet magába foglaló hálózat számára két „C” osztályú tartományt foglalunk le, a két tartományban éppen 254×2 , azaz 508 használható cím lesz. Ez azt jelenti, hogy $508 - 248$, azaz 260 olyan címünk lesz, amelyet nem használunk semmire.

Ahogyan nőtt az Internet mérete, ez a „pazarlás” egyre kevésbé volt elfogadható. Egyre kevesebb lett a kiosztható IP címek száma, egyre nehezebb volt „C” osztályú címtartományhoz jutni. Úgy tűnik, hogy az IPv4 megalkotásakor a szakemberek kissé alábecsülték az Internet fejlődésének ütemét, így kissé kicsire választották az IP címek méretét.

A bemutatott példában megoldást jelenthet a „C” osztályú címtartomány kettéosztása. Ha a hálózati cím számára 25 bitet, a hálózaton belüli gépazonosításra pedig 7 bitet használunk, olyan hálózatot nyerhetünk, amelyben $2^7 = 128$ cím található. Így, a legkisebb és a legnagyobb cím elhagyásával 126 számítógép befordulására alkalmas hálózat építhető. A „C” osztályú címtartomány fele is elegendő tehát a példában bemutatott egy-egy alhálózat üzemeltetésére. A rendszergazdá-

nak azért előnyös ez a megoldás, mert elegendő egy „C” tartomány a munkájához, nem kell kettőt igényelnie.

Felmerül még a kérdés, hogy honnan tudják a számítógépek, hogy hol végződik a cím hálózati része, és hol kezdődik a hálózaton belüli cím. Ha lehetővé tesszük, hogy ne csak az első, a második vagy a harmadik bájt végén végződjön a cím hálózati része, a számítógépeknek olyan eszközt kell biztosítanunk, amely a cím osztályától függően pontosan megadja, hogy hol húzódik a határ. Erre a feladatra ma az alhálózati maszkot használjuk. Az alhálózati maszkot minden egyes számítógépen az IP cím mellett tároljuk. Az alhálózati maszk éppen úgy 32 bitből áll, ahogyan az IP cím: azokon a helyeken, ahol az IP címben a hálózati cím található, 1 az értéke, azon a helyen pedig, ahol a hálózaton belüli cím, 0. A következő példa az alhálózati maszk használatát mutatja be.

21. példa. Szeretnénk a 192.168.1.0 címtartományban egy 60 számítógépet befogadó hálózatot telepíteni úgy, hogy a lehetőségekhez márnen minél kevesebb cím menjen veszendőbe. Számítsuk ki, milyen beállításokkal tudjuk ezt megszervezni!

Nyilvánvaló, hogy a legkisebb hálózaton belüli címméret, amely megfelel a céljainknak, a 6, mivel $2^6 = 64$. Az alhálózati maszkot tehát úgy kell megalkotnunk, hogy benne 26 bitnyi 1 (hálózatcím) és 6 bitnyi 0 (hálózaton belüli cím) legyen. Ezt írhatjuk bináris és decimális formában is:

$$1111111.1111111.1111111.11000000 = 255.255.255.192$$

Ezzel az alhálózati maszkkal az eredeti „C” osztályú címtartományt négy részre osztottuk. Az eredeti címtartományunk legkisebb és legnagyobb lehetséges érteke a következőképpen alakult:

$$11000000.10101000.00000001.00000000 = 192.168.1.0$$

$$11000000.10101000.00000001.1111111 = 192.168.1.255$$

Az új alhálózati maszk esetében ez most négy tartományt jelent, amelyek legnagyobb és legkisebb címértékei a következőképpen alakulnak.

Első címtartomány:

$$11000000.10101000.00000001.00000000 = 192.168.1.0$$

$$11000000.10101000.00000001.00111111 = 192.168.1.63$$

Második címtartomány:

$$11000000.10101000.00000001.01000000 = 192.168.1.64$$

$$11000000.10101000.00000001.01111111 = 192.168.1.127$$

Harmadik címtartomány:

$$11000000.10101000.00000001.10000000 = 192.168.1.128$$

$$11000000.10101000.00000001.10111111 = 192.168.1.191$$

Negyedik címtartomány:

11000000.10101000.00000001.11000000 = 192.168.1.192

11000000.10101000.00000001.11111111 = 192.168.1.255

Nem szabad természetesen elfelejtenünk azt, hogy a hálózatban található legnagyobb, illetve legkisebb címeket nem szabad számítógépeknek kiosztani, mert a legkisebb címet a hálózat jelzésére, a legnagyobb címet pedig a hálózaton belüli körüzenetek számára tartjuk fenn.

Az alhálózati maszk ábrázolására két jelölés terjedt el. A hosszabb jelölésmód szerint az alhálózati maszkot ugyanúgy tízes számrendszerben írjuk, mint az IP címet, a rövidebb jelölés szerint pedig egyszerűen leírjuk, hány bitet tartunk fenn a hálózat címnek a címen belül. Az alhálózati maszkot – mindenkét jelölési formában – szokás egy / jellel a számítógép, illetve a hálózat címe után írni. Ezt mutatja be a következő példa.

22. példa. Írjuk fel az előző példa harmadik címtartományának jelölését az alhálózati maszkkal együtt!

Az alhálózati maszk értéke 255.255.255.192 volt, ami rövidített jelöléssel 26. A harmadik alhálózat cím/maszk formában leírva:

192.168.1.128/255.255.255.192

192.168.1.128/26

Hasonló formában szokás a számítógépek címét is jelölni.

4.1.2. A hálózati kártya beállításai

A következő oldalakon a számítógépeken használatos áramkörök rövidítéseit néhány, a hálózat felépítésében fontos dolgot. Ezeket az ismereteket más művek részletesebben is tárgyalják[161, 160], mi csak a legsükségesebb információkat vesszük sorra linuxos szemmel.

A hálózati csatolókártya (NIC, *network interface card*) – közismertebb nevén hálózati kártya – feladata a számítógép illesztése a hálózathoz, a csomagok küldése a hálózat más számítógépei felé, és a csomagok fogadása a hálózatról. Az IBM PC-k számára rengeteg hálózati csatolót készítettek a különböző gyártók, így a csatolók világában eligazodni egyáltalán nem könnyű. A következő oldalakon olyan ismereteket találunk, amelyek segítenek az egyes hálózati csatolók kiválasztásában és telepítésében, illetve segítenek abban, hogy a hálózati csatolók telepítését elvégezzük.

A széles termékskálán segít eligazodni, ha tudjuk, hogy a csatolókat és a csatolókon található integrált áramköröket gyakran nem ugyanaz a gyártó készíti. A világon nagyon sok vállalat van, amelyek integrált áramköröket és hálózati csatolókat

is gyártanak, de nagyon sok olyan vállalat is létezik, amelyek csak hálózati csatolókat készítenek a más gyártók által forgalmazott integrált áramkörök ből. Ezt igen fontos szem előtt tartanunk, hiszen lehetséges, hogy egy ismeretlen gyártó által készített hálózati csatolóhoz könnyedén megtaláljuk a megfelelő meghajtóprogramot, ha az valamelyik ismert és elterjedt integrált áramkörre épül.

A hálózati csatoló telepítéskor a tapasztalt rendszergazdát általában nem az érdeklő, hogy „sárga villám a tavaszi naplementében” vagy „kecsesen ringó fecske” a hálózati csatoló fantázianeve, hanem az, hogy milyen integrált áramkör köré építették. Ha az integrált áramkör típusához megtaláljuk a megfelelő meghajtóprogramot, reménykedhetünk, hogy a hálózati csatoló telepítését siker koronázza.

Fontos tudnunk azt is, hogy az integrált áramköröket készítő vállalatok termékeiket folyamatosan fejlesztik, a fejlesztés újabb és újabb eredményeinek megfelelően újabb és újabb változatban kiadják. Az új áramkörök típusmegnevezése általában hasonlít az eredetihez, de attól eltér. A névben található eltérés csekély lehet – például egy a név után írt B vagy C betű –, ez azonban nem jelenti azt, hogy az áramkör ugyanazzal a meghajtóprogrammal használható! Ha hálózati csatolót vásárolunk, mindenkiéppen vizsgáljuk meg az integrált áramkör pontos nevét, mert előfordulhat, hogy egyetlen betű jelzi csak, hogy melyik csatolót használhatjuk és melyiket nem!

Azt is fontos tudnunk, hogy a különféle gyártók különféle csatolói között igen nagy minőségi különbségek vannak. Sokan úgy gondolják, hogy mivel a hálózati csatolók egyazon hálózaton képesek működni, képesek egymással tartani a kapcsolatot, nyilvánvalóan egyformák. Ez egyáltalán nem így van! A különféle csatolók közti árkülönbség akár hússzoros is lehet, ami jelzi, hogy különféle minőségű termékekről van szó⁴. Otthoni használatra, munkaállomás hálózathoz csatolására általában megteszí egy olcsóbb hálózati csatoló is, kiszolgálóba, komoly feladatra azonban minőségi terméket kell vennünk, és általában mélyen a zsebünkbe kell nyúlnunk. Ha nem áll rendelkezésünkre elegendő tapasztalat a témaban, mindenkiéppen érdemes tanácsot kérnünk olyan szakembertől, aki munkájánál fogva naprakész tapasztalatokkal rendelkezik a hálózati csatolók gyorsan változó világában⁵.

A hálózati csatolók adatátvitelt valósítanak meg a külvilág és a számítógép között. A számítógépek a hálózati kártyákat perifériaként kezelik. A perifériák kezelésének megértéséhez a következő fogalmakat kell ismernünk:

I/O cím A számítógép a perifériákat az I/O (*input/output*, bemenet/kimenet) címük alapján különbözteti meg. Egy perifériához általában több I/O cím is tartozik – a periférián belül való eligazodáshoz –, egy címet azonban csak egy periféria használhat.

A gyakorlatban a legtöbbször egy-egy címtartományt rendelünk az egyes perifériákhoz, és a meghajtóprogramok (*device driver*, illesztő program, meg-

⁴Általában... Nyilvánvalóan az is elképzelhető, hogy egy hálózati csatolót arcátlanul magas áron kínálnak eladásra.

⁵És természetesen nem hálózati csatolók arcátlanul magas áron való eladásából él.

hajtóprogram) a címtartományon belül az egyes címekhez tartozó jelentést ismerve vezérlik a perifériát.

IRQ Az IRQ (*interrupt request*, megszakítás kérése) rendszeren keresztül kérnek kiszolgálást a számítógéptől.

A különféle perifériák más-más IRQ számmal rendelkeznek, hogy a számítógép el tudja dönten, melyik perifériát kell kiszolgálni.

DMA A DMA (*direct memory access*, közvetlen memória hozzáférés) rendszer segítségével az adatok a periféria és a számítógép (operatív memória) között gyorsan mozgathatók. Az egyes perifériák számára néha előre lefoglalunk egy-egy DMA csatornát, hogy a DMA használatának lehetősége sürgős esetben elérhető legyen.

A hálózati csatolók többféle I/O cím, IRQ szám és DMA csatornaszám beállítása mellett is képesek működni, hogy a perifériák ne akadályozzák egymást a munkában. A régebbi rendszereken általában a rendszergazda feladata volt a megfelelő értékek beállítása, ma azonban ez már többé-kevésbé automatikusan történik.

GNU/Linux rendszereken az érvényben lévő IRQ, DMA és I/O beállításokat a /proc/ látszólagos állományrendszerben található állományokból olvashatjuk ki. Ezt mutatja be a következő példa.

23. példa. Meg szeretnénk ismerni a számítógépen érvényben lévő I/O és IRQ beállításokat. Olvassuk ki az adatokat a /proc/ioports és /proc/interrupts állományokból (a rövidség érdekében néhány sort törölünk a példából):

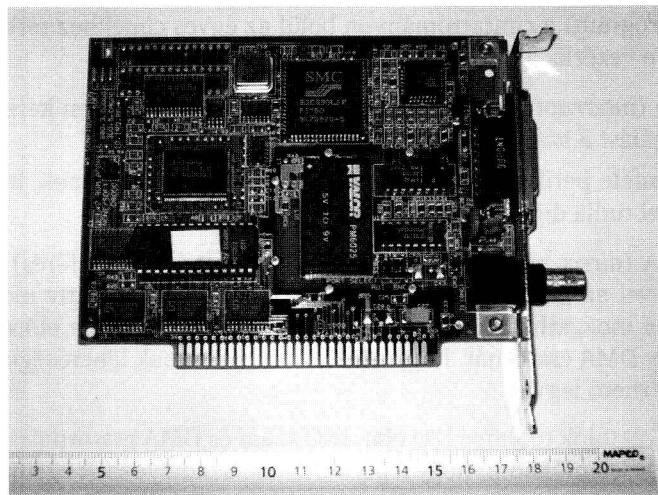
```
$ cat /proc/ioports
0000-001f : dma1
0020-0021 : pic1
0040-005f : timer
0060-006f : keyboard
$ cat /proc/interrupts
          CPU0
0:    11649370      XT-PIC  timer
1:      26320       XT-PIC  i8042
2:          0       XT-PIC  cascade
$
```

Hasonlóképpen használható a /proc/dma látszólagos állomány, amelyben a DMA beállításokról olvashatunk.

Az ISA csatolófelület

A hálózati csatolók legtöbbje utólag is beépíthető a számítógépbe, ezért valami-lyen kártyaként valósítják meg őket.

Néhány évvel ezelőtt a hálózati csatolók az ISA (*industrial standard architecture*, ipari szabványos felépítés) csatolófelületet használták. A 4.8. ábrán egy 8 bites ISA hálózati csatolót láthatunk.



4.8. ábra. ISA hálózati csatoló

Ezek a hálózati kártyák – és maga az ISA csatolófelület – ma már elavultnak tekinthető, de azért még találkozhatunk velük régebbi számítógépekben. Néhány dolgot érdemes tudnunk az ISA csatolóval ellátott hálózati kártyákról:

- A modern számítógépek általában nem rendelkeznek ISA csatolóval, ezért ezeket a csatolókat általában nem lehet modern számítógépbe építeni.

- Az ISA csatolófelület viszonylag alacsony sebességű adatátvitelt tesz lehetővé, ami korlátozza a hálózat használatának sebességét, sávszélességét.

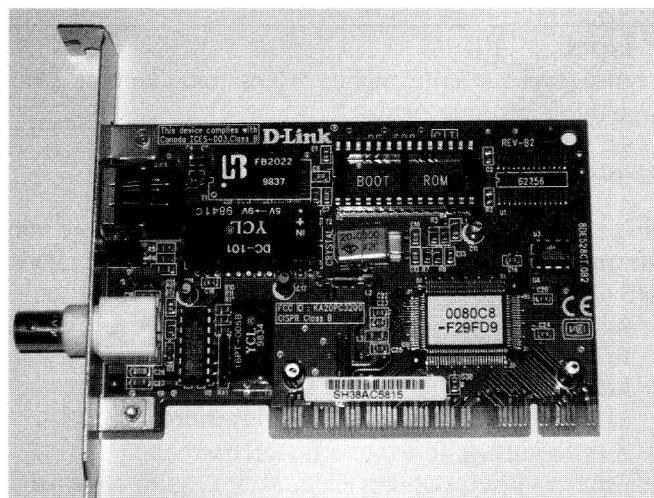
Igaz viszont, hogy régebbi számítógépek esetében ez nem probléma, hiszen a régebbi számítógépek amúgy is kisebb sávszélességet képesek használni.

- A régebbi, ISA hálózati kártyák esetében be kell állítanunk a megfelelő I/O cím és IRQ szám értékét.

A régi kártyákon a beállításokat apró kapcsolók segítségével módosíthatjuk, amelyek a hálózati kártyán találhatók. A kapcsolók állításához szét kell szednünk számítógépet, ami kissé kényelmetlenné teszi a munkát.

Az újabb kártyákon a beállításokat a gyártó által biztosított program segítségével módosíthatjuk. A gyártó által biztosított program viszont a legritkább esetben használható GNU/Linux rendszeren, ami *nagyon* kényelmetlen.

A legújabb ISA hálózati kártyák beállítása automatikus. Ezek a hálózati kártyák azokból az időkből származnak, amikor a periféria automatikus beállítása (PNP, *plug and play*, „dugd be és játsz”) gyerekcipőben járt, és sokszor nem működött (*plug and pray*, „dugd be és imádkozz”).



4.9. ábra. PCI hálózati csatoló

Mindent összevetve elmondható, hogy ma már érdemesebb elkerülni az ISA csatolófelülettel szerelt hálózati kártyák használatát, de ez természetesen nem jelenti azt, hogy mindenkorban le kell mondanunk a bolhapiacon néhány száz forintért beszerezhető eszközök használatáról.

A PCI csatolófelület

A modern hálózati csatolók általában PCI (*peripheral component interconnect, perifériaelém-csatoló*) felületen csatlakoznak a számítógéphez. A 4.9. ábrán egy PCI hálózati csatolót láthatunk.

A PCI felület két legfontosabb előnye a megnövelt adatátviteli sebesség (sávszélesség) és a kiforrott automatikus perifériabeállítás. Az automatikus beállításnak köszönhetően a PCI hálózati csatolók beállításával általában nem kell foglalkoznunk, csak a megfelelő meghajtóprogramról kell gondoskodnunk.

A szabvány szerint a PCI eszközök mindegyike rendelkezik egy azonosítószámmal (PCI ID), amelyből kideríthető, hogy ki gyártotta az eszközt, mi a pontos neve, és az azonos néven gyártott termékek közül melyik változatról van szó. A PCI eszközöktől az azonosítószám lekérdezhető, így azok egyértelműen azonosíthatók.

A GNU/Linux rendszerek általában rendelkeznek egy folyamatosan karbantartott nyilvántartással, amelyből kideríthető, hogy az adott PCI azonosítóval elláttott eszközt ki gyártotta, és mi a pontos elnevezése. Ez a nyilvántartás a legtöbb GNU/Linux rendszeren a `pci.ids` állományban található valamelyik (például `/usr/share/hwdate`) könyvtárban.

4.4. tábla: lspci

A program segítségével a számítógéphez PCI csatolón keresztül kapcsolódó perifériákról kaphatunk információkat.

lspci [kapcsolók]

A program a /proc/pci szimulált állományból kiolvassa, hogy milyen PCI eszközök csatlakoznak a számítógéphez, majd a pci.ids állományban található adatok alapján olvasható formában kiírja azokat a szabványos kimenetre.

Kapcsoló Jelentés

-v	Részletes információk kiírása.
-vv	Még részletesebb információk kiírása.
-n	Szöveges információk helyett kódszámok kiírása.
-t	A PCI rendszer ábrázolása faszerkezetben.
-d	Csak a hexadecimális kóddal megadott eszközöt mutatja.

gyártó:eszköz A gyártó és az eszköz külön-külön elhagyható.

A PCI eszközök nevének kikereséséhez az lspci program a pci.ids állományt használja. A program lekérdezi a számítógépen található PCI eszközök azonosítóját, és kiírja azokat egy listában a szabványos kimenetre. A program használatát mutatja be a következő példa:

24. példa. Meg szeretnénk tudni, hogy milyen PCI Ethernet eszközök vannak telepítve a számítógépbe. Használjuk az lspci programot!

```
$ lspci | grep Ethernet
01:01.0 Ethernet controller: Realtek Semiconductor Co., Ltd.
RTL-8139/8139C/8139C+ (rev 10)
02:00.0 Ethernet controller: 3Com Corporation: Unknown device
0013 (rev 01)
$
```

Amint látjuk, a számítógépben két Ethernet hálózati csatoló található, amelyek közül az egyik ismeretlen, nem szerepel a nyilvántartásban. A nyilvántartás frissítése után esetleg megtudhatjuk ennek az eszköznek is a nevét.

Az lspci képes az eszközöket kódjukkal megjeleníteni. Ezt mutatja be a következő példa:

25. példa. A 24. példában szerepelt egy ismeretlen hálózati csatoló, amelynek kódja 0x13 volt. Kérdezzük le az összes PCI eszköz adatait, amelynek ez a kódja!

```
$ lspci -d :13 -n -v
02:00.0 Class 0200: a727:0013 (rev 01)
          Subsystem: 10b7:1026
```

```
Flags: medium devsel, IRQ 11
Memory at 18800000 (32-bit, non-prefetchable)
Capabilities: [44] Power Management version 2
$
```

A `lspci -d` kapcsolójával megadhatjuk a gyártó és az eszköz kódját, így a program csak bizonyos eszközök adatait fogja kiírni. A `-n` kapcsolóval kérhetjük, hogy ne a szöveges leírások, hanem a kódok jelenjenek meg, a `-v` kapcsolóval pedig részletes adatokat kérhetünk.

A `pci.ids` állomány szerkesztésével mi magunk is megadhatjuk, hogy a PCI azonosítók milyen perifériákat jelölnek, de tudnunk kell, hogy ez még nem jelenti azt, hogy a Linux rendszermag kezelni, támogatni fogja az eszköz használatát. A `pci.ids` bővítése csak arra jó, hogy az eszközöt könnyebben felismerjük, az eszköz használatához a megfelelő meghajtóprogramot kell telepítenünk.

26. példa. A 25. példában található eszköz adatait írjuk be a `pci.ids` állományba. A 25 példa tanúsága szerint a gyártó azonosítója a727, a terméké pedig 0013. Helyezzük el az állományban a terméket leíró sort:

1	a727 3Com Corporation
2	0013 3CRPAG175 Wireless PC Card

Ha most indítjuk az `lspci` programot, az már a megfelelő nevet írja ki:

```
lspci -d :13
02:00.0 Ethernet controller: 3Com Corporation 3CRPAG175
Wireless PC Card (rev 01)
$
```

Ha ezzel megvagyunk, az Interneten keresztül értesíthetjük is az adatbázist nyilvántartó fejlesztőket az általunk felfedezett új azonosítóról, hogy annak adatait ismerjék az `lspci` újabb változatai.

Néhány számítógép – például a hordozható számítógépek újabb változatai – beépített hálózati csatolóval rendelkeznek. A beépített hálózati csatolók általában PCI csatolófélületet használnak a számítógéppel való kapcsolattartásra. Általában csak annyiban különböznek a PCI hálózati kártyáktól, hogy nem lehet eltávolítni őket a számítógépből. Ezeknek a hálózati eszközöknek az azonosítása során különösen hasznos lehet az `lspci` program.

A PCMCIA csatolófelület

A modern hordozható számítógépeket általában felszerelik PCMCIA (*Personal Computer Memory Card International Association*, személyi számítógépes memóriakártya nemzetközi szövetség) vagy más néven PC Card (PC kártya) csatlakozófelülettel is. A PCMCIA csatolófélületet ma már gyakran használjuk hálózati csatoló



4.10. ábra. Vezeték nélküli PCMCIA hálózati csatolók

csatlakoztatására. A 4.10. ábrán PCMCIA csatolófelülettel szerelt vezeték nélküli hálózati csatolókat láthatunk.

A PCMCIA csatolófelületre csatlakozó bővítőkártyákat el lehet távolítni, ki lehet cserélni a számítógép bekapcsolt állapotában, ezért a GNU/Linux rendszereket általában úgy építik fel, hogy a kártya eltávolítása és behelyezése esetén a hálózat beállítása megtörténjen.

4.1.3. A meghajtóprogram betöltése

A számítógép-hálózathoz való csatlakozáshoz hálózati kártyát (NIC) használunk, amelyet a használat előtt – általában a GNU/Linux telepítésekor – megfelelőképpen be kell állítani. A beállítás első lépése a megfelelő meghajtóprogram (*device driver*) telepítése, üzembe állítása. A hálózati kártya meghajtóprogramja a Linux rendszermag azon része, amely a hálózati csatolót vezérli. A meghajtóprogram tehát a rendszermag része, a legtöbb esetben a rendszermagba töltött magmodul. Ha tehát egy hálózati csatolót szeretnénk használni, a meghajtóprogramot megvalósító magmodult betöljük, és a hálózatot azon keresztül használjuk.

A rendszermagba töltött hálózati kártyát vezérlő meghajtóprogramok egy-egy csatolót hoznak létre⁶, amelyek a rendszermagon belül logikai csatlakozási pontot építenek ki a hálózat felé. Ezeknek a logikai csatolóknak a szokás szerint eth0, eth1 stb. nevekkel látja el a rendszermag (legalábbis Ethernet szabványú hálózati csatolók esetében). A meghajtóprogram által létrehozott logikai csatlakozási pontra hivatkozva végezhetjük el a hálózat további beállításait az ifconfig, route stb. programok segítségével.

⁶Több egyforma hálózati kártya esetén egy meghajtóprogram több csatolót is létrehoz.



Az eth0, eth1 stb. nevekhez nem kapcsolódik karaktereszköz- vagy blokkész köz-meghajtó könyvtárbejegyzés, azaz egyetlen GNU/Linux rendszeren sem található /dev/eth0 vagy /dev/eth1 állomány. A számítógép-hálózat egyszerűen túl gyors ahhoz, hogy ilyen módon használjuk!

27. példa. Be szeretnénk tölteni a számítógében található hálózati kártya meghajtóprogramját. A lspci szerint a számítógépen a következő hálózati kártya található:

```
$ lspci | grep Ether
01:01.0 Ethernet controller: Realtek Semiconductor Co., Ltd.
RTL-8139/8139C/8139C+ (rev 10)
$
```

Úgy gondoljuk, hogy ennek a hálózati kártyának a meghajtóprogramja a 8139too magmodulban lehet. Ellenőrizzük ezt a modinfo program segítségével!

```
$ modinfo 8139too | head -n 4
author:      Jeff Garzik <jgarzik@pobox.com>
description: RealTek RTL-8139 Fast Ethernet driver
license:     GPL
parm:        debug:8139too bitmapped message enable number
$
```

A gyanunk igazolódni látszik, bár még nem lehetünk biztosak abban, hogy a meghajtóprogram valóban képes lesz kezelni a hálózati kártyát. Most megpróbáljuk betölteni a modult:

```
$ modprobe 8139too
$
```

Úgy tűnik, hogy a magmodul betöltődött, de ez még nem jelenti azt, hogy a hálózati kártyát képes lesz kezelni. Vizsgáljuk meg hogy a magmodul milyen üzenetet helyezett el a rendszernaplóban:

```
$ tail -n 2 /var/log/messages
Jun 29 15:43:05 localhost kernel: 8139too Fast Ethernet driver
0.9.27
Jun 29 15:43:05 localhost kernel: eth0: RealTek RTL8139 at 0xc
000, 00:02:3f:15:34:bb, IRQ 11
```

Amint látjuk, a meghajtóprogram megtalálta a hálózati kártyát, a kezdeti beállításokat elvégezte, és a kártyát üzemkész állapotba hozta.

Ha a hálózati kártya meghajtóprogramját sikerült betöltenünk, akkor gondoskodhatunk arról, hogy a meghajtóprogram automatikusan betöltődjön, amikor szükség van rá. Ezt úgy tehetjük meg, ha a /etc/conf.modules,

/etc/modules.conf vagy a /etc/modprobe.conf állományban egy másodlagos nevet hozunk létre a meghajtóprogram számára. A másodlagos név legyen eth0, ha ez az első hálózati csatoló, eth1, ha a második és így tovább. Ha létrehoztuk a hálózati kártya meghajtóprogramja számára a másodlagos nevet, a hálózati alrendszer kezelésére szolgáló héjprogramok a magmodult automatikusan be tudják tölteni.

28. példa. A 27. példa hálózati kártyájának meghajtóprogramjának automatikus betöltését szeretnénk lehetővé tenni. Először vizsgáljuk meg, hogy a magmodulok másodlagos neveinek létrehozására melyik állomány szolgál az adott számítógépen!

```
$ cd /etc
$ ls -lh modules.conf conf.modules modprobe.conf
ls: modules.conf: Nincs ilyen fájl vagy könyvtár
ls: conf.modules: Nincs ilyen fájl vagy könyvtár
-rw-r--r-- 1 root root 366 jún 29 15:42 modprobe.conf
$
```

Amint látjuk, ezen a számítógépen a /etc/modprobe.conf állományban hozhatunk létre másodlagos neveket a magmodulok számára.

Helyezzük el az állományban a következő bejegyzést:

1	# Ethernet controller: Realtek Semiconductor Co., Ltd.
2	# RTL-8139/8139C/8139C+
3	alias eth0 8139too

Most már betölthető a magmodul eth0 néven is:

```
$ modprobe eth0
$
```

A legtöbb GNU/Linux terjesztés olyan indítóprogramokat tartalmaz a hálózati szolgáltatások számára, amelyek ilyen beállítás segítségével képesek automatikusan betölteni a hálózati kártyák meghajtóprogramját.

Az automatikus kártyafelderítés

A hálózati kártyák meghajtóprogramjainak betöltése kapcsán mindenkorban meg kell ismerkednünk az automatikus kártyafelderítés (*autoprobe*) fogalmával!

Amikor a hálózati kártya meghajtóprogramját a rendszermagba betöltöttük, az automatikusan megtalálta és felismerte a hálózati kártyát (lásd 27. példa). Az automatikus kártyafelderítés azonban nem minden esetben ilyen egyszerű és problémamentes!

Ha a hálózati kártya PCI csatolófelületen keresztül kapcsolódik a számítógéphez, az automatikus kártyafelderítés nem okozhat problémát, hiszen az összes PCI felülettel rendelkező áramkör egységes azonosítási rendszerrel rendelkezik,

az azonosítás ezért viszonylag egyszerűen és biztonságosan elvégezhető. Egészen más a helyzet azonban akkor, ha a hálózati kártya régebbi (például ISA) csatolófelületet használ.

Régebbi csatolófelület használata esetén a meghajtóprogram bizonyos I/O címeken megpróbálja „megszólítani” a hálózati kártyát, és figyelmesen megvizsgálja, hogy a válasz az általa kezelt kártyától származik-e. Ha nem, a meghajtóprogram tovább keres, újabb és újabb címeken próbálja meg megkeresni a hálózati kártyát, amelynek vezérlésére készült.

A perifériakezelők világában az ilyen „megszólítás” azonban korántsem veszélytelen! Ha a meghajtóprogram olyan I/O címről olvas, amely nem a megfelelő hálózati kártyával való kapcsolattartásra szolgál, ha olyan perifériát „szólít meg”, amelynek kezelésére nem képes, a periféria – és így a számítógép – a kikapcsolásig üzemképtelenné válhat. Ezért a meghajtóprogramok az automatikus felderítés során gondosan elkerülnek bizonyos I/O címeket, amelyeket veszélyesnek ítélnek! Ha az ISA felületen csatlakozó hálózati kártyánk olyan címen található, amelyet a meghajtóprogram írja veszélyesnek ítélt, az automatikus kártyafelderítés nem fogja megtalálni a kártyát. Ez ellen kétféleképpen küzdhetünk:

1. A hálózati kártyát olyan beállításokkal láthatjuk el, amelyek garantálják az automatikus kártyafelderítést.

Minden meghajtóprogram dokumentációja megadja, hogy a hálózati kártya milyen beállításai mellett működik az automatikus kártyafelderítés, nekünk pedig ezek közül a beállítások közül választva kell beállítanunk a hálózati kártyát.

Ez a módszer talán egyszerű, de sokan nem ajánlják; szerintük jobb az automatikus kártyafelderítéstől egyszer s mindenkorra megszabadulni.

2. Megoldást jelenthet, ha a meghajtóprogram betöltésekor megadjuk, hogy milyen beállításokkal található meg a hálózati kártya, azaz milyen címen „szólítható meg”.

Ez a megoldás szükségtelenné teszi a kártya „megkeresését”, és garantálja, hogy a meghajtóprogram első kísérletét siker fogja koronázni. A meghajtóprogram természetesen a szokott óvatossággal fogja a megadott címen felkeresni a hálózati kártyát, az automatikus kártyafelderítés azonban garantáltan a megfelelő típusú kártyát fogja „megszólítani”.

Ez a módszer meglehetősen biztonságos – feltéve persze, hogy a megfelelő adatokat adjuk meg –, de van egy hátránya. Ha a hálózati kártya beállításai megváltoznak, a meghajtóprogram nem fogja megtalálni (mivel nem is keresi), ezért a beállításokat át kell írnunk a megfelelő beállítóállományban.

A második módszer gyakorlati bemutatása előtt szeretnénk még egyszer felhívni a figyelmet arra, hogy régi (például ISA) csatolófelületről van szó, az újabb felületeken (például PCI) az automatikus kártyafelderítés problémamentes!

29. példa. Régebbi típusú SMC Elite hálózati kártyának meghajtóprogramjának betöltése közben el szeretnénk kerülni az automatikus kártyafelderítés veszélyeit. Vizsgáljuk meg, hogy a wd magmodul milyen paramétereket fogad!

```
$ modinfo wd | grep ^parm
parm: io:I/O base address(es)
parm: irq:IRQ number(s) (ignored for PureData boards)
parm: mem:memory base address(es)(ignored for PureData boards)
parm: mem_end:memory end address(es)
$
```

Amint látjuk, a magmodulnak az io paraméter segítségével adhatjuk meg az I/O címet, amelyen a hálózati kártya elérhető.

Helyezzük el a következő sorokat a /etc/conf.modules, /etc/modules.conf vagy /etc/modprobe.conf állományok közül abban, amelyiket a rendszerünk használja!

```
1 # SMC Elite
2 alias eth0 wd
3 options wd io=0x300
```

Figyeljük meg, hogy az I/O címet tizenhatos számrendszerben adtuk meg a 0x előtaggal!

Több hálózati kártya

Ha a GNU/Linux rendszert futtató számítógépünkbe több hálózati kártyát építünk, igen izgalmas és hasznos dolgokra használhatjuk. A következő néhány bekezdés arról szól, miképpen tölthetünk be meghajtóprogramot több hálózati kártya számára.

Ha a hálózati kártyák típusa különböző, a feladat egyszerűen megoldható. minden egyes hálózati kártya számára be kell töltenünk a megfelelő meghajtóprogramot, amelyek egymástól függetlenül vezérelve az egyes hálózati kártyákat problémamentesen működtetik. Ezt az esetet mutatja be a 47. oldalon található 18. példa.

A több hálózati kártyával kapcsolatban azonban egy apróságot mindenkorban szem előtt kell tartanunk! A rendszermagnak nincs információja a modulok betöltésére használt másodlagos nevekről. A rendszermag által adott eth0, eth1 stb. elnevezések tökéletesen függetlenek a meghajtóprogram betöltésére használt modulnévtől és a modul másodlagos nevétől!

Tegyük fel, hogy elhelyeztük a /etc/modprobe.conf állományban a következő sorokat:

```
1 alias eth0 8139too
2 alias eth1 ipw2100
```

Ha a két magmodul egyike sincs betöltve, és kiadjuk a modul betöltésére a modprobe eth1 parancsot, a rendszermagban az eth0 hálózati illesztő jön létre! Ez természetes is, hiszen a rendszermagban a hálózati illesztők elnevezése folytonos történik!

Ha viszont a számítógépet újraindítjuk, a rendszerindítást végző héjprogramok gondosan ügyelnek arra, hogy az egyes modulokat a másodlagos nevük sorrendjében töltsek be, így a létrejött hálózati illesztők neve összhangban lesz a modulok másodlagos nevével.



Legalábbis, ha minden rendben ment, azaz, ha a modulokat sikerült betölteni. Ha például az eth0 modul betöltése sikertelen, a rendszermag nem hozza létre az eth0 hálózati illesztőt. Ha viszont nincs eth0 nevű illesztő, és betölthetjük a eth1 másodlagos néven található meghajtóprogramot, az az eth0 nevű illesztőt hozza létre a rendszermagban.

A rendszermag hálózati illesztőinek elnevezési rendszere lehetővé teszi azt is, hogy a hálózati csatolókat megcseréljük. Ha két különböző hálózati kártya található a számítógépünkben, és például elbabrálunk valamit a szobánk takarítása közben, megoldást jelenthet a hálózati kártyákat meghajtó magmodulok másodlagos nevének megcserélése és a hálózat újraindítása. Ez minden esetre érdekesebb megoldás, mint az asztal alatt négykézláb előadott bűvész kedégekben...

Ha a számítógépben több egyforma hálózati csatoló található, elegendő egyetlen meghajtóprogramot betöltenünk. A modern meghajtóprogramok általában négy hálózati kártyát is képesek kezelni, így egyetlen modul betöltése elegendő lehet. Ha el akarjuk kerülni a hálózati kártyák automatikus felderítését, akár egy sorban is megadhatjuk az I/O címeket. Ezt mutatja be a következő példa:

30. példa. Két ISA csatolót használó hálózati kártyát építettünk a számítógépünkbe, amelyek mindegyikét képes kezelni a wd magmodul. El szeretnénk kerülni a hálózati kártyák automatikus felderítését, ezért megadjuk minden kártya I/O címét:

```
1 alias eth0 wd
2 alias eth1 wd
3 options wd io=0x200,0x300
```

Mivel a magmodulok másodlagos nevéhez is kapcsolhatunk paramétereket, az I/O címeket megadhatjuk külön is, a következő formában:

```
1 alias eth0 wd
2 alias eth1 wd
3 options eth0 io=0x200
4 options eth1 io=0x300
```

Ez a forma talán kissé érthetőbb, más lényeges különbség azonban nincs a két írásmód közt.

Előfordulhat, hogy több azonos típusú hálózati kártyánk van, mint amennyit a meghajtóprogram kezelni képes. Ilyen esetben a meghajtóprogramot több példányban is be kell töltönünk, ami kissé körülmenyes, mivel a Linux rendszermag nem teszi lehetővé, hogy több betöltött magmodul osztozzon egy néven. Szerencsére a modprobe -o kapcsolójával a betöltés során megadhatjuk, hogy a rendszermag milyen néven kezelje a betöltött magmodult, így minden betöltött meghajtóprogram-példánynak egyedi nevet adhatunk. Ezt mutatja be a következő példa:

31. példa. Be szeretnénk tölteni a 3c501 magmodult két példányban az eth0 és eth1 hálózati illesztő létrehozására. Helyezzük el a beállítóállományban a következő sorokat:

```
1 alias eth0 3c501
2 alias eth1 3c501
3 options eth0 -o 3c501-a io=0x200
4 options eth1 -o 3c501-b io=0x300
```

Amint látjuk, a modprobe -o kapcsolója után az általunk választott egyedi név található.

4.1.4. A hálózati csatoló beállítása

Az előző oldalakon olvashattunk arról, hogy miképpen tölthetjük be a hálózati kártyák meghajtóprogramját. A meghajtóprogramok betöltésének eredményeképpen létrejött egy vagy több hálózati csatoló, amely a rendszermag belső eszközékként a hálózattal való kapcsolattartásra szolgál. A következő néhány oldalon arról olvashatunk, hogy miképpen rendelhetünk IP címet, alhálózati maszkot és néhány egyéb beállítást ezekhez a hálózati eszközökhez. Az itt tárgyalt beállítások létfontosságúak, addig nem használhatjuk a hálózatot, amíg ezeket a beállításokat el nem végezzük.

GNU/Linux rendszereken az eth0, eth1 stb. hálózati csatolók beállítására az ifconfig program szolgál, amely nevét a interface (illesztő) szóról kapta. A program segítségével a hálózati csatolókról kérhetünk információt és statisztikai adatokat, valamint a hálózati eszközök alapvető beállítását végezhetjük el.

Ha betöltöttük a hálózati kártya meghajtóprogramját, az ifconfig program up parancsa segítségével bekapcsolhatjuk a hálózati illesztőt, paraméter nélkül futtatva pedig lekérdezhetjük a csatolók adatait⁷. Ez a lépés arra is használható, hogy ellenőrizzük: a meghajtóprogram betöltése valóban sikeres volt, az illesztő létrejött, a hálózati kártya működik⁸.

⁷Ha nem engedélyezzük a csatolót az up paraméterrel, az ifconfig -a kapcsolójával jeleníthetjük meg az adatait (a lektor).

⁸Elképzelhető olyan hardverhiba is, amely csak a hálózati forgalom megindítása után derül ki (a lektor).

32. példa. Betöltöttük az első meghajtóprogramot, és úgy gondoljuk az eth0 csatoló létrejött. Kapcsoljuk be a meghajtót, és nézzük meg, milyen adatok állnak rendelkezésünkre az eth0 meghajtóval kapcsolatban!

```
$ ifconfig eth0 up
$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:02:3F:15:34:BB
          inet6 addr: fe80::202:3fff:fe15:34bb/64 Scope:Link
          UP BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b) TX bytes:238 (238.0 b)
          Interrupt:11 Base address:0xc000
$
```

Amint látjuk, a meghajtó bekapcsolását sikeres koronázta, a meghajtóról sok érdekes – és érthetetlen – adat áll a rendelkezésünkre.

Ha a hálózatot be is állítottuk volna, a paraméter nélkül futtatott ifconfig kimenete a következő sorokhoz hasonlítható:

```
$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:02:3F:15:34:BB
          inet addr:172.17.2.97 Bcast:172.17.255.255 Mask:255.255.0.0
          inet6 addr: fe80::202:3fff:fe15:34bb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:12588 errors:692 dropped:882 overruns:692 frame:0
          TX packets:17256 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:33480731 (31.9 Mb) TX bytes:577912006 (551.1 Mb)
          Interrupt:11 Base address:0xc000

lo      Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:1109 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1109 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:105763 (103.2 Kb) TX bytes:105763 (103.2 Kb)
$
```

Amint látjuk, még sok minden kell beállítanunk ahhoz, hogy a hálózat tökéletesen működjön.

Ha az ifconfig programot paraméter nélkül futtatjuk, egy listát kapunk az összes hálózati csatolóról és azok legfontosabb adatairól. A listában található fontosabb szavak és rövidítések jelentése a következő:

4.5. tábla: ifconfig

Az ifconfig program segítségével a számítógépbe helyezett, meghajtóprogrammal ellátott hálózati csatolók beállításait kérhetjük le és változtathatjuk meg.

```
ifconfig [csatoló]
ifconfig -a
ifconfig csatoló parancs
```

Ha a programot paraméterek nélkül indítjuk, az aktív hálózati csatolók, ha a -a kapcsolóval, az összes – betöltött meghajtóprogrammal rendelkező – hálózati csatoló legfontosabb adatait írja ki. Ha a programnak egy hálózati csatoló nevét adjuk meg, kiírja annak legfontosabb adatait. A program minden más esetben a megadott hálózati csatoló beállítására használható.

Az ifconfig programot általában a rendszerindítást végző valamelyik héjprogram futtatja a megfelelő paraméterekkel, hogy a hálózatot beállítsa.

Kapcsoló	Jelentés
-up	A hálózati csatoló bekapcsolása.
-down	A hálózati csatoló kikapcsolása.
arp, -arp	Az ARP protokoll engedélyezése és tiltása. Az ARP protokollt a legtöbb esetben nem kell tiltanunk.
promisc, -promisc	A lehallgatási üzemmód engedélyezése és tiltása.
mtu szám	A megengedett legnagyobb csomagméret. Általában nem szükséges megváltoztatnunk az alapértelmezett értéket.
netmask maszk	Az alhálózati maszk megadása.
irq szám	A hálózati csatoló által használt IRQ szám megváltoztatása. Nem minden hálózati csatolónál van erre lehetőség, és szinte soha nincs rá szükség.
io_addr szám	Az I/O címterület megváltoztatása. Nem minden hálózati csatolónál van erre lehetőség, és szinte soha nincs rá szükség.
mem_start cím	A hálózati csatoló által használt memóriaterület címének megváltoztatása. Nem minden hálózati csatolónál van erre lehetőség, és szinte soha nincs rá szükség.
hw ether cím	A MAC cím megváltoztatása. Általában nincs szükségünk arra, hogy ezt a címet megadjuk.
cím	A hálózati csatolóhoz tartozó IP cím megadása. Az IP cím megadása nélkül a hálózati csatolón nem lehet IPv4 adatforgalom.

Link encap: A csatoló által megvalósított hálózati kapcsolat fizikai típusa. Itt általában a következő kulcsszavak egyike található:

Ethernet A hálózati csatoló Ethernet fizikai hálózathoz kapcsolja a számítógépet.

Local Loopback A hálózati csatolóhoz nem tartozik fizikai hálózat, a csatoló a számítógépet önmagához kapcsolja⁹.

HWaddr A fizikai hálózat jellemzőjeként a hálózati kártyához tartozó cím. Ez a cím Ethernet hálózat esetében a 6 bájt hosszúságú Ethernet cím, de más fizikai hálózat esetében természetesen különbözik ettől.

inet addr: A hálózati csatolóhoz rendelt IP cím, amely nélkülözhetetlen TCP/IP kapcsolat kiépítéséhez.

Mask: Az IP címhez tartozó alhálózati maszk, amely fontos a TCP/IP hálózat használatához.

inet6 addr: A csatolóhoz tartozó IP cím, amelyet az IPv6 (újabb TCP/IP szabvány) kapcsolatok során használ a rendszermag. Csak akkor szükséges a cím, ha a TCP/IP újabb változatát kívánjuk használni¹⁰.

Scope: Az IPv6 szabványú IP cím hatóköre, amelyre csak akkor van szükségünk, ha a TCP/IP újabb változatát is használni kívánjuk.

UP A kulcsszó azt jelenti, hogy a hálózati csatolót kapcsolták, csomagok fogadására és küldésére kész.

PROMISC A kulcsszó azt jelzi, hogy a hálózati kártya tetszőleges fizikai hálózati címmel ellátott csomagot fogad, „hallgatózó” üzemmódban van.

RX packets: A fogadott csomagok száma és a különféle hibák száma a fogadott csomagok esetében.

TX packets: A küldött csomagok száma és a különféle hibák száma az elküldött csomagok esetében.

Interrupt: A hálózati kártya által használt megszakításkérés (IRQ) száma.

Base address: A hálózati kártya I/O címe.

Az ifconfig program különféle kapcsolóival beállíthatjuk a hálózati csatolót és a hálózati kártyát. A legfontosabb kapcsolók a következők:

⁹Ez a kapcsolódási mód egyébként jóval hasznosabb annál, mint amilyennek első pillantásra tűnik.

¹⁰Ha a cím látszik, akkor az IPv6 aktív is, ezért ügyelnünk kell az esetleges szolgáltatásokra (a lektor).

up A hálózati kártya bekapcsolása, engedélyezése. Ez az alapértelmezett művelet, vagyis ha megadjuk a hálózati csatoló adatait, az `ifconfig` engedélyezi a hálózati kártya működését.

down A hálózati kártya kikapcsolása, tiltása.

mtu szám A legnagyobb küldhető csomag mérete bajtban. Az Ethernet hálózati kártyákon ennek mérete alapértelmezés szerint 1500, amit általában nem szükséges módosítani.

netmask maszk A hálózati csatolóhoz tartozó alhálózati maszk megadása. A helyes működés érdekében az alhálózati maszk értékét meg kell adnunk.

hw típus cím A hálózati kártya fizikai címének megváltoztatása, ha erre a hálózati kártya képes.

A *típus* a hálózati kártya által használt fizikai hálózat típusa, amelyet Ethernet hálózat esetén az ether kulcsszóval jelzünk.

A legtöbb esetben nem szükséges megváltoztatni a hálózati kártya fizikai címét.

cím A hálózati csatolóhoz tartozó IP cím. A helyes működéshez az IP címet meg kell adnunk.

A hálózati csatolónak a működéséhez mindenkorban rendelkeznie kell IP címmel és alhálózati maszkkal, amelyeket könnyedén beállíthatunk az ifconfig programmal. Ezt mutatja be a következő példa.

33. példa. A hálózati kártya meghajtóprogramjának magmoduljára létrehoztunk egy másodlagos nevet, amely az eth0. Mivel az ifconfig a kerneld segítségével szükség esetén betölti az eth0, eth1 stb. modult, nekünk ezt nem kell megtennünk, csak be kell állítanunk és engedélveznünk kell a hálózati csatolót.

Állítsuk be a 10.1.1.100 IP címet a 255.0.0.0 alhálózati maszkkal!

```
$ ifconfig eth0 netmask 255.0.0.0 10.1.1.100 up
$ ifconfig
eth0  Link encap:Ethernet  HWaddr 00:02:3F:15:34:BB
      inet  addr:10.1.1.100  Bcast:10.255.255.255  Mask:255.0.0.0
            inet6 addr: fe80::202:3fff:fe15:34bb/64  Scope:Link
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:0  errors:0  dropped:0  overruns:0  frame:0
                  TX packets:3  errors:0  dropped:0  overruns:0  carrier:0
                  collisions:0  txqueuelen:1000
                  RX bytes:0 (0.0 b)  TX bytes:238 (238.0 b)
                  Interrupt:11  Base address:0xc000
```

Amint látjuk, a hálózati kártya engedélyezése megtörtént, a hálózati csatoló rendelkezik IP címmel és alhálózati maszkkal, és tulajdonképpen üzemkész. Még az sem lehetetlen, hogy a helyi hálózaton tudunk csomagokat küldeni és fogadni (ha az ifconfig önszorgalomból kitöltötte az útválasztó táblázatot is):

```
$ ping 10.1.1.1
PING 10.1.1.1 (10.1.1.1) 56(84) bytes of data.
64 bytes from 10.1.1.1: icmp_seq=0 ttl=64 time=0.784 ms
64 bytes from 10.1.1.1: icmp_seq=1 ttl=64 time=0.464 ms
64 bytes from 10.1.1.1: icmp_seq=2 ttl=64 time=0.454 ms

--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.454/0.567/0.784/0.154 ms, pipe 2
$
```

A hálózat beállításával azonban még nem végeztünk, ezért túl sok eredményre ne számítsunk.

4.1.5. Az útválasztó táblázat

Amikor a Linux rendszermagnak egy csomagot kell elküldenie, ki kell választania, hogy azt milyen irányban bocsássa útjára. Ez fontos és bonyolult döntés akkor is, ha a számítógép csak egyetlen ponton kapcsolódik a hálózathoz, tehát már most foglalkoznunk kell a kérdéssel.

Az útválasztás kérdését a rendszermag az útválasztó táblázat (*route table*) alapján válaszolja meg, nekünk tehát a megfelelő parancsok kiadásával fel kell építenünk egy útválasztó táblázatot, amelyet a rendszermag a memóriában tárol, és az összes csomag elküldésekor felhasznál.



Fontos, hogy ne hagyjuk magunkat megtéveszteni az útválasztó táblázat neve által! Az útválasztó táblázatra nem csak akkor van szükség, amikor a rendszermag más számítógép számára végez útválasztást, hanem minden esetben, ha a rendszermagnak csomagot kell küldenie.

Az útválasztó táblázat kezelésére a route programot használhatjuk. A program paraméter nélkül indítva kiírja az éppen érvényben lévő útválasztó táblázatot a szabványos kimenetre, de mi egyelőre kizárolág a -n kapcsolójával együtt használjuk a programot. A route -n kapcsolója arra ad utasítást, hogy az útválasztó táblázat bejegyzéseit IP címekkel és nem a hozzájuk tartozó számítógépnevekkel írjuk ki. Mivel a számítógépnevek kezelését még nem állítottuk be, ez a kapcsol mindenkorban szükséges.



A route programot általában arra használjuk, hogy a hálózat elérését hibamentessé, tökéletesen működővé tegyük. Ha viszont nem adjuk meg a programnak a -n kapcsolót, az megpróbálja kideríteni az IP címekhez

4.6. tábla: route

Az útválasztó táblázat lekérdezése és beállítása.

```
route [kapcsolók]
route add [kapcsolók]
route del [kapcsolók]
```

Ha az add és del paraméter nélkül indítjuk, a route kiírja a szabványos kimenetre az útválasztó táblázatot. Az add paraméterrel új bejegyzést adhatunk meg, a del paraméterrel pedig bejegyzést törölhetünk az útválasztó táblázatból.

Kapcsoló	Jelentés
-n	A DNS keresés tiltása, IP címek használata a táblázat kiírásakor.
add	Bejegyzés létrehozása.
del	Bejegyzés törlése.
-host cím	A címzett számítógép IP címének megadása.
-net	A címzett hálózat hálózati címének megadása.
cím/maszk	
netmask	A címzett hálózat alhálózati maszkjának megadása.
gw	A címzett eléréséhez használandó számítógép IP címének megadása.
dev név	A hálózati csatoló nevének megadása (például eth0, eth1 stb.).

tartozó neveket a hálózat segítségével. Ez elég szerencsétlen módon azt szokta eredményezni, hogy a figyelmetlen rendszergazda elunja a várakozást, megszakítja a route futását a **[Ctrl] + [C]** billentyűkombinációval, és újra kiadja a parancsot, de most már a -n kapcsolóval együtt.

Mielőtt saját kezűleg építenénk fel az útválasztó táblázatot, vizsgáljuk meg egy tökéletesen működő számítógép útválasztó táblázatát!

34. példa. Találunk egy GNU/Linux rendszert futtató számítógépet, amely egy hálózati kártyával kapcsolódik a helyi hálózatra. Meg szeretnénk vizsgálni az útválasztó táblázatát:

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref Use Iface
10.0.0.0       0.0.0.0        255.0.0.0    U      0      0      0 eth0
127.0.0.0      0.0.0.0        255.0.0.0    U      0      0      0 lo
0.0.0.0        10.255.255.254 0.0.0.0     UG     0      0      0 eth0
```

Amint látjuk, a route program egy viszonylag egyszerű táblázatot írt a kimenetére. Valójában ennél egyszerűbb útválasztó táblázatot nem is igazán lehet elkezelní.

A route által kiírt táblázat a következő oszlopokat tartalmazza:

Destination A küldendő csomag címzettjének címe. Ha a küldendő csomag címzettje illeszkedik az itt található címre, akkor az adott sor jelöli ki, hogy a küldendő csomagot milyen úton kell elküldeni.

Ennek az oszlopnak az értelmezéséhez fel kell használnunk a harmadik oszlopban található értéket (*Genmask*), amely megadja az alhálózati maszkot. A küldendő csomag címzettjének az első és a harmadik oszlop által meghatározott címtartományba kell esnie ahhoz, hogy a sor érvényesüljön.

Különlegesen kell kezelnünk az 0.0.0.0 címet, ha az első oszlopban van! A 0.0.0.0 cím minden címre illeszkedik, az a bejegyzés tehát, amely ezt a címet tartalmazza, bármely címzett számára megfelelő.

Azt mondjuk, hogy az a bejegyzés, amely a 0.0.0.0 értéket tartalmazza az első oszlopban, az alapértelmezett átjáró (*default gateway*), ahova azokat a csomagokat küldjük, amelyek célcíme nem volt illeszthető egyetlen más bejegyzésre sem.

Gateway Az átjáró címe. Ha itt nem 0.0.0.0 található, hanem egy valós IP cím, akkor a csomagot nem közvetlenül kell küldeni, hanem az itt megadott átjárónak kell továbbítani továbbküldésre.

Genmask Az alhálózati maszk, amely megadja, hogy az első oszlopban található cím miképpen illesztendő a címzett IP címére. Az alapértelmezett átjáró esetében az itt található maszk nem hordoz használható információt.

Flags Az adott bejegyzésre érvényes módosítókapcsolók, amelyek többek között a következők lehetnek:

U A bejegyzés érvényes, használandó, be van kapcsolva.

H A bejegyzés egyetlen IP címre vonatkozik, az első oszlopnak pontosan meg kell egyeznie a címzett IP címével ahhoz, hogy a bejegyzést használja a rendszermag.

G A bejegyzés átjárón kereszttüli utat jelöl, a csomagot az átjárónak kell küldeni továbbítás céljából.

Iface A hálózati csatoló, amelyen keresztül a csomagnak távoznia kell.

A route által kiírt táblázat értelmezését segíti a következő példa.

35. példa. Értelmezzük és írjuk le szavakkal a 34. példában látható útválasztó táblázatot!

```
$ route -n
Kernel IP routing table
Destination Gateway     Genmask   Flags Metric Ref Use Iface
10.0.0.0      0.0.0.0    255.0.0.0 U        0      0      0 eth0
127.0.0.0     0.0.0.0    255.0.0.0 U        0      0      0 lo
0.0.0.0       10.255.255.254 0.0.0.0  UG      0      0      0 eth0
```

A táblázat a következő szabályokat tartalmazza:

1. Ha a csomag címzettje a 10.0.0.0/255.0.0.0 címtartományban található, a csomag közvetlenül küldendő a címzettnek az eth0 hálózati csatolón keresztül.
2. Ha ellenben a csomag címzettje a 127.0.0.0/255.0.0.0 címtartományban található, a csomagot az lo hálózati csatolón keresztül kell közvetlenül küldenünk.
3. minden más esetben továbbítani kell a csomagot a 10.255.255.254 IP című számítógépnek, amely pontosan tudja, merre kell küldeni a csomagokat helyettünk.

A táblázat könnyedén megérthető a sorok végigolvasásával.

A legutóbbi példa kapcsán egy fontos dologra mindenkorban fel kell figyelnünk. Az utolsó sorban megadott alapértelmezett átjáróra vonatkozó bejegyzés nem adja meg a végső választ az útválasztás kérdésére, hiszen csak azt írja elő, hogy a csomagot továbbítani kell a 10.255.255.254 IP című számítógépnek, viszont azt nem mondja meg, hogy milyen úton! Ha tehát a rendszermag a harmadik sor alapján az átjáróhoz kényszerül fordulni, azonnal felmerül a kérdés, hogy azt hogyan éri el. Figyeljük meg, hogy ezt az új kérdést az útválasztó táblázat újbóli alkalmazásával az első bejegyzés válaszolja meg! Alapszabályként érdemes megjegyezni, hogy alapértelmezett átjáró csak az számítógép lehet, amelyet az útválasztó táblázat más bejegyzése alapján már elérünk.

Ha megértettük, hogy milyen módon épül fel az útválasztó táblázat, már csak azt kell megtudnunk, hogy a route program segítségével miképpen építhetjük fel, hogyan módosíthatjuk.

A route első paramétereként megadhatjuk a del (delete, törlés) és az add (hozzáadás) kulcsszavakat, így sorokat törölhetünk és szűrhetünk be az útválasztó táblázatba. A sorokat a következő kapcsolókkal írhatjuk le:

- net cím Az első oszlopban megadott cím, ha a bejegyzés alhálóra vonatkozik.
- netmask maszk A harmadik oszlopban található alhálózati cím megadása.
- host cím Az első oszlopban megadott cím, ha a bejegyzés nem alhálóra, hanem egyetlen címre vonatkozik.

gw cím A második oszlopban megadott átjáró. Ha az adott bejegyzés nem használ átjárót, nem kell megadni.

default gw cím Az alapértelmezett átjáróra vonatkozó bejegyzés.

dev csatoló A hálózati csatoló – utolsó oszlop – nevének megadása.

36. példa. A 33. példában beállítottuk az eth0 hálózati csatoló IP címét és alhálózati maszkját. Készítsük el az egyszerű útválasztó táblázatot is!

Az eth0 hálózati csatoló a 10.0.0.0/255.0.0.0 alhálózatba tartozik. Ellenőrizzük, hogy üres-e az útválasztó táblázat és helyezzünk el egy bejegyzést, amelynek segítségével a rendszermag képes kapcsolódni azokhoz a számítógépekhez, amelyek ugyanebbe az alhálózatba tartoznak!

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask      Flags Metric Ref Use Iface
$ route add -net 10.0.0.0 netmask 255.0.0.0 dev eth0
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask      Flags Metric Ref Use Iface
10.0.0.0        0.0.0.0        255.0.0.0   U          0      0    0    eth0
$
```

Most már csak az alapértelmezett átjáró számára kell egy bejegyzést készítenünk, hogy az Internet többi számítógépét is elérhessük.

```
$ route add default gw 10.255.255.254
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask      Flags Metric Ref Use Iface
10.0.0.0        0.0.0.0        255.0.0.0   U          0      0    0    eth0
0.0.0.0         10.255.255.254 0.0.0.0    UG         0      0    0    eth0
$
```

Az útválasztó táblázat ezzel elkészült, a számítógép képessé vált a csomagok küldésére és fogadására, felkészítettük a kapcsolattartásra az Internet összes számítógépével.

Felmerülhet még a kérdés, hogy mi volt az a 1o hálózati csatoló a 34. példában, de ezt már igazán könnyen megválaszolhatjuk.

A szabványok lehetővé teszik, hogy minden egyes számítógép felhasználja a 127.0.0.0/255.0.0.0 IP tartományt arra, hogy saját magának küldjön csomagokat, a Linux rendszermag pedig erre a célra a 1o hálózati csatolót használja. Azok a csomagok, amelyek a 1o szimulált hálózati eszközön távoznak, ugyanitt gyorsan vissza is jönnek. Ez első pillantásra nem tűnik túlságosan hasznos lehetőségek, de ha belegondolunk, hogy lehetővé teszi az egyazon gépen futó alkalmazások közti szabványos adatáramlást, könnyen beláthatjuk, milyen nagyszerű

ötlet¹¹. Az eddig tárgyalt ismeretek alapján egyszerűen biztosíthatjuk magunknak ezt a lehetőséget, ahogyan a következő példa is bemutatja.

37. példa. Válasszuk a hagyomány szerint a 127.0.0.1 IP címet a saját géppel való kapcsolattartásra, és állítsuk be erre a címre az lo belső hálózati csatolót:

```
$ ifconfig lo netmask 255.0.0.0 127.0.0.1 up
$
```

Adjunk hozzá egy sort az útválasztó táblázathoz, hogy az új alhálóval fel tudjuk venni a kapcsolatot:

```
$ route add -net 127.0.0.0 netmask 255.0.0.0 dev lo
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref Use Iface
10.0.0.0        0.0.0.0        255.0.0.0    U      0      0   0   eth0
127.0.0.0       0.0.0.0        255.0.0.0    U      0      0   0   lo
0.0.0.0         10.255.255.254 0.0.0.0     UG     0      0   0   eth0
$
```

Ezzel lehetővé tettük, hogy a számítógépünk a hálózati kártyát megkerülve, közvetlenül küldjön magának csomagokat a 127.0.0.1 IP cím felhasználásával.

4.1.6. Összefoglalás

A számítógép-hálózat beállítását ezzel befejeztük. Összefoglalálandó az eddigi ismereteket bemutatunk egy egyszerű héjprogramot, amely hálózati csatoló és útválasztó táblázat beállításával működőképessé teszi a hálózati csatlakozást.

38. példa. Készítsük el a lehető legegyszerűbb héjprogramot, amely a hálózat indítására és leállítására használható!

	/etc/init.d/networking
1	#! /bin/bash
2	#
3	# Hálózati csatlakozást beállító és leállító program.
4	#
5	IPCIM="10.1.1.100"
6	MASZK="255.0.0.0"
7	ATJARO="10.255.255.254"
8	
9	case "\$1" in
10	start)
11	# A hálózati csatolók bekapsolása

¹¹E nélkül a lehetőség nélkül sok szolgáltatás – köztük a grafikus felületet biztosító X Window – nem működik (a lektor).

```
12      ifconfig eth0 netmask $MASZK $IPCIM up
13      ifconfig lo netmask 255.0.0.0 127.0.0.1 up
14      # Az útválasztó táblázat felépítése:
15      # A 10.0.0.0/255.0.0.0 hálózat bejegyzését az
16      # ifconfig önszorgalomból megcsinálta.
17      route add -net 127.0.0.0 netmask 255.0.0.0 dev lo
18      route add default gw $ATJARO
19      ;;
20  stop)
21      # A hálózati csatolók kikapcsolása
22      ifconfig eth0 down
23      ifconfig lo down
24      ;;
25  *)
26      echo "Használat:"
27      echo " $(basename $0) [ start | stop ]"
28 esac
```

A program magától értetődő módon indítja és leállítja a hálózatot. Meg kell jelezünk, hogy a gyakorlatban használt, a GNU/Linux terjesztésekben található hasonló célú BASH programok sokkal összetettebbek, kifinomultabbak.

4.2. A számítógépek nevei

A számítógépek számára a kapcsolattartás eszközeként tökéletesen megfelelnek az IP címek, amelyek egyértelműen azonosítják az egyes gépeket és az útválasztó táblázat alapján lehetővé teszik a csomagok útjának megtervezését. A felhasználóknak azonban kissé nehézkes volna a sok IP cím megjegyzése, és a mozi filmek végén is meglehetősen bután mutatnának, ezért a felhasználók kedvéért a szabványok lehetővé teszik, hogy az egyes számítógépeket nevekkel lássuk el.

A kezdeti időben, amikor még az Internetet sem Internetnek hívták, a számítógépek neveinek nyilvántartására szöveges állományokat használtak. A rendszer-gazda feladata volt, hogy egy új számítógép megjelenéskor annak nevét a szöveges állományban elhelyezze, és az állományt minden egyes számítógépre felsolja. Ezt a módszert az otthoni számítógép-hálózatunkban még ma is használhatjuk, de nyilvánvaló, hogy egy világméretű hálózat esetében ennél korszerűbb módszerre van szükség.

Az Interneten a számítógépek neveinek nyilvántartására a DNS-t (*domain name system*, tartománynévrendszer) használjuk, amelynek működését a vonatkozó szabványok[94, 95] írják le. A tartománynévrendszer működtetéséhez világszerte kiszolgálókat üzemeltetnek. Ezek szolgáltatásait gyakorlatilag az Internetet használó minden személy igénybe veszi, ezért nyilvánvalón fontos, hogy működésükkel és üzemeltetésükkel megismerkedjünk.

A tartománynévrendszer – ahogyan a neve is mutatja – az Interneten található számítógépeket névtartományokba osztja, és minden névtartomány számára kijelöl egy felelős rendszergazdát. A tartománynévrendszer által használt felosztás és az IP címek által használt földrajzi felosztás nem feltétlenül fedi egymást. Ha például nemzetközi vállalatot alapítunk, a számítógépek IP címei a földrajzi elhelyezkedésüknek megfelelően alakulnak, de ettől még a vállalat által használt tartománynév lehet egységes.

A tartománynévrendszer felépítése hierarchikus, a megfelelő alá- és fölérendelt-ségi viszonyok szigorú betartásával épül fel. A hierarchia legfelső része, a tartománynevek által alkotott fa gyökere, melynek neve „..”.

A . alatt található a tartománynevek első szintje (*top level domain*, első-szintű tartomány). A első-szintű tartománynevek a legtöbb esetben egy-egy országhoz tartoznak (például hu és pl, amely Magyarországhoz, illetve Lengyelországhoz tartozik), de vannak más jellegű első-szintű tartománynevek is, például az org a non-profit szervezetek.

Az első szintű tartományért felelős személyek feladata, hogy az alájuk tartozó másodszintű tartományok (*second level domain*, másodszintű tartomány) nevét úgy osszák ki, hogy minden névnek egy gazdája legyen. Nyilvánvaló, hogy a harmadszintű tartományokért felelős személyek osztják ki a negyedi szint felelősei számára a munkát, mégpedig minden nevet csak egyszer, ismétlődések nélkül.

Minden rendszergazda jogosult arra, hogy számítógépeket is bejegyezen a hozzá tartozó tartományba, bármelyik szinten is legyen a felelőssége alá tartozó tartomány. A számítógépek bejegyzésekor szintén egyértelműen kell a nevet az adott géphez – az IP címhez – rendelni, ami biztosítja, hogy a számítógépek tartománynével kiegészített nevéhez minden egyértelműen meghatározható az IP cím.



Ez az elrendezés nagyon hasonlít a könyvtárak, alkönyvtárak és állományok rendszerére. Az állományrendszerben minden könyvtárban egyértelműen azonosítja a könyvtárbejegyzést a neve – nem lehet ismétlődés –, ezért ha leírjuk a teljes elérési utat, az állományt egyértelműen azonosítottuk.

A tartománynevek ilyen jól megszerkesztett hierarchiája mellett már igen könnyű megvalósítani az adatbázisban található nevek visszakeresését. minden rendszergazda, aki valamely tartományért felelős, DNS kiszolgálót üzemeltet¹². Az adatbázisban való keresés során a keresett számítógép nevét a munkaállomás felbontja hierarchiaszintek mentén, megkérdezi az elsőszintű DNS kiszolgálótól, hogy hol található az adott névű felelős másodszintű kiszolgáló, attól megtudja, hogy melyik címen érhető el a keresett nevű harmadszintű kiszolgáló és így tovább¹³.

¹²Egy DNS kiszolgáló természetesen több tartománynak is lehet a felelőse, és egy tartományt több kiszolgáló is nyilvántarthat, hogy legyen tartalék.

¹³A DNS keresés persze ennél valójában némi képp bonyolultabb, de a lényeg szempontjából ez mellékes.

A következő oldalakon arról olvashatunk, hogyan tudjuk a számítógépek neveit és IP címeit egymáshoz rendelni szöveges adatbázisokat használva, miképpen lehetjük igénybe DNS kiszolgálók szolgáltatásait, és hogyan tudunk DNS kiszolgálót üzemeltetni saját tartományunk nevei számára.

4.2.1. A névfeloldás beállítása

A GNU C programkönyvtár tartalmazza azokat az eszközöket, amelyek a számítógépek neveinek kezeléséhez szükségesek[81, 31]. A programkönyvtár tartalmaz olyan függvényeket, amelyek a nevek ismeretében kiderítik az IP címet és fordítva, az IP címek ismeretében kiderítik a számítógép nevét. Az alkalmazások a GNU C programkönyvtár segítségével kezelik a számítógépek neveit, a helyes névfeloldáshoz tehát a programkönyvtárat kell beállítanunk, ami meglehetősen egyszerű.

A GNU/Linux rendszereken általában a `/etc/nsswitch.conf` állományban állíthatjuk be, hogy a számítógépek neveit honnan kérdezze le a GNU C programkönyvtár. A legtöbb terjesztésben nem kell módosítani ezt az állományt, mert a telepítés után általánosan használható beállításokat tartalmaz.

A következő részlet a `/etc/nsswitch.conf` állománynak azt a részét mutatja be, amely a számítógépek neveire vonatkozik:

```
1 hosts:      files dns
2 networks:   files
```

Az állományrészletből azt olvashatjuk ki, hogy a számítógépek neveit (`hosts`) a helyi állományokból (`files`) és – ha a keresett adat ott nem található – a DNS kiszolgálótól (`dns`) kell lekérdezni, míg a hálózatok neveit (`networks`) kizárolag a helyi állományokból. Ezek a beállítások a legtöbb helyen módosítás nélkül használhatók.

Ha mégis módosítani szeretnénk a névfeloldási stratégián, a `hosts` és a `networks` kulcsszavak után a következő kulcsszavakat írhatjuk be:

db Az adatok olvasása helyi adatbázis-állományokból. Az adatbázis-állományok kezelése gyorsabb, mint a szöveges állományoké, de a gyakorlatban általában olyan kevés adatról van szó, hogy nem érdemes adatbázis-állományokat használni.

files Az adatok olvasása helyi szöveges állományokból. Ezt a lehetőséget általában arra használjuk, hogy a legfontosabb adatokat akkor is elérje a számítógéünk, ha a hálózat nem elérhető.

nisplus Az adatok olvasása a NIS+ kiszolgálóról. Nagyon ritkán használjuk a NIS+ kiszolgálót számítógépek és hálózatok neveinek tárolására.

nis Az adatok olvasása NIS kiszolgálóról. A NIS kiszolgálót is ritkán használjuk számítógépek és hálózatok neveinek tárolására.

dns Az adatok olvasása a DNS kiszolgálóról. A számítógépek neveit a legtöbb esetben DNS kiszolgáló segítségével kezeljük.

4.2.2. Névfeloldás szöveges állományokból

A legtöbb GNU/Linux terjesztés a helyi szöveges állományokban tárolt számítógépneveket (és esetleg hálózatneveket) arra használja, hogy a névfeloldás akkor is működjön, ha a hálózat vagy a DNS kiszolgáló nem elérhető. A rendszerindításkor például, amikor még nem működik a hálózat, a számítógépünk csak a helyi állományokban tárolt számítógépneveket tudja elérni, és ehhez még csak tönkre sem kell mennie semminek. (Bizonyos nevek feloldására akkor is szükség lehet, ha a számítógépünk egyáltalán nem kapcsolódik számítógép-hálózathoz.)

A /etc/hosts állomány a névfeloldás szempontjából a legfontosabb szöveges adatállomány, amely számítógépek neveit és címeit tartalmazza. A szöveges állományban az egyes oszlopok jelentése a következő:

1. Az első oszlop tartalmazza a számítógép IP címét.
2. A második oszlop tartalmazza a számítógép hivatalos nevét.
3. A harmadik és további oszlopok tartalmazzák az adott számítógép másodlagos neveit.

A következő példa a /etc/hosts állományt mutatja be.

39. példa. Ha megvizsgáljuk a számítógépünkön a /etc/hosts állományt, általában megtaláljuk benne a 127.0.0.1 IP címhez tartozó nevet:

	/etc/hosts
1	#
2	# A 127.0.0.1 IP cím nevét mindig el kell érnünk.
3	#
4	127.0.0.1 localhost.localdomain localhost

Figyeljük meg, hogy az első oszlopban található IP cím után a hozzá tartozó neveket találhatjuk szóközzel elválasztva.

Természetesen szabadon bővíthetjük az állományt, tetszőleges IP címeket és neveket írhatunk bele. Ha az állományban elhelyezünk egy nevet, azt azonnal használhatjuk az összes hálózatot kezelő alkalmazással és segédprogrammal. Kis méretű hálózatok esetén ez elegendő is szokott lenni, azaz a névfeloldáshoz nem kell mást tennünk, mint az összes számítógépen elhelyezni az összes számítógép nevét és IP címét a /etc/hosts állományban. A legszerencsébb, ha egy helyen elkészítjük és kipróbaljuk az állományt, majd felmásoljuk az összes többi számítógépre.

A `/etc/hosts` állomány példájára könnyen megérthető a `/etc/networks` állomány felépítése is, amely az alhálózatok neveit tartalmazza. Ez az állomány nem a tartománynevek tárolására használatos, hanem az IP címtartományokat nevezhetjük el a segítségével. Olyan programok használják a `/etc/networks` állományban található nyilvántartást, amelyek címtartományokat kezelnek.

A `/etc/networks` állományban az oszlopok jelentése a következő:

1. Az első oszlop a hálózat neve.
2. A második oszlop a hálózat címe.

A következő példa bemutatja, miképpen adhatunk nevet a hálózati címtartományoknak.

40. példa. Elhelyeztük a következő sorokat a `/etc/networks` állományban:

```

1  #
2  # Hálózatok listája
3  #
4  kari_halo  169.254.0.0
5  D_epulet   172.17.0.0
6  localnet   127.0.0.0

```

`/etc/networks`

Ha most a `route` parancsot a `-n` kapcsoló nélkül használjuk, a hálózati címek helyett a hálózatok neve jelenik meg:

```

$ route
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref Use Iface
kari_halo      *              255.255.0.0  U       0      0    0    eth0
D_epulet        *              255.255.0.0  U       0      0    0    eth0
localnet        *              255.0.0.0   U       0      0    0    lo
default         router.szk   0.0.0.0     UG      0      0    0    eth0
$ 

```

Amint látjuk, a hálózatok neveit a `route` program a C programkönyvtár segítségével azonnal használatba vette.

4.2.3. A számítógép saját neve

A Linux rendszermag nyilvántartja a számítógép nevét, amelyen fut. A rendszermag által nyilvántartott név általában megegyezik a számítógép hálózaton nyilvántartott nevével, de lehetnek különbségek is. Például előfordulhat, hogy a számítógépnek több neve is van, esetleg több ponton is csatlakozik a számítógéphálózathoz. A rendszermag ilyen esetben is egy nevet tart nyilván. A rendszermag által nyilvántartott nevet sok alkalmazás használja arra, hogy a számítógépet azonosítsa, ezért meg kell ismerkednünk azzal, hogyan tudjuk lekérdezni és megváltoztatni.

4.7. tábla: hostname

A számítógép nevének lekérdezésére használható program.

`hostname [kapcsolók]`

A program segítségével lekérdezhettük és a szabványos kimenetre írathattuk a rendszermag által nyilvántartott számítógépnevet és az egyéb, azonosítására használatos neveket.

Kapcsoló	Jelentés
-d	A számítógép tartományneve.
-f	A számítógép teljes neve.
-i	A számítógép IP címe (vagy IP címei).
-s	A számítógép neve.

A számítógép neve a `hostname` programmal vagy az `uname` program -n kapcsolójával, esetleg a `/proc/sys/kernel/hostname` szimulált állomány olvasásával kérdezhettő le.

A számítógép nevét (a rendszermag által nyilvántartott nevet) átírhatjuk a `/proc/sys/kernel/hostname` szimulált állomány írásával. Az állományban a számítógép teljes nevét (gépnév és tartománynév) kell elhelyeznünk.

41. példa. Változtassuk meg a számítógép rendszermagban tárolt nevét!

```
$ echo "twinstar.hu" >/proc/sys/kernel/hostname
$
```

A számítógép nevének hirtelen megváltoztatása fennakadásokat okozhat a számítógép működésében. A grafikus felületre bejelentkezve például előfordulhat, hogy a számítógép nevének megváltoztatása után nem tudunk új ablakokat nyitni, mert a grafikus felület saját biztonsági rendszerét megzavarja a névváltozás.

A számítógép nevét általában a rendszerindító héjprogramok állítják be a rendszerindítás elején, hogy a névváltozás ne okozzon problémát.

4.2.4. A névkiszolgáló használata

Amint láttuk, ha a `/etc/nsswitch.conf` állományt megfelelő módon töltjük ki, a számítógépek nevéinek lekérdezésére DNS kiszolgálót használhatunk.

A DNS kiszolgálóval való kapcsolattartást a C programkönyvtár elvégzi az alkalmazások számára, nekünk csak meg kell adnunk a szükséges adatokat a DNS kiszolgáló eléréséhez.

A DNS kiszolgálóra vonatkozó adatokat a `/etc/resolv.conf` állomány tartalmazza. Az állományban a következő kifejezéseket használhatjuk:

nameserver cím A DNS kiszolgáló címének megadása, ami egyértelműen az általunk létrehozott legfontosabb feladata. Ez az egyetlen kifejezés, amelynek legalább egyszer szerepelnie kell a /etc/resolv.conf állományban, hogy a névfeloldás működjön.

Egymás után többször beírva ezt a kifejezést, a számítógép számára több DNS kiszolgálót is megadhatunk, így remélhetőleg akkor is működni fog a névfeloldás, ha az egyik kiszolgáló éppen nem érhető el.

search tartománynév A kifejezés segítségével megadhatjuk, hogy a részlegesen megadott számítógépneveket melyik tartományban keresse a névfeloldás során a C programkönyvtár, így megtakaríthatjuk a hosszú tartománynevek begépelezésének fáradtságát.

A programkönyvtár a search kulcsszó után írt tartománynevet kereséskor a név után másolja, és megkísérli úgy megkeresni a számítógép nevét a DNS kiszolgálón. Ha a tartománynével kiegészített nevet a DNS kiszolgáló nem találja, a programkönyvtár kiegészítés nélkül is megpróbálja lekérdezni a nevet. (Ha a felhasználó nem akarja, hogy a kereséskor az itt meghatározott tartományban keressen először a számítógépe, egyszerűen csak el kell helyeznie a „.” karaktert a begépelt név után.)

option ndots:n A kifejezés segítségével megadott szám meghatározza, hogy hány pontnak kell lennie a keresendő névben ahhoz, hogy a keresést előbb kiegészítés nélkül kísérleje meg a programkönyvtár. Alapértelmezés szerint ennek értéke 1, azaz ha a keresett számítógépnévben található legalább egy pont, a keresést előbb kiegészítés nélkül kíséri meg a programkönyvtár.

option timeout:n A kifejezés megadja, hogy hány másodpercet várjon a DNS kiszolgáló válaszára a C programkönyvtár, mielőtt feladja és a következő DNS kiszolgálóhoz fordul.

option rotate A kifejezés hatására a C programkönyvtár a beállított DNS kiszolgálók közül minden lekérdezéskor másikat keres fel.

Amint látható, a DNS kiszolgáló címének beállítása meglehetősen egyszerű: elelegendő egyetlen sort elhelyezni a /etc/resolv.conf állományban, és a névkiszolgálót már használhatjuk.

42. példa. Vizsgáljuk meg számítógépünkön a névkiszolgálóra vonatkozó beállításokat!

```
$ cat /etc/resolv.conf
search szk
nameserver 172.17.1.1
nameserver 172.17.1.11
$
```

Amint látjuk, két DNS kiszolgálót használunk, az alapértelmezett tartománynév pedig a .szk a keresések során.

4.3. A beállítások automatikus lekérdezése

A DHCP (*dynamic host configuration protocol*, dinamikus számítógép-beállító protokoll) [41] lehetővé teszi, hogy a számítógép automatikusan lekérdezze a hálózati beállításokat a hálózaton üzemeltetett DHCP kiszolgálótól. Ennek a szolgáltatásnak a segítségével IP cím, alhálózati maszk és alapértelmezett átjáró megadása nélkül indíthatjuk a hálózatot, mert az adatokat a DHCP ügyfél a DHCP kiszolgáló közbenjárásával automatikusan kideríti és beállítja. (A DHCP kiszolgálóról bővebben olvashatunk a 5.5. szakaszban a 150. oldalon.)

Az egyes GNU/Linux gyűjteményeket különféle DHCP ügyfélprogramokkal is használhatjuk. A rendelkezésre álló DHCP ügyfelek egyike, az *Internet Systems Consortium DHCP Client*, amely a dhclient programmal indítható. A következő példa ennek a programnak a használatát mutatja be:

43. példa. A dhclient programnak paraméterként megadhatjuk, hogy melyik hálózati csatolót szeretnénk elindítani a segítségével. Indítsuk el a programot, és annak segítségével az eth0 hálózati csatolót!

```
$ dhclient eth0
Internet Systems Consortium DHCP Client V3.0.1rc14
Copyright 2004 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/products/DHCP

Listening on LPF/eth0/00:02:3f:15:34:bb
Sending on LPF/eth0/00:02:3f:15:34:bb
Sending on Socket/fallback
DHCPCDISCOVER on eth0 to 255.255.255.255 port 67 interval 7
DHCPOffer from 10.1.1.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 10.1.1.1
bound to 10.1.1.100 -- renewal in 16985 seconds.
$
```

Amint látjuk, a program a 10.1.1.100 IP címet foglalta le a DHCP kiszolgáló segítségével.

A DHCP kiszolgáló valójában nem egyszerűen csak információkat szolgáltat a hálózat használatához, hanem pontos nyilvántartást is vezet arról, hogy az egyes munkaállomások számára milyen adatokat szolgáltatott. A TCP/IP hálózatokon – mint ahogyan azt már említettük – az IP címeket egyszerre csak egy munkaállomás használhatja. Amikor a DHCP kiszolgáló egy IP címet ad a DHCP ügyfélnek, jelzi azt is, hogy a cím meddig használható.

A dhclient program a /var/lib/dhcp/dhclient.leases állományban tárolja a DHCP kiszolgálótól kapott adatokat, itt rögzíti, hogy milyen címet kapott és azt meddig használhatja. Amikor a program a DHCP kiszolgálótól a hálózati adatokat megkapta, a hálózatot beállította, és az adatokat a /var/lib/dhcp/dhcp.leases

állományban rögzítette, a háttérben tovább fut. A dhclient automatikusan aktiválódik, amikor a DHCP kiszolgáló által biztosított beállítások lejárnak, hogy azokat frissítse.

A dhclient alapértelmezés szerinti beállítóállománya a /etc/dhclient.conf, amelyben a program működése rugalmasan meghatározható. A legtöbb esetben azonban nincs szükség arra, hogy ezt az állományt módosítsuk vagy létrehozzuk, a dhclient alapértelmezett működése a legtöbb környezetben használhatóvá teszi.



A dhclient program beállítóállománya a /etc/dhclient.conf, alapértelmezett adatállománya pedig a /var/lib/dhcp/dhcp.leases, de ezektől az állományoktól az egyes GNU/Linux terjesztések eltérhetnek, így lehetséges, hogy a beállításokat nem itt találjuk.

A dhclient a hálózati eszközök beállításához, a kapott hálózati adatok érvényesítéséhez a dhclient-script programot használja, amelyet az adott terjesztésre jellemző módon készítenek el. Az esetek túlnyomó többségében nincs szükség arra, hogy ezt a héjprogramot megváltoztassuk.

4.4. A hálózat beállítóállományai

Az előző oldalakon arról olvashattunk, hogy különféle programok segítségével hogyan helyezhetjük üzembe, miképpen állíthatjuk be a hálózatot. Az egyes GNU/Linux terjesztéket természetesen felszerelik a hálózatot elindító, beállító és leállító héjprogramokkal, amelyek ezeket a feladatokat automatikusan elvégzik. A következő oldalakon arról olvashatunk, hogy az egyes GNU/Linux terjesztések hol és milyen formában tárolják a beállítóállományokat, amelyek alapján a héjprogramok a hálózat beállítását elvégzik a rendszerindítás során.

4.4.1. Red Hat alapú terjesztések

A Red Hat alapú terjesztésekben a /etc/init.d/network BASH program indítja és állítja le a hálózatot a start és stop kapcsolók hatására. Ez a BASH program betölti és végrehajtja a /etc/sysconfig/network állomány tartalmát, amelyben a BASH nyelvtani szabályainak megfelelő értékadásokat helyezhetünk el.

A /etc/sysconfig/network állományban használható értékadások közül a legfontosabbak a következők:

NETWORKING=yes|no Az értékadással meghatározhatjuk, hogy használni kívánjuk-e a hálózatot. Ha a változó értéke yes, a hálózatot használjuk, ha no, a hálózat ki van kapcsolva.

Ezt a változót a legtöbb esetben yes értékre állítjuk, még akkor is, ha a számítógép nincs számítógép-hálózathoz kapcsolva.

HOSTNAME=név Az értékadással a számítógép nevét adhatjuk meg. Ha a számítógép állandó jelleggel számítógép-hálózathoz kapcsolódik, a számítógép teljes nevét (tartománynévvel együtt) adjuk meg itt, ha pedig csak néha kapcsolódunk rá a hálózatra, használjuk a `localhost.localdomain` kifejezést (a hálózati adatok automatikus megállapításakor ezt a nevet a rendszer felülírálja).

GATEWAY=cím Az értékadással az alapértelmezett átjáró IP címét adhatjuk meg. Ha a hálózati adatok automatikus megállapítása mellett döntünk, az értékadást kihagyhatjuk, mivel ilyen esetben az átjáró címét is automatikusan állapítjuk meg.

GATEWAYDEV=csatolónév A hálózati csatoló neve, amelyen keresztül az alapértelmezett átjáró elérhető (például `eth0`, `eth1`).

Ezt a változót a legtöbb esetben nem szükséges megadni.

NISDOMAIN=név A NIS tartomány neve (lásd 242. oldal).

Ezt a változót csak akkor kell beállítani, ha a felhasználói adatok központi kezelésére a NIS rendszert használjuk.

A hálózatot indító `/etc/init.d/network` héjprogram a beállítások beolvasása után sorra veszi a `/etc/sysconfig/network-scripts/` könyvtárban található `ifcfg-csatolónév` állományokat, és az `ifup` segítségével sorra elindítja a benne megadott hálózati csatolókat. A hálózati csatolók tulajdonságainak leírására az `ifcfg-csatolónév` állományokban a következő értékadásokat használhatjuk:

NAME=teljes_név A hálózati csatoló, amelyet a különböző programok a felhasználóval való kapcsolattartás során használnak. Ezt a nevet nem szükséges megadnunk, csak kényelmi szerepet tölt be.

DEVICE=csatolónév A hálózati csatoló neve (például `eth0`, `eth1`).

IPADDR=IP_cím A hálózati csatolóhoz tartozó IP cím. Ha a hálózati adatok automatikus megállapítását használjuk, ezt az értékadást elhagyhatjuk, egyébként nem.

NETMASK=maszk Az alhálózati maszk. A hálózati adatok automatikus megállapítása esetén ez az értékadás elhagyható.

ONBOOT=yes|no A változó segítségével azt adhatjuk meg, hogy a hálózati csatoló automatikusan elinduljon-e a rendszerindításkor. Ha a változó értéke `no`, a rendszerindításkor ez a csatoló nem indul el, kézzel kell elindítanunk.

USERCTL=yes|no A változó segítségével azt állíthatjuk be, hogy ezt a hálózati csatolót a felhasználók elindíthatják és leállíthatják-e. Ha a változó értéke `yes`, a hálózati csatolót a felhasználók leállíthatják és el is indíthatják.



4.11. ábra. A hálózat beállítása Fedora Core 2 rendszeren

BOOTPROTO=none|bootp|dhcp A változó segítségével beállíthatjuk, hogy a hálózati adatokat automatikusan akarjuk-e felderíteni. Az értékként használható kulcsszavak jelentése a következő:

none Nem használjuk a hálózati adatok automatikus megállapítását, a hálózatot a beállítóállományokban megadott adatok alapján indítjuk el.

bootp A hálózati adatok automatikus megállapításához a BOOTP protokollt használjuk.

dhcp A hálózati adatok automatikus megállapításához a DHCP protokollt használjuk.

A DHCP csereszabatos a BOOTP protokollal, de annál korszerűbb, így a legtöbb esetben a **dhcp** beállítást használjuk.

Ha az értékadásban a **bootp** vagy a **dhcp** kulcsszót használjuk, az állományban csak a **DEVICE** változónak kell értéket adnunk, a többi a rendszer automatikusan kezeli.

Amint látható, a hálózat működéséhez szükséges adatok megadása nagyon egyszerű. Ennek ellenére minden Red Hat alapú rendszer rendelkezik olyan grafikus hálózatbeállító programokkal, amelyek a hálózat beállításában segítenek. A 4.11. ábrán láthatjuk például a **system-config-network** program egyik ablakának képernyőképét. A program az előző oldalakon bemutatott beállításokat is kezeli.

4.4.2. Debian alapú terjesztések

Debian alapú GNU/Linux rendszereken a hálózat indítását és leállítását a `/etc/init.d/networking` BASH program végezi. A program a hálózat indításakor elindítja az `ifup` programot a -a kapcsolóval, hogy az az összes hálózati csatolót elindítsa. A hálózat leállítása hasonlóan történik az `ifdown` program -a kapcsolójával.

Az `ifup` és `ifdown` programok a hálózati eszközök kezelését a `/etc/network/interfaces` beállítóállomány alapján végzik. Ha tehát be akarjuk állítani a hálózati eszközöket, ezt az állományt kell szerkesztenünk.

A `/etc/network/interfaces` állományba a szokásos módon – a # karakterrel – írhatunk megjegyzéseket. Az állományban – többek közt – a következő kifejezéseket használhatjuk a hálózati eszközök beállítására:

auto csatoló1 csatoló2 Az állományban található `auto` kulcsszó után felsorolt hálózati csatolók a rendszerindításkor automatikusan elindulnak.

A hálózati csatoló vagy csatolók nevének – `eth0`, `eth1` stb. – az `auto` kulcsszóval egy sorban, szóközzel elválasztva kell sorakozniuk.

iface csatoló család indítás Az `iface` kulcsszóval készített kifejezések segítségével az egyes hálózati csatolókat állíthatjuk be. A kifejezés utáni sorokban további kifejezések sorakozhatnak a csatoló beállítására.

A kifejezésben található `csatoló` a hálózati csatoló nevét (például `eth0`, `eth1` stb.) adja meg. A beállítóállományban több helyen szerepelhet a `iface` kulcsszóval felépített kifejezés, mindegyikben más-más csatoló beállításai-val.

A kifejezésben szereplő `család` a hálózati protokollcsaládot adja meg, amelyet a hálózati csatoló használ. A legtöbb esetben itt a `inet` kulcsszó áll, amely az IPv4 protokollcsaládot jelöli, de más kulcsszavak is a rendelkezésünkre állnak.

A kifejezésben szereplő `indítás` meghatározza, milyen módon kívánjuk megadni a hálózati csatoló beállításait. A következő kulcsszavakat használhat-juk:

static Ezzel a kulcsszóval azt jelezhetjük, hogy a hálózati csatoló által használt beállításokat a beállítóállományban kívánjuk megadni a `iface` kulcsszóval létrehozott kifejezés után.

dhcp Ezzel a kulcsszóval jelezhetjük, hogy a hálózati csatoló beállításait a DHCP protokoll szerint akarjuk lekérdezni a hálózati csatoló indításakor.

A `iface` kifejezés után ebben az esetben is megadhatunk beállításokat.

bootp A kulcsszóval jelezhetjük, hogy a beállításokat a DHCP protokollhoz hasonló, annál régebbi BOOTP protokoll szerint kívánjuk lekérdezni a hálózatról.

loopback Ezzel a kulcsszóval jelezhetjük, hogy a hálózati csatoló a számítógépet saját magával összekötő szimulált csatoló.

Bizonyos esetekben már az eddig bemutatott egyszerű kifejezésekkel is beállíthatjuk a hálózati csatolókat. Ezt mutatja be a következő példa.

44. példa. Készítsük el a /etc/network/interfaces beállítóállományt két hálózati csatoló számára. Az lo csatoló legyen egy szimulált hálózati csatoló, amely a számítógépet önmagával köti össze, az eth0 pedig legyen a hálózatra kapcsolt Ethernet csatoló, amelynek beállításait DHCP szerint szerint a rendszerindításkor kérdezzük le!

```
/etc/network/interfaces
1 #
2 # Szimulált csatoló (automatikusan indul):
3 #
4 auto lo
5 iface lo inet loopback
6
7 #
8 # Ethernet csatoló (automatikusan, DHCP-vel indul):
9 #
10 auto eth0
11 iface eth0 inet dhcp
```

Figyeljük meg az állományban, hogyan gondoskodtunk a hálózati csatolók rendszerindításkor történő bekapcsolásáról (az auto kulcsszóval), és miképpen adtuk meg a két csatoló alapbeállításait (az iface kulcsszóval).

Ha az iface kulcsszóval felépített kifejezésben a megadott protokollcsalád inet vagy inet6 (azaz IPv4, illetve IPv6) és az indítás módja static, a kifejezés utáni sorokban a következő dolgokat állíthatjuk be:

address cím A hálózati csatolóhoz tartozó IP cím. Ezt a kifejezést az IPv4 és az IPv6 protokollcsalád esetében is használnunk kell (static indítási módnál).

netmask alhálózati maszk A hálózati csatolóhoz tartozó alhálózati maszk megadása. Ezt a kifejezést az IPv4 és az IPv6 protokollcsalád esetében is használnunk kell (static indítási módnál).

gateway cím Az alapértelmezett átjáró címének megadása. Ezt a kifejezést azoknál a hálózati csatolóknál kell használnunk, amelyek azzal a hálózattal kötik össze a számítógépet, amelyen az alapértelmezett átjáró is van.

broadcast cím A csatlakozó hálózathoz tartozó körüzenet küldésére használható cím. Ezt a kifejezést az IPv4 protokollcsaládot használó csatolók esetében használhatjuk, de általában nincs rá szükség, mert a körüzenetek címe kiszámítható az IP címből és az alhálózati maszkból.

network cím A hálózat címe, amellyel a hálózati csatoló összeköti a számítógépet. Ezt a kifejezést az IPv4 protokollcsaládot használó hálózati csatolók esetében használhatjuk, de általában nincs rá szükség.

(A dokumentáció szerint régebbi – 2.0.xx változatú – Linux rendszermagánál használnunk kell.)

Ezekkel a kifejezésekkel a /etc/network/interfaces állományban a hálózati csatoló beállításait megadhatjuk, így a hálózati csatoló DHCP kiszolgáló használata nélkül is elindítható. Ezt mutatja be a következő példa!

```
1  #
2  # Szimulált csatoló (automatikusan indul):
3  #
4  auto lo
5  iface lo inet loopback
6
7  #
8  # Ethernet csatoló (automatikusan indul):
9  #
10 auto eth0
11 iface eth0 inet static
12   address 10.1.1.98
13   netmask 255.0.0.0
14   gateway 10.1.1.100
```

Figyeljük meg, hogy a hálózati csatoló beállítására használt iface kulcsszó után megadtuk a legfontosabb beállításokat, így a hálózati csatoló beállítása DHCP kiszolgáló nélkül is megtörténhet.

A /etc/network/interfaces állományban további kifejezéseket is használhatunk a hálózati csatolók beállítására. Ezeknek a kifejezéseknek a használatáról a man interfaces parancs segítségével olvashatunk.

5. fejezet

Hálózati szolgáltatások

A következő oldalakon arról olvashatunk, hogy miképpen nyújtanak az egyes számítógépek szolgáltatásokat egymásnak a hálózaton keresztül. Megismерkedhetünk az alapfogalmakkal – amelyek akkor is hasznosak lehetnek, ha nem akarunk szolgáltatások nyújtásával bővíteni – és néhány eszközzel, amelyek felhasználásával könnyen hozhatunk létre és tarthatunk üzemen hálózati szolgáltatásokat.

5.1. A hálózat alapprotokolljai

Amint láttuk, az IP címek használata lehetővé teszi, hogy a számítógép-hálózatra kötött számítógépek adatcsomagokat küldjenek egymásnak a hálózaton keresztül. Láttuk, hogy a csomagok továbbítása akkor is lehetséges, ha a számítógép-hálózat világmeretű.

Az eddig bemutatott eszközök azonban nem nyújtanak megoldást jónéhány problémára, nem adnak eszközt jónéhány feladat elvégzésére. A megoldatlan kérdések közül a legfontosabbak a következők:

- A hálózatra kapcsolt számítógépek általában egyszerre több programot, több alkalmazást is futtatnak. Hogyan lehetne egy időben több hálózati kapcsolatot kezelni két számítógép között, azaz miképpen volna lehetséges a fogadó oldalon eldönteni egy csomagról, hogy azt melyik alkalmazás kapta?
- A hálózati csomagok mérete korlátozott, az adatátvitel során pedig szeretnénk tetszőleges adatmennyiséggel dolgozni. Hogyan volna lehetséges az átvienő adathalmazt darabokra szabdalni a küldő oldalon és újra összeállítani a fogadó oldalon?
- A hálózat működése általában esetleges, hiszen az adatcsomagra a földrésekkel is átívelő útjuk során mindenféle veszélyek leselkednek. Szeretnénk

az adatok biztonságáról – amennyire csak lehetséges – gondoskodni. Hogyan volna lehetséges érzékelni, az adatcsomagok eltúnését és sérülését, és az adatok újraküldésével javítani a hibát?

A felsorolt kérdésekre az IP csomagok továbbítását leíró és megvalósító rendszerek nem képesek, ez azonban nem jelenti azt, hogy az IP protokollok és azok megvalósítása nem használható fel az ilyen megoldásokban. Fontos megértenünk, hogy az IP csomagokat továbbító rendszer nyújtja az alapot e három fontos kérdés megvalósítására született eszközök számára. A kérdéseinkben megfogalmazódó kihívásra válaszul olyan protokollok (és programok) születtek, amelyek az IP csomagok továbbítására létrehozott eszközöket használják. A következő oldalakon bemutatott eszközök a gyakorlatban olyan programok, amelyek az IP csomagokat továbbító programokra épülve további szolgáltatásokat valósítanak meg, azokat a szolgáltatásokat, amelyek választ adnak kérdéseinkre.

Vegyük most sorra, hogy milyen módszereket használhatunk a több egyidejű kapcsolat, a korlátlan adatmennyiség és a hibatűrő adatátvitel megvalósítására!

A több egy időben történő átvitel megvalósítható, ha minden egyes IP csomaghoz további azonosítószámokat csatolunk, amelyek megadják, hogy melyik alkalmazás számára küldte a távoli számítógép a csomagot. A küldő számítógép elhelyezi a csomagban, hogy a címzett számítógépen melyik alkalmazás számára küldi a csomagot, a fogadó oldalon pedig egyszerűen kézbesítjük a megfelelő alkalmazásnak a beérkezett adatokat.

A gyakorlatban elterjedt módszer szerint minden csomaghoz rendelünk két számot, amelyeket küldő és fogadó hálózati kapunak (*port*, *kapu*, *por*) nevezünk. A több egy időben történő adatátvitelt megvalósító eszközök tehát szimmetrikus elrendezést használnak, a küldő és a fogadó oldalon is tartozik egy azonosítószám (hálózati kapu) az adatcsomaghoz, de ez a módszerünk lényegét nem érinti. minden csomaghoz azonosítót rendelünk, így minden esetben szét tudjuk választani, hogy az egyes csomagok melyik alkalmazás számára érkeztek.

A tetszőleges mennyiségű adat átvitelét igen könnyen megvalósíthatjuk. Egy-szerűen szét kell szabdalni az átvendő adatmennyiséget darabokra, az ellenállomáson pedig újra össze kell illesztenünk őket. Így a nagyobb adatmennyiség átviteléhez több csomagot használunk, és hosszabb ideig is tart, de a csomagmérőt nem korlátozza az adatátvitelt.

Kissé bonyolítja a helyzetet, hogy az IP címekkel ellátott csomagok sorrendje az adatátvitel során megváltozhat. Előfordulhat, hogy az elsőként elküldött csomag útja tovább tart, ezért másodikként vagy harmadikként érkezik meg. Ha egyszerűen csak egymás után másolnánk a beérkezett csomagokat, könnyen lehet, hogy az elküldött adatok jócskán összekeverednének, ezért megoldást kell találnunk a csomagok helyes sorrendbe való rendezésére. A megoldás ismét csak egyszerű. Nem kell mást tennünk, mint minden csomaghoz egy sorszámat rendelni, és a csomagokban elhelyezett sorszám alapján a fogadó oldalon helyre kell állítani a sorrendet. Ha a hálózaton az átvitel során a csomagok összekeverednek, a fogadó számítógép helyreállítja a sorrendjüket a bennük található sorszám alapján.

Vizsgáljuk meg most, hogy mit tehetünk a csomagok biztonsága érdekében! Első pillantásra nyilvánvaló, hogy nem tudunk olyan programot készíteni, amely garanciát vállalna az adatátvitel működőképességéért. Ha az adatátvitelre használt kábel elszakad – és nincs más útvonal –, a továbbítást végző számítógép tönkremegy – és nincs ami kiváltaná –, az adatátvitel meg fog szakadni, és ezen semmilyen program nem segíthet.

A biztonság érdekében azonban megtehetjük, hogy az átvitelt egy válaszcsomagban adott visszajelzés segítségével ellenőrizzük. A válaszcsomagban a fogadó igazolhatja a csomag vételét, és a sikeres ellenőrzésre leírhatja a csomag jellemzőit. Ha a küldő úgy találja, hogy a visszaigazolásban a csomag jellemzői helytelenek, tudja, hogy a csomag megsérült, és újra elküldi. Úgyszintén újra próbálja küldeni a csomagot a küldő, ha a válasz egy bizonyos időn belül nem érkezik meg, hiszen ekkor feltételezheti, hogy útközben elkallódott.

Ilyen módon tehát hibaellenőrző és hibajavító eszközöket építhetünk az IP csomagok küldésére használt rendszerre, és a kezelhető, javítható hibákkal nem az alkalmazásoknak kell foglalkozniuk.

A több egyidejű kapcsolatra, nagy mennyiségű adat átvitelére és hibakezelésre három fontos protokollt használunk, amelyek mindegyike IP csomagok továbbításán alapul. Ezeket a protokollokat mutatjuk be a következő oldalakon.

5.1.1. Az UDP protokoll

Az UDP (*user datagram protocol*, felhasználói adatcsomag protokoll) az IP csomagok átvitelén alapuló protokoll, amelyet sok alkalmazás használ adatátvitelre [120]. Annak érdekében, hogy az általánosan használt UDP protokoll megkülönböztessük a hálózati alkalmazások működésének leírását tartalmazó protokolloktól, az UDP-t a hálózat tárgyalása során néha alapprotokollként említjük.

Az UDP alapprotokoll főbb tulajdonságai a következők:

1. A protokoll lehetővé teszi a többszálú adatátvitelt, és a csomagok azonosításához a küldő és címzett számítógépeken hálózati kaput rendel.

A hálózati kapuk az adatátvitelben részt vevő folyamatokat azonosító két-bájtos címek, amelyek lehetővé teszik, hogy a fogadó oldalon a címzett folyamatot azonosítsuk.

2. A protokoll nem teszi lehetővé nagyobb adatmennyiségek átvitelét, azaz nem tartalmaz olyan eszközöket, amelyek a csomagok sorrendjének helyreállítására volnának használhatók.

Valójában maga a protokoll is innen kapta a nevét. A *felhasználói adatcsomag protokoll* elnevezés szerint az UDP olyan protokoll, amely lehetővé teszi a felhasználói programoknak, hogy adatcsomagokat küldjenek és fogadjanak. A számítógép-hálózatot az alkalmazások az UDP segítségével csomagok továbbítására alkalmas eszközöként használják.

3. A protokoll nem tartalmaz a hibák javítására szolgáló, válaszcsomagokon alapuló eszközöket. (Minden hálózati csomag tartalmaz ugyan ellenőrző összegeket, de ezek nem védenek például a csomagok elvesztése ellen.)

Az UDP csomagokat használó alkalmazások közt az adatátvitel általában a következő lépésekben jön létre:

1. A kiszolgálóoldalon a szolgáltatást nyújtó alkalmazás jelzi az operációs rendszernek, hogy egy bizonyos hálózati kapun UDP csomagokat szeretne fogadni, azaz el szeretné érni, hogy az összes beérkező UDP csomag, amelynek címzettjeként az adott kapu van megnevezve, hozzá kerüljön.
2. Az operációs rendszer megvizsgálja, hogy a nyilvántartása szerint ez a hálózati kapu foglalt-e, azaz van-e már olyan alkalmazás, amely igényt jelentett be rá. Ha a kapu szabad, és minden más is rendben van, az alkalmazásnak jelzi, hogy a hálózati kapu megnyitását nyilvántartásba vette.
3. Az alkalmazás a következő lépésben olvasni próbálja a kaput. Ennek következtében a legtöbb esetben az alkalmazást futtató folyamat blokkolt állapotba kerül, amíg a hálózatról nem érkezik a kapura UDP csomag.
4. Az ügyféloldalon a szolgáltatást igénybe vevő alkalmazás tudja, hogy milyen jellegű szolgáltatásra van szüksége, és azt is, hogy az adott szolgáltatást nyújtó kiszolgáló a hagyomány szerint melyik kapun várja a kéréseket.
5. Az ügyfélalkalmazás jelzi az operációs rendszernek, hogy UDP csomagot szeretne küldeni, és ehhez egy szabad kapura van szüksége a küldő oldalon. Az operációs rendszer választ egy szabad kaput, és lefoglalja azt a folyamat számára.
6. Az alkalmazás előkészít egy memóriaterületet, amely az egy csomagban elküldhető adatmennyiségnél nem nagyobb, és jelzi az operációs rendszernek, hogy az adatokat el kívánja küldeni.
7. A szolgáltatást nyújtó kiszolgáló számítógépre megérkezik az UDP csomag, a csomagra várakozó folyamat blokkolása feloldódik, az kiolvassa és feldolgozza az adatokat. Az operációs rendszer a kiszolgáló folyamat számára elérhetővé teszi a küldő számítógép IP címét és a küldött csomagon található forrás hálózati kaput. A folyamat tehát „tudja”, hogy a csomagot melyik számítógép küldte, a küldéskor melyik hálózati kaput használta.
8. Az események ezen pontján mind a kiszolgáló-, mind pedig az ügyféloldalon rendelkezésre áll egy-egy kapu, és a két, egymással kapcsolatban álló folyamat ismeri a másik oldalon használt hálózati kapu számát.

Mind a kiszolgáló, mind pedig az ügyfél küldhet UDP csomagot vagy csomagokat a másik oldalra, de természetesen semmi nem garantálja, hogy

azok a megfelelő sorrendben érkeznek meg (vagy megérkeznek-e egyáltalán) a másik oldalra.

Figyeljük meg, hogy a *kiszolgáló* és *ügyfél* kifejezések azt jelzik, hogy a folyamatok milyen szerepet töltötték be a kapcsolat kiépítésékor.

9. Mind a küldő, mind pedig a fogadó oldal dönthet úgy, hogy a kaput lezárja, a kapcsolattartásra fenntartott erőforrásokat felszabadítja.

A csomagküldés különböző fázisainak támogatására a C programkönyvtár kényelmesen használható eszközöket biztosít az alkalmazások számára, így könnyen írhatunk olyan programokat, amelyek az UDP alapprotokollt használva kapcsolatot létesítenek a hálózaton keresztül[81, 31].

5.1.2. A TCP rotokoll

A TCP (*transmission control protocol*, átvitelvezérlő protokoll) IP csomagok átvitelen alapuló protokoll[129]. A legtöbb alkalmazás, amellyel az Internet felhasználói találkoznak, a TCP protokollt használja adatátvitelre. Annak érdekében, hogy az általánosan használt TCP protokollt megkülönböztessük a hálózati alkalmazások működésének leírását tartalmazóktól, a TCP-t a hálózat tárgyalása során néha alapprotokollként emlíjtjük.

A TCP alapprotokoll főbb tulajdonságai a következők:

1. Lehetővé teszi, hogy két számítógép között egy időben több kapcsolat is létezzen. Erre a hálózati kapuk rendszerét használja, mégpedig pontosan úgy, ahogyan azt az UDP esetében láttuk.
2. Lehetővé teszi nagyobb adatmennyiséget továbbítását. A protokoll alapján a TCP kapcsolatot megvalósító IP csomagokban található egy sorszám, amely jelzi, hogy az adott csomag a kapcsolat hányadik csomagja. A sorszám alapján a fogadó oldalon a csomagok helyes sorrendje helyreállítható, a küldött adathalmaz reprodukálható.
3. A TCP protokoll gondoskodik a csomagok válaszcsomagokon alapuló ellenőrzéséről.

A TCP protokoll előírja, hogy az adatcsomagok vételét válaszcsomagok segítségével igazolni kell. Ha a küldő számítógép bizonyos időn belül nem kapja meg a visszaigazolást az adatok címzettjétől, az adatok újraküldésével pótolja az elveszett csomagokat.

A TCP kapcsolatok kiépítése az UDP kapcsolatoknál látott lépésekben megy végbe, azaz előbb a kiszolgálóoldalon nyitunk egy hálózati kaput olvasásra, majd az ügyféloldalon – egy tetszőlegesen megválasztott kapu felhasználásával – felkeressük ezt a kaput. Az UDP és a TCP kapcsolatok közti különbség a kiépült kapcsolatok használatakor jelentkezik. A TCP kapcsolatok használata során a következő fontos tulajdonságokat figyelhetjük meg:

1. A kiépített kapcsolat két oldalán található folyamatok a kapcsolatot írásra és olvasásra is felhasználhatják, az írás és olvasás során azonban tetszőleges adamennyiséggel dolgozhatnak.
A küldéskor nem kell az alkalmazásnak a csomagméréssel foglalkoznia, bízhat abban, hogy a TCP-megvalósítás az átadott adatokat a csomagmérétnek megfelelő méretűre szabdalja.
2. A TCP kapcsolat olvasása során az alkalmazás megadhatja, hogy mekkora adatmennyiséget képes fogadni, azaz a már beérkezett adatmennyiség mekkora részét képes az adott pillanatban feldolgozni.
3. Az olvasást és az írást is folytathatja az alkalmazás, azaz a nagyobb adatmennyiségeket az alkalmazások tetszőleges részekben is átvihetik, a TCP protokoll alapján történő adatátvitel során azonban a fogadó oldalon nem deríthető ki, hogy a küldő az adatokat milyen méretű részletekben indította útjára.

Ez azt jelenti, hogy ha például a küldő oldalon előbb 4, majd 8 kilobájtnyi adatot írunk a TCP kapcsolatba, a fogadó oldalon nem tudjuk kideríteni, hogy mekkora lépésben történt az adatátvitel. Lehetséges, hogy azt tapasztaljuk a fogadó oldalon, hogy 12 kilobájt érkezett meg, de az is lehet, hogy előbb 2, majd 10 kilobájt olvasására nyílik módunk.

Általánosságban elmondható, hogy a TCP protokoll a csővezetékekhez hasonló az adatátvitelre, és UNIX rendszereken az ilyen kapcsolatokat csővezetékként is használhatjuk[81].

5.1.3. Az ICMP protokoll

Az ICMP (*internet control message protocol*, internetes vezérlőüzenet protokoll) az adatátvitelt támogató protokoll, amely IP csomagok átvitelén alapul[128]. Az ICMP csomagok nem tartalmaznak az alkalmazások által egymásnak küldött adatokat, az ICMP csomagokat az operációs rendszerek használják a többi alapprotokoll használata közben az adminisztratív feladatok ellátására. Az ICMP protokoll legfontosabb tulajdonságai:

1. Az ICMP csomagok esetében nem beszélhetünk hálózati kapukról.
Az ICMP csomaggal az egyik operációs rendszer tud üzenni a másiknak, ezért nincs szükség arra, hogy egy időben több ICMP kapcsolat létezzen.
2. A szabvány nem teszi lehetővé a csomagsorrend helyreállítását, és erre nincs szükség, mert a csomagok nem adat hordozására készültek.
Valójában nem is azt mondjuk, hogy az ICMP csomagban elhelyezett adatokkal vezéreljük a hálózati kapcsolatot, hanem azt, hogy a különféle típusú ICMP csomagok más és más jelentéssel rendelkeznek, de ez nyilvánvalóan csak szóhasználat kérdése.

3. A ICMP protokoll nem használ válaszcsomagok küldésén alapuló hibaellenőrző vagy hibajavító eszközt.

A legtöbb alkalmazás nem kezel ICMP csomagokat, így a legtöbb alkalmazásprogramozó nem foglalkozik az ICMP csomagok világával. Az ICMP csomagokat az operációs rendszert készítő rendszerprogramozónak és a rendszer üzemeltetéséért felelős rendszergazdának kell ismernie.

Tekintsük át vázlatosan, hogy milyen feladatokat látnak el az ICMP csomagok a hálózati kapcsolat vezérlésében:

1. Ha egy útválasztó azt látja, hogy a számára továbbításra beérkezett csomagok címzettje nem érhető el, ICMP csomagban válaszolva erről a küldőt értesítheti.

Ha az ICMP csomagok nem érkeznek meg, a küldő alkalmazás jó darabig küldözgetné a csomagokat, abban bízva, hogy azok egyszer csak megérkeznek az ellenállomásra. (Egy idő után persze gyanússá válna a dolog, és feladná az alkalmazás, de az ICMP csomagok segítségével ez a folyamat gyorsítható.)

2. Ha az egyik számítógép csomagot szeretne küldeni a másiknak, szerencsés, ha meggyőződik arról, hogy az csomagok vételére alkalmas (be van kapcsolva, nem szakadt a kábel és így tovább). Az ellenőrzésre az operációs rendszer ICMP csomagokat használ, mivel azok kezelése egyszerű és nem foglal sok erőforrást, nem terheli túlzott mértékben a hálózatot.

Ezek természetesen csak példák, a gyakorlatban az ICMP csomagok világa sokszínűbb, izgalmasabb.

5.2. Hálózati kapuk és szolgáltatások

Az UDP és a TCP protokoll a hálózati kapuk számozására kétbájtos címeket használ, így a hálózati kapuk a 0 és 65535 közötti egész számokkal azonosíthatók. Mivel az UDP és a TCP hálózati kapuk egymástól elkülönülnek, összesen 65536 darab UDP és 65536 darab TCP hálózati kapu áll az alkalmazások rendelkezésére.

A 0 és 1023 közé eső hálózati kapuk egy részét a szabvány[140] jól ismert szolgáltatásokhoz rendeli. UNIX rendszereken a 0 – 1023 számú kapukat kiváltságos kapuknak (*privileged ports*) nevezik, és csak a rendszergazda által futtatt alkalmazásoknak engedjük a használatukat. Mivel a kiváltságos kapukat csak a rendszergazda által futtatott alkalmazások használhatják, az ügyfélprogramok bízhatnak abban, hogy a felkeresett számítógép kiváltságos kapuin a szolgáltatás szabványainak eleget tevő kiszolgálóprogramot találják. (Kivéve persze azt az esetet, amikor az ellenállomás valamelyen okból nem kíván szolgáltatást nyújtani. Az ilyen esetekben a kapcsolatfelvétel meghiúsul.)

A 1024 és 49151 közé eső hálózati kapuk nem kiváltságosak, de sokszor szintén szolgáltatások nyújtására használjuk őket. Ezeknek a kapuknak a szolgáltatáshoz

való rendelésére ajánlások álnak a rendelkezésünkre, amelyek azonban nem kötelező jellegűek. Ezeket a kapukat általában bejegyzett kapuknak (*registered ports*) nevezzük.

A 49152 és 65535 közti kapuk szabadon felhasználhatók. Ezeket a kapukat általában magáncélú kapuknak (*private ports*) nevezzük. A magáncélú kapukat tetszőleges szolgáltatás nyújtására igénybe vehetjük, és általában új kapcsolat létrehozásakor is ezeket a kapukat használja az operációs rendszer. Ez azt jelenti, hogy ezek mögött a kapuk mögött állnak az ügyfélprogramot, amelyek a választ várják a kiszolgálótól.

5.2.1. A szolgáltatások nyilvántartása

Amint láttuk, a hálózaton keresztül egymáshoz kapcsolódó alkalmazások az UDP és a TCP alaproottokollt használva a hálózati kapuk segítségével képesek felvenni egymással a kapcsolatot. A kiszolgálóprogramok az általuk nyújtott szolgáltatásnak megfelelő számú hálózati kapun várják az ügyfélprogramok jelentkezését, az ügyfélprogramok a szolgáltatásnak megfelelő kapun keresik a kapcsolatot. Nyilvánvaló, hogy a kapcsolat akkor jöhet létre zökkenőmentesen, az alkalmazások akkor „értik meg egymást”, ha minden oldal tisztában van, azzal hogy az egyes szolgáltatásokhoz milyen hálózati kapu tartozik. A különböző szolgáltatásokhoz tartozó kapuk számát szabvány rögzíti[84], hogy a szolgáltatást nyújtó és a szolgáltatást igénybe vevő oldal egymásra találhasson.

A C programkönyvtár az alkalmazások számára a szolgáltatások nyilvántartását kezelő eszközöket is biztosít. A szolgáltatások nyilvántartására a UNIX rendszereken általában szöveges állományokat használunk, amelyekből az alkalmazások a C programkönyvtár segítségével kiolvashatják az adatokat.

A /etc/services állomány a szolgáltatások nevét, és a szabvány szerint[140] hozzájuk rendelt hálózati kapuk számát tartalmazza. A következő állományrészlet a legfontosabb szolgáltatások adatait tartalmazza olyan formában, ahogy a /etc/services állományban találjuk őket:

```

1 # /etc/services: Szolgáltatás-adatbázis (részlet)
2 # Az egyes oszlopok jelentése:
3 # <név>    <kapu>/<protokoll>  <másodlagos nevek>
4 #
5 echo      7/tcp
6 echo      7/udp
7 #
8 # File Transfer Protocol
9 #
10 ftp-data 20/tcp
11 ftp-data 20/udp
12 ftp       21/tcp
13 ftp       21/udp  fsp  fspd

```

```
14 #
15 # Secure Shell
16 #
17 ssh      22/tcp
18 ssh      22/udp
19 #
20 # Telnet
21 #
22 telnet   23/tcp
23 telnet   23/udp
24 #
25 # Simple Mail Transfer Protocol
26 #
27 smtp     25/tcp  mail
28 smtp     25/udp  mail
29 #
30 # Domain Name Service
31 #
32 domain   53/tcp
33 domain   53/udp
34 #
35 # Finger
36 #
37 finger   79/tcp
38 finger   79/udp
39 #
40 # Hypertext Transfer Protocol
41 #
42 http     80/tcp  www www-http
43 http     80/udp  www www-http
44 #
45 # Post Office Protocol
46 #
47 pop3    110/tcp  pop-3
48 pop3    110/udp  pop-3
49 #
50 # Remote Procedure Call
51 #
52 sunrpc  111/tcp  portmapper
53 sunrpc  111/udp  portmapper
54 #
55 # Interactive (Interim) Mail Access Protocol
56 #
57 imap    143/tcp  imap2
```

```
58 imap      143/udp  imap2
59 #
60 # X Display Manager Control Protocol
61 #
62 xdmcp     177/tcp
63 xdmcp     177/udp
64 #
65 # IMAP v3
66 #
67 imap3     220/tcp
68 imap3     220/udp
69 #
70 # HTTP over SSL
71 #
72 https     443/tcp
73 https     443/udp
74 #
75 # SMTP over SSL
76 #
77 smtps    465/tcp
78 #
79 # IMAP over SSL
80 #
81 imaps     993/tcp
82 imaps     993/udp
83 #
84 # POP3 over SSL
85 #
86 pop3s    995/tcp
87 pop3s    995/udp
88 #
89 # X Window System
90 #
91 x11      6000/tcp  X
92 #
93 # Internet Relay Chat
94 #
95 ircd     6667/tcp
96 ircd     6667/udp
97 #
98 # X Window Font Server
99 #
100 xfs     7100/tcp
```

Amint látjuk, a `/etc/services` állományban a megszokott módon, a # karakter segítségével helyezhetünk el megjegyzéseket. Az állomány a következő információkat tartalmazza a különböző oszlopokban:

1. Az első oszlopban található a szolgáltatás közismert neve.
2. A második oszlopban található a szolgáltatáshoz tartozó hálózati kapu száma és a szolgáltatás igénybe vételéhez használt hálózati alapprotokoll rövidítése.
A szokás szerint, ha egy adott szolgáltatáshoz bejegyezzük a nyilvántartásba a TCP alapprotokolt, akkor azonnal bejegyezzük az UDP protokolt is és fordítva. A bejegyzés nem jelenti azt, hogy az adott szolgáltatást megvalósító kiszolgálóprogram valójában minden protokoll felhasználásával el is érhető.
3. A további oszlopokban a szolgáltatások esetleges további nevei – másodlagos nevei – találhatók.

A `/etc/services` nyilvántartás alapján működő programok képesek névvel hivatkozni az egyes szolgáltatásokra – így az egyes hálózati kapukra –, és képesek kideríteni az egyes kapuk nevét. Ez nyilvánvalóan megkönyíti az alkalmazások használatát és az általuk kiírt üzenetek megértését, és lehetőséget biztosít arra, hogy a szolgáltatásokhoz használt kapuk számát megváltoztassuk anélkül, hogy az alkalmazáson változtatnánk¹.

A következő lista a legfontosabb szolgáltatásokat mutatja be a `/etc/services` állományban is használt nevük alapján:

echo A szolgáltatás az összes küldött adatot egyszerűen visszaküldi változatlan formában.

Az echo szolgáltatást a számítógépek közti kapcsolatok ellenőrzésére használhatjuk.

ftp Az FTP (*file transfer protocol*, állományviteli protokoll) protokoll állományok másolását teszi lehetővé számítógépek között. Az FTP nem használ titkosítást a felhasználói nevek és a jelszavak átvitelére, ezért a legtöbb esetben csak mindenki számára elérhető adatok szolgáltatására használjuk.

Az FTP lehetővé teszi, hogy az adatátvitelt vezérlő parancsok továbbítására és az adatátvitel külön csatornán történjen, ezért a szolgáltatások között megtaláljuk a parancsok átvitelére használt **ftp**, valamint az adatátvitelre használt **ftp-data** szolgáltatást is.

ssh Az elavult **rsh** (*remote shell*, távoli héj) szolgáltatás titkosítással biztonságossá tett változata (*secure shell*, biztonságos héj), amely igen hasznos és nagyon elterjedt.

¹Ha megváltoztatjuk a `/etc/services` tartalmát, akkor persze számolunk kell a következményekkel, hiszen a világon senki sem fogja tudni, hogy melyik kapun biztosítjuk a szolgáltatásokat.

telnet A telnet (*telephone network*) szolgáltatás segítségével távolról jelentkezhetünk be a számítógépre, és a héj segítségével parancsokat hajthatunk végre. Mivel a szolgáltatás nem használ titkosítást a felhasználói nevek és a jelszavak átvitelére, ma már elavultnak tekintjük, és nem használjuk nyilvános hálózatokon.

Mivel a telnet program egyszerűen elküldi a felhasználó által lenyomott billentyűk karaktereit a kiszolgálónak, és megjeleníti a visszakapott karaktereket, sokszor használjuk más szolgáltatások vizsgálatára. Próbaképpen felvehetjük a kapcsolatot az egyes szolgáltatásokat nyújtó kiszolgálóprogramokkal a telnet program segítségével, és begépelve a szolgáltatás igénybe vételére használható parancsokat megfigyelhetjük a kiszolgáló válaszait.

smtp Az SMTP (*simple mail transfer protocol*, egyszerű levéltovábbítási protokoll) az elektronikus levelek továbbítására szolgáló szolgáltatást írja le (lásd a 8.4. szakasz a 254. oldalon).

Ez a szolgáltatás a levelek továbbítására és nem a megérkezett levelek elolvasására szolgál.

domain A szolgáltatás a DNS (*domain name service*, tartománynév-szolgáltatás) protokoll szerint a számítógépek nevének lekérdezésére használható.

finger A szolgáltatás segítségével adatokat kérhetünk a távoli számítógép jogos felhasználóról, többek között arról, hogy az adott pillanatban be vannak-e jelentkezve.

Sok rendszergazda nem tartja szerencsésnek ezt a szolgáltatást, és nem is engedélyezi.

http A HTTP (*hypertext transfer protocol*, hiperszöveg-átviteli protokoll) a weblapok átvitelére szolgáló szabvány, a http szolgáltatásra tehát a webkiszolgálnak van szüksége.

pop3 A POP3 (*post office protocol*, postaprotokoll) az elektronikus levelezésben használható, a felhasználó leveleinek letöltésére szolgáló szabvány.

Ez a protokoll nem használ titkosítást a felhasználók neveinek és a jelszavainak az átvitelére, ezért ma már nem használjuk titkosító kiegészítés nélkül.

sunrpc A szolgáltatás segítségével elérhetővé válik az RPC (*remote procedure call*, távoli eljáráshívás).

imap A szolgáltatás segítségével a POP3 szabványhoz hasonló, annál fejlettebb levelezési rendszer érhető el. Az IMAP (*interactive mail access protocol*, interaktív levélletöltő protokoll) a felhasználók neve és jelszava átvitele közben nem használ titkosítást, ezért ma már nem használjuk titkosító kiegészítés nélkül.

xdmcp Az XDMCP (*X display manager control protocol*, X képernyőkezelő vezérlő protokoll) lehetővé teszi, hogy távoli számítógépekre grafikus felületen keresztül belépjünk.

ircd Az IRC (*internet relay chat*, internethet csevegőrendszer) igen népszerű szolgáltatás, amelynek segítségével többen tarthatják egymással egy időben a kapcsolatot.

xfs Az xfs szolgáltatást az X Window betűtípus-kiszolgálója használja betűtípusok elérhetővé tételere. Az X Window betűtípus-kiszolgálója lehetővé teszi, hogy a betűtípusokat egy helyen tároljuk és sok helyen felhasználjuk.

A szolgáltatások közt találhatunk olyanokat, amelyek neve a felsorolt nevek valamelyikével megegyezik, de egy s betűs kiegészítést tartalmaz a név végén (például [https](https://), [pop3s](pop3s://) stb.). Ezek a szolgáltatások az eredeti szolgáltatások titkosítással kiegészített változatai.



Az átvitt adatok védelmének érdekében mindenkorban használjuk a szolgáltatások titkosított változatát, különben nagy valószínűség szerint igen rövid idő alatt komoly problémáink adódnak a behatolókkal! Titkosítatlan formában jelszavakat küldeni az Interneten ma már az őrültséggel határos felelőtlenség.

Szolgáltatás nyilvántartás nélkül (távoli eljáráshívás)

Az RPC (*remote procedure call*, távoli eljáráshívás) a hálózati szolgáltatások igénybe vételének egy módja, amelyet eredetileg a Sun Microsystems dolgozott ki. Mára az RPC használatának módját szabvány rögzítette [158].

Az RPC tulajdonképpen az egyes szolgáltatásokhoz tartozó hálózati kapuk lekérdezésének módja, amely lehetővé teszi, hogy az alkalmazások akkor is egymásra találjanak, ha a használt szolgáltatás nem szerepel a szolgáltatások nyilvántartásában. Az RPC használata lehetővé teszi, hogy új szolgáltatásneveket – és hozzájuk tartozó kapukat – vezessünk be és tegyük elérhetővé mások számára is, de ne kelljen a protokollokat megváltoztatni, illetve a szolgáltatások nyilvántartását minden egyes számítógépen frissíteni.

Maga az RPC szolgáltatás roppant elmés, a következő lépések alapján működik:

1. A kiszolgáló számítógépen először elindul a portmapper (kapuleképező) szolgáltatást megvalósító program.

A portmapper szolgáltatás az RPC rendszer lényege, amely szabványos szolgáltatásként megtalálható, hiszen a szolgáltatások jegyzékében a 111-es számot rendeljük hozzá.

2. Következő lépésként azok a kiszolgálóprogramok indulnak el, amelyek az RPC rendszeren keresztül kereshetők fel.

Minden egyes kiszolgáló felveszi a kapcsolatot a portmapper kiszolgálóval és nyilvántartásba veteti, hogy milyen néven és melyik kapun nyújt szolgáltatást. Nem fontos, hogy minden induláskor ugyanazon a kapun legyen elérhető a szolgáltatás, csak az a fontos, hogy a portmapper tudja, hogy az adott néven szereplő szolgáltatáshoz melyik kapu tartozik.

3. Következő lépésként az ügyfélprogram indul el az ellenállomáson.

Az ügyfélprogram az igénybe veendő szolgáltatás nevét elküldi a kiszolgáló 111-es kapujára, mert tudja, hogy ott található a portmapper, minden RPC szolgáltatások tudója.

4. A portmapper kikeresi a nyilvántartásba vett szolgáltatások közül az adott néven szereplő szolgáltatást, és a kapu számát visszaküldi az ügyfélnek.
5. A válasz megérkezésekor az ügyfél tudni fogja, hogy az általa keresett szolgáltatás melyik kapun érhető el. Ettől a ponttól a kapcsolat kiépítése a megsokott módon folytatódhat.

Az RPC rendszer használatával kapcsolatban mindenképpen érdemes megjegyeznünk, hogy a portmapper pótolhatatlan szerepet játszik a bemutatott folyamatban, ha tehát nem érhető el a portmapper, egyetlen RPC alapokon megvalósított szolgáltatás sem fog működni.

Úgyszintén nem működik az az RPC alapú szolgáltatás, amelyet a portmapper kiszolgáló előtt indítottunk el. Ha a portmapper szolgáltatást újraindítjuk, az összes RPC alapokon megvalósított kiszolgálóprogramot újra kell indítanunk, hogy azok nyilvántartásba vetessék magukat.

5.2.2. A szolgáltatások felderítése

Az nmap program segítségével a helyi vagy távoli számítógép által nyújtott szolgáltatásokat deríthetjük fel. A program sorra veszi a célszámítógép hálózati kapuit, és minden kapun megkísérli felvenni a kapcsolatot a számítógéppel, majd kiírja a szabványos kimenetére, hogy az egyes kapukon milyen sikерrel járt a kapcsolatfelvételi kísérlet.



A számítógépek ilyen módon való felderítése jó kiindulási pont lehet egy esetleges támadáshoz, ezért a tapasztalt rendszergazda már a felderítést is támadásként kezeli.

Soha ne használjuk az nmap programot – és semmi hasonlót sem – idegen számítógépen! Idegen számítógép biztonsági vizsgálata támadásnak, provokációjának minősül, amit könnyű érzékelni!

Az nmap a vizsgált hálózati kapukat a következő módon csoportosítja:

open A vizsgált kapu nyitott, kapcsolatfelvételre kész. Az ilyen kapuk mögött általában aktív szolgáltatások találhatók.

filtered A vizsgált kaput valamilyen eszköz védi, így nem lehet kideríteni, hogy van-e aktív szolgáltatás a kapu mögött. Sejthető, hogy a program ezekben az esetekben aktív szolgáltatást védő tűzfalrendszert talált.

closed A vizsgált kapuról egyértelműen kideríthető volt, hogy az nincs nyitva, mögötte aktív szolgáltatás nincs. Az nmap dokumentációja ezt az állapotot **unfiltered** állapotnak is nevezi.

Természetesen nem lehetünk teljesen biztosak abban, hogy ezeken a kapukon nem érhető el szolgáltatás, hiszen lehetséges, hogy egy védelmi rendszer esetleg elfedte előlünk a szolgáltatást, és már jelezte is a rendszerazonosításnak a behatolási kísérletünket.

Az nmap képes arra is, hogy a célszámítógép adta válaszokból felderítse, hogy az milyen operációs rendszert, annak melyik változatát futtatja. Ezt a program úgy éri el, hogy az egyes operációs rendszerek hálózati alrendszerei közti finom különbségeket az adatbázisában található adatokkal összeveti, és megvizsgálja a válaszok „ujjlenyomatát”. A program a -O kapcsoló hatására ilyen vizsgálatokat is végez.

46. példa. Vizsgáljuk meg, milyen kapuk vannak nyitva a számítógépünkön!

\$ nmap localhost

```
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-07-10 23:43 CEST
Interesting ports on localhost.localdomain (127.0.0.1):
The 1654 ports scanned but not shown below are in state: closed
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
111/tcp   open  rpcbind
631/tcp   open  ipp
6000/tcp  open  X11

Nmap run completed -- 1 IP address (1 host up) scanned in 1.324 seconds
$
```

Amint látjuk, a számítógépen öt szolgáltatás érhető el. Érdemes megfontolnunk, hogy ezekre valóban szükségünk van-e, és ha nem, a szükségtelen szolgáltatásokat le kell állítani.

A kapuk felderítéséről szóló rész végén még egyszer megemlíjtük, hogy a mások által üzemeltetett számítógépek ilyen módon történő feltérképezése sokak szerint támadásnak minősül, ezért minden csatlakozó csak a saját gépeinken szabad használnunk ezt a módszert.

5.3. Az Internet szuperkiszolgáló

Az Internet szuperkiszolgáló olyan program, amely más kiszolgálóprogramok számára könnyíti meg a hálózati szolgáltatások nyújtását. A szuperkiszolgáló kifejezés arra utal, hogy sok, egymástól független hálózatos szolgáltatás érhető el ilyen módon, mert maga a szuperkiszolgáló is sok szolgáltatást valósít meg, és sok más kiszolgáló is rajta keresztül nyújtja a szolgáltatásokat.

Az Internet szuperkiszolgáló a beállítóállománya – vagy beállítóállományai – alapján fogadja a hálózatról beérkezett kapcsolatfelvételi kéréseket, és szükség esetén elindítja a szolgáltatásokat nyújtó programokat.

A szuperkiszolgáló a szolgáltatást nyújtó alkalmazás elindítása után is komoly szerepet vállal a munkában: a hálózati kapcsolatot a szolgáltatás igénybe vételenek ideje alatt kezeli, a bejövő adatokat a szolgáltatást megvalósító program szabványos bemenetére küldi, a program szabványos kimenetén megjelenő adatokat pedig továbbítja az ügyfélnek.

Az Internet szuperkiszolgáló természetesen nagyon megkönnyíti a szolgáltatást megvalósító alkalmazás készítőjének munkáját, hiszen átvállalja a hálózat kezelésének terhét. Az Internet szuperkiszolgáló felhasználásával minden program, amely képes a szabványos bemenet és a szabványos kimenet kezelésére, hálózati szolgáltatások nyújtására is alkalmassá válik. Ilyen módon akár egy pár sornyi BASH programot is kiszolgálóként használhatunk (lásd a 48. példát a 131. oldalon).

GNU/Linux rendszereken az `inetd` és a `xinetd` programok terjedtek el Internet szuperkiszolgálóként. Az `inetd` igen kiforrott és közismert program, a `xinetd` viszont újabb és kifinomultabban állítható. Nyilvánvaló, hogy minden programnak népes rajongótábora van a rendszergazdák körében².

5.3.1. Az `inetd` beállítása és használata

Az `inetd` régen kipróbált, stabilan működő Internet szuperkiszolgáló, amely minden működésében, minden beállítóállományának szerkezetében teljesen megegyezik a más UNIX-változatokon használatos megvalósításokkal. Az `inetd` a szolgáltatásoknál megszokott módon indítható és állítható le, ahogyan azt a 14. oldalon található 1. példában láttuk.

²És természetesen sokan vannak, akik egyiket sem használják, hiszen minél kevesebb szolgáltatást nyújtunk, annál kisebb a hiba esélye.

5.1. tábla: inetd

Internet szuperkiszolgáló több szolgáltatás közös kezelésére.

inetd [kapcsolók] [beállítóállomány]

Az inetd indítása után a beállítóállománya alapján meghatározott hálózati kapukat megnyitja olvasásra, a beérkezett kéréseket fogadja. Ha kérés érkezik valamelyik kapura, a program a beállítóállománya alapján meghatározott alkalmazást elindítja, a hálózatról olvasott adatokat a program szabványos bemenetére irányítja, a program szabványos kimenetén megjelenő adatokat pedig a hálózaton válaszként elküldi.

Az inetd program indítását és leállítását általában a szolgáltatásoknál megszokott módon a /etc/init.d/ könyvtárban található héjprogram végzi.

Kapcsoló	Jelentés
-d	Hibakereső üzemmód bekapsolása.

Az inetd induláskor a /etc/inetd.conf állományból olvassa a beállításait, és az ott található szolgáltatások nyújtásával végzi feladatát. A következő néhány sor ennek a beállítóállománynak egy részletét mutatja be:

1	# Szolg.	Típus	Prot.	Mód	Felh.	Program
2	time	stream	tcp	nowait	root	internal
3	#time	dgram	udp	wait	root	internal
4						
5	smtp	stream	tcp	nowait	root	/usr/sbin/sendmail -bs

Amint a példából is látható az állományban a szokásos módon, a # karakter után helyezhetünk el megjegyzéseket. A /etc/inetd.conf állomány a következő oszlopokból áll:

1. Az első oszlop a szolgáltatás nevét tartalmazza, ahogyan az a /etc/services állomány első oszlopában olvasható. Ez az oszlop határozza meg, hogy az adott sorban leírt szolgáltatás melyik logikai kapun keresztül érhető el.
2. A második oszlopból – többek közt – a dgram és a stream kulcsszavakat használhatjuk. A dgram (datagram, adatcsomag) jelzi, ha egyes csomagok küldésén alapuló szolgáltatásról van szó, míg a stream (stream, csatorna) azt, ha folyamatos adatfolyamról.
3. A harmadik oszlop a használt hálózati alapprotokoll nevét adja meg. Itt többek közt az udp és a tcp kulcsszavakat használhatjuk az UDP, illetve a TCP protokoll jelölésére.

Mivel az UDP adatcsomagok küldését teszi lehetővé, nyilvánvaló, hogy a hozzá tartozó sor második oszlopában a dgram kulcsszó szerepel. Ennek megfelelően azon sorokban, ahol a harmadik oszlopba a tcp kulcsszót írjuk, a második oszlopban a stream szónak kell szerepelnie³.

4. A negyedik oszlopban a wait vagy nowait kulcsszónak kell szerepelnie.

Azok a szolgáltatások, amelyek nem adatcsomagok küldésén alapulnak, kötelezően a nowait bejegyzést kell, hogy kapják a negyedik oszlopban, ami jelzi az inetd kiszolgálónak, hogy a szolgáltatást megvalósító program kilépését nem kell megvárnia, több beérkező kérés esetén több példányban is el kell indítania.

Az adatcsomagokon alapuló szolgáltatások esetén a wait jelzi, hogy a szolgáltatást megvalósító programot egy példányban kell elindítani és az az egy példány fogja sorra kiszolgálni az összes kérést. Az adatcsomagokon alapuló szolgáltatások esetén is használhatjuk a nowait kulcsszót, amely jelzi, hogy a szolgáltatást megvalósító programot szükség esetén több példányban is el kell indítani.

5. Az ötödik oszlopban annak a felhasználónak a felhasználói nevét olvashatjuk, akinek a névében a szolgáltatást megvalósító kiszolgálóprogramot futtatni kell.

Általában szerencsés, ha a szolgáltatásokat nyújtó program nem a rendszer-gazda névében fut, mivel így kisebb a veszélye annak, hogy a szolgáltatás támadásával a támadó teljhatalomhoz jut.

6. A hatodik oszlopban a szolgáltatást megvalósító program neve található.

Ha itt az internal kulcsszót találjuk, akkor az adott szolgáltatást maga az inetd valósítja meg. Arról, hogy az adott inetd változat milyen szolgáltatásokat képes nyújtani, a program dokumentációjában olvashatunk.

Ha a hatodik oszlopban nem az internal kulcsszó található, akkor az inetd az itt található programot kíséri meg elindítani, és a hálózati forgalmat ennek a programnak a szabványos csatornái felé irányítja.

Amint látható, az inetd program és a beállítóállománya is igen egyszerű, könnyen érthető. Fontos azonban, hogy felhívjuk a figyelmet a biztonsági megfontolásokra, amelyeket a xinetd programról szóló rész után, a 132. oldalon található 5.3.3. szakaszban részletesebben is megvizsgálunk.

³Felmerülhet a kérdés, hogy ha a második és harmadik oszlop kölcsönösen meghatározzák egymást, akkor egyáltalán miért van szükség két oszlopra, miért kell ugyanazt az információt még egyszer megismételni. A válasz egyszerű: a TCP és az UDP protokollokon kívül még sok más hálózati alapprotokoll is létezik, de ezekre az egyszerűség miatt nem térünk ki.

5.2. tábla: xinetd

Internet szuperkiszolgálót megvalósító program.

xinetd [kapcsolók]

A xinetd Internet szuperkiszolgálót valósít meg, a segítségével a szabványos csatornákon keresztül valósíthatunk meg hálózati szolgáltatásokat. A xinetd programot általában a szolgáltatásoknál megszokott módon a /etc/init.d/könyvtárban található héjprogrammal indítjuk.

Kapcsoló Jelentés

- d Hibakereső üzemmód hibakereső üzenetek kiírásával.
- f fájl A beállítóállomány megadása.
- dontfork Alapértelmezés szerint a program indítás után démonként, a háttérben fut tovább. A kapcsoló ezt a viselkedést tiltja.

5.3.2. A xinetd beállítása és használata

A xinetd modern, rugalmasan beállítható Internet szuperkiszolgáló, amelyet többek között a Red Hat alapú GNU/Linux gyűjtemények újabb változatai használnak. A program a szolgáltatások esetében megszokott módon indítható és állítható le (lásd 1. példa a 14. oldalon).

A xinetd beállítóállománya a /etc/xinetd.conf szöveges állomány. Az állományban a szokásos módon (a # karakter segítségével) helyezhetünk el megjegyzéseket, és használhatjuk a következő kifejezéseket is:

defaults { ... } A kifejezés segítségével a szolgáltatások számára érvényes alapbeállításokat adhatjuk meg, amelyek azokra a szolgáltatásokra fejtik ki hatásukat, amelyeknél az adott beállítást nem adjuk meg külön.

A kifejezésben a {} jelek között adhatjuk meg a beállításokat.

service név { ... } A kifejezés segítségével egy új szolgáltatást vehetünk nyilvántartásba és állíthatunk be.

A kifejezésben szereplő név a szolgáltatás neve, a {} jelek között pedig a szolgáltatás beállításai szerepelnek.

A xinetd csak olyan szolgáltatásokat nyújt, amelyeket a service kifejezéssel létrehozunk és beállítunk.

includedir könyvtárnév A kifejezással arra adhatunk utasítást a xinetd programnak, hogy az adott könyvtár minden állományát beállítóállományként értelmezze. A beállítóállományt ezzel a kifejezással tehát szétszabdalhatjuk részeire.

Ez a kifejezés sokkal hasznosabb, mint ahogyan azt első pillantásra gondolnánk. A segítségével elérhetjük, hogy minden, a `xinetd` segítségével szolgáltatást nyújtó kiszolgálóprogram külön `xinetd` beállítóállománnyal rendelkezzen, ami fontos, hiszen ezeket a kiszolgálóprogramokat általában külön-külön programcsomagokból telepítjük.

A `xinetd` beállítóállományában (vagy beállítóállományaiban) az egyes szolgáltatások beállítására értékadásszerű kifejezéseket használhatunk. A `service` kulcsszó után megadjuk a szolgáltatás nevét, a `{}` jelek pedig a következő értékadásokat használhatjuk:

`type=típus` A szolgáltatás típusa, amely meghatározza, hogy milyen alapokon valósítják meg a szolgáltatás. A `típus` a következő kulcsszavak egyike lehet:

`RPC` A szolgáltatás távoli eljáráshíváson (RPC) alapul.

`INTERNAL` A szolgáltatást nem külső program valósítja meg, hanem maga a `xinetd`.

`UNLISTED` Ez a kulcsszó azt jelzi, hogy a szolgáltatás nem szerepel a szolgáltatások nyilvántartásában (a `/etc/services` állományban). Ha a szolgáltatás nem szerepel a szolgáltatások nyilvántartásában, és nem használjuk az `UNLISTED` kulcsszót, a `xinetd` tiltja a szolgáltatást.

Ha a kulcsszó önmagában áll, a szolgáltatás neve nem található meg a `/etc/services` állományban, és így meg kell adnunk a használandó hálózati kapu számát a `port=szám` kifejezés segítségével.

Ha a kulcsszó az RPC kulcsszó mellett található, a szolgáltatás nem szerepel a `/etc/rpc` állományban, és az RPC számát meg kell adnunk a `rpc_number=szám` kifejezés segítségével.

Ha a szolgáltatás nem RPC alapú, és szerepel a `/etc/services` nyilvántartásban, ezt a kifejezést nem kell használnunk.

`disable=yes|no` Az értékadással a szolgáltatást tilthatjuk anélkül, hogy a beállítóállományból törölünk.

Ha a kifejezésben a `yes` kulcsszó szerepel, a szolgáltatás nem érhető el.

`socket_type=típus` A kifejezés segítségével megadhatjuk, hogy a szolgáltatásnak milyen jellegű hálózati kapcsolatra van szüksége. A `típus` helyén a következő kulcsszavak egyike állhat:

`stream` A szolgáltatás folyamatos adatkapcsolaton alapul, amelyet a megnyitás és lezárás között a kapcsolatban álló felek csővezetékként használhatnak.

Az Interneten az ilyen jellegű kapcsolatok biztosítására általában a TCP protokollt használjuk.

`dgram` A szolgáltatás egyedi csomagküldésen alapul, amelynek lényege, hogy a kapcsolat egyetlen csomag küldésére épül ki.

Az Interneten az ilyen jellegű kapcsolatok biztosítására általában az UDP protokollt használjuk.

`raw` A szolgáltatás megvalósításához a hálózati csatolóhoz való közvetlen hozzáférés szükséges.

Ezt a módszert általában akkor használjuk, ha a szabványos csomagok kezelésére készült eszközök nem felelnek meg az igényeinknek.

`protocol=protokoll` A kifejezés segítségével megadhatjuk, hogy a szolgáltatás milyen hálózati szabvánnyra épül.

A kifejezésben szereplő `protokoll` a protokollok nyilvántartására szolgáló /etc/protocols állományban meghatározott név bármelyike lehet, de a legtöbb esetben az `udp` vagy a `tcp` protokollnevet használjuk.

Ha ez a kifejezés hiányzik a szolgáltatás meghatározásából, a program a szolgáltatáshoz a nyilvántartásba bejegyzett alapértelmezett protokollt használja.

Megfigyelhető, hogy amelyik szolgáltatás esetében a kapcsolattípus `dgram`, ott a protokoll helyén általában az `udp`, ahol pedig a kapcsolattípus `stream`, ott `tcp` kulcsszó található.

`wait=yes|no` A kifejezéssel megadhatjuk, hogy az újabb kapcsolatfelvételi kérések fogadása felfüggesztődjön-e a szolgáltatás igénybe vétele alatt.

Ha a kifejezésben a `yes` kulcsszó áll, a `xinetd` nem fogad a szolgáltatás igénybe vételére irányuló kéréseket, amíg a szolgáltatást kiszolgáló program be nem fejezte futását. Ez azt jelenti, hogy a szolgáltatást egy időben csak egy ügyfél veheti igénybe. A `dgram` kapcsolattípusra épülő szolgáltatások általában ilyenek, hiszen ezek rövid idő alatt befejeződnek, és így semmiképp nincs szüksége hosszú várakozásra.

Ha a kifejezésben a `no` kulcsszó szerepel, a `xinetd` a kapcsolatkéréseket a szolgáltatás igénybe vételének ideje alatt zavartalanul fogadja, a különböző ügyfelek kiszolgálása így időben átlapolódhat. A `stream` kapcsolattípusra épülő szolgáltatások általában ilyenek, hiszen ezeknek a folyamatos kapcsolatra épülő szolgáltatásoknak a kiszolgálása általában hosszú ideig tart.

`user=felhasználó` A kifejezés segítségével megadhatjuk, hogy a szolgáltatást megvalósító kiszolgáló melyik felhasználó nevében fusson. A felhasználót felhasználói nevével és azonosítószámával is megadhatjuk.

Ahhoz, hogy a `xinetd` be tudja állítani a kiszolgálót futtató felhasználó személyét, rendszergazdai jogkörrel kell rendelkeznie, azaz a `xinetd` programot a rendszergazdának kell futtatnia.

Biztonsági okokból szerencsésebb, ha nem a rendszerelő futtatja a hálózati szolgáltatásokat megvalósító programokat, így ezt a kifejezést – ha csak tehetjük – érdemes használni. (Bár tagadhatatlan, hogy a legbiztonságosabb, ha semmiféle szolgáltatást nem nyújtunk, néha meg kell alkudnunk ebben a kérdésben.)

`group=csoport` A kifejezés segítségével beállíthatjuk, hogy a szolgáltatást megvalósító kiszolgáló melyik felhasználói csoport nevében fusson.

`instances=szám` A kifejezés segítségével megadhatjuk, hogy egy időben hány ügyfél veheti igénybe a szolgáltatást, azaz hány példányban futhat a szolgáltatást megvalósító kiszolgáló.

Az alapértelmezett érték UNLIMITED, ami azt jelenti, hogy tetszőlegesen sok ügyfél használhatja egy időben a szolgáltatást. Ezt nyilvánvalóan érdemes korlátozni.

`server=állománynév` A kifejezés segítségével megadhatjuk, hogy hol található a szolgáltatást megvalósító program.

Nem kell használnunk ezt a kifejezést, ha a szolgáltatást maga a `xinetd` valósítja meg.

`server_args=argumentumok` A kifejezés segítségével megadhatjuk a szolgáltatást megvalósító program számára átadandó argumentumokat.

`only_from=listá` A kifejezés segítségével szűkíthetjük a szolgáltatást igénybe vevő számítógépek listáját. Ha olyan számítógép kísérli meg igénybe venni a szolgáltatást, amely nem szerepel a listában, a `xinetd` a kapcsolatfelvételi kérést elutasítja.

A kifejezésben szereplő lista a következő formájú elemekből állhat (példák):

10.1.1.1 A listaelem tartalmazhat IP címet a szokásos – pontokkal elválasztott – formában.

10.0.0.0 Ha a lista elemeként szereplő IP cím egy vagy több 0 tagra végződik, a program a címet egy hálózat címének tekinti, és az adott hálózat minden címére illeszkedőként kezeli.

10.1.1.1/8 Ha a lista elemeként szereplő IP cím után a / jellel megadjuk a címben hálózati részként szereplő bitek számát, a program a listaelement a hálózat minden címére illeszkedőként kezeli.

`név` Ha számítógép nevét adjuk meg listaelemként, a `xinetd` fordított DNS kereséssel megállapítja, hogy mi az ügyfél neve, és összehasonlíta az általunk megadott névvel.

`.név` Ha a megadott számítógép neve a . karakterrel kezdődik, a program mindenazon munkaállomások számára elérhetővé teszi a szolgáltatást, amelyek ebbe a tartományba tartoznak. A szolgáltatást kérő számítógép nevét ebben az esetben is fordított DNS kéréssel deríti fel a program.

no_access=lista A kifejezés segítségével megadhatjuk azoknak a munkaállomásoknak a listáját, amelyek nem vehetik igénybe a szolgáltatást. A kifejezésben szereplő lista az **only_from=lista** alakú kifejezés listájával megegyező formájú, de természetesen ellentétes hatású.

access_times=óra:perc-óra:perc A kifejezés segítségével megadhatjuk, hogy a szolgáltatást mely időszakban lehet használni. Az órák és a percek számának mozása 0-tól indul.

rpc_number=szám A kifejezés segítségével megadhatjuk a távoli eljáráshívás számát az olyan szolgáltatásoknál, amelyek nincsenek felsorolva a /etc/rpc állományban.

Ezen szolgáltatások leírásában természetesen szerepel a következő sor:

1	type = RPC UNLISTED
---	---------------------

port=szám A kifejezés segítségével megadhatjuk, hogy a szolgáltatás melyik hálózati kapun vehető igénybe.

Csak akkor van értelme megadni a kapu számát, ha a szolgáltatás nem szerepel a /etc/services állományban.

redirect=ip_cím kapu A szolgáltatás igénybe vételére irányuló csomagok átirányítása másik számítógépre. Ezzel a kifejezéssel elérhetjük, hogy a szolgáltatást az ügyfélprogram egy másik számítógépen vegye igénybe.

bind=ip_cím E kifejezéssel elérhetjük, hogy az Internet szuperkiszolgáló az adott szolgáltatást a több hálózati csatolóval, több IP címmel rendelkező számítógép csak egy hálózati csatolóján nyújtsa. A szolgáltatás csak azon a csatolón lesz elérhető, amelynek IP címét a kifejezésben megadtuk.

per_source=szám A kijezéssel előírhatjuk, hogy egy ügyfél legfeljebb hány példányban vehesse igénybe az adott szolgáltatást egy időben. A nyilvános szolgáltatások igénybe vételét szerencsés lehet ilyen módon korlátozni, hogy a számítógépet ne terhelhessék túl rosszakaróink.

cps=szám szám A megengedett legnagyobb másodpercenkénti kapcsolatfelvétel száma (*connection per second*).

A kifejezés első száma megadja, hogy az Internet szuperkiszolgáló egy másodperc alatt hány kapcsolatkérést fogad ezzel a szolgáltatással kapcsolatban. Ha a beérkezett kérések száma meghaladja ezt a számot, a kiszolgáló a szolgáltatást időlegesen szünetelteti.

A kifejezés második száma megadja, hogy hány másodpercre szüneteltesse a szolgáltatást a kiszolgáló, ha a kapcsolatfelvételi kérések száma túl magas.

A `xinetd` alapértelmezés szerint legfeljebb 50 kapcsolatfelvételi kérést fogad egy szolgáltatáshoz, és ha a kapcsolatfelvételi kérések száma ezt meghaladja 10 másodpercen keresztül szünetelte a szolgáltatást.

`rlimit_as=szám` A kifejezéssel megadhatjuk, hogy a `xinetd` az általa indított ki-szolgálóprogram memóriafelhasználását milyen értékre korlátozza. A szám után használhatjuk a K, illetve az M betűket a kilobájt és a megabájt rövidítéseként.

A `xinetd` beállításainak megváltoztatására a leggyakrabban szolgáltatások engedélyezése és tiltása miatt van szükség. Ilyen esetben egyszerűen átírjuk az adott szolgáltatás jellemzői közt szereplő `disable` kulcsszó értékét, és a `SIGHUP` jelzéssel utasítjuk a `xinetd` programot futtató folyamatot, hogy olvassa újra a beállítóállományokat. Ezt mutatja be a következő példa.

47. példa. Zárt belső hálózatunkon engedélyezni szeretnénk a `pop3` szolgáltatást, annak ellenére, hogy tisztaiban vagyunk annak veszélyeivel. Változtassuk meg a `xinetd` beállításait!

Első lépésként ellenőrizzük, hogy a szolgáltatás jelenleg elérhető-e:

```
$ telnet 10.1.1.1 pop3
Trying 10.1.1.1...
telnet: connect to address 10.1.1.1: Connection refused
$
```

Amint látjuk, a `pop3` szolgáltatáshoz rendelt hálózati kapu zárva van, a szolgáltatás nem elérhető.

Most keressük meg a `pop3` szolgáltatáshoz tartozó sorokat a beállítóállományokban. Vizsgáljuk meg a `xinetd` beállítására szolgáló `/etc/xinetd.conf` állományt!

```
$ cat /etc/xinetd.conf

includedir /etc/xinetd.d
$
```

Amint látjuk, a `xinetd` beállítása szolgáltatásoknál külön beállítóállományokban történik a `/etc/xinetd.d/` könyvtárban. Írassuk ki a könyvtárban található könyvtárbejegyzések nevét, amelyekben a `pop` betűszó megtalálható!

```
$ ls /etc/xinetd.d/*pop*
/etc/xinetd.d/ipop2  /etc/xinetd.d/ipop3  /etc/xinetd.d/pop3s
```

Nyilvánvaló, hogy ezek közül az állományok közül az `ipop3` szolgál a `pop3` szolgáltatás beállítására. Keressük meg az állományban a `disable` kulcsszóval kezdődő részt, és írjuk át, hogy utána a `no` kulcsszó álljon:

1	<code>disable = no</code>
---	---------------------------

Küldjük most a futó xinetd példánynak a SIGHUP üzenetet, hogy az újraolvassa a beállítóállományokat, és a megfelelő módon megváltoztassa a viselkedését!

```
$ ps aux | grep xinetd
root  3334  0.2  0.3  2108  908 ?    S   18:42  0:00 xinetd
$ kill -SIGHUP 3334
$
```

Ha ezek után próbáljuk felvenni a kapcsolatot a számítógépen található pop3 szolgáltatóprogrammal, az azonnal bejelentkezik:

```
$ telnet 10.1.1.1 pop3
Trying 10.1.1.1...
Connected to gep.otthon.
Escape character is '^]'.
+OK POP3 notebook v2001.78rh server ready
```

A xinetd segítségével könnyedén futtathatunk egyszerű héjprogramokat hálózati szolgáltatások megvalósítására. Ennek módját ismerhetjük meg a következő példából.

48. példa. Készítsünk beállításokat, amelyekkel kipróbálhatjuk a xinetd beállításait!

Hozzunk létre egy szolgáltatást a következő beállításokkal:

1 #	/sbin/minta
2 # Próbaállomány hálózati szolgáltatáshoz	
3 #	
4 service probi	
5 {	
6 type = UNLISTED	
7 socket_type = stream	
8 protocol = tcp	
9 port = 777	
10 wait = no	
11 server = /sbin/probi.sh	
12 user = root	
13 }	

Figyeljük meg, hogy olyan szolgáltatásnevet használtunk, amely nyilvánvalóan nem létezik a szolgáltatások nyilvántartásában. A szolgáltatás nem RPC alapú és nem szerepel a nyilvántartásban, így mindenkorban meg kell adnunk, hogy milyen szabványt használunk (TCP) és melyik hálózati kapun nyújtjuk a szolgáltatást (777). A szolgáltatást a /sbin/probi.sh program nyújtja, amely a következő sorokból áll:

```

1  #
2  # Kiírunk egy üzenetet és egy parancskérő jelet
3  #
4  echo "Ez a probi szolgáltatás:"
5  echo -n "# "
6  #
7  # Beolvassuk a felhasználó parancsát
8  #
9  read PARANCS
10 #
11 # Eldobjuk belőle a kocsivissza karaktert, és vissza-
12 # írjuk kérdőjelekkel
13 #
14 echo "$PARANCS???" | tr -d '\r'

```

A program a 4–5. sorokban kiír egy üzenetet és a parancskérő jelet, majd a 9. sorban beolvas egy sort. A beolvasott sorból a 14. sorban hívott `tr` program eltávolítja a kocsivissza karaktert, és visszaírja a hálózatra. (A kocsivissza karakter eltávolítására azért van szükség, mert az Interneten a sorok végét a kocsivissza és az újsor karakterek együttes használatával jelezzük, a UNIX alkalmazásokban azonban csak az újsor karaktert használjuk.)

A `xinetd` újraindítása után az új szolgáltatást kipróbálhatjuk, ha a `telnet` program segítségével kapcsolódunk a 777-es számú hálózati kapura.

```
$ telnet localhost 777
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Ez a probi szolgáltatás:
# help
help???
Connection closed by foreign host.
$
```

Természetes, hogy a szolgáltatást más program igénybe vételevel is használhatjuk, a `telnet` azonban – amint a példából látható – jó eszköz a szolgáltatások kipróbálására.

5.3.3. Az Internet szuperkiszolgáló védelme

A hálózaton nyújtott szolgáltatások biztonsági szempontból igen fontosak, ezért néhány szót kell ejtenünk arról, hogy az Internet szuperkiszolgáló használata esetén hogyan védhetjük meg a számítógépet a hálózat felől érkező támadásoktól.

Mindenekelőtt tudnunk kell, hogy minden szolgáltatás egy-egy támadási felületet jelent, ezért alapszabály, hogy csak azokat a szolgáltatásokat szabad engedélyezni, amelyekre valóban szükségünk van. Ez talán a számítógépes biztonság legfontosabb szabálya.



Ha hálózatra kötött számítógépre telepítettünk GNU/Linuxot, az első dolgunk az legyen, hogy megvizsgáljuk, milyen szolgáltatások érhetők el.

Vizsgáljuk meg az Internet szuperkiszolgáló beállítóállományát (vagy beállítóállományait), és tiltsuk le az összes szolgáltatást, amelyre előre láthatólag nem lesz szükségünk. Ha szerencsénk van, egyetlen olyan szolgáltatást sem fogunk találni, amelyre okvetlenül szükségünk van, és leállíthatjuk az Internet szuperkiszolgálót.

Fontos tudnunk azt is, hogy bizonyos szolgáltatások a felhasználók jelszavait titkosítás nélkül küldik át a hálózaton. Ezeknek a szolgáltatásoknak a használatát ma már nem engedhetjük meg magunknak, legfeljebb zárt hálózatokon. A jelszótitkosítást nem használó – régebbi szabványok által szabályozott – szolgáltatások legtöbbjét az Internet szuperkiszolgáló kezeli, így ezeket az Internet szuperkiszolgáló beállítóállományában kell letiltanunk. (Bár a modern GNU/Linux terjesztések esetében ezek a szolgáltatások általában alapértelmezés szerint tiltottak.)

Amint láttuk, a `xinetd` meglehetősen fejlett korlátozó eszközökkel rendelkezik. Használjuk ezeket, és korlátozzuk a szolgáltatások igénybe vételét. A korlátozó eszközök segítségével sok támadásnak elejét vehetjük.

Az `inetd` nem rendelkezik olyan kifinomult eszköztárral a szolgáltatások korlátozására, mint a `xinetd`, az `inetd`-t és a `xinetd`-t is felkészítették azonban az ilyen célokra kifejlesztett `tcpd` program használatára. A `tcpd` beállítóállományai segítségével tehát az `inetd` és a `xinetd` által nyújtott szolgáltatások hatóköre is szűkíthető.

A `xinetd` alapértelmezés szerint a `libwrap` programkönyvtár segítségével automatikusan kezeli a `tcpd` program beállítóállományaiban megfogalmazott korlátozásokat, az `inetd`-nek azonban külön utasítást kell adnunk, hogy a szolgáltatást megvalósító kiszolgálóprogramot a `tcpd` segítségével indítsa. Ezt mutatja be a következő példa.

49. példa. Használni szeretnénk a `finger` szolgáltatást az `inetd` segítségével. A beállítóállományban a következő sort találjuk:

```
1 finger stream tcp nowait nobody in.fingerd
```

Érjük el, hogy a `finger` szolgáltatás használatát a `tcpd` program segítségével korlátozhassuk! Ehhez módosítsuk a beállítóállományt úgy, hogy a szolgáltatást megvalósító kiszolgálót a `tcpd` indítsa el:

```
1 finger stream tcp nowait nobody /usr/sbin/tcpd in.fingerd
```

A beállítóállomány módosítása után az inetd Internet szuperkiszolgálót újra kell indítani, vagy a SIGHUP üzenetet kell küldeni a folyamatnak.

A szolgáltatások igénybe vételét korlátozó `tcpd` program az alábbi lépések alapján végzi munkáját:

1. A program megvizsgálja, hogy a szolgáltatás igénybe vételére érkezett kéres illeszkedik-e a `/etc/hosts.allow` állomány valamelyik szabályára. Ha igen, a megfelelő szabály alkalmazásával a szolgáltatás igénybe vételét engedélyezi.
2. Ha az 1. pontban leírt keresés nem jár eredménnyel, a program megvizsgálja a `/etc/hosts.deny` állományban található szabályokat. Ha ebben az állományban a szolgáltatás kérésére illeszkedő szabályt talál, a megfelelő szabály alkalmazásával a kérést elutasítja, a szolgáltatást megvalósító kiszolgálóprogramot nem indítja el.
3. Ha az 1. és a 2. pontban megfogalmazott keresések nem jártak sikerrel, a `tcpd` a szolgáltatást engedélyezi.

A `/etc/hosts.allow` és a `/etc/hosts.deny` állományokban a szokásos módon – a # karakter után – helyezhetünk el megjegyzéseket. Az állományokba a szabályok a következő kifejezések formájában írhatók be:

kiszolgálók : **ügyfelek** Az ilyen kifejezésekkel leírt szabályokat a `tcpd` sorra megkíséri illeszteni a szolgáltatásra és a szolgáltatást igénybe venni kívánó ügyfélre. A szabályok közül az első illeszkedő fejt ki a hatását.

A kifejezésben található részek jelentése a következő:

kiszolgálók A kiszolgálóprogramok nevének – szóközökkel elválasztott – lista, amelyekre az adott szabály vonatkozik.

A kiszolgálóprogramok nevében különféle helyettesítő karaktereket használhatunk az egyszerűsítés érdekében. A következőkben ezekre a helyettesítő jelekre még visszatérünk.

ügyfelek Az ügyfelek – szóközökkel elválasztott – lista, amelyekre az adott szabály vonatkozik.

Az ügyfelek leírásakor használható egyszerűsítő kifejezésekre a későbbiekben még visszatérünk.

kiszolgálók : **ügyfelek** : **parancs** Az előző kifejezéssel megegyező hatású ez a kifejezés is, annyi különbséggel, hogy ha a `tcp` a szabályt alkalmazza, a szabályban megadott parancsot a héj segítségével végrehajtatja.

A szabályban található **kiszolgálók** és **ügyfelek** lista a már bemutatott szabályhoz hasonlóan értelmezhető. A szabály **parancs** részében a következő kifejezéseket használhatjuk:

- %a Az ügyfél IP címe.
- %c A szolgáltatási igénnel jelentkező ügyfél összes adata *felhasználó@számítógépnév* alakban, ha a szolgáltatási igénnel jelentkező felhasználó neve elérhető volt.
A *tcpd* képes a megfelelő szabvány[66] alapján lekérdezni az ügyfél-géptől a szolgáltatási igénnel jelentkező felhasználó felhasználói nevét. Ha az ügyfélgép a felhasználó nevét kiadja, a *tcpd* figyelembe veszi a működése során.
- %d A szolgáltatást nyújtó kiszolgálóprogram neve.
- %h A szolgáltatást igénybe vevő ügyfélgép teljes neve, ha a név elérhető volt, illetve az IP címe, ha nem.
- %u A szolgáltatást igénybe vevő felhasználó felhasználói neve.

A */etc/hosts.allow* és a */etc/hosts.deny* állományokban a következő eszközöket használhatjuk a szabályok egyszerűsítésére, összevonására:

- * A számítógépek neveinek és IP címeinek megadásakor használhatjuk a * karaktert, amely az állománynevek esetében megsokott módon használható. A * karakter nem használható együtt a . helyettesítő karakterrel, illetve *cím/maszk* formájú hálózatmegadás esetén.
 - ? A számítógépek neveinek és IP címeinek megadásakor használhatjuk a ? karaktert. A ? helyettesítő karakter az állománynevek esetében megsokott módon használható.
A * karakter szintén nem használható együtt a . helyettesítő karakterrel *cím/maszk* formájú hálózatmegadás esetén.
 - . A számítógépek neveinek és IP címeinek megadásakor különleges jelentése van a . karakternek, ha a szöveg elején, illetve végén – első, illetve utolsó helyen – található.
Ha a számítógép neve egy . karakterrel kezdődik, a név az adott névtartomány minden tagjára illeszkedni fog. A .hu név például minden számítógépre illeszkedni fog, amelynek első szintű tartományneve a hu.
Ha a számítógép IP címét a . karakterrel zárjuk, a kifejezés minden IP címre illeszkedni fog, amely az adott címtartományban van. A 192.168. kifejezés például minden számítógépre illeszkedik, amelynek IP címe a 192.168.0.0/255.255.0.0 tartományba esik.
- / Az ilyen karakterrel kezdődő karakterláncokat a *tcpd* állománynévként értelmezi, az állományban található listát beolvassa, és soronként vizsgálja a mintaegyezt.
- A / karakterrel kezdődő karakterlánc minden számítógépre illeszkedni fog, amelynek adatai illeszkednek az adott állomány bármely mintájára.

Az egyszerű karaktereken túl használhatjuk a következő kulcsszavakat is:

ALL Ez a kifejezés minden számítógépre és minden kiszolgáló nevére illeszkedik.

UNKNOWN Illeszkedik minden ismeretlen felhasználói névre (a felhasználói nevet nem lehet lekérdezni), minden ismeretlen nevű számítógépre (a név nem deríthető ki helyi állományból és a DNS adatbázisból) és minden ismeretlen című számítógépre (az ügyfélprogram ismeretlen hálózati alapprotokollt használ).

KNOWN Illeszkedik minden felhasználóra, akinek neve ismert és minden számítógépre, amelynek neve is és IP címe ismert.

PARANOID Illeszkedik minden számítógépre, amelynek a neve nem egyértelmű a DNS adatbázis alapján. A név akkor nem egyértelmű, ha a névhez rendelt IP címet visszakeresve ahoz nem az adott név van rendelve, azaz ha a DNS bejegyzés és a fordított DNS bejegyzés nem egyértelmű.

EXCEPT E kulcsszó segítségével kivételeket adhatunk meg (*except*, kivéve) lista EXCEPT lista formában.

5.4. A névkiszolgáló üzembe helyezése és beállítása

A BIND (*Berkeley Internet name daemon*, Berkeley internetes névdémon) programcsomag DNS kiszolgálót valósít meg, azaz a BIND segítségével DNS ügyfelek számára névinformációkat szolgáltathatunk[1]. A következő oldalakon arról olvashatunk, miképpen helyezhetünk üzembe névkiszolgálót és hogyan kezelhetjük a segítségével nyilvántartott tartományokat[79, 6].

A BIND kiszolgálót megvalósító állomány a named program, amelyet a szokásos módon lehet szolgáltatásként indítani és leállítani (lásd 1. példa a 14. oldalon).

A BIND fő beállítóállománya a /etc/named.conf szöveges állomány. Néhány GNU/Linux terjesztésben az állományt a /etc/bind/named.conf néven találjuk meg.

A beállítóállomány szöveges, benne megjegyzésekkel a C Programozási nyelv szabályai szerint helyezhetünk el. Ez azt jelenti, hogy a // kifejezés után, a sor végeig minden szöveg megjegyzésnek számít, valamint megjegyzésnek számít a /* és a */ kifejezések közti szöveg is.

A /etc/named.conf beállítóállományban szereplő fontosabb szakaszok a következők lehetnek:

options { ... }; A kifejezés segítségével a teljes névkiszolgálóra érvényes beállításokat adhatunk meg.

zone "név" { ... }; A kifejezés segítségével egy zónát írhatunk le. A zóna a DNS kiszolgáló által használt egyik alapfogalom, a tartománynevek egy tartományát jelenti.

5.3. tábla: named

DNS szolgáltatást megvalósító program.

named [kapcsolók]

A named DNS szolgáltatást nyújtó program. A programot általában a szolgáltatásoknál megszokott módon a `/etc/init.d/` könyvtárban található héjprogram segítségével indítjuk.

Kapcsoló Jelentés

- c fájl A beállítóállomány megadása.
- d szám Hibakereső üzemmód az üzenetek részletezőségének megadásával.
- f Alepértelmezés szerint a program indítás után démonként, a háttérben fut. A kapcsoló ezt a viselkedést tiltja.

A kifejezésben szereplő név azt adja meg, hogy az adott zóna mely tartomány adatait írja le. A DNS (névhez tartozó IP cím) és a fordított DNS (IP címhez tartozó név) szolgáltatások bevezetéséhez külön zónákat kell létrehoznunk:

- Ha az adott zóna számítógépek neveinek IP címeiké való alakítására szolgál (DNS szolgáltatás), a tartománynevet kell megadnunk zónánévként.
- Ha az adott zóna IP címek számítógépnevekké való átalakítására szolgál, le kell írnunk az IP cím értékes tagjait fordított sorrendben, a végehez pedig hozzá kell illesztenünk az `in-addr.arpa` kifejezést.

Az options szakaszban használható legfontosabb kifejezések a következők:

`directory "könyvtárnév";` A kifejezés segítségével megadhatjuk annak a könyvtárnak az elérési útját, amelyben a zónaadatokat, a DNS kiszolgáló által szolgáltatott adatokat tároljuk.

A zone szakaszban használható kifejezések közül a legfontosabbak a következők:

`type típus;` A kifejezés megadja, hogy milyen típusú a zóna, a névkiszolgáló minden szerepet tölt be ennek a zónának az életében. A kifejezésben szereplő típus a következő kulcsszavak egyike lehet:

`master` A kulcsszó arra utal, hogy a zóna elsődleges, a zónában található DNS adatok nyilvántartásának ez a kiszolgáló az elsődleges helye.

Minden névtartományhoz pontosan egy elsődleges névkiszolgáló tarthat, azaz minden zóna pontosan egy helyen szerepelhet elsődleges zónaként.

slave A kulcsszó szerint ez a zóna másodlagos, a névkiszolgáló a zónához tartozó adatok másodlagos névkiszolgálójaként tölti be szerepét.

A másodlagos névkiszolgálók szerepe a terhelés megosztása és az elődleges névkiszolgálók ideiglenes helyettesítése hiba esetén.

A névtartományokhoz több másodlagos zóna is tartozhat, azaz több másodlagos névkiszolgáló is szolgáltathatja a zóna adatait, de az ajánlások szerint legalább egy névkiszolgálót használnunk kell a biztonság érdekében.

forward Az ilyen zónák a DNS kérések továbbadására szolgálnak.

hint Az ilyen zónák segítségével a DNS kiszolgáló képes felderíteni a névkiszolgálókat, amelyek a „.” tartományhoz tartoznak.

file "állománynév"; A zónához tartozó adatok tárolására használt szöveges állomány nevének megadása. A **master** zónák esetében a rendszergazdának kell létrehozni és karbantartania ezt az állományt, a **slave** zónák esetében pedig az elődleges DNS kiszolgálóról letöltött adatok tárolására szolgál az állomány.

A kifejezésben megadott állománynév abszolút elérési úttal és relatív elérési úttal is megadható. Ha relatív elérési utat használunk, a névkiszolgáló az állomány kereséséhez az **options** szakaszban megadott **directory** kifejezést használja.

A zónaadatok tárolására szolgáló állományokban, amelyekben a DNS kiszolgáló által szolgáltatott adatok találhatók, különféle típusú bejegyzéseket használhatunk. A bejegyzések formátuma a következő:

név Az első oszlopban a bejegyzés nevét találjuk, azt a nevet, amely alapján a bejegyzés kikereshető.

Ha az első oszlopban található szöveges érték nem a . karakterrel végződik, a kiszolgáló a névhez automatikusan hozzáilleszti a zóna nevét.

TTL A TTL (*time to live*, hátralévő élettartam) mező segítségével megadhatjuk, hogy az adott bejegyzést más DNS kiszolgálók mennyi ideig tárolhatják ideiglenes állományaikban. Ha a TTL értékben tárolt idő letelik, a DNS kiszolgálók mindenkorban újra lekérdezik a bejegyzés értékét.

A TTL értéket mértékegység nélkül megadhatjuk másodpercben, a h mértékegységgel órában és a d mértékegységgel napban.

A TTL értékét nem kötelező minden bejegyzésnél megadni, azaz a bejegyzések mezői közül a második elmaradhat. Ilyen esetben a bejegyzésre az alapértelmezett TTL érték érvényes, amelyet az állomány legelején, az első bejegyzés előtt kell megadnunk. Az alapértelmezett TTL értéket a \$TTL szám formában kell megadnunk, külön sorban, amely az állomány első sora lesz.

osztály Ez a mező meghatározza, hogy milyen osztályba tartozik az adott bejegyzés. A mezőben a következő kulcsszót használhatjuk:

IN A kulcsszó jelzi, hogy az internetes névszolgáltatáshoz tartozik a bejegyzés. Mivel a DNS kiszolgálót általában kizárolag ilyen célokra használjuk, a bejegyzések mindegyike ehhez az osztályhoz szokott tartozni.

típus Ez a mező megadja, hogy milyen típusú az adott bejegyzés. A következő kulcsszavakat használhatjuk típusként:

SOA A SOA (*start of authority*, hatáskör kezdete) típusú bejegyzés a teljes zónára vonatkozó néhány alapvetően fontos adatot tartalmaz. minden zónaállomány első bejegyzéseként – az alapértelmezés szerinti TTL érték után – pontosan egy SOA bejegyzésnek kell lennie.

A SOA bejegyzések formája kötött, a következő példa szerint alakul:

1	@ IN SOA localhost. root.localhost. (
2	1997022700 ; Sorszám
3	28800 ; Frissítés
4	14400 ; Újra
5	3600000 ; Lejár
6	86400) ; Minimum

A bejegyzés első karaktere a @, amely a zóna nevét helyettesíti a teljes állományban. A SOA bejegyzés neve tehát a zóna nevével egyezik meg! A bejegyzés értékének oszlopai közül az első – példánkban localhost. – a zónához tartozó elsődleges DNS kiszolgáló neve.

A bejegyzés következő típusa – a példánkban root.localhost – a zónáért felelős személy elektronikus levélcíme, amelyben a @ karaktert lecseréltük . karakterre. (Ebből nyilvánvalóan következik, hogy a zónafelelős felhasználói nevében nem szerepelhet . karakter.)

A bejegyzésben következő számok közül az első az állomány változatszáma, amelyet minden módosításkor érdemes frissítenünk, hogy a távoli DNS kiszolgálók tudomására jusson a zóna módosítása a SOA bejegyzés lekérdezésével.

A további számok a zónaadatok távoli kiszolgálón való ideiglenes tárolását vezérlik.

A Az ilyen bejegyzések a számítógép nevéhez rendelnek címet.

Az A bejegyzések használata a következő minta alapján könnyen megérthető:

1	www IN A 10.1.1.1
2	ns.ak.hu. IN A 10.1.1.2

Amint látjuk, az A bejegyzésekhez tartozó név a számítógép neve, a hozzá tartozó érték pedig a számítógép címe.

Az első oszlopban található neveket – ha azok nem a . karakterrel végződnek – a kiszolgáló automatikusan kiegészíti a zóna nevével.

PTR A PTR (*pointer*, mutató) segítségével olyan bejegyzéseket készíthetünk, amelyek más bejegyzésekre mutatnak. Ezt a lehetőséget elsősorban arra használjuk, hogy a címek ismeretében a számítógépek nevét visszakereshessük, azaz fordított DNS bejegyzést hozzunk létre.

Az ilyen típusú bejegyzések használata könnyen megérthető a következő minta alapján:

1	IN PTR	localhost
---	--------	-----------

A bejegyzés neve helyén az 1 áll, amelynek a végére – mivel nem a . karakterrel végződik – a kiszolgáló a zóna nevét illeszti. (A zóna neve fordított DNS zóna esetében minden az in-addr. arpa kulcsszóval végződik, ez jelzi, hogy a zóna címeket képez le nevekre.)

NS Az NS (*name server*, névkiszolgáló) bejegyzés a zónához tartozó DNS kiszolgálók nevének meghatározására vonatkozik. Az ilyen bejegyzéseknek nincs neve, csak értéke, ami a kiszolgáló neve.

Minden zónához legalább egy NS bejegyzésnek lennie kell. A SOA bejegyzésben megadott DNS kiszolgálónak a zónán belül található NS bejegyzésben szerepelnie kell. Az NS bejegyzésnek viszont valamelyik A bejegyzésben kell szerepelnie, ahol IP címet rendelünk a zóna névkiszolgálójához.

Az NS bejegyzés használatát könnyen megérthetjük a következő minta alapján:

1	IN NS	ns1
---	-------	-----

Figyeljük meg, hogy az NS bejegyzésnek nincs neve, az értéke pedig a DNS kiszolgáló neve, amelynek végéhez – mivel nem . karakterrel végződik – a kiszolgáló automatikusan hozzáilleszti a zóna nevét.

CNAME A CNAME (*canonical name*, szabályos név) kulcsszó segítségével másodlagos nevekhez tartozó szabályos (elsődleges) nevet megadó bejegyzéseket készíthetünk. Az ilyen bejegyzések segítségével egy IP címhez több nevet is rendelhetünk.

A CNAME típusú bejegyzések használatát könnyen megérthetjük a következő minta alapján:

1	www	IN CNAME	server
---	-----	----------	--------

A bejegyzés szerint a www másodlagos név a server elsődleges névre vonatkozik. A DNS kiszolgáló mind a másodlagos, mind pedig az elsődleges név végére másolja a zóna nevét, hiszen egyik sem . karakterrel végződik.

Fontos megjegyeznünk, hogy a CNAME bejegyzés mindenkor elődleges – azaz A bejegyzésben szereplő – névhez rendeli a másodlagos nevet, a CNAME bejegyzés utolsó oszlopában tehát nem állhat másodlagos név. Fontos lehet tudnunk azt is, hogy bizonyos célokra nem használhatunk másodlagos nevet. (SOA, NS, MX bejegyzésben nem szerepelhet másodlagos név.)

- MX Az MX (*mail exchanger*, levéltovábbító) kulcsszó segítségével megadhatjuk, hogy a zónához tartozó e-mail címekre érkező leveleket melyik számítógép fogadja.

Az MX bejegyzéseket a levéltovábbító programok használják – amikor elektronikus levelet küldenek a mi zónánkba – annak eldöntésére, hogy melyik számítógép fogadja a leveleket. Az elektronikus levelek megérkezhetnek hozzáink akkor is, ha a zónához nem tartozik egyetlen MX bejegyzés sem, mégis hasznos a használatuk, mivel a segítségükkel másodlagos levelfogadó gépeket jelölhetünk ki, amelyek biztonsági tartalékokat képeznek.

Az MX bejegyzések használatát könnyen megérthetjük a következő minta alapján:

1	IN	MX	10	mail1
2	IN	MX	20	mail2
3	barat.hu.	IN	MX	20 mail1

A mintában láthatjuk, hogy az MX bejegyzésekhez nem csak a leveleket fogadó gépek neve tartozik, hanem egy szám is. A szám megadja a levelfogadó számítógépek használatának sorrendjét. A levelet küldő számítógép mindenkor az alacsonyabb számmal jelzett számítógépnek próbálja meg elküldeni a levelet, a magasabb számmal jelzett számítógépet csak hiba esetén keresi fel.

Amint a mintában is látjuk, az MX bejegyzés első oszlopa elmaradhat. Ilyen esetben a DNS kiszolgáló automatikusan behelyettesíti a zóna nevét.

A minta harmadik sorában megadtuk egy zóna nevét. Ezt a bejegyzést arra használjuk, hogy egy távoli zóna számára üzemeltessünk tartalék levelfogadó kiszolgálót.

adatok A bejegyzések utolsó oszlopában (vagy oszlopaiban) a bejegyzéshez tartozó adatok találhatók. A különféle bejegyzéstípusokhoz – amint láttuk – különféle adattípusok tartoznak.

A sok új ismeret bemutatása után nyilvánvalóan hasznos lesz egy példa, amely egy olyan DNS kiszolgáló beállítóállományait mutatja be, amely egy névtartamnyt szolgáltat. A következő példa receptként felhasználható a bind beállítása során.

50. példa. Készítsük el a BIND számára a beállítóállományt és az adatokat tartalmazó állományokat egy tartomány DNS és fordított DNS szolgáltatásához!

Első lépésként készítsük el a beállítóállományt, amely a következő sorokat tartalmazza:

```
1 options {
2     directory "/var/named";
3 };
4
5 zone "1.10.in-addr.arpa" {
6     type master;
7     file "db.1.10";
8 };
9
10 zone "lkk" {
11     type master;
12     file "db.lkk";
13 };
```

/etc/named.conf

Figyeljük meg, hogy az állományban beállítottuk az adatállományok tárolására használt könyvtár nevét az options szakaszban, és két zónát hoztunk létre a fordított DNS és a DNS lekérdezések számára. A zónák leírása csak a zónaneveket és az adatok tárolására szolgáló állományok neveit tartalmazza.

A következő lépés a db.lkk állomány létrehozása és kitöltése a /var/named/könyvtárban:

```
1 $TTL 3h
2 @ IN SOA ns1.lkk. hostmaster.ns1.lkk. (
3                         20040711
4                         43200
5                         3600
6                         86400
7                         86400 )
8             IN NS   ns1
9 ns1          IN A    10.1.1.1
10 notebook    IN A    10.1.1.100
```

/var/named/db.lkk

Figyeljük meg, hogy SOA bejegyzés után a NS bejegyzést találjuk. Ezek után két A bejegyzés következik, amelyek közük az első kötelező (minden NS bejegyzéshez kell tartoznia legalább egy A bejegyzésnek), a második pedig nem.

A következő lépés a db.1.10 állomány létrehozása a /var/named/ könyvtárban. Ez az állomány a következő sorokat tartalmazza:

```

1 $TTL 3h
2 @ IN SOA ns1.lkk. hostmaster.ns1.lkk. (
3                               20040711
4                               43200
5                               3600
6                               86400
7                               86400 )
8           IN NS      ns1.lkk.
9   1.1     IN PTR    ns1.lkk.
10  100.1   IN PTR    notebook.lkk.

```

Figyeljük meg, hogy az állomány az előbb bemutatott állományhoz hasonló, az ott tárolt A bejegyzésekhez tartozó PTR, azaz fordított bejegyzéseket tartalmazza.

Ha elindítjuk a DNS kiszolgálót, és beállítjuk a /etc/resolv.conf állományban, hogy a számítógépünk használja is, az adatbázisokban elhelyezett neveket le is kérdezhetjük:

```

$ host notebook.lkk
notebook.lkk has address 10.1.1.100
$ host 10.1.1.100
100.1.1.10.in-addr.arpa domain name pointer notebook.lkk.
$ 

```

Amint látjuk, minden tökéletesen működik.

5.4.1. A másodlagos névkiszolgáló

A másodlagos DNS kiszolgáló fontos szerepe akkor érvényesül, ha valamilyen hiba miatt nem elérhető az elsődleges DNS kiszolgáló. Ha nem használunk másodlagos névkiszolgálót a saját zónáink számára, és az elsődleges névkiszolgáló nem működik, az egész világon elérhetetlennek válnak a zónánkhoz tartozó számítógépnevek, így az általunk üzemeltetett szolgáltatások elérhetetlennek válnak. (Nem is beszélve arról, hogy a hálózatunkon megáll az élet, mert senki sem tudja a körülötte található számítógépek neveit.)

Fontos tehát, hogy másodlagos DNS kiszolgálót – vagy másodlagos kiszolgálókat – üzemeltessünk, sőt ajánlatos a másodlagos DNS kiszolgálót minél messzebb elhelyezni az elsődleges kiszolgálótól. Nyilvánvaló, hogy semmire sem megyünk a másodlagos kiszolgálóval, ha éppen az a hálózati eszköz megy tönkre, amelyen az elsődleges kiszolgálóval osztoznak⁴.

⁴Tartománynév igényléskor feltétel, hogy rendelkezzünk egy külön hálózaton elhelyezett másodlagos DNS kiszolgálóval (a lektor).



Az üzemeltetést végző szakemberek szokás szerint keresnek egy kollégát egy másik városban, és kölcsönösen másodlagos DNS szolgáltatást nyújtanak egymásnak. Ez a módszer legalább olyan zseniális, mint amilyen egyszerű. Abban az esetben, ha fontos szolgáltatásokat nyújtunk a nagyvilág számára, mindenkorban javasolható a kölcsönös DNS szolgáltatás bevezetése.

A másodlagos DNS kiszolgáló beállítása igen egyszerű. Egyszerűen létre kell hoznunk a zónákat a másodlagos kiszolgálón is, be kell jegyeznünk a type kulcsszóval, hogy ezek másodlagos zónák, és meg kell adnunk az elsődleges zóna helyét. Ez utóbbi feladatra a zóna létrehozásakor a masters kulcsszó használható a következő formában:

`masters {cím1; cím2;};` A kifejezés segítségével a másodlagos zónaként szolgáltatott zónához rendelhetünk elsődleges DNS kiszolgálót. minden másodlagos zónához legalább egy elsődleges kiszolgálót meg kell adnunk.

Elsődleges DNS kiszolgálóból többet is megadhatunk, de a legtöbb esetben csak egy elsődleges kiszolgálót jegyzünk be a kifejezés segítségével.

A masters kulcsszó hatására a névkiszolgáló induláskor letölti a teljes zónát az elsődleges kiszolgálóról, majd a megszokott formában DNS szolgáltatást nyújt. A másodlagos kiszolgáló a zónához tartozó SOA bejegyzés alapján a zóna adatait időről időre frissíti.

Másodlagos DNS kiszolgálót tehát igen egyszerűen készíthetünk az általunk kezelt zónához. Ezt mutatja be a következő példa:

51. példa. Helyezzünk üzembe másodlagos DNS kiszolgálót a lehető legegyszerűbb módon az általunk kezelt zóna számára!

Legyen az elsődleges DNS kiszolgálón a `/etc/named.conf` állomány tartalma a következő:

```

1 options {
2   directory "/var/named";
3 };
4
5 zone "1.10.in-addr.arpa" {
6   type master;
7   file "db.1.10";
8 };
9
10 zone "lkk" {
11   type master;
12   file "db.lkk";
13 };

```

A legegyszerűbb, ha ezt az állományt átmásoljuk a másodlagos DNS szolgáltató szerepére kiszemelt számítógépre, és csekély mértékben módosítjuk. Legyen a másodlagos kiszolgálón található /etc/named.conf a következő:

```
1 zone "1.10.in-addr.arpa" {
2     type slave;
3     masters { 10.1.1.1; };
4 };
5
6 zone "lkk" {
7     type slave;
8     masters { 10.1.1.1; };
9 };
```

Amint látjuk, a DNS és a fordított DNS szolgáltatás céljára használt zónát másodlagos zónaként létrehoztuk, és megadtuk, hogy melyik számítógépen található az elsődleges példány.

Természetesen sokat finomíthatunk a beállításokon, de a másodlagos DNS kiszolgáló már ezekkel a beállításokkal is üzemképes.

Ha azt akarjuk, hogy a másodlagos DNS kiszolgáló a letöltött bejegyzéseket állományba is mentse, a szokásos módon – a file kulcsszóval – adhatunk meg állományneveket a zónákon belül. Nem szabad azonban szem elől téveszteni, hogy a másodlagos DNS kiszolgálónak a letöltött adatokat ezekben az állományokban el kell helyeznie, a named programnak tehát írás jogra van szüksége az állományokhoz.

5.4.2. Feloldó gyorsítótáras névkiszolgáló

Sokszor előfordul, hogy nem akarunk saját zónát létrehozni, nem kívánunk saját névtartományt üzemeltetni, mégis szükségünk van a bind programra feloldó gyorsítótáras névkiszolgáló üzemeltetéséhez (*caching name server*).

A feloldó gyorsítótáras névkiszolgáló nem tart nyilván saját zónákat, nem feladata a körülbelül számára nyújtandó DNS szolgáltatás, kizárolag a saját számítógép-hálózatunk felé szolgáltat. Az ilyen gyorsítótáron alapuló szolgáltatás működését az alábbi pontokban foglalhatjuk össze:

- A saját hálózatunkon található munkaállomások minden esetben a helyi, feloldó gyorsítótáras DNS kiszolgálóhoz fordulnak névinformációkért.
- A DNS kiszolgáló megbízottként jár el, a hivatalos DNS kiszolgálókat felkeresve kideríti a névinformációkat a munkaállomások helyett.
- A feloldó gyorsítótáras névkiszolgáló a lekért DNS információkat tárolja, a következő kérést tehát ki tudja szolgálni a hivatalos DNS kiszolgáló felkeresése nélkül.

Ha tehát feloldó gyorsítótáras névkiszolgálót üzemeltetünk, a DNS kérések ki-szolgálása gyorsabb lehet, és a külvilág felé kiépített hálózatunk terhelése csökkenhet, hiszen nem kell minden egyes DNS kérés miatt a távoli névkiszolgálókhöz fordulni.



Ha időszakos internetkapcsolattal rendelkezünk, a feloldó gyorsítótáras névkiszolgáló használatának közvetlenül érzékelhető anyagi haszna is lehet. Ennek oka az, hogy a távoli DNS információk helyi tárolása miatt nem kell minden egyes DNS kérés miatt felépíteni a hálózati kapcsolatot, nem kell minden név feloldásához „kitárcsázni”.

Feloldó gyorsítótáras névkiszolgáló üzemeltetéséhez egy egyszerű bejegyzést kell elhelyeznünk a bind beállítóállományában, a következő formában:

```
1 zone "." IN {
2     type hint;
3     file "named.ca";
4 }
```

A bejegyzés megadja, hol található a „.” tartomány névkiszolgálóit felsoroló állomány és azt, hogy a teljes névtartomány elérhető ezeken a kiszolgálókon ke-restülről. A feloldó gyorsítótáras névkiszolgáló beállítóállományai és a „.” névtarto-mányhoz tartozó névkiszolgálók listája a legtöbb GNU/Linux terjesztéshez elér-hető külön programcsomagként (például caching-nameserver néven).

5.4.3. Biztonsági beállítások

A DNS kiszolgáló létfontosságú adatokat szolgáltat az egész világnak, így ter-mészetes, hogy a biztonsági beállítások fontos szerepet kapnak az üzemeltetése során. Az Interneten meglehetősen sok ötletet, módszert találhatunk a DNS biz-ttonságos üzemeltetéséhez, mi azonban csak néhány egyszerű, elengedhetetlenül fontos eszközt mutatunk be.

A BIND mai változatai lehetővé teszik, hogy néhány kifejezéssel korlátozzuk a DNS szolgáltatás igénybe vételét. Ezeket a kifejezéseket az options szakaszban is elhelyezhetjük, hogy az összes zónára érvényesek legyenek, és a zone szaka-szokban is, hogy az adott zónára legyenek érvényesek. A zónában elhelyezett kifejezések minden felülbírálják az options szakaszban elhelyezett kifejezések ha-tását, vagyis az alapértelmezett beállításokat az options szakaszban kell elhelyez-nünk, az egyes zónában pedig az ezektől való eltéréseket kell felsorolnunk.

A BIND beállítóállományában az options és a zone szakaszokon belül a követ-kező kifejezéseket használhatjuk a biztonsági korlátozások bevezetésére:

`allow-query {cím1; cím2;};` A kifejezés segítségével megadhatjuk, hogy az egyes címekkel rendelkező vagy címtartományokba tartozó számítógépek számára DNS szolgáltatást végzünk-e. Alapértelmezett esetben minden számítógép számára engedélyezett a szolgáltatás igénybe vétele.

Nyilvánvaló, hogy az általunk kezelt zónák adatait idegenek számára is szolgáltatnunk kell, más zónákat azonban legfeljebb csak a saját hálózatunk munkaállomásai számára szükséges lekérdeznünk. Elsősorban azért van szükség arra, hogy idegen zónákat szolgáltassunk, hogy azok a munkaállomások is DNS szolgáltatást használhassanak, amelyek nem tudnak maguk utánajárni az adatoknak, de – amint azt láttuk – gyorsíthatja is a hálózat működését, ha egy számítógépen kezeljük a DNS lekérdezéseket.

`allow-recursion {cím1; cím2;};` A kifejezés segítségével megadhatjuk, hogy mely munkaállomások számára végzünk teljes felderítést, azaz mely munkaállomások számára vagyunk hajlandóak utánajárni más DNS kiszolgálókon az adatoknak. Alapértelmezés szerint minden számítógép számára engedélyezett ez a szolgáltatás.

Nyilvánvaló, hogy ezt a szolgáltatást csak a saját hálózatunkon található munkaállomások számára kell elvégeznünk.

`allow-transfer {cím1; cím2;};` A kifejezés segítségével beállíthatjuk, hogy mely munkaállomások számára engedélyezett teljes zónák letöltése. Alapértelmezés szerint minden számítógép számára engedélyezett ez a szolgáltatás.

Nyilvánvaló, hogy a másodlagos DNS kiszolgálók számára engedélyezni kell ezt a szolgáltatást, hogy időről időre letölthessék a teljes zónákat, hiszen azokat szükség esetén nekik is szolgáltatniuk kell.

Szerencsés tiltani a zónák letöltését külső munkaállomások számára, hiszen a zónaletöltés egyrészt feleslegesen terhel a kiszolgálót és a hálózatot, másrészt a segítségével a támadók gyorsan és egyszerűen mérhetik fel a teljes hálózatunkat.

A kifejezésekben szereplő címek helyén megadhatjuk az any kulcsszót, amely minden címre érvényes, és teljes címtartományokat is a `cím/hálóbitek_száma` formában.

A DNS szolgáltatások jogait korlátozó kifejezések használatát mutatja be a következő példa.

52. példa. Készítünk elsődleges és másodlagos DNS kiszolgálóbeállításokat, amelyek tartalmazzák a megfelelő biztonsági korlátozásokat is.

Az elsődleges DNS kiszolgáló beállítására szolgáló `/etc/named.conf` legyen a következő:

```
1 options {
2     directory "/var/named";
```

```
3  /*
4   * Alapértelmezés szerint csak a saját
5   * munkaállomások kérdezhetnek le DNS bejegyzéseket.
6   */
7  allow-query {10.1.1.1/8; localhost; };
8  /*
9   * Csak a saját munkaállomások számára járunk utána
10  * a kéréseknek.
11  */
12 allow-recursion{ 10.1.1.1/8; localhost; };
13 query-source address * port 53;
14 };
15
16 /*
17 * Gyorsítótáras névszolgáltatás: alapértelmezett
18 * beállítások, csak a saját gépeink számára.
19 */
20 zone "." IN {
21     type hint;
22     file "named.root";
23 };
24
25 /*
26 * Saját zónák következnek.
27 */
28 zone "1.10.in-addr.arpa" {
29     type master;
30     file "db.1.10";
31     /*
32      * A másodlagos DNS kiszolgálónk számára a zóna
33      * letöltése engedélyezett. Szintén engedélyezett a
34      * saját gép számára, hibakeresés céljából.
35      */
36     allow-transfer { 10.1.1.100; localhost; };
37     /*
38      * Ezt a zónát az egész világ számára szolgáltatjuk.
39      */
40     allow-query{ any; };
41 };
42
43 zone "lkk" {
44     type master;
45     file "db.lkk";
46     /*
```

```

47  * A másodlagos DNS kiszolgálónk számára a zóna
48  * letöltése engedélyezett. Szintén engedélyezett a
49  * saját gép számára, hibakeresés céljából.
50  */
51  allow-transfer { 10.1.1.100; localhost; };
52  /*
53  * Ezt a zónát az egész világ számára szolgáltatjuk.
54  */
55  allow-query{ any; };
56 };

```

A beállítóállomány magától értetődő módon gyorsítótáras névszolgáltatást nyújt a saját hálózatunkon üzemelő munkaállomások számára és egy névtartomány két zónáját szolgáltatja az egész világnak.

A következő sorok a másodlagos DNS kiszolgálónk /etc/named.conf beállítóállományát mutatják be. A beállítások az elsődleges DNS kiszolgálóhoz hasonlóan alakulnak, kivéve, hogy erről a számítógépről más számítógépek teljes zónákat nem tölthetnek le, mert erre nincs szükség.

```

1 options {
2   directory "/var/named";
3   allow-query {10.1.1.1/8; localhost; };
4   allow-recursion{ 10.1.1.1/8; localhost; };
5   /*
6    * Kizárolag hibakeresés céljából.
7    */
8   allow-transfer { localhost; };
9   query-source address * port 53;
10 };
11 /*
12  * Gyorsítótáras névszolgáltatás: alapértelmezett
13  * beállítások, csak a saját gépeink számára.
14  */
15 zone "." IN {
16   type hint;
17   file "named.root";
18 };
19 /*
20  * Saját zónák következnek.
21  */
22 zone "1.10.in-addr.arpa" {
23   type slave;

```

```

26 | masters { 10.1.1.1; };
27 |   file "db.1.10";
28 |   allow-query { any; };
29 | };
30 |
31 zone "lkk" {
32   type slave;
33   masters { 10.1.1.1; };
34   file "db.lkk";
35   allow-query{ any; };
36 };

```

A beállítások szerint a másodlagos DNS kiszolgálónk gyorsítótáras névszolgáltatást nyújt a saját számítógépeink számára, valamint a saját zónáinkat szolgáltatja az egész világnak.

A DNS kiszolgáló beállításairól szóló rész végén meg kell jegyeznünk, hogy több, grafikus felülettel is rendelkező program érhető el a BIND beállítására. A 5.1. ábrán láthatjuk például a `system-config-bind` programot, amely Red Hat alapú rendszereken használható a névkiszolgáló beállítására.



5.1. ábra. A BIND beállításai grafikus felületen

5.5. A hálózati beállítások szolgáltatása

Nagyobb hálózatok üzemeltetése során a hálózatban található munkaállomások hálózati adatainak beállítása és nyilvántartása terhes, unalmas feladat. Ráadásul,

ha a hálózati adatok valamelyike megváltozik, a rendszergazdának az összes munkaállomáson egyenként meg kell változtatnia a beállításokat, ami meglehetősen lassú és türelmet igénylő munka. Érdemes tehát automatikussá tennünk a hálózati adatok lekérdezését, hogy könnyebben kezelhető és rugalmasabb legyen az általunk üzemeltetett számítógép-hálózat.

A hálózati beállítások központi tárolását és a munkaállomások automatikus hálózati beállítását támogatja a BOOTP (*internet bootstrap protocol*, internethoz rendszereindító protokoll) és a DHCP (*dynamic host configuration protocol*, dinamikus számítógép-beállító protokoll)[41] is, melyek közül a BOOTP a régebbi, a DHCP pedig vele felülről csereszabatos. A következő néhány oldalon a DHCP szolgáltatás üzeme helyezéséről és karbantartásáról olvashatunk.

5.5.1. A DHCP rendszer működése

A DHCP a hálózati adatok lekérdezésére alkalmas protokoll, amelynek működése meglehetősen egyszerű. Lényege, hogy hálózati beállításokat nem a munkaállomásokon, hanem egy központi számítógépen, a DHCP kiszolgálón tároljuk.

Amikor a munkaállomást bekapcsoljuk, és azon a hálózati szolgáltatások elindulnak, a hálózati csatolóján keresztül egy körüzenetet küld, amelyben kéri, hogy a hálózaton található DHCP kiszolgáló értesítse az érvényes hálózati beállításokról. A számítógép kizárálag körüzenetet képes küldeni a hálózaton, hiszen semmilyen információ nem áll rendelkezésre a hálózat beállításairól. A kiszolgálónak ezért az adott hálózaton kell lennie, hiszen az útválasztók nem továbbítják a körüzeneteket, így azok más hálózaton található DHCP kiszolgálóhoz nem jutnak el.

A DHCP kiszolgáló, miután megkapta a kérést, a beállítóállományában megvizsgálja, hogy a kérést küldő számítógép szerepel-e a nyilvántartásban. A kiszolgáló a keresést a DHCP ügyfélként jelentkező munkaállomás Ethernet címe alapján végzi. Ha a munkaállomás szerepel a nyilvántartásban, a DHCP kiszolgáló a rá vonatkozó hálózati beállításokat elküldi, így az ügyfél azokat felhasználhatja a hálózat beállításához.

A DHCP kiszolgáló képes azokat az ügyfeleket is kezelní, amelyek nem szerepelnek a nyilvántartásában! Ha a rendszergazda úgynevezett dinamikusan kiosztatott IP címtartományt hozott létre a beállítások között, a kiszolgáló keres egy használaton kívüli címet, és azt osztja ki használatra.

A dinamikusan kiosztott címtartomány tehát olyan IP címeket tartalmaz, amelyeket a munkaállomások igény szerint használhatnak, a DHCP kiszolgáló pedig gondoskodik arról, hogy egyszerre mindig csak egy ügyfél kapja meg az egyes címeket. Az IP címek dinamikus kiosztásának következetében nem garantált, hogy az ügyfél minden bekapcsolás során ugyanazt az IP címet kapja, de garantált, hogy minden működő hálózati beállítást kap.

A dinamikusan kiosztatható címkészletbe tartozó címek száma és az ügyfelek száma közt természetesen meg kell találnunk az egyensúlyt. A legfontosabb, hogy minden bekapcsolt számítógépre jusszon legalább egy kiosztatható cím. Ha a hálózaton például 100 ügyfél található, melyek közül egy időben átlagosan 10 van be-

kapcsolva, elegendő lehet egy 25 címet magába foglaló dinamikusan kiosztatható címtartomány használata. Nem annyi IP címre van tehát szükségünk, amennyi munkaállomásunk van, hanem annyira, amennyi egyszerre be van kapcsolva. Ez a DHCP használatának egyik komoly előnye.

A dinamikusan kiosztott címtartomány használatának kellemes hatása, hogy a DHCP kiszolgálónak nem kell ismernie a munkaállomást ahhoz, hogy számára használható beállításokat adjon át. Ha egy vendég érkezik a munkahelyünkre, és csatlakoztatja hordozható számítógépét a hálózatra, a DHCP kiszolgáló gondoskodhat számára a hálózati beállításokról, és IP címet foglalhat neki. A hordozható számítógépek elterjedésével ez a lehetőség mérhetetlenül megkönníti a hálózati rendszergazda és a felhasználók életét is.

A DHCP dinamikus jellege nem csak az IP címek kiosztása közben megfigyelhető címcserékre vonatkozik, hanem azt is jelenti, hogy a kiszolgáló képes kierőszakolni a hálózati adatok megváltoztatását. Ez azt jelenti, hogy amikor a DHCP ügyfél megkapja a hálózati beállításokat, megtudja azt is, hogy azok meddig érvényesek. Ha a hálózati beállítások érvényessége lejár, a DHCP ügyfélnek új hálózati beállításokat kell kérnie! Ha tehát a hálózati beállításokat a DHCP kiszolgálón megváltoztatjuk, a megváltozott értékek a beállított idő elteltével minden munkaállomáshoz eljutnak.

Nem szabad lebecsülnünk a hálózati beállítások bonyolultságát! Látni fogjuk, hogy a DHCP beállítóállományban igen sokféle hálózati beállítás szerepelhet. Mivel ezeket a DHCP kiszolgáló által megkövetelt formában kell a beállítóállományban elhelyeznünk, nem kell ismernünk a munkaállomás hálózati beállítások kezelésére használt programjait. Ha a hálózaton sokféle operációs rendszer található a munkaállomásokon, a hálózati rendszergazdának nagymértékben egyszerűsíti a munkáját a hálózati beállítások egységes formátuma. Nem kell tudnia, *hogyan* kell beállítani a hálózati adatokat az egyes operációs rendszerek esetében, csak azt kell tudnia, *mit* kell beállítania.

Végezetül meg kell említenünk egy ritkábban használt, de annál érdekesebb szolgáltatást, amelyet mind a BOOTP, mind pedig a DHCP rendszer képes kezelni. A professzionális munkaállomások legtöbbje, de ma már a személyi számítógépek némelyike is képes BOOTP vagy DHCP szolgáltatást használni az operációs rendszer segítsége nélkül is. (Valójában a hálózati kártyán elhelyezett ROM memória segítségével a régebbi személyi számítógépek is alkalmassá tehetők erre.)

Ha a munkaállomás az operációs rendszer nélkül is képes DHCP kérést küldeni és a választ fogadni, akkor kihasználhatjuk a DHCP kiszolgáló rendszerindító szolgáltatását. Ennek segítségével a munkaállomást utasíthatjuk valamely állomány letöltésére és arra, hogy a letöltött állományt operációs rendszerként indítsa el. Ez igen érdekes lehetőség, hiszen így olyan számítógépeket is használhatunk munkaállomásként, amelyekben nincs merevlemez vagy más háttértár, és amelyek az operációs rendszert bekapcsoláskor a hálózatról töltik le! Ezt a csodálatos lehetőséget kihasználva a munkaállomások operációs rendszerét központi, biztonságos helyen tárolhatjuk, védve a felhasználótól, a felhasználó unokaöccsétől és annak barátaitól a 4/D osztályból.

5.4. tábla: dhcpcd

DHCP szolgáltatást nyújtó program.

`dhcpcd [kapcsolók]`

A dhcpcd program DHCP szolgáltatást nyújt, más számítógépek számára adja át a hálózat használatához szükséges legfontosabb adatokat. A programot általában a szolgáltatások esetében megszokott módon, a `/etc/init.d/` könyvtárban található héjprogrammal indítjuk.

Kapcsoló	Jelentés
-f	Alapértelmezés szerint a program indítás után démonként, a háttérben fut. Ezt a viselkedést tilthatjuk ezzel a kapcsolóval.
-d	Hibakereső üzemmód, nyomkövető üzenetek kiírásával.
-cf fájl	A beállítóállomány megadása.
-lf fájl	A kiosztott IP címek adatait tartalmazó állomány nevének megadása.
-t	A szolgáltatás tiltása: a program csak a beállítóállomány ellenőrzését végzi el és kilép.

5.5.2. A DHCP kiszolgáló üzembe helyezése

A legtöbb GNU/Linux terjesztésben rendelkezésünkre áll az *Internet Systems Consortium* által készített DHCP kiszolgáló, amely BOOTP és DHCP szolgáltatást valósít meg.

A DHCP kiszolgálót megvalósító program a dhcpcd, amely szolgáltatásként a szokásos módon indítható és állítható le (lásd 1. példa a 14. oldalon). A dhcpcd program beállítóállománya a `/etc/dhcpcd.conf` szöveges állomány. Ebben az állományban a legfontosabb kifejezések a következők:

`subnet cím netmask maszk{...}` A kifejezés segítségével egy alhálót leíró szakaszat adhatunk meg a DHCP kiszolgálónak. Az alháló leírásában a `{}` jelek között szerepel minden, kizárolag az alhálóra vonatkozó információ.

A DHCP kiszolgálónak minden hálózati csatoló számára szüksége van egy ilyen leírásra. minden hálózati csatolóhoz készítenünk kell egy leírást a beállítóállományba vagy tiltanunk kell az DHCP szolgáltatást az adott hálózati csatolon⁵.

`range cím1 cím2;` A kifejezés segítségével tetszőleges címtartományt jelölhetünk ki, hogy az abban található IP címeket a DHCP kiszolgáló dinamikus

⁵A DHCP kiszolgáló újabb változatainak nem kell minden hálózati csatolóhoz ilyen leírást készítenünk.

módon kioszsa a jelentkező számítógépeknek. A kifejezésnek a subnet szakaszon belül kell lennie, és illeszkednie kell a subnet szakasz által leírt alhálóba.

`option routers cím;` A kifejezés segítségével az alapértelmezett átjáró címét adhatjuk meg a DHCP ügyfeleknek.

Mivel általában minden alhálón más-más átjárót használunk, a kifejezés a legtöbb esetben az alhálót leíró subnet kifejezésen belül található.

`option subnet-mask maszk;` A kifejezés segítségével megadhatjuk az alhálózati maszkot a DHCP ügyfelek számára.

Mivel az alhálózati maszk az alhálóra vonatkozik, ezt a kifejezést általában az alhálót leíró subnet kifejezésen belül használjuk.

`option domain-name "tartománynév";` A kifejezéssel a tartománynevet adhatjuk meg a DHCP ügyfelek számára.

Mivel sokszor ugyanazt a tartománynevet több alhálón is használjuk, ezt a kifejezést sokszor az alhálókat leíró subnet szakaszokon kívül helyezzük el.

`option domain-name-servers cím1, cím2;` A kifejezés segítségével a DNS kiszolgáló – vagy kiszolgálók – címeit adhatjuk meg a DHCP ügyfelek számára. A kifejezésbe egy vagy több kiszolgáló címét is beírhatjuk, amelyeket a sorrend megváltoztatása nélkül fog a DHCP ügyfél elhelyezni a /etc/resolv.conf állományban.

Mivel a legtöbb esetben az összes alhálón ugyanazokat a DNS kiszolgálókat használjuk, ezt a kifejezést általában az alhálókat leíró subnet kifejezéseken kívül adjuk meg.

`ddns-update-style interim;` A kifejezés megadja, hogy a DHCP kiszolgáló minden módon frissítse a DNS kiszolgáló adatbázisát, amikor bejegyzést készít a névkiszolgálón az adott ügyfél nevével és IP címével. Erre a szolgáltatásra sokszor nincs szükségünk, a kifejezésnek azonban szerepelnie kell a beállítóállományban.

Mivel ez a kifejezés a DHCP kiszolgáló működését befolyásolja, általában az alhálókat leíró subnet kifejezéseken kívül adjuk meg.

A dokumentáció szerint a DNS bejegyzések frissítésére vonatkozó eszközök és beállítások a közeljövőben változni fognak.

A következő példa a /etc/dhcpd.conf beállítóállomány létrehozásának első lépésein mutatja be.

53. példa. Készítsük el a lehető legegyszerűbb beállítóállományt a /etc/dhcpd.conf állományban a DHCP kiszolgáló számára, amely lehetővé teszi az IP címek és a legfontosabb hálózati adatok szolgáltatását!

```

1   /etc/dhcpd.conf
2   ddns-update-style interim;
3   option domain-name "otthon";
4   option domain-name-servers 10.1.1.100;
5
6   subnet 10.0.0.0 netmask 255.0.0.0 {
7     option subnet-mask 255.0.0.0;
8     option routers 10.1.1.254;
9     #
10    # Dinamikusan kiosztott címtartomány
11    #
12    range 10.1.1.2 10.1.1.100;
13 }
```

A példa tartalmaz minden elemet, amely egy DHCP kiszolgáló üzembe helyezéséhez szükséges, a DHCP ügyfelek pedig a példa alapján minden szükséges információt megkapnak a hálózat használatához.

Figyeljük meg, hogy a beállítóállomány egyetlen munkaállomásról sem tartalmaz közvetlen adatokat, csak a 10.1.1.2 címtől 10.1.1.100 címig terjedő tartományt osztja ki!

5.5.3. A munkaállomások nyilvántartásba vétele

Ha egy vagy több munkaállomást nem dinamikus módon akarunk ellátni IP címmel, hanem számukra állandó jelleggel kívánunk hálózati adatokat biztosítani, a munkaállomásokat nyilvántartásba kell vennünk.

A nyilvántartás alapja a hálózati kártya címe, amely az Ethernet hálózatokban a jól ismert hatbájtos Ethernet cím. A számítógépek nyilvántartásba vételéhez használhatjuk a következő kifejezéseket a DHCP kiszolgáló beállítóállományában:

host gépnév { ... } Minden host kifejezés egy-egy számítógép nyilvántartásba vételére szolgál. Amikor a DHCP kiszolgálóhoz egy kérés érkezik, az előbb a host kifejezések segítségével nyilvántartott számítógépek között próbálja megkeresni a munkaállomást, és csak akkor oszt ki a dinamikusan kiosztott címtartományok közül címet, ha a munkaállomás nincs nyilvántartva.

A host kifejezés a legtöbb esetben a subnet kifejezésen belül található, így a nyilvántartásba vett számítógépre érvényesek a subnet rész beállításai. Ha a nyilvántartásba vett számítógépeket a subnet kifejezésektől független módon – például gyártó szerint – kívánjuk csoportosítani, használhatjuk a group kifejezést is a csoportosításhoz.

group { ... } A nyilvántartásba vett munkaállomások csoportosítására használható kifejezés. Ezt a kifejezést nem kötelező használnunk!

A group kifejezésen belül felsorolhatjuk az adott csoportra érvényes beállításokat, majd a nyilvántartásba vett munkaállomásokat, így a csoportra érvényes beállításokat nem kell minden munkaállomás esetében egyenként megadnunk.

hardware ethernet ethernet_cím; A kifejezés segítségével azonosítjuk a munkaállomást az Ethernet címével.

Ennek a kifejezésnek mindenképpen szerepelnie kell a munkaállomás azonosítására szolgáló részen belül.

fixed-address cím; A kifejezés segítségével állandó IP címet rendelhetünk a nyilvántartásba vett munkaállomáshoz.

A **fixed-address** kifejezésnek a munkaállomás nyilvántartásba vételére szolgáló host kifejezésen belül kell lennie.

A következő példa bemutatja, hogyan tudunk állandó IP címet rendelni a munkaállomáshoz a /etc/dhcpd.conf állományban.

54. példa. Alakítsuk át a 53. példa beállítóállományát úgy, hogy nyilvántartásba veszünk egy munkaállomást, és ahhoz állandó IP címet rendelünk!

Vizsgáljuk meg a DHCP kiszolgáló által elhelyezett naplóbejegyzéseket, és keressük ki azokat, amelyek az adott munkaállomásra vonatkoznak!

```
Jul 11 10:12:27 gep dhcpcd: DHCPREQUEST for 10.1.1.99 from 00:02:3f:15:34:bb via eth0
Jul 11 10:12:27 gep dhcpcd: DHCPACK on 10.1.1.99 to 00:02:3f:15:34:bb via eth0
```

Látjuk, hogy a munkaállomás Ethernet címe 00:02:3f:15:34:bb. Ezt a címet használjuk fel a /etc/dhcpd.conf állomány bővítésekor:

```

1 ddns-update-style interim;
2 option domain-name "otthon";
3 option domain-name-servers 10.1.1.100;
4
5 subnet 10.0.0.0 netmask 255.0.0.0 {
6   option subnet-mask 255.0.0.0;
7   option routers 10.1.1.254;
8   #
9   # Dinamikusan kiosztott címtartomány
10  #
11  range 10.1.1.2 10.1.1.99;
12  host notebook {
13    hardware ethernet 00:02:3f:15:34:bb;
14    fixed-address 10.1.1.100;
15  }
16 }
```

Figyeljük meg, hogy a nyilvántartásba vett munkaállomáshoz a 10.1.1.100 IP címet rendeltük, ezt a dinamikusan kiosztható címek tartományából eltávolítottuk, és a nyilvántartásba az Ethernet címet pontosan beírtuk!

Indítsuk most el a DHCP kiszolgálót a dhcpcd -d kapcsolójával, amely a hibakeresést könnnyíti meg, majd a munkaállomáson indítsuk újra a hálózatot:

```
$ dhcpcd -d
Internet Software Consortium DHCP Server V3.0pl1
Copyright 1995-2001 Internet Software Consortium.
All rights reserved.
For info, please visit http://www.isc.org/products/DHCP
Wrote 0 deleted host decls to leases file.
Wrote 0 new dynamic host decls to leases file.
Wrote 1 leases to leases file.
Listening on LPF/eth0/00:e0:29:29:da:ba/10.0/8
Sending on   LPF/eth0/00:e0:29:29:da:ba/10.0/8
Sending on   Socket/fallback/fallback-net
DHCPREQUEST for 10.1.1.99 from 00:02:3f:15:34:bb via eth0:
lease 10.1.1.99 unavailable.
DHCPNAK on 10.1.1.99 to 00:02:3f:15:34:bb via eth0
DHCPDISCOVER from 00:02:3f:15:34:bb via eth0
DHCPOFFER on 10.1.1.100 to 00:02:3f:15:34:bb via eth0
DHCPREQUEST for 10.1.1.100 (10.1.1.1) from 00:02:3f:15:34:bb
via eth0
DHCPACK on 10.1.1.100 to 00:02:3f:15:34:bb via eth0
```

Figyeljük meg az eseményeket! Az ügyfél első lépésként a DHCPREQUEST (request, kérés) üzenettel kérte a jól megszokott IP címét, de azt a kiszolgáló a DHCPNAK (not acknowledged, nincs nyugtázva) üzenettel visszautasította. Ez után az ügyfél a DHCPDISCOVER (discover, felderít) üzenettel tetszőleges IP cím után puhatolózott, amelyre válaszul felajánlotta számára a kiszolgáló az új címet a DHCPOFFER (offer, ajánlat) üzenetben. Most az ügyfél kimondottan kérte az új IP címet a DHCPREQUEST üzenettel, amelyet a kiszolgáló a DHCPACK (acknowledge, nyugtáz) üzenettel nyugtázott.

A munkaállomás tehát az új, számára lefoglalt IP címet kapta.

5.5.4. A DHCP kiszolgáló beállításai

A DHCP kiszolgáló beállítására a bemutatott kifejezésekben kívül sok más kifejezést is használhatunk a /etc/dhcpcd.conf állományban, melyek közül csak néhányat sorolunk fel a következő listában:

max-lease-time szám; A kifejezés segítségével megadhatjuk, hogy a munkaállomások legfeljebb hány másodpercig kaphatják meg az IP címet.

Ha az idő lejár, a munkaállomások természetesen újra kérhetik a hálózati adatokat, és akár az előzőleg kapott IP címet is újra megkaphatják.

`default-lease-time szám;` A kifejezés segítségével megadhatjuk, hogy alapér-telmezés szerint hány másodpercre kapják meg az IP címet a munkaállomások.

`filename "állománynév";` A kifejezés segítségével megadhatjuk annak az állománynak a nevét, amelyet a munkaállomásnak le kell töltenie és el kell indítania a rendszerindításkor.

Ezt a kifejezést csak akkor használjuk, ha a munkaállomás rendszerindítását a hálózaton keresztül letöltött állománnyal szeretnénk biztosítani.

`next-server cím;` A kifejezés segítségével megadhatjuk, hogy a rendszerindításkor letöltendő állomány melyik számítógépről származzon.

A kifejezésben a kiszolgáló IP címe vagy neve is szerepelhet.

`ping-check true | false;` A kiosztandó IP cím ellenőrzésének engedélyezése és tiltása.

Mielőtt a kiválasztott IP címet elküldené a munkaállomásnak a DHCP kiszolgáló, megkíséri elérni a kiválasztott IP címen esetleg üzemelő munkaállomást. Ha a kiszolgáló úgy találja, hogy az adott IP címen már üzemel egy munkaállomás, a címet nem küldi el. (Valójában ilyen esetekben semmit nem küld.)

Amikor a kiválasztott címet ellenőrzi, a DHCP kiszolgáló egy másodpercet vár a válaszra. Ez azt eredményezi, hogy a DHCP ügyfél legalább egy másodpercet kell, hogy várjon a válaszra.

A kiválasztott IP cím ellenőrzését kikapcsolhatjuk ennek a kifejezésnek a segítségével. Ha a kifejezésben a `false` kulcsszó szerepel, a DHCP kiszolgáló nem ellenőrzi a kiválasztott IP címet.

5.5.5. Egyéb hálózati beállítások szolgáltatása

A DHCP kiszolgáló beállítóállományában a hálózat nagyon sokféle beállítását megadhatjuk, a hálózatot kifinomult módon behangolhatjuk. A `/etc/dhcpd.conf` állományban elhelyezhető ilyen kifejezések rögzítése több leírást kaphatunk a `man dhcp-options` parancs segítségével.

A következő lista a beállításhoz használt legfontosabb kifejezéseket tartalmazza:

`option arp-cache-timeout szám;` A kifejezés segítségével megadhatjuk, hogy az `arp` gyorsítótárban mennyi ideig maradjanak meg az IP címekhez tartozó Ethernet címek.

`option bootfile-name "állománynév";` A kifejezés segítségével megadhatjuk, hogy a DHCP ügyfélnek melyik állományt kell letöltenie és elindítani a rendszerindításkor.

`option broadcast-address cím;` A kifejezéssel megadhatjuk a körüzenetek küldésére használt címet.

`option default-ip-ttl szám;` A kifejezés segítségével megadhatjuk, hogy a ki-menő IP csomagokat a DHCP ügyfél milyen TTL (*time to live*, hátralévő élet-tartam) értékkel küldje el.

A csomagokhoz tartozó TTL értéket minden útválasztó csökkenti, és ha eléri a 0-t, a csomagot eldobja. Így garantálható, hogy az esetleges hurkokban nem keringenek az örökkévalóságig a csomagok.

`option font-servers cím1, cím2;` A kifejezés segítségével megadhatjuk egy vagy több betűkiszolgáló címét, amelyeket a grafikus felület használ betűtípusok letöltésére.

`option host-name "név";` A kifejezés segítségével nevet rendelhetünk a DHCP ügyfélként jelentkező munkaállomáshoz.

A kifejezésben szereplő név tartalmazhat tartománynevet is, de azt inkább az `option domain-name` kifejezéssel szerencsés megadni.

`option ieee802-3-encapsulation "true | false";` A kifejezéssel beállíthatjuk, hogy az ügyfél Ethernet II. (`false`) vagy Ethernet 802.3 (`true`) szabványú Ethernet csomagokat használjon.

`option interface-mtu szám;` A kifejezés segítségével beállíthatjuk, hogy a DHCP ügyfél számára mekkora legyen a legnagyobb hálózatra helyezhető csomag mérete bájtban.

`option ip-forwarding "true | false";` A kapcsoló segítségével beállíthatjuk, hogy a DHCP ügyfél az IP csomagokat továbbítsa-e. Ha az adott munkaállomást útválasztóként szeretnénk használni, a csomagokat továbbítania kell, és így a kifejezésbe a `true` kulcsszót kell beírnunk.

`option lpr servers cím1, cím2;` A kifejezés segítségével egy vagy több nyomtatókiszolgáló címét adhatjuk át a DHCP ügyfélnek.

`option nds context "környezet";` A kifejezés segítségével a Novell Netware ügyfélprogramok számára adhatjuk meg az alapértelmezett NDS (*netware directory service*) környezet értékét, amelyet a felhasználó a belépéskor meg-változtathat.

`option netbios name servers cím1, cím2;` A kifejezés segítségével az ügyfél által használandó NetBIOS névkiszolgáló (WINS kiszolgáló) vagy kiszolgálók címét adhatjuk meg.

`option netbios node type szám;` A kifejezés segítségével megadhatjuk, hogy az DHCP ügyfél milyen módon végezze a NetBIOS nevek felderítését. A kifejezésben található szám a következő értékeket veheti fel:

- 1 A nevek kiderítése körüzenetek segítségével.
- 2 A nevek kiderítése WINS kiszolgáló segítségével.
- 4 Az ügyfél előbb körüzenetekkel próbálkozik, majd, ha az sikertelen, a WINS kiszolgálóval.
- 8 Az ügyfél előbb a WINS kiszolgálóval próbálkozik, majd, ha az sikertelen, körüzenettel.

A kifejezés roppant hasznos, mert lehetővé teszi, hogy a NetBIOS kapcsolatot használó munkaállomásokat lebeszéljük arról, hogy az örökös körüzenetekkel pazarolják az erőforrásokat.

option nis domain "tartománynév"; A kifejezés segítségével beállíthatjuk a munkaállomás által használt NIS tartománynevet.

option nis servers cím1, cím2; A kifejezés segítségével megadhatjuk, hogy a munkaállomás milyen címen keresse a NIS kiszolgálót vagy kiszolgálókat.

option nisplus domain "tartománynév"; A kifejezés segítségével megadhatjuk a munkaállomás által használt NIS+ tartománynevet.

option nisplus servers cím1, cím2; A kifejezés segítségével megadhatjuk, hogy a munkaállomás milyen címen keresse a NIS+ kiszolgálót vagy kiszolgálókat.

option nntp server cím1, cím2; A kifejezés segítségével megadhatjuk az internetes hírcsoportokat szolgáltató kiszolgáló vagy kiszolgálók címét.

option pop server cím1, cím2; A kifejezés segítségével megadhatjuk a munkaállomásnak a rendelkezésére álló POP3 kiszolgáló vagy kiszolgálók címét, ahonnan a felhasználók elektronikus leveleit lehetővé tehetjük.

option smtp server cím1, cím2; A kifejezés segítségével megadhatjuk a munkaállomásnak a rendelkezésére álló SMTP kiszolgáló vagy kiszolgálók címét, amelyek felé a felhasználók elektronikus leveleit elküldheti, ha nem képes közvetlenül továbbítani azokat.

option tftp server name "név"; A kifejezés segítségével megadhatjuk a DHCP ügyfélnek a TFTP kiszolgáló nevét. A legtöbb esetben a hálózaton keresztül történő rendszerindításra használjuk a TFTP kiszolgálót.

option time offset szám; A kifejezés segítségével megadhatjuk az ügyfél által használandó időzónát az UTC-től (*coordinated universal time*, központi általános idő) való eltérés megadásával.

option time servers cím1, cím2; A kifejezés segítségével megadhatjuk a DHCP ügyfélnek a rendelkezésére álló, pontos időt szolgáltató kiszolgáló vagy kiszolgálók címét.

`option x display manager cím1, cím2;` A kifejezés segítségével megadhatjuk annak a kiszolgálónak vagy azoknak a kiszolgálóknak a címét, amelyekhez az X Window System szabványú grafikus felület segítségével kapcsolódni lehet.

A bemutatott kifejezések különösen hasznosak lehetnek olyan számítógéphálózatok esetében, ahol sokféle operációs rendszer található.

6. fejezet

A hálózat lehallgatása

Amint láttuk, a számítógép-hálózat a számítógépek közti kapcsolattartást biztosító adatcsomagokat eljuttatja a küldő számítógéptől a címzett számítógéphez. Azt is láttuk, hogy valójában nem csak a címzett számítógép kapja meg az adatcsomagokat, hanem a hálózat más számítógépei is.

Az előző oldalakon megismerkedhettünk az Ethernet és az IP címzési rendszerrel, és megtudtuk azt, hogy az Ethernet hálózati kártyák csak azokat a csomagokat küldik az operációs rendszernek feldolgozásra, amelyeken a saját Ethernet címük szerepel címzettként, vagy körüzenetként mindenki számára fontosak. Az Ethernet kártya tehát gondoskodik arról, hogy a számítógépen futó operációs rendszer és az alkalmazások ne kapják meg azokat az adatcsomagokat, amelyek nem nekik szólnak.

A helyzet azonban nem ilyen egyszerű! A modern hálózati kártyák rendelkeznek egy lehallgató üzemmóddal (*promiscuous mode*, válogatás nélküli üzemmód), amely arra szolgál, hogy minden adatcsomagot megvizsgálhassunk. A hálózati kártya, amely lehallgató üzemmódban működik, minden adatcsomagot továbbít az operációs rendszernek, függetlenül attól, hogy az adatcsomagot kinek szánták.

Ha a hálózati kártyát lehallgató üzemmódba kapcsoljuk, a segítségével a teljes hálózati forgalmat megfigyelhetjük, lehallgathatjuk. A lehallgató üzemmódot így kitűnően használhatjuk hibakeresésre, a hálózat működésének megfigyelésére. Azonban nem csak hibakeresésre használható a lehallgató üzemmód, hanem a hálózaton forgalmazott adatokhoz való illetéktelen hozzáférésre is. GNU/Linux rendszereken természetesen csak a rendszergazda kapcsolhatja be a lehallgató üzemmódot, ha azonban a hálózat valamelyik számítógépén a behatoló rendszergazdai jogokhoz jut, a lehallgatást használva további számítógépek adatforgalmát figyelheti meg.

Nyilvánvaló, hogy a lehallgató üzemmód bekapsolása és a hálózat ilyen eszközzel való megfigyelése komoly biztonsági kérdéseket vet fel, ezért minden megfontoltan kell a lehallgatás eszközéhez nyúlnunk.



Mindig csak olyan hálózaton alkalmazzuk a lehallgatást, amely a fennhatóságunk alá tartozik, amelyet mi üzemeltetünk. Tudnunk kell, hogy léteznek olyan eszközök, amelyekkel a lehallgatás felderíthető, amelyekkel a lehallgató személyére fény derülhet!

Mindig hívjuk fel a felhasználók figyelmét arra, hogy a számítógép-hálózat üzemeltetése közben szükség esetén a lehallgatás eszközét is igénybe vesszük, és arra is, hogy az adataik védelmének érdekében titkosító rendszereket használhatnak!

Nem mondhatunk le ugyanis a lehallgatásról, mint a hibakeresés egyik legfontosabb eszközéről. A számítógép-hálózaton ugyanis csak úgy tudjuk megkeresni a legtöbb hibát, ha pontosan tudjuk, hogy mi történik, milyen adatcsomagok vannak a hálózaton, milyen okból akadt el az adatforgalom.

Meg kell még említenünk azt is, hogy a lehallgató üzemmód fokozott terhelést róhat a lehallgatásra használt számítógépre. Mivel a hálózati kártya válogatás nélkül küldi a beérkezett adatcsomagokat, a számítógépnek, az operációs rendszerek azokat a szokásosnál gyorsabban kell feldolgozni. A ma használt legtöbb számítógép már képes az ilyen nagy adatmennyiségek fogadására és feldolgozására, de ez nyilvánvalóan nem jelenti azt, hogy bármely számítógéppel bármely hálózatot le lehet hallgatni.

GNU/Linux rendszerekre sokféle lehallgatóprogram készült. Léteznek egyszerűbb, karakteres felületen vezérelhető, és bonyolultabb, több szolgáltatást nyújtó grafikus felülettel felszerelt lehallgatóprogramok is. A következő néhány oldalon az egyik legfejlettebb lehallgatóprogram használatát mutatjuk be, és kihasználjuk a lehetőséget arra, hogy a hálózati kapcsolattartás néhány finom részletére is felhívjuk a figyelmet. A hálózat lehallgatása közben ugyanis sokat tanulhatunk annak működéséről, felépítéséről¹.

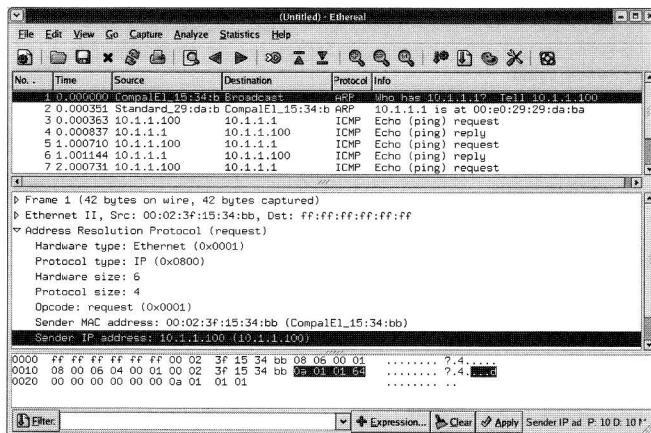
6.1. A program indítása

Az ethereal program segítségével grafikus felület segítségével hallgathatjuk le a hálózatot és elemezhetjük az eredményt. A program nagyon könnyen használható, igen sokat tanulhatunk a segítségével a hálózatok működéséről.

A 6.1. ábrán láthatjuk az ethereal főablakát a lehallgatás közben. Figyeljük meg az ablak egyes részeit!

A program főablakának felső részén a szokásos elemeket láthatjuk: legfelül a menüsor, közvetlen alatta pedig az eszköztár látható. Az ablak nagyobb részét a három sávra osztott adatterület foglalja el. A felső adatterület az elfogott csomagokat mutatja, minden csomagról a legfontosabb információkat ábrázolva egy-egy sorban. Az ablaknak ezen a részén kiválaszthatjuk az egyes csomagokat részletes tanulmányozás céljából.

¹Lehallgatás közben figyelembe kell vennünk az intelligensebb hálózati eszközöket, amelyek lahallgató módban is elfedhetik alólunk a forgalom egy részét (a lektor).



6.1. ábra. Az ethereal főablaka

Az ablak középső részén a kiválasztott csomag elemzését láthatjuk. Az ethereal igen sokféle hálózati szabványt, csomagfelépítést ismer, így a csomagban található információkat olvasható formában képes elérni tární a középső részen látható formában. A hálózatról lehallgatással megszerzett és elemzett csomagok böngészésével csodálatosan hatékony eszközt kapunk a hálózat vizsgálatához és a hálózat működésének megértéséhez.

A képernyő alsó részében a csomagot eredeti formájában, adathalmazként látjuk. Az alsó részen a program kiemeli azokat a bájtokat, amelyeket a középső részen kiválasztunk, így azonosíthatjuk az egyes részeket.

A főablak legalsó sorában a csomagszűrőt állíthatjuk be. A csomagszűrő nagy szolgálatot tesz zsúfolt hálózat vizsgálata közben, hiszen szűrés nélkül esélyünk sem volna a sok ezer csomag közül megtalálni azt, amelyik éppen érdekel minket.

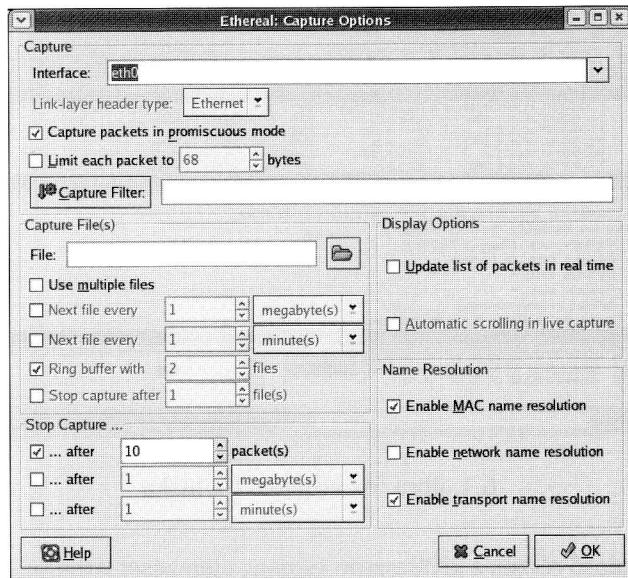
6.2. A lehallgatás

A hálózat vizsgálata általában a hálózat lehallgatásával kezdődik. A *capture* menü *start...* menüpontjával kezdeményezhetjük a hálózat lehallgatását. Ekkor az 6.2. ábrán látható ablak jelenik meg.

Az ablakban a legfontosabb elemek a következők:

interface A lenyiló listában beállíthatjuk, hogy melyik hálózati csatolón szeretnékn a hálózatot lehallgatni. Ha az *any* szót választjuk a listából, egyszerre hallgathatjuk le az összes hálózati csatolóra kapcsolódó hálózatot.

capture packets in promiscuous mode A jelölőnégyzettel bekapsolhatjuk a hálózati kártya lehallgató üzemmódját, hogy azokat a csomagokat is lássuk, amelyeket nem a mi számítógépünknek küldtek. Ha nem kapcsoljuk be ezt a jelölőnégyzetet, csak a számítógép számára küldött csomagokat és a körüzenetként küldött csomagokat figyelhetjük meg.



6.2. ábra. A lehallgatás kezdete

limit each packet A kapcsoló és a mellette található beviteli mező segítségével korlátozhatjuk a lehallgatás során tárolt csomagok méretét.

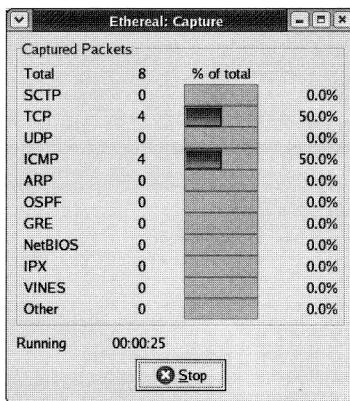
Mivel a legfontosabb információk a csomagok elején találhatók, ezzel a lehetséggel töredékére csökkenthetjük a lehallgatás során keletkező adatokat, anélkül, hogy a hibakereséshez fontos információk elvesznének.

capture filter Itt állíthatjuk be a lehallgatás során használt szűrőt. A beviteli mezőbe beírhatjuk a szűrésre használt parancsot, vagy kiválaszthatjuk a nyomógombbal az előre létrehozott és tárolt szűrőket. A szűrők használatára a későbbiekben visszatérünk.

capture file(s) Az itt található eszközök segítségével a lehallgatás eredményét állományba vagy állományokba menthetjük. Ha a lehallgatás eredményeképpen kapott adatokat több állományba osztjuk szét, a keletkező állományok nem lesznek kezelhetetlenül nagyok.

stop capture Ezekkel az eszközökkel beállíthatjuk, hogy mikor álljon le a lehallgatás. Ha egyik jelölőnégyzetet sem kapcsoljuk be, a lehallgatást nekünk kell leállítanunk egy nyomógomb lenyomásával.

display options Ezekkel a jelölőnégyzetekkel beállíthatjuk, mi történjen a képernyőn a lehallgatás közben. Beállíthatjuk, hogy folyamatosan jelenjenek-e meg a csomagok a képernyőn, és a lista kövesse-e az újabb csomagokat.



6.3. ábra. A lehallgatás állása

A csomagok lehallgatás közben történő kiértékelése gyors számítógépet igényel.

name resolution A jelölőnégyzetekkel beállíthatjuk, hogy milyen névfeloldási eszközöket használjon a program. A névfeloldás segítségével elérhetjük, hogy a hálózati adatok ne számokkal, hanem nevekkel jelenjenek meg.

Ha beállítottuk, hogy milyen módon szeretnénk lehallgatni a hálózatot, az OK gomb segítségével elindíthatjuk a lehallgatást.

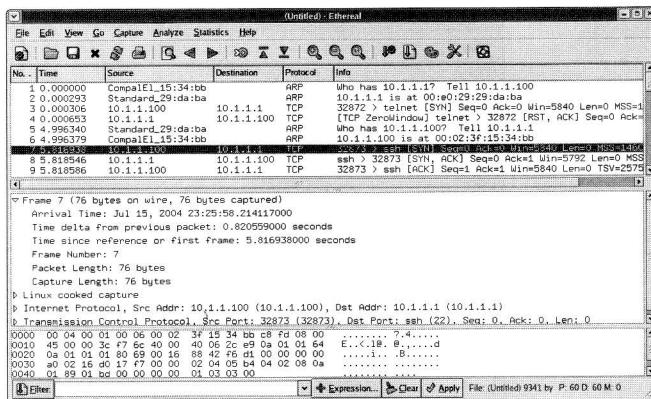
A lehallgatás közben a 6.3. ábrán látható ablakon figyelhetjük meg, hogy hány csomagot sikerült begyűjtenünk. Az ablakon látható nyomógombbal bármikor megszakíthatjuk a hálózat lehallgatását, és elkezdhetjük az adatok elemzését.

6.3. Az adatok elemzése

Miután befejeztük a hálózat lehallgatását, elemezhetjük az elfogott csomagokat, amelyeket a program a főablakban megjelenít (6.4. ábra). Az ablak felső részén az elfogott csomagok listáját láthatjuk. A listában a csomagokról megjelenő oszlopok átrendezhetők a program beállításainak megváltoztatásával. Alapesetben a következő oszlopokat látjuk:

no. A csomag sorszáma. A program a lehallgatás kezdetétől fogva megszámozza az elfogott csomagokat, és a sorszámot jeleníti meg az első oszloban.

time A csomag beérkezésének időpontja. Ez az oszlop fontos lehet, ha tudni szeretnénk, hogy az egyes kérésekre a számítógépek mennyi idő elteltével voltak képesek válaszolni.



4. ábra. A lehallgatott adatok

source A csomagot küldő hálózati eszköz címe. Itt általában az eszköz IP címe (vagy az IP címhez tartozó név) áll; ha ezt nem lehet kideríteni, a küldő eszköz Ethernet címe jelenik meg.

destination A címzett hálózati eszköz címe (IP vagy Ethernet cím, esetleg név).

protocol A csomag alapprotokollja (TCP, UDP stb.).

info A csomag rövid leírása, amelyből az avatott szem igen sok információt ki tud olvasni.

Az elfogott csomagok legfontosabb adatait tartalmazó listából az egér segítségével bármelyik csomagot kiválaszthatjuk, hogy részletesen megvizsgáljuk. Ha a listaelemre kattintunk, az adott csomag részletes tartalma megjelenik a lista alatti két nagyobb mezőben. A középső mezőben a csomag tartalma „emberi fogyasztásra alkalmas” változatban olvasható, az alsó mezőben pedig eredeti formájában, tizenhatos számrendszerben ábrázolva. Vizsgáljuk meg jobban a középső mezőt!

A középső mező faszerkezetben ábrázolja a csomagot. Az egyes ágakat a sorok elején található háromszögre kattintva kinyithatjuk és becsukhatjuk. A kinyitott ágakon belül újabb ágakat, azokon belül pedig újabb ágakat találunk a csomag bonyolultságának megfelelő mélységgel. Ha az egyes ágakat az egérrel kijelöljük, az ablak alsó részében megfigyelhetjük, hogy az ág által képviselt információt az *ethereal* a csomag mely részéből olvasta ki.

A csomag középső mezőben található ábrázolása nagyon szemléletesen mutatja be a hálózati kapcsolattartásra használt adatcsomagok felépítését és kezelését. Ahogyan felülről lefelé, az ágakat kinyitva elolvassuk a csomag tartalmát, megérjük, hogy a számítógép operációs rendszere miképpen értelmezi a hálózati csomagokat.

6.1. tábla: ping

A hálózati kapcsolat ellenőrzése.

ping [kapcsolók] címzett

A ping program a hálózati kapcsolatot ellenőrzi oly módon, hogy csomagokat küld a címzettnek, az pedig válaszcsomagokat küld, jelezve, hogy a kapcsolat adatátvitelre használható állapotban van.

Kapcsoló Jelentés

- | | |
|----------|--|
| -A | Amint a válasz megérkezett, a program újabb csomagot küld.
A rendszergazdai jogokkal nem rendelkező felhasználók számára a legkisebb ismétlődési idő 200ms. |
| -c szám | Adott számú csomag küldése után a program automatikusan leáll. |
| -f | Csomagok folyamatos küldése (csak a rendszergazda számára). |
| -n | A számítógépnevek lekérdezésének tiltása. |
| -s méret | A küldött csomagok mérete bájtban. |
| -w szám | A program a másodpercben megadott idő letelte után automatikusan leáll. |

6.4. A kapcsolattartás módjai

A következő oldalakon néhány példával illusztrálva mutatjuk be a hálózati kapcsolattartás módjait, legfontosabb elemeit. A példák segítségével lehetőségünk nyílik arra, hogy kissé részletesebben elmélyedjünk a számítógép-hálózatok működésének rejtelmeiben, ami mindenképpen hasznos a hibakeresés szempontjából.

6.4.1. A csomagküldés előkészítése

Vizsgáljuk meg, hogy mi történik, amikor egy számítógép felveszi a kapcsolatot egy számítógéppel, amellyel a számítógép-hálózat közvetlenül összeköti! Használjuk a kapcsolatfelvételre a ping parancsot, amely alapértelmezés szerint másodpercenként egy csomagot küld az ellenállomásnak, és kiírja a szabványos kiemenetre, hogy mennyi idő alatt kapta meg a csomagra a választ!

A következő sorok az ethereal főablakának felső részéből származnak (néhány oszlopot az egyszerűség kedvéért eltávolítottunk):

No.	Source	Destination	Protocol	Info
1	CompaEL_15:34:bb	Broadcast	ARP	Who has 10.1.1.98? Tell 10.1.1.100
2	SunMicro_1d:a4:48	CompaEL_15:34:bb	ARP	10.1.1.98 is at 00:03:ba:1d:a4:48
3	10.1.1.100	10.1.1.98	ICMP	Echo (ping) request
4	10.1.1.98	10.1.1.100	ICMP	Echo (ping) reply
5	10.1.1.100	10.1.1.98	ICMP	Echo (ping) request
6	10.1.1.98	10.1.1.100	ICMP	Echo (ping) reply

A táblázatban látható adatcsomagok szerepe a következő:

1. Az első csomagot az a számítógép küldi, amelyik a hálózati kapcsolat kiépítését kezdeményezte. A csomag célja az ellenállomás Ethernet címének kiderítése.

Láttuk, hogy a hálózati kártyák csak azokat a hálózati csomagokat továbbítják az operációs rendszerek, amelyen címeztek a saját Ethernet címük van, vagy körüzenetként minden számítógépnek szólnak. A küldő számítógépnek tehát ki kell derítenie, hogy a címzett IP címéhez (itt 10.1.1.98) milyen Ethernet cím tartozik.

A kapcsolatfelvételt kiépítő számítógép az első csomagban körüzenetként (a címzett helyén a Broadcast szó olvasható) egy kérdést intéz az összes jelenlévő számítógéphez, amelyben megtudakolja, hogy melyik Ethernet címhez tartozik a kérdéses (itt 10.1.1.98) IP cím.

2. A körüzenetben feltett kérdést minden számítógép feldolgozza, és benne az egyik magára ismer. Ez a számítógép a második csomagban válaszol, jelezve, hogy a keresett IP cím (10.1.1.98) melyik Ethernet címen érhető el (00:03:ba:1d:a4:48)

A második csomag feladójának címéből láthajuk, hogy az a számítógép válaszol, amelyik magára ismer az IP cím alapján (kissé ugyan másképpen jelenik meg az Ethernet cím az első oszlopban, de ugyanarról a címről van szó). Ez nem feltétlenül van minden így, de az egyszerűbb hálózatok esetében minden számítógép a saját Ethernet címét hirdeti, a saját Ethernet címéért felelős.

3. A harmadik csomagot szintén a kapcsolatkiépítést kezdeményező számítógép küldi, amely most már pontosan ismeri a címzett Ethernet címét. (A harmadik sorban nem látszik az Ethernet cím, de a csomagban nyilvánvalóan benne van, csak helytakarékkosság miatt nem jelenik meg.)

A harmadik csomag az IP címzésre épülő ICMP alapprotokollú, kapcsolatellenőrzésre szolgáló válaszkérő csomag (*echo request*, visszhangkérés). A ping program ilyen csomagok küldésére használatos.

4. A negyedik csomagot a harmadik csomag címzettje küldi válaszul a kapcsolatot ellenőrző harmadik csomagra. A csomag IP címzésre épülő ICMP alapprotokollú kapcsolatellenőrző válaszcsomag (*ping reply*, visszhangválasz).

Ezt a csomagot az ellenállomás operációs rendszere küldi, automatikusan válaszolva a kérésre.

5. A további csomagok a ping program által másodpercenként küldött kéréseket és a rájuk adott válaszokat tartalmazzák.

Ha a helyi hálózaton valamelyik számítógép IP címzésén alapuló csomagot akar küldeni egy másik számítógépnek, előbb az ARP (*address resolution protocol*, cím-feloldó protokoll) protokoll szerinti kéréssel kideríti az ellenállomás Ethernet címét. Az Ethernet ARP csomagok formáját és értelmezését természetesen szabvány rögzíti[107].

Ha néhány percen keresztül kísérletezünk a hálózat lehallgatásával, hamar észrevesszük, hogy a kapcsolat kiépítése nem minden jár együtt ARP csomagok különbsével és azok megválasztásával. Ez azért van így, mert a hálózatba kötött számítógépek operációs rendszerei – így természetesen a Linux rendszermag is – tárolják a lekérdezett Ethernet címeket és a hozzájuk tartozó IP címeket az ARP gyorsítótárban. Ha az operációs rendszer az ellenállomás Ethernet címét az ARP gyorsítótárban megtalálja, nyilvánvalónak szükségtelen azt körüzenettel kideríteni.

A hálózatról lekérdezett és az ARP gyorsítótárban elhelyezett bejegyzések néhány perc után elévülnek. A címet ilyenkor egy körüzenet segítségével újra le kell kérdezni. Az ARP bejegyzések elévülése nyilvánvalónak komoly szerepet kap, ha valamelyik számítógépben kicseréljük a hálózati kártyát, hiszen ilyenkor a hálózat többi számítógépének „észre kell vennie”, hogy az adott IP cím már egy új Ethernet címen érhető el.

A Linux rendszermag által kezelt ARP táblázat lekérdezésére és megváltoztatására az arp program használható.

Vizsgáljuk most meg, mi történik, ha az ellenállomás nem érhető el! A következő néhány csomag egy meghiúsult kapcsolatfelvétel során keletkezett:

No.	Source	Destination	Protocol	Info
1	CompalEl 15:34:bb	Broadcast	ARP	Who has 10.1.1.12? Tell 10.1.1.100
2	CompalEl 15:34:bb	Broadcast	ARP	Who has 10.1.1.12? Tell 10.1.1.100
3	CompalEl 15:34:bb	Broadcast	ARP	Who has 10.1.1.12? Tell 10.1.1.100
4	CompalEl 15:34:bb	Broadcast	ARP	Who has 10.1.1.12? Tell 10.1.1.100
5	CompalEl 15:34:bb	Broadcast	ARP	Who has 10.1.1.12? Tell 10.1.1.100

Az eddigi ismeretek alapján már könnyen megérthetjük, mi történt. Egyszerűen arról van szó, hogy a 10.1.1.100 IP című eszköz keresi a 10.1.1.12 című eszközt, de úgy tűnik, ebben a címben egyetlen operációs rendszer sem ismer magára. Nyoma sincs a hálózaton 10.1.1.12 IP című eszköznek.



Ha ilyen hibát észlelünk, és már minden ötletből kifogytunk, érdemes lehallgatást kezdeni a címzett számítógépen. Meg kell néznünk, hogy megkapja-e a csomagokat, magára ismer-e, és válaszol-e rájuk. Ha ezekre a kérdésekre megtaláltuk a választ, a hibát is megtaláltuk.

Ha viszont az ellenállomás megkapja az ARP csomagokat, és válaszol is rájuk, de a válasz nem jut el a kérdezőhöz, a csomagok csak egyik irányba jutnak át a hálózaton. Ez első pillantásra lehetetlennek tűnik, de többféleképpen is elképzelhető. Hibás lehet a hálózati kábel, valamelyik hálózati kártya és a meghajtóprogram beállításai (IRQ) is.

Ha a Linux rendszermag nem kap választ az ARP kérdésre, egy darabig szemérmesesen hallgat a kudarcról a kapcsolatiépítést kérő alkalmazás előtt. Újra és újra megpróbálkozik a kérés elküldésével, de rövidesen feladja, és jelzi a hibát az alkalmazásnak. Az alkalmazás ilyenkor eldöntheti, hogy újra kéri a kapcsolatot vagy feladja és jelzi a felhasználónak a hibát. A ping például soha nem adja fel a próbálkozást. Jelzi ugyan a hibát a felhasználónak, de folyamatosan próbálkozik, hogy a kapcsolat helyreállását észrevehessük.

A következő sorok egy ARP csomagot mutatnak be olyan formában, ahogyan azt az ethereal segítségével megjeleníthetjük:

```

1 Frame 1 (42 bytes on wire, 42 bytes captured)
2 Arrival Time: Jul 16, 2004 00:08:07.900810000
3 Time delta from previous packet: 0.000000000 seconds
4 Time since reference or first frame: 0.000000000 seconds
5 Frame Number: 1
6 Packet Length: 42 bytes
7 Capture Length: 42 bytes
8 Ethernet II, Src: 00:02:3f:15:34:bb, Dst: ff:ff:ff:ff:ff:ff
9 Destination: ff:ff:ff:ff:ff:ff (Broadcast)
10 Source: 00:02:3f:15:34:bb (CompalEl_15:34:bb)
11 Type: ARP (0x0806)
12 Address Resolution Protocol (request)
13 Hardware type: Ethernet (0x0001)
14 Protocol type: IP (0x0800)
15 Hardware size: 6
16 Protocol size: 4
17 Opcode: request (0x0001)
18 Sender MAC address: 00:02:3f:15:34:bb (CompalEl_15:34:bb)
19 Sender IP address: 10.1.1.100 (10.1.1.100)
20 Target MAC address: 00:00:00:00:00:00 (00:00:00_00:00:00)
21 Target IP address: 10.1.1.1 (10.1.1.1)
```

Az 1–7. sorok az ethereal által szolgáltatott információk, amelyek a csomag beérkezésének körülményeit írják le. Az 1. sor tanúsága szerint ez a csomag az első lehallgatott csomag, amely 42 bájtot tartalmaz és 42 bájtot sikerült is lehallgatni. A 2–4. sorok a csomag beérkezésének időpontját adják meg. Az 5–7. sorok az első sorban is olvasható információkat ismétlik meg.

A 8–11. sorokban az Ethernet szabvány segítségével átadott információkat olvashatjuk. A 8. sorban láthatjuk, hogy Ethernet II. szabványú csomagról van szó, a 9. sorban olvashatjuk a címzett Ethernet címét (körüzenet), a 10. sorban pedig a küldő Ethernet címét.

A 12. sorban láthatjuk, hogy ez az Ethernet csomag az ARP szabvány szerint felépített adatokat hordoz. A 19. sorban olvashatjuk, hogy mi a küldő hálózati eszköz IP címe, a 21. sorban pedig azt, hogy milyen IP című gépet keres a feladó ezzel a csomaggal. A címzett Ethernet címe a 20. sorban természetesen üres, hiszen éppen ennek a címnek a kiderítésére született a csomag.

Figyeljük meg, milyen jól érzékeltetik a bemutatott sorok – és főképpen a ethereal főablaka – a csomag felépítését! Láthatjuk, hogy az ARP szabvány által felépített csomag egy Ethernet csomagban helyezkedik el, az ARP csomag az Ethernet csomag szempontjából egyszerűen adatnak minősül.

A következő sorok az előző példában bemutatott ARP kérésre adott választ mutatják be. Figyeljük meg, hogy ebben a mintában az első két szakaszt nem nyitottuk ki, a csomag beérkezésének körülményeiről és az Ethernet II. csomaginformációkról nem ábrázoltuk a részleteket.

```

1 Frame 2 (60 bytes on wire, 60 bytes captured)
2 Ethernet II, Src: 00:e0:29:29:da:ba, Dst: 00:02:3f:15:34:bb
3 Address Resolution Protocol (reply)
4   Hardware type: Ethernet (0x0001)
5   Protocol type: IP (0x0800)
6   Hardware size: 6
7   Protocol size: 4
8   Opcode: reply (0x0002)
9   Sender MAC address: 00:e0:29:29:da:ba (Standard_29:da:ba)
10  Sender IP address: 10.1.1.1 (10.1.1.1)
11  Target MAC address: 00:02:3f:15:34:bb (CompaEl_15:34:bb)
12  Target IP address: 10.1.1.100 (10.1.1.100)

```

A példa 2. sorában láthatjuk, hogy ez a csomag az előző csomaggal ellentétben nem körüzenet, az ARP válasz kimondottan az ARP kérést küldő hálózati eszköznek szól. A válaszcsomag 8. sorában szerepel a keresett IP című (10. sor) számítógép Ethernet címe (9. sor).

6.4.2. Csomag küldése

Vizsgáljuk meg, mi történik a hálózaton, amikor egy számítógép nevét kérdezzük le! A következő sorok az ethereal főablakának felső részéből származnak, és a névfeloldás lépései mutatják be. Itt is csak a legfontosabb oszlopokat hagytuk meg.

No.	Source	Destination	Protocol	Info
1	CompaEl_15:34:bb	Broadcast	ARP	Who has 10.1.1.1? Tell 10.1.1.100
2	gep.lkk	CompaEl_15:34:bb	ARP	10.1.1.1 is at 00:e0:29:29:da:ba
3	10.1.1.100	10.1.1.1	DNS	Standard query A sun.lkk
4	10.1.1.1	10.1.1.100	DNS	Standard query response A 10.1.1.98

A táblázatban bemutatott négy adatcsomag egy DNS névkiszolgálóhoz intézett kérdést és az arra adott választ tartalmazza. A kérdést a 10.1.1.100 IP című hálózati eszköz küldte a 10.1.1.1 IP című DNS kiszolgálónak, amely a kérdést azonnal meg is válaszolta. Az egyes csomagok szerepe a következő:

1. Az első csomag egy ARP csomag, amely a 10.1.1.1 IP című DNS kiszolgáló Ethernet címét tudakolja meg a hálózaton.
2. A második csomag az ARP kérdésre adott válasz. A csomaggal a 10.1.1.1 IP című hálózati eszköz válaszként jelzi a saját Ethernet címét.
3. A harmadik csomag a DNS kérdést tartalmazza, amellyel a 10.1.1.100 IP című DNS ügyfél kérdezi meg a sun.lkk nevű számítógép IP címét a DNS kiszolgálótól, melynek címe 10.1.1.1.

Figyeljük meg, hogy a kérdésben szerepel a keresett DNS bejegyzés típusa is! A DNS kiszolgáló kimondottan A típusú bejegyzést keres a kiszolgálón.

4. A negyedik csomag a DNS kiszolgáló által adott választ tartalmazza, amely szerint a keresett IP cím a 10.1.1.98.

A táblázatban a 3–4. csomag sorában a protokoll helyén a DNS rövidítést olvashatjuk, ami nyilvánvalóan azt jelenti, hogy ezek a csomagok a DNS protokoll szerint épülnek fel. A csomagok részletes vizsgálata során látni fogjuk, hogy a lekérdezésre és a válaszadásra használt csomagok az UDP alapprotokoll szerint épülnek fel, azaz a DNS ügyfél és a DNS kiszolgáló egyaránt olyan UDP protokollú IP címzési rendszerre épülő Ethernet csomagokat használ, amelynek tartalma a DNS protokoll szerint épül fel.

A következő sorok a DNS kérés közvetítésére használt csomag szerkezetét mutatják be. Vizsgáljuk meg e sorokat figyelmesen!

```

1 Frame 3 (67 bytes on wire, 67 bytes captured)
2 Ethernet II, Src: 00:02:3f:15:34:bb, Dst: 00:e0:29:29:da:ba
3   Destination: 00:e0:29:29:da:ba (gep.lkk)
4   Source: 00:02:3f:15:34:bb (CompalEl_15:34:bb)
5   Type: IP (0x0800)
6 Internet Protocol, Src Addr: 10.1.1.100, Dst Addr: 10.1.1.1
7   Version: 4
8   Header length: 20 bytes
9   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
10  Total Length: 53
11  Identification: 0x0000 (0)
12  Flags: 0x04 (Don't Fragment)
13  Fragment offset: 0
14  Time to live: 64
15  Protocol: UDP (0x11)
16  Header checksum: 0x2452 (correct)
17  Source: 10.1.1.100 (10.1.1.100)
18  Destination: 10.1.1.1 (10.1.1.1)
19 User Datagram Protocol, Src Port: 32769 (32769), Dst Port: 53 (53)
20  Source port: 32769 (32769)
21  Destination port: 53 (53)
22  Length: 33
23  Checksum: 0x91f8 (correct)
24 Domain Name System (query)
25  Transaction ID: 0xec5c
26  Flags: 0x0100 (Standard query)
27  Questions: 1
28  Answer RRs: 0
29  Authority RRs: 0
30  Additional RRs: 0
31  Queries
32    sun.lkk: type A, class inet
33      Name: sun.lkk
34      Type: Host address
35      Class: inet

```

Az 1. sorban olvashatjuk az ethereal által a csomag lehallgatásának körülmenyeiről adott kiegészítő információkat. Ezt a szakaszt most nem ábrázoltuk részletesen.

A 2. sorban olvashatjuk az Ethernet szabvány szerint felépített csomaginformációkat, azaz azokat a sorokat, amelyeket az Ethernet eszközök az Ethernet szabvány szerint kezelnek. A 3. sorban olvashatjuk a címzett Ethernet címét, a 4.

sorban a küldő Ethernet címét, az 5. sor tanúsága szerint pedig egy IP típusú csomagról, az IP címek rendszerére épülő csomagról van szó. Az Ethernet szabványú csomag adatként tehát egy IP csomagot tartalmaz.

A 6. sortól kezdődően olvashatjuk az adatcsomagnak az IP címzési rendszeren alapuló adatait (valójában a 6. sort kissé lerövidítettük, hogy kiférjen). Az IP címzési rendszeren alapuló adatokat részletesen ábrázoltuk, a 7–18. sorok ezeket mutatják be.

A 7. sorban azt olvashatjuk, hogy a csomag változatának száma 4., ami azt jelenti, hogy ez az IP csomag az IPv4 protokoll szerint épül fel. A 8. sorban azt olvashatjuk, hogy a csomag IP protokoll szerint felépülő adatokat tartalmazó fejléce 20 bájt hosszú, a 10. sorban pedig azt, hogy ez az IP protokoll szerint felépített csomag 53 bájt hosszú. A méretet rövid fejszámolással ellenőrizhetjük: az 53 bájtos IP csomag egy Ethernet csomagban található. Az Ethernet csomag az IP csomagot megnöveli a küldő Ethernet címével (6 bájt), a címzett Ethernet címével (6 bájt) és a típussal (2 bájt): mivel $53 + 6 + 6 + 2 = 67$, megkaptuk a csomag teljes méretét.

A 12. sorban azt olvashatjuk, hogy a küldő előírja, a csomagot nem szabad darabokra vágni (*don't fragment*, ne darabold). Az IP protokoll szerint akkor szükséges feldarabolni egy adatcsomagot részekre, ha útja során olyan hálózatra érkezik, amelyen a használható legnagyobb csomagmérőt kisebb, mint a csomag mérete. A darabokra vágott csomag helyreállítása és kezelése meglehetősen bonyolult, és biztonsági problémákat is felvet. Nyilvánvalóan ez is közrejátszott abban, hogy a csomagot küldő kifejezetten előírta, hogy a csomagot nem szabad darabolni.



Felmerül a kérdés, hogy mi történik, ha egy olyan csomag, amelyet nem szabad feldarabolni, olyan hálózatra érkezik, amelyen a használható legnagyobb csomagmérőt kisebb, mint a csomag mérete. A csomagot továbbküldeni nem lehet, mert túl nagy, darabolni nem lehet, mert a küldő megtiltotta, ezért a csomag nem továbbítható.

A továbbítással megbízott számítógép ilyenkor egy ICMP üzenetben visszajelez a csomag küldőjének, hogy legközelebb legyen szíves kisebb csomagokat küldeni, és egyszerűen eldobja a csomagot.

A 13. sorban olvashatjuk, hogy ez a csomag az eredeti csomag hányadik darabja. Mivel a csomagot nem is szabad darabolni, nyilvánvaló, hogy itt 0-t látunk. Ez a csomag egészen van!

A 14. sorban az IP csomagra vonatkozó TTL értéket olvashatjuk. Ezt a számlálót minden továbbító aktív eszköz eggel csökkenti, és ha bármelyik úgy találja, hogy a számláló elérte a 0-t, a csomagot eldobja. A csomagok előregszenek és meghalnak, ami biztosítja, hogy rosszul beállított hálózatban sem keringhetnek az örökkévalóságig.

6.2. tábla: traceroute

A csomagok útját nyomon követő program.

traceroute [kapcsolók] címzett

A traceroute program csomagokat küld a megadott számítógép felé, és kideríti, hogy a csomagok milyen úton érik el a címzettet. Kitűnően használható nagy méretű számítógép-hálózatok útvonalszerkezetének felderítésére.

Kapcsoló Jelentés

- i név A csomagok küldésére használt hálózati csatoló nevének megadása.
- w szám A válaszokra várakozás idejének korlátozása megadott másodpercre.



Nyilvánvaló, hogy az előregedett csomagot eldobó számítógép egy ICMP csomagban értesítheti a küldő számítógépet arról, hogy az általa küldött adatcsomag végelgyengülésben elpusztult. A küldő erre válaszul például növelheti az általa küldött csomagok TTL értékét.

Pontosan így működik a csomagok útját végigkövető traceroute program. A traceroute elküld egy csomagot a címzettnek a hétralévő életartamot 1-re állítva, majd megfigyeli, melyik számítógép küldi az üzenetet a csomag haláláról. Ezután a program küld egy csomagot 2-es TTL értékkal, majd 3-as értékkel és így tovább. minden egyes csomagra ICMP választ fog kapni, mégpedig attól a számítógéptől, amelyik a célállomáshoz kapcsolatot biztosító lánc következő eleme. Egyszerűen zseniális...

A csomagot ábrázoló 17. sorban olvashatjuk a küldő IP címét, míg a 18. sorban a címzett IP címe olvasható. Nyilvánvaló, hogy az IP szabvány szerint felépülő csomagrész legfontosabb adatai ezek.

A 15. sorban azt olvashatjuk, hogy ez az IP csomag adatként az UDP alapprotokoll szerint felépített csomagot hordoz, és valóban, a 19. sortól kezdve láthatók az UDP protokoll szerint ábrázolt adatok. A 20. sorban a forrásként használt hálózati kapu, a 21. sorban pedig a célként használt hálózati kapu számát olvashatjuk. Az UDP csomag az 53. hálózati kapura érkezik, amelyen a protokoll szerint a DNS kiszolgáló várja a kéréseket. Nyilvánvaló, hogy a csomag további részeit a DNS protokoll szerint kell értelmezniünk.

Valóban, az ethereal a 24–35. sorokban a DNS protokoll szerint ábrázolta az adatcsomag további részeit. A 27. sorban olvashatjuk, hogy ez a csomag pontosan egy kérdést tartalmaz, a 28–30. sorokban pedig arról értesülhetünk, hogy a csomag további információkat nem tartalmaz. A 32–35. sorokban magát a kérdést olvashatjuk, amelyet a csomag a DNS ügyfélétől a DNS kiszolgáló felé továb-

bított. A következő sorok a DNS kiszolgáló válaszát tartalmazó UDP csomagot ábrázolják.

```

1 Frame 4 (117 bytes on wire, 117 bytes captured)
2 Ethernet II, Src: 00:e0:29:29:da:ba, Dst: 00:02:3f:15:34:bb
3   Destination: 00:02:3f:15:34:bb (CompaqEl_15:34:bb)
4   Source: 00:e0:29:29:da:ba (gep.lkk)
5   Type: IP (0x0800)
6 Internet Protocol, Src Addr: 10.1.1.1, Dst Addr: 10.1.1.100
7     Version: 4
8     Header length: 20 bytes
9     Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
10    Total Length: 103
11    Identification: 0x0000 (0)
12    Flags: 0x04 (Don't Fragment)
13    Fragment offset: 0
14    Time to live: 64
15    Protocol: UDP (0x11)
16    Header checksum: 0x2420 (correct)
17    Source: 10.1.1.1 (10.1.1.1)
18    Destination: 10.1.1.100 (10.1.1.100)
19 User Datagram Protocol, Src Port: 53 (53), Dst Port: 32769 (32769)
20  Source port: 53 (53)
21  Destination port: 32769 (32769)
22  Length: 83
23  Checksum: 0xff04 (correct)
24 Domain Name System (response)
25   Transaction ID: 0xec5c
26   Flags: 0x8580 (Standard query response, No error)
27   Questions: 1
28   Answer RRs: 1
29   Authority RRs: 1
30   Additional RRs: 1
31   Queries
32     sun.lkk: type A, class inet
33       Name: sun.lkk
34       Type: Host address
35       Class: inet
36   Answers
37     sun.lkk: type A, class inet, addr 10.1.1.98
38       Name: sun.lkk
39       Type: Host address
40       Class: inet
41       Time to live: 3 hours
42       Data length: 4
43       Addr: 10.1.1.98
44   Authoritative nameservers
45     lkk: type NS, class inet, ns ns1.lkk
46       Name: lkk
47       Type: Authoritative name server
48       Class: inet
49       Time to live: 3 hours
50       Data length: 6
51       Name server: ns1.lkk
52   Additional records
53     ns1.lkk: type A, class inet, addr 10.1.1.1
54       Name: ns1.lkk
55       Type: Host address
56       Class: inet
57       Time to live: 3 hours
58       Data length: 4
59       Addr: 10.1.1.1

```

Figyeljük meg, hogy a 3–4. sorokban az előző csomaghoz képest megcserélőd-

tek a címzett és a küldő hálózati eszköz Ethernet címei. Ez jelzi, hogy a csomag válaszcsomagként az előző csomaggal ellentétes irányban haladt.

A 17–18. sorokban sorokban láthatók a küldő és címzett IP címei. Ezek is megcserélődtek, a küldő most a DNS kiszolgáló, a címzett a DNS ügyfél.

A 20–21. sorokban a küldő és a forrás hálózati kapu látható az UDP alapprototollnak megfelelő módon leírva. Az előző csomaghoz képest ezek is megcserélve szerepelnek a válaszcsomagban.

A 24–59. sorokban a DNS protokoll szerint ábrázolt válasz olvasható. A 27. sor tanúsága szerint a DNS kiszolgáló a válaszban megismételte a kérdést is. A kérdés a 31–35. sorban olvasható.

A 28. sorban arról olvashatunk, hogy a válaszcsomag egy választ tartalmaz. A választ a 36–43. sorokban olvashatjuk, amelyből kiderül a sun.1kk keresett IP címe. Figyeljük meg, hogy a választ tartalmazó sorok közül a 41. sorban a DNS bejegyzés hátralévő élettartama is szerepel. E szerint a sor szerint a sun.1kk IP címét 3 órán keresztül megőrizhetjük, de aztán újra meg kell kérdezniünk.

A 29. sor szerint a DNS kiszolgáló elküldte az ügyfélnek azt is, hogy a kérdéses tartománynak melyik DNS kiszolgáló a felelőse. A 30. sorban előre jelzett további DNS bejegyzés az 52–59. sorban olvasható, és a tartomány hivatalos DNS kiszolgálójának IP címét tartalmazza.

6.4.3. Folyamatos kapcsolat használata

Vizsgáljuk meg, hogy mi történik a hálózaton, amikor egy webböngésző letölt egy weblapot a webkiszolgálóról. A webböngésző és a webkiszolgáló közti adatátvitel nem korlátozódik egy-egy csomag küldésére, hiszen a weblapok többsége nem fér el egy hálózati csomagban, ezért az előzőeknél összetettebb viselkedésre számítunk. Nyilvánvaló, hogy a weblapok letöltésére az alkalmazások a TCP alapprototolált használják, hiszen ez az az IP alapú protokoll, amely nagyobb adatmennyiségek átvitelére is használható. Arra is számíthatunk, hogy a lehallgatás eredménye képpen összetettebb adatszerkezeteket kapunk, hiszen a TCP protokoll az eddig vizsgált protokolloknál bonyolultabb.

A következő csomagok a webböngésző és a webkiszolgáló közti kapcsolattartást mutatják be. Az itt olvasható 18 csomag lehallgatása közben a webböngészőt elindítottuk, egy weblapot letöltöttünk, és a webböngészőből kiléptünk.

No.	Source	Destination	Protocol	Info
1	CompaEl_15:34:bb	Broadcast	ARP	Who has 10.1.1.1? Tell 10.1.1.100
2	gep.lkk	CompaEl_15:34:bb	ARP	10.1.1.1 is at 00:e0:29:29:da:ba
3	10.1.1.100	10.1.1.1	DNS	Standard query AAAA sun.lkk
4	10.1.1.1	10.1.1.100	DNS	Standard query response
5	10.1.1.100	10.1.1.1	DNS	Standard query A sun.lkk
6	10.1.1.1	10.1.1.100	DNS	Standard query response A 10.1.1.98
7	CompaEl_15:34:bb	Broadcast	ARP	Who has 10.1.1.98? Tell 10.1.1.100
8	sun.lkk	CompaEl_15:34:bb	ARP	10.1.1.98 is at 00:03:ba:1d:a4:48
9	10.1.1.100	10.1.1.98	TCP	32824 > http [SYN] Seq=0 Ack=0 Win=5840 Len=0 MSS=1460 TSV=9615955 TSER=0 WS=0
10	10.1.1.98	10.1.1.100	TCP	http > 32824 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=66767 TSER=9615955 WS=0
11	10.1.1.100	10.1.1.98	TCP	32824 > http [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=9615955 TSER=66767
12	10.1.1.100	10.1.1.98	HTTP	GET / HTTP/1.1
13	10.1.1.98	10.1.1.100	TCP	http > 32824 [ACK] Seq=1 Ack=201 Win=6432 Len=0 TSV=66767 TSER=9615956
14	10.1.1.98	10.1.1.100	HTTP	HTTP/1.1 200 OK (text/html)
15	10.1.1.100	10.1.1.98	TCP	32824 > http [ACK] Seq=201 Ack=717 Win=7160 Len=0 TSV=9615959 TSER=66767
16	10.1.1.100	10.1.1.98	TCP	32824 > http [FIN, ACK] Seq=201 Ack=717 Win=7160 Len=0 TSV=9617165 TSER=66767
17	10.1.1.98	10.1.1.100	TCP	http > 32824 [FIN, ACK] Seq=717 Ack=202 Win=6432 Len=0 TSV=66888 TSER=9617165
18	10.1.1.100	10.1.1.98	TCP	32824 > http [ACK] Seq=202 Ack=718 Win=7160 Len=0 TSV=9617165 TSER=66888

A kapcsolattartás során használt csomagok szerepe a következő:

1. Az első csomagban a webböngészőt futtató számítógép ARP csomagban érdeklődik a DNS kiszolgáló Ethernet címe felől.
2. A második csomagban a DNS kiszolgáló megadja a saját Ethernet címét az ARP csomagra válaszolva.
3. A harmadik csomagban a webböngészőt futtató számítógép érdeklődik a webkiszolgáló IP címe felől a DNS kiszolgálótól. A kérdés kimondottan IPv6 címrre vonatkozik.
4. A negyedik csomagban a DNS kiszolgáló válaszul elmondja, hogy ennek a számítógépnek nincs IPv6 címe.
5. Az ötödik csomagban a webböngészőt futtató számítógép újra a webkiszolgáló címe felől érdeklődik, most már megelégedve az IPv4 címmel is.
6. A hatodik csomagban a DNS kiszolgáló válaszul megadja a webkiszolgáló IPv4 címét. Sikerült tehát meglelni a webkiszolgálót, amellyel fel kell vennie a böngészőnek a kapcsolatot.

7. A webböngésző és a webkiszolgáló ugyanazon az alhálózaton belül található, ezért a hetedik csomagban a webböngészőt futtató számítógép egy ARP csomagban megkíséri kideríteni a webkiszolgáló Ethernet címét.
8. A következő csomagban a webkiszolgáló válaszul megadja a saját Ethernet címét.
9. Az igazán izgalmas dolgok a kilencedik csomaggal kezdődnek, amikor a webböngészőt futtató számítógép TCP kapcsolat kiépítését kezdeményezi egy TCP protokollú csomaggal.

A kilencedik csomag egy TCP csomag, amely a 80-as hálózati kapura irányul. A csomagban be van kapcsolva a SYN bit (*synchronize sequence numbers*, sorszámok egyeztetése), amely jelzi, hogy a küldő fel kívánja venni a kapcsolatot a TCP alapprototkollt használva. A csomag segítségével a küldő azt is jelzi, hogy az általa küldött bájtokat milyen sorszámtól kezdve fogja számozni.

Az olyan TCP csomagok, amelyekben a SYN bit be van kapcsolva, de az ACK bit nincs a kapcsolat felvételét kér. Az ilyen csomagokat syn csomagoknak is szoktuk nevezni.

10. A tizedik csomagban a webkiszolgálót futtató számítógép jelzi, hogy kész a kapcsolatfelvételre. Ebben a csomagban be van kapcsolva a SYN bit és az ACK bit (*acknowledgement*, nyugtázás).

Ezzel a csomaggal a felkeresett számítógép egyrérszről igazolja az előző csomag vételét, másrészről jelzi, hogy az általa küldött bájtokat milyen sorszámtól kezdve fogja számozni.

11. E csomaggal a kapcsolatot kezdeményező számítógép jelzi, hogy az előző – a tizedik – csomagot megkapta.

Ebben a csomagban csak az ACK bit van bekapcsolva. Az ilyen csomagok a fogadott csomagok visszaigazolására szolgálnak.

Ezzel a csomaggal a kapcsolat kiépítése befejeződött, az adatok küldése megkezdődhet.

12. A tizenkettedik csomaggal a kiépült TCP kapcsolaton keresztül a webböngésző kérést intéz a webkiszolgáló felé. A kérés a TCP protokoll szempontjából egyszerűen átvendő adat, amely a webkiszolgáló számára jelzi, hogy mit szeretne a böngésző.

13. Ezzel a csomaggal a webböngészőt futtató számítógép igazolja vissza az előző csomag vételét. A csomagban be van kapcsolva az ACK bit, amelyből láthatjuk, hogy a csomag adatok vételét igazolja vissza. Az Ack=201 ki-fejezésből azt is láthatjuk, hogy a visszaigazoló csomag az első 201 bájt vételének igazolása. (Valójában – amint láttuk – a küldött bájtokat nem szükségszerűen számozzuk 0-tól kezdődően.)

A TCP alapprotokoll nem írja elő, hogy minden egyes csomagot igazoló cso-maggal kell megválaszolni, a címzett egy igazoló csomaggal több vett cso-magot is igazolhat.

14. A tizenegyedik csomagban a webkiszolgáló által küldött weblap található. A példa készítésekor olyan egyszerű weblapot használtunk, ami elfért egyetlen csomagban is, de ha ez nem így lenne, a webkiszolgáló több csomagban küldené el az adatokat.
15. A következő csomagban a webböngészőt futtató számítógép igazolja vissza az adatok vételét. Figyeljük meg, hogy az ACK bit be van kapcsolva és az Ack=717 kifejezésből láthatjuk, hogy a visszaigazolás a 717. bájtig az összes adat fogadását igazolja.
16. A tizenhatodik csomaggal a webböngésző megkezdi a felépített kapcsolat bontását. A csomagban be van kapcsolva a FIN bit (*finalize*, befejez), ami jelzi, hogy a kapcsolat befejezését kéri a webböngésző.
A csomag ugyanakkor a tizenötödik csomaghoz hasonlóan visszaigazolja az első 201 bájt vételét (ACK és Ack=717). Általánosságban elmondható, hogy a küldött csomagok sokszor másodlagos visszaigazoló szereppel is bírnak.
17. A tizenhetedik csomaggal a webböngészőt futtató számítógép szintén a kapcsolat megszakítását jelzi. Ezzel a két oldal megegyezett abban, hogy a kapcsolatot megszakítják.
18. A tizennyolcadik és egyben utolsó csomaggal a webböngészőt futtató számítógép visszaigazolja az utolsó csomag vételét, és ezzel a kapcsolat befejeződik.

Foglaljuk össze néhány szóban a TCP csomagok által megalósított folyamatos adatátvitel működését, vizsgáljuk meg, hogy milyen eszközök biztosítják az adatátvitel zavartalanosságát!

Elsőként meg kell említenünk a TCP csomagokban található SYN (*synchronize*, összhangol) és ACK (*acknowledge*, nyugtá) biteket. Ha a csomagban a SYN bit be van kapcsolva és az ACK bit nincs, az adott csomag kapcsolatfelvételt kérő TCP csomag, amelyben a kapcsolatot kérő munkaállomás (ügyfél) össze kívánja hangolni az adatbájtok számozását az ellenállomással. Az ilyen csomagokban a Seq (*sequence*, sorozat) érték határozza meg, hogy a küldő munkaállomás a legelső (nulla sorszámú) általa küldött adatbájthoz milyen sorszámot rendel.

Ha a TCP csomagban a SYN és az ACK bit is be van kapcsolva, akkor az adott csomag a kapcsolatfelvételt kérő csomag visszaigazolása. Ebben a csomagban a Seq értéke megadja, hogy a küldő a legelső (nulla sorszámú) küldött adatbájthoz milyen sorszámot rendel.

A kapcsolatfelvétel során tehát kialakul az adatbájtok számozásának a rendje. A TCP alapprotokoll szerint a kapcsolatot kérő és a kapcsolatot elfogadó számítógépek véletlenszerű sorszámot rendelnek a legelső adatbájthoz, és a későbbiekben e szerint számozzák a csomagokban található bajtokat.

A kezdő csomagokban és a kapcsolat további soraiban is megtalálható az Ack érték – amit nem szabad összekevernünk az ACK bittel. Az Ack mező értéke megadja, hogy a küldő mennyi adatot kapott; értéke pontosan megegyezik az általa várt következő bajt sorszámaival. Az Ack mező segítségével a fogadó tudni fogja, hogy az ő általa küldött adatbájtok közül hány érkezett meg, így kiderítheti, hogy szükség van-e a csomagok újraküldésére.

Mindkét oldal csomagjaiban megtalálható a Win (window, ablak) érték, amely megadja, hogy a csomagot küldő hány bajt fogadására kész. Ha a csomagok túlságosan gyorsan érkeznek, a fogadó a válaszul küldött csomagokban a Win értékének csökkentésével jelezheti, hogy csökkenteni kell az iramon, mert nem képes az adott ütemben feldolgozni a csomagokat. Ez a rendszer lehetővé teszi, hogy lassú és gyors számítógépek között is hatékony kapcsolattartás működhessen.

Ezek után vizsgáljuk meg a bemutatott kapcsolat két TCP csomagi részleteiben is! Az első csomag a webböngésző által a webkiszolgálónak küldött csomag, amelyben egy adott weblap elküldését kéri.

```

1 Frame 12 (266 bytes on wire, 266 bytes captured)
2 Ethernet II, Src: 00:02:3f:15:34:bb, Dst: 00:03:ba:1d:a4:48
3   Destination: 00:03:ba:1d:a4:48 (SunMicro_1d:a4:48)
4   Source: 00:02:3f:15:34:bb (CompaqEl_15:34:bb)
5   Type: IP (0x0800)
6 Internet Protocol, Src Addr: 10.1.1.100, Dst Addr: 10.1.1.98
7   Version: 4
8   Header length: 20 bytes
9   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
10  Total Length: 252
11  Identification: 0x872a (34602)
12  Flags: 0x04 (Don't Fragment)
13  Fragment offset: 0
14  Time to live: 64
15  Protocol: TCP (0x06)
16  Header checksum: 0x9c0a (correct)
17  Source: 10.1.1.100 (10.1.1.100)
18  Destination: 10.1.1.98 (10.1.1.98)
19 Transmission Control Protocol, Src Port: 32824 (32824), Dst Port: http (80), Seq: 1, \
20 Ack: 1, Len: 200
21   Source port: 32824 (32824)
22   Destination port: http (80)
23   Sequence number: 1 (relative sequence number)
24   Next sequence number: 201 (relative sequence number)
25   Acknowledgement number: 1 (relative ack number)
26   Header length: 32 bytes
27   Flags: 0x0018 (PSH, ACK)
28   Window size: 5840
29   Checksum: 0xe5da (correct)
30   Options: (12 bytes)
31 Hypertext Transfer Protocol
32   GET / HTTP/1.1\r\n
33   Host: sun.lkk.\r\n
34   User-Agent: ELinks/0.9.1 (textmode; Linux; 80x24)\r\n
35   Referer: http://sun.lkk./\r\n
36   Accept: */*\r\n

```

```

37 Accept-Encoding: bzip2, gzip\r\n
38 Accept-Language: hu\r\n
39 Connection: Keep-Alive\r\n
40 \r\n

```

A csomag a 19. sor tanúsága szerint a TCP alapprotokoll szerint épül fel. A 27. sorban láthatjuk, hogy nem kapcsolatfelvételi kérelemről van szó, hiszen a SYN kapcsoló nincs bekapcsolva.

A 23. sorban láthatjuk, hogy ez az adatcsomag a böngésző által küldött 1. bájt-nál kezdődik, amiből láthatjuk, hogy a böngésző ez előtt a csomag előtt nem küldött adatbájtokat a kiszolgálónak (a csomagszámozás 0-ról indult). A 24. sor alapján láthatjuk, hogy a böngésző a következő általa küldött csomagban a 201. adatbájtot fogja küldeni, bár nyilván nem lehetünk benne biztosak, hogy lesz még küldendő adat.

A 25. sorban láthatjuk, hogy a böngészőt futtató munkaállomás a kiszolgálótól az 1. adatbájtot várja a válaszcsomagban, ami jelzi, hogy a kiszolgáló még nem küldött egyetlen adatbájtot sem (a csomagszámozás 0-tól indult).

A 28. sor szerint a böngészőt futtató munkaállomás 5840 bájt fogadására készült fel, a kiszolgáló ennél nagyobb adatmennyiséget nem küldhet.

A 28. sorban olvashatjuk, hogy az ügyfél a kiszolgáló 80-as logikai kapujára küldte a csomagot. Ez a logikai kapu a webkiszolgáló jól ismert logikai kapuja, a 31. sorban kezdődő adatterület a HTTP (*hypertext transfer protocol*) protokoll szerint felépülő kérés. Olvassuk el figyelemesen a 32–40. sorok között található kérést, és figyeljük meg, hogyan kéri a GET parancs segítségével a / dokumentum letöltését a böngésző! Láthatjuk, hogy a HTTP protokoll szöveges kapcsolattartási módot ír elő, ahol a sorok a \r\n kettős jellel zárolnak, a kérés végét pedig az üres sor jelzi.

Vizsgáljuk most meg a webkiszolgáló által küldött válaszcsomagot, és hasonlítsuk össze a kérést tartalmazó csomaggal!

```

1 Frame 14 (782 bytes on wire, 782 bytes captured)
2 Ethernet II, Src: 00:03:ba:1d:a4:48, Dst: 00:02:3f:15:34:bb
3   Destination: 00:02:3f:15:34:bb (Compalel_15:34:bb)
4   Source: 00:03:ba:1d:a4:48 (SunMicro_1d:a4:48)
5   Type: IP (0x0800)
6 Internet Protocol, Src Addr: 10.1.1.98, Dst Addr: 10.1.1.100
7   Version: 4
8   Header length: 20 bytes
9   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
10  Total Length: 768
11  Identification: 0x0622 (1570)
12  Flags: 0x04 (Don't Fragment)
13  Fragment offset: 0
14  Time to live: 64
15  Protocol: TCP (0x06)
16  Header checksum: 0xb0f (correct)
17  Source: 10.1.1.98 (10.1.1.98)
18  Destination: 10.1.1.100 (10.1.1.100)
19 Transmission Control Protocol, Src Port: http (80), Dst Port: 32824 (32824), Seq: 1, \
20 Ack: 201, Len: 716
21   Source port: http (80)
22   Destination port: 32824 (32824)

```

```

23     Sequence number: 1      (relative sequence number)
24     Next sequence number: 717      (relative sequence number)
25     Acknowledgement number: 201    (relative ack number)
26     Header length: 32 bytes
27     Flags: 0x0018 (PSH, ACK)
28     Window size: 6432
29     Checksum: 0x6819 (correct)
30     Options: (12 bytes)
31 Hypertext Transfer Protocol
32     HTTP/1.1 200 OK\r\n
33     Date: Tue, 27 Jul 2004 09:01:54 GMT\r\n
34     Server: Apache/1.3.26 (Unix) Debian GNU\slash Linux mod_perl/1.26\r\n
35     Last-Modified: Fri, 23 Jul 2004 19:23:37 GMT\r\n
36     ETag: "301fc-17c-410165b9"\r\n
37     Accept-Ranges: bytes\r\n
38     Content-Length: 380\r\n
39     Keep-Alive: timeout=15, max=100\r\n
40     Connection: Keep-Alive\r\n
41     Content-Type: text/html; charset=iso-8859-1\r\n
42     \r\n
43 Line-based text data: text/html

```

A válaszként küldött csomagot összehasonlítva a kérést tartalmazó csomaggal láthatjuk hogyan feleltethetők meg egymásnak a küldő és a címzett adatai (MAC cím, IP cím, TCP logikai kapu), hogyan nyugtázza a kiszolgálóoldal az ügyféloldal által küldött csomag adatbájtjait, hogyan alakul az adatbájtok sorszámozása.

Figyeljük meg a 32–42. sorokban található adatsorokat. Ez a rész a webkiszolgáló által küldött fejlécet tartalmazza, amelynek formáját a HTTP protokoll írja le. A HTTP protokoll szerint a webkiszolgáló által visszaküldött weblapok elé fejlécet kell illeszteni, amely szöveges adja meg a küldött adatok formáját és legfontosabb jellemzőit. A fejléc sorait a \r\n kettős jel zárja le, a fejléc végét pedig egy üres sor jelzi.

A fejléc után – a 43. sorban – az átküldött weblap következik. Ezt a részt nem mutatjuk be részletesen, csak jelezük, de nyilvánvaló, hogy a hálózat lehallgatása során könnyen olvasható szöveges adatsorokat kapunk, mivel a HTTP protokoll nem használ titkosítást.

6.5. A csomagok szűrése

Az ethereal programban többféle célra használhatunk nyelvi kifejezésekkel felépített szűrőket:

- A lehallgatószűrők (*capture filters*) segítségével a hálózaton lehallgatott csomagokat szűrhetjük. Azokat a csomagokat, amelyeket az éppen beállított lehallgatószűrő kiszűr, a program figyelmen kívül hagyja. Ezzel a szűrővel tehát nagymértékben csökkenhetjük a program munkáját, hiszen csak azokat a csomagokat kell feldolgozni, amelyeket a szűrő átengedett.

Ezt a szűrőt a lehallgatás előtt kell beállítanunk a megjelenő ablakban vagy a *capture* menü *capture filters* menüpontja alatt.

- A megjelenítő szűrő (*display filter*) a lehallgatás után használható a megjelenítendő csomagok kiválasztására. Ezzel a szűrővel a lehallgatott adatok elemzése során választhatjuk ki, hogy a tárolt csomagok közül melyeket kívánjuk megjeleníteni.

Ezt a szűrőt a program főablakának felső részén kell beállítanunk a *filter* nyomógombnál. A főablak felső részén csak akkor jelenik meg a szűrőbeállítás, ha a *view* menü *filter toolbar* menüpontja ki van választva.

- A színválasztó szűrő (*coloring rules*) segítségével a lehallgatott csomagokhoz rendelhetünk színeket. Így könnyebben átláthatóvá tehetjük a megjelenített csomagok listáját.

A színválasztó szűrőket a *view* menü *coloring rules* menüpontját választva módosíthatjuk, de a színes megjelenítéshez a *view* menü *colorize packet list* menüpontját is bekapcsolva kell tartanunk.

- A szűrőket használhatjuk a statisztikai adatok grafikonos megjelenítésekor is. A programnak ezt a szolgáltatását a *statistics* menü *i/o graphs* menüpontja alatt találhatjuk meg.

A szűrők beállítására egy viszonylag egyszerű, de annál több elemet tartalmazó nyelvet használhatunk. A nyelv minden egyes kifejezése egy logikai értéket ad az adott csomaggal összevetve. Ha a kifejezés logikai igaz értéket ad az adott csomaggal, a csomag „átment” a szűrőn, illetve ha a logikai érték hamis, a csomag „fennakadt” a szűrőn.

A szűrkifejezésben a csomag egyes mezőire, egyes tulajdonságaira hivatkozhatunk, valamint különféle állandókat és műveleti jeleket adhatunk meg. A csomagok mezőinek elnevezése faszerkezetet követ, ahol az egyes ágakat „.” karakterrel kell elválasztanunk. A csomagok mezőire önmagukban is hivatkozhatunk – igaz logikai értéket képviselnek, ha az adott csomagban léteznek –, és a megfelelő típusú értéket hordoznak – állandókkal vagy más mezőkkel összehasonlíthatók. Ezt mutatja be a következő példa.

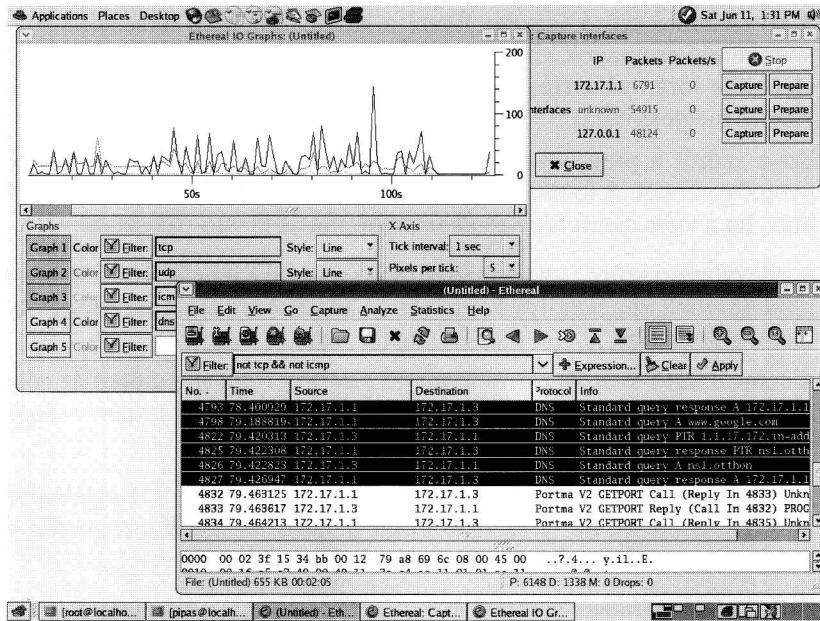
55. példa. A TCP csomagokat kiszűrhetjük a *tcp* kifejezéssel, amely igaz logikai értéket ad minden TCP csomag esetében.

A TCP csomagon belül található jelzőbitekre a *tcp.flags* kifejezéssel hivatkozhatunk, amely egy egész számot ad. Az adatok elemzése során könnyedén megállapíthatjuk, hogy hogyan szűrhetjük ki azokat a csomagokat, ahol csak az ACK bit van bekapcsolva:

```
1  tcp.flags == 0x10
```

Ha mindenötöt a csomagokat meg akarjuk jeleníteni, amelyeknél az ACK bit be van kapcsolva, használhatjuk az *ack* mezőt:

```
1  tcp.flags.ack == 1
```



6.5. ábra. A szűrők használata a lehallgatott adatok elemzése közben

Ezt egyszerűbben `tcp.flags.ack` formában is írhatjuk.

A következőkben megvizsgáljuk a szűrőkifejezések legfontosabb elemeit és műveleti jeleit. Erről a téma ról a `man ethereal-filter` parancs segítségével bővebben is olvashatunk.

6.5.1. Állandók és műveleti jelek

A szűrőkifejezésben az állandókat a következő formában adhatjuk meg:

számok A szűrőkifejezésekben a számokat a szokásos módon adhatjuk meg tízes (10), nyolcas (010) és tizenhetes számrendszerben.

logikai értékek A hamis logikai értéket a 0 számmal, az igaz logikai értéket az 1 számmal jelezhetjük.

IP címek Az IP címeket pontozott formában tízes számrendszerben (172.17.1.1) vagy tartománynével (akarmi.hu) adhatjuk meg.

IP címtartomány Az IP címek tartománya a címmel vagy a névvel és a hálózati címrész hosszával adható meg (172.17.1.0/24 vagy akarmi.hu/24).

Jel	Jelentés
eq, ==	egyenlő
ne, !=	nem egyenlő
gt, >	nagyobb, mint
lt, <	kisebb, mint
ge, >=	nagyobb vagy egyenlő
le, <=	kisebb vagy egyenlő

6.3. táblázat. Az ethereal szűrőkifejezések összehasonlító műveletei

Forma	Jelentés
[i:j]	i-edik elemtől j darab elem
[i-j]	i-edik elemtől j-edik elemig
[i]	az i-edik elem
[:j]	az első j elem
[i:]	az i-edik elemtől az összes elem
[x,y]	az x és y által meghatározott részek egymás után

6.4. táblázat. Az ethereal szűrőkifejezések részláncműveletei

Ethernet cím Az Ethernet címeket tizenhatos számrendszerben adhatjuk meg, az egyes bájtokat kettősponttal, ponttal vagy vesszővel elválasztva (például ff:ff:ff:ff:ff:ff).

Hasonlóképpen hivatkozhatunk minden mezőre, amelynek típusa adott hosszúságú bájtsorozat (tömb).

karakterláncok A karakterláncokat "" jelek közé írva használjuk. Ha a " jel része a karakterláncnak, egy \ jelet kell elé helyezni, ha pedig a \ jel része a karakterláncnak, duplán kell írni.

A szűrőkifejezésekben a következő műveleti jeleket használhatjuk:

összehasonlítás Az összehasonlításra használható műveleti jeleket a 6.3. táblázat mutatja be. Látható, hogy betűkkel és műveleti jelekkel is megadhatjuk ezeket a műveleteket.

részlánc A [] jelekkel karakterláncok és tömbök részét választhatjuk ki. A [] jeleket a mező neve után kell írnunk, jelentésük a 6.4. táblázatban leírt módon alakul.

logikai műveletek A logikai műveleti jelek a szokásos módon használhatók a logikai és (and, &&), a logikai vagy (or, ||) illetve a logikai tagadás (not, !) jelzésére.

`egész contains rész` A kifejezés a rész keresésére használható karakterláncban vagy tömbben (`contain`, tartalmaz). A bemutatott kifejezés igaz értéket ad, ha az egész karakterláncban megtalálható a rész karakterlánc.

`szöveg matches szabkjf`. Igaz értéket ad, ha a szöveg illeszkedik a megadott szabályos kifejezésre. A szabályos kifejezések használt nyelvjárásáról bővebben a `man pcrepattern` parancs segítségével olvashatunk.

A dokumentáció szerint az `ethereal` csak akkor képes az ilyen kifejezéseket értelmezni, ha a program fordításakor kértük ezt a lehetőséget.

6.5.2. Mezőnevek

A következőkben néhány szóban bemutatjuk a legfontosabb csomagformátumok legfontosabb mezőit. Felhívjuk a figyelmet arra, hogy az itt bemutatott szabványokon és mezőkön kívül még rengeteg található a dokumentációban, amelyet a `man ethereal-filter` parancs segítségével érhetünk el.

Bármely csomag

Minden Ethernet csomag esetében használhatjuk a következő állandókat:

`frame.cap_len` A lehallgatott csomagmáret bájtban (egész).

`frame.number` A csomag sorszáma (egész).

`frame(pkt_len` A csomag teljes mérete (egész).

56. példa. Válasszuk ki azokat a csomagokat, amelyek mérete nagyobb, mint 500 bájt!

```
1 frame(pkt_len > 500
```

Válasszuk most ki azokat a DNS csomagokat, amelyek mérete nagyobb, mint 500 bájt!

```
1 frame(pkt_len > 500 && dns
```

Ethernet csomagok

Minden Ethernet csomag esetében használhatjuk a következő állandókat:

`eth.dst` A cél Ethernet címe (6 bájtos Ethernet MAC cím).

`eth.len` A csomag mérete bájtban (egész).

`eth.src` A cél Ethernet címe (6 bájtos Ethernet címe).

57. példa. Válasszuk ki azokat a csomagokat, amelyeknek Ethernet címzettjeként nem körüzenet szerepel!

```
1 eth.dst != ff:ff:ff:ff:ff:ff
```

Válasszuk ki azokat a csomagokat, amelyeknek a címzettjeként egy bizonyos Ethernet című számítógép szerepel!

```
1 eth.dst == 00:12:79:a8:69:6c
```

Adjuk meg az egyszerűség kedvéért kizárolag az utolsó két bájtot!

```
1 eth.dst[4:] == 69:6c
```

ARP csomagok

Az ARP csomagok szűrésére használhatjuk – többek között – a következő mezőneveket:

arp.dst.hw A címzett hardvercíme (bájttömb).

arp.dst.hw_mac A címzett MAC címe (Ethernet cím).

arp.dst.proto_ipv4 A címzett IPv4 címe (IP cím).

arp.src.hw A forrás hardvercíme (bájttömb).

arp.src.hw_mac A forrás MAC címe (Ethernet cím).

arp.src.proto_ipv4 A forrás IPv4 címe (IP cím).

58. példa. Válasszuk ki azokat az ARP csomagokat, amelyek egy adott IP című számítógép felől érdeklődnek!

```
1 arp.dst.proto_ip == 172.17.1.2
```

Válasszuk ki azokat az ARP csomagokat, amelyek egy bizonyos Ethernet című számítógép IP címét közlik, de nem az adott számítógépről érkeznek!

```
1 arp.src.hw_mac==00:02:3f:15:34:bb && eth.src!=00:02:3f:15:34:bb
```

IP csomagok

Az IP csomagok szűrésére a következő mezőneveket használhatjuk:

`ip.checksum_bad` Az ellenőrzőösszeg hibás (logikai).

`ip.dst` A címzett IPv4 címe (IP cím).

`ip.flags` A csomagtípus bitek (egész).

`ip.flags.df` A csomagot tilos darabolni (logikai).

`ip.flags.mf` A csomag további darabokból áll (logikai).

`ip.frag_offset` A csomagdarab sorszáma (egész).

`ip.fragment` A csomag csomagdarab (logikai).

`ip.hdr_len` A csomag fejlécének mérete bájtban (egész).

`ip.len` A csomag teljes hossza bájtban (egész).

`ip.proto` A használt szabvány száma, ahogyan a `/etc/protocols` állományban megtalálható (egész).

`ip.src` A forrás IPv4 címe (IP cím).

`ip.tos` A kért kezelési mód (lásd lejjebb) (egész).

`ip.tos.cost` A legkisebb költséggel küldendő tovább (logikai).

`ip.tos.delay` A legkisebb késlekedési idővel küldendő tovább (logikai).

`ip.tos.reliability` A legmegbízhatóbb útvonalon küldendő tovább (logikai).

`ip.tos.throughput` A legnagyobb sávszélességű csatornán küldendő tovább (logikai).

`ip.tt1` A csomag hátralévő élettartama (egész).

59. példa. Válasszuk ki azokat az IP csomagokat, amelyek egy adott számítógépről indultak!

```
1 ip.src == 172.17.1.3
```

Válasszuk ki most azokat az IP csomagokat, amelyek egy adott hálózati címtartományba tartozó számítógéprekről indultak!

```
1 ip.src == 172.17.1.3/24
```

UDP csomagok

Az UDP alapprotokollú csomagok szűrésekor a következő mezőneveket használhatjuk:

`udp.checksum_bad` A csomag ellenőrzőösszege hibás (logikai).

`udp.dstport` A cél logikai kapu (egész).

`udp.length` A csomag mérete bájtban (egész).

`udp.srcport` A forrás logikai kapu (egész).

60. példa. Válasszuk ki azokat az UDP alapprotokollú csomagokat, amelyek az 53. logikai kapura érkeztek!

```
1  udp.dstport == 53
```

Válasszuk ki ezek közül azokat a csomagokat, amelyek forrás logikai kapuja nagyobb, mint 1024!

```
1  udp.dstport == 53 && udp.srcport > 1024
```

TCP csomagok

A TCP alapprotokollú csomagok szűrésekor a következő mezőneveket használhatjuk (meg kell jegyeznünk, hogy a dokumentáció meglehetősen sok, a hibakereséskor használható elemzőértéket is felsorol, amelyek a program által felfedezett hibák szerint szűrnék):

`tcp.checksum_bad` Az ellenőrzőösszeg hibás (logikai).

`tcp.dstport` A címzett logikai kapu (egész).

`tcp.flags.ack` Az ACK bit értéke (logikai).

`tcp.flags.fin` A FIN bit értéke, amely jelzi, hogy a küldő meg kívánja szakítani a kapcsolatot (logikai).

`tcp.flags.syn` A SYN bit értéke (logikai).

`tcp.hdr_len` A fejléc mérete bájtban (egész).

`tcp.len` A csomag teljes mérete bájtban (egész).

`tcp.seq` A Seq mező értéke, az adatbájt sorszáma (egész).

`tcp.srcport` A forrás logikai kapu (egész).

`tcp.window_size` A Win mező értéke (egész)

61. példa. Válasszuk ki azokat a csomagokat, amelyekben a SYN bit be, az ACK bit pedig ki van kapcsolva! Ezek a kapcsolat felépítését kérő TCP csomagok.

```
1  tcp.flags.syn == 1 && tcp.flags.ack != 1
```

Mivel a `tcp.flags.syn` és a `tcp.flags.ack` logikai értéket hordoznak, felmerülhet a kérdés, hogy miért nem a következő – egyszerűbb – formában adtuk meg a szűrőkifejezést:

```
1  tcp.flags.syn && !tcp.flags.ack
```

Ennek a kifejezésnek azonban a következő a jelentése: „válasszuk ki azokat a csomagokat, amelyekben a `tcp.flags.syn` és a `tcp.flags.ack` mezők léteznek”.

SMTP csomagok

Az elektronikus levelek továbbítására szolgáló SMTP szabvány esetében a következő mezőneveket használhatjuk:

`smtp.req` A csomag SMTP kérést tartalmaz (logikai).

`smtp.req.command` A kérést leíró parancs (karakterlánc).

`smtp.req.parameter` A kérést leíró parancs paramétere (karakterlánc).

`smtp.response.code` A csomag által hordozott válaszkód (egész).

`smtp.rsp` A csomag SMTP válasz a kiszolgálótól.

`smtp.rsp.parameter` A válasz paramétere (karakterlánc).

62. példa. Válasszuk ki azokat a csomagokat, amelyekben a levelezést fogadó kiszolgáló jelzi, hogy az adott tartományból nem fogad leveleket! Mivel az SMTP szabvány erre az üzenetre az 550-es kódot használja (lásd a 261. oldalon), a következő szűrő felel meg a céljainknak:

```
1  smtp.response.code == 550
```

FTP csomagok

Az állományok átvitelére szolgáló FTP protokollú csomagok szűrésére a következő mezőneveket használhatjuk:

`ftp.request` Igaz értékű, ha a csomag kérést tartalmaz (logikai).

`ftp.request.arg` A kérés argumentuma (karakterlánc).
`ftp.request.command` A kérés parancsa (karakterlánc).
`ftp.response` Igaz értékű, ha a csomag választ tartalmaz (logikai).
`ftp.response.arg` A válasz argumentuma (karakterlánc).
`ftp.response.code` A válasz kódja (egész).

HTTP csomagok

A weblapok átvitelére szolgáló HTTP protokollú csomagok szűrésére a következő mezőneveket használhatjuk (a dokumentáció ezeken a mezőneveken kívül még sok hasznos mezőt felsorol):

`http.accept` A webböngésző által elfogadott MIME típusokat jelző mező értéke a kérésben (karakterlánc).
`http.accept_encoding` A webböngésző által elfogadott kódolást megadó mező értéke a kérésben (karakterlánc).
`http.accept_language` A webböngésző által elfogadott nyelvet megadó mező értéke a kérésben (karakterlánc).
`http.content_encoding` A válasz tartalmának kódolását megadó mező értéke (karakterlánc).
`http.content_length` A válaszul küldött tartalom méretét bajtban jelző mező értéke (karakterlánc).
`http.content_type` A válaszul küldött tartalom MIME típusát megadó mező értéke (karakterlánc).
`http.request` Igaz, ha a csomag HTTP protokollú kérést tartalmaz (logikai).
`http.request.method` A kérés típusát (GET, POST) megadó kulcsszó, amely a HTTP protokollban kulcsfontosságú (karakterlánc).
`http.request.uri` A kért dokumentum erőforrásleírása, a kért „hivatkozás” (karakterlánc).
`http.response` Igaz értékű, ha a csomag HTTP válasz (logikai).
`http.response.code` A válasz kódja (például 200 hibamentes esetben), amely meghatározza, hogy sikerült-e a kérést teljesíteni, és ha nem, mi volt a hiba (egész).
`http.server` A kiszolgálóprogram neve a válaszcsomagban (karakterlánc).

`http.user_agent` A webböngésző neve a kérésben (karakterlánc).

63. példa. Válasszuk ki azokat a HTTP protokollt használó kéréseket, amelyekben szerepel a `.gif` karakterlánc!

```
1 http.request && http contains .gif
```

Ezek a csomagok nagy valószínűséggel a GIF formátum alapján felépített képeket kérnek a kiszolgálótól.

A rendszernapló csomagjai

Amint láttuk a UNIX rendszerek rendszernaplójának kezelését végző `syslog` a számítógép-hálózaton keresztül is képes üzeneteket küldeni és fogadni. A rendszernaplázást hálózaton közvetítő szabványnak megfelelő csomagok szűrésére a következő mezőneveket használhatjuk:

`syslog.facility` A naplóbejegyzés témaköre (egész).

`syslog.level` A naplóbejegyzés fontosságának szintje (egész).

`syslog.msg` A naplóbejegyzés szövege (karakterlánc).

7. fejezet

Hálózatok összekapcsolása

Láttuk, hogy a számítógépek a saját IP címük és az alhálózati maszk segítségével képesek megállapítani, hogy a számítógép, amellyel fel kívánják venni a kapcsolatot, ugyanabban a címtartományban van-e, amelyikben ők maguk. Azt is láttuk, hogy az alhálózaton belüli számítógépek felkeresésében az ARP szabvány hogyan segíti a számítógépet, hogy miképpen lehet az ARP szabvány segítségével kideríteni egy adott IP című számítógép Ethernet címét.

A könyv előző fejezeteiben azt is láttuk, hogy az útválasztó táblázat alapján hogyan lehet eldönteni, mi a teendő, ha a csomagokat nem lehet közvetlenül továbbítani a címzettnek, mert az nem a csomagot küldő számítógéppel azonos fizikai hálózaton található. Ilyen esetben a számítógép – az útválasztó táblázat alapján – a csomagot egy harmadik számítógépnek, az útválasztónak (*router*) küldi el. A csomag továbbítását az útválasztó magára vállalja.



Előfordulhat, hogy egy adott fizikai hálózaton több IP címtartomány is jelen van egy időben. Ilyen esetben nem csak a számítógéppel azonos címtartományban található számítógépeknek lehet közvetlenül, az ARP szabvány segítségével elküldeni a csomagot, hanem az összes számítógépnek, amely az adott fizikai hálózaton található. Az ilyen felépítés azonban nem okozhat problémát, hiszen ilyen esetben az útválasztó táblázatból kiderül, hogy melyek azok a címtartományok, amelyeknek közvetlenül küldhető el a csomag.

Nagyon egyszerű a csomagok útválasztó felé való küldésének módja is. A csomagot küldő számítógép egyszerűen kideríti az ARP szabvány szerint, hogy mi az útválasztó Ethernet címe, elhelyezi a csomagban az igazi címzett IP címét, és az Ethernet cím felhasználásával elküldi az útválasztónak. Az útválasztó megkapja a csomagot az Ethernet cím alapján, és a címzett IP címéből kideríti, hogy az nem neki szól. Ha a csomaggal egyébként minden rendben van, az útválasztó egyszerűen továbbítja azt a saját útválasztó táblázata alapján. Az útválasztó számítógép tehát egyszerűen a saját útválasztó táblázata alapján továbbküldi a csomagot,

gyakorlatilag ugyanúgy, mintha az a saját adatcsomagja lenne. Ha egy számítógép képes csomagokat küldeni, útválasztóként is használható, feltéve persze, hogy a megfelelő beállításokat elvégeztük.

Néhány apróságot azért érdemes megemlítenünk az útválasztó számítógépekkel kapcsolatban:

- Az útválasztónak általában nem szabad továbbküldenie a körüzeneteket, hogy az adott fizikai hálózat belső forgalma ne terhelje a vele kapcsolatban álló többi hálózatot.

Ha tehát a körüzeneteket is továbbítani kívánjuk, valószínűleg nem útválasztó számítógépre van szükségünk.

- Az útválasztó számítógép általában csak bizonyos IP címtartományból érkező csomagokat fogad el továbbításra. A legtöbb esetben például nem továbbítja a magáncélra használható IP címtartományokból érkező csomagokat. Ha az ilyen csomagokat is továbbítani kívánjuk – mert például több alhálózatból álló magánhálózatot üzemeltetünk –, az útválasztó számítógép megfelelő beállításáról esetleg gondoskodnunk kell.
- Ha elegendően sok hálózati csatolót helyezünk el egy számítógépben, és elegendően nagy hálózatok forgalmát próbáljuk meg áterőszakolni az adott számítógépen, akkor fennakadások lehetnek a forgalomban. A GNU/Linux egy régebbi számítógépen is képes néhány tucat számítógép forgalmát továbbítani, de ha nagysebességű (és nagyméretű) hálózatokat próbálunk meg összekapcsolni, érdemes egy kis fejszámolást végezni, mielőtt a boltba mennénk.

A következő oldalakon arról olvashatunk, hogyan tudunk hálózatokat összekapcsolni a GNU/Linux segítségével, hogyan gondoskodhatunk a hálózatok védelméről, és milyen módon bizonyosodhatunk meg arról, hogy a hálózatra kapcsolt számítógépeink (nagyjából) biztonságban vannak.

7.1. Az útválasztó beállítása

A Linux rendszermag alapértelmezés szerint nem vállal útválasztó szolgáltatásokat, azaz mások számára nem továbbítja a hálózaton érkezett csomagokat. Hiába tud tehát csomagokat küldeni a számítógépünk két hálózat felé is, a csomagokat mások számára nem továbbítja.

Ha be akarjuk kapcsolni a csomagtovábbítást, egyszerűen egy 1-es karaktert kell küldenünk a /proc/sys/net/ipv4/ip_forward látszólagos állományba. Ezt mutatja be a következő példa.

64. példa. Készítsünk héjprogramot, amely bekapcsolja a számítógépünkön a csomagtovábbítást IPv4 csomagok számára! A program igen egyszerű:

```
1 #! /bin/bash
2 echo 1 >/proc/sys/net/ipv4/ip_forward
```

/root/bin/forward.sh

7.2. A tűzfalak

Napjainkban a hálózati biztonság kiemelt fontosságú, egyetlen rendszergazda sem hagyhatja figyelmen kívül az általa üzemeltetett számítógép-hálózat biztonsági kérdéseit. A Linux rendszermag igen hatékony és rugalmas biztonsági rendszerrel van felszerelve, amely képes megvédeni a munkaállomásként üzemeltetett számítógépet, vagy akár a GNU/Linux által kiszolgált teljes hálózatot.

Az újabb (2.4 és a feletti változatú) Linux rendszermagok részét képezi a *netfilter* (*network filter*, hálózatszűrő), amely hálózati csomagok szűrésére és átalakítására teszi képessé az operációs rendszert. A *netfilter* a Linux rendszermag hálózati alrendszereibe épített programrészek gyűjteménye, amelynek használatával a különböző magmodulok képesek alapjaiban megváltoztatni a rendszermag ezen részének viselkedését. A rendszermaghoz készített *iptables* (*IP tables*, IP táblázatok) programcsomag magmodulokból és felhasználói programokból áll, amelyek segítségével könnyedén megvalósíthatjuk a csomagszűrésen alapuló védelmet. Az ilyen – csomagszűrésen alapuló – védelmet a szakirodalom általában tűzfalnak (*firewall*) nevezi.



Az angol szakirodalomban sokszor emlegetik a gépkosci motorháza és az utastér közt található vaslemezt, amely megakadályozza, hogy a motortűz átterjedjen az utastérre. Magyar nyelven talán ismerősebben hangszik a háztömbök padlásait egymástól elválasztó fal, amelynek hasonló a szerepe. A számítógépes tűzfal rendszerek esetében egyszerűen arról van szó, hogy a rendszermagba épített védelem elválasztja a hálózatot az alkalmazástól vagy a belső hálózaton üzemeltetett számítógépektől.

A régebbi Linux rendszermagokban más programrészek gondoskodtak a védelemről. A legrégebbi rendszermagok esetében az *ipfwadm* alkalmazást, az újabbakban a *ipchains* alkalmazást használtuk a tűzfal beállítására, ezért ezekre a régebbi rendszerekre általában *ipfwadm* és *ipchains* rendszerként hivatkozunk, míg az új, modern tűzfalakat sokszor *iptables* néven szokás emlegetni. Mivel a régebbi rendszerek megismerésébe már nem igazán érdemes energiát fektetni mi csak az *iptables* rendszerek használatát és működését mutatjuk be.

7.2.1. Egyszerű csomagszűrés

A hálózati csomagok egyszerű szűrésére két okból is szükségünk lehet. Ha a számítógép munkaállomásként az Internetre kapcsolódik mindenkorban szükséges, hogy a számítógépet megvédjük az Internet felől érkező támadásoktól – ezért már gyakorlatilag minden GNU/Linux terjesztés támogatja a tűzfal használatát –, ha pedig a számítógépünk valamelyen általunk üzemeltetett hálózatot kapcsol az Internetre, a tűzfal használata gyakorlatilag elengedhetetlen.



A számítógépeket üzemeltető szakemberek sok minden kitalálnak annak érdekében, hogy kevesebbet kelljen dolgozniuk, hiszen a lustaság a számítástechnika egyik legfontosabb eleme. Nem szabad azonban félvárról vennünk a biztonság kérdését! Gondolhatjuk, hogy a mi számítógépünk biztonságos¹, gondolhatjuk, hogy a mi számítógépünk nem támadja senki², bízhatunk a szerencsénkben³, bízhatunk abban, hogy a számítógépünk működése nem is fontos⁴, ez mit sem számít. Egy megfelelően beállított tűzfalra mindenkorban szükségünk van, ha a számítógépet az Internetre kötjük!

Az egyszerű csomagszűrés működése a következő lépésekben foglalható össze:

1. A számítógép indításának valamely pontján lefut az a héjprogram, amely a csomagszűrő tűzfalat helyezi üzembe. Ez a héjprogram az iptables programot hívja, hogy a csomagszűrő szabályokat beállítsa.
2. Az iptables a Linux rendszermaggal felvész a kapcsolatot, hogy a csomagok szűrésére vonatkozó szabályokat elhelyezze a rendszermagban.
3. A rendszermag az automatikus magmodul-betöltés rendszerének segítségével betölti a szükséges magmodulokat. Elsősorban az ip_tables modulra lesz szükség, amely az iptables rendszeren alapuló csomagszűrés alapmodulja, de további modulok betöltésére is sor kerülhet.
4. Amint a héjprogram betöltötte a csomagszűrő szabályokat, megtörténhet a hálózati csomagok fogadásának – és az esetleges továbbításának – engedélyezése.
5. Az üzemelő rendszermag a fogadott és küldött hálózati csomagok kezelése során figyelembe veszi a szabályokat, így a szabályok segítségével a hálózati alrendszer biztonságát finomhangolhatjuk.
6. Amikor a hálózati alrendszer leállítjuk (például azért, mert a számítógépet kikapcsoljuk), a csomagszűrő szabályokat a rendszermag eldobja. A hálózat indításakor a szabályokat újra be kell töltenünk a rendszermagba. Erről általában az első pontban említett héjprogram gondoskodik.

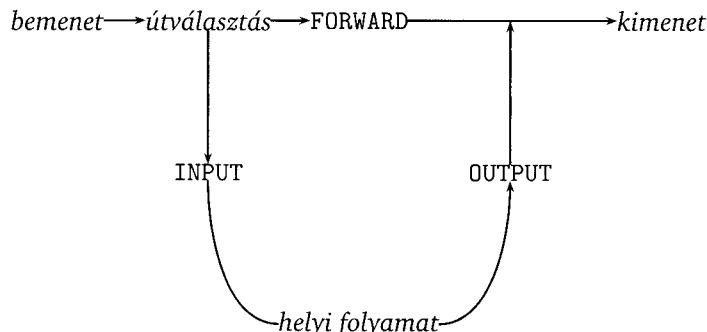
A rendszergazda szerepe a bemutatott folyamatban általában az, hogy elkezítse a csomagszűrő szabályokat betöltő héjprogramot, vagy egy már meglévő héjprogram beállítóállományát az igényeinek megfelelően módosítsa. A következő oldalakon arról olvashatunk hogyan tudunk olyan héjprogramot írni, amely a tűzfal csomagszűrő szabályait a rendszermagba tölti az iptables program segítségével.

¹Biztonságos számítógép nem létezik.

²A legkevésbé fontos számítógépet is támadják.

³A támadások száma exponenciálisan nő, ezért minden nap nagyobb és nagyobb szerencsére lesz szükségünk. Csalóka tehát az „eddig még jó...” hozzállás.

⁴Viszont a mi feltört számítógépünk ról támadott számítógép fontos lehet és igen kinos magyarázkodásnak nézhetünk elébe, ha a támadást a mi számítógépünkig nyomozza vissza valaki.



7.1. ábra. Az egyszerű csomagszűrés szabálylistái

A csomagszűrő szabályok adatforgalmát a 7.1. ábrán követhetjük nyomon.

A számítógépre érkező adatcsomagok az ábra *bemenet* nevű pontján jelennek meg, ahonnan az *útválasztás* névvel jelzett programrészhez érkeznek. A rendszermagnak ez a része minden csomag esetében megvizsgálja, azt más számítógép felé továbbítani kell-e. A más számítógép felé továbbítandó csomagok a FORWARD szabályrendszerhez érkeznek, míg a helyi számítógéphez – helyi folyamathoz – érkező csomagok az INPUT szabályrendszerre kerülnek.

A továbbítandó adatcsomagok, amelyek „túlélik” a FORWARD szabályrendszer megpróbáltatásait, eljutnak a kimenetre. A FORWARD szabályrendszer alapján megfelelőnek ítélt csomagokat a számítógép továbbítja a címzett irányába.

A helyi számítógépre érkező csomagok, amelyeket a rendszermag a INPUT szabályrendszer alapján elfogadhatónak ítélt meg, eljutnak a helyi folyamathoz. A rendszermag ezeket a csomagokat valamelyik felhasználói folyamat felé továbbítja.

Ha valamelyik helyi folyamat adatcsomagot próbál küldeni a hálózatra, az előbb a OUTPUT szabályrendszerre érkezik. Az adatcsomag továbbítása csak akkor történhet meg, ha a csomag megfelel az OUTPUT szabályrendszernek.

Fontos, hogy megértsük, az ábrán látható *bemenet/kimenet* pontok nem a számítógépünk bemeneti és kimeneti hálózati csatolóját jelentik! A bemenet azt a pontot jelzi, ahol a rendszermag megkapja a hálózati csomagot, a kimenet pedig azt a pontot, ahol a csomag elhagyja a rendszermagot, függetlenül attól, hogy ez melyik hálózati csatolon keresztül történik.

Vizsgájuk most meg a INPUT, OUTPUT és FORWARD szabályrendszereket! A szabályrendszerek minden szabály egy csomagformátumot és egy sorsot (*target*, cél) tartalmaz. A rendszermag sorra megvizsgálja, hogy az adott adatcsomag tulajdonságai megfeleltethetők-e az egyes szabályok által leírt formátumnak, és ha igen, az adott szabály határozza meg az adatcsomag sorsát. A rendszermag a szabályok vizsgálatát az első illeszkedő szabálynál befejezi, hiszen az első illeszkedő

szabály meghatározza a csomag sorsát⁵. Ha a rendszermag a szabályrendszer utolsó szabályát is megvizsgálta, és az sem volt illeszthető az adatcsomag tulajdonságaira, a szabályrendszer alapszabályát (*policy*, házirend) használja, az dönti el az adatcsomag sorsát.

E bevezető után megvizsgáljuk, hogy miképpen használható az *iptables* program az INPUT, OUTPUT és FORWARD szabályrendszer kezelésére.

A szabályrendszerek lekérdezése

A szabályrendszerekben található szabályok kiíratása igen egyszerű művelet, a kiírt adatok értelmezése azonban annál bonyolultabb. A következő sorokban bemutatjuk, hogyan írhatjuk ki a szabályokat, de a kiírt adatok pontos jelentése csak a következő oldalak gondos áttanulmányozása után érhető meg igazán. Az *iptables* programot használva a következő formában kérhetjük az érvényben lévő szabályok lekérdezését:

```
iptables --list [név] [-n] [-v] A kifejezésben szereplő --list vagy -L kapcsolóval a rendszermagban érvényben lévő szabályok kiíratását kérhetjük. A kifejezésben olvasható név a szabályrendszer nevét határozza meg; ha elhagyjuk, a program minden szabályrendszer tartalmát kiíratja.
```

A -n kapcsolóval megadhatjuk, hogy a számítógépek nevei helyett azok IP címei jelenjenek meg. Ezt a kapcsolót általában használjuk, mert ha probléma van a hálózattal, a DNS kiszolgáló nem érhető el és a program iszonyatosan lelassul.

A -v kapcsolóval a szabályok részletes kiírását kérhetjük. Akkor használjuk ezt a kapcsolót, amikor a szabályokat tüzetesen meg akarjuk vizsgálni.

65. példa. A következő példa egy üres szabályrendszert, védtelen számítógépet mutat be:

```
$ iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source                   destination
Chain FORWARD (policy ACCEPT)
target     prot opt source                   destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source                   destination
$
```

Figyeljük meg, hogy minden szabályrendszer üres, a szabályrendszerek neve alatt csak egy fejlécet látunk, egyetlen szabály sem jelenik meg.

⁵Néha azonban ez nem így van, de az egyszerűség érdekében ettől most tekintsünk el.

Az avatott szem azt is észreveszi, hogy a szabályrendszerhez kapcsolódó alapszabály az ACCEPT (elfogad), aminek megfelelően minden szabályrendszer minden csomagot elfogad.

66. példa. A következő példa egy szabályrendszert mutat be, amelyben egyetlen szabály található:

```
$ iptables -L INPUT -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP      tcp   --  0.0.0.0/0            0.0.0.0/0          tcp dpt:!22
$ iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP      tcp   --  anywhere           anywhere          tcp dpt:!ssh
$
```

A szabált IP címekkel és tartománynevekkel is lekérdeztük, hogy láthassuk a különbséget.

A példában bemutatott listák egyes oszlopaiban a következő információkat találjuk:

target Az oszlop a szabályra illeszthető csomagok sorsát tartalmazza.

prot Az oszlop a szabályhoz tartozó szabványt adja meg. A szabályra csak azok a csomagok illeszthetők, amelyek e szabvány alapján épülnek fel.

opt Az oszlop a csomagban található néhány jelzőbit előírt állapotát adja meg. A példában a bitek helyén a - karakter áll, ami azt jelzi, hogy a szabály nem tartalmaz ilyen jellegű megkötést.

source A szabály által előírt forrás IP címet (esetleg tartománynevét) tartalmazza. A példában a 0.0.0.0/0 és az anywhere (bárhova) kifejezések állnak ebben az oszlopból, ami azt jelenti, hogy a szabály nem tartalmaz megkötést a küldő IP címére vonatkozólag.

destination Az oszlop a szabályban előírt címzett IP címet tartalmazza.

tcp dpt:!ssh Az utolsó oszlopan további adatokat olvashatunk. A példában szereplő dpt a *destination port* (cél logikai kapu) rövidítése.

Az iptables által kiírt táblázatban a ! a tagadás jelzésére szolgál, az adott mező értékét ellentétre változtatja⁶.

67. példa. A következő példa egy összetettebb szabályrendszer részletes listáját mutatja be. A táblázatot kissé módosítottuk helytakarékkosság céljából, a túlságosan hosszú sorokat a \ jellel jelöltük.

⁶Figyeljünk rá, hogy a BASH sajátosan értelmezi a ! jelet, ha elfelejtünk mögé szóközt helyezni, és ez látszólag érthatélosztó problémákat okozhat (a lektor).

```
$ iptables -L -v
Chain INPUT (policy DROP 2974 packets, 187K bytes)
pkts bytes target prot opt in out source destination
  40  5485 ACCEPT all  --  lo  any anywhere anywhere
   28  2191 ACCEPT icmp --  any any anywhere anywhere
  483 36144 ACCEPT udp  --  any any anywhere anywhere      udp dpts:\1024:65535
1661 66456 ACCEPT tcp  --  any any anywhere anywhere      tcp dpts:\1024:65535 flags:!SYN,RST,ACK/SYN
    747 73055 ACCEPT tcp  --  any any 172.17.1.0/24 anywhere      tcp dpt:s\sh
24682 3083K ACCEPT tcp  --  any any anywhere anywhere      tcp dpt:h\http

Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 27291 packets, 4310K bytes)
pkts bytes target prot opt in out source destination
     0    0 ACCEPT icmp --  any lo  anywhere anywhere      icmp echo\-\-reply
     0    0 DROP   icmp --  any any anywhere !172.17.1.0/24 icmp echo\-\-reply
     8   320 DROP   icmp --  any !lo anywhere anywhere      icmp time\stamp-reply
$
```

Amint látjuk, a program által készített listában szerepelnek olyan információk is, amelyek az egyszerűsített listában nem jelentek meg. A szabályokat sokszor kizárálag a részletes lista alapján lehet megérteni!

A példában szereplő új oszlopok jelentése a következő:

pkts A csomagok száma, amelyek esetében az adott szabályt kellett alkalmazni.
Amint látjuk, az alapszabályoknál is megjelenik ez az érték.

bytes Az adatmennyiség, amelyet az adott szabály alkalmazása érintett. Amint látható, az alapszabály esetében is megjelenik ez az érték.

in A szabályban előírt bemenő hálózati csatoló neve.

out A szabályban előírt kimenő hálózati csatoló neve.

Az alapszabály megváltoztatása

Láttuk, hogy a szabályrendszerhez tartozik egy alapszabály, ami meghatározza, hogy mi a sorsa azoknak a csomagoknak, amelyekre egyetlen szabály sem volt illeszthető. A szabályrendszerhez tartozó alapszabály megváltoztatására az iptables a következő formában használható:

`iptables --policy név sors A --policy vagy -P` kapcsolóval adhatjuk meg, hogy mi legyen a sorsa azoknak a csomagoknak, amelyekre egyetlen szabály sem volt illeszthető. A kapcsoló után megadott név a szabályrendszer neve, a sors pedig az alapszabály által előírt sors.

A csomagok sorsára a következőben többször is visszatérünk, fokozatosan mutatjuk be a használható kulcsszavakat. Az alapszabályként használható legfontosabb sorsok a következők:

`ACCEPT` Az illeszkedő csomagot a szabály alapján a rendszermag elfogadja, azaz a csomag a szabályrendszernek megfelel.

`DROP` Az illeszkedő csomagot a szabály alapján a rendszermag eldobja. Az ilyen csomagról a csomagot küldő folyamat vagy számítógép nem kap értesítést.

68. példa. A következő példa bemutatja, hogyan tilthatunk ki minden átmenő forgalmat a számítógépünkéről az alapszabály megváltoztatásával:

```
1 iptables --policy FORWARD DROP
```

Nem szabad szem elől téveszteni, hogy a parancs végrehajtása után az alapszabály ugyan az lesz, hogy az átmenő csomagokat figyelmen kívül kell hagyni, de ha léteznek olyan szabályok a FORWARD szabályrendszerben, amelyek elfogadják az adott csomagot, a csomagra nem az alapszabály lesz az érvényes.

A szabályrendszer kiürítése

A szabályrendszerben található szabályokat egyetlen parancs kiadásával törölhetjük. Ezt mutatjuk be a következő néhány sorban.

A szabályrendszer törlése kapcsán meg kell jegyeznünk, hogy sokak szerint egyetlen pillanatra sem engedhetjük meg magunknak, hogy a számítógépünk védetlen legyen, mert előfordulhat, hogy éppen abban a pillanatban ér minket támadás (helyesebben mondva abban a pillanatban is támadás éri a számítógépünket). Ha tehát törölni akarjuk a szabályokat, és új szabályrendszert akarunk felépíteni, akkor előbb az alapszabály segítségével meg kell tiltanunk minden csomag fogadását, és csak utána szabad a szabályrendszer szabályait törölnünk.

Ez a probléma egy mélyebb jelentőségű kérdést is felvet. Állitsuk az alapszabályt tiltó jellegűre, és soroljuk fel a kivételeket, amelyeket megengedünk, vagy állítsuk az alapszabályt megengedő jellegűre, és soroljuk fel a kivételeket, amelyeket tiltunk. Az előbbi módszernek (minden tiltunk, amit nem soroltunk fel) sokkal több követője van, mivel biztonságosabb, előfordulhat azonban olyan eset, amikor a második (minden megengedünk, amit nem soroltunk fel) módszer célravezetőbb. Ha különlegesen megengedő tűzfalat telepítünk, ami csak a legnyilvánvalóbb támadásokat szűri ki, akkor a megengedő elrendezést érdemes használni. Ekkor viszont a hálózatunkat egy belső, szigorú tűzfallal kell védenünk! Ez azt jelenti,

hogy a megengedő elrendezést csak többszintű tűzfalrendszer esetén használjuk, ami a kimondottan nagy számítógép-hálózatokra jellemző, ahol a több száz vagy több ezer munkaállomás védelmét hierarchikus csomagszűrő rendszerrel látjuk el.

A rövid bevezető után következzék a szabályrendszer összes szabályának törlésére használható parancs:

`iptables --flush [név]` A kifejezésben szereplő `--flush` vagy `-F` kapcsolóval a szabályrendszer minden szabálya törölhető. A parancsban szereplő név a szabályrendszer neve, amelyből a szabályokat törölni szeretnénk. Ha nem adunk meg nevet, a program az `INPUT`, a `FORWARD` és az `OUTPUT` szabályrendszer szabályait is törli.

A szabályrendszerek törlése kapcsán meg kell jegyeznünk, hogy az `iptables` programmal új szabályrendszereket is létrehozhatunk (erre a későbbiekben még visszatérünk). Ha a `--flush` kapcsoló után nem adunk meg nevet, a program ezeket az általunk létrehozott szabályrendszereket is törli, ezért a három beépített szabályrendszer egyenkénti törlése nem teljesen egyenértékű a név nélkül kiadott parancs hatásával.

69. példa. A következő példa tiltásra állítja a három szabályrendszer alapszabályát, majd törli az összes szabályt:

```

1 #
2 # Mindhárom szabályrendszer alapszabálya tiltó.
3 #
4 iptables --policy FORWARD REJECT
5 iptables --policy INPUT REJECT
6 iptables --policy OUTPUT REJECT
7 #
8 # minden szabály törlése
9 #
10 iptables --flush

```

Új szabály elhelyezése

A szabályrendszerekbe a következő kapcsolók segítségével helyezhetünk el új szabályokat:

`iptables --append név szabály` A szabályrendszerhez a `--append` vagy `-A` kapcsoló segítségével fűzhetünk új szabályt. A kapcsolót használva a szabályt a szabályrendszer végén, az utolsó szabály után hozzuk létre.

A parancsban található név a szabályrendszer neve (`INPUT`, `OUTPUT` vagy `FORWARD`), a szabály pedig a szabály pontos leírása, amelyre a későbbiekben még visszatérünk.

Név	Jelentés
icmp-net-unreachable	a hálózat nem elérhető
icmp-host-unreachable	a számítógép nem elérhető
icmp-port-unreachable	a kapu nem elérhető
icmp-proto-unreachable	a szabvány nem elérhető
icmp-net-prohibited	a hálózat tiltott
icmp-host-prohibited	a számítógép tiltott

7.1. táblázat. A küldhető ICMP csomagtípusok

`iptables --insert név [szám]szabály` A `--insert` vagy `-I` kapcsolóval tetszőleges helyre szúrhatunk be szabályt a szabályrendszerbe. A kapcsoló után megadott szám az elhelyezett új szabály sorszáma, ahol a szabályok számozása 1-től indul. Ha nem adunk meg sorszámot, a szabály a szabályrendszer elejére kerül.

`iptables --replace név [szám]szabály` A `--replace` vagy `-R` kapcsolóval szintén tetszőleges helyen helyezhetünk el szabályt, de az ott található szabályt felülírva.

Amikor új szabályt helyezünk el valamelyik szabályrendszerben, le kell írnunk magát a szabályt, amely megadja, hogy az milyen csomag esetében használható, és milyen sorsot ír elő az adatcsomag számára. A szabály megadására a következő kapcsolókat használhatjuk:

`--protocol [!] protokoll` A `--protocol` vagy `-p` kapcsolóval a szabályban szereplő protokollet. A *protokoll* helyén a `tcp`, `udp`, `icmp` és az `all` (minden) kulcsszó, valamint a `/etc/protocols` állományban található összes többi protokollnév állhat.

A protokoll előtt elhelyezhetjük a `!` jelet a tagadás jelzésére. Ekkor a szabály azokra a csomagokra lesz érvényes, amelyek nem az adott protokoll szerint épülnek fel.

`--source [!] cím[/maszk]` A `--source` vagy `-s` kapcsolóval megadhatjuk a szabályban szereplő forrás IP címét. Ha a cím után megadjuk az alhálózati maszkot tízes számrendszerben vagy a hálózati címben található bitek számával, a szabály több címre is illeszkedhet.

Ha a cím előtt elhelyezzük a `!` jelet a tagadás jelzéseként, a cím illesztését az ellentettjére változtathatjuk.

`--destination [!] cím[/maszk]` A `--destination` vagy `-d` kapcsolóval megadhatjuk a szabályban található cél IP címét a már bemutatott forrás IP címhez hasonló szabályok alapján.

--jump sors A --jump vagy -j kapcsolóval megadhatjuk a szabályra illeszkedő adatcsomagok sorsát.

A kapcsoló után megadható kulcsszavak közül a legfontosabbak a következők:

ACCEPT Az illeszkedő csomagot a szabály alapján a rendszermag elfogadja.

DROP A csomagot a szabály alapján a rendszermag eldobja. Erről a csomagot küldő folyamat vagy számítógép nem kap értesítést.

Az ellenállomásnak persze egy idő után gyanússá válhat, hogy a csomagjait eldobáljuk, de ez viszonylag hosszú időt is igénybe vehet, és felesleges hálózati terhelést jelenthet. Ha azt szeretnénk, hogy az ellenállomás értesüljön az adatcsomag elutasításáról, használjuk a következő kulcsszót.

REJECT Az illeszkedő csomagot a rendszermag eldobja, és erről – egy ICMP csomagban – értesíti az ellenállomást.

Azt, hogy a Linux rendszermag milyen típusú ICMP csomaggal jelezze az adatcsomag elutasítását, a --reject-with kapcsolóval adhatjuk meg, amelyet az 7.1. táblázat valamelyik kulcsszava követ.

Amint látjuk, a szabályokban körülbelül ugyanolyan sorsot jelölhetünk ki a csomag számára, mint a szabályrendszer alapszabályában. Az iptables bővíteként még néhány más sors is a rendelkezésünkre áll, amelyekre a későbbiekben visszatérünk⁷.

--in-interface [!] név A --in-interface vagy -i kapcsolóval olyan szabályokat készíthetünk, amelyek csak azokra a csomagokra illeszthetők, amelyek az adott nevű hálózati csatolón érkeztek a számítógépre.

Ezt a kapcsolót csak az INPUT és a FORWARD szabályrendszeren belül elhelyezett szabályok esetében használhatjuk.

A ! karakter a tagadás jelzésére használható.

--out-interface [!] név A --out-interface vagy -o kapcsolóval megadhatjuk a kimenő hálózati csatoló nevét (például eth0). A szabály csak azokra a csomagokra illeszkedik, amelyek ezen a hálózati csatolón keresztül kísérlik meg elhagyni a számítógépet.

Ezt a kapcsolót csak a FORWARD és az OUTPUT szabályrendszerben használhatjuk.

A ! karakterrel a tagadást jelezhetjük, a segítségével olyan szabályt készíthetünk, amely nem illeszthető a csomagokra, amelyek az adott hálózati csatolón keresztül hagyják el a számítógépet.

⁷Valójában már a REJECT sors is bővíttés, ráadásul magyar szakember, KADLECSIK JÓZSEF alkotása.

Név	Jelentés
destination-unreachable	a cél nem elérhető
echo-request	kapcsolatellenőrző válasz kérése
echo-reply	kapcsolatellenőrző válaszcsomag
fragmentation-needed	csomagdarabolás szükséges
host-unreachable	a számítógép nem elérhető
network-unreachable	a hálózat nem elérhető
port-unreachable	a logikai kapu nem elérhető
ttl-exceeded	a csomag végelgyengülésben meghalt
timestamp-request	óra értékének lekérdezése
timestamp-reply	óra értékének küldése

7.2. táblázat. A legfontosabb ICMP csomagtípusok

- [!] --fragment A --fragment vagy -f kapcsolóval megadhatjuk, mi történjék a csomagtöredékekkel. A ! itt is a tagadás jele, a segítségével megadhatjuk, mi történjék azokkal a csomagokkal, amelyek nem csomagtöredékek.
- A csomagtöredékekben nem található meg a csomag legfontosabb tulajdon-ságait leíró fejléc, ezért azok tulajdonságai nem kideríthetők. Mivel a cso-magtöredékek igen kifinomult támadások eszközei lehetnek – és kifinomult támadóprogramokat bárki képes futtatni –, a csomagtöredékeket általában kiszúrjük.

70. példa. Egy drasztikus lépéssel tiltsuk ki az összes számítógépet a gépünkéről, amely nem egy adott hálózatba tartozik!

```
1 iptables --insert INPUT --source ! 172.17.1.1/24 --jump DROP
```

Az elhelyezett szabály a INPUT szabályrendszer elejére kerül, aminek két fontos következménye is van. Egyrészről nyilvánvaló, hogy a számítógéünk esetleg hajlandó csomagokat továbbítani más számítógépek számára is, hiszen a FORWARD szabályrendszert nem módosítottuk, másrészről viszont mindegy, hogy milyen szabályok voltak az INPUT szabályrendszerben, hiszen ez a szabály kerül első helyre, és ez minden csomagot eldob.

Ha a szabály elhelyezésekor megadjuk, hogy a szabály az ICMP alapprotokollú csomagokra vonatkozik, az iptables betölti a icmp modult, amely elérhetővé teszi a következő kapcsolót:

--icmp-type [!] név A kapcsolóval megadhatjuk a szabályban szereplő ICMP csomagtípust. A ! a tagadás jelzsére szolgál, a segítségével a szabályban nem jelzett összes ICMP csomagtípusra hivatkozhatunk.

Az ICMP csomagtípusok neveit a iptables -p icmp -h kapcsolójával írat-hatjuk ki. A legfontosabb csomagtípusokat a 7.2. táblázat tartalmazza.

71. példa. Tiltsuk le a számítógépünkön az óra lekérdezésére használt ICMP cso-magok fogadását! Ez a lépés különösen hasznos lehet akkor, ha nem akarjuk, hogy bárki megtudhassa, mi a számítógép beépített órájának aktuális állása, mert az időtől is függő titkosítórendszert használunk.

```
1  iptables --append INPUT --protocol icmp \
2          --icmp-type timestamp-request \
3          --jump DROP
```

A példát kissé átalakítva készíthetünk olyan szabályt, amely a hálózatellenőrzésre használt echo kérést közvetítő csomagokat tiltja. Így viszonylag jól „elrejtőzhetünk” a hálózaton, hiszen a számítógépünk nem válaszol a ping programmal kül-dött kérésekre.

Tiltsuk le az echo kéréseket, de hagyjuk érintetlenül őket egy bizonyos címtar-tomány számítógépe számára!

```
1  iptables --append INPUT --protocol icmp \
2          --icmp-type echo-request \
3          --source ! 172.17.1.1/255.255.0.0 \
4          --jump DROP
```

Mivel a számítógépünk a 127.0.0.1 címen éri el saját magát, most szerencsétlen módon számára is sikerült le tiltanunk ezeket a csomagokat. Egy újabb szabály elhelyezésével engedélyezhetjük az összes ICMP csomag fogadását:

```
1  iptables --insert INPUT --protocol icmp \
2          --source 127.0.0.0/255.0.0.0 --jump ACCEPT
```

Most a szabályrendszerünk a következő:

```
$ iptables -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT    icmp --  127.0.0.0/8      anywhere
DROP      icmp --  anywhere            anywhere      icmp timestamp-request
DROP      icmp --  !172.17.0.0/16    anywhere      icmp echo-request
$
```

A szabályrendszer viszonylag egyszerű, de a céljainknak esetleg megfelelhet.

Ha a szabály elhelyezésekor megadjuk, hogy a szabály az UDP alapprototokollú csomagokra vonatkozik, az iptables betölti az udp modult, amely elérhetővé teszi a következő kapcsolókat:

--source-port [!] kapu[:kapu] A kifejezés segítségével megadhatjuk a for-rásként szereplő logikai kapu számát. A kapu száma helyett használhatjuk a /etc/services állományban szereplő neveket is.

Név	Jelentés
SYN	kapcsolatkérés, sorszámok egyeztetése
ACK	visszaigazolás, nyugtázás
FIN	kapcsolat zárása, befejezés
RST	FIXME
URG	sürgős
PSH	FIXME
ALL	minden kapcsoló
NONE	egyetlen kapcsoló sem

7.3. táblázat. A TCP kapcsolók elnevezése

Ha a kapu száma után : jellel elválasztva egy újabb számot – esetleg nevet – is megadunk, a logikai kapu egész tartományát elhelyezhetjük egy szabályban. Az első számot elhagyhatjuk (például :100), annak alapértelmezett értéke 0. A második számot szintén elhagyhatjuk (például 100:), ennek alapértelmezett értéke 65535.

--destination-port [!] kapu[:kapu] Ezzel a kifejezéssel a forrás logikai kapu értékének mintájára megadhatjuk a cél logikai kaput is.

Ha a szabály elhelyezésekor megadjuk, hogy a szabály a TCP alapprototollú csomagokra vonatkozik a szabály, az iptables betölti a tcp modult és elérhetővé teszi a következő kapcsolókat:

--source-port [!] kapu[:kapu] A kifejezés segítségével az UDP prototollú csomagokhoz hasonló módon megadhatjuk a TCP csomagokban forrásként megadott logikai kapu értékét.

--destination-port [!] kapu[:kapu] A címzett logikai kapu megadása a már ismert módon történik.

--tcp-flags [!] maszk be A csomagban található TCP kapcsolók vizsgálatát állíthatjuk be ezzel a kifejezéssel. A kifejezésben található *maszk* lista megadja, hogy mely biteket vizsgáljuk, míg a *be* lista felsorolja, hogy mely biteknek kell bekapcsoltnak lenniük. Azoknak a biteknek, amelyeket a *maszk* listában felsoroltunk, de a *be* listában nem, kikapcsolt állapotban kell lenniük ahhoz, hogy a csomag illeszkedjen a szabályra.

Mind a *maszk*, mind pedig a *be* listában kulcsszavakat adhatunk meg vesszővel elválasztott listában. A bitek jelzésére használható kulcsszavakat a 7.3. táblázatban láthatjuk.

[!] --syn A kapcsoló segítségével megadhatjuk, hogy mi történjék azokkal a TCP csomagokkal, amelyekben a SYN bit be van kapcsolva, de az ACK

bit nincs. Ezzel a kapcsolóval a TCP kapcsolatot „egyirányúsíthatjuk”, ami roppant hasznos – bár kissé önző – módja a védekezésnek.

Amint láttuk, azok a TCP csomagok, amelyekben a SYN bit be van kapcsolva, de az ACK bit nincs, kapcsolatkiépítést kérnek. Ha ezeket a csomagokat eldobjuk, külső számítógép nem tudja felvenni a számítógépünkkel – vagy az általa védett hálózattal – a kapcsolatot TCP protokollt használva.

Ha viszont eldobnánk azokat a csomagokat is, amelyekben mind a SYN, mind pedig az ACK bit be van kapcsolva, akkor a saját számítógépünk sem lenne képes kapcsolatot kezdeményezni, hiszen láttuk, hogy a kapcsolatkérő csomagokra adott első válaszcsomagban e két bit be van kapcsolva.

Ha tehát csak használni akarjuk az Internetet, de nem akarjuk, hogy az Internet is használja a számítógépünket, mindenéppen javasolható a -syn kapcsolóval készített szabályok használata.

72. példa. Tiltsuk, az összes TCP csomag fogadását, kizárolag az ssh kapura érkező csomagokat engedélyezzük!

```
1 iptables --append INPUT --protocol tcp \
2   --destination-port ! ssh --jump REJECT
```

A példában szereplő szabály hatása az INPUT szabályrendszerben már megtalálható egyéb szabályoktól függ, de ha más szabály nincs a szabályrendszerben, akkor minden csomagot visszautasít, amely nem az ssh logikai kapura érkezik.

Szabály törlése

A következő kifejezések segítségével egy-egy szabályt törölhetünk a szabályrendserekből:

`iptables --delete név szám` A szabályrendszer adott számú szabályának törlését a --delete vagy -D kapcsolóval végezhetjük el. A kapcsoló után megadott név a szabályrendszer neve, a szám pedig a szabály sorszáma. A szabályok számozása a 1-től kezdődik.

`iptables --append név szabály` A szabályok törlését a szabály leírásával is kérhetjük. Ilyen esetben a programot az új szabály létrehozásánál már látott kapcsolókkal használhatjuk.

Összefoglalás

A következő példa segítségével összefoglaljuk az `iptables` eddig bemutatott képességeit. A példa egy túzfalszabályokat létrehozó egyszerű héjprogramot mutat be, amely egy munkaállomás védelmét látja el. Érdemes azonban felhívni a figyelmet arra, hogy ez a példa inkább csak az ismeretek összefoglalására alkalmas, oly egyszerű és oly kevés elemet tartalmaz, hogy komoly munkára nem használható.

73. példa. A következő példa egy igen egyszerű – monhatnánk primitív – tűzfalat valósít meg munkaállomások számára.

```
1 #! /bin/bash                                         /root/bin/firewall-1.sh
2 #
3 # „Jobb mint a semmi” tűzfal szabályainak felépítése.
4 #
5 #
6 #
7 # Nem továbbítunk csomagokat senki számára, a FORWARD
8 # szabályrendszer minden csomagot eldob.
9 #
10 iptables --policy FORWARD DROP
11 iptables --flush FORWARD
12 #
13 #
14 # Az INPUT szabályrendszer felépítése. Előbb minden csomag fo-
15 # gadását letiltjuk, majd néhány csomag fogadását engedélyezzük.
16 #
17 iptables --policy INPUT DROP
18 iptables --flush INPUT
19 #
20 # Mindent fogadunk az lo csatolón a helyi számítógépről.
21 iptables --append INPUT --in-interface lo --jump ACCEPT
22 #
23 # minden ICMP csomagot fogadunk, mert fontos üzenet lehet
24 # bennük.
25 iptables --append INPUT --protocol icmp --jump ACCEPT
26 #
27 # A felső UDP és TCP kapukon minden fogadunk. Ez meglehetősen
28 # szerencsétlen döntés, de csak az eddig tárgyalt eszközökkel
29 # használhatjuk, és egyszerűségre törekszünk.
30 iptables --append INPUT --protocol udp \
31             --destination-port 1024: \
32             --jump ACCEPT
33 iptables --append INPUT --protocol tcp \
34             ! --syn \
35             --destination-port 1024: \
36             --jump ACCEPT \
37 #
38 # A saját hálózatunkon fogadjuk az SSH kéréseket.
39 iptables --append INPUT --protocol tcp \
40             --destination-port ssh \
41             --source 172.17.1.0/24 \
```

```

42          --jump ACCEPT
43
44 # Mindenhonnan fogadjuk a HTTP kéréseket. Ez egy munkaállomás
45 # esetében szokatlan, de a dokumentáció böngészéséhez jól jön.
46 iptables --append INPUT --protocol tcp \
47             --destination-port http \
48             --jump ACCEPT
49
50
51 #
52 # Az OUTPUT szabályrendszer. Majdnem minden engedünk, csak
53 # néhány apróságot tiltunk.
54 #
55 iptables --policy OUTPUT ACCEPT
56 iptables --flush OUTPUT
57
58 # Engedélyezzük az echo csomagokra adott választ a helyi
59 # számítógép felé, majd azonnal letiltjuk minden, a háló-
60 # zatunkon kívüli számítógép számára.
61 iptables --append OUTPUT --protocol icmp \
62             --icmp-type echo-reply \
63             --out-interface lo \
64             --jump ACCEPT
65
66 iptables --append OUTPUT --protocol icmp \
67             --icmp-type echo-reply \
68             --destination ! 172.17.1.0/24 \
69             --jump DROP
70
71 # A sajátgépen kívül mindenki letiltjuk az idő elárulását.
72 iptables --append OUTPUT --protocol icmp \
73             --icmp-type timestamp-reply \
74             --out-interface ! lo \
75             --jump DROP

```

A program áttanulmányozása során láthatjuk, hogy igen egyszerű szabályrendserről van szó, ami nem elég kifinomult ugyan a munkaállomás védelmére, de mindenkorban hozzájárul a biztonság növeléséhez.

A program által létrehozott szabályrendszer részletes listája a 201. oldalon a 67. példában található.

7.2.2. A csomag előtörténete

Amint láttuk, a SYN bit vizsgálatával megtudhattuk, hogy az adott TCP csomag új kapcsolat részeként, vagy egy már felépített kapcsolat részeként jutott a számítógéünkre. Ez roppant hasznos lehetőség, hiszen általában azt szeretnénk elérni, hogy a számítógéünk által már kiépített kapcsolatok csomagjait a csomagszűrő átengedje, viszont ne tegye lehetővé a külső számítógépek kapcsolatfelépítő csomagjainak fogadását. A SYN bit vizsgálatán alapuló módszernek azonban két komoly hiányossága is van:

- Kizárálag a TCP adatcsomagokban található SYN bit, ezért ezzel a módszerrel nem kezelhetők például az UDP csomagok.
- Léteznek olyan hálózati protokollok, amelyek lehetővé teszik, hogy a kiszolgáló új kapcsolatot építve teljesítse az ügyfél kéréseit.

Ilyen protokoll például az FTP (aktív FTP), ahol a kiszolgáló az ügyfél által kért adatokat egy általa nyitott új TCP kapcsolat segítségével küldi el. Nyilvánvaló, hogy az ilyen kapcsolatok felépítését a csomagszűrő túzfalnak engedélyeznie kell, nem tilthatja tehát az összes kapcsolatkezdeményezést tartalmazó csomag fogadását.

Mindkét problémánkat megoldhatjuk, ha a csomagszűrő túzfalat képessé tessük arra, hogy emlékezzen a rajta áthaladt csomagokra, és számítson a válasz-csomagokra, amelyeket át kell engednie. Az ilyen túzfal képes arra, hogy megjegyezze, mit kértünk a külvilágktól, és a beérkezett csomagokról ez alapján el tudja döntenı, hogy azok a mi kérésünkre érkezett válaszcsomagok-e.

A következőkben először bemutatjuk, hogyan lehet programmodulokat betölteni az `iptables` programrendszerbe, majd megvizsgáljuk, hogyan lehet a csomagok előtörténetét is figyelembe vevő szabályokat készíteni.

A modulok betöltése

Az `iptables` – és maga a `netfilter` – igen átgondolt tervezéssel készült (ami nem is meglepő, hiszen ez a Linux rendszermag csomagszűrő rendszerének harmadik generációja), nagyon rugalmas beállítási lehetőségekkel. Az `iptables`/`netfilter` programrendszer a rugalmas beállítási lehetőségeket annak köszönheti, hogy képes külső modulokat betölteni és a Linux rendszermag hálózatkezelését a modulok segítségével módosítani.

Modulok betöltését az `iptables` következő kapcsolójával kérhetjük:

`--match név` A kapcsoló segítségével az `iptables` szolgáltatásait bővíti modult tölthetünk be. A kapcsoló hatására a rendszermag a megfelelő magmodult betölti, az `iptables` pedig elérhetővé teszi a modul beállítására használható kapcsolókat.

A kifejezésben található `név` a betöltendő modul neve. A `man iptables` parancs segítségével sok hasznos modulról kaphatunk információt.

Az `iptables -m` kapcsolójával betölthető modulok közül a legfontosabbak a következők:

`addrtype` A modul segítségével a csomagokat az IP cím típusa (körüzenet, több számítógép számára küldött üzenet stb.) alapján szűrhetjük.

`condition` A csomag segítségével beállítófelületet készíthetünk a szabályrendszer szerek számára a /proc/látszólagos állományok segítségével.

`connmark` A modul segítségével meg lehet vizsgálni azokat a jelzőbiteket, amelyeket az erre a célra szolgáló CONNMARK sors segítségével beállítottunk. A modul tehát segítséget ad ahhoz, hogy a szabályrendszer egyik szabálya által megjelölt csomagokat egy másik szabály segítségével kiszűrjük.

`connrate` A modul segítségével annak az adatkapcsolatnak a pillanatnyi sávszélesség-terhelését vizsgálhatjuk, amelyhez az adott csomag kapcsolódik.

`dstlimit` A modul segítségével korlátot állíthatunk be az egy címre vagy egy logikai kapura másodpercenként küldhető csomagokra.

`icmp` A modul segítségével az ICMP csomagokat típusuk szerint szűrhetjük. Ezt a modult az `iptables` automatikusan betölti, ha a szabályban megadjuk az ICMP protokollt.

`iprange` A programmodul segítségével tetszőleges IP tartományok alapján szűrhetjük a csomagokat.

`length` A programcsomagok méretük alapján szűrhetők ezzel a modullal.

`limit` E modul segítségével a szabályra adott idő alatt illeszkedő csomagok számát korlátozhatjuk.

`mac` Ha betöljük ezt a modult, az Ethernet cím alapján szűrhetjük a csomagokat.

`mport` A modul segítségével egy szabályban több logikai kapu címét is elhelyezhetjük, hogy egyszerűsíthessük – és gyorsíthassuk – a csomagszűrést.

`nth` A modul segítségével minden n -edik csomagot szűrhetünk (valamint próbára tehetjük a fantáziánkat is, miközben megkísérlünk olyan helyzetet kigondolni, amely ennek a modulnak a használatát szükségessé teszi).

`owner` A modul segítségével a helyi számítógépen létrehozott adatcsomagokhoz tartozó felhasználót és programot azonosíthatjuk, illetve felhasználói azonosító, csoportazonosító és programnév alapján is szűrhetünk.

`random` Ezzel a modullal véletlenszerűen szűrhetünk, megadva, hogy a csomagok hány százaléka feleljen meg a szabálynak.

state A modul segítségével megtudhatjuk, hogy az adatcsomag kiépített kapcsolat, vagy új kapcsolat részeként jelent-e meg a Linux rendszermagban.

tcp A modul segítségével TCP protokollú csomagok néhány fontos tulajdonságát vizsgálhatjuk. A modult az `iptables` automatikusan betölti, ha a szabályban előírjuk a TCP protokoll használatát.

time A modul segítségével a szabályok alkalmazását időhöz köthetjük. Kitűnően használható ez az eszköz arra, hogy a szolgáltatásokat csak bizonyos napszakban tegyük elérhetővé.

ttl Az adatcsomagban található TTL értékét vizsgálhatjuk, ha betöljük ezt a modult.

udp Az UDP protokoll néhány fontos elemét vizsgálhatjuk e modullal. A modult az `iptables` automatikusan betölti, ha a szabályban az UDP protokoll használatát írjuk elő.

A felsorolt modulok közül a legfontosabbakra a későbbieken visszatérünk, hogy részletesebben is bemutassuk őket.

Az előtörténetet kezelő modul

Ha az `iptables` futtatásakor a `state` modult betöltöttük a `--match state` kapcsóval, használhatjuk a következő kifejezést:

--state állapot A kapcsoló segítségével olyan szabályt készíthetünk, amely figyelembe veszi az adatcsomag előtörténetét. A kifejezésben szereplő `állapot` megadja, hogy a csomag milyen kapcsolat része. Az `állapot` helyén a következő kulcsszavak állhatnak:

INVALID A kulcsszó jelzi, hogy a Linux rendszermag nem tudta megállapítani, hogy a csomagnak mi az előtörténete.

ESTABLISHED Ez a kulcsszó jelzi, hogy a csomag egy olyan kapcsolat részét alkotja, amit a kapcsolat két végpontján található számítógépek már felépítettek, azaz a kapcsolat részeként már minden irányba haladtak adatcsomagok.

NEW Ez a kulcsszó jelzi, hogy a csomag egy új kapcsolat részét alkotja.

RELATED Ez a kulcsszó jelzi, hogy a csomag új kapcsolat részét alkotja, amelyet azonban egy már kialakított kapcsolat miatt kellett létrehozni.

Az Interneten gyakran használunk olyan kiszolgálókat, amelyek működésük során újabb kapcsolatot építenek fel, hogy kiszolgálják az ügyfélprogram kérést. Az ilyen alkalmazásoknak szükségük van rá, hogy a csomagszűrő túzfal felismerje a kiépítendő új kapcsolat okát és engedélyezze az adatforgalmat.

A következő példa ezt az igen fontos és hasznos eszközt használja fel csomagszűrő tűzfal készítésére.

74. példa. A következő példaprogram egy némi képp kifinomultabb tűzfalat valósít meg a csomagok előtörtenetének figyelembe vételevel.

```
1 #! /bin/bash
2 #
3 # „Kicsit jobb” tűzfal szabályrendszerének betöltése.
4 #
5
6 # A nyilvánosan elérhető szolgáltatások listája
7 SZOLG="ssh https"
8
9 # Csomagokat senkinek sem továbbítunk.
10 echo "A FORWARD szabályrendszer."
11 iptables --policy FORWARD DROP
12 iptables --flush FORWARD
13 iptables --append FORWARD --jump REJECT \
               --reject-with icmp-net-unreachable
14
15 #
16 #
17 # A kiépített kapcsolatokat beengedjük, egyébként csak
18 # néhány szolgáltatást nyújtunk.
19 #
20 echo "Az INPUT szabályrendszer"
21 iptables --policy INPUT DROP
22 iptables --flush INPUT
23
24 # A saját gépünkről minden elfogadunk.
25 iptables --append INPUT --in-interface lo --jump ACCEPT
26
27 # Az ICMP csomagokat elfogadjuk.
28 iptables --append INPUT --protocol icmp --jump ACCEPT
29
30 # A már kiépített kapcsolatok csomagjait elfogadjuk.
31 iptables --append INPUT --match state \
               --state ESTABLISHED \
               --jump ACCEPT
32
33 iptables --append INPUT --match state \
               --state RELATED \
               --jump ACCEPT
34
35
36
37
38 # Bizonyos szolgáltatásokat elérhetővé teszünk.
```

```
39 | echo -n " Szolgáltatás engedélyezése: "
40 | for S in $SZOLG; do
41 |   echo -n "$S "
42 |   iptables --append INPUT --protocol tcp \
43 |             --destination-port $S \
44 |             --match state \
45 |             --state NEW \
46 |             --jump ACCEPT
47 | done
48 | echo
49 |
50 | # Egyébként minden udvariasan visszautasítunk.
51 | iptables --append INPUT --jump REJECT \
52 |             --reject-with icmp-host-unreachable
53 |
54 | # Nem engedjük ki az echo és time ICMP csomagokat, hogy
55 | # egy kissé elrejtőzzünk.
56 | echo "Az OUTPUT szabályrendszer"
57 | iptables --policy OUTPUT ACCEPT
58 | iptables --flush OUTPUT
59 |
60 | iptables --append OUTPUT --protocol icmp \
61 |             --icmp-type echo-reply \
62 |             --out-interface lo \
63 |             --jump ACCEPT
64 |
65 | iptables --append OUTPUT --protocol icmp \
66 |             --icmp-type echo-reply \
67 |             --jump DROP
68 |
69 | iptables --append OUTPUT --protocol icmp \
70 |             --icmp-type timestamp-reply \
71 |             --out-interface ! lo \
72 |             --jump DROP
```

A program a magyarázatok és az előzőekben tárgyalt ismeretek alapján viszonylag könnyen megérthető, érdemes azonban néhány apróságot külön kiemelni.

A szabályrendszerek alapszabálya nem lehet REJECT, a tűzfalunk viszont a REJECT sors segítségével ICMP válaszüzeneteket küld. Ezt úgy érhetjük el, hogy a szabályrendszer utolsó szabályaként minden csomagot visszautasító, REJECT bejegyzésű szabályt használunk. Kivéve természetesen az OUTPUT szabályrendszert, ahol kimenő ICMP csomagokat szűrünk.

Figyeljük meg, hogy a program lefutásakor előbb gondosan tiltja a forgalmat az alapszabállyal és csak utána törli a szabályrendszer szabályait. Ha tehát tiltó

jellegű szabályok voltak a szabályrendszerben, a program egy pillanatra sem lazít a rendszabályokon. Megint kivétel az OUTPUT szabályrendszer, ahol nem zavar-tattuk magunkat attól, ha esetleg néhány ICMP csomag kimegy a hálózatra.

A szolgáltatások engedélyezésekor egy ciklust használtunk, amely a program elején értéket kapó változóból olvassa ki az engedélyezendő szolgáltatásokat. Ilyen egyszerű kényelmi szolgáltatást sok túzfalszabályokat felépítő héjprogram használ.

A túzfalszabályokat egy szimulált támadás és néhány perc használat után az iptables a következő formában írta ki (a sorokat helytakarékkosság érdekében kissé módosítottuk, a túlságosan hosszú sorokat a \ jellel jelöltük):

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
    0     0 ACCEPT all  --  lo  any anywhere anywhere
   15  1735 ACCEPT icmp --  any any anywhere anywhere
  749 78135 ACCEPT all  --  any any anywhere anywhere state ESTABLISHED
    0     0 ACCEPT all  --  any any anywhere anywhere state RELATED
   90  4840 ACCEPT tcp  --  any any anywhere anywhere tcp dpt:ssh state N\W
      2    120 ACCEPT tcp  --  any any anywhere anywhere tcp dpt:https state\NEW
  5076  317K REJECT all  --  any any anywhere anywhere reject-with icmp-ho\st-unreachable

Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
    0     0 REJECT all  --  any any anywhere anywhere reject-with icmp-ne\t-unreachable

Chain OUTPUT (policy ACCEPT 1536 packets, 193K bytes)
pkts bytes target prot opt in out source destination
    0     0 ACCEPT icmp --  any lo  anywhere anywhere icmp echo-reply
   11  1584 DROP   icmp --  any any anywhere anywhere icmp echo-reply
    2    80 DROP   icmp --  any !lo anywhere anywhere icmp timestamp-reply
```

Figyeljük meg a sorok elején található számokat, amelyek jól jelzik, hogy az egyes szabályok alkalmazására hányszor került sor! Egy néhány másodperces támadás során a Linux rendszermag több mint 5000 adatcsomagot dobott el anélkül, hogy ez a számítógépet használó személyt megzavarta volna a munkában, anélkül hogy egyetlen színes ablak is megjelent volna a képernyón, amelyben a túzfal azt tudakolja, hogy mit tegyen.

7.2.3. Saját szabályrendszerek készítése

He elég sokáig csiholjuk a csomagszűrő szabályokat, a szabályrendszerekben olyan sok szabály lesz, hogy nem tudunk eligazodni a saját művünkön. A sza-

bályrendszerek egyszerűsítésének érdekében saját szabályrendszereket hozhatunk létre⁸.

A saját szabályrendszereket úgy használhatjuk, mint az alprogramokat (függvények, eljárások) a programozásban: a szabályrendszert egyszer kell létrehoznunk, és több helyen is hivatkozhatunk rájuk. A saját szabályrendszerek létrehozásához és használatához két új csomagsorsorot is meg kell ismernünk:

név Az általunk létrehozott szabályrendszerek nevét más szabályrendszerekben sorsként felhasználhatjuk. Ilyen esetben a Linux rendszermag a csomag vizsgálatát az adott szabályrendszerben, az ott található első szabállyal folytatja.

RETURN A saját szabályrendszerben sorsként használhatjuk a RETURN kulcsszót. Ez a kulcsszó azt jelenti, hogy a szabályrendszer szabályainak vizsgálatát be kell fejezni, vissza kell térti a hívó szabályrendszer soron következő szabályához.

A következő példaprogram saját szabályrendszer használatát mutatja be a hamisított IP című csomagok szűrésén.

A hamis forráscímű csomagok küldése (*IP spoofing*) a csomagszűrő tűzfalak kijátszásának egyszerű módszere, lényege, hogy a támadó olyan IP csomagokat küld, amelyek feladójá „megbízható”, amelyeket a csomagszűrő szabályok alapján a tűzfal elfogad. Az ilyen támadás ellen könnyen védekezhetünk, megtehetjük például, hogy nem csak a feladó IP címét vizsgáljuk, hanem azt is, hogy a csomag melyik hálózati csatolón érkezett. A példánk egyszerűen eldobja a csomagokat, amelyek érkezési irányá nincs összhangban az állítólagos feladó címével.

75. példa. A következő példa bemutatja, hogyan használhatunk saját szabályrendszert hamisított IP című csomagok kiszűrésére. A csomagszűrő szabályok lista a következő (a sorokat helytakarékkosság céljából kissé módosítottuk):

```
$ iptables -L -v -n
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in     out source          destination
      0      0 SPOOFP all   --  *       *    0.0.0.0/0      0.0.0.0/0

Chain INPUT (policy ACCEPT 239 packets, 23675 bytes)
pkts bytes target prot opt in     out source          destination
  195 19556 SPOOFP all   --  *       *    0.0.0.0/0      0.0.0.0/0

Chain OUTPUT (policy ACCEPT 50585 packets, 71M bytes)
pkts bytes target prot opt in     out source          destination

Chain SPOOFP (2 references)
pkts bytes target prot opt in     out source          destination
      0      0 DROP   all   --  eth0  *    !172.17.0.0/16 0.0.0.0/0
```

⁸Saját szabályrendszer létrehozásával a csomagszűrő tűzfal működése gyorsítható is (a lektor).

```

0      0 DROP    all   --  !eth0 *   172.17.0.0/16  0.0.0.0/0
0      0 DROP    all   --  !lo   *   127.0.0.0/8   0.0.0.0/0
195 19556 RETURN all   --  *     *   0.0.0.0/0   0.0.0.0/0
$
```

Könnyen felismerhetjük a példában szereplő szabályrendszerek közt az új szabályrendszert, amely a SPOOFP nevet kapta. A SPOOFP a hamisított című adatcsomagokat próbálja meg kiszűrni. A szabályrendszer eldobja az összes csomagot, amely a 127.0.0.0/8 címtartományból érkezik, de nem az lo csatolón keresztül kaptuk, illetve eldobja az összes csomagot, amely a 172.17.0.0/16 címtartományból, de nem az eth0 csatolón érkezett. Vegyük észre, hogy a szabályok feltételezik, hogy az eth0 csatolón csak egy címtartományt lehet elérni!

A SPOOFP szabály hívását az INPUT és a FORWARD szabályrendszerben is meg-találhatjuk, a szabályokat két helyen is felhasználtuk. A SPOOFP szabályrendszer utolsó szabálya a RETURN sorsor tartalmazza, minden csomagra illeszkedik, így visszaadja a csomagot a hívó szabályrendszernek további vizsgálatra.

A példában szereplő szabályokat a 220. oldalon található 76. példa programja hozta létre.

Szabályrendszerek létrehozása

Szabályrendszerek létrehozására a következő kifejezést használhatjuk:

`iptables --new-chain név` A kifejezés segítségével új szabályrendszert hozhatunk létre. A kifejezésben szereplő név a létrehozandó új szabályrendszer neve.

Szabályrendszerek törlése

A szabályrendszereket a következő kifejezés segítségével törölhetjük:

`iptables --delete-chain [név]` A kifejezésben szereplő `--delete-chain` vagy `-X` kapcsolóval szabályrendszereket törölhetünk. Az iptables csak akkor képes törölni a szabályrendszert, ha más szabályrendszerekben nincs rá hivatkozás, előbb tehát a hivatkozó szabályokat kell törölnünk a többi szabályrendszerből.

A kifejezésben szereplő név a törlendő szabályrendszer neve. Ha nem adjuk meg a nevet, az iptables megkíséri törölni a beépített szabályrendszereken kívül az összes szabályrendszert.

Összefoglalás

A következő példaprogram szabályrendszer létrehozását mutatja be:

76. példa. Hamisított IP címről érkező csomagok szűrésére használhatunk a következő példaprogramhoz hasonló sorokat:

```
1 #! /bin/bash
2 #
3 # Saját szabályrendszer használata
4 #
5 $BELSO_IP="172.17.0.0/16"
6 $BELSO_NIC="eth0"
7
8 iptables --new-chain SPOOFP
9
10 iptables --append SPOOFP \
11     --source ! $BELSO_IP \
12     --in-interface $BELSO_NIC \
13     --jump DROP
14
15 iptables --append SPOOFP \
16     --source $BELSO_IP \
17     --in-interface ! $BELSO_NIC \
18     --jump DROP
19
20 iptables --append SPOOFP \
21     --source 127.0.0.0/8 \
22     --in-interface ! lo \
23     --jump DROP
24
25 iptables --append SPOOFP \
26     --jump RETURN
27
28 iptables --insert INPUT \
29     --jump SPOOFP
30
31 iptables --insert FORWARD \
32     --jump SPOOFP
```

A szabályrendszer törlésére a következő parancsokat használhatjuk:

```
$ iptables -F
$ iptables -X
$
```

A parancsok az összes szabályt és saját szabályrendszert törlik. Ha az alapszabály az ACCEPT, minden csomagszűrést kikapcsolnak!

7.2.4. A címfordítás

Néha hasznos lehet, ha az adatcsomagot továbbító számítógép a továbbítandó csomag tulajdonságait megváltoztatja. A tulajdonságok megváltoztatásának (*mangle*, *elferdítés*) célja a legtöbb esetben a csomagban található IP cím megváltoztatása, ezért sokszor használjuk a NAT (*network address translation*, hálózati címfordítás) rövidítést a jelzésére. A NAT mellett sokszor használjuk az SNAT (*source network address translation*, hálózati forrácímfordítás), illetve DNAT (*destination network address translation*, cél hálózati címfordítás) kifejezéseket, aszerint, hogy az adatcsomagban található forrás vagy címzett cím megváltoztatása-e a célunk.



A DNAT és az SNAT nem tipikus tűzfalszolgáltatások, hiszen a csomagszűrő tűzfal elsősorban csomagok szűrését végzi, nem pedig a csomagok megváltoztatását. Mivel azonban az iptables segítségével használhatjuk ezeket az eszközöket is, és a legtöbb esetben ugyanazon a számítógépen végezzük a címfordítást is, amelyiken a csomagszűrést, a számítógépet továbbra is tűzfalként emlegetjük.

Mind az SNAT, mind pedig a DNAT fontos gyakorlati szerepet kap minden nap-jaink hálózataiban, fontos tehát, hogy ismerjük ezeket az eszközöket. A következő oldalon bemutatjuk, hogyan használhatjuk a Linux rendszermagot és az iptables programot címfordításra, valamint megmutatjuk a címfordításnak a gyakorlat szempontjából legfontosabb alkalmazási területeit⁹.

A táblák

Az iptables (*IP tables*, IP táblák) programról már eddig is sok szó esett, de a nevének jelentése eddig homályban maradt. A program azért kapta a nevét a táblákról, mert a benne található szabályrendszerek táblázatokba szervezhetők. Az eddig bemutatott szabályrendszerek (INPUT, OUTPUT és FORWARD) mindegyike ugyanabban a táblában található, és mivel ez az alapértelmezett tábla, nem is kellett tudnunk a többiről. A címfordítás miatt most viszont meg kell ismerkednünk a többi táblával is.

Az iptables programmal a következő táblákat kezelhetjük:

filter Ez a tábla a csomagok szűrésére szolgál, benne olyan szabályokat szokás elhelyezni, amelyek a csomagszűrő tűzfalat alkotva megvédi a számítógépet és az általa védett hálózatot a támadásoktól.

Az iptables program számára ez a tábla az alapértelmezett; ha az itt található szabályrendszereket akarjuk kezelni, nem kell megadni a tábla nevét. (A könyv eddigi részeiben ezt a táblát kezeltük, de ezt külön nem jeleztük a programnak.)

⁹Az SNAT és a DNAT bizonyos hálózati szabványokat sért, így több protokoll esetében is – FTP, IRC, AFS – jelentkezhetnek problémák (a lektor).

Amint már láttuk, ebben a táblában minden megtalálhatók a következő beépített szabályrendszerek:

INPUT A helyi számítógépre érkező csomagok szűrésére szolgáló szabályrendszer.

OUTPUT A helyi számítógépen született csomagok szűrésére szolgáló szabályrendszer.

FORWARD A számítógépen áthaladó adatcsomagok szűrésére szolgáló szabályrendszer. Ezek az adatcsomagok nem a helyi számítógépen születtek, és nem a helyi számítógépre igyekeznek.

nat A hálózati címek megváltoztatására irányuló szabályok elhelyezésére szolgáló tábla, amelyben az SNAT és DNAT szolgáltatásokat valósíthatjuk meg.

Ebben a táblában foglalnak helyet a következő beépített szabályrendszerek:

PREROUTING Ebben a szabályrendszerekben azokat a címfordítást végző szabályokat helyezhetjük el, amelyeket rögtön a csomag beérkezése után szeretnénk életbe léptetni, az előtt, mielőtt a helyi számítógépre érkező és a továbbítás céljából elküldött csomagokat a rendszermag szétválasztotta volna.

OUTPUT Ebben a szabályrendszerekben a helyi számítógépen született csomagok címfordítását végezhetjük el.

POSTROUTING Ebben a szabályrendszerekben a helyi számítógépen keletkezett és más számítógép kérésére továbbított adatcsomagok címfordítását egyaránt elvezethetjük, mégpedig közvetlenül azelőtt, hogy elhagynák a számítótépet.

Fontos tudnunk, hogy erre a táblára minden csak a kapcsolat első adatcsomagai kerülnek. Amikor a szabályrendszerek alapján a Linux rendszermag eldönti, hogy milyen módon kell megváltoztatni a csomag címzettjének vagy feladójának címét, egyben azt is meghatározza, hogy mi legyen a kapcsolat további adatcsomagjainak sorsa, hogyan kell azokat módosítani. A kapcsolat során született további adatesomagokra tehát nem érvényesek a nat tábla szabályrendszereinek szabályai.

mangle Ebben a táblában olyan szabályrendszereket helyezhetünk el, amelyek megváltoztatják a csomagokat, de nem címfordítást végeznek, sem a forrás, sem pedig a címzett IP címét nem változtatják meg. Ez a tábla a gyakorlat szempontjából nem olyan fontos, mint a többi, ezért nem tárgyaljuk részletesen.

raw Általában kivételek érvényesítésére használt tábla, amelyet nem tárgyalunk részletesen.

Azt, hogy melyik táblát kívánjuk kezelni, az iptables programnak a következő kifejezéssel jelezhetjük:

--table név A program --table vagy -t kapcsolójával megadhatjuk, hogy melyik táblát kívánjuk kezelni. A kifejezésben szereplő név a tábla neve. Az alapértelmezett tábla a filter, ennek kezelése során a tábla nevét nem kötelező megadni.

A következő példa bemutatja, hogyan használhatjuk a kifejezést adott tábla meghatározására.

77. példa. Kérdezzük le a nat tábla szabályrendszereit az iptables programmal!

```
$ iptables -t nat -L -n
Chain OUTPUT (policy ACCEPT)
target     prot opt source          destination
SNAT       all   --  0.0.0.0/0    0.0.0.0/0    to:10.10.1.1

Chain POSTROUTING (policy ACCEPT)
target     prot opt source          destination
SNAT       all   --  0.0.0.0/0    0.0.0.0/0    to:10.10.1.1

Chain PREROUTING (policy ACCEPT)
target     prot opt source          destination
```

A példában látható válasz szerint csak egyetlen szabály van, amelyik forrás IP címfordítást végez.

A forráscím megváltoztatása

A nat táblában a forrás hálózati címek megváltoztatására használhatjuk a következő sorsokat:

SNAT Azoknak a csomagoknak, amelyeknek ezt a sorsot jelöljük ki, a Linux rendszermag megváltoztatja a forrás hálózati címét, és az esetükben a szabályrendszer szabályainak vizsgálatát befejezi. Ha tehát egy adatcsomagra illeszkedik egy olyan szabály, amelyik ezt a sorsot jelöli ki a csomag számára, a csomagra már nem lesz egyetlen szabály sem érvényes az adott szabályrendszerben.

Ez a sors csak a nat táble POSTROUTING szabályrendszerében használható.

Ha egy adatcsomagnak a szabályok az SNAT sorsot jelölik ki, a rendszermag gondoskodik arról, hogy a kapcsolat további csomagjainak is ez legyen a sorsa. Ez fontos, hiszen a címzett számítógép nem tudná feldolgozni a folyamatos kapcsolat csomagjait, ha azok más-más forráscímmel érkeznének.

A Linux rendszermag gondoskodik arról, hogy az SNAT sorsú csomagokra válaszul érkezett csomagokat az eredeti küldőhöz irányítsa. Ezt a műveletet

néha de-SNAT műveletként emlegetjük, jelezve ezzel, hogy az SNAT művelet fordítottjáról van szó.

Ha az iptables programnak SNAT sorsú szabály elhelyezésére adunk utasítást, használnunk kell a következő kapcsolót:

--to-source cím A kifejezés segítségével megadhatjuk, hogy milyen forráscímmel lássa el a szabály alapján a rendszermag a kimenő csomagokat. (Valójában ezt a kapcsolót összetettebb formában is megadhatjuk, de az egyszerűség kedvéért most csak az egyszerű formát tárgyaljuk.)

Ha a kifejezést többször is megadjuk, a szabály minden alkalmazásakor a soron következő címet kapja a csomag. Ha például két címet adunk meg, az első csomag az első címet kapja, a második csomag a második címet, a harmadik ismét az elsőt, a negyedik a másodikat és így tovább¹⁰.

MASQUERADE Ennek a sorsnak a hatása nagyjából megegyezik az SNAT sors hatásával, azzal az apró különbséggel, hogy ennek a sorsnak a használata során nem kell megadnunk a címet, amelyet a csomag forráscímként fog kapni. Az adatcsomagok e sorsnak a hatására mindenkor azt a címet kapják forráscímként, amelyik a kimenő hálózati csatolóhoz van rendelve.

Az ilyen sors végrehajtása kissé több munkát jelent a rendszermagnak – minden kimenő csomag esetében megvizsgálja a csatoló hálózati címét –, ezért csak abban az esetben javasolható a használata, ha a kimenő hálózati csatoló DHCP szabvány szerint időről időre megváltoztatja az IP címét.

Ezt a sorsot a fejlesztők kimondottan olyan felhasználók kedvéért készítették, akik otthoni internetfelhasználóként a szolgáltatótól dinamikus IP címet kapnak, de szeretnék a teljes házi számítógép-hálózatot az Internetre kötni¹¹.

Az SNAT segítségével az útválasztó képes eltitkolni, hogy a kimenő adatcsomagok a belső hálózat mely számítógépről indultak. Az SNAT tehát egyfajta védeottséget, intimitást biztosít a belső hálózat felhasználóinak, különösen akkor, ha sokan vannak, hiszen a külvilág nem tudhatja, hogy pontosan melyik számítógép hozta létre a hálózati forgalmat. Fontos lehet azonban tudnunk, hogy a belső hálózaton használt alkalmazás bármilyen adatot elhelyezhet a csomagban, elküldheti például a belső hálózaton használt IP címet, a felhasználó felhasználói nevét vagy akár a bankszámlaszámát is. Tökéletes biztonságot, tökéletes névtelenséget az SNAT sem garantál.

¹⁰A szakirodalom ezt a körkörös, egyenlő elosztást biztosító módszert round-robin eljárásnak hívja, utalva egy angolszász jogi fogalomra, az iratok körkörös elrendezésben való aláírására. Ez az aláírás-minta biztosítja, hogy a dokumentumot aláírók között ne legyen első és ne legyen utolsó.

¹¹Az ilyen tevékenységet az internetszolgáltatóval kötött szerződés vagy az internetszolgáltató más szabályzata tilthatja. Fontos lehet tudnunk azt is, hogy a szolgáltató esetleg képes lehet felderíteni, hogy az adott számítógép mögött belső hálózat van.

Nagyon jól használható az SNAT akkor is, ha a belső hálózaton magáncélra használható IP tartomány van kiosztva. Az ilyen hálózati címeket nem küldhetjük az Internetre, az SNAT azonban megoldja ezt a problémát. Az SNAT használatával elérhetjük, hogy a teljes hálózatunk egyetlen IP címmel üzemeljen.

A következő nagyon egyszerű példa bemutatja, hogyan használhatjuk a forrás-cím átirását, hogyan készíthetünk SNAT feladatokat ellátó szabályrendszert.

78. példa. Készítsünk egyszerű héjprogramot, amely a forrás hálózati cím fordításával köt össze két hálózatot! (A példaprogram által elhelyezett szabályt a 224. oldalon található 77. példa mutatja be.)

```

1  #! /bin/bash
2
3  KULSO_IP="10.10.1.1"
4  KULSO_NIC="eth1"
5
6  #
7  # Ami a külső csatolón elhagyja a számítógépet, az a mi IP
8  # címünket kapja.
9  #
10 iptables --table nat \
11     --flush
12
13 iptables --table nat \
14     --append POSTROUTING \
15     --out-interface $KULSO_NIC \
16     --jump SNAT \
17     --to-source $KULSO_IP
18
19 # Engedélyezzük a csomagok továbbítását.
20 echo 1 >/proc/sys/net/ipv4/ip_forward

```

A példa előállítására használt számítógépben két hálózati csatoló volt, de ugyanez a módszer használható több hálózati csatoló esetén is. A programban kizárolag a külső hálózati csatoló neve és a külső hálózati csatoló IP címe található meg, a többi hálózati csatoló beállítása nem lényeges.

Az SNAT működése a programban igen egyszerűen fogalmazódik meg: „ami ezen a hálózati csatolón hagyja el a számítógépet, az ezt a forráscímet kapja”.

A példa kapcsán fontosnak tartjuk megjegyezni, hogy ez a szabályrendszer nem szűri a csomagokat, védettséget nem ad a belső hálózatnak. Célja az SNAT minél egyszerűbb bemutatása.

A bemutatott egyszerű példa nem védi a belső hálózatot attól, hogy a külső számítógépek felvegyék a kapcsolatot. Az SNAT eltitkolja a külső hálózat elől a belső

hálózaton található számítógépek IP címeit, de ha valamelyik külső számítógép ismeri a belső hálózat címtartományát, tud csomagot küldeni oda. Ez ellen nyilvánvalóan védekezhetünk úgy, hogy a csomagszűrő szabályok segítségével szűrjük a külső hálózat felől érkező csomagokat.

Tudnunk kell azonban, hogy mielőtt a bejövő adatcsomag a filter tábla FORWARD szabályrendszerére kerülne, átesik a de-SNAT eljáráson. A külső hálózat számítógépéptől érkező válaszcsomag megkapja az igazi címzett címét – amelyet a külső számítógép nem is ismer –, mielőtt a szűrést végző szabályt a rendszermag megvizsgálná.

A következő példa bemutatja, hogyan szűrhetjük ki a külső hálózat felől érkező kéréseket.

79. példa. Egészítsük ki a 78. példában található programot oly módon, hogy az védje meg a belső hálózatot a külső hálózatról érkező kapcsolatfelvételi kísérletek ellen!

Vegyük észre, hogy nem dobjuk el azokat a csomagokat, amelyek a külső hálózatról érkeznek a belső hálózatra – feltételezve, hogy az SNAT miatt minden csomag egyazon címre érkezik –, mert amikor a csomagokat szűrjük, már az igazi címzett címe szerepel az adatcsomagban!

Eldobhatjuk viszont azokat a csomagokat, amelyek a külső hálózati csatolón érkeztek és nem egy már meglévő kapcsolat részét képezik. Ilyen beállítást hoz létre a következő néhány sor:

```
1 #! /bin/bash
2 #
3 # Forrás IP címfordítás és egyszerű szűrés.
4 #
5
6 KULSO_IP="10.10.1.1"
7 KULSO_NIC="eth1"
8
9 #
10 # minden kapcsolatfelvételi kérést elutasítunk, ami a külső
11 # hálózatról a belső felé tart.
12 #
13 echo "Csomagszűrés..."
14 iptables --policy FORWARD DROP
15 iptables --flush
16 iptables --append FORWARD \
17     --in-interface $KULSO_NIC \
18     --match state --state NEW \
19     --jump REJECT
20 iptables --policy FORWARD ACCEPT
21
```

```

22 #
23 # Ami a külső csatolón elhagyja a számítógépet, az a mi
24 # IP címünket kapja.
25 #
26 echo "Forráscím megváltoztatása..."
27 iptables --table nat \
28     --flush
29
30 iptables --table nat \
31     --append POSTROUTING \
32     --out-interface $KULSO_NIC \
33     --jump SNAT \
34     --to-source $KULSO_IP
35
36 # Engedélyezzük a csomagok továbbítását.
37 echo "Csomagtovábbítás engedélyezése..."
38 echo 1 >/proc/sys/net/ipv4/ip_forward
39

```

A programrészlet minden kapcsolatfelvételi kérést – minden csomagot, amely nem egy már meglévő kapcsolat része – elutasít, így a külső számítógépek nem tudják felvenni a kapcsolatot a belső számítógépekkel.

Nyilvánvaló, hogy ez a programrészlet nem teszi lehetővé a saját gépen futó alkalmazások elérését a külő hálózatról, hiszen ahhoz az INPUT szabályrendszert kellene hasonlóképpen módosítani.

A célcím megváltoztatása

Amint arról már szó volt, a DNAT kifejezéssel az adatcsomagban szereplő címzett címének megváltoztatását jelöljük. Ez a módszer az adatcsomagokat eredeti címzettjük helyett egy új címzett felé irányítja.

Az iptables esetében a következő sorsot a címzett címének megváltoztatására használhatjuk:

DNAT Azoknak a csomagoknak, amelyeknek ezt a sorsot jelöltük ki, a rendszermag a címzett helyén szereplő címét megváltoztatja, és esetükben a szabályrendszer szabályainak vizsgálatát befejezi.

Ezt a sorsot csak a nat tábla PREROUTING, illetve OUTPUT szabályrendszereiben használhatjuk.

Ha egy adatcsomagnak a DNAT sorsot jelöltük ki, a Linux rendszermag gondoskodik arról, hogy a kapcsolathoz tartozó további csomagok címzett címe is hasonlóan megváltozzon.

A rendszermag gondoskodik arról, hogy az átirányított csomagokra válaszul érkező csomagok forráscímét megváltoztassa, mintha azok az eredeti címről érkeztek volna. Ezt a folyamatot szokás az un-DNAT kifejezéssel jelölni.

Ha DNAT sorsú szabályt készítünk, használnunk kell a következő kapcsolót:

--to-destination *cím* A kifejezés segítségével megadhatjuk, hogy a rendszermag milyen címre továbbítsa a csomagot, milyen címzett címmel lássa el azt. (A valóságban ezt a kifejezést összetettebb formában is használhatjuk.)

Ha a kifejezést többször is megadjuk, a cím megváltoztatását a már ismert round-robin módszer alapján végzi a rendszermag. E lehetőséget felhasználhatjuk például arra, hogy a nagyszámú bejövő kérést különálló gépekre osszuk szét.

A célcím megváltoztatását használhatjuk például arra, hogy a csomagszűrést és forráscímfordítást végző számítógépen a külső forrásból származó kéréseket a belső hálózaton található kiszolgálókra engedjük. Azt szokás modani ilyen esetben, hogy „lyukat ütünk a tűzfalon” a kérések beeresztése érdekében. A kifejezés jól érzékelte a műveletet.

A következő példa bemutatja, hogyan üthetünk lyukat a tűzfalon az adott jellegű kérések beeresztéséhez.

80. példa. Módosítsuk a 79. példában található programot úgy, hogy a cél címének átírásával továbbítsa a külső hálózatról érkező SSH kapcsolatfelvételi kéréseket egy belső kiszolgálóra! A következő program ezt a feladatot is elvégzi:

```
1 #!/bin/bash
2 #
3 # Forrás IP címfordítás és egyszerű szűrés az ssh forgalom
4 # beengedésével.
5 #
6
7 KULSO_IP="10.10.1.1"
8 KULSO_NIC="eth1"
9 SSH_SERVER="172.17.1.3"
10
11 #
12 # minden kapcsolatfelvételi kérést elutasítunk, ami a külső
13 # hálózatról a belső felé tart, kivéve az ssh kiszolgálóra
14 # igyekvő csomagokat.
15 #
16 echo "Csomagszűrés..."
17 iptables --policy FORWARD DROP
18 iptables --flush
```

```
19
20 iptables --append FORWARD \
21     --protocol tcp \
22     --in-interface $KULSO_NIC \
23     --match state --state NEW \
24     --destination $SSH_SERVER \
25     --dport ssh \
26     --jump ACCEPT
27
28 iptables --append FORWARD \
29     --in-interface $KULSO_NIC \
30     --match state --state NEW \
31     --jump REJECT
32
33 iptables --policy FORWARD ACCEPT
34
35 #
36 # Ami a külső csatolón elhagyja a számítógépet, az a mi
37 # IP címünket kapja.
38 #
39 echo "Forráscím megváltoztatása..."
40 iptables --table nat \
41     --flush
42 iptables --table nat \
43     --append POSTROUTING \
44     --out-interface $KULSO_NIC \
45     --jump SNAT \
46     --to-source $KULSO_IP
47 #
48 # Lyukakat ütünk a tűzfalba a befelejővő csomagok számára.
49 #
50 echo "Célcímfordítás..."
51 iptables --table nat \
52     --append PREROUTING \
53     --protocol tcp \
54     --in-interface $KULSO_NIC \
55     --destination $KULSO_IP \
56     --dport ssh \
57     --jump DNAT --to-destination $SSH_SERVER
58
59 # Engedélyezzük a csomagok továbbítását.
60 echo "Csomagtovábbítás engedélyezése..."
61 echo 1 >/proc/sys/net/ipv4/ip_forward
```

Figyeljük meg, hogy a forgalom beengedéséhez két helyen kellett módosítani a szabályrendszert! Egyfelől gondoskodnunk kellett arról, hogy a helyi gépre érkező SSH kapcsolatot alkotó csomagok megkapják a belső kiszolgáló IP címét címezettként (51–57. sor), másrészről gondoskodnunk kellett arról, hogy a belső kiszolgálóra igyekvő csomagokat a csomagszűrő átengedje (20–26. sor).

A példát mintaként használva természetesen más szolgáltatások továbbítását is engedélyezhetjük. Az ilyen módon engedélyezett összes szolgáltatást igénybe vehetik a külső hálózat számítógépei, ha a csomagszűrő tűzfalra csatlakoznak. A válaszcsomagokat a belső hálózaton található kiszolgáló fogja küldeni, de ezt a szolgáltatást igénybe vevő számítógép nem tudhatja.



A könyvben bemutatott, tűzfalszabályokat létrehozó héjprogramok egyike sem teljes. Vannak olyan programok, amelyek az egy hálózati csatolóval ellátott számítógépek védelmét igyekeznek bemutatni, és olyanok, amelyek egy belső hálózat forgalmát kezelik. Nincs a könyvben olyan program, amely minden feladatkört felvállalná!

A könyvben jellegzetesen hiányos programokat mutattunk be. Ezzel az volt a célunk, hogy egyértelműen jelezzük, hogy nem programokat, hanem ismereteket szeretnénk átadni. Az Interneten sok jó és kevésbé jó, előre beállított, könnyen használható héjprogram található a tűzfalak beállítására és kezelésére.

A naplázás

A tűzfalszabályokkal való kísérletezés során igen hasznos lehet a LOG sors, amely arra ad utasítást a rendszermagnak, hogy a rendszernapló segítségével rögzítse az illeszkedő csomag legfontosabb tulajdonságait. A LOG sors által előírt napló-bejegyzés elkészítése után a rendszermag a csomagra illeszkedő szabályt tovább keresi, a LOG sors tehát valójában nem befolyásolja a csomag életét.

Ha a szabályrendszereinkbe naplózást előíró szabályokat helyezünk, a csomagok útját, a szabályok működését követhetjük. Ez a szolgáltatás igen nagy segítséget nyújt a rendszermagba épített csomagszűrő és csomagmódosító rendszer megismeréséhez.

Ha a tűzfalunkat tökéletesen beállítottuk, a naplózást előíró LOG sors továbbra is hasznunkra lehet a betörési kísérletek, valamint a hálózat módosítása miatt bekövetkező hibák felderítésében. A folyamatos használat alatt azonban érdemes korlátozni a naplóbejegyzések számát, hiszen nagyobb hálózati forgalom esetén a csomagok egyedi naplózása teljesíthetetlen feladatokat róhat a számítógépunkre.

A naplázás bekapcsolását és korlátozását mutatja be a következő példa.

81. példa. A csomagszűrő szabályok között adott egy szabály, amely a külső hálózatról a belső hálózatra haladó, kapcsolatkiépítést kérő csomagokat szűri. Szeretnénk látni azoknak a csomagoknak a legfontosabb adatait, amelyek ennek a szabálynak esnek áldozatul.

Készítsük el a szabály másolatát, amely ugyanazokra a csomagokra illeszkedik, amelyekre a naplózni kívánt szabály, majd módosításuk az új szabályt, hogy naplózást végezzen! (Természetesen nyilvánvaló, hogy előbb kell naplóznunk, és csak utána szabad eldobnunk a csomagot...)

```

1  iptables --append FORWARD \
2      --in-interface $KULSO_NIC \
3      --match state --state NEW \
4      --match limit --limit "10/minute" --limit-burst 5 \
5      --jump LOG \
6      --log-prefix "Kérés befele:"
7
8  iptables --append FORWARD \
9      --in-interface $KULSO_NIC \
10     --match state --state NEW \
11     --jump REJECT

```

A szabály alapján a következő alakú naplóbejegyzések születnek:

```

1 Jun 26 11:34:09 localhost kernel: Kérés befele:IN=eth1 OUT=eth0
2 SRC=10.10.5.1 DST=172.17.1.3 LEN=84 TOS=0x00 PREC=0x00 TTL=63
3 ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=1008 SEQ=6
4 Jun 26 11:34:15 localhost kernel: Kérés befele:IN=eth1 OUT=eth0
5 SRC=10.10.5.1 DST=172.17.1.3 LEN=84 TOS=0x00 PREC=0x00 TTL=63
6 ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=1009 SEQ=101

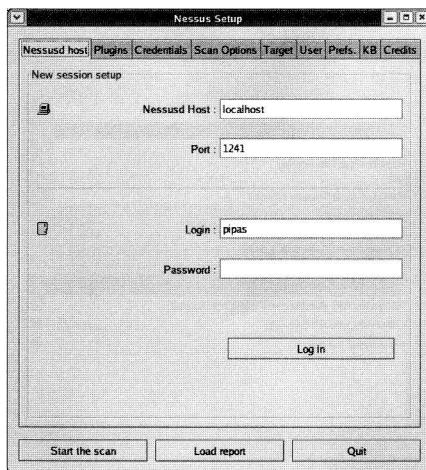
```

A naplóbejegyzéseket vizsgálva láthatjuk, hogy a rendszermag nem minden csomagot naplózott, tehát érvényesítette a szabályban található korlátozást, amely megvédi a rendszernapló túlterhelésétől.

7.3. A biztonság ellenőrzése

A számítógép-hálózatok üzemeltetése során fontos szerepet kap a biztonság, biztonsági szempontból pedig a legfontosabb, hogy a hibákat, biztonsági réseket és problémákat a rendszergazda vegye észre legelőször. Ebben a munkában nagy segítségünkre lehet a nessus nevű program.

A nessus program alapjában véve számítógépek hálózaton keresztül történő támadására szolgál. A segítségével a felügyeletünk alá tartozó számítógépeket, munkaállomásokat és kiszolgálókat biztonsági szempontból ellenőrizhetjük. Fontos azonban, hogy tisztában legyünk azzal, hogy amit csinálunk, az valójában a számítógépek megtámadása, ezért soha nem szabad mások által üzemeltetett számítógépeket vizsgálni a segítségével.



7.2. ábra. A nessus indulóképernyője

7.3.1. A támadóprogram telepítése

A program használatához telepítenünk kell a következő programcsomagokat:

libnasl A programcsomag egy programkönyvtárat tartalmaz, amely a **nessus** támadóprogramokat értelmezi és hajtja végre. A **NASL** (*nessus attack scripting language*, nessus támadó nyelv) rövidítés is erre utal.

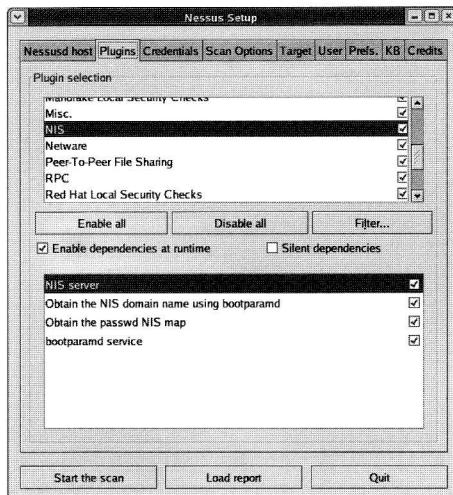
A **nessus** igen nagy erőssége, hogy a különféle programok és alkalmazások hibáinak feltárása után nagyon rövid idő alatt készülnek el a hibát jelző programmodulok, amelyek az adott hibát megtalálják a hálózatunkon. Ezt a programfejlesztők úgy érték el, hogy könnyen kezelhető egyszerű programozási nyelvet hoztak létre és építettek be a **nessus** programba, lehetővé téve ezzel, hogy egy-egy hiba ellenőrzésére könnyen és gyorsan megírhasuk a programmodult.

A **libnasl** forráskönyvtárában a **doc/** alkönyvtárból található a dokumentáció, amely lehetővé teszi, hogy saját hibaellenőrző programokat írunk.

nessus-libraries Ebben a programcsomagban olyan programkönyvtárakat találunk, amelyekre a **nessus** programnak szüksége van a futáshoz.

nessus-core Ebben a programcsomagban található maga a **nessus** program és a **nessusd** program, amely a tulajdonképpeni támadásokat végzi.

A **nessus** ügyfél-kiszolgáló elven épül fel. A felhasználó a **nessus** programot használja, ami ügyfélprogramként kapcsolódik a **nessusd** programhoz, amely megtámadja a kijelölt célpontokat. A két program természetesen képes tartani a kapcsolatot a hálózaton keresztül, így a támadásoknak nem



7.3. ábra. A nessus támadóprogramjai

feltétlenül kell arról a számítógépről származnia, amelyik előtt a felhasználó ül.

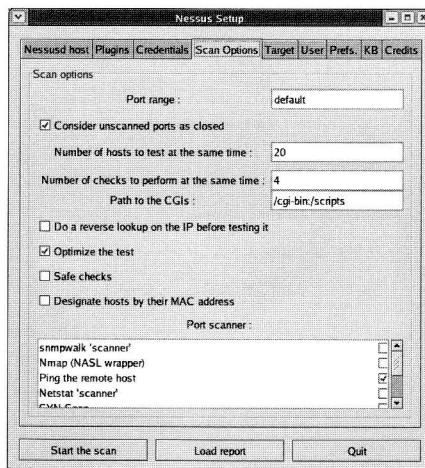
nessus-plugins Ebben a programcsomagban találhatók a NASL nyelven megírt támadóprogramok, amelyek az egyes hibák felderítésére használhatók. Ezeket a programokat a nessus programcsomagtól függetlenül tudjuk bővíteni, frissíteni.

Ha telepítettük a programcsomagokat, futtatnunk kell a **nessus-adduser** programot, hogy új felhasználót hozzunk létre a nessus nyilvántartásában. Az új felhasználó nevét és jelszavát megadva képes lesz támadásokat indítani a program segítségével.

7.3.2. A program indítása

A program használatához előbb el kell indítanunk a **nessusd** programot, amely a támadásokat kezdeményezi. A program kiírja a betöltött támadóprogramok számát (mintegy 7000 programról van szó!), majd elindul és fogadja a kapcsolatfelvételi kéréseket.

Ha a **nessusd** program már fut, elindíthatjuk a **nessus** programot, amely egy felhasználói felületet biztosít a hálózat átvizsgálásához. Indítás után a nessus a 7.2. ábrán látható ablakot jeleníti meg. Az ablak felső részén megadhatjuk, hogy melyik számítógépen fut a **nessusd** program, az alsó részén pedig azt, hogy milyen felhasználói névvel és jelszóval kívánunk bejelentkezni. Itt begépelhetjük a **nessus-adduser** programmal létrehozott felhasználó nevét és jelszavát.



7.4. ábra. A nessus támadás tulajdonságai

Ha begépeltük a nevet és a jelszót, bejelentkezhetünk a *log in* nyomógombbal. A bejelentkezés kissé lassú lehet, mert bejelentkezéskor a nessus letölti a nessusd által ismert ellenőrzőprogramok legfontosabb adatait, és ez elég sok időt vehet igénybe.

7.3.3. A támadás előkészítése

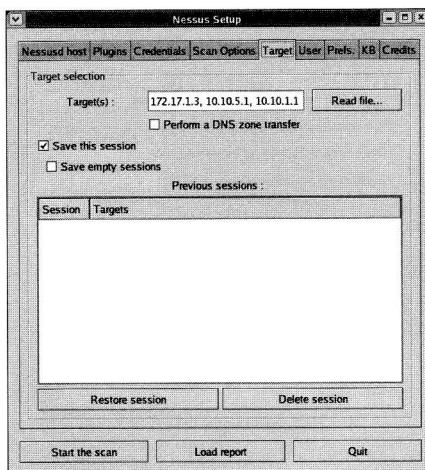
A sikeres bejelentkezés után be kell állítanunk, hogy milyen számítógépeket kívánunk megtámadni és azt, hogy milyen körülmények között szeretnénk lefolytatni a támadást.

A képernyő tetején kiválasztva a *plugins* fület, a támadást tulajdonképpen végző programok közül válogathatunk (7.3. ábra). A felső listában kiválaszthatjuk a téma kört, az alsó listában pedig a konkrét támadást. Ha kis hálózatot vagy csak egy számítógépet támadunk, érdemes lehet kijelölni az összes támadást az *enable all* myomógomb segítségével, nagyobb hálózatok esetén azonban a mindenre kiterjedő alapos átvizsgálás igen sok időt vehet igénybe, ezért hasznos lehet bizonyos támadóprogramok kiiktatása.

Az ablak alsó listájában található konkrét támadásra kattintva megtudhatjuk, hogy a támadás pontosan milyen biztonsági hiányosság felderítésére szolgál. Ez segíthet annak eldöntésében, hogy használnunk kell-e az adott támadóprogramot.

Ha az ablak felső részén található *scan options* fülre kattintunk, a biztonsági ellenőrzés néhány fontos elemét állíthatjuk be (7.4. ábra).

A *port range* mezőben megadhatjuk, hogy milyen hálózati kapukat kívánunk vizsgálni. Ha ezt a mezőt kitöljtük, és az alatta található, *consider unscanned ports as closed* (tekintsük zártnak a nem vizsgált kapukat) lehetőséget bekapcsoljuk,



7.5. ábra. A nessus támadás kiszemelt célpontjai

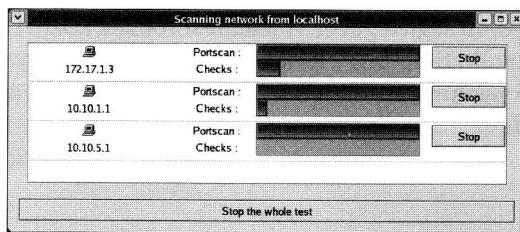
meglehetősen sok időt megtakaríthatunk, különösen nagyobb hálózatok átvizsgálása során. Ebben az esetben azonban lehetséges, hogy elkerüli valami a figyelmiüköt, hiszen a program nem vizsgál meg minden kapcsolatfelvételi lehetőséget.

A *number of hosts to test at the same time* (az egy időben vizsgált számítógépek száma), valamint a *number of checks to perform at the same time* (az egy időben végzett ellenőrzések száma) mezők beállításának segítségével gyorsíthatjuk a vizsgálatot, ha a támadó számítógépe elegendően gyors.

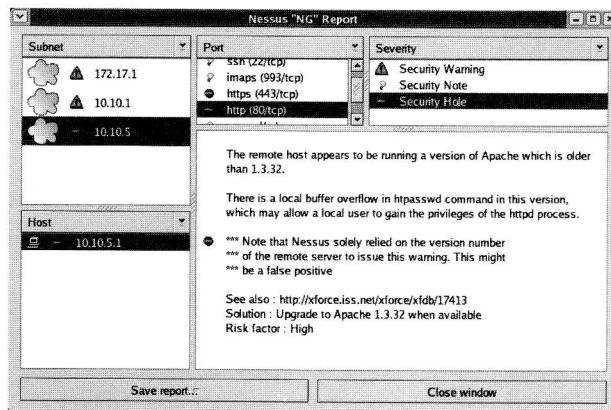
Nagyon fontos a *safe checks* mező állapotja. Ha ez a jelölőnégyzet be van kapcsolva, a nessus csak olyan támadóprogramokat futtat, amelyek nem okozhatják a támadott számítógép károsodását. Ha kikapcsoljuk ezt a mezőt, akkor előfordulhat, hogy a megtámadott számítógép működése ideiglenesen akadozik, a számítógép egy időre – esetleg az újraindításáig – nem működik. Nyilvánvaló, hogy ha alapos ellenőrzést akarunk végezni, engedélyeznünk kell a teljes ellenőrzést, előtte azonban nem árt elbarrikádozni magunkat, hogy ne érjenek el a munkájukban akadályozott felhasználóink, akik azt tapasztalják, hogy húszas csoporthokban „fagy le” az összes számítógép az épületben.

A nessus a konkrét támadások előtt megvizsgálja, hogy a támadott számítógép mely hálózati kapui vannak nyitva. Azt, hogy e fontos vizsgálatot milyen módszerrel végezze el, a képernyő alján a *port scanner* részben állíthatjuk be.

A legfontosabb beállítandó mező a megtámadott számítógép vagy számítógépek címét vagy nevét tartalmazza. A képernyő felső részén található *target* fülre kattintva beállíthatjuk, hogy a program mely számítógépet vagy számítógépeket támadja meg (7.5. ábra). Amint láthatjuk az ábrán, a *target(s)* mezőbe beírhatjuk egy vagy több számítógép IP címét (ha egynél több címet írunk be, a címeket



7.6. ábra. A nessus támadás közben



7.7. ábra. A nessus támadás eredménye

vesszővel kell elválasztanunk), de beolvashatjuk a címeket állományokból is, a nessus segítségével régebben mentett listát is betölthetünk, vagy dönthetünk úgy is, hogy a DNS kiszolgáló egy teljes névtartományát megtámadjuk.

Ha a támadás összes tulajdonságát beállítottuk, a támadást az ablak alsó részében található *start the scan* feliratú nyomógombbal indíthatjuk meg. Ha a támadás elindult, a program egy új ablakban folyamatosan jelzi, hogy az egyes megtámadott számítógépek vizsgálatában hol tart (7.6. ábra). Amint látható, a program egy időben több számítógéppel is kísérletezik, de a vizsgálat még így is sokáig tarthat.

7.3.4. Az eredmény kiértékelése

A munka legizgalmasabb, mégis legegyszerűbb szakasza az eredmények kiértékelés, a nessus által készített biztonsági jelentés áttanulmányozása.

A támadás végeztével a program egy új ablakot jelenít meg (7.7. ábra), amelyben az egyes támadóprogramok üzeneteit, eredményeit böngészhetjük át.

A nessus által készített teljes jelentés menthető az ablak alsó részében található

save report nyomógomb segítségével. A jelentést többek közt HTML, L^AT_EX és ASCII formátumban is menthetjük, hogy később kinyomtassuk vagy átolvassuk. A rendelkezésünkre álló jelentésformátumok közül talán a HTML a leghasználhatóbb és a legolvashatóbb...

8. fejezet

Hálózati alkalmazások

E fejezetben hálózati alkalmazásokról, a számítógép-hálózaton keresztül elérhető magasszintű szolgáltatásokról lesz szó. A fejezetet áttanulmányozva olyan szolgáltatásokat indíthatunk el és üzemeltethetünk, amelyek értelmet adnak a hálózat létrehozásának.

8.1. A hálózati állományrendszer (NFS)

A Sun Microsystems által kifejlesztett NFS (*network file system*, hálózati állományrendszer) lehetővé teszi, hogy a távoli számítógépeken található állományokat helyi állományként kezeljük[157]. Habár az NFS mára meglehetősen elavultnak tekinthető, használata és kezelése olyan egyszerű, hogy érdemes néhány szót ejtünk róla.



Az NFS bírálói elsősorban azt róják fel, hogy a behatolások ellen nem ad megfelelő védelmet. Ha ezt a rendszert használjuk UNIX rendszerek állományainak megosztására, akkor mindenkorban szem előtt kell tartanunk ezt. Mind az NFS, mind pedig a vele szoros együttműködésben használt NIS csak védett, megbízható hálózatokon belül javasolható megoldás¹.

Az NFS UNIX rendszerek állománymegosztására használható. Jellemző példája ennek egy GNU/Linuxot futtató számítógépeket tartalmazó hálózat, ahol azt szeretnénk elérni, hogy a felhasználók bármely gépet használva ugyanazt a saját könyvtárat tudják kezelni. Az NFS lehetővé teszi, hogy a felhasználók minden számítógépen helyi állományként kezeljék az állományokat, amelyek a saját könyvtárakban vannak.

¹Érdemes az NFS 4-es változatát használni, ami képes Kerberos és más eszközök segítségével javítani a biztonságot (a lektor).

Az NFS ügyfél-kiszolgáló felépítésű, ami azt jelenti, hogy az állományok fizikailag egyetlen számítógépen – a kiszolgálón – tárolódnak, a többi számítógépen – ügyfeleken – pedig helyi állományoknak látszanak ugyan, de a helyi háttér-tárra soha nem kerülnek. Az NFS beilleszthető, lehetővé teszi, hogy a kiszolgálón található könyvtárakat az ügyféloldalon található könyvtárakba illesszük. Az állományok tartalma minden állományhozzáféréskor a hálózaton keresztül jut el a kiszolgálón található háttértárhoz és vissza.

Az NFS az állományok tartalmát és minden tulajdonságát képes megosztani a hálózat gépei között, de az állománytulajdonságok között szereplő tulajdonos és tulajdonoscsoporthoz azonosítót (UID, GID) a rendszer csak akkor tudja névként használni, ha a kiszolgáló és az ügyfél számítógépek közös felhasználói nyilvántartást használnak.

8.1.1. Az NFS kiszolgáló

Az NFS kiszolgáló az a számítógép, amelyik a helyi háttértáron található állományokat „felajánlja” és elérhetővé teszi más számítógépek számára. Az NFS kiszolgálót igen könnyű beállítani és elindítani, de soha nem szabad elfelednünk, hogy a használata csak védett hálózaton javasolható. Az NFS is az RPC (*remote procedure call*, távoli eljáráshívás) rendszeren alapul, ezért annak az NFS indítása előtt már működnie kell.

Az NFS kiszolgáló a /etc(exports) állomány alapján működik, ez az állomány határozza meg, hogy milyen könyvtárakat szolgáltasson a hálózaton. Az állomány minden sora egy-egy könyvtárra vonatkozik. Az első oszlop határozza meg, melyik könyvtárról van szó, a további oszlopai pedig megadják, hogy az egyes munkaál-lomások milyen módon használhatják azt. A következő példa ezt mutatja be.

82. példa. A következő állományrészlet a /etc(exports) állományban két könyvtárat tesz elérhetővé külső számítógépek számára:

1	/home	twinstar.1kk(rw)	gamma.szk(rw,no_root_squash)
2	/home-1	*.1kk(ro)	

A példa 1. sora a /home/ könyvtárra, 2. sora pedig a /home-1/ könyvtárra vonatkozik. Fontos, hogy a könyvtárakat nem ismételhetjük meg, nem vonatkozhat egy könyvtárra a /etc(exports több sora is.

Az állomány második és további oszlopai azt írják le, hogy az adott könyvtárat milyen számítógépek számára, milyen feltételek mellett szolgáltatja a kiszolgáló. Amint a példa 1. sorában láthatjuk, több számítógép is felsorolható, mégpedig szóközökkel elválasztva. A 2. sorban látható, hogy a számítógépek neveiben használhatjuk a * karaktert az általánosításra. A 2. sor minden számítógépre érvényes, amelyek neve .1kk végeződésű.

Az exports állományban szereplő alkönyvtáraknak nem kell a helyi adathordozók beillesztési pontjainak lennie. Az NFS tehát könyvtákat szolgáltat, nem pedig adathordozókat.

A számítógép neve után zárójelek között kapcsolókat adhatunk meg, amelyek leírják, hogy milyen módon, milyen feltételekkel szolgáltatjuk az adott könyvtár tartalmát. Az itt használható kulcsszavak közül a legfontosabbak a következők:

rw Az elérés olvasásra és írásra is engedélyezett. Alapértelmezés szerint az NFS kiszolgáló csak olvasható módon engedélyezi a hozzáférést, ha tehát lehetővé akarjuk tenni, hogy a munkaállomások írjanak a kiszolgáló adathordozójára, ezt a kapcsolót kell használnunk.

ro Csak olvasható hozzáférés. Ez az alapértelmezett hozzáférési mód.

sync Ha ezt a kapcsolót használjuk, az adatokat a kiszolgáló azonnal rögzíti a háttértárra. Ez lassítja a működést, de némi védetséget jelent a meghibásodásokból adódó adatvesztés ellen. Ezt a kapcsolót csak ritkán használjuk.

anonuid A kulcsszó után egy felhasználói azonosítót kell megadnunk az = jellel.

Az NFS kiszolgáló a „legkevesebb joggal rendelkező”, azonosítatlan felhasználót ennek a felhasználónak a jogaival azonosítja.

anongid Ugyanaz, mint az anonuid, csak csoportra.

root_squash A távoli rendszeren lévő rendszergazda a helyi gépen nem a rendszergazda jogaival rendelkezik, hanem az anonuid és anongid kulcsszavakkal megadott felhasználó jogaival. Ennek az alapértelmezett viselkedésnek a jelzésére használhatjuk a root_squash kulcsszót (*squash*, *ledorongol*).

no_root_squash Ha ezt a kulcsszót használjuk, az ügyféloldalon rendszergazdaként azonosított felhasználó az NFS kiszolgálóról beillesztett alkönyvtárban is rendszergazdai jogokkal fog rendelkezni. Ez az üzemmód biztonsági okokból általában nem javasolható.

all_squash Az ügyféloldal összes felhasználója a kiszolgálón az anonuid és anongid által meghatározott felhasználó jogaival fog rendelkezni.

no_all_squash Az all_squash ellentéte. Ez az alapértelmezett viselkedés.

Ha a /etc(exports állományt megfelelően kitöltöttük, elindíthatjuk az NFS kiszolgálószolgáltatást. Az NFS kiszolgálót általában a szolgáltatásoknál megszokott módon (1 példa) indíthatjuk és állíthatjuk le.

8.1.2. Az NFS ügyfél

A kiszolgálóoldalon „felajánlott” alkönyvtákat az ügyféloldalon a mount program segítségével illeszthetjük be az állományrendszerbe. A programnak az állományrendszer típusaként az nfs kulcsszót kell megadnunk, ahogyan azt a következő példában is láthatjuk.

83. példa. Illessük be adott kiszolgáló NFS állományrendszerét egy könyvtárba a mount program segítségével:

```
$ mount -t nfs alpha.lkk:/home /home/exported
```

Amint látjuk, a mount számára meg kell adnunk, hogy a beilleszteni kívánt állományrendszer típusa nfs.

A beillesztéskor megadtuk az NFS kiszolgáló nevét, majd egy kettőspont után a kiszolgálón található könyvtár nevét, amelyet be szeretnénk illeszteni.

A beillesztést automatikussá tehetjük, ha a megfelelő sort elhelyezzük az állományrendszer-táblázatban (/etc/fstab). Ezt mutatja be a következő példa.

84. példa. A 83. példa állományrendszerét jegyezzük fel a /etc/fstab állományban!

1	alpha.lkk:/home	/home/exported	nfs	defaults	0	0
---	-----------------	----------------	-----	----------	---	---

Amint látjuk, a feladat igen könnyen elvégezhető.

Összefoglalásképpen érdemes megemlíteni, hogy gyakorlatilag minden, UNIX rendszert használó számítógépen ilyen egyszerűen kezelhető az NFS állományrendszer, ami ráadásul képes kapcsolatot teremteni az egyes UNIX-változatok között.

8.2. Felhasználói adatbázis megosztása (NIS)

A Sun Microsystems által kifejlesztett *yellow pages* rendszer alkalmas arra, hogy a felhasználók adatait egyetlen gépen tárolva azokat egy teljes hálózaton felhasználjuk. A *yellow pages* rendszert később átkeresztelték NIS-re (*network information service*, hálózati információszolgáltatás). A NIS segítségével lehetőségünk nyílik arra, hogy a felhasználók számára egységes jelszórendszer alakítsunk ki, amelyet a rendszergazdának csak egyetlen helyen, a kiszolgálón kell kezelnie.



Mivel a *yellow pages* név már foglalt volt, a Sun Microsystems átkeresztelte a programrendszert NIS-re, annak fejlettebb változatát pedig NIS+-ra, sokan azonban még ma is a régi nevet használják.

Tudnunk kell, hogy a NIS rendszer úgynevezett tartományok (*NIS domain*) alapján dolgozik. minden NIS kiszolgáló és NIS ügyfél egy-egy tartományhoz tartozik, a felhasználók nyilvántartása pedig e tartományra érvényes. Ez teszi lehetővé, hogy egy fizikai hálózaton több munkacsoport, több egymástól független NIS rendszer működhessen. A NIS rendszer használatához elengedhetetlenül fontos, hogy egyedi NIS tartománynevét válasszunk, és azt mind az ügyfél, mind pedig a kiszolgáló gépeken beállítsuk.

A NIS tartománynév beállítására és lekérdezésére a domainname program szolgál. Beállításkor a parancs után a tartománynevet kell megadnunk, lekérdezéskor pedig a parancs a szabványos kimenetére írja a nevet. Ezt mutatja be a következő példa.

85. példa. Kérdezzük le a számítógépünkön érvényes NIS tartománynevet!

```
$ domainname  
Linux  
$
```

Ha a környezet nevét be akarnánk állítani, a következő módon tehetnénk meg:

```
$ domainname masvalami  
$
```

Ehhez természetesen rendszergazdai jogokra volna szükségünk.

A domainname program segítségével beállított tartománynév a számítógép újraindításáig marad érvényben. Ha azt szeretnénk, hogy a tartománynév az újraindítás után is megmaradjon, a megfelelő beállítóállományt kell módosítanunk.



Red Hat alapú terjesztésekben a /etc/sysconfig/network állományban található NISDOMAIN változónak adott érték lesz a NIS tartománynév a rendszer indítása után – de természetesen már a NIS indulása előtt. A NIS tartománynevet itt érdemes beállítani.



Debian alapú rendszereken a /etc/defaultdomain állomány tartalmazza a NIS tartomány nevét. Ennek a szöveges állománynak csak a tartománynevet kell tartalmaznia. Ha NIS rendszert használunk, a tartománynevet itt érdemes beállítani.

Ma már a legtöbb GNU/Linux terjesztés grafikus beállítóprogramokat tartalmaz a felhasználói azonosítás beállítására is. A NIS tartománynevet és a NIS kiszolgáló nevét általában ezekkel a programokkal is beállíthatjuk.

8.2.1. A NIS kiszolgáló üzembe helyezése

A NIS kiszolgáló az a számítógép, amelyik a tárolt felhasználói adatokat más számítógépek számára szolgáltatja. Maga a program képes ugyan bármilyen adatbázist kezelni, telepítésének célja azonban általában kizárolag a felhasználói adatok kezelése.

A NIS kiszolgáló a felhasználók jelszavát, felhasználói azonosítószámát (UID) és felhasználói nevét szolgáltatva lehetővé teszi, hogy egységes felhasználói azonosítókat használunk kiterjedt UNIX hálózatokon.

A NIS kiszolgáló nem közvetlenül a rendszeren található felhasználói adatbázis-ból veszi az adatokat, a gyorsabb működés érdekében azokból bináris formátumú állományokat készít. Ha tehát egy új felhasználót hozunk létre, létre kell hoz-nunk vagy frissítenünk kell a NIS bináris adatbázisát. Ehhez a `/var/yp/` könyvtárban kell lefuttatnunk a `make` programot. Ez a könyvtár ugyanis tartalmaz egy `Makefile` állományt, amely pontosan leírja a `make` számára, hogy milyen műveleteket kell elvégeznie az adatbázis előállításához, illetve frissítéséhez.

A bináris adatbázis első létrehozásakor a `/var/yp/` könyvtárban egy – a NIS tartománynévvel megegyező nevű – alkönyvtár jön létre, amely a bináris adatbázist tartalmazza. Ha a művelet után egy (`none`) nevű alkönyvtárat találunk, akkor nem volt beállítva a NIS tartománynév.

A NIS tartomány adatbázisainak létrehozását mutatja be a következő példa.

86. példa. Frissítsük a NIS adatbázisokat azon a számítógépen, amelyet NIS kiszolgálónak választottunk!

```
$ cd /var/yp/
$ make
gmake[1]: Entering directory '/var/yp/Linux'
Updating netidbyname...
failed to send 'clear' to local ypser: RPC: Program not registered
make[1]: Leaving directory '/var/yp/Linux'
```

Láthatjuk, hogy az adatbázis frissítése után a `Makefile` alapján a `make` megpróbálta jelezni a futó NIS kiszolgálónak, hogy megváltoztak az adatbázisok. Ez nem sikerült, hiszen a kiszolgálóprogram nem futott. Ha elindítjuk a kiszolgálót, ez a hibaüzenet nem jelenik meg.

A bináris adatbázisokat létrehozó `Makefile` kapcsán meg kell vizsgálnunk, hogy milyen adatbázisokat kell használni, hogy a felhasználók bejelentkezhessenek a munkaállomásokon. Az elterjedten használt GNU/Linux terjesztések esetében általában nem kell megváltoztatni a `Makefile` állományt, de előfordulhat, hogy a létrehozandó adatbázisok tulajdonságainak beállításához a `Makefile` elején található értékadásokat módosítanunk kell.

Érdemes megismerkednünk a következő adatbázisnevekkel:

`group.bygid` A felhasználói csoportok adatait tartalmazó adatbázis, amelyet csoportazonosító alapján lehet lekérdezni.

`groupbyname` A felhasználói csoportok adatait tartalmazó adatbázis, a felhasználói csoportok nevei alapján.

`passwdbyname` A felhasználók adatait tartalmazó adatbázis a felhasználói nevek alapján.

`passwd.byuid` A felhasználók adatait tartalmazó adatbázis a felhasználók azonosítószáma alapján.

Az árnyékjelszórendszer használata esetén némi problémát okozhat a jelszavak kezelése a NIS kiszolgálón. A régebbi GNU/Linux rendszerek NIS kiszolgálói árnyékjelszórendszer használata esetén a shadowbyname adatbázist is szolgáltatták, amelyben megtalálható volt a felhasználók felhasználói neve és jelszava is. Ezt az adatbázist használva a munkaállomás képes volt ellenőrizni a felhasználók jelszavát. Újabb GNU/Linux rendszerekben a szolgáltatott adatbázisokat létrehozó Makefile képes az árnyékjelszórendszerből a felhasználói jelszavakat kigyűjteni és a szokásos módon, a passwdbyname és passwdbyuid adatbázisokban elhelyezni. Bizonyos gyűjteményekben ezt a Makefile állomány elejének módosításával engedélyezni kell. Ezt mutatja be a következő példa:

87. példa. Ellenőrizzük, hogy a NIS adatbázisait létrehozó Makefile a jelszavakat kigyűjt-e az árnyékjelszórendszerből! A következő sorok ebből az állományból származnak:

```
1 # Should we merge the passwd file with the shadow file ?
2 # MERGE_PASSWD=true/false
3 MERGE_PASSWD=true
4
5 # Should we merge the group file with the gshadow file ?
6 # MERGE_GROUP=true/false
7 MERGE_GROUP=true
```

A megjegyzések magyar fordítása a következő:

```
1 # Egyesítsük a passwd állományt a shadow állománnyal?
2 # MERGE_PASSWD=true/false
3 MERGE_PASSWD=true
4
5 # Egyesítsük a group állományt a gshadow állománnyal?
6 # MERGE_GROUP=true/false
7 MERGE_GROUP=true
```

Amint látjuk, a két változó értékét true értékre állítva az árnyékjelszórendszer tartalma is bekerül a szolgáltatott adatbázisba, így a munkaállomások megkapják a kódolt jelszót, és képesek lesznek a felhasználó által begépelt jelszó ellenőrzésére.

A NIS kiszolgáló indításához be kell állítanunk azt, hogy mely IP címtartományok számára szeretnénk NIS szolgáltatást nyújtani. Erre a securenets nevű állomány szolgál, amely általában a /var/yp/ könyvtárban található.



Fontos tudnunk, hogy a NIS szolgáltatás egyik gyenge pontja a hálózati biztonság. mindenkorban javasolható, hogy csak a legszükségesebb – és legmegbízhatóbb – hálózatok felé engedélyezzük a szolgáltatást, olyan

hálózaton, amelyet idegenek nem használnak (például otthon) vagy amelyet gondosan karbantartott csomagszűrő tűzfal véd. Nem elegendő azonban, ha a fizikai hálózat bizonyos címtartományaira korlátozzuk a NIS szolgáltatást a securenets állomány segítségével, hiszen a hálózaton folyó forgalmat le lehet hallgatni.

A securenets állományba alhálózati maszkokat és IP címeket írhatunk, amelyek felé NIS szolgáltatást kívánunk nyújtani. Ezt mutatja be a következő példa.

88. példa. A következő példa két bejegyzést tartalmazó állományt mutat, amelyek közül az első magára a NIS kiszolgálóra mutat. A NIS kiszolgálónak minden elérhetőnek kell lennie a saját számítógépről! A példa második bejegyzése a 10.10.0.0 „B” osztályú címtartományra vonatkozik, az itt található számítógépek számára engedélyezi a NIS szolgáltatást.

```

1 # A helyi gép felé szolgáltatunk.
2 255.0.0.0      127.0.0.0
3
4 # Erre a hálózatra szolgáltatunk.
5 255.255.0.0    10.10.0.0

```

Amint láthatjuk, megjegyzésekkel szokásos módon helyezhetünk el az állományban.

A NIS kiszolgáló további beállításait a /etc/ypserv.conf állomány tartalmazza. Az állományban néhány kulcsszóval beállíthatjuk a kiszolgáló működési paramétereit – ezekről a paraméterekről a man ypserv.conf parancs begépelésével olvashatunk bővebben –, valamint meghatározhatjuk, hogy a kiszolgáló az egyes adatbázisok mely számítógépek számára szolgáltassa.

A szolgáltatott adatbázisok leírására a különféle terjesztések, a NIS kiszolgáló különféle változatai más-más formájú bejegyzéseket használnak. A két szabályforma a következő:

cím:tartomány:adatbázis:védelem Ennek az újabb rendszereken használt változatnak az esetében a szabályok a következő részekből állnak:

cím A számítógépek IP címei, amelyekre az adott bejegyzés vonatkozik. Itt általában teljes címtartományokat szokás megadni a szokásos formában (például 172.17.0.0/255.255.0.0).

Ha az első oszlopban megadott IP címet csonkoljuk, a megfelelő módon kialakított címtartományt jelenti. Az előző példa egyszerűsített formában 172.17. lesz.

Az első oszlopban használhatjuk a * és ? helyettesítő karaktereket is, a * például az összes címtartományt jelenti.

tartomány A tartomány neve. Itt is használhatjuk a * helyettesítő karaktert.

Ha egyszerűen egy * karaktert írunk ebbe az oszlopa, az az összes szolgáltatott tartományt jelenti.

adatbázis A szolgáltatott adatbázis neve. Ha a mezőben a * karaktert használjuk, minden adatbázist szolgáltatunk.

védelem A használt biztonsági beállítást jelző kulcsszó. A none kulcsszó jelzi, hogy semmilyen biztonsági megkötést nem teszünk, a deny jelzi, hogy az adatbázist nem szolgáltatjuk, illetve a port kulcsszó jelzi, hogy csak akkor engedélyezzük az adatbázis lekérdezését, ha a kérést a megfelelő hálózati kapuról kapta a kiszolgáló. Ez utóbbi módszer némi védettséget jelent, hiszen így a kérésnek olyan kapuról kell érkeznie, amelyet GNU/Linux rendszeren csak a rendszergazda használhat.

cím:tartomány:védelem:jelszó védelem A kifejezésben szereplő cím, tartomány és védelem mezők jelentése a már bemutatott módon alakul. A jelszó védelem mezőben meghatározhatjuk, hogy a NIS kiszolgáló távolítsa-e el a kódolt jelszót az adatbázisból a kérések kiszolgálása előtt. Ha itt a yes kulcsszó található, a kiszolgáló a munkaállomásnak nem küldi el a jelszavakat, ha ellenben a no kulcsszót használjuk, a munkaállomás megkapja az adatbázisokban található jelszavakat is.

A következő példa bemutatja, hogyan szolgáltathatjuk a felhasználók bejelentkezéséhez szükséges adatokat egy olyan NIS kiszolgálón, amely a felhasználói neveket és jelszavakat a shadow.byuid adatbázisban tárolja.

89. példa. A 87. példa Makefile állománya segítségével létrehozott adatbázisok szolgáltatására készítük el a /etc/ypserv.conf bejegyzéseit, amelyek lehetővé teszik a felhasználók bejelentkezését! Az állományban található megjegyzések segítenek ebben.

	/etc/ypserv.conf
1	#
2	# Host : Domain : Map : Security
3	#
4	* : * : passwdbyname : port
5	* : * : passwdbyuid : port
6	* : * : groupbyname : port
7	* : * : groupbygid : port

Amint látható, a fontosabb adatbázisokat felsoroltuk, és gondoskodtunk róla, hogy a NIS kiszolgáló csak a megfelelő hálózati kapuról érkező kéréseket szolgálja ki.

Ha a fenti beállításokat elvégeztük, és az adatbázisokat létrehoztuk, el kell indítanunk a NIS kiszolgálót.

8.1. tábla: ypserv

NIS szolgáltatást megvalósító program.

ypserv [kapcsolók]

Az ypserv program NIS szolgáltatást valósít meg. A programot általában a szolgáltatásoknál megszokott módon, a /etc/init.d könyvtárban található héjprogrammal indítjuk.

Kapcsoló Jelentés

- | | |
|---------------|--|
| -d | Hibakereső üzemmód a nyomkövetést segítő üzenetek kiírásával. A program a kapcsoló hatására nem démonként, hanem az előtérben fut. |
| -i név | A használt hálózati csatoló nevének megadása. A többi hálózati csatolón a program nem szolgáltat. |



A NIS kiszolgáló indításához Red Hat rendszereken a /etc/init.d/ könyvtárban található ypserv héjprogramot kell futtatnunk a start paraméterrel. A NIS ügyfél indításához Red Hat alapú rendszereken az ypbind programot kell használnunk, amely ugyanebben a könyvtárban található.



A NIS indításához és leállításához Debian alapú GNU/Linux terjesztések esetében a /etc/init.d/ könyvtárban található nis programot kell start, illetve stop paraméterrel elindítani. A program elején található változó határozza meg, hogy NIS kiszolgálót vagy csak NIS ügyfelet akarunk indítani. A beállítást a /etc/default/nis állományban végezhetjük el.

8.2.2. A NIS ügyfél

Működő NIS kiszolgálóhoz a NIS ügyfél segítségével csatlakozva azonnal használhatóvá válnak a kiszolgálóoldalon létrehozott felhasználói nevek, vagyis a kiszolgálóoldali felhasználói nevekkel és jelszavakkal az ügyféloldali számítógépen is be lehet jelentkezni.

A NIS ügyfél használatához a /etc/yp.conf állományt kell helyesen kitöltenünk és az ügyfélprogramot kell elindítanunk. Többféleképpen is megadhatjuk, hogy milyen módon akarunk a NIS kiszolgálóhoz kapcsolódni, de ha csak egy NIS kiszolgálót használunk, akkor a legegyszerűbb, ha a következő mintát használjuk:

1 domain Linux server 10.10.1.1

A példában a 10.10.1.1 IP című NIS kiszolgálóhoz csatlakozunk, amelyen a Linux nevű NIS tartományt használjuk.

Ha a beállítást a fenti módszerrel elvégeztük, elindíthatjuk a gépen a NIS ügyfél-programot, amely lehetővé teszi a felhasználók belépését a távoli gépen érvényes nevükkel és jelszavukkal.



A NIS ügyfél indításához Red Hat alapú GNU/Linux gyűjteményekben a /etc/init.d/ könyvtárban található ypbind nevű programot kell elindítanunk a start paraméterrel. Az ügyfél leállításához ugyanezt a programot kell futtatnunk stop paraméterrel.



Debian alapú rendszereken a NIS ügyfél indításához és leállításához a NIS kiszolgálónál is használt /etc/init.d/ alkönyvtárban található nis programot kell start, illetve stop paraméterrel futtatni.

Ha elindítottuk a NIS kiszolgálót és ügyfelet, a kapcsolat ellenőrzéséhez használhatjuk az ypcat programot, amely a parancssorban megadott adatbázis minden bejegyzését lekéri a kiszolgálóról, és kiírja a szabványos kimenetére.

8.2.3. Jelszóváltoztatás a NIS rendszerben

A passwd parancs nem képes arra, hogy NIS rendszert használó ügyfeleken megváltoztassa a felhasználók nevét, mivel csak helyi állományokat tud megnyitni – nincs felkészítve a NIS alrendszerrel való kapcsolattartásra. A passwd program ráadásul a NIS kiszolgálón is használhatatlan, mivel csak a helyi felhasználói adatbázist változtatja meg, nem frissíti a NIS bináris adatbázisait.

Megoldást az yppasswd program használata adhat. Ez a program képes arra, hogy felvegye a kapcsolatot a NIS kiszolgálóval, és kérje a jelszó megváltoztatását. NIS rendszert használó környezetben szokás a passwd programot helyettesíteni az yppasswd programmal – az eredeti programot felülírni. Így a felhasználók a megszokott módon tudják változtatni a jelszavukat mind az ügyfél-, mind pedig a kiszolgálóoldalon.



Red Hat alapú rendszereken a jelszó megváltoztatásának engedélyezéséhez a szolgáltatások esetében megszokott módon el kell indítanunk az yppasswd szolgáltatást. Más rendszereken ennek a szolgáltatásnak az indítása automatikusan megtörténhet a NIS rendszer indításakor.

Ha a NIS kiszolgálót és NIS ügyfélprogramot is elindítottuk, valamint a jelszó megváltoztatását is engedélyeztük, a felhasználói adatbázis kezelését egy számítógépen, a NIS kiszolgálón elvégezhetjük, ráadásul a felhasználók a megszokott módon használhatják a felhasználói nevüket és jelszavukat.

8.3. Állományok szolgáltatása

Állományok elérhetővé tételere használható az FTP (*file transfer protocol*) protokoll[135], amelyet ma már inkább csak mindenki által elérhető, szabadon letölthető állományok elérhetővé tételere használunk. A következő néhány oldalon arról olvashatunk, hogy miképpen tudunk ilyen szolgáltatást nyújtani.

Az évek során sok FTP szolgáltatást nyújtó program készült a GNU/Linux rendszerekhez, melyek közül mi csak a vsftpd (*very secure FTP daemon*, nagyon biztonságos FTP démon) programot mutatjuk be.

8.3.1. Az FTP kiszolgáló indítása és leállítása

A vsftpd önállóan is képes hálózati kéréseket fogadni, de alkalmas az Internet szuperkiszolgálóval való együttműködésre is. Ha az Internet szuperkiszolgálóval használjuk a programot, a szuperkiszolgáló fogja elindítani, ha pedig önállóan akarjuk használni, a szolgáltatásoknál megszokott módon (1. példa) kell elindítanunk. A program dokumentációja az önálló üzemmódot javasolja.

8.3.2. Az FTP kiszolgáló beállítása

A vsftpd program beállítóállománya a `/etc/vsftpd.conf` állomány, amelyet a `/etc/vsftpd.conf` vagy `/etc/vsftpd/vsftpd.conf` helyen találhatunk meg. A beállítóállományban a szokásos módon, a `#` jel után helyezhetünk el megjegyzéseket. A beállítóállományban a `=` jelrelévrehozott értékadásokat helyezhetünk el, arra azonban ügyelnünk kell, hogy a `=` jel előtt és után nem használhatunk szóközt. A beállítóállomány formáját mutatja be a következő példa:

90. példa. A következő két sor a `/etc/vsftpd.conf` beállítóállomány egy részletét mutatja be:

```

1 #
2 # Az anonymous névvel nem lehet belépni.
3 #
4 anonymous_enable=NO
5 #
6 # A program önállóan fut.
7 #
8 listen=YES

```

Amint láthatjuk, a beállítóállományban az „igen” jelzésére a YES, a „nem” jelzésére a NO szót használhatjuk.

A vsftpd beállítóállományának formájáról, a használható kulcsszavakról a `man vsftpd.conf` parancs begépelésével kaphatunk részletes információt. A legfontosabb kulcsszavak a következők:

anonymous_enable Ha be van kapcsolva, a jelszóval nem rendelkező felhasználók beléphetnek az anonymous vagy az ftp felhasználói név segítségével. Alapértelmezés szerint ez a lehetőség bekapcsolt, tehát kimondottan le kell tiltanunk, ha nem akarjuk használni.

dirmassage_enable Ha be van kapcsolva ez a lehetőség, a program minden könyvtárban keresi – és a felhasználók számára kiírja – a .message állomány tartalmát. Ez a kapcsoló alapértelmezés szerint kikapcsolt.

download_enable Ha kikapcsoljuk ezt a kapcsolót, a kiszolgáló minden állományletöltésre vonatkozó kérést visszautasít.

listen Ha be van kapcsolva ez az üzemmód, a kéréseket a vsftpd program fogadja, nem az Internet szuperkiszolgáló gondoskodik a hálózati kapcsolatok kezeléséről. Alapértelmezés szerint ez az üzemmód kikapcsolt.

local_enable Ha bekapcsoljuk ezt a kapcsolót, a helyi felhasználók a felhasználói nevükkel és jelszavukkal bejelentkezhetnek, hogy állományokat töltse-nek le és fel. Alapértelmezés szerint ez a kapcsoló kikapcsolt.

Fontos megemlítenünk, hogy az FTP szabvány nem használ titkosítást, ezért nagyon körültekintően kell eljárnunk, ha a hálózatunkon felhasználói nevet és jelszót akarunk kérni a felhasználóktól!

no_anon_password Ha ezt a kapcsolót bekapcsoljuk, a program nem kér jelszót az anonymous és a ftp felhasználóktól. Szokás szerint ezektől a felhasználóktól az elektronikus levélcímüket kérjük jelszó gyanánt. Alapértelmezés szerint ez a lehetőség kikapcsolt, a felhasználóktól elektronikus levélcímét kér a program.

ssl_enable Ha ezt a kapcsolót bekapcsoljuk, a vsftpd a felhasználói név és jelszó, a parancsok és az adatok átvitele közben a titkosítást végző SSL programkönyvtárat használja. Ha az FTP kiszolgáló az SSL titkosítást használja, akkor csak olyan programmal lehet elérni az állományokat, amely szintén képes erre (például **ftp-ssl** programcsomag).

A kapcsoló alapértelmezés szerint ki kapcsolt.

Nyilvánvaló, hogy a felhasználói neveket és jelszavakat titkosítás nélkül nem szabad átküldeni olyan hálózaton, amit a rendszergazdán kívül más is használ, az FTP azonban inkább csak nyilvános állományok szolgáltatására használatos. (Ha jelszóval elérhető adatállományokat szeretnénk szolgáltatni az Interneten, akkor inkább az SSH programcsomag használata ajánlható.)

tcp_wrappers Ha bekapcsoljuk ezt a kapcsolót, a program a beérkezett kéréseket a /etc/hosts.allow és /etc/hosts.deny állományok alapján ellenőrzi, mielőtt kiszolgálná. Ez a lehetőség alapértelmezett esetben ki kapcsolt.

xferlog_enable Ha ez a kapcsoló engedélyezett, a program minden állomány-műveletet naplóz a `/var/log/vsftpd.log` állományban. Alapértelmezés szerint ez a lehetőség kikapcsolt.

Mindenképpen javasolható, hogy a naplót bekapcsoljuk, hiszen ellenkező esetben nem fogjuk tudni, hogy milyen állományokat töltöttek le a kiszolgálónkról.

anon_max_rate Az ilyen kulcsszó után megadott szám segítségével korlátozzhatjuk a felhasználói névvel és jelszóval nem rendelkező felhasználónk rendelkezésére álló sávszélességet. A szám megadja, hogy egy másodperc alatt hány bajtnyi adatot tölthetnek le.

Ennek a beállításnak az értéke alapértelmezés szerint 0, ami azt jelzi, hogy nem korlátozzuk a letöltés sebességét.

max_clients Az egyszerre kiszolgálható munkaállomások száma. Alapértelmezés szerint nincs korlát.

anon_root Ha beállítjuk ennek a változónak az értékét, a jelszóval nem rendelkező felhasználók bejelentkezésekor ez a könyvtár lesz a kezdeti munkakönyvtár. (Ez a változó nem biztonsági, csak kényelmi célokra szolgál, a program a felhasználókat a démont futtató felhasználó saját könyvtárába zárja.)

pam_service_name A szolgáltatás neve, amelyet a PAM rendszerben használ a program. Ez a név határozza meg, hogy a PAM melyik beállítóállománya alapján történik a felhasználók azonosítása.

guest_username Annak a felhasználónak a felhasználói neve, akinek a nevében a `vsftpd` fut, amikor saját névvel és jelszóval nem rendelkező felhasználókat szolgál ki. Alapértelmezés szerint ez a felhasználó az `ftp`.

Fontos tudnunk, hogy a felhasználói névvel nem rendelkező, `ftp` vagy `anonymous` felhasználói névvel bejelentkezett felhasználók csak ennek a felhasználónak a saját könyvtárát láthatják! A legtöbb GNU/Linux terjesztés esetében ez a könyvtár a `/home/ftp` vagy a `/var/ftp`.

deny_file Az állományokhoz való hozzáférést meg fogja tagadni a program, ha azoknak a neve illeszkedik az itt állománynév-helyettesítő karakterek segítségével megadott valamelyik mintára. Alapértelmezés szerint ez a változó üres, nincsenek tiltott állománynevek.

hide_file Az állományok nevét, amelyek az itt megadott minták valamelyikére illeszkednek, a program eltitkolja az ellenállomás elől. Alapértelmezés szerint ennek a változónak az értéke is üres.

nopriv_user Annak a felhasználónak a felhasználói neve, amelyiknek a nevében a program fut, amikor nincs szüksége különleges jogokra. Alapértelmezés szerint ennek a változónak az értéke `nobody`.

A következő példa egy egyszerű beállítóállományt mutat be a vsftpd program számára.

91. példa. A következő sorok egy a vsftpd program beállítóállományaként olyan FTP kiszolgálót valósítanak meg, amely csak nyilvánosan elérhető állományok szolgáltatására, anonymous vagy ftp felhasználói névvel használható:

```
/etc/vsftpd.conf
1  #
2  # Egyszerű beállítóállomány anonymous FTP kiszolgáló
3  # üzemeltetéshez.
4  #
5
6  # A programot önállóan futtatjuk.
7  listen=YES
8
9  # Csak jelszó nélkül is elérhető adatokat szolgáltatunk
10 # (ez az alapértelmezés szerinti viselkedés is).
11 anonymous_enable=YES
12 local_enable=YES
13
14 # Naplózzuk a használatot.
15 xferlog_enable=YES
16
17 # Kiírjuk a .message állomány tartalmát.
18 dirmessage_enable=YES
19
20 # Használjuk a hosts.deny és a hosts.allow állományokat.
21 tcp_wrappers=YES
22
23 # Egyszerre csak ennyi ügyfelet fogadunk.
24 max_clients=25
25
26 # A munkakönyvtár kezdetben az ftp felhasználó saját
27 # könyvtárában található pub könyvtár.
28 anon_root=pub
29
30 # A következő állományokat nem mutatjuk és nem szolgáltatjuk.
31 deny_file={*.mp3,*.avi,*.mpg,.message}
32 hide_file={*.mp3,*.avi,*.mpg,.message}
```

A beállítóállomány a megjegyzések és a már tárgyalt ismeretek segítségével könnyen megérthető.

8.4. Az elektronikus levelek fogadása

Az SMTP (*simple mail transfer protocol*, egyszerű levéltovábbító protokoll) elektronikus levelek továbbítására szolgál[133]. Ha ilyen protokollt megvalósító kiszolgálót üzemeltetünk, a felhasználóink elektronikus leveleket kaphatnak más számítógépekről, ezért az SMTP kiszolgáló ma már alapszolgáltatásnak tekintetű.

GNU/Linux rendszereken az SMTP több megvalósítása is elérhető, az egyes terjesztésekben választhatunk, hogy melyik levéltovábbító rendszert használjuk. Az egyik legrégebb óta használt SMTP-megvalósítás a `sendmail`, amelyet hosszú évek óta használnak elektronikus levelek továbbítására[26]. A következő néhány oldalon arról olvashatunk, hogyan állíthatjuk be igényeinknek megfelelően a `sendmail` programot.

8.4.1. A levélfogadás indítása, leállítása és működése

A `sendmail` programot elsősorban levelek fogadására használjuk (bár képes levelek küldésére is). A program szolgáltatásként a szokásos módon indítható (lásd az 1. példát a 14. oldalon), indítása után a beérkező elektronikus leveleket fogadja.

A `sendmail` modern változatai szolgáltatásként indítva két példányban futnak. Az egyik példány feladata a külső forrásból érkező levelek fogadása, a másik példányé pedig a belső forrásból származó levelek fogadása és továbbítása. A két `sendmail`-példány – ahogyan látni fogjuk – egymástól függetlenül működik, mindenek saját beállítóállomány van.

A `sendmail` indítás után az `smtp` szolgáltatás számára fenntartott 25. hálózati kapun várakozik, és fogadja az elektronikus leveleket. Az SMTP protokoll által leírt kapcsolat szöveges, ezért a `telnet` program segítségével csatlakozhatunk és megvizsgálhatjuk a `sendmail` programot működés közben. Ezt mutatja be a következő példa.

92. példa. Kapcsolódunk saját számítógéünk `smtp` szolgáltatást nyújtó hálózati kapujára, és begépelt parancsok segítségével adjunk át egy elektronikus levelet!

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^>'.
220 twinstar.lkk ESMTP Sendmail 8.9.3/8.9.3; Fri, 29 Jun 2001
14:52:36+0200
helo localhost.localdomain
250 pipas-nb.szk Hello localhost.localdomain [127.0.0.1], pleased to meet you
mail from: pipas@amilux.lkk
250 pipas@amilux.lkk... Sender ok
rcpt to: pipas@twinstar.lkk
250 pipas@twinstar.lkk... Recipient ok
data
```

```
354 Enter mail, end with "." on a line by itself
```

```
Probalevel
```

```
Probaszoveg
```

```
.
```

```
250 OAA10504 Message accepted for delivery
```

```
quit
```

```
221 twinstar.lkk closing connection
```

```
Connection closed by foreign host.
```

```
$
```

Amint látjuk, a szöveges kapcsolattartási mód lehetővé tette számunkra, hogy működés közben próbáljuk ki a levéltovábbító rendszert.

A levéltovábbító rendszer parancsairól rövid, de jól használható leírást kapunk, ha begépeljük a help parancsot, amelyet a sendmail értelmez, és amelyre az általa fogadott parancsok listájával válaszol.



A példa nyilvánvalóvá tette, hogy az SMTP protokoll szerint felépített kapcsolat nem ellenőrzi a feladó címét, így egyértelművé válik a számunkra, hogy soha nem lehetünk biztosak az elektronikus levelek feladójának személyében.

A sendmail segítségével a parancssorból egyszerűen küldhetünk elektronikus leveleket. Egyszerűen be kell írnunk a parancsra argumentumként a címzett elektronikus levélcímét, és a program elküldi a szabványos bemenetére érkező sorokat elektronikus levélként.

93. példa. Küldjünk levelet a sendmail program segítségével!

```
$ sendmail joe@gep.lkk <file.txt
```

```
$
```

Amint látjuk, a levélküldés nagyon egyszerű.

A sendmail a beérkező leveleket a /var/spool/mail/ könyvtárban található állományokba rendezi. minden felhasználó – aki kapott már levelet – rendelkezik itt egy állománnyal, amelynek neve megegyezik a felhasználói nevével. Ezek az állományok tartalmazzák az összes beérkező levelet, amelyeket a felhasználók még nem olvastak el.

A beérkezett leveleket tároló állományok tulajdonosa a címzett, így a felhasználó felelős az állomány által foglalt lemezterületért. Ez teszi lehetővé a rendszer-gazda számára, hogy a lemezkorlát segítségével korlátozza a rendszeren tartható olvasatlan levelek mennyiségét.



Közdedvelt támadási módszer az Interneten a nagyméretű elektronikus levelekkel való elárasztás, a levélbombázás, ezért ajánlatos mindenkinék korlátozni a háttértárfelhasználását a /var/spool/mail/ könyvtárat tar-

talmazó lemezrészben. A korlátozás alól a rendszerelő nem lehet kivétel, hiszen a levelekkel elárasztás a legtöbb esetben éppen a rendszerelő elektronikus levelcímének felhasználásával történik.

8.4.2. Beállítóállományok

A sendmail beállítóállományai meglehetősen összetett szerkezetűek, szinte minden rész a helyes beállítás. A rendszerelő feladatának megkönnyítése érdekében a sendmail az m4 program makrónyelvére épülő makróprogramokat használ a beállítóállományok létrehozására.

A sendmail működését befolyásoló legfontosabb állományok a következők:

/etc/mail/sendmail.cf Ez az állomány a sendmail fő beállítóállománya, amely meghatározza a legfontosabb beállításokat és a többi beállítóállomány használatát.

A sendmail a /etc/mail/sendmail.cf beállítóállományt a levelek fogadásakor használja, azaz a sendmail akkor olvassa be a beállításokat ebből az állományból, ha külső forrásból érkező levelek fogadására indítjuk el.

/etc/mail/sendmail.mc A /etc/mail/sendmail.cf beállítóállomány olyan bonyolult, hogy e modern időkben automatikusan eszközökkel állítjuk elő. Amikor a sendmail program beállításait meg akarjuk változtatni, a /etc/mail/sendmail.mc programot megváltoztatjuk, majd ebből az állományból állítjuk elő a /etc/mail/sendmail.cf állományt.

A sendmail.mc állományból az m4 program segítségével hozhatjuk létre a sendmail.cf állományt.

/etc/mail/submit.cf Ezt a beállítóállományt használja a sendmail fő beállítóállományként, ha levelek továbbítására indítottuk el. A sendmail akkor olvassa be a beállításait ebből az állományból, ha a feladata a helyi számítógépen keletkezett levelek továbbítása a külvilág felé.

/etc/mail/submit.mc A /etc/mail/submit.cf állomány előállítására használható állomány.

A submit.mc állományból az m4 program segítségével hozhatjuk létre a submit.cf állományt.

Ha megvizsgáljuk a /etc/mail/ könyvtárat, előfordulhat, hogy egy Makefile állományt találunk benne. A /etc/mail/Makefile projektállomány segítségével a sendmail beállítóállományait újra létrehozhatjuk a make program segítségével, ha tehát módosítottunk valamelyik beállítóállományon, egyszerűen csak el kell indítanunk a make programot a /etc/mail/ könyvtárban.

Az m4 számára készült makróállományok legfontosabb kifejezései a következők:

`include('állománynév')` Ez a kifejezés parancsot ad az állománynévvel megadott állomány betöltésére. A `sendmail` beállítóállományainak elkészítéséhez szükségünk van a `cf.m4` makróállomány betöltésére, ezért ennek a kifejezésnek mindenképpen szerepelnie kell az állomány elején.

A `cf.m4` állomány megkereséséhez használhatjuk az állománykereső segédprogramokat, de tudnunk kell, hogy ha a `cf.m4` állomány és a `sendmail` program nem ugyanabban a programcsomagban található, előfordulhat, hogy a `cf.m4` állomány nincs telepítve, csak a `sendmail` program.

`VERSIONID('szöveg')` A beállítóállomány változatának neve, illetve száma. A kifejezésben szereplő szöveg bekerül az elkészített beállítóállományba, így azonosíthatjuk a beállítóállományt.

Ennek a kifejezésnek nem kell okvetlenül szerepelnie az állományban, de szerencsés, ha nem hagyjuk ki, hogy ne keverjük össze az állományokat.

`OSTYPE('operációs rendszer')` A kifejezés segítségével megadhatjuk, hogy milyen operációs rendszer számára szeretnénk elkészíteni a beállítóállományt, milyen operációs rendszeren szeretnénk futtatni a `sendmail` programot.

Ennek a kifejezésnek mindenképpen szerepelnie kell a makróállományban. GNU/Linux rendszeren a kifejezésben szereplő operációs rendszert a `linux` kulcsszóval jelöljük.

`MAILER(levelező)` A kifejezés segítségével megadhatjuk, hogy a `sendmail` milyen módszert használjon a levelek továbbítására. A kifejezés többször is szerepelhet az állományban, ha többféle levélküldési és fogadási módszert is használni akarunk.

A `MAILER(levelező)` kifejezést – vagy kifejezéseket – a makróállomány végén célszerű használni, amikor az egyéb kifejezések eredményei már elérhetők.

A kifejezésben a levélküldő és fogadó rendszerek helyén a következő kulcsszavakat használhatjuk:

`local` A helyi levelek továbbítására használt módszer, amely az állományokat egyszerűen bemásolja a megfelelő könyvtárba. Erre a levéltoovábbítási módszerre a legtöbb esetben szükségünk van, ezért alapértelmezés szerint automatikusan bekerül a beállítóállományba.

`smtp` A levelek SMTP szerinti továbbítása. Erre a levéltaovábbítási módszerre a legtöbb számítógépen szükségünk van, ezért szerepelnie kell a makróállományban.

`uucp` Levelek továbbítása UUCP (*UNIX to UNIX copy*, másolás UNIX-ról UNIX-ra) protokoll szerint. Ezt a levélküldési módot ritkábban használjuk, a legtöbb esetben nem szükséges szerepelnie a makróállományban.

Ha a `stmp` és a `uucp` levélküldési módot is használjuk, az `smtp` módot írjuk előre!

A `sendmail` dokumentációjában további levéltovábbító és -fogadó eszközökről olvashatunk.

DAEMON_OPTIONS ('kapcsolók') A kifejezés segítségével a levelek fogadására nyitott hálózati kapcsolat tulajdonságait adhatjuk meg különféle kapcsolók segítségével. Az alapbeállítások tulajdonképpen a legtöbb esetben megfelelőek, de előfordulhat, hogy a GNU/Linux terjesztéshez kapott beállítóállomány a levelek fogadását ezzel a kifejezéssel tiltja.

Ha a sendmail nem fogad külső kapcsolatokat, tehát csak a saját gépünkéről lehet csatlakozni hozzá, nagy valószínűséggel segít, ha megkeressük ezt a kifejezést a makróállományban, eltávolítjuk, újra létrehozzuk a program beállítóállományát, és újraindítjuk a szolgáltatást.

A kifejezésben vesszővel elválasztva adhatunk meg értékkedásokat (például `Port=smtp`, `Addr=127.0.0.1`, `Name=MTA`). Az értékadásban a következő kulcsszavak szerepelhetnek:

Name=név A kifejezéssel beállítandó szolgáltatás neve. A név többek között lehet:

MTA a külső számítógépekről beérkező levelek fogadását végző `sendmail` (*message transfer agent*)

MSA a kifelé menő üzeneteket továbbító `sendmail` (*message submission agent*).

Port=szám A kifejezéssel megadhatjuk, hogy melyik hálózati kapun nyújtuk a szolgáltatást. A kifejezésben szereplő számot a kapuhoz rendelt szolgáltatás nevével is jelölhetjük.

Addr=IP_cím A kifejezés segítségével megadhatjuk, hogy mely címekről fogadunk kéréseket.

94. példa. Készítsük el a lehető legegyszerűbb makróállományt a `sedmail.cf` létrehozására, majd készítsük el a `sendmail` beállítóállományát (a példában a szöveges állandókat egyszeres idézőjelek közé helyeztük, ami nem kötelező)!

```
/etc/mail/sendmail.mc
1 dnl
2 dnl sendmail.mc: Makróállomány sendmail-hez
3 dnl Fordítás:
4 dnl           m4 sendmail.mc >sendmail.cf
5 dnl
6 include('/usr/share/sendmail-cf/m4/cf.m4')
7 VERSIONID('proba')
8 OSTYPE('linux')
9 MAILER(smtp)
```

A példában szerepel az `include` kulcsszó a makrócsomag betöltésére, a változatszámot meghatározó `VERSIONID` kulcsszó (ez nem kötelező), az `OSTYPE` az operációs rendszer típusával és a `MAILER` a használandó levéltovábbítási rendszer kijelölésére. Az állomány ebben a formában működőképes beállítást hoz létre.

A bemutatott állomány jóval egyszerűbb, mint a legtöbb *GNU/Linux* terjesztés által alapértelmezés szerint használt beállítóállomány, ez azonban természetesen nem jelenti azt, hogy le kell cserélnünk azt erre. A példánk azért ilyen egyszerű, hogy könnyen megérthessük a felépítését.

A beállítóállomány ezek után könnyedén elkészíthető az `m4` segítségével:

```
$ m4 sendmail.mc >sendmail.cf  
$
```

Ezek után a `sendmail` programot a szolgáltatások esetében megszokott módon indíthatjuk.

8.4.3. A szolgáltatások engedélyezése és tiltása

A `sendmail` beállítóállományaiban könnyen beállíthatjuk, hogy a leveleket küldő külső számítógépek milyen elbánásban részesüljenek. Beállíthatjuk, hogy melyek azok a számítógépek, amelyektől nem fogadunk el leveleket, melyek azok, amelyek számára különleges szolgáltatásként levéltovábbítást végzünk és így tovább. A következő néhány bekezdésben arról lesz szó, hogy miképpen állíthatjuk be a levelet küldő számítógépek számára nyújtott szolgáltatásokat, a küldő oldal jogait.

Nagyon fontos, hogy megértsük a levél továbbküldésének (*relaying*) szolgáltatását. Ez a szolgáltatás lehetővé teszi a küldő számítógép számára, hogy megkérje a mi levéltovábbító rendszerünket, hogy a levelet egy harmadik számítógép felé továbbítsa.

A levéltovábbküldés az olyan primitív munkaállomások számára hasznos, amelyek nem képesek maguk továbbítani a leveleket. Ezek a munkaállomások egy „okos” levéltovábbküldő számítógép” (*smart relay host*) segítségére szorulnak, amely helyettük továbbítja az elektronikus leveleket.

A levelek továbbküldése azonban komoly veszélyeket is rejt magában, hiszen a kéretlen reklámleveleket küldő társaságok ezt a szolgáltatást kihasználva a mi számítógépunkon keresztül kísérelhetnek meg leveleket küldeni. Fontos, hogy a levelek továbbküldését csak a saját magunk által üzemeltetett munkaállomásoknak engedélyezzük, és azoknak is csak akkor, ha mindenkiéppen szükségük van erre a szolgáltatásra.



A mai időkben, amikor a kéretlen levelek küldői, a vírusok és a levelezőrendszereket üzemeltető rendszergazdák közötti háborúság hihetetlen mértékben kiterjedt és eldurvult, a levéltovábbító alrendszer biztonsági beállításai igen fontosak. A `sendmail` mai változatai szigorú beállításokkal tele-

pülnek minden GNU/Linux terjesztésben, mégis fontos, hogy figyeljünk a biztonságra. Igen fontos, hogy ismeretlen számítógépek számára ne nyújtsunk levélto-vábbítási szolgáltatást és folyamatosan figyeljünk a levelezőrendszer forgalmára!

A sendmail.mc állományban a következő kifejezéseket használhatjuk a legfontosabb biztonsági beállítások létrehozására:

FEATURE(access_db) Ha ez a kifejezés szerepel a sendmail.mc állományban, a sendmail figyelembe veszi a /etc/mail/access.db állományt, amely az egyes leveleket küldő számítógépek jogait sorolja fel. A küldő számítógépek jogainak korlátozása nagyon fontos, ezért ezt a kifejezést a legtöbb esetben használjuk.

A /etc/mail/access.db állomány előállítására a későbbiekben részletesen visszatérünk.

FEATURE(relay_hosts_only) Alapértelmezés szerint a sendmail a /etc/mail/access.db állomány bejegyzéseit DNS tartományokra is értelmezi. Ha ez a kifejezés szerepel a sendmail.mc állományban, a bejegyzéseket a sendmail csak számítógépnévként értelmezi, azaz minden számítógépet egyenként fel kell sorolni.

Ezzel a kifejezéssel tehát szigorúbbá tehetjük a /etc/mail/access.db értelmezését.

FEATURE(relay_entire_domain) Ha ez a kifejezés szerepel a sendmail.mc állományban, a levelek továbbítása engedélyezett lesz a számítógéppel azonos DNS tartományban található számítógépek számára.

Ugyanezt a hatást úgy is elérhetjük, hogy a saját tartományunkat a továbbküldés engedélyezésével adjuk meg a /etc/mail/access.db állományban.

Amint láttuk a levélküldő számítógépek jogainak leírása alapértelmezés szerint a /etc/mail/access.db bináris adatbázis-állományban található, amelyet a make programmal hozhatunk létre, a /etc/mail/Makefile alapján a /etc/mail/access állományból. (Valójában az állományt a makemap program hozza létre, lásd a 99. példát a 265. oldalon.)

A /etc/mail/access szöveges állomány két oszlopot tartalmaz. A két oszlop jelentése a következő:

1. Az állomány első oszlopában a tartománynév vagy számítógépnév található, amelyre az adott sor vonatkozik.

A sendmail az első illeszkedő sort veszi figyelembe a küldő számítógép jogainak eldöntésekor, így mindenkor a szűkebb névtartományt megadó sort érdemes előre írni az állományban.

2. A második oszlop megadja, hogy milyen elbírálás alá esik az adott névre illeszkedő nevű küldő. Itt a következő kulcsszavakat használhatjuk:

REJECT A tartományból nem fogadunk elektronikus leveleket, a levelet küldő számítógépet visszautasítjuk.

550 Az adott tartományból nem fogadunk leveleket, az elutasító válaszba pedig a sendmail beírja az 550 után írt szöveget.

Ez a kulcsszó hatásában megegyezik a REJECT hatásával, azzal az apró különbséggel, hogy magunk határozzuk meg a hibaüzenet szövegét. A hibaüzenet szövegét a levél feladója megkapja egy válaszlevélben, amelyet a kézbesíthetetlenségről kap a saját levelezőrendszerétől.

DISCARD A tartományból nem fogadunk elektronikus leveleket, de a küldő számítógépet nem utasítjuk vissza. A levelet átvesszük, az átvételt visszaigazoljuk, majd a levelet nemes egyszerűséggel eldobjuk.

OK A tartományból a leveleket elfogadjuk. Ha a tartományba található valamelyik számítógép levelet küld valamelyik helyi felhasználónak, a levelet elfogadjuk és megkíséreljük elhelyezni a felhasználó levelesládájában.

A jogok vizsgálatánál az OK az alapértelmezés szerinti érték, vagyis mindenki számára engedélyezett a levelek küldése azoknak a helyi számítógépen található felhasználóknak, akik nem szerepelnek az access.db állományban.

RELAY A tartományból a levelet akkor is elfogadjuk, ha azt nem helyi felhasználónak küldték, azaz vállaljuk a tartomány számára a levelek továbbítását.

Ezt a jogot csak olyan számítógépek számára állítsuk be, amelyek a mi fennhatóságunk alá tartoznak, ellenkező esetben könnyen lehet, hogy kérülten reklámlevelek továbbítására használják fel a számítógépünket, és ennek következtében mindenhonnan kitiltják a leveleinket.

A levelek továbbítására primitív levéltovábbító rendszert futtató munkaállomásoknak van szükségük, amelyek nem képesek önállóan kézbesíteni a leveleket.

A szolgáltatások engedélyezése kapcsán meg kell említenünk, hogy az access állomány első oszlopában található tartománynév nem a levél feladójának elektronikus levélcímében szereplő tartománynevet jelenti, hanem a levelet küldő számítógép nevét. A sendmail megvizsgálja, hogy annak a számítógépének mi a neve, amelyik felvette vele a kapcsolatot, és ezt a nevet keresi az access állományban. Ez természetes is, hiszen – amint azt láttuk – a levél feladójának levélcíme könnyen hamisítható, megbízhatatlan adat.

A következő példa bemutatja, miképpen állíthatjuk be a levelet küldő kiszolgálók jogait.

95. példa. Szeretnénk bizonyos névtartományok jogait korlátozni a levélküldés során. Ehhez első lépésként ellenőrizzük, hogy a beállítóállományban az access.db használata be van-e állítva!

```
$ grep access sendmail.mc
FEATURE('access_db', 'hash -T<TMPF> -o /etc/mail/access.db')dnl
$
```

Amint látjuk, az `access.db` állomány használata be van állítva a makróállományban, amelyből a `sendmail` beállítóállománya készült.

A beállítóállományban a következő szabályokat helyezzük el:

/etc/mail/access	
1	localhost.localdomain RELAY
2	localhost RELAY
3	127.0.0.1 RELAY
4	jo.pelda.hu OK
5	pelda.hu REJECT
6	ellenseg.com 550 Tunes innen, nem latunk szivesen!

Amint látjuk, vállaljuk a levelek továbbítását a helyi számítógép, és fogadjuk a `jo.pelda.hu` számítógép leveleit a helyi felhasználók számára. Nem fogadjuk viszont a `pelda.hu` tartomány más számítógépeinek leveleit (5. sor). A példában szereplő `ellenseg.com` tartomány számítógépeinek közeledését csípős megjegyzéssel elutasítjuk, bármit is szeretnének elérni.

Az állomány szerkesztése után a `access.db` állományt frissítenünk kell a `make` programmal a `Makefile` alapján.

8.4.4. Idegen levelek fogadása

A következő néhány bekezdésben arról olvashatunk, hogy miképpen fogadhatunk olyan leveleket, amelyekben a címzett elektronikus levélcíme nem egyezik meg a számítógépünk nevével.

Gyakran előfordul, hogy a számítógépünk több tartomány levelezésért felelős. Egyszerű esetben a felhasználók listája, a postaládák minden tartományban meggyeznek, egyszerűen csak annyit szeretnénk, ha a leveleket egy helyre lehetne irányítani, egy számítógépen lehetne fogadni.

Ebben a munkában az első lépés az, hogy a számítógép számára több DNS bejegyzést készítünk. Ha a küldő oldal a levél küldése előtt lekérdezi a DNS kiszolgálót, könnyen megállapíthatja, hogy hova kell küldenie a levelet. A fogadó oldalon azonban nem ilyen egyszerű a helyzet.

Tegyük fel, hogy a számítógép, amelyen a `sendmail` programot futtatjuk, a `mail.akarmi.hu` elsődleges tartománynévvel van ellátva, de szeretnénk az `@akarmi.hu` elektronikus levélcímekre érkező leveleket is fogadni. Ha azonban minden figyelmeztetés nélkül beérkezik egy levél a `jozsi@akarmi.hu` címre egy távoli számítógépről, a `sendmail` nem fogadja. Azt a hibaüzenetet fogja adni, hogy a levelek továbbítását nem vállalja a küldő számítógép számára, azaz nem ismeri fel, hogy a levél éppen neki szól.

Az idegen feladójú levelek fogadását a `sendmail.mc` következő kifejezésével kérhetjük:

FEATURE(use_cw_file) A kifejezés hatására az elkészített beállítóállományba olyan parancs kerül, amelynek hatására a sendmail figyelembe veszi a /etc/mail/local-host-names állományt, amelybe azokat a DNS tartományneveket írhatjuk, amelyek számára leveleket fogadunk.

96. példa. Szeretnénk ha a mail.akarmi.hu számítógép az akarmi.hu és a masvalami.hu végződésű elektronikus leveleket is fogadná. Ellenőrizzük, hogy a sendmail.mc állományban megtalálható-e a megfelelő bejegyzés!

```
$ grep cw_file /etc/mail/sendmail.mc
```

```
FEATURE(use_cw_file)dnl
```

```
$
```

Ha igen akkor a local-host-names állományra való hivatkozás már valószínűleg bekerült a beállítóállományba:

```
$ grep local-host-names /etc/mail/sendmail.cf
```

```
Fw/etc/mail/local-host-names
```

```
$
```

Töltsük ki a /etc/mail/local-host-names állományt a következőképpen:

- | | |
|---|--------------|
| 1 | akarmi.hu |
| 2 | masvalami.hu |

Az állomány mentése után a sendmail a felsorolt végződésű levélcímekre érkező leveleket fogadja, és megpróbálja kézbesíteni a helyi felhasználók postaládáiba.

8.4.5. Látszólagos felhasználók

A látszólagos felhasználó (*virtual user*) kifejezés olyan felhasználói postaládára utal, amelyhez nem tartozik azonos nevű valódi felhasználó. Ilyen látszólagos felhasználóra általában akkor van szükségünk, ha a számítógépünk több tartománynével is fogad elektronikus leveleket. Ezt az esetet mutatja be a következő példa.

97. példa. Tegyük fel, hogy két tartománynév számára is fogadunk elektronikus leveleket. A /etc/mail/local-host-names állományunk a következőképpen van kitöltve:

- | | |
|---|-----------|
| 1 | domain.hu |
| 2 | dmn.hu |

Ha ezek után egy elektronikus levél érkezik a pip@domain.hu levélcímre, azt a sendmail a pip helyi felhasználónak kézbesíti. Ha egy következő levél címzettje a pip@dmn.hu, a levelet a sendmail elfogadja, és szintén a pip helyi felhasználónak adja át.

Tisztán látszik, hogy a két tartománynév nincs egymástól elszigetelve, a tartománynevek ugyanahhoz a postaládákhöz vezetnek!

A sendmail képes látszólagos felhasználókat kezelni, az elektronikus levélcímeket a helyi felhasználók postaládához rendelő táblázat alapján. A következő kifejezéssel kérhetjük ezt a szolgáltatást.

FEATURE(virtusertable) Ha ezt a kifejezést elhelyezzük a `sendmail.mc` állományban, a belőle készített beállítóállomány olyan utasítást fog tartalmazni, amelynek hatására a sendmail figyelembe veszi a látszólagos felhasználók adatait tartalmazó táblázatot.

A látszólagos felhasználók adatait tartalmazó adatbázis alapértelmezés szerint a `/etc/mail/virtusertable` szöveges állományban található, amelyből elkészíthető a `/etc/mail/virtusertable.db` bináris adatbázis-állomány.

A levelezésért felelős rendszergazdának a feladata, hogy a `/etc/mail/virtusertable` állományban elhelyezze azokat a szöveges sorokat, amelyek a látszólagos felhasználókhöz (az elektronikus levélcímekhez) valódi felhasználókat rendelnek. Az állományban a szokásos módon helyezhetünk el megjegyzéseket a # jel után, az adatokat pedig tabulátor karakterekkel elválasztott kétoszlopos formában írhatjuk be.

A virtusertable állomány első oszlopa tartalmazza az elektronikus levélcímet, a második oszlopa pedig a valódi felhasználó felhasználói nevét. Az egyszerű cím–felhasználó egymáshoz rendelésén kívül néhány egyéni eszköz is a rendelkezésünkre áll. Ezeket a következő példa mutatja be.

98. példa. A következő néhány sor a virtusertable állományból származik. Értelemzzük a sorokat!

	/etc/mail/virtusertable
1	#
2	# Látszólagos felhasználók adatbázisa
3	#
4	pip@domain.hu jkovacs
5	pip@dmn.hu jpista
6	nem@domain.hu error:nouser Nincs ilyen felhasznalo
7	@masdomain.hu %1@masdn.hu

Az állomány első három sora megjegyzés, az adatok a negyedik sorban kezdődnek.

A 4. és 5. sorok egy-egy egyszerű elektronikus levélcímet rendelnek létező felhasználók postaládáihoz. Figyeljük meg, hogy a levélcímek felhasználói nevet tartalmazó részei megegyeznek, a levelek mégis más-más felhasználó postaládáiba kerülnek. Ezt a szolgáltatást csak a látszólagos felhasználók bevezetésével tudjuk nyújtani.

A példa 6. sorában egy olyan bejegyzést találunk, amely a levelet visszautasítja, jelezve a küldőnek, hogy nincs ilyen felhasználó a nyilvántartásban. Ezen

beállítások alapján tehát, ha a `nem@dmn.hu` címre érkezik levél, azt a `nem` nevű helyi felhasználó kapja meg. Ha viszont a `nem@domain.hu` címre érkezik levél, azt a megfelelő hibaüzenettel visszaküldjük.

A 7. sorban egy olyan bejegyzést találunk, amely több elektronikus levélcímre is illeszkedik, hiszen az első oszlopban található cím nem teljes (a @ jellel kezdődik).

A 7. sor bejegyzése abból a szempontból is különleges, hogy a második oszlopban nem helyi felhasználó neve áll, hanem egy távoli levélcím (megtalálható benne a @ karakter). A második oszlop levélcímének felhasználói nevet jelző részének helyén a %1 található, amely az eredeti cím első mezőjét jelenti. A 7. sor címezettje tehát ugyanaz a felhasználó lesz, aki eredetileg is volt, a levelet azonban továbbítja a levelezőrendszer az új tartománynévvel jelzett számítógép felé.

Ha a látszólagos felhasználók táblázatát a minta alapján elkészítjük, a `virtusertable` állományból előállíthatjuk a `virtusertable.db` állományt a `makemap` vagy a `/etc/mail/` könyvtárban indított `make` programmal, amely a `Makefile` állomány alapján végzi feladatát. A `makemap` program használatát mutatja be a következő példa.

99. példa. A `makemap` program használata igen egyszerű. Paraméterként meg kell adnunk, hogy a bináris adatbázist milyen állományformátumban kívánjuk létrehozni és azt, hogy az adatbázist melyik állományban kívánjuk elhelyezni. A program az adatbázist a szabványos bemenetről olvassa. A `makemap` által kezelt állományformátumokról a `man makemap` parancs begépelésével tudhatunk meg többet.

A következő parancs a látszólagos felhasználók adatbázisát állítja elő a szöveges állományból:

```
$ makemap hash virtusertable.db < virtusertable  
$
```

A beállítások után a `sendmail` a megfelelő módon kezeli a beérkezett leveleket, bár hasznos lehet újraindítani, hogy biztosak lehessünk a dolgunkban.

8.5. A webkiszolgáló

A webkiszolgáló a számítógépekre érkezett, HTTP (*hypertext transfer protocol*, hiperszöveg-átviteli protokoll)[8, 48, 173] protokollnak megfelelő kéréseket szolgálja ki. A HTTP protokoll segítségével a legtöbbször nyilvános, multimédiás (szöveg, kép, mozgókép és hang) jellegű adatokat szokás szolgáltatni.

Az Interneten a legelterjedtebb webkiszolgáló az Apache, amelyet többek között GNU/Linux rendszerekre is telepíthetünk. A legtöbb GNU/Linux terjesztés tartalmazza. A következő oldalakon arról olvashatunk, miképpen állíthatjuk be az Apache webkiszolgálót igényeinknek megfelelően.

8.2. tábla: httpd

Az Apache webkiszolgálót megvalósító program.

`httpd [kapcsolók]`

A `httpd` program az Apache része és a HTTP protokoll alapján működő kiszolgálót valósít meg. A `httpd` program indítását általában a szolgáltatások esetében megszokott módon, a `/etc/init.d/könyvtárban` található héjprogram végzi.

Kapcsoló	Jelentés
<code>-f</code> fájl	A beállítóállomány megadása.
<code>-e</code> szám	A naplózás részletességeinek beállítása. Általában hibakereséskor vesszük hasznát e kapcsolónak.
<code>-l</code>	A kiszolgálóprogramba beépített modulok nevének kiírása.
<code>-t</code>	A program nem indul el, csak ellenőri a beállítóállományt, és azonnal kilép.
<code>-X</code>	Hibakereső üzemmód, a program csak egy példányban indul el, és az előtérben fut.

8.5.1. A webkiszolgáló indítása és leállítása

Az Apache webkiszolgálót megvalósító program a `httpd`, amelynek indítása és leállítása a szolgáltatások esetében megszokott módon (1. példa) történik.

8.5.2. A webkiszolgáló alapbeállításai

A különféle GNU/Linux terjesztésekben az Apache beállítóállománya más-más könyvtárakban található.

Debian GNU/Linux terjesztés esetén az Apache beállítóállományainak könyvtára a `/etc/apache/`, Red Hat alapú GNU/Linux terjesztésnél pedig a `/etc/httpd/`. Az Apache fő beállítóállománya a könyvtáron belül a `conf/httpd.conf` állomány.

Igen fontosak az Apache naplóállományai, amelyekből értesülhetünk a hibáról. Debian GNU/Linux terjesztés esetén ezeket a naplóállományokat a `/var/log/apache/`, Red Hat GNU/Linux terjesztés esetén pedig a `/var/log/httpd/` könyvtárban találjuk.

A `httpd.conf` beállítóállományban a szokásos módon – a `#` karakter segítségével – helyezhetünk el megjegyzéseket. A beállítóállomány kifejezéseiiben a könyvtár- és állományneveket – ha azok szóköz karaktert tartalmaznak – "" közé kell írnunk. A kifejezésekben szereplő könyvtárnevek végére a legtöbb helyen nem szabad / karaktert írnunk.

Az Apache `httpd.conf` beállítóállományában használható legfontosabb kifejezések a következők:

ServerRoot "könyvtárnév" A könyvtár neve, ahol a webkiszolgáló beállítóállományai és naplóállományai találhatók. A legtöbb GNU/Linux gyűjtemény esetében ez a könyvtár a /etc/httpd/ vagy a /etc/apache/.

PidFile "állománynév" Az állomány neve, amelyben a webkiszolgáló a futása alatt a folyamataazonosítóját (PID) tárolja.

Listen IP_cím:kapu A kifejezés segítségével megadhatjuk, hogy a webkiszolgáló melyik hálózati eszközön (melyik címen) és melyik hálózati kapun fogadja a kéréseket. Ha a kifejezésből az IP címet elhagyjuk, a kiszolgáló minden rendelkezésre álló hálózati eszközön fogadja a kéréseket. A webkiszolgáló szokás szerint a 80-as, esetleg a 8080-as hálózati kapun keresztül fogadja a kéréseket a webböngészőktől.

Ennek a kifejezésnek szerepelnie kell a beállítóállományban.

User felhasználónév A kifejezés segítségével megadhatjuk, hogy a webkiszolgáló melyik felhasználó nevében fusson.

Group csoportnév A kifejezés segítségével megadhatjuk, hogy a webkiszolgáló melyik felhasználói csoport nevében fusson.

ServerAdmin levélcím A kifejezés segítségével megadhatjuk a webkiszolgálóért felelős webmester elektronikus levélcímét.

A webkiszolgáló ezt a címet az általa létrehozott hibajelentő lapokon elhelyezi, így a cím nyilvánossá válhat.

ServerName név:kapu A kifejezés segítségével megadhatjuk a webkiszolgáló nevét, ami megegyezik a számítógép nevével.

Ha nem adjuk meg ezt a kifejezést a beállítóállományban, a webkiszolgáló induláskor a szokásos módon lekérdezi a számítógép nevét és azt használja. Ha a webkiszolgálót a rendszer indításakor automatikusan indítjuk, érdekes megadni ezt a kifejezést, hogy elkerülhessük a felesleges várakozást, ha a DNS kiszolgáló nem érhető el.

DocumentRoot "könyvtárnév" A kifejezés segítségével megadhatjuk, hogy melyik könyvtárban találhatók a webkiszolgáló által szolgáltatott dokumentumok.

ErrorLog "állománynév" A kifejezés segítségével megadhatjuk, hogy a webkiszolgáló melyik állományban helyezze el a naplóbejegyzéseket.

Az Apache saját maga kezeli a naplóbejegyzéseket – nem használja a rendszernaplót –, ezért ha a kifejezésben megadott állománynév nem / karakterrel kezdődik, a naplóállomány a ServerRoot kifejezésben megadott könyvtáron belül lesz.

`DefaultType` típus A kifejezés segítségével megadhatjuk, hogy a webkiszolgáló alapértelmezés szerint milyen állománytípushat jelentsen a webböngésző számára.

A kifejezésben szereplő típus az átküldött adatok MIME (*multipurpose internet mail extension*, többcélú internethasználati levélkiegészítés)[49] típusa, ami lehet például `text/plain` vagy `text/html`.

A bemutatott kifejezések felhasználásával elkezthetjük egy egyszerű, de működőképes webkiszolgáló beállítóállományát. Ezt mutatja be a következő példa.

100. példa. Készítsünk egyszerű beállításokat az Apache webkiszolgáló számára!

```
1  #
2  # A fontosabb állományok és könyvtárak helye:
3  #
4  ServerRoot "/etc/httpd"
5  DocumentRoot "/var/www/html"
6  PidFile "run/httpd.pid"
7
8  #
9  # A szolgáltatás nyújtására használt hálózati kapu
10 #
11 Listen 80
12
13 #
14 # A programot futtató felhasználó és csoport
15 #
16 User "apache"
17 Group "apache"
18
19 #
20 # A webmester
21 #
22 ServerAdmin root@www.1kk
23
24 #
25 # Az alapértelmezett állományformátum html
26 #
27 DefaultType text/html
```

A bemutatott beállításokkal a kiszolgáló elindul és működik, de túl sok szolgáltatást nem várhatunk tőle.

8.5.3. Az Apache moduljai

Az Apache programcsomag sok kiegészítőmodult tartalmaz, amelyeket csak akkor kell betöltenünk, ha az általuk megvalósított szolgáltatásokra szükségünk van. A modulok betöltésére a `httpd.conf` állományban a következő kifejezést használhatjuk:

`LoadModule név állománynév` A kifejezés hatására a webkiszolgáló megkísérli modulként betölteni az állományban található programot.

A kifejezésben a *név* a modulváltozó neve, amely a modul tárolására használt állományban található. A modulváltozó neve az Apache modulok esetében a modulnévből a `_module` végződés hozzáillesztésével kapható meg (*modulnév_module*).

A kifejezésben az *állománynév* a modult tároló állomány neve. Az állománynév megadható abszolút elérési úttal vagy a `ServerRoot` kifejezéssel megadott könyvtárhoz képest.

Az Apache modulokat tároló állományok általában a `/etc/httpd/` könyvtár `modules` alkönyvtárában vannak, nevük a modulnévből a `mod_` előtag és a `.so` végződés hozzáillesztésével kapható meg (`mod_modulnév.so`)

Az Apache számára készült modulok közül a legfontosabbak a következők:

`access` A modul segítségével a webböngésző adatai alapján dönthetjük el, hogy az adott állományhoz vagy könyvtárhoz biztosítjuk-e a hozzáférési jogot.

A modul használatáról és beállításairól bővebben olvashatunk a 278. oldalon kezdődő 8.5.5. szakaszban.

`actions` A modul segítségével CGI programokat futtathatunk, ha a webböngésző adott típusú állományt kér, vagy adott típusú tranzakciót kezdeményez (adatok letöltése, adatok küldése, állomány küldése).

`alias` A modul lehetővé teszi, hogy a helyi állományok elérésére álneveket hozunk létre, a webböngésző kéréseit a könyvtárszerkezet különféle helyeire irányítva.

`asis` A modul segítségével elérhetjük, hogy bizonyos állományokat a webkiszolgáló a szokásos fejléc nélkül küldje el. Az ilyen állományok első soraiban a megfelelő fejlécnek kell szerepelnie.

`auth` A modul segítségével egyszerű szöveges állományokon alapuló felhasználóazonosítást használhatunk.

`auth_anon` A modul segítségével bárki számára elérhető erőforrásokat szolgáltathatunk az *anonymous* FTP kiszolgálókon megsokott módszerrel, vagyis az *anonymous* felhasználói név, illetve jelszóként az elektronikus levélcím begépelésével.

auth_dbm A modul felhasználóazonosítást tesz lehetővé DBM típusú adatbázis-állományokban tárolt felhasználói adatok felhasználásával.

autoindex A modul betöltése után a könyvtárak lekérdezésekor automatikusan létrehozott listát küldhetünk a böngészőnek. A lista a könyvtárban található könyvtárbejegyzések legfontosabb adatait tartalmazza, és a segítségével az egyes könyvtárbejegyzések a listából kiválasztva elérhetők a böngésző számára.

A modul használatára részletesebben kitérünk a 273. oldalon kezdődő 8.5.4. szakaszban.

cgi A modul segítségével CGI programokat futtathatunk, és az eredményt elküldhetjük a webböngészőnek.

A CGI programokról bővebben a 283. oldalon kezdődő 8.5.6. szakaszban olvashatunk.

cgid A modul segítségével bizonyos rendszereken gyorsabban, kevesebb erőforrás felhasználásával futtathatjuk a CGI programokat. A modul működésében és használatában szinte teljesen megegyezik a `mod_cgi` modullal.

charset_lite A modul lehetővé teszi, hogy a szolgáltatott állományok tartalmát módosított karakterkódolással küldjük el a böngészőnek.

dav A modul segítségével a HTTP kapcsolat DAV (*distributed authoring and versioning*, elosztott fejlesztés és változatkövetés) kiegészítése is elérhetővé válik a webkiszolgálón[2].

deflate A modul tömörítést valósít meg a kiszolgáló és a böngésző között.

dir A modul hatására a webkiszolgáló a könyvtárakban található tárgymutató-állományokat (*index files*) küldi el, ha a kért erőforrás egy könyvtár neve.

A modul használatára részletesebben kitérünk a 273. oldalon kezdődő 8.5.4. szakaszban.

env A modul segítségével a CGI programok számára indításkor átadott környezeti változók értékét módosíthatjuk tetszésünk szerint.

expires A modul felhasználásával beállíthatjuk, hogy a böngésző által helyi gyorsítótárba mentett állományok mikor veszítsék el érvényességüket.

ext_filter A modul segítségével az állományok tartalmát a webböngészőnek való elküldés előtt tetszőleges szűrőprogrammal módosíthatjuk. Bármely programot használhatjuk, amelyik az adatokat a szabványos bemenetről olvassa, és az eredményeket a szabványos kimenetre írja.

file_cache A modul használatával előírhatjuk, hogy a webkiszolgáló bizonyos állományokat a gyorsabb működés érdekében folyamatosan a memoriában tartson.

headers A modul segítségével a webkiszolgáló által az állományok küldésekor használt fejléceket módosíthatjuk, befolyásolhatjuk.

imap A modul .map állományok feldolgozását végzi. A .map állományok segítségével olyan képeket helyezhetünk el a weben, amelyek egyes részeire kattintva más-más oldalakat látogathat meg az olvasó.

include A modul segítségével feldolgozhatjuk a weblapok tartalmát az elküldés előtt, és a bennük elhelyezett különleges utasításokat végrehajthatjuk. A modul a HTML SSI (*server side includes*, kiszolgálóoldali beágyazások) kiterjesztését valósítja meg.

info A modul segítségével szimulált könyvtár szolgáltatását kérhetjük. A szimulált könyvtár a kiszolgáló beállításait szolgáltatja a böngésző számára.

log_config A modul felhasználásával lehetővé válik a webböngészők kéréseinek pontos és rugalmasan állítható naplázása.

mime A modul segítségével a különféle állománytípusokat és formátumokat kezelhetjük kifinomult módon.

mime_magic A modul lehetővé teszi, hogy a szolgáltatott állományok típusát ne a nevük, hanem a tartalmuk alapján állapítsa meg a webkiszolgáló.

A modul használatára a 276. oldalon visszatérünk.

negotiation A modul lehetővé teszi, hogy a böngésző a közte és a kiszolgáló között zajló tárgyalás alapján a neki legmegfelelőbb állományformátumokat kapja.

rewrite A modul lehetővé teszi, hogy a körülményektől függő módon átalakítsuk a webböngésző által kért webcímét, mielőtt kiszolgálnánk.

setenvif A modul segítségével a webböngésző kérésének (és főképpen típusának) megfelelően környezeti változók értékét állíthatjuk be. Ilyen módon a webkiszolgáló különféle webböngészők kérésére különféleképpen reagálhat.

so A modul segítségével megosztott könyvtárakat tölthetünk be.

speling A modul megkíséri a felhasználó által hibásan begépelt weblapok címeit kijavítani. A modul képes kijavítani a kisbetűk és nagybetűk összekerverését és az egybetűs elgépeléseket.

ssl A modul titkosítást valósít meg a webböngésző és a webkiszolgáló között.

status A modul a weben elérhető, könnyen érthető adatokat szolgáltat a webkiszolgáló működéséről, terheléséről.

suexec A modul segítségével meghatározhatjuk, hogy az egyes CGI programok melyik felhasználó és melyik csoport nevében fussanak.

userdir A modul lehetővé teszi, hogy a felhasználók weblapokat szolgáltassanak a saját könyvtárunkból.

usertrack A modul segítségével nyomon követhetjük a webböngészőt kezelő felhasználók tevékenységét.

vhost_alias A modul segítségével a böngésző által a kérésben használt kiszolgálónévtől tehetjük függővé, hogy milyen weblapot szolgáltatunk.

Néhány modult nem kell betöltenünk az Apache beállítóállományában, mert azok a webkiszolgáló bináris állományába eleve bele vannak szerkesztve. Az állományba szerkesztett modulok listáját a httpd program -l kapcsolójával kaphatjuk meg, ahogyan ezt a következő példa bemutatja.

101. példa. Írassuk ki az Apache webkiszolgáló programállományába szerkesztett modulok listáját! Használjuk a -l kapcsolót!

```
$ httpd -l
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
$
```

Amint látjuk, a modulok neve után a .c végződés található.

Az Apache modulok általában külön kifejezések segítségével állíthatók be, amelyek a modulok betöltése nélkül nem használhatók.

Annak érdekében, hogy a modulok beállítására szolgáló kifejezéseket a kiszolgáló ne vegye figyelembe, ha a modul nincs betöltve, a <IfModule></IfModule> párost szokás használni. (Ezt a védelmet a példáinkban általában nem használjuk, mert a beállítás enélkül is lehetséges, és egyszerűbb marad a beállítóállomány.)

<IfModule mod_modulnév.c>...</IfModule> A kifejezésbe – a „...” rész helyére – írt sorokat az Apache webkiszolgáló csak akkor veszi figyelembe, ha a kifejezésben megadott nevű modult betöltötték, vagy a webkiszolgáló bináris programállományába beleszerkesztették.

A következő példa a modulok beállítására szolgáló utasítások védelmét mutatja be a dir modult beállító utasítással.

102. példa. A DirectoryIndex kulcsszó csak akkor használható a beállítóállományban, ha a dir modult betöltöttük. A következő állományrészlet bemutatja, miképpen védhetjük meg a DirectoryIndex kifejezést a végrehajtástól, ha a modul betöltését eltávolítanánk az állományból:

```
1 LoadModule dir_module modules/mod_dir.so
2
3 <IfModule mod_dir.c>
4   # A könyvtárakban először ezeket keressük
5   DirectoryIndex index.html index.htm
6 </IfModule>
```

A állományrészletben bemutatott módszert használva a `dir` modul betöltését ideiglenesen eltávolíthatjuk, ha egy `#` jelet teszünk a sor elejére:

```
1 #LoadModule dir_module modules/mod_dir.so
2
3 <IfModule mod_dir.c>
4   # A könyvtárakban először ezeket keressük
5   DirectoryIndex index.html index.htm
6 </IfModule>
```

Így a `DirectoryIndex` nem okoz problémát, egyszerűen nem hajtódiik végre.

8.5.4. Könyvtárak szolgáltatása

Az Apache néhány modulja és beállítóeszköze lehetővé teszi, hogy a webböngésző a webkiszolgálón elhelyezett könyvtárak tartalmát jelenítse meg. Ez igen kényelmes szolgáltatás, hiszen így a felhasználónak a böngészőjében nem kell begépeleznie az állomány nevét, elegendő a számítógép és a könyvtár nevét megjegyeznie.

A következő lista a könyvtárak szolgáltatására használható modulokat és azok legfontosabb beállításait tartalmazza.

`dir` A modul betöltésével lehetővé válik a webcímben található állománynév elhagyása. Ha a modult betöltöttük, és a webkiszolgálótól kért cím nem tartalmaz állománynevet, a webkiszolgáló a könyvtárban található `index.html` állományt próbálja meg elküldeni a webböngészőnek.

A modul viselkedése a következő kifejezés segítségével befolyásolható:

`DirectoryIndex állomány1 állomány2` A kifejezés segítségével megadható, hogy könyvtárakra vonatkozó kérdések esetén a webböngésző mely állományokat próbálja meg szolgáltatni.

A kifejezésben egynél több állomány is megadható. Ekkor a kiszolgáló előbb az első állományt keresi, és ha azt nem találja, a következőkkel próbálkozik.

A kifejezésben szereplő állományneveknek nem kell a kért könyvtárban lenniük, ahogyan azt a következő minta alapján láthatjuk:

```
1 DirectoryIndex index.html index.htm /nincs.htm
```

A mintában szereplő kifejezés hatására a kiszolgáló előbb az index.html állományt keresi, majd – ha azt nem találja – az index.htm állománnyal próbálkozik. Ha egyik állomány sem található a kért könyvtárban, a kiszolgáló a DocumentRoot által meghatározott könyvtárban található nincs.htm állományt küldi el a böngészőnek.

autoindex A modul betöltése után a könyvtárak szolgáltatásakor az Apache létrehoz egy listát a könyvtárban található könyvtárbejegyzések legfontosabb adataiból, és azt küldi el a böngésző számára (lásd 8.1. ábra).

E modul működése összefügg a dir modul működésével. Ha a böngésző által kért címben nincs állománynév, előbb a dir modul kísérli meg megkeresni a könyvtárhoz tartozó állományt, az autoindex modul csak akkor hozza létre az állománylistát, ha ez nem sikerül.

Az autoindex modul használatát a beállítóállományban adott könyvtárakra nézve letilthatjuk, amennyiben e könyvtárak teljes tartalmát nem akarjuk nyilvánosságra hozni.

A autoindex modul viselkedését a következő kifejezések segítségével befolyásolhatjuk:

IndexOptions kapcsoló1 kapcsoló2 A kifejezés segítségével a modul által létrehozott állománylista néhány tulajdonságát változtathatjuk meg.

A kifejezésben tetszőlegesen sok kapcsolót sorolhatunk fel, melyek formája a következő lehet:

FancyIndexing A kapcsoló részletesebb, díszesebb lista készítésére adutasítást, ami tartalmaz például ikonokat is. mindenképpen érdekes bekapcsolnunk ezt a szolgáltatást, hiszen ellenkező esetben azt mondhatnák rólunk, hogy múlt századi webkiszolgálót üzemeltetünk.

Az autoindex beállítására használható további kifejezéket úgy mutatjuk be, hogy feltételezzük, a FancyIndexing be van kapcsolva.

NameWidth=szám A kifejezés segítségével megadhatjuk, hogy hány karakter szélességű helyen jelenjenek meg a könyvtárbejegyzések nevei. A webkiszolgáló a hosszabb neveket rövidíti, de természetesen ettől az állomány még elérhető marad a weblapon keresztül.

Igen fontos, hogy ezt az oszlopot a használt állománynevekhez igazítsuk, fontos, hogy a felhasználó az állomány nevének végét is lássa, hiszen ott található a változat száma és az állománynév-kiterjesztés, amelyek igen fontos információkat hordoznak az állományról.

Mivel még az operációs rendszereket gyártó vállalatok számára sem mindig világos, megismételjük ezt a fontos igazságot:
A felhasználó szeretné látni az állomány nevének végét is!

DescriptionWidth=szám A kifejezéssel az elkészült lista utolsó oszlopának szélességét adhatjuk meg.

Az utolsó oszlop az állomány típusának rövid leírását adja, ami nem olyan fontos, hiszen a *felhasználó amúgy is ismeri az állomány típusát a névkiterejtséből*.

FoldersFirst A kapcsoló hatására az elkészített listában az alkönyvtárak sorai a többi könyvtárbejegyzés sorai előtt fognak elhelyezkedni.

IconsAreLinks A kapcsoló hatására a könyvtárbejegyzések megnyithatók lesznek az ikonra való kattintással is.

VersionSort A kapcsoló hatására a kiszolgáló megkísérli az állománynevekben található változatszám szerint sorba rendezni az állományokat.

AddIcon ikon kiterjesztés1 kiterjesztés2 A kifejezés segítségével megadhatjuk, hogy az adott állománykiterjesztéshez milyen ikon tartozzon a listában. A kifejezésben szereplő ikon a webkiszolgálón található képállomány nevét adjza meg, amelyet a webböngésző a sor elején fog megjeleníteni.

Egy kifejezésben több kiterjesztés is szerepelhet, így ugyanazt az ikont több állománytípus jelzésére is használhatjuk.

A kifejezés a webkiszolgáló beállítóállományában többször is szerepelhet, így sorra megadhatjuk az egyes állománykiterjesztésekhez tartozó ikon nevét.

AddIconByType ikon típus1 típus2 A kifejezés segítségével megadhatjuk az egyes állománytípusokhoz tartozó ikonokat.

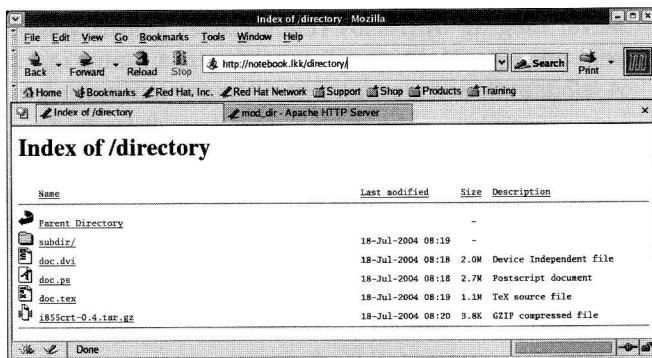
A kifejezésben található ikon a sorok előtt megjelenő képállomány neve, a kifejezésben szereplő típusok pedig az ikonhoz tartozó MIME típusok. A MIME típusokban használhatunk állománynév-helyettesítő karaktereket is (például `text/*`).

A kifejezés a beállítóállományban többször is szerepelhet, így sorra megadhatjuk az összes szolgáltatott állománytípushoz tartozó ikon nevét.

Ez a kifejezés csak akkor fejt ki hatását, ha a `mime` vagy a `mime_magic` modul is be van töltve.

DefaultIcon ikon A kifejezés segítségével meghatározhatjuk, milyen ikon kerüljön az ismeretlen állománytípusokhoz tartozó sorok elejére.

AddDescription "leírás" kiterjesztés1 kiterjesztés2 A kifejezés segítségével a létrehozott állománylista utolsó oszlopában látható rövid típus-



8.1. ábra. Az Apache által létrehozott állománylista

leírást adhatjuk meg az egyes állománykiterjesztésekhez. A kifejezés a AddIcon kulcsszóval felépített kifejezéshez hasonlóan használható.

Sajnos az Apache nem ismeri a AddDescriptionByType kulcsszót, így a lista utolsó oszlopában található rövid leírás nem minden tükrözi a valódi állománytípus, nem több egy egyszerű tippnél.

mime_magic A MIME típusok kezeléséhez a mime vagy a mime_magic modult kell töltenünk. A mime_magic modul nagy előnye, hogy az állományok típusát a tartalmuk alapján azonosítja, így a könyvtárbejegyzések listájában szereplő ikonok valódi információt közölhetnek az állományról.

A modul használatához a következő kifejezésnek szerepelnie kell a beállító-állományban:

MimeMagicFile **állománynév** A kifejezéssel meg kell adnunk, hogy az egyes állománytípusok tartalmának jellemző mintázatait melyik állományból olvassa a modul. Ha ezt nem adjuk meg a kifejezéssel, a modul betöltődik ugyan, de hatástalan marad.

Az Apache jelenlegi változatai tartalmaznak egy betölthető állományt a ServerRoot könyvtárban conf/magic néven.

A következő példa olyan teljes Apache beállítóállományt mutat be, amelyben a könyvtárlisták létrehozásához és az index.html állományok automatikus szolgáltatásához szükséges beállítások is szerepelnek.

103. példa. A következő példa bemutatja a dir, autoindex és a mime_magic modulok használatát.

```
1 #  
2 # Az alapbeállítások  
3 #
```

```
4 | ServerRoot "/etc/httpd"
5 | DocumentRoot "/var/www/html"
6 | PidFile "run/httpd.pid"
7 | Listen 80
8 | User "apache"
9 | Group "apache"
10 | ServerAdmin root@www.lkk
11 | DefaultType text/html
12 |
13 | # A modulok betöltése
14 |
15 | LoadModule dir_module modules/mod_dir.so
16 | LoadModule autoindex_module modules/mod_autoindex.so
17 | LoadModule mime_magic_module modules/mod_mime_magic.so
18 |
19 | # A mime_magic enélkül nem működik
20 | MimeMagicFile conf/magic
21 |
22 | # A könyvtárakban először ezeket keressük
23 | DirectoryIndex index.html index.htm
24 |
25 | # Az autoindex szolgáltatás beállításai
26 | IndexOptions FancyIndexing VersionSort \
27 |     NameWidth=60 DescriptionWidth=50\
28 |     FoldersFirst
29 |
30 | AddIconByType /icons/text.gif text/*
31 | AddIconByType /icons/image2.gif image/*
32 | AddIconByType /icons/sound2.gif audio/*
33 | AddIconByType /icons/movie.gif video/*
34 |
35 | AddIcon /icons/tar.gif .tar
36 | AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
37 | AddIcon /icons/a.gif .ps .ai .eps
38 | AddIcon /icons/layout.gif .html .shtml .htm .pdf
39 | AddIcon /icons/dvi.gif .dvi
40 | AddIcon /icons/tex.gif .tex
41 |
42 | AddIcon /icons/back.gif ..
43 | AddIcon /icons/hand.right.gif README
44 | AddIcon /icons/folder.gif ^^DIRECTORY^^
45 | AddIcon /icons/blank.gif ^^BLANKICON^^
46 |
47 | DefaultIcon /icons/unknown.gif
```

```

48
49 AddDescription "GZIP compressed file" .gz
50 AddDescription "GZIP compressed tar archive" .tgz

```

A beállítóállomány szándékosan csak kevés beállítást tartalmaz, a gyakorlatban általában több állománytípust sorolunk fel.

Figyeljük meg a 42–45. sorokban, hogy miképpen rendelhetünk értéket a különleges állománynevekhez!

Az állománylisták előállítása kapcsán meg kell jegyeznünk, hogy sok webmester ezt a szolgáltatást biztonsági szempontból helytelennek tartja, nem javasolja a bevezetését.

8.5.5. Biztonsági korlátozások

Az Apache beállítóállományában bizonyos beállítások a szolgáltatott könyvtárrakra külön is megadhatók. Különösen a biztonsági beállítások esetében hasznos ez a lehetőség, hiszen előfordulhat, hogy a webkiszolgáló által szolgáltatott könyvtárrakra egyedi biztonsági beállításokat akarunk megadni.

A biztonsági beállítások hatásának könyvtárrakra, állományokra, címekre való korlátozására szolgálnak a következő eszközök:

<Directory "könyvtárnév">...</Directory> A kifejezésben – a „...” helyén – szereplő biztonsági beállítások a kifejezésben szereplő könyvtárra és *annak alkönyvtáraina* korlátozódnak.

A kifejezésben található könyvtárnév annak a könyvtárnak neve, ahonnan az Apache az állományokat szolgáltatja, nem pedig a webböngésző címsorában látható könyvtárnév!

A kifejezésbe írt könyvtárnév tartalmazhat állománynév-helyettesítő karaktereket, ezek használata azonban általában szükségtelen.

Ha egy könyvtárra vagy könyvtárbejegyzésre a beállítóállomány több <Directory></Directory> szakasza is érvényes, a beállításokat a kiszolgáló a kifejezésekben található könyvtárnevek növekvő sorrendje szerint sorra veszi, és egymás után érvényesíti. A kiszolgáló így a könyvtárak beállításainak öröklődését biztosítja.

Például a /var/www/html beállításait örökli a /var/www/html/munka könyvtár, mert az előbbi neve rövidebb.

<DirectoryMatch "könyvtárnév">...</DirectoryMatch> A kifejezés használata megegyezik a <Directory></Directory> kifejezésével, a külöbség csak annyi, hogy itt a könyvtárnév megadásakor szabályos kifejezéseket is használhatunk.

.htaccess A könyvtárakra érvényes korlátozások megadhatók a könyvtárakban elhelyezett .htaccess nevű rejtett állományban is. Ez az eszköz tehát lehetővé teszi, hogy ne a beállítóállományban, hanem magában a szolgáltatott könyvtárban helyezzük el a biztonsági beállításokat.

<Files állománynév>...</Files> A kifejezésben – a „...” helyén – szereplő biztonsági beállítások azokra az állományokra (könyvtárakra nem!) érvényesek, amelyek neve a kifejezésben található állománynévre illeszkedik.

A kifejezésben található állománynév tartalmazhat állománynév-helyettesítő karaktereket.

A <Files></Files> kifejezés szerepelhet a <Directory></Directory> kifejezésen belül, ahol a hatása az adott könyvtárakra vonatkozik.

<FilesMatch "állománynév">...</FilesMatch> A kifejezés hatása megegyezik a <Files></Files> kifejezés hatásával, a külöbség csak annyi, hogy itt az állománynév megadásakor szabályos kifejezéseket is használhatunk.

<Location cím>...</Location> A kifejezésben – a „...” helyén – szereplő biztonsági beállítások azokra a könyvtárakra és alkönyvtáraikra érvényesek, amelyeket a webböngésző a megadott címmel ért el.

A kifejezésben szereplő cím nem tartalmazhatja a kiszolgáló nevét, a használt hálózati kaput stb., kizárolag a könyvtárak nevét.

A kifejezésben megadott cím tartalmazhat állománynév-helyettesítő karaktereket, ezek használva azonban a legtöbb esetben elkerülhető.

Ha egy könyvtárra vagy könyvtárbejegyzésre a beállítóállomány több <Location></Location> szakasza is érvényes, a beállításokat a kiszolgáló a kifejezésekben található címek növekvő sorrendje szerint sorra veszi, és egy más után érvényesíti.

A következő kifejezések hatókörét korlátozzák, ha az előzőleg bemutatott <Directory></Directory>, <Location></Location> stb. kifejezéseken belül használjuk őket.

core Ez a modulnév az állandóan elérhető szolgáltatásokhoz van rendelve. Ezt a modult soha nem kell betöltenünk, a hozzá tartozó szolgáltatások mégis mindenkor elérhetők.

A core modulhoz a következő biztonsági beállítások tartoznak:

Options kapcsoló1 kapcsoló2 A kifejezés segítségével beállíthatjuk, hogy a webkiszolgáló mely szolgáltatásai legyenek elérhetők, engedélyeztetek.

A kifejezésben szereplő kapcsolók alapértelmezés szerint egy-egy szolgáltatás engedélyezésére szolgálnak, de elhelyezhetünk a szolgáltatások neve előtt egy – karaktert, hogy a szolgáltatást letiltsuk.

A kifejezésbe írható kapcsolók közül a legfontosabbak a következők:

ExecCGI A könyvtárban található programok futtatása CGI programként. A CGI programok használatára a 283. oldalon visszatérünk.

FollowSymLinks A könyvtárban található közvetett hivatkozások követése. Ezt a szolgáltatást óvatosan kell használnunk, nehogy a böngésző egy elfelejtett közvetett hivatkozáson keresztül kijussan a dokumentumok tárolására szolgáló könyvtárból, és illetékenyül adatokhoz jusson.

A webkiszolgáló a FollowSymLinks kapcsolót nem veszi figyelembe a <Location></Location> kifejezésekben belül.

Indexes Az autoindex modul használatának engedélyezése, illetve tiltása az adott könyvtárában.

SymLinksIfOwnerMatch Ha ez a kapcsoló be van kapcsolva, a webkiszolgáló követi a közvetett hivatkozásokat, de csak akkor, ha a hivatkozásnak és az általa kijelölt könyvtárbejegyzésnek ugyanaz a tulajdonosa.

A SymLinksIfOwnerMatch kapcsolót a kiszolgáló nem veszi figyelembe a <Location></Location> kifejezésben belül.

AllowOverride jog1 jog2 A kifejezés meghatározza, hogy a könyvtárakban elhelyezett, biztonsági beállításokat tartalmazó .htaccess állományok milyen biztonsági beállításokat bírálhatnak felül.

A kifejezésben a következő kulcsszavakat sorolhatjuk fel:

None A .htaccess állomány egyetlen biztonsági beállítást sem bírálhat felül, az állományt a kiszolgáló nem veszi figyelembe.

All A .htaccess állomány bármilyen biztonsági beállítást felülbíráthat.

AuthConfig A .htaccess kifejezés felülbíráhatja a böngészőt vezérlő felhasználó azonosítására vonatkozó beállításokat. (A felhasználó azonosításáról a 286. oldalon bővebben olvashatunk.)

Indexes A .htaccess állomány felülbíráthatja az autoindex modul beállításait.

Limit A .htaccess állomány felülbíráhatja, hogy a webböngészők melyik számítógépről férhetnek hozzá a könyvtárban található adatokhoz.

Options A .htaccess állomány a könyvtárhoz tartozó egyéb biztonsági kapcsolókat – amelyeket az Options kapcsolóval hoztunk létre – felülbíráhatja.

access Ez a modul kimondottan a biztonsági beállítások kezelésére szolgál.

Az access modul beállítására a következő kifejezéseket használhatjuk:

Allow from cím A kifejezés segítségével hozzáférési jogot biztosíthatunk a webkiszolgálón található anyagokhoz.

A kifejezésben szereplő cím meghatározza, hogy mely számítógépek számára biztosítunk hozzáférést. A következő minták bemutatják, minden formában használhatjuk a címet:

- all Ha a kifejezésben ez a kulcsszó szerepel a cím helyén, minden számítógép számára biztosítjuk a hozzáférést.
- ak.hu Ha a kifejezésben egy számítógép neve szerepel, az adott számítógép számára biztosítjuk a hozzáférést. Ha a kifejezésben tartománynév szerepel, a tartomány minden számítógépe számára biztosítjuk a hozzáférést.
- 10.1.1.1 Ha a kifejezésben egy IP cím szerepel, az adott címmel rendelkező számítógép számára biztosítjuk a hozzáférést.
- 10 Ha a kifejezésben részleges IP cím szerepel, a címrészlet által jelzett hálózat minden tagja számára biztosítjuk a hozzáférést.
- 10.1.1.1/255.0.0.0 Ha a kifejezésben egy IP cím és egy alhálózati maszk szerepel, az adott alhálózat minden számítógépe számára biztosítjuk a hozzáférést.
- 10.1.1.1/8 Ha a kifejezésben egy IP cím és a hálózati cím hossza szerepel, az adott hálózat minden tagja számára biztosítjuk a hozzáférést.

Deny from cím A kifejezés segítségével megtilthatjuk a hozzáférést a webkiszolgálón található adatokhoz.

A kifejezésben szereplő cím formája és jelentése megegyezik a **Allow** kifejezésben található cím formájával és jelentésével, a különbség csak annyi, hogy itt a hozzáférést tiltjuk és nem engedélyezzük.

Order sorrend A kifejezés segítségével meghatározhatjuk, hogy az **Allow** és **Deny** kifejezésekben megfogalmazott engedélyeket és tiltásokat miképpen értelmezze a kiszolgáló.

A kifejezésben szereplő sorrend a következők egyike lehet:

Allow,Deny Ez az értelmezési mód a szigorúbb, alapértelmezés szerint minden ügyfél számára tiltott a hozzáférés.

Első lépésként a kiszolgáló az **Allow from cím** alakú kifejezéseket értékeli ki. Ha az ügyfélre nem illeszthető egyetlen **Allow** rész sem, a kiszolgáló megtagadja a hozzáférést.

Második lépésként a kiszolgáló a **Deny from cím** alakú kifejezéseket értékeli ki. Ha az ügyfélre illeszthető valamelyik **Deny** rész, a kiszolgáló megtagadja a hozzáférést.

Ennél az értelmezési módnál tehát csak azok az ügyfelek érhetik el az adatokat, amelyekre illeszthető valamelyik **Allow** kifejezés és nem illeszthető egyetlen **Deny** rész sem.

Az **Allow,Deny** kifejezésben nem használhatunk szóközt a vessző előtt, illetve után.

Deny ,Allow Ez az értelmezési mód az enyhébb, alapértelmezés szerint minden ügyfél számára engedélyezett a hozzáférés.

Első lépésként a kiszolgáló a Deny from cím alakú kifejezéseket értékeli ki. Ha az ügyfélre nem illeszthető egyetlen Deny from kifejezés sem, a kiszolgáló engedélyezi számára a hozzáférést.

Második lépésként a kiszolgáló az Allow from cím alakú kifejezéseket értékeli ki. Ha valamelyik Allow kifejezés illeszthető valamelyik ügyfélre, a kiszolgáló engedélyezi számára a hozzáférést. (Akkor is, ha valamelyik Deny illeszthető volt az ügyfélre.)

Ennél az értelmezési módnál tehát minden ügyfél elérheti az adatokat, amelyek számára ez nincs kimondottan megtiltva, de még ezek között is lehetünk kivételt.

A Deny,Allow kifejezésben nem használhatunk szóközt a vessző előtt, illetve után.

104. példa. A következő részlet bemutatja, hogyan használhatjuk a szigorúbb értelmezést a jogok megadásánál.

```

1 <Directory /adatok/titkos>
2   Order Allow,Deny
3   Allow from 1kk
4   Deny from kivetel.1kk
5 </Directory>
```

A részlet a kiszolgáló által szolgáltatott adatok adatok/titkos könyvtárára és annak külön nem szabályozott alkönyvtáraira vonatkozik. (Ez a legtöbb esetben a /var/www/html/adatok/titkos könyvtár.)

A bemutatott részlet minden ügyfélnek megtiltja a hozzáférést – ez az alapértelmezett értelmezés –, kivéve az 1kk tartomány gépeit. A tartomány gépei közül a kivetel.1kk kivételt képez, számára nem engedélyezzük a hozzáférést.

105. példa. A következő részlet bemutatja, miképpen használhatjuk az enyhébb értelmezést az adatok elérésének korlátozására.

```

1 <Location /pub>
2   Order Deny,Allow
3   Deny from 10.1.1.1/255.0.0.0
4   Allow from 10.1.1.100
5 </Location>
```

A részlet azokra az adatokra vonatkozik, amelyeket a kiszolgáló /pub kezdetű címen szolgáltat.

Ez a részlet minden számítógépnek biztosítja a hozzáférést – ez az alapértelmezett viselkedés –, kivéve a 10.0.0.0 A osztályú címtartomány tagjait, melyek közül kizárolag a 10.1.1.100 című számítógép férhet hozzá ezekhez az adatokhoz.

8.5.6. Programok futtatása a webkiszolgálón

A Web ma már nem csak egyszerűen weblapok gyűjteménye, nem csak egy nagy könyv, amelyet kedvünkre olvasgathatunk, hanem élő, a kérdéseinkre válaszoló, a felhasználókra reagáló rendszer. A legtöbb webkiszolgáló ma már képes programokat futtatva a felhasználók kéréseit kielégíteni.

A weben jelentkező felhasználók igényeit programok futtatásával kiszolgálni legegyszerűbben CGI (*common gateway interface*, általános átvárofelület) programokkal lehet. A következő oldalakon arról olvashatunk miképpen lehet az Apache webkiszolgálót CGI programok futtatására felkészíteni és hogyan tudunk CGI programokat készíteni.

A CGI programok futtatása a következő lépésekben foglalható össze:

1. A webböngésző elküldi a kérését a webkiszolgálónak. A kérés egy bizonyos állomány letöltésére vonatkozik.
2. Az Apache a kért állomány nevének kiterjesztésből vagy az állományt tartalmazó könyvtár nevéből arra következtet, hogy a kért állomány egy CGI program.
3. Az Apache a kérés körülményeit, a webböngésző adatait környezeti változókba helyezi, majd elindítja a kért állományt.
A CGI programokat tehát nem elküldenie, hanem futtatnia kell a kiszolgálónak. A CGI programok mindenkor a webkiszolgálón futnak.
4. A CGI program az általa létrehozott adatokat a szabványos kimenetére írja, amelyet az Apache beolvas, és elküld a webböngésző felé.

Bármilyen programozási nyelven készíthetünk tehát CGI programot, ami lehetőséget ad a programozónak a környezeti változók értékének lekérdezésére és a szabványos kimenetre való írásra. A legtöbb CGI programot kétségtől Perl programozási nyelven készítik[55], de az is írhat CGI programot, aki nem ismeri ezt a nyelvet.

Az Apache webkiszolgáló a legkönnyebben a cgi modul segítségével készíthető fel CGI programok futtatására:

`alias` A modul betöltésével lehetőségünk nyílik CGI programok futtatására.

Azt, hogy mely állományok minősülnek programnak, a következő kifejezésekkel állíthatjuk be:

`AddHandler cgi-script kiterjesztés` A kifejezés segítségével megadhatjuk, hogy melyek azok az állománynév-kiterjesztések, amelyek jelzik, hogy CGI programokról van szó, amelyeket a webkiszolgálónak futtatnia kell.

Ennek a kifejezésnek a használatához a mime modult is be kell töltönünk!

ScriptAlias cím könyvtár A kifejezés segítségével kijelölhetünk egy könyvtárat a CGI programok tárolására.

A kifejezésben szereplő könyvtár a CGI programok tárolására szolgáló könyvtár, a cím pedig az a cím, amelyen a kiszolgáló a könyvtárat elérhetővé teszi. A kifejezés hatására minden kérés, amely a kifejezésben szereplő címmel kezdődik, CGI program futtatását fogja eredményezni.

Ennek a kifejezésnek a használatához az alias modult be kell töltönünk.

A következő két példa bemutatja a CGI programok meghatározására használt két módszert. Az egyik példa minden .cgi kiterjesztésű állományt CGI programnak tekint, a másik példa pedig nyilvántartásba veszi a CGI programok tárolására használt könyvtárat.

106. példa. Állítsuk be úgy az Apache webkiszolgálót, hogy az összes .cgi kiterjesztésű névvel ellátott állományt CGI programnak tekintse és futtassa!

A következő sorok elvégzik a szükséges beállításokat:

```

1 LoadModule mime_module modules/mod_mime.so
2 LoadModule cgi_module modules/mod_cgi.so
3
4 #
5 # minden állomány CGI program, amelynek a .cgi a kiterjesztése.
6 #
7 AddHandler cgi-script .cgi

```

Amint látjuk, ehhez a beállításhoz a mime és a cgi modulok betöltésére van szükség.

107. példa. Állítsuk be úgy az Apache webkiszolgálót, hogy a /var/www/cgi-bin/ könyvtárat /cgi-bin/ néven szolgáltassa, a benne található könyvtárak és alkönyvtárak állományait pedig CGI programoknak tekintse! (Ezek a szokásos beállítások.)

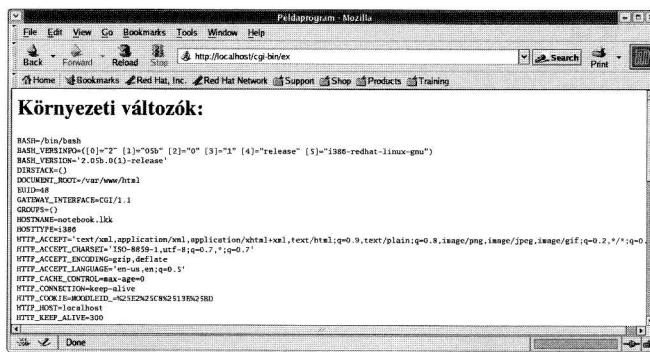
A beállításokat elvégezhetjük a következő sorok segítségével.

```

1 LoadModule alias_module modules/mod_alias.so
2 LoadModule cgi_module modules/mod_cgi.so
3
4 #
5 # minden CGI program, ami q /var/www/cgi-bin könyvtárban van.
6 #
7 ScriptAlias /cgi-bin /var/www/cgi-bin

```

Figyeljük meg, hogy a beállításokhoz szükség volt a alias és a cgi modulok betöltésére!



8.2. ábra. Egyszerű BASH CGI program eredménye

Webkiszolgálón futtatható programok készítése

Amikor a webkiszolgáló a webböngészőnek egy állományt küld, az állomány tartalma előre szabványos[48] fejlécet illeszt. Ha azonban az adatok CGI programról erednek, a kiszolgáló elvárja, hogy a fejléc egy részét a CGI program készítse el. Ha a programunk nem kezdi a szabványos kimenetére küldött sorokat a szabványos fejléccel, a webkiszolgáló nem küldi el az adatokat. (Az adatok helyett hibajelentést küld a webböngészőnek, és a hibát a hibanaplóban is elhelyezi.)

Szerencsére az egyszerűbb fejlécek elkészítése nagyon egyszerű. A következő két sor bemutatja a legegyszerűbb fejlécet, amit CGI program használhat:

```

1 Content-type: text/html
2 
```

A fejléc első sora a küldött adatok típusát adja meg a böngészőnek, a második – üres – sor pedig jelzi a fejléc végét. A CGI program által küldött fejléc végét mindenkorral jelezünk!

A következő példa egy igen egyszerű CGI programot mutat be, amely ezt az egyszerű fejléket használja. A program által létrehozott weblap a 8.2. ábrán látható.

108. példa. Készítsünk egyszerű BASH programot, amely CGI programként futtatva kiírja a webböngésző ablakába a környezeti változói nevét és értékét! A program segítségével könnyedén nyomon követhetjük az Apache által a CGI program számára átadott környezeti változók értékét.

```

1 #!/bin/bash
2 #
3 # Példa cgi program készítésére
4 #
5 echo -e "Content-type: text/html\r\n"
```

```

6 | echo -e "\r\n"
7 |
8 | echo "<html><head>"
9 | echo "  <title>Példaprogram</title>"
10 | echo "<body bgcolor=white>"
11 | echo "  <h1>Környezeti változók:</h1>"
12 | echo "  <pre>"
13 |
14 | set
15 |
16 | echo "  </pre>"
17 | echo "</body></html>"
```

A program 5–6. sorában található echo utasítások kiírják a szabványos kimenetre a HTTP fejléc egy részét². Ezek nélkül a sorok nélkül a programunk nem volna CGI programként használható.

A program 8–12. sorai a webböngésző számára készített oldal első sorait, a 14. sorban található set utasítás a az összes környezeti változó nevét és értékét, a 16–17. sorok pedig a weblap utolsó sorait írják ki a szabványos kimenetre.

8.5.7. Felhasználók azonosítása

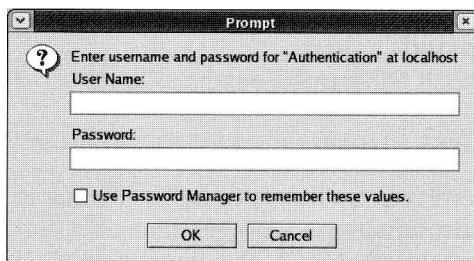
Az Apache webkiszolgáló rendelkezik egy egyszerű felhasználóazonosítási rendszerrel, amely lehetővé teszi, hogy a kisszolgálón elhelyezett állományok eléréset jelszavas azonosítástól tegyük függővé. Az elterjedten használt webböngészők mindenike képes együttműködni a felhasználóazonosítás során.

Amikor a webkiszolgáló jelszavas azonosítást kér, a webböngészők egy ablakot jelenítenek meg (8.3. ábra), amelyben felhasználói nevet és jelszót kérnek a felhasználótól. A böngészőprogramok a nevet és jelszót nem kérik újra, azt addig használják, amíg a böngészőprogramból ki nem lépünk.

A webkiszolgáló és webböngésző által megvalósított felhasználóazonosítás legnagyobb előnye, hogy igen egyszerű a használata. Létre kell hoznunk egy felhasználónyilvántartást, és be kell állítanunk az azonosítást a webböngésző beállítóállományában, és a jelszavas védelem már működik is. Ez a jelszavas védelem azonban nem a lehető legerősebb, ezért fontos adatok védelmére nem használható! Ha életbevágóan fontos adatokat szeretnénk webes felületen keresztül ellenőrzött módon szolgáltatni, más eszközt kell keresnünk!

A jelszavas védelemhez elsősorban egy felhasználónyilvántartásra van szükségünk. Az Apache webkiszolgáló alapértelmezés szerint szöveges állományokat használ a felhasználók, felhasználói csoportok és jelszavak nyilvántartására. A webkiszolgáló beállítóállományában több ilyen állományra is hivatkozhatunk,

²Valójában csak a továbbított adatok típusát, és a fejléc végét jelző üres sort kell a programnak előállítania (a lektor).



8.3. ábra. A webböngésző jelszókérése

a szolgáltatott adatok egy-egy részéhez más-más felhasználói nyilvántartást rendelve.

A felhasználói neveket és jelszavakat tartalmazó állományt legegyszerűbben a htpasswd programmal hozhatjuk létre, ahogyan azt a következő példa bemutatja³.

109. példa. Hozzuk létre a /etc/htpasswd állományt az Apache webkiszolgáló számára, és helyezzük el benne egy felhasználó felhasználói nevét és jelszavát!

```
$ htpasswd -c /etc/htpasswd pipas
New password:
Re-type new password:
Adding password for user pipas
$
```

A htpasswd a -c kapcsoló hatására létrehozza a felhasználók és a jelszavak tárolására használt állományt. Amikor az állomány már létezik – például a következő felhasználó hozzáadásakor –, már nem kell használnunk ezt a kapcsolót.

Az Apache webkiszolgálót felkészítették egy fejlettebb felhasználói azonosítórendszer használatára, amely a jelszavak hálózaton való átküldése során titkosítást használ. Ennek a rendszernek a használatához a jelszavakat tartalmazó állományt a htdigest programmal kell létrehoznunk. A htdigest program használata majdnem teljesen megegyezik a htpasswd program használatával, a különbség csak annyi, hogy paraméterként egy témakört is meg kell adnunk.

110. példa. Hozzunk létre felhasználói adatok tárolására használható szöveges állományt a htdigest programmal!

```
$ htdigest -c /etc/htdigest Authentication pipas
Adding password for pipas in realm Authentication.
New password:
Re-type new password:
$
```

³A program segítségével kódolt jelszót hozhatunk ugyan létre, de a bejelentkezés során ettől még kódolatlan formában történik a jelszó átvitele (a lektor).

8.3. tábla: htpasswd

Az Apache által használt felhasználói adatbázis kezelése.

htpasswd [kapcsolók] állománynév felhasználói_név

A program segítségével létrehozhatjuk és módosíthatjuk az Apache által kezelt felhasználói adatbázist tartalmazó szöveges állományt, a felhasználók jelszavát módosíthatjuk, illetve új felhasználókat adhatunk az adatbázishoz. Az Apache felhasználói adatbázisa tetszőleges állományban (vagy állományokban) lehet, a webkiszolgáló beállítóállománya alapján.

Kapcsoló Jelentés

- c Az állomány létrehozása.
- m Az MD5 jelszókódolás használata.
- D A felhasználó törlése a jelszóadatbázisból.

Amint láthatjuk, a felhasználó neve mellett téma körként az Authentication szót adtuk meg. A program egy felhasználó számára több jelszót is képes ugyanabban az állományban tárolni más-más téma kör felhasználásával.

Amint látható, a htdigest programot is a -c kapcsolóval utasíthatjuk az állomány létrehozására.

Az Apache a szöveges felhasználó-nyilvántartás alapján képes a felhasználókat csoportokba rendezve kezelní. Ehhez egy (vagy több) csoportnyilvántartást kell létrehoznunk. A csoportnyilvántartást biztosító szöveges állományban minden csoport külön sorban helyezkedik el, ahogyan a következő példa is bemutatja.

111. példa. Rendezzük csoportokba az Apache webkiszolgáló által nyilvántartott felhasználókat! Helyezzük el a /etc/htgroup állományban a következő sorokat:

```
1 user: fred joe bender
2 managers: admin
```

Az állomány alapján két felhasználói csoport létezik user és managers néven, az elsőben három, a másodikban pedig egy felhasználóval.

A felhasználók azonosításában a következő modulok kapnak szerepet:

core A core modul – amely a külön betöltés nélkül is rendelkezésünkre álló eszközöket tartalmazza – a következő eszközöket biztosítja a felhasználóazonosításhoz:

AuthType típus A kifejezés segítségével megadhatjuk, milyen jellegű felhasználói azonosítást szeretnénk használni. A kifejezésben szereplő típus a következők egyike lehet:

Basic Ezzel a kulcsszóval az egyszerű felhasználóazonosítás használatára adhatunk utasítást. Az egyszerű felhasználóazonosítást a mod_auth modul segítségével biztosíthatjuk. Ez a felhasználóazonosítás a felhasználói nevek és jelszavak tárolására a htpasswd program által létrehozott szöveges állományt használja.

A Basic kulcsszóval előírt egyszerű felhasználóazonosítás legnagyobb hátránya, hogy a felhasználói név és a jelszó kódolatlanul, lehallgatható módon halad a hálózaton.

Digest A kulcsszóval előírhatjuk, hogy a webkiszolgáló a felhasználók azonosítására az összetettebb, titkosítással védett rendszert használja.

Ezt az azonosítási módot többek között a mod_auth_digest modul támogatásával használhatjuk.

A Digest azonosítási rendszer a htdigest program által előállított szöveges állomány alapján működik.

AuthName "név" A kifejezés segítségével megadhatjuk a webböngészőnek, hogy milyen szöveget írjon ki a felhasználói név és a jelszó bekérésekor. A kifejezésben megadott név alapján a felhasználó megtudhatja, milyen okból kér jelszót a webböngésző.

Ha a Digest azonosítási módot használjuk, az AuthName kulcsszó után megadott névnek megfelelő témaállományt kell a jelszónak a nyilvántartásban szereplnie.

Require követelmény A kifejezés segítségével megadhatjuk, hogy mi az állományokhoz és alkönyvtárakhoz való hozzáférés feltétele. A kifejezésben szereplő követelmény formája a következők egyike lehet:

user felhasználó1 felhasználó2 Az állományokhoz és az alkönyvtárakhoz csak a felsorolt nevű felhasználók férhetnek hozzá.

group csoport1 csoport2 Az állományokhoz és az alkönyvtárakhoz csak azok a felhasználók férhetnek hozzá, akik a megadott csoportok egyikébe tartoznak.

valid-user Az állományokhoz és alkönyvtárakhoz csak azok a felhasználók férhetnek hozzá, akik a nyilvántartásban szereplő jelszavakkal azonosították magukat.

file-owner Az állományokhoz és alkönyvtárakhoz csak a tulajdonosaiak férhetnek hozzá.

file-group Az állományokhoz és alkönyvtárakhoz csak azok a felhasználók férhetnek hozzá, akik tagjai a tulajdonoscsoportnak.

mod_auth A mod_auth modul a felhasználók azonosításához a következő fontos eszközöket biztosítja:

AuthUserFile állománynév A kifejezés segítségével megadhatjuk annak az állománynak a nevét, amelyben a felhasználók felhasználói nevét és jelszavát tároljuk.

AuthGroupFile állománynév A kifejezés segítségével megadhatjuk annak az állománynak a nevét, amelyben a felhasználói csoportok adatait tároljuk.

mod_auth_digest Ennek a modulnak a szolgáltatásaként a felhasználóazonosítást biztonságosabbá tehetjük. A modul a vonatkozó szabvány alapján gondoskodik arról, hogy a webböngésző és a webkiszolgáló a jelszavakat kódolt formában küldje át a hálózaton.

A **mod_auth_digest** modul számára a felhasználói neveket és jelszavakat tartalmazó állományt a **htdigest** programmal kezelhetjük.

A modul beállítására a következő kifejezéseket használhatjuk:

AuthDigestFile állománynév A kifejezés segítségével megadhatjuk, hogy melyik állományban találhatók a felhasználók felhasználói nevei és jelszavai.

AuthDigestGroupFile állománynév A kifejezés segítségével megadhatjuk, hogy melyik állományban tároljuk a felhasználói csoportok adatait.

A következő két példa bemutatja, hogyan használhatjuk e két felhasználóazonosítási rendszert.

112. példa. Módosítsuk a webkiszolgáló beállítóállományát oly módon, hogy az összes szolgáltatott állomány és alkönyvtár eléréséhez jelszót kérjen a felhasználóktól! A jelszavak tárolására használjuk a 109. példában létrehozott felhasználói nyilvántartást!

```

1 LoadModule auth_module modules/mod_auth.so
2 <Directory />
3   AuthUserFile /etc/htpasswd
4   AuthName "Authentication"
5   AuthType Basic
6   Require valid-user
7 </Directory>
```

Amint látható a beállítóállomány részletéből, a felhasználói azonosításra az egy-szerű felhasználóazonosítást használjuk, ami a felhasználói nevek és jelszavak hálózaton való átvitelére nem használ titkosítást.

113. példa. Módosítsuk a webkiszolgáló beállítóállományát oly módon, hogy az a 110. feladatban létrehozott felhasználói nyilvántartást használja! Helyezzük el a következő sorokat a beállítóállományban:

```

1 <Directory />
2   AuthDigestFile /etc/htdigest
```

```
3   AuthName "Authentication"
4   AuthType Digest
5   Require valid-user
6 </Directory>
```

Figyeljük meg, hogy az AuthName után megadott szöveg és a felhasználó jelszavának létrehozásakor megadott témakör megegyezik!

Irodalomjegyzék

- [1] Paul Albitz és Cricket Liu. *DNS and BIND*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 1992 október.
- [2] Saqib Ali. Apache-alapú webdav szerver, ldap és ssl szolgáltatásokkal. World Wide Web dokumentum. A [3] magyar fordítása.
- [3] Saqib Ali. Apache based webdav server with ldap and ssl. World Wide Web dokumentum.
- [4] H. Alvestrand, S. Kille, R. Miles, M. Rose és S. Thompson. RFC 1495: Mapping between X.400 and RFC-822 message bodies, 1993 augusztus. Az RFC2156 [68] elavulttá tette. Az RFC987, RFC1026, RFC1138, RFC1148, RFC1327 [69, 70, 71, 72, 57] ajánlásokat felülíró ajánlás. Állapot: javasolt szabvány.
- [5] M. Andrews. RFC 2308: Negative caching of DNS queries (DNS NCACHE), 1998 március. Az RFC1034, RFC1035 [94, 95] frissítése. Állapot: javasolt szabvány.
- [6] D. Barr. RFC 1912: Common DNS operational and configuration errors, 1996 február. Az RFC1537 [7] ajánlást felülíró ajánlás. Állapot: információs.
- [7] P. Beertema. RFC 1537: Common DNS data file configuration errors, 1993 október. Az RFC1912 [6] elavulttá tette. Állapot: információs.
- [8] T. Berners-Lee, R. Fielding és H. Frystyk. RFC 1945: Hypertext Transfer Protocol — HTTP/1.0, 1996 május. Állapot: információs.
- [9] A. Bhushan, B. Braden, W. Crowther, E. Harslem, J. Heafner, A. McKenzie, J. Melvin, B. Sundberg, D. Watson és J. White. RFC 172: The file transfer protocol, 1971 június. Az RFC0265 [12] elavulttá tette. Az RFC0114 [13] frissítése. Az RFC0238 [21] tovább frissítette. Állapot: ismeretlen.

- [10] A. Bhushan, B. Braden, W. Crowther, E. Harslem, J. Heafner, A. McKenzie, B. Sundberg, D. Watson és J. White. RFC 264: The data transfer protocol, 1972 január. Az RFC0354 [15] elavulttá tette. Az RFC0171 [11] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
- [11] A. Bhushan, R. Braden, W. Crowther, E. Harslem, J. Heafner, A. McKenzie, J. Melvin, B. Sundberg, D. Watson és J. White. RFC 171: The Data Transfer Protocol, 1971 június. Az RFC0264 [10] elavulttá tette. Az RFC0114 [13] frissítése. Az RFC0238 [21] tovább frissítette. Nem elérhető az Interneten. Állapot: ismeretlen.
- [12] A. Bhushan, R. Braden, W. Crowther, E. Harslem, J. F. Heafner, A. M. McKenzie, J. T. Melvin, R. L. Sundberg, R. W. Watson és J. E. White. RFC 265: The File Transfer Protocol, 1971 december. Az RFC0354 [15] elavulttá tette. Az RFC0172 [9] ajánlást felülíró ajánlás. Az RFC0294 [14] tovább frissítette. Állapot: ismeretlen.
- [13] A. K. Bhushan. RFC 114: File transfer protocol, 1971 április. Az RFC0141, RFC0172, RFC0171 [58, 9, 11] tovább frissítette. Állapot: ismeretlen. Nem elérhető az Interneten.
- [14] A. K. Bhushan. RFC 294: The use of „set data type” transaction in file transfer protocol, 1972 január. Az RFC0265 [12] frissítése. Állapot: ismeretlen.
- [15] A. K. Bhushan. RFC 354: File transfer protocol, 1972 július. Az RFC0542 [103] elavulttá tette. Az RFC0264, RFC0265 [10, 12] ajánlásokat felülíró ajánlás. Az RFC0385 [16] tovább frissítette. Állapot: ismeretlen. Formátum: TXT=58074 bájt.
- [16] A. K. Bhushan. RFC 385: Comments on the file transfer protocol, 1972 augusztus. Az RFC0354 [15] frissítése. Az RFC0414 [17] tovább frissítette. Állapot: ismeretlen.
- [17] A. K. Bhushan. RFC 414: File transfer protocol (FTP) status and further comments, 1972 december. Az RFC0385 [16] frissítése. Állapot: ismeretlen. Nem elérhető az Interneten.
- [18] N. Borenstein és N. Freed. RFC 1341: MIME (Multipurpose Internet Mail Extensions): Mechanisms for specifying and describing the format of Internet message bodies, 1992 június. Az RFC1521 [19] elavulttá tette. Állapot: javasolt szabvány.
- [19] N. Borenstein és N. Freed. RFC 1521: MIME (Multipurpose Internet Mail Extensions) part one: Mechanisms for specifying and describing the format of Internet message bodies, 1993 szeptember. Az RFC2045, RFC2046, RFC2047, RFC2048, RFC2049 [49, 50, 102, 52, 51] elavulttá tette. Az RFC1341 [18] ajánlást felülíró ajánlás. Az RFC1590 [134] tovább frissítette. Állapot: szabványvázlat.

- [20] R. Braden. STD 3: Requirements for internet hosts — communication layers, 1989 október. Lásd még RFC1122, RFC1123 [22, 23].
- [21] R. T. Braden. RFC 238: Comments on DTP and FTP proposals, 1971 szep-tember. Az RFC0171, RFC0172 [11, 9] frissítése. Az RFC0269 [24] tovább frissítette. Állapot: ismeretlen.
- [22] R. T. Braden. RFC 1122: Requirements for Internet hosts — communica-tion layers, 1989 október. Lásd még STD0003 [20]. Állapot: szabvány.
- [23] R. T. Braden. RFC 1123: Requirements for Internet hosts — application and support, 1989 október. Lásd még STD0003 [20]. Az RFC0822 [27] frissítése. Az RFC2181 [45] tovább frissítette. Állapot: szabvány.
- [24] H. Brodie. RFC 269: Some experience with file transfer, 1971 december. Az RFC0122, RFC0238 [171, 21] frissítése. Állapot: ismeretlen. Formátum: TXT=5961 bájt.
- [25] V. G. Cerf és J. Postel. RFC 322: Well known socket numbers, 1972 március. Állapot: ismeretlen.
- [26] Bryan Costales és Eric Allman. *sendmail*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 3. kiadás, 2002.
- [27] D. Crocker. RFC 822: Standard for the format of ARPA Internet text mes-sages, 1982 augusztus. Lásd még STD0011 [30]. Az RFC0733 [29] aján-lást felülíró ajánlás. Az RFC1123, RFC1138, RFC1148, RFC1327, RFC2156 [23, 71, 72, 57, 68] tovább frissítette. Állapot: szabvány.
- [28] D. Crocker, K. T. Pogran, J. Vittal és D. A. Henderson. RFC 724: Proposed official standard for the format of ARPA network messages, 1977 május. Az RFC0733 [29] elavulttá tette. Állapot: ismeretlen. Nem elérhető az In-terneten.
- [29] D. Crocker, J. Vittal, K. T. Pogran és D. A. Henderson. RFC 733: Stan-dard for the format of ARPA network text messages, 1977 november. Az RFC0822 [27] elavulttá tette. Az RFC0724 [28] ajánlást felülíró ajánlás. Állapot: ismeretlen.
- [30] David H. Crocker. STD 11: Standard for the format of ARPA Internet text messages, 1982 augusztus. Lásd még RFC0822 [27]. Az RFC0733 [29] ajánlást felülíró ajánlás.
- [31] David A. (Allan) Curry. *UNIX Systems Programming for SVR4*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 1996 július.

- [32] C. Davis, P. Vixie, T. Goodwin és I. Dickinson. RFC 1876: A means for expressing location information in the domain name system, 1996 január. Az RFC1034, RFC1035 [94, 95] frissítése. Állapot: kísérleti.
- [33] S. Deering és R. Hinden. RFC 1883: Internet Protocol, version 6 (IPv6) specification, 1995 december. Az RFC2460 [34] elavulttá tette. Állapot: javasolt szabvány.
- [34] S. Deering és R. Hinden. RFC 2460: Internet Protocol, Version 6 (IPv6) specification, 1998 december. Az RFC1883 [33] ajánlást felülíró ajánlás. Állapot: szabványvázlat.
- [35] S. E. Deering. RFC 988: Host extensions for IP multicasting, 1986 július. Az RFC1054, RFC1112 [36, 37] elavulttá tette. Az RFC0966 [38] ajánlást felülíró ajánlás. Állapot: ismeretlen.
- [36] S. E. Deering. RFC 1054: Host extensions for IP multicasting, 1988 május. Az RFC1112 [37] elavulttá tette. Az RFC0988 [35] ajánlást felülíró ajánlás. Állapot: ismeretlen.
- [37] S. E. Deering. RFC 1112: Host extensions for IP multicasting, 1989 augusztus. Az RFC0988, RFC1054 [35, 36] ajánlásokat felülíró ajánlás. Lásd még STD0005 [130]. Az RFC2236 [47] tovább frissítette. Állapot: szabvány.
- [38] S. E. Deering és D. R. Cheriton. RFC 966: Host groups: A multicast extension to the Internet Protocol, 1985 december. Az RFC0988 [35] elavulttá tette. Állapot: ismeretlen.
- [39] R. Droms. RFC 1531: Dynamic host configuration protocol, 1993 október. Az RFC1541 [40], RFC2131 [41] elavulttá tette. Állapot: javasolt szabvány.
- [40] R. Droms. RFC 1541: Dynamic host configuration protocol, 1993 október. Az RFC2131 [41] elavulttá tette. Az RFC1531 [39] ajánlást felülíró ajánlás. Állapot: javasolt szabvány.
- [41] R. Droms. RFC 2131: Dynamic host configuration protocol, 1997 március. Az RFC1541 [40] ajánlást felülíró ajánlás. Állapot: szabványvázlat.
- [42] D. Eastlake. RFC 2137: Secure domain name system dynamic update, 1997 április. Az RFC1035 [95] frissítése. Állapot: javasolt szabvány.
- [43] D. Eastlake, 3rd és C. Kaufman. RFC 2065: Domain name system security extensions, 1997 január. Az RFC1034, RFC1035 [94, 95] frissítése. Állapot: javasolt szabvány.
- [44] R. Elz és R. Bush. RFC 1982: Serial number arithmetic, 1996 augusztus. Az RFC1034, RFC1035 [94, 95] frissítése. Állapot: javasolt szabvány.

- [45] R. Elz és R. Bush. RFC 2181: Clarifications to the DNS specification, 1997 július. Az RFC1034, RFC1035, RFC1123 [94, 95, 23] frissítése. Állapot: javasolt szabvány.
- [46] C. F. Everhart, L. A. Mamakos, R. Ullmann és P. V. Mockapetris. RFC 1183: New DNS RR definitions, 1990 október. Az RFC1034, RFC1035 [94, 95] frissítése. Állapot: kísérleti.
- [47] W. Fenner. RFC 2236: Internet Group Management Protocol, version 2, 1997 november. Az RFC1112 [37] frissítése. Állapot: javasolt szabvány.
- [48] R. Fielding, J. Gettys, J. Mogul, H. Frystyk és T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol — HTTP/1.1, 1997 január. Állapot: javasolt szabvány.
- [49] N. Freed és N. Borenstein. RFC 2045: Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet message bodies, 1996 november. Az RFC1521, RFC1522, RFC1590 [19, 101, 134] ajánlásokat felülíró ajánlás. Az RFC2184, RFC2231 [53, 54] tovább frissítette. Állapot: szabványvázlat.
- [50] N. Freed és N. Borenstein. RFC 2046: Multipurpose Internet Mail Extensions (MIME) part two: Media types, 1996 november. Az RFC1521, RFC1522, RFC1590 [19, 101, 134] ajánlásokat felülíró ajánlás. Állapot: szabványvázlat.
- [51] N. Freed és N. Borenstein. RFC 2049: Multipurpose Internet Mail Extension (MIME) part five: Conformance criteria and examples, 1996 november. Az RFC1521, RFC1522, RFC1590 [19, 101, 134] ajánlásokat felülíró ajánlás. Állapot: szabványvázlat.
- [52] N. Freed, J. Klensin és J. Postel. RFC 2048: Multipurpose Internet Mail Extensions (MIME) part four: Registration procedures, 1996 november. Az RFC1521, RFC1522, RFC1590 [19, 101, 134] ajánlásokat felülíró ajánlás. Állapot: jelenleg javasolt gyakorlat.
- [53] N. Freed és K. Moore. RFC 2184: MIME parameter value and encoded word extensions: Character sets, languages, and continuations, 1997 augusztus. Az RFC2184, RFC2231 [53, 54] elavulttá tette. Az RFC2045, RFC2047, RFC2183 [49, 102, 163] frissítése. Állapot: javasolt szabvány.
- [54] N. Freed és K. Moore. RFC 2231: MIME parameter value and encoded word extensions: Character sets, languages, and continuations, 1997 november. Az RFC2184 [53] ajánlást felülíró ajánlás. Az RFC2045, RFC2047, RFC2183 [49, 102, 163] frissítése. Állapot: javasolt szabvány.

- [55] Scott Guelich, Shishir Gundavaram és Gunther Birznieks. *CGI Programming with Perl*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2. kiadás, 2000 július.
- [56] Silvia Hagen. *IPv6 Essentials*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2002.
- [57] S. Hardcastle-Kille. RFC 1327: Mapping between X.400(1988) /ISO 10021 and RFC 822, 1992 május. Az RFC1495, RFC2156 [4, 68] elavulttá tette. Az RFC987, RFC1026, RFC1138, RFC1148 [69, 70, 71, 72] ajánlásokat felülíró ajánlás. Az RFC0822, RFC0822 [27, 27] frissítése. Állapot: javasolt szabvány.
- [58] E. Harslem és J. F. Heafner. RFC 141: Comments on RFC 114: A file transfer protocol, 1971 április. Az RFC0114 [13] frissítése. Állapot: ismeretlen.
- [59] E. Harslem és R. Stoughton. RFC 225: Rand/UCSB network graphics experiment, 1971 szeptember. Az RFC0074 [169] frissítése. Állapot: ismeretlen. Nem elérhető az Interneten.
- [60] Gilbert Held. *Ethernet Networks: Design, Implementation, Operation & Management*. John Wiley & Sons, Ltd., 4. kiadás, 2003.
- [61] M. Horowitz és S. Lunt. RFC 2228: FTP security extensions, 1997 október. Az RFC0959 [135] frissítése. Állapot: javasolt szabvány.
- [62] Craig Hunt. *Networking Personal Computers with TCP/IP: Building TCP/IP Networks*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 1995 július.
- [63] Craig Hunt. *TCP/IP Network Administration*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 3. kiadás, 2002.
- [64] Information Sciences Institute. University of Southern California. RFC 212: NWG meeting on network usage, 1971 augusztus. Az RFC0207 [164] ajánlást felülíró ajánlás. Az RFC0222 [89] tovább frissítette. Állapot: ismeretlen.
- [65] M. St. Johns. RFC 912: Authentication service, 1984 szeptember. Az RFC0931 [66] elavulttá tette. Állapot: ismeretlen.
- [66] M. St. Johns. RFC 931: Authentication server, 1985 január. Obsoleted RFC1413 [67]. Az RFC0912 [65] ajánlást felülíró ajánlás. Állapot: ismeretlen.

- [67] M. St. Johns. RFC 1413: Identification protocol, 1993 január. Az RFC0931 [66] ajánlást felülíró ajánlás. Állapot: javasolt szabvány.
- [68] S. Kille. RFC 2156: MIXER (Mime Internet X.400 Enhanced Relay): Mapping between X.400 and RFC 822/MIME, 1998 január. Az RFC0987, RFC1026, RFC1138, RFC1148, RFC1327, RFC1495 [69, 70, 71, 72, 57, 4] ajánlásokat felülíró ajánlás. Az RFC0822 [27] frissítése. Állapot: javasolt szabvány.
- [69] S. E. Kille. RFC 987: Mapping between X.400 and RFC 822, 1986 június. Az RFC2156 [68] elavulttá tette. Az RFC1026, RFC1138, RFC1148 [70, 71, 72] tovább frissítette. Állapot: ismeretlen.
- [70] S. E. Kille. RFC 1026: Addendum to RFC 987: (mapping between X.400 and RFC-822), 1987 szeptember. Az RFC1327, RFC1495, RFC2156 [57, 4, 68] elavulttá tette. Az RFC0987 [69] frissítése. Az RFC1138, RFC1148 [71, 72] tovább frissítette. Állapot: ismeretlen.
- [71] S. E. Kille. RFC 1138: Mapping between X.400(1988) /ISO 10021 and RFC 822, 1989 december. Az RFC1327, RFC1495, RFC2156 [57, 4, 68] elavulttá tette. Az RFC0822, RFC0987, RFC1026 [27, 69, 70] frissítése. Az RFC1148 [72] tovább frissítette. Állapot: kísérleti.
- [72] S. E. Kille. RFC 1148: Mapping between X.400(1988) /ISO 10021 and RFC 822, 1990 március. Az RFC1327, RFC1495, RFC2156 [57, 4, 68] elavulttá tette. Az RFC0822, RFC0987, RFC1026, RFC1138 [27, 69, 70, 71] frissítése. Állapot: kísérleti.
- [73] S. Kirkpatrick, M. K. Stahl és M. Recker. RFC 1166: Internet numbers, 1990 július. Az RFC1117, RFC1062, RFC1020 [153, 152, 151] ajánlásokat felülíró ajánlás. Állapot: információs.
- [74] J. Klensin, N. Freed, M. Rose, E. Stefferud és D. Crocker. RFC 1651: SMTP service extensions, 1994 július. Az RFC1869, STD0010 [75, 136] elavulttá tette. Az RFC1425 [76] ajánlást felülíró ajánlás. Állapot: szabványvázlat.
- [75] J. Klensin, N. Freed, M. Rose, E. Stefferud és D. Crocker. RFC 1869: SMTP service extensions, 1995 november. Lásd még STD0010 [136]. Az RFC1651 [74] ajánlást felülíró ajánlás. Állapot: szabvány.
- [76] J. Klensin, WG Chair, N. Freed, M. Rose, E. Stefferud és D. Crocker. RFC 1425: SMTP service extensions, 1993 február. Az RFC1651 [74] elavulttá tette. Állapot: javasolt szabvány.
- [77] M. Krilanovich. RFC 399: SMFS login and logout, 1972 szeptember. Az RFC0431 [78] elavulttá tette. Az RFC0122 [171] frissítése. Állapot: ismeretlen.

- [78] M. Krilanovich. RFC 431: Update on SMFS login and logout, 1972 december. Az RFC0399 [77] ajánlást felülíró ajánlás. Az RFC0122 [171] frissítése. Állapot: ismeretlen.
- [79] Nicolai Langfeld, Jamie Norish, and et. al. Dns hogyan. World Wide Web dokumentum. Hungarian translation by Füri Zoltán of [80].
- [80] Nicolai Langfeld, Jamie Norish, and et. al. Dns howto. World Wide Web dokumentum. Also available in Hungarian [79] translation.
- [81] Pere László. *Unix-GNU/Linux: Programozás C nyelven*. Kiskapu, Budapest, 2003.
- [82] Pere László. *GNU/Linux rendszerek üzemeltetése I.* Kiskapu, Budapest, 2005.
- [83] E. Lear, E. Fair, D. Crocker és T. Kessler. RFC 1627: Network 10 considered harmful (some practices shouldn't be codified), 1994 június. Az BCP0005, RFC1918 [?, 138] elavulttá tette. Állapot: információs.
- [84] B. M. Leiner. RFC 1077: Critical issues in high bandwidth networking, 1988 november. Állapot: ismeretlen.
- [85] Ravi Malhotra. *IP Routing*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2002.
- [86] B. Manning. RFC 1348: DNS NSAP RRs, 1992 július. Az RFC1637 [87] elavulttá tette. Az RFC1034, RFC1035 [94, 95] frissítése. Az RFC1637 [87] tovább frissítette. Állapot: kísérleti.
- [87] B. Manning és R. Colella. RFC 1637: DNS NSAP resource records, 1994 július. Az RFC1706 [88] elavulttá tette. Az RFC1348 [86] ajánlást felülíró ajánlás. Az RFC1348 [86] frissítése. Állapot: kísérleti.
- [88] B. Manning és R. Colella. RFC 1706: DNS NSAP resource records, 1994 október. Az RFC1637 [87] ajánlást felülíró ajánlás. Állapot: információs.
- [89] R. M. Metcalfe. RFC 222: Subject: System programmer's workshop, 1971 szeptember. Az RFC0212 [64] frissítése. Az RFC0234 [165] tovább frissítette. Állapot: ismeretlen. Nem elérhető az Interneten.
- [90] P. Mockapetris. STD 13: Domain Names — Concepts and Facilities, 1987 november. Lásd még RFC1034, RFC1035 [94, 95].
- [91] P. V. Mockapetris. RFC 882: Domain names: Concepts and facilities, 1983 november. Az RFC1034, RFC1035 [94, 95] elavulttá tette. Az RFC0973 [93] tovább frissítette. Állapot: ismeretlen.

- [92] P. V. Mockapetris. RFC 883: Domain names: Implementation specification, 1983 november. Az RFC1034, RFC1035 [94, 95] elavulttá tette. Az RFC0973 [93] tovább frissítette. Állapot: ismeretlen.
- [93] P. V. Mockapetris. RFC 973: Domain system changes and observations, 1986 január. Az RFC1034, RFC1035 [94, 95] elavulttá tette. Az RFC0882, RFC0883 [91, 92] frissítése. Állapot: ismeretlen.
- [94] P. V. Mockapetris. RFC 1034: Domain names — concepts and facilities, 1987 november. Az RFC0973, RFC0882, RFC0883 [93, 91, 92] ajánlásokat felülíró ajánlás. Lásd még STD0013 [90]. Az RFC1101, RFC1183, RFC1348, RFC1876, RFC1982, RFC2065, RFC2181, RFC2308 [96, 46, 86, 32, 44, 43, 45, 5] tovább frissítette. Állapot: szabvány.
- [95] P. V. Mockapetris. RFC 1035: Domain names — implementation and specification, 1987 november. Az RFC0973, RFC0882, RFC0883 [93, 91, 92] ajánlásokat felülíró ajánlás. Lásd még STD0013 [90]. Az RFC1101, RFC1183, RFC1348, RFC1876, RFC1982, RFC1995, RFC1996, RFC2065, RFC2181, RFC2136, RFC2137, RFC2308 [96, 46, 86, 32, 44, 106, 166, 43, 45, 167, 42, 5] tovább frissítette. Állapot: szabvány.
- [96] P. V. Mockapetris. RFC 1101: DNS encoding of network names and other types, 1989 április. Az RFC1034, RFC1035 [94, 95] frissítése. Állapot: ismeretlen.
- [97] J. C. Mogul. RFC 919: Broadcasting Internet datagrams, 1984 október. Lásd még STD0005 [130]. Állapot: szabvány.
- [98] J. C. Mogul. RFC 922: Broadcasting Internet datagrams in the presence of subnets, 1984 október. Lásd még STD0005 [130]. Állapot: szabvány.
- [99] J. C. Mogul és J. Postel. RFC 950: Internet Standard Subnetting Procedure, 1985 augusztus. Az RFC0792 [128] frissítése. Lásd még STD0005 [130]. Állapot: szabvány.
- [100] K. Moore. RFC 1342: Representation of non-ASCII text in Internet message headers, 1992 június. Az RFC1522 [101] elavulttá tette. Állapot: információs.
- [101] K. Moore. RFC 1522: MIME (Multipurpose Internet Mail Extensions) part two: Message header extensions for non-ASCII text, 1993 szeptember. Az RFC2045, RFC2046, RFC2047, RFC2048, RFC2049 [49, 50, 102, 52, 51] elavulttá tette. Az RFC1342 [100] ajánlást felülíró ajánlás. Állapot: szabványvázlat.
- [102] K. Moore. RFC 2047: MIME (Multipurpose Internet Mail Extensions) part three: Message header extensions for non-ASCII text, 1996 november. Az

- RFC1521, RFC1522, RFC1590 [19, 101, 134] ajánlásokat felülíró ajánlás. Az RFC2184, RFC2231 [53, 54] tovább frissítette. Állapot: szabványvázlat.
- [103] N. Neigus. RFC 542: File transfer protocol, 1973 július. Az RFC0765 [119] elavulttá tette. Az RFC0354 [15] ajánlást felülíró ajánlás. Állapot: ismeretlen. Formátum: TXT=100666 bájt.
 - [104] N. Neigus és J. Postel. RFC 503: Socket number list, 1973 április. Az RFC0739 [113] elavulttá tette. Az RFC0433 [111] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
 - [105] A. G. Nemeth. RFC 43: Proposed meeting, 1970 április. Az RFC0122 [171] frissítése. Állapot: ismeretlen.
 - [106] M. Ohta. RFC 1995: Incremental zone transfer in DNS, 1996 augusztus. Az RFC1035 [95] frissítése. Állapot: javasolt szabvány.
 - [107] D. C. Plummer. RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, 1982 november. Állapot: szabvány.
 - [108] J. Postel. RFC 204: Sockets in use, 1971 augusztus. Az RFC0234 [165] tovább frissítette. Állapot: ismeretlen. Formátum: TXT=1379 bájt.
 - [109] J. Postel. RFC 317: Official host-host protocol modification: Assigned link numbers, 1972 március. Az RFC0604 [112] elavulttá tette. Állapot: ismeretlen. Nem elérhető az Interneten.
 - [110] J. Postel. RFC 349: Proposed standard socket numbers, 1972 május. Lásd még RFC0322, RFC0204 [25, 108]. Az RFC0433 [111] elavulttá tette. Állapot: ismeretlen. Formátum: TXT=1663 bájt.
 - [111] J. Postel. RFC 433: Socket number list, 1972 december. Az RFC0503 [104] elavulttá tette. Az RFC0349 [110] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
 - [112] J. Postel. RFC 604: Assigned link numbers, 1973 december. Az RFC0739 [113] elavulttá tette. Az RFC0317 [109] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
 - [113] J. Postel. RFC 739: Assigned numbers, 1977 november. Az RFC0750 [114] elavulttá tette. Az RFC0604, RFC0503 [112, 104] ajánlásokat felülíró ajánlás. Állapot: régi.
 - [114] J. Postel. RFC 750: Assigned numbers, 1978 szeptember. Az RFC0755 [115] elavulttá tette. Az RFC0739 [113] ajánlást felülíró ajánlás. Állapot: régi. Nem elérhető az Interneten.

- [115] J. Postel. RFC 755: Assigned numbers, 1979 május. Az RFC0758 [116] elavulttá tette. Az RFC0750 [114] ajánlást felülíró ajánlás. Állapot: régi. Nem elérhető az Interneten.
- [116] J. Postel. RFC 758: Assigned numbers, 1979 augusztus. Az RFC0762 [118] elavulttá tette. Az RFC0755 [115] ajánlást felülíró ajánlás. Állapot: régi. Nem elérhető az Interneten.
- [117] J. Postel. RFC 760: DoD standard Internet Protocol, 1980 január. Az RFC0791, RFC0777 [127, 124] elavulttá tette. Állapot: ismeretlen. Nem elérhető az Interneten.
- [118] J. Postel. RFC 762: Assigned numbers, 1980 január. Az RFC0770 [121] elavulttá tette. Az RFC0758 [116] ajánlást felülíró ajánlás. Állapot: régi. Nem elérhető az Interneten.
- [119] J. Postel. RFC 765: File transfer protocol specification, 1980 június. Az RFC0959 [135] elavulttá tette. Az RFC0542 [103] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
- [120] J. Postel. RFC 768: User datagram protocol, 1980 augusztus. Állapot: szabvány. Lásd még STD0006 [122].
- [121] J. Postel. RFC 770: Assigned numbers, 1980 szeptember. Az RFC0776 [123] elavulttá tette. Az RFC0762 [118] ajánlást felülíró ajánlás. Állapot: régi. Nem elérhető az Interneten.
- [122] J. Postel. STD 6: User Datagram Protocol, 1980 augusztus. Lásd még RFC0768 [120].
- [123] J. Postel. RFC 776: Assigned numbers, 1981 január. Az RFC0790 [126] elavulttá tette. Az RFC0770 [121] ajánlást felülíró ajánlás. Állapot: régi. Nem elérhető az Interneten.
- [124] J. Postel. RFC 777: Internet Control Message Protocol, 1981 április. Az RFC0792 [128] elavulttá tette. Az RFC0760 [117] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
- [125] J. Postel. RFC 788: Simple mail transfer protocol, 1981 november. Az RFC0821 [133] elavulttá tette. Az RFC0780 [155] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
- [126] J. Postel. RFC 790: Assigned numbers, 1981 szeptember. Az RFC0820 [132] elavulttá tette. Az RFC0776 [123] ajánlást felülíró ajánlás. Állapot: régi.
- [127] J. Postel. RFC 791: Internet Protocol, 1981 szeptember. Az RFC0760 [117] ajánlást felülíró ajánlás. Lásd még STD0005 [130]. Állapot: szabvány.

- [128] J. Postel. RFC 792: Internet Control Message Protocol, 1981 szeptember. Az RFC0777 [124] ajánlást felülíró ajánlás. Az RFC0950 [99] tovább frissítette. Lásd még STD0005 [130]. Állapot: szabvány.
- [129] J. Postel. RFC 793: Transmission control protocol, 1981 szeptember. Lásd még STD0007 [131]. Állapot: szabvány.
- [130] J. Postel. STD 5: Internet Protocol: DARPA Internet Program Protocol Specification, 1981 szeptember. Lásd még RFC0791, RFC0792, RFC0919, RFC0922, RFC0950, RFC1112 [127, 128, 97, 98, 99, 37].
- [131] J. Postel. STD 7: Transmission Control Protocol: DARPA Internet Program Protocol Specification, 1981 szeptember. Lásd még RFC0793 [129].
- [132] J. Postel. RFC 820: Assigned numbers, 1982 augusztus. Az RFC0870 [142] elavulttá tette. Az RFC0790 [126] ajánlást felülíró ajánlás. Állapot: régi.
- [133] J. Postel. RFC 821: Simple mail transfer protocol, 1982 augusztus. Lásd még STD0010 [136]. Az RFC0788 [125] ajánlást felülíró ajánlás. Állapot: szabvány.
- [134] J. Postel. RFC 1590: Media type registration procedure, 1994 március. Az RFC2045, RFC2046, RFC2047, RFC2048, RFC2049 [49, 50, 102, 52, 51] elavulttá tette. Az RFC1521 [19] frissítése. Állapot: információs.
- [135] J. Postel és J. K. Reynolds. RFC 959: File transfer protocol, 1985 október. Az RFC0765 [119] ajánlást felülíró ajánlás. Az RFC2228 [61] tovább frissítette. Állapot: szabvány.
- [136] Jonathan B. Postel. STD 10: Simple mail transfer protocol, 1982 augusztus. Lásd még RFC0821, RFC1869 [133, 75]. Az RFC2821 [?] elavulttá tette. Az RFC788, RFC780, RFC772 [125, 155, 154] ajánlásokat felülíró ajánlás.
- [137] Y. Rekhter, B. Moskowitz, D. Karrenberg és G. de Groot. RFC 1597: Address allocation for private internets, 1994 március. Az BCP0005, RFC1918 [?, 138] elavulttá tette. Állapot: információs.
- [138] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot és E. Lear. RFC 1918: Address allocation for private internets, 1996 február. Lásd még BCP0005 [?]. Az RFC1627, RFC1597 [83, 137] ajánlásokat felülíró ajánlás. Állapot: jelenleg javasolt gyakorlat.
- [139] J. Reynolds és J. Postel. RFC 1340: ASSIGNED NUMBERS, 1992 július. Az RFC1700 [140] elavulttá tette. Az RFC1060 [150] ajánlást felülíró ajánlás. Állapot: régi.
- [140] J. Reynolds és J. Postel. RFC 1700: ASSIGNED NUMBERS, 1994 október. Lásd még STD0002 [141]. Az RFC1340 [139] ajánlást felülíró ajánlás. Állapot: szabvány.

- [141] J. Reynolds és J. Postel. STD 2: Assigned numbers, 1994 október. Lásd még RFC1700 [140]. Az RFC1340 [139] ajánlást felülíró ajánlás.
- [142] J. K. Reynolds és J. Postel. RFC 870: Assigned numbers, 1983 október. Az RFC0900 [143] elavulttá tette. Az RFC0820 [132] ajánlást felülíró ajánlás. Állapot: régi.
- [143] J. K. Reynolds és J. Postel. RFC 900: Assigned numbers, 1984 június. Az RFC0923 [144] elavulttá tette. Az RFC0870 [142] ajánlást felülíró ajánlás. Állapot: régi.
- [144] J. K. Reynolds és J. Postel. RFC 923: Assigned numbers, 1984 október. Az RFC0943 [145] elavulttá tette. Az RFC0900 [143] ajánlást felülíró ajánlás. Állapot: régi.
- [145] J. K. Reynolds és J. Postel. RFC 943: Assigned numbers, 1985 április. Az RFC0960 [146] elavulttá tette. Az RFC0923 [144] ajánlást felülíró ajánlás. Állapot: régi.
- [146] J. K. Reynolds és J. Postel. RFC 960: Assigned numbers, 1985 december. Az RFC0990 [147] elavulttá tette. Az RFC0943 [145] ajánlást felülíró ajánlás. Állapot: régi.
- [147] J. K. Reynolds és J. Postel. RFC 990: Assigned numbers, 1986 november. Az RFC1010 [148] elavulttá tette. Az RFC0960 [146] ajánlást felülíró ajánlás. Az RFC0997 [149] tovább frissítette. Állapot: régi.
- [148] J. K. Reynolds és J. Postel. RFC 1010: Assigned numbers, 1987 május. Az RFC1060 [150] elavulttá tette. Az RFC0990 [147] ajánlást felülíró ajánlás. Állapot: régi.
- [149] J. K. Reynolds és J. Postel. RFC 997: Internet numbers, 1987 március. Az RFC1020, RFC1117 [151, 153] elavulttá tette. Az RFC0990 [147] frissítése. Állapot: ismeretlen.
- [150] J. K. Reynolds és J. Postel. RFC 1060: Assigned numbers, 1990 március. Az RFC1340 [139] elavulttá tette. Az RFC1010 [148] ajánlást felülíró ajánlás. Állapot: régi.
- [151] S. Romano és M. K. Stahl. RFC 1020: Internet numbers, 1987 november. Az RFC1062, RFC1117, RFC1166 [152, 153, 73] elavulttá tette. Az RFC0997 [149] ajánlást felülíró ajánlás. Állapot: ismeretlen.
- [152] S. Romano, M. K. Stahl, és M. Recker. RFC 1062: Internet numbers, 1988 augusztus. Az RFC1117, RFC1166 [153, 73] elavulttá tette. Az RFC1020 [151] ajánlást felülíró ajánlás. Állapot: ismeretlen.

- [153] S. Romano, M. K. Stahl és M. Recker. RFC 1117: Internet numbers, 1989 augusztus. Az RFC1166 [73] elavulttá tette. Az RFC1062, RFC1020, RFC0997 [152, 151, 149] ajánlásokat felülíró ajánlás. Állapot: információs.
- [154] S. Sluizer és J. Postel. RFC 772: Mail transfer protocol, 1980 szeptember. Az RFC0780 [155] elavulttá tette. Állapot: ismeretlen. Nem elérhető az Interneten.
- [155] S. Sluizer és J. Postel. RFC 780: Mail transfer protocol, 1981 május. Az RFC0788 [125] elavulttá tette. Az RFC0772 [154] ajánlást felülíró ajánlás. Állapot: ismeretlen. Nem elérhető az Interneten.
- [156] Charles E. Spurgeon. *Ethernet: The Definitive Guide*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2000 február.
- [157] Hal Stern. *Managing NFS and NIS*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 1991 június.
- [158] Sun Microsystems, Inc. RFC 1050: RPC: Remote procedure call protocol specification, 1988 április. Az RFC1057 [159] elavulttá tette. Állapot: régi.
- [159] Sun Microsystems, Inc. RFC 1057: RPC: Remote procedure call protocol specification: Version 2, 1988 június. Az RFC1050 [158] ajánlást felülíró ajánlás. Állapot: információs.
- [160] Barbara Fritchman Thompson és Robert Bruce Thompson. *PC Hardware in a Nutshell*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2. kiadás, 2002.
- [161] Robert Bruce Thompson és Barbara Fritchman Thompson. *PC Hardware in a Nutshell*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2000 október.
- [162] R. Troost és S. Dorner. RFC 1806: Communicating presentation information in Internet messages: The content-disposition header, 1995 június. Az RFC2183 [163] tovább frissítette. Állapot: kísérleti.
- [163] R. Troost, S. Dorner és K. Moore. RFC 2183: Communicating presentation information in Internet messages: The content-disposition header field, 1997 augusztus. Az RFC1806 [162] frissítése. Az RFC2184, RFC2231 [53, 54] tovább frissítette. Állapot: javasolt szabvány.

- [164] A. Vezza. RFC 207: September network working group meeting, 1971 augusztus. Az RFC0212 [64] elavulttá tette. Állapot: ismeretlen. Formátum: TXT=3356 bájt.
- [165] A. Vezza. RFC 234: Network working group meeting schedule, 1971 október. Az RFC0222, RFC0204 [89, 108] frissítése. Állapot: ismeretlen. Nem elérhető az Interneten.
- [166] P. Vixie. RFC 1996: A mechanism for prompt notification of zone changes (DNS NOTIFY), 1996 augusztus. Az RFC1035 [95] frissítése. Állapot: javasolt szabvány.
- [167] P. Vixie, Editor, S. Thomson, Y. Rekhter és J. Bound. RFC 2136: Dynamic updates in the domain name system (DNS UPDATE), 1997 április. Az RFC1035 [95] frissítése. Állapot: javasolt szabvány.
- [168] D. Waitzman. RFC 1149: Standard for the transmission of IP datagrams on avian carriers, 1990 április. Állapot: kísérleti.
- [169] J. E. White. RFC 74: Specifications for network use of the UCSB on-line system, 1970 október. Az RFC0217, RFC0225 [172, 59] tovább frissítette. Állapot: ismeretlen. Nem elérhető az Interneten.
- [170] J. E. White. RFC 105: Network specifications for remote job entry and remote job output retrieval at UCSB, 1971 március. Az RFC0217 [172] tovább frissítette. Állapot: ismeretlen.
- [171] J. E. White. RFC 122: Network specifications for UCSB's simple-minded file system, 1971 április. Az RFC0217, RFC0269, RFC0399, RFC0043, RFC0431 [172, 24, 77, 105, 78] tovább frissítette. Állapot: ismeretlen. Nem elérhető az Interneten.
- [172] J. E. White. RFC 217: Specifications changes for OLS, RJE/RJOR, and SMFS, 1971 szeptember. Az RFC0074, RFC0105, RFC0122 [169, 170, 171] frissítése. Állapot: ismeretlen.
- [173] Clinton Wong. *HTTP Pocket Reference: Hypertext Transfer Protocol*. O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, és 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2000 május.

Tárgymutató

(none), 244
*, 18, 19, 26, 47, 135, 246, 247
*/ , 136
. , 26
-, 19, 26, 279
--to-destination *cím*, 229
. , 94, 99, 135, 138–140
.LP, 34
.SH, 33
.TH, 32
.config, 40
.htaccess, 279, 280
.message, 251
/, 19, 26, 135, 266, 267

/etc/modules, 47
/etc/modules.conf, 46, 48, 78, 80
/etc/named.conf, 136, 144, 145, 147,
 149
/etc/network/interfaces, 104–106
/etc/networks, 97
/etc/nsswitch.conf, 95, 98
/etc/protocols, 127, 190, 205
/etc/resolv.conf, 98, 99, 143, 154
/etc/rpc, 126, 129
/etc/services, 114, 117, 123, 126,
 129, 208
/etc/sysconfig/network, 101, 243
/etc/syslog.conf, 13–15, 18
/etc/syslogd.conf, 16, 18, 20
/etc/vsftpd.conf, 250
/etc/vsftpd/vsftpd.conf, 250
/etc/xinetd.conf, 125, 130
/etc/xinetd.d, 130
/etc/yp.conf, 248
/etc/ypserv.conf, 246, 247
/home/, 11
/home/ftp/, 252
/lib/modules/, 42
/proc/, 71, 214
/proc/sys/kernel/hostname, 98
/proc/sys/net/ipv4/ip_forward,
 196
/proc/dma, 71
/proc/interrupts, 71
/proc/iports, 71
/proc/modules, 44
/proc/pci, 74
/proc/sys/kernel/modprobe, 48
/sbin/modprobe, 48
/usr/sbin/, 23
/usr/share/hwdate, 73
/usr/src/, 38
/usr/src/linux, 38
/usr/src/linux-2.6.5, 38
/var/lib/dhcp/dhclient.leases,
 100
/var/log/apache/, 266
/var/log/httpd/, 266
/var/ftp/, 252
/var/lib/dhcp/dhcp.leases, 101
/var/log/, 13, 14
/var/log/crit.log, 18
/var/log/vsftpd.log, 252
/var/named/, 142
/var/spool/crond/, 28
/var/spool/mail/, 255
/var/www/cgi-bin/, 284
/var/yp/, 244, 245
:, 209
;, 18
<Files állománynév>...</Files>,
 279
<IfModule mod_modul-
 név.c>...</IfModule>,

kiszolgálók : ügyfelek : parancs, 134
modulnév_module, 269
szöveg matches szabkif., 188
 0-9, 26
 0.0.0.0, 89
 0.0.0.0/0, 201
 0x, 80
 10.0.0.0, 67
 10.0.0.0/255.0.0.0, 90, 91
 10.10.0.0, 246
 1000BASE-T, 56
 100BASE-T, 56
 10BASE-2, 55
 10BASE-T, 56
 10am Jul 31, 22
 11:30, 22
 127.0.0.0/255.0.0.0, 90, 91
 127.0.0.0/8, 220
 127.0.0.1, 92, 96, 208
 172.16.0.0, 67
 172.17.0.0/16, 220
 172.31.0.0, 67
 192.168.0.0, 67
 192.168.255.0, 67
 3c501, 82
 8139too, 47, 77
 1, 160, 196
 2, 160
 4, 160
 8, 160
 550, 192, 261
 A, 139–143
 a–z, 19
 a3d, 45
 ACCEPT, 201, 203, 206, 221
 access, 269, 280
 access_times=óra:perc-óra:perc, 129
 ACK, 180–182, 185, 191, 192, 209, 210
 ACK, 209
 Ack, 182
 acknowledge, 157, 181
 acknowledgement, 180
 actions, 269
 AddHandler cgi-script kiterjesztés,
 283
 AddIcon ikon kiterjesztés1 kiterjesz-
 tés2, 275
 AddIconByType ikon típus1 típus2,
 275
 Addr=IP_cím, 258
 address resolution protocol, 171
 address cím, 105
 addrtype, 214
 AFS, 222
 alert, 17
 alias, 47, 48, 269, 283, 284
 alias név modul, 47
 alias:, 45
 ALL, 136, 209
 All, 280
 all, 205
 all_squash, 241
 Allow, 281
 Allow from cím, 280
 Allow,Deny, 281
 allow-query {cím1; cím2;};, 147
 allow-recursion {cím1; cím2;};,
 147
 allow-transfer {cím1; cím2;};,
 147
 AllowOverride jog1 jog2, 280
 anacron, 22, 28, 29
 anacrontab, 29
 and, 187
 anon_max_rate, 252
 anon_root, 252
 anongid, 241
 anonuid, 241
 anonymous, 269
 anonymous, 251–253
 anonymous_enable, 251
 anouid, 241
 any, 147, 165
 anywhere, 201
 apropos, 33, 34
 arch/i386/boot/bzImage, 42

- ARP, 171–173, 179, 180, 189, 195
arp, 158, 171
arp.dst.hw, 189
arp.dst.hw_mac, 189
arp.dst.proto_ipv4, 189
arp.src.hw, 189
arp.src.hw_mac, 189
arp.src.proto_ipv4, 189
ASCII, 238
asis, 269
at, 21–24, 30
at.allow, 24
at.deny, 24
at>, 23
atd, 23, 25
atq, 22, 24
atrm, 22, 24
auth, 269
auth_anon, 269
auth_dbm, 270
AuthConfig, 280
AuthDigestFile állománynév, 290
AuthDigestGroupFile állománynév,
 290
AuthGroupFile állománynév, 290
AUTHOR, 31
author:, 45
authpriv, 17
AuthType típus, 288
AuthUserFile állománynév, 289
auto, 11
auto csatoló1 csatoló2, 104
autoindex, 270, 274, 276, 280
autoprobe, 78

Base address:, 85
Basic, 289
batch, 23, 24
Berkeley Internet name daemon, 136
BIND, 136, 142, 146, 150
bind, 141, 145, 146
bind=ip_cím, 129
block-major-n, 49
BOOTP, 103, 104, 151–153

bootp, 103, 104
Broadcast, 170
broadcast, 62
broadcast cím, 105
BUGS, 31
bytes, 202

caching name server, 145
caching-nameserver, 146
canonical name, 140
capture filters, 184
Cat1, 55
Category 1, 55
cf.m4, 257
CGI, 269, 270, 272, 280, 283–286
cgi, 270, 283, 284
cgid, 270
char-major-n, 49
charset_lite, 270
closed, 121
CNAME, 140, 141
collision, 62
coloring rules, 185
common gateway interface, 283
condition, 214
conf/httpd.conf, 266
conf/magic, 276
connection per second, 129
CONNMARK, 214
connmark, 214
connrate, 214
contain, 188
coordinated universal time, 160
core, 279, 288
cps=szám szám, 129
crit, 17
cron, 17, 21, 30
crond, 25–28
crontab, 25–28
crossover cable, 58

daemon, 17
DAEMON_OPTIONS('kapcsolók'), 258
datagram, 123

DAV, 270
dav, 270
db, 95
DBM, 270
`ddns-update-style interim;`, 154
de-SNAT, 225, 227
debug, 17
default gateway, 89
default-lease-time *szám*;, 158
DefaultIcon *ikon*, 275
defaults {...}, 125
DefaultType *típus*, 268
deflate, 270
delete, 90
deny, 247
Deny from *cím*, 281
Deny, Allow, 282
deny_file, 252
depends:, 46
depmod, 46–48
DESCRIPTION, 31
description:, 45
DescriptionWidth=*szám*, 275
Destination, 89
destination, 201
destination network address translation, 222
destination port, 201
destination-unreachable, 207
device driver, 70, 76
DEVICE=*csatolónév*, 102
dgram, 123, 124, 127
dhclient, 100, 101
dhclient-script, 101
DHCP, 100, 101, 103–106, 151–160,
 225
dhcp, 103, 104
DHCPACK, 157
dhcpd, 153, 157
DHCPDISCOVER, 157
DHCPOAK, 157
DHCPOFFER, 157
DHCPREQUEST, 157
Digest, 289

dir, 270, 272–274, 276
direct memory access, 71
DirectoryIndex *állomány1* *állomány2*, 273
dirmassage_enable, 251
DISCARD, 261
discover, 157
display filter, 185
distributed authoring and versioning, 270
DMA, 71
DNAT, 222, 223, 228
DNAT, 228, 229
DNS, 93–96, 98, 99, 118, 128, 136–
 147, 149, 150, 154, 173, 174,
 176–179, 188, 200, 237, 260,
 262, 263, 267
dns, 95, 96
Documentation/Changes, 38
DocumentRoot, 274
domain, 118
domain name service, 118
domain name system, 93
domainname, 243
don't fragment, 175
download_enable, 251
dpt, 201
DROP, 203, 206
dstlimit, 214
dynamic host configuration protocol, 100, 151
echo, 117, 208, 286
echo request, 170
echo-reply, 207
echo-request, 207
edit, 25
EIA/TIA-568, 55
emerg, 17, 18
env, 270
err, 17
ESTABLISHED, 215
eth.dst, 188
eth.len, 188

eth.src, 188
eth0, 47–49, 76–78, 80–83, 86, 90, 91, 100, 102, 104, 105, 206, 220
eth1, 47–49, 76–78, 80–82, 86, 102, 104
ether, 86
ethereal, 164, 165, 168, 169, 172–174, 176, 184, 188
Ethernet, 85
Ethernet address, 61
EXAMPLES, 31
EXCEPT, 136
except, 136
ExecCGI, 280
expires, 270
exports, 241
ext_filter, 270
false, 158
FancyIndexing, 274
fat, 47
FEATURE(access_db), 260
FEATURE(relay_entire_domain), 260
FEATURE(relay_hosts_only), 260
FEATURE(use_cw_file), 263
FEATURE(virtusertable), 264
ff:ff:ff:ff:ff, 62
file transfer protocol, 117, 250
file-group, 289
file-owner, 289
file_cache, 270
FILES, 31
files, 95
filter, 222, 224, 227
filtered, 121
FIN, 181, 191
FIN, 209
finalize, 181
finger, 118, 133
firewall, 197
fixed-address cím;, 156
Flags, 89
floppy, 49
FoldersFirst, 275
FollowSymLinks, 280
FORWARD, 199, 200, 203, 204, 206, 207, 220, 222, 223, 227
forward, 138
fragmentation-needed, 207
frame.cap_len, 188
frame.number, 188
frame(pkt_len, 188
FTP, 117, 192, 213, 222, 250, 251, 253
ftp, 117, 251–253
ftp-data, 117
ftp-ssl, 251
ftp.request, 192
ftp.request.arg, 193
ftp.request.command, 193
ftp.response, 193
ftp.response.arg, 193
ftp.response.code, 193
G, 89
Gateway, 89
gateway cím, 105
GATEWAY=cím, 102
GATEWAYDEV=csatolónév, 102
Genmask, 89
GET, 193
GET, 183
GID, 240
GIF, 194
group {...}, 155
group csoport1 csoport2, 289
Group csoporthnév, 267
group.bygid, 244
groupbyname, 244
group=csoport, 128
guest_username, 252
H, 89
hardware ethernet ethernet_cím;, 156
headers, 271
help, 255
hide_file, 252
hint, 138

host gépnév {...}, 155
host-unreachable, 207
hostname, 98
HOSTNAME=név, 102
hosts, 95
htdigest, 287–290
HTML, 238, 271
htpasswd, 287–289
HTTP, 118, 183, 184, 193, 194, 265,
 270, 286
http, 118
http.accept, 193
http.accept_encoding, 193
http.accept_language, 193
http.content_encoding, 193
http.content_length, 193
http.content_type, 193
http.requestes, 193
http.request.method, 193
http.request.uri, 193
http.response, 193
http.response.code, 193
http.server, 193
http.user_agent, 194
httpd, 266, 272
httpd.conf, 266, 269
https, 119
hu, 94
hub, 63
HWaddr, 85
hypertext transfer protocol, 118, 183,
 265
I/O, 71, 72, 80, 81, 85
IBM, 69
ICMP, 64, 112, 113, 170, 175, 176,
 205–208, 214, 217, 218
icmp, 205, 207, 214
icmp-host-prohibited, 205
icmp-host-unreachable, 205
icmp-net-prohibited, 205
icmp-net-unreachable, 205
icmp-port-unreachable, 205
icmp-proto-unreachable, 205
IconsAreLinks, 275
IEEE 802.3, 55
Iface, 89
iface csatoló család indítás, 104
ifcfg-csatolónév, 102
ifconfig, 76, 82–87
ifdown, 104
ifup, 102, 104
IMAP, 118
imap, 118, 271
IN, 139
in, 202
in-addr.arpa, 137, 140
include, 47, 271
include('állománynév'), 257
includedir könyvtárnev, 125
index files, 270
index.htm, 274
index.html, 273, 274, 276
Indexes, 280
IndexOptions kapcsoló1 kapcsoló2,
 274
industrial standard architecture, 71
inet, 104, 105
inet addr:, 85
inet6, 105
inet6 addr:, 85
inetd, 122–124, 133, 134
info, 17, 271
INPUT, 199, 200, 204, 206, 207, 210,
 220, 222, 223, 228
input/output, 70
insmod, 46
install modul parancs, 47
instances=szám, 128
interactive mail access protocol, 118
interface, 82
INTERNAL, 126
internal, 124
internet bootstrap protocol, 151
internet control message protocol, 112
internet protocol, 64
internet protocol version 4, 64
internet protocol version 6., 64

- internet relay chat, 119
Internet Systems Consortium, 153
Internet Systems Consortium DHCP Client, 100
interrupt request, 71
Interrupt:, 85
INVALID, 215
io, 80
IP, 64–67, 82, 85–87, 89–97, 100, 102, 105, 107–112, 128, 135–137, 140, 151–154, 156–159, 163, 168, 170–176, 178, 179, 184, 186, 189, 190, 195–197, 200, 201, 205, 214, 219, 220, 222, 223, 225–227, 231, 236, 245, 246, 249, 267, 281
IP cím, 189
IP spoofing, 219
IP tables, 197, 222
ip.checksum_bad, 190
ip.dst, 190
ip.flags, 190
ip.flags.df, 190
ip.flags.mf, 190
ip.frag_offset, 190
ip.fragment, 190
ip.hdr_len, 190
ip.len, 190
ip.proto, 190
ip.src, 190
ip.tos, 190
ip.tos.cost, 190
ip.tos.delay, 190
ip.tos.reliability, 190
ip.tos.throughput, 190
ip.ttl, 190
ip_tables, 198
IPADDR=*IP_cím*, 102
ipchains, 197
ipfwadm, 197
iprange, 214
iptables, 197, 198, 200–202, 204–210, 213–215, 218, 220, 222, 224, 225, 228
iptables, 203
iptables --list [*név*] [-n] [-v], 200
IPv4, 64, 67, 175, 189, 190, 196
IPv6, 64, 85, 179
ipw2100, 47
IRC, 119, 222
ircd, 119
IRQ, 71, 72, 85
ISA, 71–73, 79, 81
kern, 17, 18
kerneld, 86
kiszolgáló, 111
KNOWN, 136
length, 214
levélbombázás, 255
libnas1, 233
libwrap, 133
license:, 45
Limit, 280
limit, 214
Link encap:, 85
linux, 257
linux-x.x.x, 38
linux/socket.h, 49
listen, 251
Listen *IP_cím:kapu*, 267
lo, 90–92, 105, 220
LoadModule *név* *állománynév*, 269
local, 257
Local Loopback, 85
local-host-names, 263
local0, 17
local7, 17
local_enable, 251
localhost.localdomain, 102
localn, 17
LOG, 231
log files, 13
log_config, 271
logger, 20
loopback, 104

- lpr, 17
ls, 11
lsmod, 43, 44
lspci, 74, 75, 77

m4, 256, 259
MAC, 61, 184, 189
mac, 214
mail, 17
mail exchanger, 141
mail.*, 16
MAILER(levelező), 257
maillog, 13
make, 39–42, 244, 256, 260, 262, 265
Makefile, 244, 245, 247, 265
makemap, 260, 265
makewhatis, 33, 34
man, 30, 106, 158, 186, 188, 213, 246,
 250, 265
mangle, 222
mangle, 223
Mask:, 85
MASQUERADE, 225
master, 137, 138
masters {cím1; cím2;}, 144
masvalami.hu, 263
max-lease-time szám;, 157
max_clients, 252
maximal transfer unit, 60
media access control, 61
message submission agent, 258
message transfer agent, 258
messages, 13
MIME, 193, 268, 275
mime, 271, 275, 283, 284
mime_magic, 271, 275, 276
MimeTypeFile állománynév, 276
mkfifo, 19
mod_auth, 289
mod_auth_digest, 289, 290
mod_cgi, 270
mod_modulnév.so, 269
modinfo, 44, 45, 77
modprobe, 46–49, 81, 82

modules, 269
modules.dep, 46, 48
mount, 241, 242
mport, 214
MSA, 258
msdos, 47
MTA, 258
MTU, 60
multicast address, 65
multipurpose internet mail extension,
 268
MX, 141

NÉV, 33
NAME, 31, 33
name server, 140
Name=név, 258
NAME=teljes_név, 102
named, 136, 137, 145
nameserver cím, 99
NameWidth=szám, 274
NASL, 233, 234
NAT, 222
nat, 223, 224, 228
NDS, 159
negotiation, 271
nessus, 232–237
nessus attack scripting language, 233
nessus-adduser, 234
nessus-core, 233
nessus-libraries, 233
nessus-plugins, 234
nessusd, 233–235
net-pf-n, 49
netfilter, 197, 213
netmask alhálózati maszk, 105
NETMASK=maszk, 102
netware directory service, 159
network address translation, 222
network file system, 239
network filter, 197
network information service, 242
network interface card, 69
network cím, 106

network-unreachable, 207
networks, 95
NEW, 215
news, 17
next-server *cím*;, 158
NFS, 239–242
nfs, 241, 242
NIC, 69, 76
NIS, 96, 102, 160, 239, 242–249
nis, 96, 248, 249
NIS domain, 242
NIS+, 95
NISDOMAIN, 243
NISDOMAIN=*név*, 102
nisplus, 95
nmap, 120, 121
NO, 250
no, 247
no_access=*lista*, 129
no_all_squash, 241
no_anon_password, 251
no_root_squash, 241
nobody, 252
NONE, 209
None, 280
none, 103, 247
nopriv_user, 252
not, 187
not acknowledged, 157
notice, 17, 18
now + 7days, 22
nowait, 124
NS, 140–142
nth, 214

offer, 157
OK, 261
only_from=*lista*, 128
open, 121
opt, 201
option arp-cache-timeout *szám*,
 158
option broadcast-address *cím*;
 159
option default-ip-ttl *szám*;, 159
option domain-name-servers *cím1*,
 cím2;, 154
option font-servers *cím1*, *cím2*;
 159
option interface-mtu *szám*;, 159
option lpr servers *cím1*, *cím2*;
 159
option ndots:*n*, 99
option netbios name servers
 cím1, *cím2*;, 159
option netbios node type *szám*;
 159
option nis servers *cím1*, *cím2*;
 160
option nisplus servers *cím1*,
 cím2;, 160
option nntp server *cím1*, *cím2*;
 160
option pop server *cím1*, *cím2*;
 160
option rotate, 99
option routers *cím*;, 154
option smtp server *cím1*, *cím2*;
 160
option subnet-mask *maszk*;, 154
option time offset *szám*;, 160
option time servers *cím1*, *cím2*;
 160
option timeout:*n*, 99
option x display manager *cím1*,
 cím2;, 161
OPTIONS, 31
Options, 280
options, 47
options {...};, 136
Options kapcsoló1 kapcsoló2, 279
or, 187
Order sorrend, 281
org, 94
OSTYPE('operációs rendszer'), 257
out, 202
OUTPUT, 218
OUTPUT, 199, 200, 204, 206, 217, 222,

- 223, 228
owner, 214
PAM, 30, 252
pam_service_name, 252
PARANOID, 136
parm:, 45
parport_lowlevel, 49
passwd, 249
passwdbyname, 244, 245
passwdbyuid, 244, 245
PC Card, 75
PCI, 73–75, 78, 79
PCI ID, 73
pci.ids, 73–75
PCMCIA, 75, 76
per_source=szám, 129
peripheral component interconnect, 73
Personal Computer Memory Card International Association, 75
PID, 267
ping, 169–172, 208
ping reply, 170
pkts, 202
pl, 94
plug and play, 72
plug and pray, 72
PNP, 72
pointer, 140
policy, 200
POP3, 118, 160
pop3, 118, 130, 131
pop3s, 119
port, 108
port, 247
port-unreachable, 207
Port=szám, 258
port=szám, 129
portmapper, 119, 120
POST, 193
post office protocol, 118
POSTROUTING, 223, 224
PREROUTING, 223, 228
private ports, 114
privileged ports, 113
programok
 anacron, 29
 apropos, 33
 at, 22
 atq, 22
 atrm, 22
 crond, 25
 crontab, 26
 depmod, 46
 dhcpd, 153
 hostname, 98
 htpasswd, 288
 httpd, 266
 logger, 20
 lsmod, 43
 lspci, 74
 makewhatis, 34
 man, 30
 mkfifo, 19
 modprobe, 48
 ping, 169
 rmmod, 45
 route, 88
 syslogd, 16
 traceroute, 176
 xinetd, 125
 ypserv, 248
PROMISC, 85
promiscuous mode, 163
prot, 201
protocol family, 49
protocol=protokoll, 127
PSH, 209
PTR, 140
PTR, 140, 143
rövidítések
 1000BASE-T, 56
 100BASE-T, 56
 10BASE-2, 55
 10BASE-T, 56
 ACK, 180–182, 185, 191, 192, 209,
 210

- AFS, 222
ARP, 171–173, 179, 180, 189, 195
ASCII, 238
BIND, 136, 142, 146, 150
BOOTP, 103, 104, 151–153
CGI, 269, 270, 272, 280, 283–286
DAV, 270
DBM, 270
de-SNAT, 225, 227
DHCP, 100, 101, 103–106, 151–160, 225
DMA, 71
DNAT, 222, 223, 228
DNS, 93–96, 98, 99, 118, 128, 136–147, 149, 150, 154, 173, 174, 176–179, 188, 200, 237, 260, 262, 263, 267
EIA/TIA-568, 55
FIN, 181, 191
FTP, 117, 192, 213, 222, 250, 251, 253
GET, 193
GID, 240
GIF, 194
HTML, 238, 271
HTTP, 118, 183, 184, 193, 194, 265, 270, 286
I/O, 71, 72, 80, 81, 85
IBM, 69
ICMP, 64, 112, 113, 170, 175, 176, 205–208, 214, 217, 218
IEEE 802.3, 55
IMAP, 118
IP, 64–67, 82, 85–87, 89–97, 100, 102, 105, 107–112, 128, 135–137, 140, 151–154, 156–159, 163, 168, 170–176, 178, 179, 184, 186, 189, 190, 195–197, 200, 201, 205, 214, 219, 220, 222, 223, 225–227, 231, 236, 245, 246, 249, 267, 281
IP cím, 189
IPv4, 64, 67, 175, 189, 190, 196
IPv6, 64, 85, 179
IRC, 119, 222
IRQ, 71, 72, 85
ISA, 71–73, 79, 81
MAC, 61, 184, 189
MIME, 193, 268, 275
MTU, 60
NASL, 233, 234
NAT, 222
NDS, 159
NFS, 239–242
NIC, 69, 76
NIS, 96, 102, 160, 239, 242–249
NIS+, 95
OUTPUT, 218
PAM, 30, 252
PC Card, 75
PCI, 73–75, 78, 79
PCI ID, 73
PCMCIA, 75, 76
PID, 267
PNP, 72
POP3, 118, 160
POST, 193
PTR, 140
RFC, 12
RJ-45, 56–58, 60
ROM, 152
RPC, 119, 120, 126, 131, 240
SMTP, 118, 160, 192, 254, 255, 257
SNAT, 222–227
SOA, 139
SSH, 251
SSI, 271
SSL, 251
STP, 55, 56
SYN, 180, 181, 183, 191, 192, 209, 210, 213
TCP, 64, 111–114, 117, 123, 124, 126, 131, 178, 180–185, 191, 192, 209, 210, 213, 215
TCP/IP, 64, 85, 100
TFTP, 160
TTL, 138, 139, 159, 175, 176, 215

UDP, 64, 109–111, 113, 114, 117, 123, 124, 127, 174, 176–178, 191, 208, 209, 213, 215
UID, 240, 243
un-DNAT, 229
USB, 49
UTC, 160
UTP, 55, 56, 58, 59
UUCP, 257
Win, 182, 192
WINS, 160
XDMCP, 119
random, 214
range *cím1* *cím2*; , 153
raw, 127, 223
registered jack, 56
redirect=ip_cím kapu, 129
registered ports, 114
REJECT, 206, 217, 261
RELATED, 215
RELAY, 261
relaying, 259
remote procedure call, 118, 119, 240
remote shell, 117
remove, 47
request, 157
Require követelmény, 289
RETURN, 219, 220
rewrite, 271
RFC, 12
RJ-45, 56–58, 60
rlimit_as=, 130
rmmod, 44, 45
ro, 241
ROM, 152
root_squash, 241
round-robin, 225, 229
route, 76, 87–90, 97
route table, 87
router, 195
RPC, 119, 120, 126, 131, 240
RPC, 118, 126
rpc_number=szám, 129
rsh, 117
RST, 209
rw, 241
RX packets:, 85
Scope:, 85
ScriptAlias cím könyvtár, 284
search, 99
search tartománynév, 99
second level domain, 94
secure, 13
secure shell, 117
securenets, 245, 246
sedmail.cf, 258
SEE ALSO, 31
sendmail, 254–265
sendmail.mc, 260, 262–264
Seq, 181, 191
sequence, 181
server, 140
server side includes, 271
server=állománynév, 128
server_args=argumentumok, 128
ServerAdmin levélcím, 267
ServerName név:kapu, 267
ServerRoot, 267, 269, 276
service név {...}, 125
set, 286
setenvif, 271
shadowbyname, 245
shadow.byuid, 247
shielded twisted pair, 55
SIGHUP, 20, 130, 131, 134
simple mail transfer protocol, 118, 254
slave, 138
smart relay host, 259
SMTP, 118, 160, 192, 254, 255, 257
smtp, 118, 254, 257
smtp.req, 192
smtp.req.command, 192
smtp.req.parameter, 192
smtp.response.code, 192
smtp.rsp, 192
smtp.rsp.parameter, 192
SNAT, 222–227

SNAT, 224, 225
snd-card-*n*, 49
so, 271
SOA, 139
SOA, 139–142, 144
socket_type=*típus*, 126
source, 201
source network address translation, 222
speling, 271
squash, 241
SSH, 251
ssh, 117, 210
SSI, 271
SSL, 251
ssl, 271
ssl_enable, 251
start, 15, 101
start of authority, 139
state, 215
static, 104, 105
status, 271
stmp, 257
stop, 14, 101
STP, 55, 56
stream, 123
stream, 123, 124, 126, 127
subnet, 155
subnet *cím netmask maszk{...}*, 153
suexec, 272
sunrpc, 118
switch, 63
SymLinksIfOwnerMatch, 280
SYN, 180, 181, 183, 191, 192, 209, 210, 213
SYN, 209
syn, 180
sync, 241
synchronize, 181
synchronize sequence numbers, 180
SYNOPSIS, 31
syslog, 17, 194
syslog.facility, 194
syslog.level, 194
syslog.msg, 194
syslogd, 14–16, 19, 20
system log, 13
system-config-bind, 150
system-config-network, 103
target, 199
target, 201
TCP, 64, 111–114, 117, 123, 124, 126, 131, 178, 180–185, 191, 192, 209, 210, 213, 215
tcp, 123, 124, 127, 134, 185, 205, 209, 215
tcp dpt:
 tcp dpt:
 ssh, 201
tcp.checksum_bad, 191
tcp.dstport, 191
tcp.flags, 185
tcp.flags.ack, 186, 191, 192
tcp.flags.fin, 191
tcp.flags.syn, 191, 192
tcp.hdr_len, 191
tcp.len, 191
tcp.seq, 191
tcp.srcport, 191
tcp.window_size, 192
TCP/IP, 64, 85, 100
tcp_wrappers, 251
tcpd, 133–135
telephone network, 118
telnet, 118, 132, 254
text/*, 275
text/html, 268
text/plain, 268
TFTP, 160
time, 215
time to live, 138, 159
timestamp-reply, 207
timestamp-request, 207
tkman, 30
top level domain, 94
tr, 132

traceroute, 176
transmission control protocol, 111
troff, 31, 32
true, 245
TTL, 138, 139, 159, 175, 176, 215
ttl, 215
ttl-exceeded, 207
TX packets:, 85
type *típus*;, 137
type=*típus*, 126

U, 89
UDP, 64, 109–111, 113, 114, 117, 123, 124, 127, 174, 176–178, 191, 208, 209, 213, 215
udp, 123, 127, 205, 208, 215
udp.checksum_bad, 191
udp.dstport, 191
udp.length, 191
udp.srcport, 191
UID, 240, 243
un-DNAT, 229
uname, 98
unfiltered, 121
UNIX to UNIX copy, 17, 257
UNKNOWN, 136
UNLIMITED, 128
UNLISTED, 126
unshielded twisted pair, 55
UP, 85
up, 82
URG, 209
USB, 49
usb-controller, 49
usb-controller1, 49
usb-controller2, 49
user, 17, 20
user datagram protocol, 109
user *felhasználó1 felhasználó2*, 289
User *felhasználónév*, 267
user=*felhasználó*, 127
userdir, 272
usertrack, 272
UTC, 160

UTP, 55, 56, 58, 59
UUCP, 257
uucp, 17, 257

valid-user, 289
vermagic:, 45
VERSIONID('szöveg'), 257
VersionSort, 275
very secure FTP daemon, 250
vhost_alias, 272
virtual user, 263
virtusertable, 264, 265
virtusertable.db, 265
vsftpd, 250–253
vsftpd.conf, 250

wait, 124
warning, 17
wd, 80, 81
Win, 182, 192
window, 182
WINS, 160

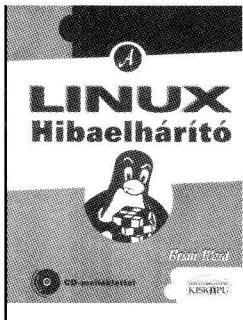
X display manager control protocol, 119
XDMCP, 119
xdmcp, 119
Xerox Palo Alto Research Center, 52
xferlog_enable, 252
xfs, 119
xinetd, 122, 125–128, 130–133
xman, 30

yellow pages, 242
YES, 250
yes, 247
ypbind, 248, 249
ypcat, 249
yppasswd, 249
ypserv, 248

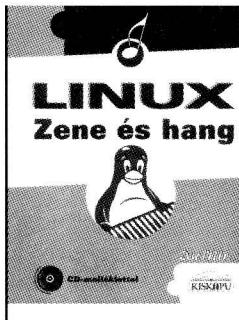
zone, 146

LINUX világába

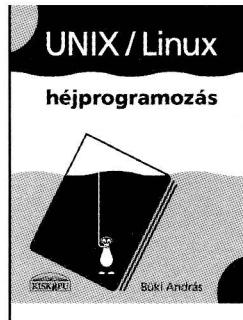
Kapu a



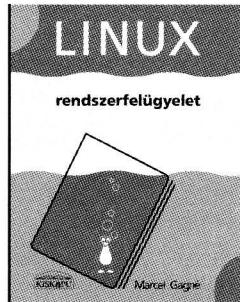
Ár: 3220 Ft
281 oldal
felhasználói szint:
kezdő, haladó
melléklet: CD



Ár: 4900 Ft
397 oldal
felhasználói szint:
kezdő, haladó
melléklet: CD



Ár: 2660 Ft
256 oldal
felhasználói szint:
kezdő-haladó



Ár: 6440 Ft
672 oldal
felhasználói szint:
kezdő–profi



Ár: 2660 Ft
256 oldal
felhasználói szint:
kezdő

www.kiskapu.hu

