Széchenyi István Egyetem

Műszaki Tudományi Kar
Informatikai és Villamosmérnöki Intézet
Távközlési Tanszék

Villamosmérnöki szak
Távközlés-informatika szakirány

UNIX

Hálózati operációs rendszerek I. jegyzet

Írta és szerkesztette: Molnár Zoltán Vilmos

2003.

1. Bevezetés

Ez a jegyzet a Széchenyi István Egyetem Távközlési Tanszék – Távközlés informatika szakirányon tartott Hálózati operációs rendszerek I. című tantárgyhoz (ta65vi) készült. Felhívnám a figyelmet, hogy a jegyzet magában nem elegendő, a tantárgy teljesítéséhez ajánlott a gyakorlatokon és az előadáson való részvétel!

1.1 UNIX története

A UNIX első változatát 1969-ben készítette el Ken Thomson és Dennis Ritchie az AT&T Bell Laboratóriumában egy PDP-7 típusú számítógépre. 1973-ban a UNIX rendszermagját átírták C nyelvre, és ingyenesen hozzáférhetővé tették az egyetemek számára. A 80-as évek elején már százezernél is több számítógépen futott UNIX. A gondot az jelentette, hogy az egységesség ellenőrzése hiányában mindenhol átszerkesztették így sok változat alakult ki. Ezekből két jelentősebb a Berkley egyetemen fejlesztett BSD UNIX, a másik AT&T hivatalos változata a System V (System Five, Relase 4-nél tart SVR4) amit az USL (Unix Syste Laboratories = USL) fejleszt tovább. A két szabványt próbálták valamelyest egyesíteni, így született meg az IEEE az ANSI és az ISO együttműködésével a POSIX (Protable Operating System Interface (x)) ajánlás.
Fontos megkülönböztetni a UNIX és a „Unix” használatát, míg a UNIX az USL licensszel rendelkező USL forráskódból származó rendszereket jelöli, a „Unix” az összes „Unix típusú” rendszert.

1.2 UNIX és a LINUX kapcsolata, a Linux története

A Unix alá kiadott felhasználói programok terjedésével amiket forráskódban (C nyelvben megírva) adtak ki bárki fordítgathatta és átírhatta kedve szerint, ezért Richard Stallman létrehozta az FSF (Free Software Foundation) alapítványt melynek célja egy szabadon (forráskódban is) ingyen hozzáférhető szoftverkörnyezet biztosítása bárki számára. Az FSF része a GNU project (GNU is Not UNIX), amely pedig minél teljesebb UNIX rendszert kíván létrehozni és biztosítani. Ennek a jogi

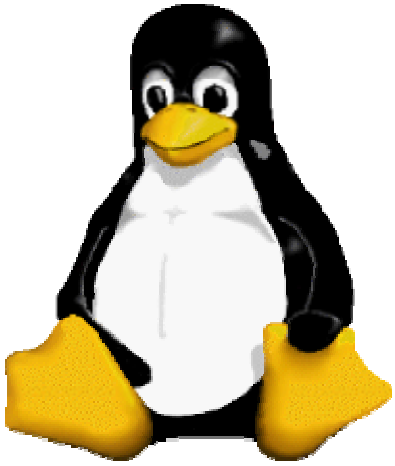
megfogalmazása a GPL (GNU Public License). A GNU környezet segítségével megnyílt az ajtó a Unix típusú rendszer IBM PC –re való adoptálására. Linus Torvalds egyedül nekiállt a PC alapú Unix-os rendszermag megírásának. Először assembly nyelven (0.01 verzió, 1991. augusztus vége) írta. Ez egy nagyon kezdetleges rendszer volt, igazából a Minix alatt lehetett fordítani és még lemezmeghajtót sem tudott kezelni. Linus ezután C nyelvben kezdte el fejleszteni a rendszerét, ami meggyorsította a fejlesztést. 1991 okt. 5 hirdette meg Linus az első „hivatalos" 0.02 verziót. Ezen már futott GCC (GNU C Compiler) és a BASH. A terjesztés célja nem az volt, hogy felhasználókat szerezzen, hanem a hogy segítséget szerezzen a rendszermag (KERNEL) fejlesztésében.

A 0.11 verzió megjelenése új korszakot nyitott a Linux történetében.

A 0.11 verzió újdonságai:

- demand loading
- kód és adatmegosztás nem kapcsolódó processzek közt
- sokkal jobb floppy-vezérlők (most már többnyire működnek)
- hibajavítások
- Hercules/MDA/CGA/EGA/VGA támogatás
- a konzol hangot is ad (Óh! Fantasztikus rendszermag!)
- mkfs/fsck/fdisk (fájlrendszer karbantartó programok)
- amerikai/német/francia/finn billentyűzet
- a com1/2 sebessége beállítható

A fejlesztés a POSIX -nek való megfelelés felé terelte a fejlesztést. Ekkor már próbáltak elszakadni a MINIX -től. A szétválás eléggé szégyenletesen zajlott le Linus és a Minix atyja Andrew Tannenbaum között egy internetes hírcsoportban.



1. ábra a Linux 1.0.0 hivatalos emblémája

A Linux 1.0.0 1994 márciusában jelent meg. Ez már teljesen a POSIX szabványnak megfelelt. A verziót ezen túl három ponttal elválasztott számmal jelölték. Az első úgynevezett fő verziószámot jelöli. Akkor változtatják nagyobbra, ha valami a rendszermagot érintő alapvető változtatás történik. Ilyen volt a 2.0.0 megjelenésénél a betölthető rendszermodulok megjelenése. A második szám az egyes fejlődési szakaszokat jelöli, és itt fontos megemlíteni, hogy ha ez a szám páros (pl.: 2.4.20), akkor az egy stabil, fejlesztők által garantált működésű rendszermag, ha viszont páratlan (pl.: 2.5.69) akkor ez még csak teszt változat, nem stabil. A harmadik szám a kisebb változtatásokkal növekszik. A legfrissebb verzió mindig megtalálható a http://www.kernel.org Internet címen.

1.3 Linux disztribúciók

A Linux így önmaga még csak egy működő rendszermag, amivel igazából semmit sem tudnánk kezdeni, így szükség van kezelő és felhasználói szoftverekre, hogy teljes rendszert alkossunk. A különböző Linux disztribúciók a meglévő rendszermag köré építették fel saját rendszereiket, tartalmazzák a felhasználói programokat, és könnyedén tudjuk ezeket gépünkre telepíteni. Mi a tanulmányaink során a Debian Linuxszal (3.0 verzióval) fogunk foglalkozni. A Debian Linuxról bővebbet a http://www.debian.org címen olvashatunk. Magyarországi tükörszervert, pedig az ftp://ftp.hu.debian.org címen találhatunk.

2. Alapvető parancsok a UNIX-ban

Mielőtt nekiállunk részletesen elemezni a Unix rendszereket, szükséges néhány alap parancs ismerete. A Unix-ban minden felhasználó egy neki megszokott környezetben dolgozik. Ezeket SHELL-ekek hívjuk. Alap estben lehet a BASH. A BASH kezelése egyszerű, és nagyon felhasználó barát. A kiadott parancsokat visszahívhatjuk a fel és le gombot nyomva. A parancsok fájlnevek és könyvtárnevek megadásánál nem szükséges a teljes nevet kiírni, hanem a TAB gomb lenyomásával kiegészíti azt. Alapvető parancsok:

ls = list = fájlok és könyvtárak listázása. Szintakszis: *ls [kapcsolók] <milyen könyvtár>*. Legtöbbször használt kapcsolók: *-l* long tehát hosszú listázás: fájlok és könyvtárak jogai, tulajdonos és csoport kiírás. *–a* attribútum azaz a **.**-al kezdődő rejtett fájlok megjelenítése. A kapcsolókat lehet egymás után megadni pl.: *ls –la /home*

cd azaz change directory = könyvtár váltás. Ugyan úgy mint a dos-ban a . az aktuális könyvtárat jelöli a .. egyel visszalép. Pl.: *cd /home/lencse* teljes elérést megadva, vagy lépésenként:

lencse@tilb:/# cd home

lencse@tilb:/home# cd lencse

lencse@tilb:~/#

cp azaz copy = másolás. *cp <mit> <hova>* pl.:

root@tilb:# cp /home/lencse/kiskutya.gif /home/drmomo/bloki.gif

mv azaz move = mozgatás: *mv <mit> <hova>*, pl.:

root@tilb:# mv /home/lencse/kismacska.jpg /home/drmomo/cica.jpeg

rm azaz remove = törlés: *rm <mit>***.** Leggyakrabban használt kapcsolók: *-r* rekurzív törlés, *-f* force mindenképpen töröljön. **Felhívnám a figyelmet arra, hogy a labor gépén root-ként kiadott *rm –rf /** parancsért fenékberúgás jár (tavalyi példából kiindulva a *rm –rf /home /** is törli a gyökér könyvtárat, mert felsorolás jelleggel a home könyvtár törlése után a / -t is letörli) !!!**

rmdir azaz remove direcrtory = könyvtár törlése. Csak üres könyvtárat lehet letörölni.

root@tilb:/# rmdir /home/buksi

pwd aktuális könyvtár kiíratása, ahol éppen tartózkodunk

cat szöveges állományok megjelenítése az alapértelmezett kimenetre. Pl.:

root@tilb:/# cat /home/lencse/kiskutya.txt

df megadja a helyfoglaltságot azoknak a köteteknek (partícióknak) amik a rendszerbe fel vannak mountolva.

du a fájlok és könyvtárak helyfoglalását adja meg. Kapcsolók: *-a*, *–k*: kilobyte, *-m*: megabyte

ps processz lista. Kapcsolók: *a* minden processz, *u* processz-t futtató felhasználó neve, *x*: minden egyéb (nem felhasználók által elíndított rendszerszintű pl.: syslogd) processz kiíratása. A kapcsolók megadásánál nincs szükség a '–' megadására. Pl.:

root@tilb:/# ps aux

touch fájllétrehozás (0 byte méretű). Pl.: touch /home/lencse/macska.txt

mkdir könyvtárlétrehozás Pl.: *mkdir /home/lencse/hallgatok*

man azaz manual = minden parancshoz és felinstallált futtatható programhoz segítséget, leírást kapunk. Ezt a **man** nevű paranccsal lehet megtekinteni.

Házi feladat olvasmány: man bash ☺

more kiírja a megadott szöveges állományt az alapértelmezett kimenetre oldalanként tördelve. Pl.: more hosszuszoveg.txt

less mint a more, csak lehet soronként és oldalanként fel, -le lépkedni.

head szöveges állomány kiíratása (megadott tíz sor).

tail ugyanúgy szöveges állomány kiíratása (megadott utolsó 10 sor), viszont *–f* kapcsolóval lehet követni a szöveges állomány változását. Kiválóan alkalmas a log fájlok követésére.

Pl.: *tail –f /var/log/syslog*

tar fájlok és könyvtárak fájlba csomagolása (nem tömörítés!) Szintaxis: *tar [-opciók] [mit] [hova]*. Kapcsolók: *a* = add: becsomagol, *x* = extract: kicsomagol, *v* = verbose: képernyőn megjelenít, *f* = fájl, *z* = ha gzip-el (ki/be)tömöríteni akarjuk, *y,I* = ha bzip2-vel (ki/be)tömöríteni akarjuk.

ln link létrehozása. Későbbiekben el lesz magyarázva a link fogalma. Alap esetben ún.

su felhasználók közötti váltás (*su <felhasználónév>*, su - root-ra váltás)

hard linket hoz létre, *-s* kapcsolóval pedig szimbolikus linket lehet létrehozni.
Szintaktika: *ln [kapcsolók] <fájl> <link neve>*
Ráérő időnkben (pl.: gyakorlaton) próbálgassuk az alap parancsokat, és szokjuk meg ezek használatát.
Fontos jó tanács: Soha ne használjunk ékezetes karaktereket fájlok és könyvtárak nevénél, mert csak magunkkal vagy másokkal szúrhatunk ki!

2.1 Az *apt*, a Debian Linux csomagkezelője

Az *apt* rendszerű csomagkezelés a Debian Linux jellegzetessége. A rendszerünkön nyílván van tartva, hogy milyen csomagok vannak felinstallálva. Ezeket bármikor kedvünk szerint módosíthatjuk. Erre való az *apt-get* parancs. Szintakszis: *apt-get <opció> <csomagnév>*
Ennek sok-sok opciója van, ezek közül csak néhány legfontosabbat nézzünk meg.

- *update*: csomaglista frissítése. A rendszerünk installálásakor az install script megkérdezi, hogy milyen médiumokról telepítjük rendszerünket. Ezeket mésnéven apt source-oknak (apt forrásoknak) nevezzük. Az apt források lehetnek az installáló 7 cd, vagy lehet közvetlenül HTTP-ről, FTP-ről telepíteni. Ha már a meglévő rendszerünkön szeretnénk apt forrást változtatni, vagy újjal bővíteni, akkor a /etc/apt/apt_source fájlt kell módosítanunk.
- *install*: csomag felinstallálása a rendszerünkre
- *remove*: csomag eltávolítása a rendszerünkről

Egy másik nagyon fontos program az apt-cache. Szintaktikája: apt-cache <opció> <csomagnév>. Opciók:

- *search*: csomag keresés valamilyen jellegzetesség (pl.: név vagy funkció alapján)
- *showpkg*: csomag információ (függőségek, verziószám stb…)

Az *aptitude* programmal könnyedén tudunk böngészni a már installált és a még nem installált programok között. Ez alapesetben nem kerül installálásra, így nekünk kell azt telepíteni: *apt-get install aptitude*.

2.2 A *VI* kezelése

A *VI* egy alapvető szövegszerkesztő program UNIX rendszerekhez. A kezelése a következő funkció gombokkal lehetséges:

↑ k

h ← → l

↓ j

: gombbal a következő funkciók érhetők el: a = beírás, i = beszúrás, o = új sor, x betű törlés, q = elvet, kilép, w = írás (lemezre), del = sortörlés

Példa: kilépés, mentéssel = :wq!

3. A UNIX felépítése

Ebben a fejezetben megismerkedünk a UNIX felépítésével mind elméleti mind gyakorlati szempontból. Megismerjük a többfelhasználós és többfeladatos rendszerek működési elvét. A gyakorlat alkalmazhatóság érdekében a UNIX alapú rendszerek működési elvét a Linux-on keresztül fogjuk tanulmányozni.

3.1 Többfelhasználós rendszer lényege

A számítógépek szolgáltatásainak – felhasználási területtől függően – elérhetőnek kell lennie több felhasználó számára is. Ez az igény egyszerűen úgy jelentkezhet, hogy a felhasználók nem egyidejűleg, hanem felváltva használják a számítógépet. Ilyenkor már szükséges olyan szolgáltatásokat bevezetni, amely lehetővé teszi, hogy minden felhasználó önálló futtatási környezetben dolgozhasson, például mindig saját állományait futtathassa, másokét ne. Emiatt védelmi elemeket kell az operációs rendszerbe építeni: egy felhasználó csak a számára engedélyezett erőforrásokhoz és adatokhoz férhessen hozzá, a rendszer kritikus paraméterei pedig csak a rendszergazda, illetve az általa meghatározott csoportok számára legyenek hozzáférhetők. Fontos a felhasználók megkülönböztetése a skálázási szempontok miatt is. Egy „szőke viháncoló titkárnő” tudásszinttel rendelkező felhasználó elé nem fogunk fájlrendszer karbantartását végző programokat kitenni.

A többfeladatosság igénye például akkor merül fel, amikor egy számítógépet szerverként üzemeltetünk, azaz szolgáltatásokat nyújtanak egyszerre több felhasználó számára (például WEB, FTP, E-MAIL). Ez annyit jelent, hogy a számítógép erőforrásait egy időben több program között kell felosztani, és az erőforrások allokálását is ellenőrzötten kell végezni. Ezt csak az ún. multitaszkos, azaz többfeladatos operációs rendszerek tudják teljesíteni. Ilyennek tekinthető pl. az összes UNIX klón, OS/2 stb. Tipikusan nem többfeladatos operációs rendszer pl. a DOS.

A többszálon futó végrehajtás legnagyobb veszélye, hogy egy hibás működésű program miatt leállhat egy vagy az összes többi program is, amely a számítógépen fut. Ezen nem kívánt „baleset” megelőzése érdekében az operációs rendszer beépített védelemmel rendelkezik (amit a Microsoft termékek nem mondhatnak el magukról☺).

Minden futó program külön memóriaszeletet használ, és a perifériákhoz történő hozzáférés is csak a rendszer hívásain keresztül lehetséges. Az operációs rendszer feladata az is, hogy figyelje azt, hogy egy program nem kezdeményez-e hibás, vagy illetéktelen hozzáférést a rendszer valamely részéhez. Ilyen esetben az operációs rendszer közbeavatkozik, és a hibás működésű program futtatását megszünteti.

Tehát a többfeladatos és többfelhasználós redszerek tulajdonságai összefoglalva:

- Egyszerre több program futhat
- A felhasználók egyedi azonosítóval rendelkeznek (authentikáció)
- Az erőforrás elosztás szabályzott
- A rendszer működése védett mind a felhasználókkal mind a hibás működésű programokkal szemben
- A rendszer ellenőrzött hozzáférést biztosít az egyes hardver elemekhez és egyéb erőforrásokhoz

3.2 Fájlrendszer típusok

Egy UNIX-os rendszer installálásakor a legelső lépés a fájlrendszer létrehozása. A legelterjetebb fájlrendszer a Linuxnál az EXT2. Ezzel lehet a legnagyobb sebességet és megbízhatóságot elérni. Az újabb rendszermagok elterjedésével lehetőségünk van másfajta fájlrendszerre installálni az alap rendszerünket. Az EXT3, ReiserFS, JFS annyiban különböznek az EXT2-től, hogy azok naplózott fájlrendszerek. Minden bejegyzés a partíción naplózva van, így rendszer leállás esetén (nem lefagyásra gondoltam, hanem áramszünetre) nem kell a partíciónkat hosszasan végigellenőrizni. A Linux képes DOS FAT és FAT32 illetve NTFS/HPFS partíciókat is kezelni. Az utóbbiaknál a partícióra való írás még bizonytalan. A hálózati meghajtókat is fájlrendszerként kezeli a UNIX. Ilyen pédául az NFS és az SMBFS is.

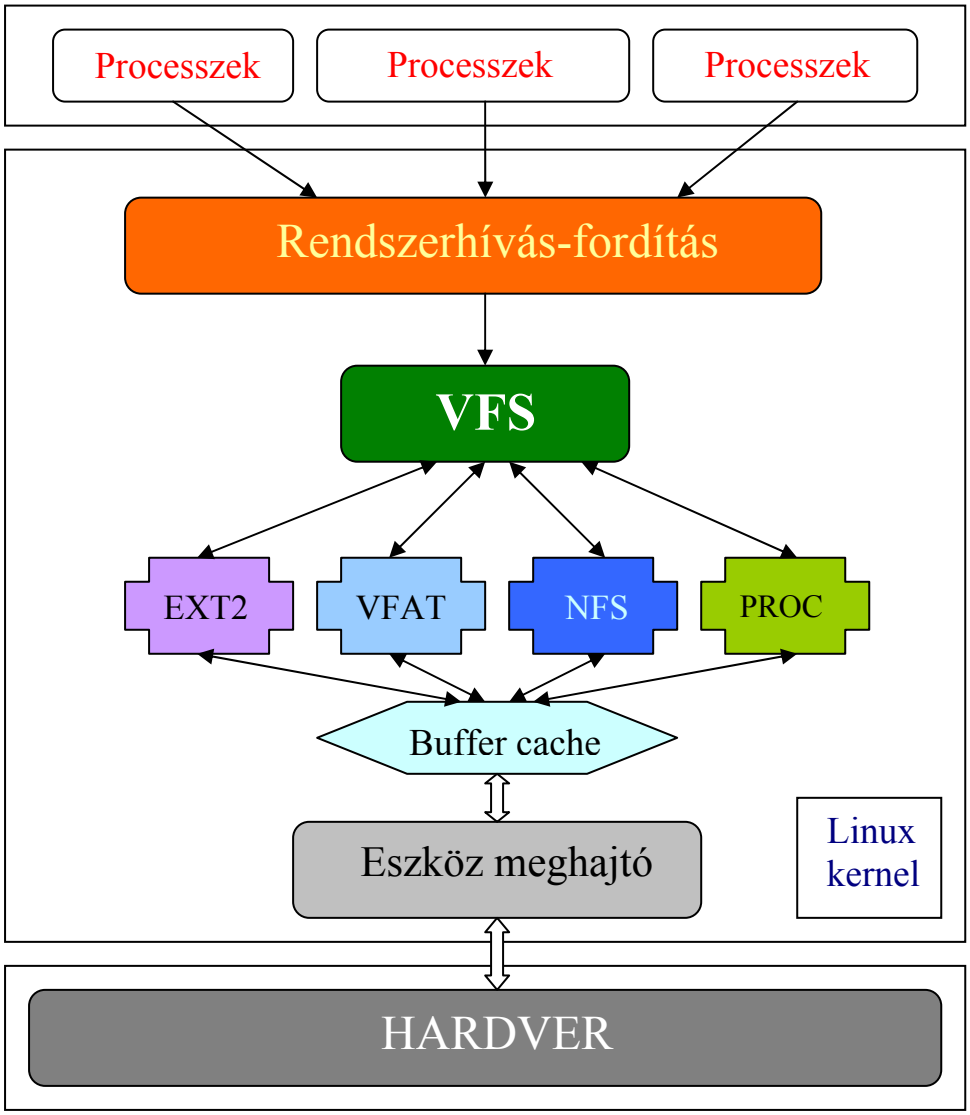
3.3 A VFS, virtuális fájlrendszer

A Linux a különleges fájlrendszerek egységes kezelhetőségének érdekében tartalmaz egy ún. virtuális fájlrendszert, ami egy szoftver réteg. Ez a szoftver réteg elvonatkoztat az egyes fájlrendszerek típusától és egy egységes kezelési felületet biztosít a felhasználói alkalmazások számára. A 2. ábra szemlélteti a VFS elhelyezkedését a Linux rendszerstruktúrában.

A VFS biztosítja, hogy a felhasználói alkalmazások minden kezelt fájlrendszert azonos módon lássanak. Ezt a gyakorlatban úgy tapasztaljuk meg, hogy az összes hardvereszközön lévő fájlrendszer tartalmát a root directory (gyökér könyvtár) alatt lévő külön alkönyvtárakban érhetjük el. Ezen könyvtárak helyét a „*mount*” folyamattal tudjuk kijelölni. A VFS számon tartja a *mount*-olt fájlrendszerek helyét és állapotát. Így elérhetjük, hogy egy másik számítógép megosztását valamilyen hálózati protokoll segítségével (pl.: NFS, SAMBA) saját gépünk egy alkönyvtárában lássuk.

Mount-olni a *mount* parancsal lehet. Szintaktikája: mount [kapcsolók] <mit> <hova>. Kapcsoló **–t** partíciófajta. Ha a */etc/fstab* –ba bejegyeztünk meghajtókat, illetve partíciókat egyszerűen csak a kijelölt alkönyvtárt kell megadni. Pl.: cd-rom meghajtó felmountolása, ha bejegyeztük az *fstab*-ba: mount /cdrom. A */etc/fstab* azon meghajtókat tartalmazza, amiket a rendszer induláskor automatikusan felmountol, ha tudja. A */etc/mtab* az éppen felmountolt meghajtókat jelzi.

A proc fájlrendszer. A /proc könyvtár a rendszer elemeket tartalmazza. Ezek szöveges fájlonként jelennek meg a rendszerben, és bármikor olvashatjuk őket (pl.: *cat /proc/cpuinfo* információkat ad a processzorunkról). Érdemes egyszer végig böngészni a könyvtárat, hogy mit találhatunk még benne…



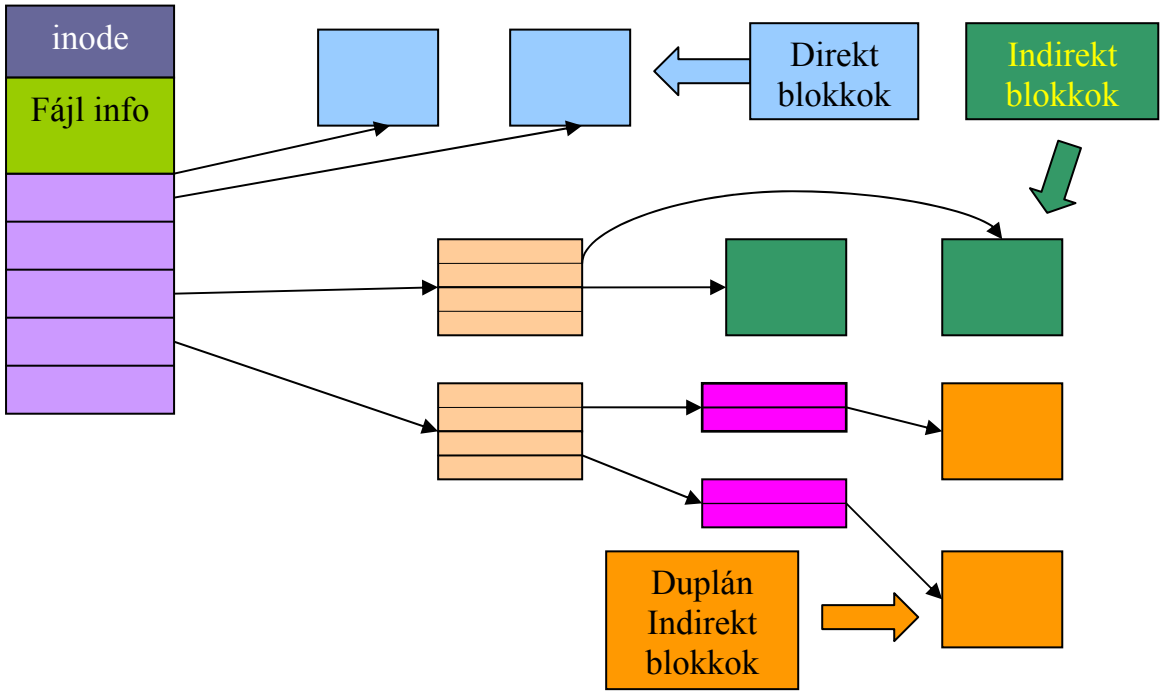
2. ábra: Virtuális fájlrendszer elhelyezkedése a Linux rendszerben

3.4 Fájlrendszer, *inode*-ok, linkek

A Linux és az összes UNIX alapú rendszer legalapvetőbb védelme a fájlrendszer. Ez tárolja az összes fájl és alkönyvtár elérhetőségi szabályát, felhasználó-csoportosításokat, kezeli a linkeket. A fájlrendszer gondoskodik az adathalmazok tárolásáról és a szabad lemezterület menedzseléséről. A fájlrendszerrel rokon modul még a „*Quota subsystem*” (kvóta alrendszer) mely a felhasználók tulajdonában lévő fájlok maximális helyfoglalását korlátozza.

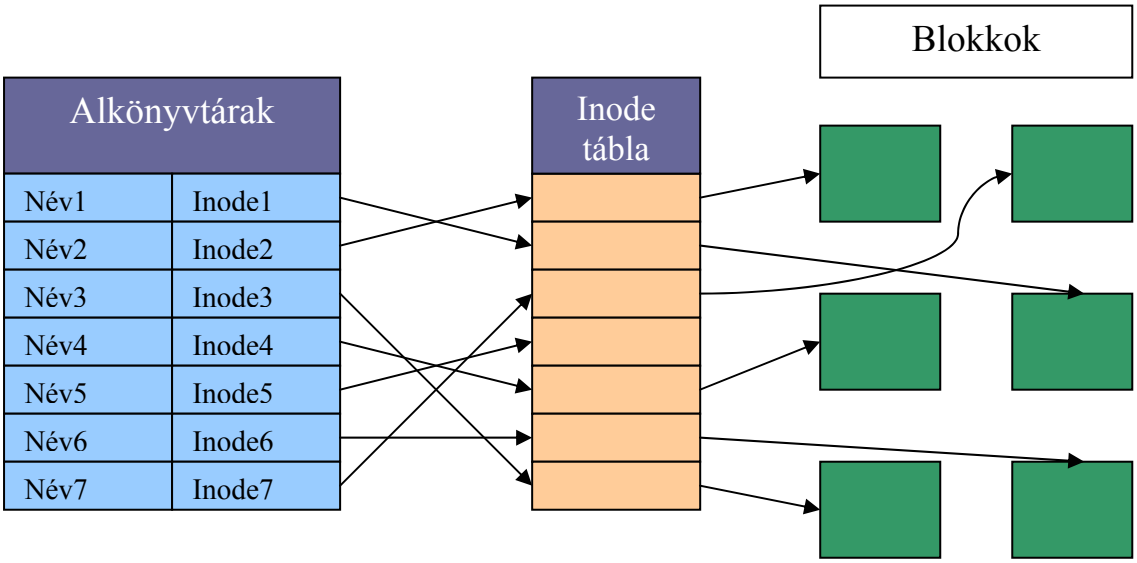
Inode-ok. Minden fájlt egy *inode*-dal írhatunk le a fájlrendszerben, ez információkat tartalmaz a fájlról: típus, jogok, tulajdonságok, időbélyeg, méret, és az adatblokkok helye (mutatója, azaz pointere). Az adatblokkok ún. foglalási egységek, mérete

1Kbyte többszöröse lehet (1024, 2048, 4096 byte a szokásos). Ha például egy 10Kbyte-os JPEG fájt helyezünk el egy 4096 byte-os minimális blokkméretű partíción, akkor az valójában 12Kbyte-ot fog elfoglalni, mert legalább 3 darab egyenként 4 Kbyte-os blokkot fog lefoglalni. Ennek a 3 darab blokknak a fizikai címét (a lemezen hányadik sorban, illetve szektorban helyezkedik el az adatblokk) írjuk bele az *inode* megfelelő adatmezejébe. Ha az *inode*-ban lévő mutató egy másik *inode*-ra mutat, akkor indirekt blokkról beszélünk.



3. ábra: inode-ok

Alkönyvtárak. Az alkönyvtárak tulajdonképpen speciális fájlok, ahol a fájl tartalma egy lista, amely azoknak a fájloknak az *inode*-jára mutat, amelyek az alkönyvtáron belül vannak. Az alkönyvtárakat is ugyanolyan *inode*-ok írják le, mint a fájlokat, csak a fájltípus mezőben alkönyvtár-jelzés van („**d**” bejegyzés). Természetesen egy alkönyvtár bejegyzése mutathat másik alkönyvtár bejegyzésre is. Az alkönyvtár bejegyzéseket a 4. ábra szemlélteti.



4. ábra: Alkönyvtárak bejegyzései

Linkek. A UNIX fájlrendszernél bevezették a link fogalmát, amely azt jelenti, hogy egy fájlt vagy alkönyvtárat leíró *inode*-ra több alkönyvtárbejegyzés mutathat. Az egyik típus az ún. „*hard link*” úgy jön létre, ha egy alkönyvtárban készítünk egy új bejegyzést, amely már létező fájlra (*inode*-ra) mutat. Minden *inode* tartalmaz egy számlálót, amely mutatja, hány helyről hivatkozunk rá. Ha létrehozunk egy *hard link*-et, akkor ez a számláló érték nő, ha törlünk egy *hard link*-et csökken. Amikor az utolsó hivatkozást is eltávolítottuk (pl.: *rm* paranccsal) akkor az *inode*-ot az operációs rendszer törli a táblázatból, akkor a fájl megszűnik létezni (tehát ha letörlünk egy fájlt, nem az általa foglalt helyet nullázzuk ki, hanem az arra hivatkozó *inode*-ot töröljük a táblázatból). A *hard link*–ek legnagyobb hátránya, hogy csak egy partíción belül használhatók. Pl. nem tehetünk *hard link*-et két külön partíció közé. Ezen felül nem engedi az operációs rendszer a könyvtárra mutató *hard link* készítését sem, nehogy véletlenül rekurzió alakuljon ki az alkönyvtárak között. A linkek másik fajtája az úgynevezett szimbolikus link (*symbolic link*) ami tulajdonképpen egy fájl, amely egy fájl nevet és elérési útvonalát tartalmaz (létrehozása: ln –s <fájlnév> <link neve>). Ha egy szimbolikus linket szeretnénk elérni, az operációs rendszer kicseréli arra az elérési útra amelyre a link mutat, és ehhez a névhez tartozó *inode*-ot keresi meg.

Eszközfájlok. A UNIX rendszerekben az általunk támogatott hardver eszközöket speciális, ún. eszközfájlokon keresztül lehet elérni. Ha egy ilyen fájl elérését

kezdeményezzük, a **kernel** automatikusan meghívja az adott hardver elemet kezelő rutint, és szabványos I/O műveleten keresztül közvetíti az eredményt, mintha azt az eszközfájlból olvasnánk ki. Léteznek karakteres és blokk elérésű eszközök. A karakteres **getchar()** és **putchar()** C függvényeken keresztül karakterként írhatóak, azaz egy műveletre egyetlen egy bájtot adnak vissza. Ilyen eszköz például az egér (PS/2 → /dev/psaux). A blokk eszközök a **read()** és **write()** C függvényen keresztül egyszerre több száz bájt írásával, olvasásával érhetők el. Ilyen eszköz például az összes lemezmeghajtó (IDE0 → /dev/hda0, SCSI0 → /dev/sda0). Az eszközfájlok két azonosítószámmal rendelkeznek. Ezek a „*major number*” és a „*minor number*”. Ez a két szám egyértelműen azonosít egy eszközt. A legtöbb UNIX rendszer nem hozza létre ezeket az eszközfájlokat, a felhasználó, vagy az operációs rendszer telepítője készíti el ezeket az *mknod* paranccsal.

3.5 Hogy néz ki egy UNIX rendszer

Mint minden rendszer esetében megszokhattuk itt is fájlokkal, és könyvtárakkal kell dolgoznunk. Viszont a már ismertebb operációs rendszerekkel szemben (nem szándékozom megemlíteni a nevét) itt a meghajtókra nem betűjelekkel hivatkozunk. Az egész rendszerünk egy főkönyvtárból nyílik (főkönyvtár hivatkozása: /). Ezt más néven **root**–nak (magyarul: gyökérnek) hívjuk. Ezekben találhatók az alábbi könyvtárak.

/	=	Főkönyvtár
/bin	=	Futtatható bináris állomány
/boot	=	BOOT partíciót szokták ide csatlakoztatni
/dev	=	DEVICES azaz meghajtók
/etc	=	A rendszer konfigurációs fájljai
/home	=	felhasználók könyvtárja
/lib	=	Library azaz könyvtár-programok
/proc	=	Processz-, és rendszer-információk
/root	=	Rendszergazda home-ja
/tmp	=	Ideiglenes könyvtár, újraindulásnál törlődik
/usr	=	Alkalmazási és egyéb programok

/sbin = ROOT-ként futtatható bináris állomány

/var = Egyéb programok által használt könyvtár

3.6 Fájlok és jogaik a UNIX-ban

Mivel a UNIX rendszer több felhasználós rendszer az esetleges felhasználóink adatait egymástól védeni kell. A UNIX-ban a védelem háromszintű. Első szint a felhasználó jogai, ugyanis a fájlok és könyvtárak hozzá vannak rendelve a felhasználóinkhoz (ezek a USER-ek). Továbbá hozzá van rendelve, hogy milyen csoportba tartozik (ezek a GROUP-ok), ez a második szint. A harmadik szint azt szabályozza, hogy hogyan férhet hozzá a „nagyvilág” (tehát az a felhasználó, aki nem a tulajdonosa, illetve nincsen a csoportban).

Egy fájlhoz és könyvtárhoz 10 bit információt jegyez be a fájlrendszer. Ezek a következők: Első bit a bejegyzés típusát jeleli. A bejegyzés **d** ha könyvtár, **b** ha blokk típusú speciális fájl, **c** ha karakter típusú speciális fájl, **l** ha link, **s** ha a fájl socket és – (üres bejegyzés) ha egyszerű fájl. A fájlok neve után nem kötelező, de célszerű megadni a kiterjesztését, hogy tudjuk milyen fajta fájlról van szó (Pl.: alma.jpg – tudjuk, hogy kép fájl). A következő három bit a tulajdonos jogit határozza meg. Ez alapesetben **rwx** azaz (binárisan 111) = (decimálisan) 7. Az első bit az **r** azaz olvasási jogot jelent. A második bit a **w** az írás jog. A harmadik a **x** azt jelöli, hogy van-e joga futtatni a fájlt (csak bináris illetve script fájlokat érdemes futtathatóvá tenni). Könyvtárak esetében az **r** és **w** értelem szerint a könyvtárra vonatkozó olvasási és írási jogokat jelenti, az **x** viszont arra ad engedélyt, hogy a könyvtárban tallózunk. A következő három bit a csoportok jogát határozza meg hasonló képen mint a user esetében. Az utolsó három bit pedig a világ számára határozza meg a jogokat.

Egy példa: a diak nevű könyvtár jogai a következőek: **d rwx r-x ---** ennek bináris értéke a kövekező (csak a jogokat tekintve) 111 101 000, ez decimálisan 7 5 0. Változtassuk meg, hogy a világ számára is böngészhető legyen, de csak olvasni lehessen. A jogok megváltoztatására a *chmod* parancsot használjuk.

root@tilb:/home# chmod 755 diak

Lehet másféle képen is megadni:

root@tilb:/home# chmod +r+x diak

Tulajdonos megadása: *chown tulajdonos bejegyzés*

root@tilb:/# chown diak /home/diak

Csoport megadása: *chgrp csoport bejegyzés*

root@tilb:/# chgrp users /home/diak

Másféle képpen:

root@tilb:/# chown :users /home/diak

Lehet egyszerűbben megadni egy parancsból a felhasználót és csoportot

root@tilb:/# chown diak:users /home/diak

4 Felhasználók kezelése a UNIX-ban

Nézzük meg root-ként a */etc/passwd* fájlt. Ez a fájl tartalmazza a UNIX rendszerekben a felhasználók adatait. Régebben itt helyezkedett el a felhasználó jelszava is. Biztonsági okokból áttették a későbbi verziókban a jelszavakat a */etc/shadow* fájlba. Könnyen belátható miért. Nézzük meg a jogokat a */etc/passwd* fájlra (-rw-r--r--). Látható hogy mindenki által olvasható, mert nem minden processz fut root jogokkal, és néhánynak szüksége van a felhasználók azonosítására, és ezek ezt a fájlt használják, így a kódolt jelszó is megfejthetővé válik.
Mit tartalmaz a */etc/passwd* fájl?

root:x:0:0:Ez a rendszergazda account,,,:/root:/bin/bash
drmomo:x:1000:100:Molnár Zoltán,laboros,L1-7,tel.szám,:/home/drmomo:/bin/bash
…
Nézzük meg részletesen lépésről lépésre:

1. Felhasználói név (user account): root
2. Régen itt volt a jelszó, de ez már shadow fájban található: x
3. Felhasználói azonosító szám (UID= User ID): 0
4. Csoportazonosító szám (GID= Group ID): 0
5. Név, és további információk (ezek ”,”-vel elválasztva)
6. Home könyvtár: /root
7. Alapértelmezett shell: /bin/bash

Ha szeretnénk létrehozni felhasználót, akkor root-ként editálni kell (mindenki a szívéhez nőtt editáló programot használhatja: pl: *pico*, *nano*, *joe, mcedit*, vagy akár Lencse tanár úr kedvencét a vi-t) a */etc/passwd* fájlt. Értelemszerűen bejegyezni az új felhasználó adatait. Ezután szinkronizálni kell a passwd fájlt a shadow fájlal. Erre szolgáló parancs: *pwconv*
Ezután hozzuk létre a megadott home könyvtárat. Adjunk rá jogokat, és adjuk meg a felhasználói jelszót a *passwd* parancsal.

Erre szolgál (egyszerűbb módszer) a Debian linux-ban egy *adduser* script, ezt értelemszerűen kitöltve ugyanazt véghezvihetjük, mint a fapados editáló módszerrel ami, viszont minden UNIX rendszerben ugyanúgy működik (ha valakit bővebben érdekel: man adduser).

Egy példa a *buksi* felhasználó felvételére:

root@tilb:/# pico /etc/passwd ← editáljuk a passwd fájlt

Bejegyezzük:

buksi:x:1077:100:Kamu Bela,o csak fake user,,:/home/buksi:/bin/bash

CTRL+X, YES, ENTER

root@tilb:/# pwconv ← szinkronizálás

root@tilb:/# mkdir /home/buksi ← home létrehozás

root@tilb:/# chown buksi:users /home/buksi ← tulaj, csop. megadás

root@tilb:/# passwd buksi ← jelszó megadás

Changing local password for buksi.

New password: ← ide írjuk be a jelszót

Retype new password: ← megerősítjük a jelszót

Ha háromszor is kéri a jelszót az azért lehetséges, mert túl egyszerűt adtunk neki.

5 Felhasználói korlátozások

Valljuk be, hogy eddig semmi ördöngös nem volt eddigi tanulmányaink során. Ez a fejezet már kezd egy kisebb kihívássá válni a rendszergazda kezdemények számára. A UNIX rendszerekben lehet és kell is a felhasználóinknak a rendszerünk adta lehetőségeket korlátozni, nehogy visszaéljenek a jószívűségünkel. Ez lehet akár tároló hely, memória, vagy processzoridő.
Először nézzük meg, hogyan lehet a tároló helyet korlátozni, azaz mérnökien fogalmazva: kvótázni.

5.1 Quota

Mivel egy adott rendszerben a háttértárak kapacitása véges, korlátozni, illetve maximalizálni kell a felhasználók helyfoglalását. Erre a UNIX-okba beépített támogatás van, az ún. kvóta alrendszer (Quota subsystem). A kvóta alrendszer állandóan figyeli egy adott felhasználó helyfoglalását, és ha eléri az őrá kiszabott határt (hard limit) a rendszer „*write error*” hibaüzenettel megtagadja a további írást a lemezre. A kvóta beállításokat felhasználókra és csoportokra egyaránt vonatkoztathatjuk, illetve az összes külön felmountolt lemezegységre vagy partícióra megadhatjuk azokat.
Ahhoz, hogy a kvóta működni tudjon a Linux kernelbe be kell fordítani a quota támogatását (a kvóta működik EXT2, EXT3, REISERFS, esetén is, JFS esetében azonban még nem [2.4.21-es kernel]!).
Tegyük fel, hogy a rendszerünk */home* könyvtárban lévő felhasználói könyvtárak helyfoglalását szeretnénk korlátozni. A következő műveleteket kell elvégeznünk a kvóta rendszer üzembe állításához:
Ellenőrizzük, hogy a /home könyvtár egy külön partíción helyezkedik el:

```
root@pc0:/# mount
/dev/hda1 on /boot type ext2 (rw)
/dev/hda3 on / type ext2 (rw)
/dev/hda4 on /home type ext2 (rw)
none on /dev/pts type devpts (rw, gid=5, mode=620)
```

none on /proc type proc (rw)

Ezután a kvótázó fájlrendszer gyökerébe (jelen esetben a /home) létrehozzuk a következő két fájlt: *quota.user, quota.group*

root@pc0:/home:# touch quota.user quota.group

Adjunk rá jogot, hogy csak root olvashassa és módosíthassa:

root@pc0:/home# chmod 600 quota.user

root@pc0:/home# chmod 600 quota.group

Ezután a */etc/fstab* –ban megjelöljük, hogy a */home* partíció kvótázva lesz.

root@pc0:/# pico /etc/fstab

A megfelelő sort a következőképpen módosítsuk:

/dev/hda4 /home ext2 defaults,usrquota,grpquota 1 1

Miután módosítottuk az *fstab*-ot mountoljuk újra a */home* partíciót, majd aktiváljuk a kvóta rendszert.

root@pc0:/# mount –o remount –o rw /dev/hda4

root@pc0:/home# quotacheck –avug

Scanning /dev/hda4 [/home] done

Checked 4 directories and 72 files

Using quotafile /home/quota.user

Using quotafile /home/quota.group

root@pc0:/home# quotaon –avug

/dev/hda4: group quotas turned on

/dev/hda4: user quotas turned on

Ha most megnézzük az általunk létrehozott fájlokat, láthatjuk, hogy már nem 0 a méretük, a kvótázással kapcsolatos információkat tartalmazza.

Ezután ha a felhasználó beírja a quota parancsot, megnézheti a rendelkezésre álló tárhelyet:

diak@pc0:~> quota

Disk quotas for user diak (uid: 1000): none

Természetesen, még senkinek nem állítottunk be kvóta értékeket, ezért továbbra is korlátlan tárhellyel rendelkeznek felhasználóink. A korlátozást edquota paranccsal állíthatjuk be:

root@pc0:/home# edquota –u diak
Disk quotas for user diak (uid 1000):
Filesystem blocks soft hard inodes soft hard
/dev/hda4 263288 200000 250000 2129 15000 17000
~
~

A fenti mezőben a „*soft*” jelenti azt a határt, aminél több adatot a felhasználó nem tárolhat huzamosabb ideig a lemezen. Az edquota –t paranccsal beállíthatjuk az ún. „*grace period*” időtartamot, akkor ezen időtartam alatt még írhat a felhasználó a lemezre, de legfeljebb a *hard limit* erejéig. Ha a grace period lejár a soft limit is „*hard limitté*” válik.

5.2 Ulimit

Előfordulhat, hogy egy futó program erőforrás felhasználását szeretnénk korlátozni, pl.: nem szeretnénk, hogy egy program túlzottan sok processzoridőt vegyen el. Ezeket a paramétereket az általunk futtatott shell-re és az ebből futtatott programokra vonatkozóan az ulimit paranccsal állíthatjuk be.
Az *ulimit* a következő paramétereket fogadja el:

- -a: – megmutatja az aktuális határokat
- -c: – a „core” fájl maximális mérete
- -d: – a maximális adatszegmens mérete
- -f: – a maximális fájlméret
- -n: – nyitott fájlok maximális száma
- -s: – maximális veremméret
- -t: – maximális processzoridő másodpercben
- -u – az adott felhasználó által futtatott processzek maximális száma
- -v – a shell által használható maximális virtuális memória mérete

Bővebb információ: *man bash, /ulimit* alatt.

6 Linux KERNEL

A Linux kernel forrást C nyelven írták, szabadon terjeszthető, és bárki átírhatja kedve szerint, ha rendelkezik megfelelő C programozási érzékekkel. Ebben a fejezetben az alapvető ismereteket sajátítjuk el, hogy hogyan készítsük el saját igényeink szerint a rendszermagunkat. A jegyzet írásakor a 2.4.21–es kernel volt a legfrissebb, úgyhogy ezen megyünk végig részletesen.

6.1 Konfiguráció elkészítése

Mielőtt nekiállnánk a konfiguráció elkészítésnek, vizsgáljuk meg, hogy a fordításhoz fel vannak-e telepítve a megfelelő fordító, és könyvtár programok (library fájlok). Mivel majdnem minden rendszermag verzióhoz más-más fordító library-k szükségesek vizsgáljuk meg, hogy a fordítandó rendszermaghoz milyen csomagok kellenek (*apt-cache showpkg <kernelsource>*). A 2.4-es verziójú kernel-hez a libncurses5-dev és a GCC 2.96-os verziójú csomagok szükségeltetnek. Szerencsére a kernel forrás tartalmazza a make-fájlt, így nem kell nekünk egyenként a C fájlokat lefordítani. Szerezzük be a számunkra legmegfelelőbb kernelforrást. Ne terheljük az ország kimenő sávszélességét, használjuk a letöltésre a magyar tükörszervereket. Pl.:
ftp://ftp.kfki.hu/pub/linux/ftp.kernel.org/pub/linux/kernel/v2.4/linux-2.4.21.tar.gz
Miután letöltöttük csomagoljuk ki a /usr/src könyvtárba (*tar –xvzf linux-2.4.21.tar.gz*).
Kibontás után készítsünk egy szimbolikus linket a könyvtárra.
root@pc0:/usr/src# ln –s linux-2.4.21 linux
Lépjünk be a könyvtárba, majd írjuk be a következő parancsot:
root@pc0:/usr/src/linux# make menuconfig
Fontos, hogy a fordítást mindig root-ként végezzük, a jogok miatt!
Rövid várakozás után megjelenik a konfigurációkészítő menü rendszer. A menüben fel illetve le nyilakkal lépkedhetünk, az egyes almenükbe ENTER lenyomásával tudunk belépni. Visszafelé haladni az ESC billentyű megnyomásával tudunk. Az egyes meghajtó programok (program kódok) kiválasztásánál a SPACE gomb többszöri nyomogatásával lehet választani, hogy a rendszermagba, vagy modulként kívánjuk lefordítani azokat. Az ”m” lenyomásával modulként jelölhetjük ki őket. Érdemes

megjegyezni, hogy nem minden programkód fordítható modulba (pl.: a kernelmodul betöltése funkciót nem tudnánk modulként engedélyeztetni!).

Az kernelbe fordított meghajtó programkódok előnye, hogy mindig szerepel a rendszerben, így nem kell azt külön betöltögetni, így mindig rendelkezésre áll. Érdemes azokat a meghajtó programokat kernelbe fordítani, amik a rendszerünk számára nélkülözhetetlenek (pl.: IDE-ATA, SCSI vezérlő stb…).

Modulként viszont érdemes olyan eszközök meghajtó programjait fordítani, amiket nem mindig használunk, illetve használat után el szeretnénk távolítani a rendszerünkből. Tipikusan ilyenek az USB-s eszközök (UHCI, OHCI), de illik a hálózati csatoló kártyák meghajtóit is modulként lefordítani. A lefordított modulokat az *insmod* vagy a *modprobe* parancs segítségével tölthetjük be, illetve az *rmmod* parancsal távolíthatjuk el. Az *lsmod* parancsal, pedig a betöltött modulokat listázhatjuk ki. Szinte minden Linux disztribúció esetében a modulok a */lib/modules/--kernel verziószám--* könyvtárban helyezkedik el. Mivel a rendszerünk az éppen aktuális verziószámú könyvtárat automatikusan eléri, elég a modul betöltéséhez csak a modul nevét megadni (pl.: Realtek 8139 típusú hálózati interface modul betöltése: *modprobe 8139too*).

Az első menüpontban kiválasztható, hogy a tesztelés fázisában lévő (még nem végleges) meghajtó programokat is feltüntesse a menüben, vagy nem (*Code maturing level options → Prompt for development and/or incomplete code/drivers*). Itt érdemes ezt a funkciót engedélyezni, mert az esetleges új eszközeinknek lehet, hogy ilyenek a meghajtó programjai.

Tovább haladva a betölthető modulok támogatását tudjuk engedélyezni (*Loadable module support → Enable loadable module support*). A 2.0 Linux megjelenése óta lehet meghajtó programokat modulként fordítani. Erre az insmod és a modprobe parancsok használhatók.

Harmadik menüpontban a processzorunk fajtáját tudjuk kiválasztani (Processor type and features). Itt különösen fontos, ha maximális teljesítményt szeretnénk a rendszerünknek, akkor a processzorunknak megfelelőt válasszuk ki, de kiindulva abból, hogy a mai IBM PC-k processzorai rendre i386 kompatibilisek, elegendő lehet számunkra 386 kompatibilisre fordítani. Sok más funkció található itt például a

sokprocesszoros rendszerek (*Symmetric multi-processing support*), vagy a Co-processzor emuláció (*Math emulation*) támogatása.

A negyedik menüpont alatt az általános beállításokat (*General setup*) találhatjuk. Számítógép-hálózatok támogatása (*Networking support*), PCI, ISA busz, EISA, MCA, PCMCIA/Cardbus, stb. eszközök támogatása. Kiválaszthatjuk kernelünk milyen bináris állományként forduljon le (ELF, vagy a.out). Megtalálható az a.out (GCC-vel fordított bináris állományok) futtatásának, illetve ELF és MISC bináris állományok futtatásának támogatása, az APM (*Advanced power management*) támogatása, amely segítségével a Linux is képes leállítani az ATX tápellátású számítógépeket.

Ezek voltak a fő beállítási opciók, most pedig csak a számunkra fontos menüpontokat vesszük át:

Párhuzamos port támogatás (*Parallel port support*) → Érdemes modulként lefordítani

Plug and Play konfiguráció (*Plug and play support*) → Ez mehet a kernelbe

Block devices: - Érdemes a floppy támogatást kernelbe fordítani, ha szükségünk van néhány meghajtóra (pl.: párhuzamos portra kapcsolható cd-rom támogatás) azt modulként fordítsuk le

Multi-device support → Ezen menüpont alatt kapcsolhatjuk be pl. a RAID támogatást

Hálózati opciók (*Networking options*), ezt nézzük kicsit részletesebben:

Packet socket

Network packet fitering → Itt állítható be, hogy csomagszűrést akarunk használni. Lenyíló menüjét lejjebb találhatjuk meg.

Socket filtering → Szükséges az ARP (MAC szintű protokollok) számára pl.: dhcpd

UNIX Domain sockets

TCP/IP networking → TCP/IP protokollcsalád támogatása. A lenyíló menüben található: *IP: multicasting* = multicast címzés az IP-n, *IP: advanced router*, *IP kernel level autoconfig*, *IP tunneling* = IP alagút támogatás, *IP: GRE tunnels over IP*, *IP: multicastrouting* = multicast routolás támogatása.

IP csomagszűrő konfiguráció (*IP: Netfilter Configuration*)

Connection tracking → Csatlakozás segítő az alkalmazás szintű protokollok számára (Pl.: ftp, irc, tftp …) NAT-hoz (9. fejezet)

Userspace queueing via NETLINK → A későbbiekben bemutatott shaperd forgalomkorlátozáshoz szükséges modul, vigyázat még experimental, azaz néha előidézhet bolondságokat!!!

IP tables support → A későbbiekben (a 9.fejezetben) bemutatott hálózati forgalomirányításhoz és csomagszűréshez szükséges programkódok. Itt érdemes az almenüben lenyíló összes (nem experimental) funkciót modulként lefordítani, mert későbbiekben szükség lesz rá, és ha használni szeretnénk az iptables automatikusan betölti azokat.

ARP tables support, ARP packet filtering → szintén fontos a MAC szintű kezelőprogramok számára.

Visszalépve egy menüpontot ismét a hálózati opcióknál találjuk magunkat. Itt a következő pont az IPv6 protokoll (*The IPv6 protocol*). Ez is még fejlesztés stádiumában álló funkció, sok program (pl.: apache web szerver) tartalmaz biztonsági réseket az IPv6 alkalmazásánál.

IPv6 esetében is használhatjuk a Linux kernelünket csomagszűrésre (*IPv6: Netfilter Configuration*). Ezt a funkciót az iptables6 programmal tudjuk használni, azonban ez még fejlesztési stádiumban lévő funkció, nem meglepő esetleges kernel pánik okozás.

További hasznos funkciók: *802.1Q VLAN support*. A VLAN (lásd: Kommunikációs redszerek programozása) használatára beállíthatjuk esetleges szoftveres hálózati hidunkat vagy routerünket.

The IPX protocol: IPX hálózati protokoll támogatása

802.1d Ethernet bridgeing: Ethernet hálózati híd támogtása

További menükben megtalálhatjuk az IDE/ATA, SCSI, karakteres és grafikus megjelenítők, hang- és videókártyák, hálózati csatolók meghajtó programjait. Ezt mindenki saját „*masinájához*” konfigurálhatja, azonban amit lehet érdemes modulként lefordítani, mert ha szükség van rá, akkor csak be kell tölteni azokat…

7 Shell scriptek

7.1 Shell-ek fajtáji, kezelésük

Azt a programot, amely a rendszer használata során parancsainkat várja, és végrehajtja Shell-nek (parancsértelmezőnek) hívjuk. A parancsértelmező típusa és viselkedése rendszerenkként változó, illetve egy adott operációs rendszeren belül akár több fajta Shell-t is találhatunk, attól függően, hogy melyiket szoktuk meg, vagy melyiket szeretjük. A továbbiakban bemutatunk röviden néhány Shell típust, és a különbségek érzékeltetésére megmutatjuk, hogy az egyes típusoknál hogyan kell egy környezeti változó értékét beállítani.

Néhány Shell típus a UNIX világából:

- bash (Pl.: Linux GNU Hurd) [Mérete ≈ 500Kbyte]
- csh (Pl.: IBM AIX, IRIX) [Mérete ≈ 320Kbyte]
- ksh (Pl.: AT&T UNIX, Solaris) [Mérete ≈ 630Kbyte]
- ash (Pl.: Minimum Shell, boot lemez esetén) [Mérete ≈ 60Kbyte]

A bash. A Linux különböző kiadásai általában a *bash* nevű Shell-t használják alapértelmezésként. A *bash* különböző kényelmi szolgáltatásokat nyújt a parancsok begépelésének megkönnyítésére:

- A ↑ és ↓ billentyűkkel böngészhetjük a régebben kiadott parancsokat
- A [TAB↔] megnyomásával egy fájl, parancs vagy könyvtár nevet egészíti ki
- A [CTRL-R] billentyűkombináció megnyomása után kereshetünk a régebben beírt parancsok között úgy, hogy elkezdjük újra begépelni az első karaktereket

A *bash* esetén például a TERM környezeti változó beállítását a következőképp végezhetjük el:

```
bash-2.05> export TERM=vt100
bash-2.05> set |grep TERM
COLORTERM=
```

TERM=vt100

A bash megengedi azt is, hogy a parancsokat átnevezzük, illetve összetett parancsokat rövidekkel helyettesítsünk:

bash-2.05> alias ls='ls --color'
bash-2.05> alias dir='ls –la'
bash-2.05> alias cp='logout' (☺)

Az *unalias* paranccsal megszüntetjük az *alias*-okat:

bash-2.05> alias
alias cp='logout'
alias dir='ls –la'
alias ls='ls --color'
bash-2.05> unalias cp
alias dir='ls –la'
alias ls='ls --color'

A ksh. A ksh egy másik elterjedt Shell típus, melyet főleg a programozók kedvelnek, mivel szkriptelési lehetőségei bővebbek, mint pl. a bash vagy a csh esetén. A környezeti változók beállítása, illetve az alias-ok létrehozása itt is hasonlóképp történik:

$ export TERM=vt100
$ alias dir='ls --color'
$ unalias dir

A ksh-nak számos kiterjesztése létezik, pl. a dtksh (desktop ksh) olyan modulokat is tartalmaz, amely a Motif grafikus programkönyvtárral együttműködve közvetlen grafikus megjelenítésre is alkalmas.

A csh. A csh szinte minden UNIX-al együtt született Shell, a legrégebben alkalmazzák, szintaktikája nagyon közel áll a C nyelvhez. A ↑ és ↓ billentyűk itt is rendelkezésre állnak, parancs-visszakeresés céljára, illetve a [TAB↔] kiegészítő funkció is működik. Környezeti változók beállítása:

% setenv TERM vt100

7.2 Shell scriptek írása

Itt a UNIX Shell-ek programozásának alapjait nézzük meg, konkrétan a bash script-ekről lesz szó.

Mielőtt bármibe is belekezdenénk, nézzünk meg néhány szakkifelezést, melyeket használnak az angol szakirodalomban.

Jel	*Magyar jelentés*	*Angol jelentés*
{}	Kapcsos zárójel	(Curly) brace
( )	Zárójel	Parenthesis
[ ]	Szögletes zárójel	(square) bracket
”	Macskaköröm	Double quote
’	Aposztróf	(Single) quote
\	Vissza per	Backslash

A kapcsos zárójel kifejtése (Brace expansion). A kapcsos zárójel a bash-ban szövegrészek automatikus behelyettesítésére használható:

bash-2.05> echo {Smith,Taylor}@ieee.org

Smith@ieee.org Taylor@ieee.org

A helyettesítés csak arra szóra működik, amelybe belefoglaltuk:

bash-2.05> echo E-mail: lencse@rs1.{szif,sze}.hu

E-mail: lencse@rs1.szif.hu E-mail: lencse@rs1.sze.hu

Lássunk egy összetettebb példát is:

bash-2.05> echo lencse@{{rs1,mail}.,””}{sze,szif}.hu

lencse@rs1.sze.hu lencse@rs1.szif.hu lencse@mail.sze.hu lencse@mail.szif.hu

lencse@sze.hu lencse@szif.hu

A tilde kifejtése (Tilde expansion).

A ~ (tilde) jel speciális jelentéssel bír a UNIX shellek többségénél: ha egy felhasználónevet írunk utána, akkor az egész string egyenértékű lesz a felhasználó home könyvtárának elérési útjával. Ha mögé / jelet, illetve további elérési utat írunk, saját home könyvtárunkhoz képest értelmezett relatív elérési utat adunk meg:

bash-2.05> echo ~

/root

bash-2.05> echo ~lencse

/home/lencse

bash-2.05> echo ~/jegyzet

/root/jegyzet

bash-2.05> echo ~lencse/public_html

/home/lencse/public_html

További szempontok:

- egy szóban legfeljebb 1 darab szerepelhet
- az előtte álló szövegrész változatlan marad
- egymagában alkalmazva a saját home könyvtárunkat kapjuk

Paraméterek és változók kifejtése (Parameter expansoin)

A bash-ban a változók ugyanúgy tetszőleges értéket tartalmazhatnak, mint az egyéb nyelvekben. Ha az értéküket szeretnénk megadni, dollárjel segítségével kell a tartalmukra hivatkozni:

bash-2.05> alma=apple

bash-2.05> echo alma

alma

bash-2.05> echo $alma

apple

bash-2.05>echo ”Az alma angolul: $alma”

Az alma angolul: apple

Eredmények helyettesítése (Command substitution)

Néha szükséges, hogy egy parancs kimenetét felhasználjuk valamilyen célra, ilyenkor a parancsot következőképp tudjuk behelyettesíteni:

bash-2.05> mkdir gyak

bash-2.05> cd gyak

bash-2.05> echo egy ketto harom negy ot hat het > file_list.txt

bash-2.05> cat file_list.txt

egy ketto harom negy ot hat het

**bash-2.05> touch ’cat file_list.txt’*

bash-2.05> ls

egy file_list.txt harom hat het ketto negy ot

Más lehetséges megoldása a *-al jelölt parancsnak:

bash-2.05> touch $(cat file_list.txt)

Matematikai kifejezések kiértékelése (Arithmetic expansion)

Ahhoz, hogy a bash a matematikai kifejezéseket kiértékelje, speciális zárójelzést kell alkalmazni:

*bash-2.05> echo 2*3*

*2*3*

*bash-2.05> echo $((12*33))*

36

*bash-2.05> echo $((2**(2,3)))*

8

bash-2.05> echo $((2!=(a=3)))

1

A bash a kiértékelt kifejezések értékét long integer típusú változókban tárolja el. A kiértékelés prioritása, szintaktikája és módja a C nyelvével azonos.

Szavakra bontás (Word splitting)

A bash fontos tulajdonsága, hogy egy bemenetre érkező több szálból álló stringet vagy adatsort szavakra bont. A szavak tárolását egy IFS (Internal Field Separator) nevű változóban megadott karakterek alapján dönti el. Ezek alapesetben a szóköz, TAB, újsor karakterek. A szavakra bontás miatt azokat a fájl neveket, melyekben szóköz található, védő idézőjelekkel vagy backslash-el kell ellátni.

bash-2.05> touch ”trukkos nev”
bash-2.05> ls –l
total 1
-rw-r--r-- 1 lencse users 7 Sep 17 10:31 trukkos nev
bash-2.05> rm trukkos nev
rm: cannot remove ’trukkos’: No such file or directory
rm: cannot remove ’nev’: No such file or directory

Elérési utak kifejtése (Path name expansion)

Ha egy parancsban fájlok vagy könyvtárak egy meghatározott csoportjára szeretnénk hivatkozni, akkor az úgynevezett „joker” karaktereket kell használnunk. A bash esetén ezek a következők:

* A helyén nulla vagy bármennyi tetszőleges karakter állhat

? A helyén egy darab tetszőleges karakter állhat

[XYZ] A helyén a felsorolt karakterek bármelyikéből 1 darab állhat

[a-g] A helyén a megadott karakter-tartományból való karakterek bármelyike állhat

[^XYZ] A helyén megadott karakterek közül egyik sem állhat

bash-2.05> mkdir gyak
bash-2.05> cd gyak
bash-2.05> toch 11 111 121 131 141
*bash-2.05> ls 1*1*

11 111 121 131 141

bash-2.05> ls 1?1

111 121 131 141

bash-2.05> ls 1[1,2]1

111 121

bash-2.05> ls 1[1-3]1

111 121 131

bash-2.05> ls 1[^2]1

111 131 141

A joker karakterek esetén a fájlnév elején álló „.” karaktert speciálisan kell kezelni, mert erre nem érvényesek a joker szabályok. Explicit illeszkedésre van szükség:

bash-2.05> mkdir gyak

bash-2.05> cd gyak

bash-2.05> touch .1 .11 1 11

*bash-2.05> ls *1**

1 11

*bash-2.05> ls ?1**

11

*bash-2.05> ls .1**

.1 .11

Idézőjelek kifejtése (Quote removal)

Munkáink során az egybefüggő szöveges változókat (string-eket) általában idézőjel szerű karakterekkel védjük. Ilyenek a ’ és a ”. Pl.:

bash-2.05> echo ”Ez itt a string”

Ez itt a string

Az idéző jelek is megvédhetők:

bash-2.05> echo \”Ez itt a string\”

”Ez itt a string”

A standard ki és bemenetek irányítása

A UNIX-ban három standard I/O csatorna létezik:

I/O	File descriptor
Standard input	0
Standard output	1
Standard error	2

A standard bemenetről fogadják a programok a bemenő adatokat, és a standard kimeneten olvashatjuk a program kimenő üzeneteit. A standard error a hiba kimenet, ide írják a programok a hibaüzeneteket. Alapértelmezésben a standard out és a standard error a konzolra, vagy a terminálra vannak irányítva, azonban a felhasználóknak lehetősége van átirányítani ezeket a ki és bemeneteket.

A standard kimenet átirányítása:

bash-2.05> ls –l > lista

bash-2.05> echo elso sor > file

bash-2.05> echo masodik sor >> file

bash-2.05> cat file

elso sor

masodik sor

A standard bemenet irányítása pl. kernel patchelés esetén:

bash-2.05> patch –p1 < /usr/src/patch-2.4.22rc3.patch

A következő példában a sort illetve a grep programok standard bemenetről veszik az adatot, ami viszont egy pipe (cső) kimenetéhez van csatlakoztatva:

bash-2.05> touch alma korte dinnye uborka

bash-2.05> ls | short –r

uborka

korte

dinnye

alma

bash-2.05> ls | grep alma

alma

Inditáskori parancsfájlok.

Ha a bash egy interaktív login shell (pl. mikor belépek egy gépre), akkor:

Először, ha van, akkor végrehajtja

/etc/profile

Utána ebben a sorrendben próbálva az elsőt (ami létezik és végrehajtható)

~/.bash_profile, ~/.bash_login, and ~/.profile

A user által kiadott parancsok bekerülnek:

~/.bash_history

Ha a bash login shell, mikor kilép, végrehajtja:

~/.bash_logout

Interaktív, de nem login shell:

~/.bashrc

Nem interaktív shell:

BASH_ENV-et kifejti, és azt használja fájlnévként, de nem használja a PATH-t...

Fontosabb környezeti változók.

1. amiket a bash használ:

IFS

PATH

HOME

MAIL/MAILCHECK/MAILPATH

PS1, 2, (3,4)

HISTORY, HISTSIZE

HISTFILE, HISTFILESIZE

2. Amiket beállít

PWD, OLDPWD

UID, EUID

HOSTNAME

7.3 Egyszerű shell script példák

Néhány egyszerű példa szemlélteti, hogy a shell scriptel milyen feladatok oldhatók meg, ami más módon nehezen megoldható, vagy időigényes lenne.

A *sed* parancs (más néven stream editor) szöveges állományok kezelésére szolgál. A használata egyszerű, egy program standard outputját irányíthatjuk a *sed*-be, és műveleteket végezhetünk vele. A *–s* opció segítségével a megadott sztring-et kicserélhetjük egy másikra, míg a –g opció a globális feldolgozást (minden megadott sztringet) jelent.

Példánkban létrehoztunk egy proba nevű fájlt, ami tartalmazza a *korte* és alma sztringeket. Az alábbi parancs a *korte* sztringet kicseréli almára, és kiíratjuk a standard output-ra.

lencse@teacher:~/# cat proba | sed s /alma/korte /g

alma alma

Egy újabb példa tanulságos lehet minden rendszergazda palánta számára. Szeretnénk a könyvtárunkban található *.htm* fájlokat *.html*-re kicserélni. Ezt megtehetjük a következő script segítségével:

Fájlok létrehozása:

lencse@teacher:~/# touch 1.htm 2.htm .htm 3.html

*lencse@teacher:~/# for i in *; do mv `echo $i | sed ”s/\.htm$/.html/”`;done*

Ahol a *for i in* * ; do; done egy ciklus a \ a . –ot (mint helyettesítő karaktert) megvédi, a *.htm$* a *htm*-re végződő string-eket jelöli. A ciklus *-ig számlál, amennyi fájl van a könyvtárban. A script minden egyes fájlt megpróbál átnevezni, mivel a nem *.htm* végződésű fájlokon a *sed* nem végez konverziót, az *mv* nem nevezi át ugyanarra a névre a fájlt, így hibaüzenetet ír ki. A ` ` karakterek használata a paraméter kifejtését jelenti. Másképpen a $( ) – tudjuk megadni.

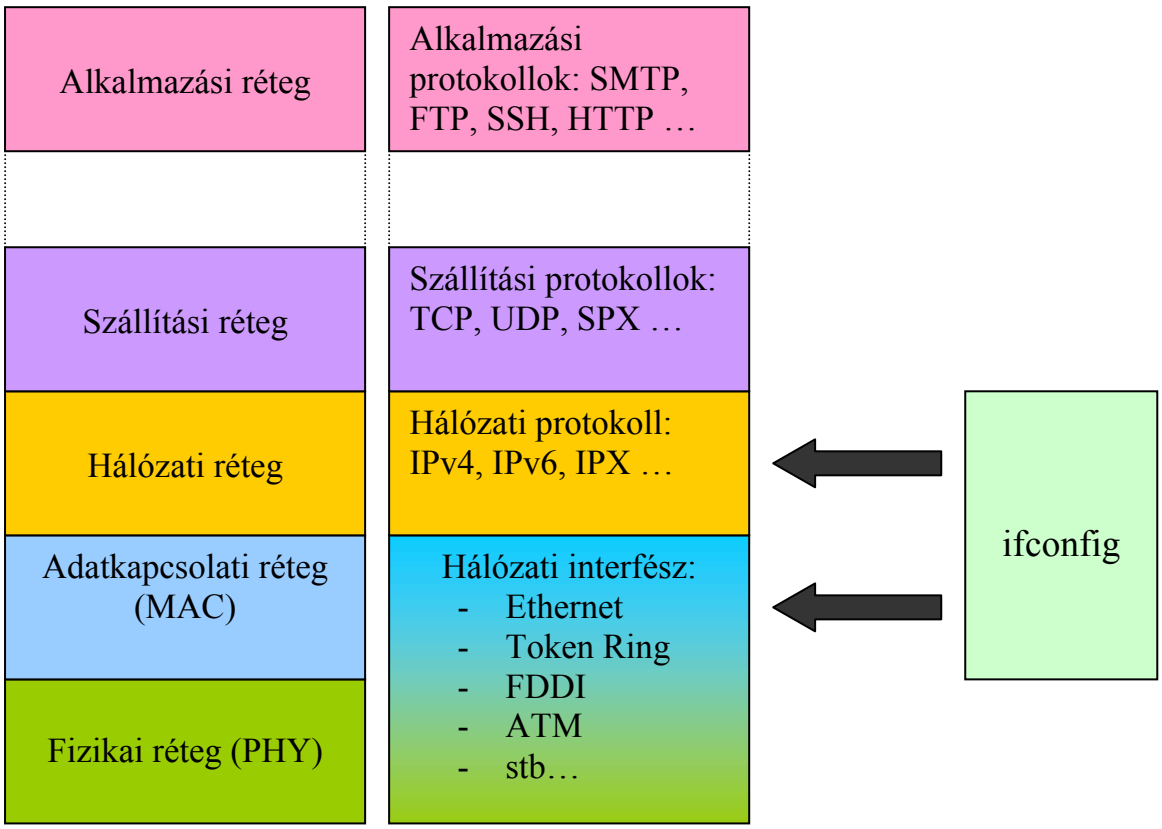
Egy újabb hasznos program a *tr* ami string-ek fordítását végzi. A következő példánk a nagybetűből álló (pl.: MS-DOS partícióról másolt) fájlok neveit kisbetsűre nevezi át. A *tr*-hez használt paraméter az [:upper:] a nagybetűket, míg a [:lower:] a kisbetűket jelöli.

*lencse@teacher:~/# for i in *; do mv $i `echo $i | tr [:upper:][:lower:]`; done*

8 Hálózati interfészek konfigurációja

8.1 Interfészek paraméterezése

A hálózati beállításokat minden UNIX rendszerben az *ifconfig* paranccsal tudjuk végezni. Emlékezzünk vissza a számítógép-hálózatok tantárgyban tanultakra: a hálózati interfészeknek szükségük van felsőbb szintű protokollokra, hogy használhatóak legyenek számunkra (szolgáltatásokat nyújthassunk rajta).



5. ábra: Az ifconfig hatásköre az OSI modell szerint

Az 5. árba szerint látható, hogy az *ifconfig*-gal a hálózati interfészünk adatkapcsolati és hálózati rétegbeli tulajdonságait paraméterezzük. Magát a beállításokat a kernel végzi el, az *ifconfig* csak a kernelnek ad utasítást a paraméterezésekre.

Adatkapcsolati réteg paraméterezése. A UNIX rendszerekben a hálózati interfészekre hivatkozni kell. A Linux esetében az Ethernet típusú hálózati

csatolókártyák hivatkozása: eth0, eth1 … függően attól, hány darab Ethernet hálózati kártya van felkonfigurálva a rendszerünkben. A ponttól-pontig kapcsolatú eszközökre (telefonos modem, adsl modem) például ppp0 –ként hivatkozunk. Vezeték nélküli interfészek esetén wlan0.

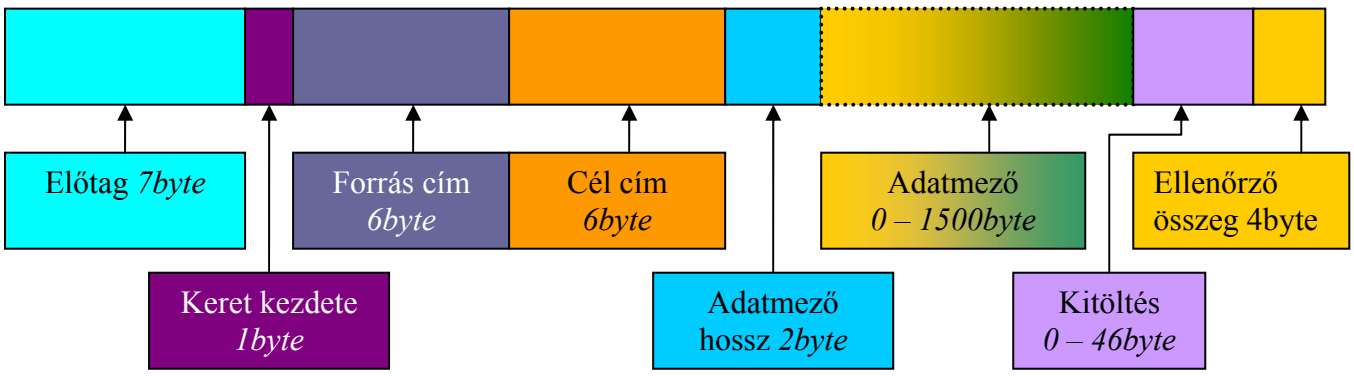
Vizsgáljuk meg Ethernet esetén az adatkapcsolati réteg paramétereit!

Mivel a labor gépein rendszerindításkor felkonfigurálódik az eth0 hálózati interfész, vizsgáljuk meg az ifconfig kimenetét!

root@pc0:/# ifconfig eth0

```
eth0   Link encap:Ethernet  HWaddr 00:06:25:0B:03:36
       inet addr:193.224.130.170  Bcast:193.224.130.191  Mask:255.255.255.224
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:1208 errors:0 dropped:0 overruns:0 frame:0
       TX packets:819 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       RX bytes:111845 (109.2 KiB)  TX bytes:379338 (370.4 KiB)
       Interrupt:3 Base address:0x100
```

Nézzük milyen paraméterek tartoznak az adatkapcsolati rétegbe. Idézzük fel az Ethernet keretszerkezetét (6. ábra).



6. ábra: Ethernet (IEEE802.3) keretszerkezete

Az MTU azaz Maximum Transfer Unit (maximálisan átvihető byte-ok száma egy keretben) értéke jelen esetben 1500byte, tehát a maximum adatmező hossz. Egyes esetekben szükséges ennek a módosítása azt az *ifconfig <interface> mtu <szám byte-ban>* parancsal tehetjük meg.

Az *ifconfig*-gal állíthatjuk a hálózati csatolókártyánk MAC címét is (*ha a kártya meghajtó modulja támogatja ezt a funkciót*). A jelenlegi MAC címünk az ifconfig kimenetének jobboldali legfelső sorában láthatjuk. Az interfész MAC címét átállíthatjuk kedvünk szerint, de figyelni kell a szabályokra: nem adhatunk olyan MAC címet, ami már szerepel a hálózati szegmensben, illetve nyílván van tartva a switch címlistájában. A cím 6byte-os (6 darab kétszer 0 – F hexadecimális számokból állhat, 2 szám után kettősponttal választjuk el őket). Pl.: 00:06:36:88:44:3F. A címet az *ifconfig <interface> hw ether <új MAC cím>* paranccsal változtathatjuk meg.

Az *ifconfig* továbbá a kernel által mért átvitt adatok mennyiségét is leméri az adott interfészen: RX packets, RX bytes, TX packets, TX bytes. Az RX packets a fogadott csomagokat, az RX bytes a fogadott byte-okat, míg a TX packets a küldött csomagokat, a TX bytes pedig a küldött byte-okat jelzi.

Méri az elveszett csomagokat, és az ütközéseket is.

Érdemes összehasonlítani több gép esetén, hogy milyen eredmények mérhetőek HUB és Switch használatakor.

Hálózati konfiguráció. A hálózati réteg paraméterezése során meg kell adnunk, hogy milyen protokoll szerint szeretnénk felkonfigurálni a hálózatunkat. Számítógép-hálózatok gyakorlaton az *ifconfig*-gal csak IP címet adtunk meg. Ez azért lehetséges, mert az alapértelmezett hálózati protokoll a TCP/IP, de az *ifconfig*-gal lehetséges IPv6-ot és IPX-et is felkonfigurálni.

IPv4 felkonfigurálás: ifconfig <interface> inet <32bit-es IP cím> <netmask> <broadcast> up

Az *ifconfig* (helyes paraméterezése során) a hálózatcímet automatikusan kiszámítja.

A *route* paranccsal meggyőződhetünk erről.

Az alapértelmezett átjárót szintén a *route* paranccsal adhatjuk meg.
route add default gw <32bit-es IP cím>

IPv6 felkonfigurálás: ifconfig <interface> inet6 <128bit-es IP cím / tartomány>
A hálózatcímünket és az alapértelmezett átjárót a *route6* paranccsal adhatjuk meg.

IPX felkonfigurálás: ifconfig <interface> ipx <hálózatcím> <gépcím == MAC cím>
Az IPX átjárót az *ipx_route* paranccsal adhatjuk meg.

8.2 Az arp, arping, és a rarp

Egy kis elmélet: Az arp (Address Resolution Protocol = Cím felbontó protokoll) OSI 2. rétegbeli protokoll. A Routerek a hálózati szegmensben úgy juttatják el a kereteket a címzettnek, hogy arp-t használva felderítik a szegmensen lévő hálózati interfészek MAC címeit. Ezeket az IP cím, MAC cím párosokat eltárolják. Ezt hívják arp cacheing–nek. A UNIX-ban használt arp parancs arra szolgál, hogy ezt az arp cache-t kiírassuk. Az arping paranccsal megnézhetjük az adott IP című interfész MAC címét. *Figyelem: a Routerek OSI 3. rétegében kötik össze a hálózati szegmenseket, így egy nem szegmensünkön lévő interfész IP címét arping-elve a Router felénk eső interfészének MAC címét fogjuk megkapni (lásd: Számítógép-hálózatok).*

8.3 Hálózati híd létrehozása kettő vagy több interfész között

A hálózati híd (más néven: *bridge*) nem más, mint ütközési *domain*-eket választ szét hálózati szegmensek között. A *bridge* segítségével összekapcsolhatunk több hálózati szegmenset adatkapcsolati szinten. A hálózati híd funkció a Linux kernelben megtalálható funkció, ami a tanult *bridging* módszerek közül a DEC által fejlesztett feszítőfa (azaz Spanning Tree) algoritmust (IEEE 802.1d) használja. A bridge üzemmódot akkor vehetjük igénybe, ha azt kernelbe, vagy modulba lefordítjuk. Ajánlott modulként lefordítani, így csak akkor lesz jelen a rendszerünkbe, ha betöltjük azt. A funkciót beállító programot Bridge Utils-nak hívják. Debian csomagkészletben megtalálható (installálás: *apt-get install bridge-utils*).

Bridge létrehozása. Előfordulhat, hogy nem áll rendelkezésünkre hardveres bridge, ilyen esetekben nekünk kell létrehozni azt szoftveres úton.

Első lépésként meg kell vizsgálni, hogy be vannak –e töltve a hálózati hídként használandó interfészek moduljai. Ezt az *lsmod* paranccsal vizsgálhatjuk meg. Ha nincsenek, töltsük be őket. (Labor gép esetében: *modprobe 8139too* = SMC hálózati kártya, *modprobe eepro100* = Alaplapi INTEL hálózati kártya). Miután meggyőződtünk, hogy be vannak töltve a modulok, vizsgáljuk meg, hogy ne legyenek paraméterezve az interfészeink (*ifconfig eth0 down, ifconfig eth1 down*). Ezután nekiállhatunk a bridge létrehozásának. Minden interfésznek végezzük el a beállítását (melyek a hidat képezni fogják) a következő módon.

root@pc0:/# ifconfig eth0 0.0.0.0 up

root@pc0:/# ifconfig eth1 0.0.0.0 up

Töltsük be a bridge kernel-modult (amennyiben modulként lett lefordítva).

root@pc0:/# modprobe bridge

Most definiálom a *br0* logikai interfészt, ami a hidat jelenti. Ez az interfész fizikailag is meg fog megjelenni az *ifconfig* nyilvántartásában. Ezt a *brctl* (bridge control) programmal tudom megtenni.

root@pc0:/# brctl addbr br0

Majd hozzáfűzöm a *br0* logikai hídhoz a fizikai interfészeket. Szintén a *brctl* parancsot használva.

root@pc0:/# brctl addif br0 eth0 eth1

Ezután konfiguráljuk fel a *br0* logikai interfészt.

root@pc0:/# ifconfig br0 193.224.130.170 netmask 255.255.255.224 broadcast 193.224.130.191 up

root@pc0:/# route add default gw 193.224.130.161

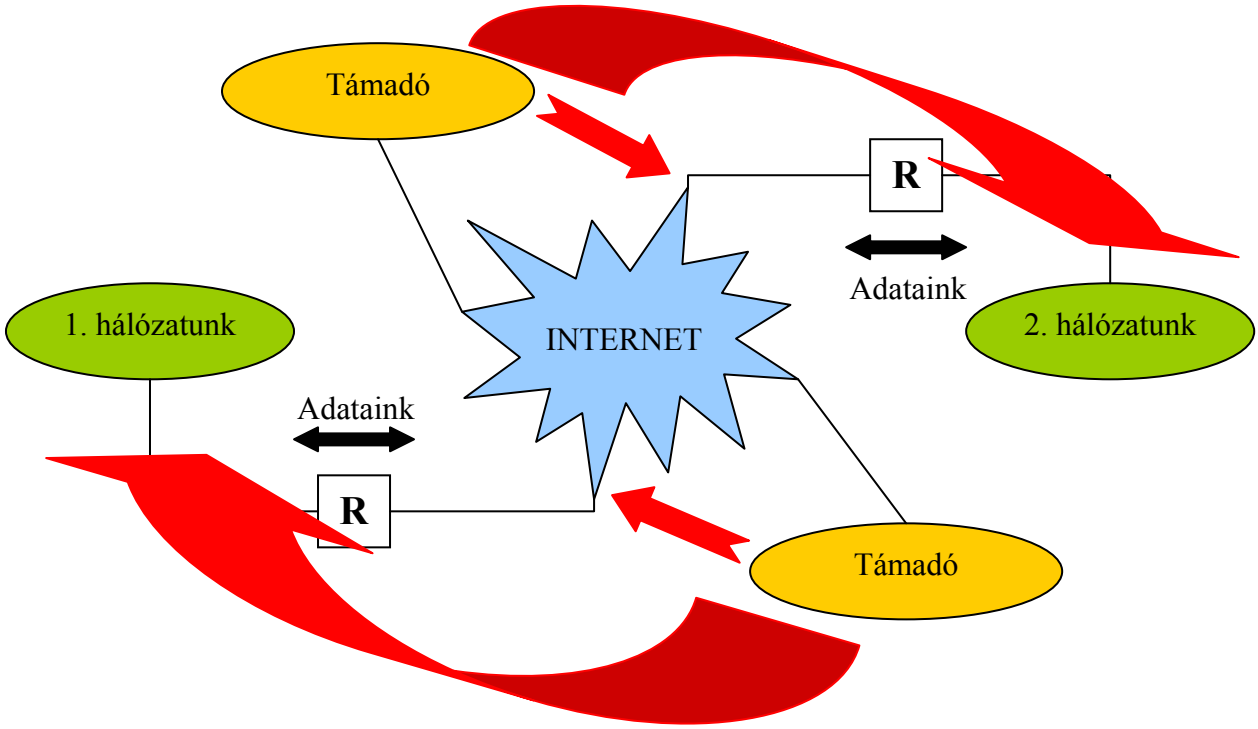
Majd végül indítsuk el a bridge –et irányító démont.

root@pc0:/# brctld

8.4 Hálózati alagút, VPN és IPsec

Vegyünk egy egyszerű esetet. A cégünk két telephellyel rendelkezik. Milyen megoldásaink lehetnek, hogy összekössük őket Interneten keresztül?

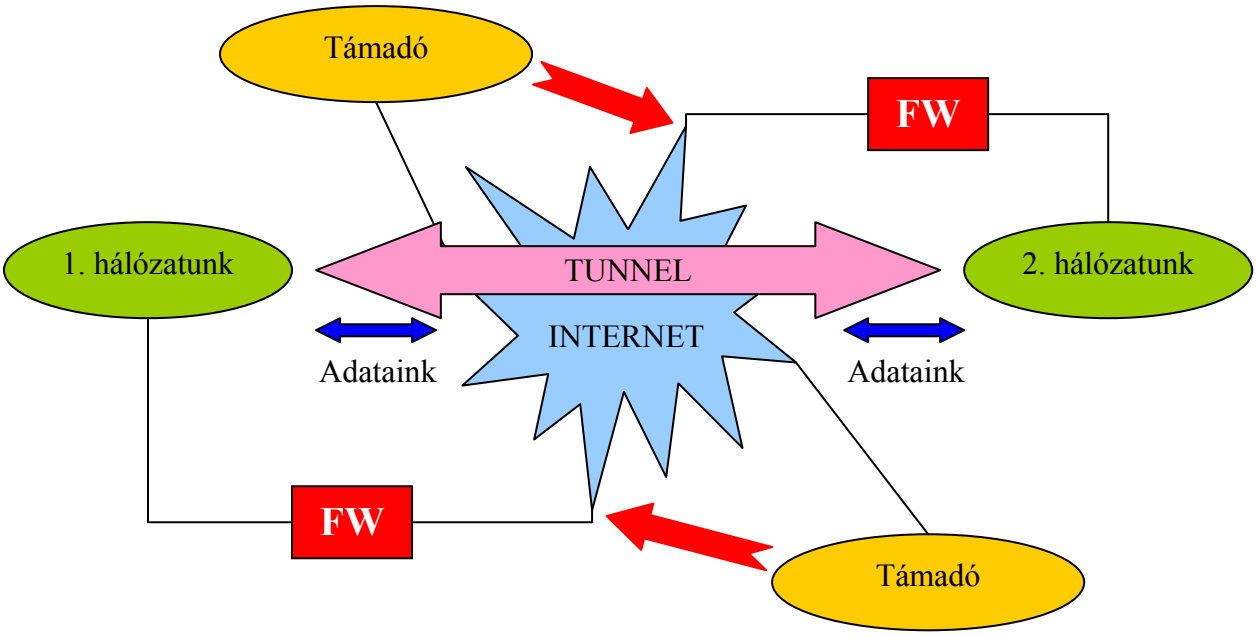
- Router alkalmazása: A gépeink kívülről is láthatóak, a két telephely közötti forgalmat nyilvánosan az Interneten keresztül üzemeltetjük (7. ábra)
- Tűzfal alkalmazása: A gépeink belső hálózaton vannak „elrejtve”, belső IP címmel, és hálózati alagutat hozunk létre az Interneten keresztül (8. ábra)
- Tűzfal alkalmazása VPN & IPsec funkcióval: Azaz tűzfal mögött helyezkednek el a számítógépek, és az Interneten *virtuális magánhálózatban* mennek az adataink, amit IPsec–el titkosítunk (9. ábra)



7. ábra: Adatcsere Interneten keresztül és támadási lehetőségek

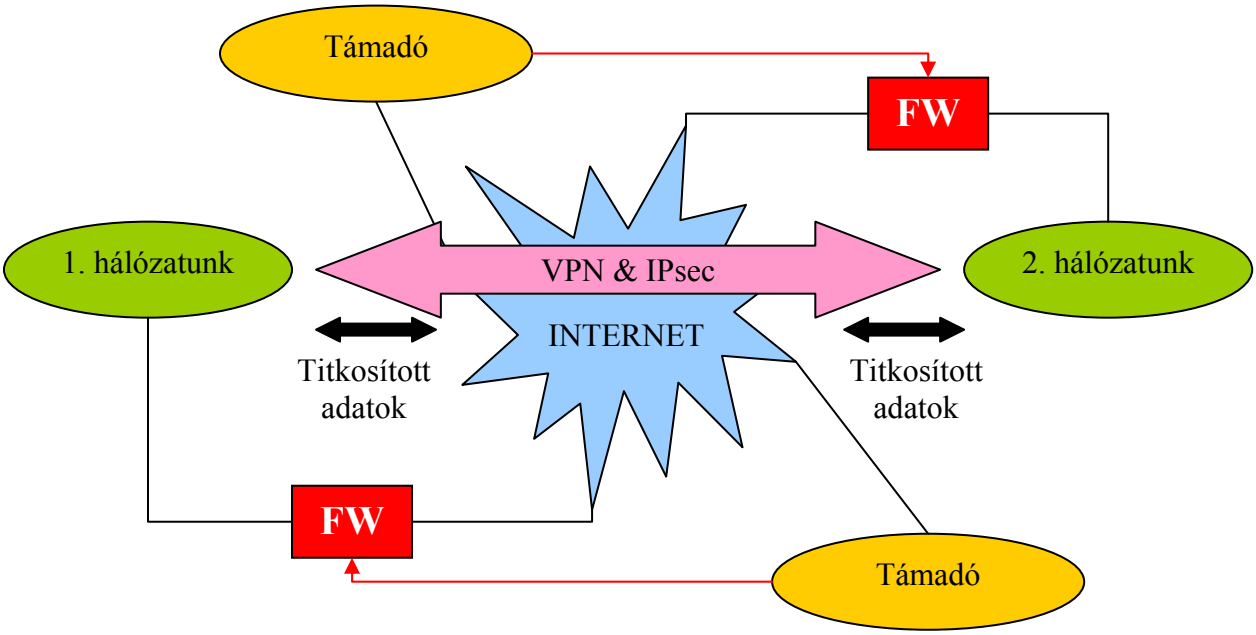
A 7. ábrán látható megoldás nem túl biztonságos, ugyanis a támadó kívülről eléri a céges gépeket. A feltört gépekről könnyedén megszerezheti a cégünk titkos adatait. A másik lehetséges támadás, amikor a cég külső forgalmát figyeli a támadó, és onnan szerzi meg az adatokat, amiket a telephelyek között forgalmazunk.

Hálózati alagút (tunnel). A tűzfal megakadályozza, hogy a külső támadók betörjenek a hálózatunkba, viszont az alagutakban áramló adatokat könnyedén le tudják hallgatni, ugyanis azok nincsenek titkosítva. Igaz a hálózati alagút kissé megnehezíti a támadó feladatát, mert a *tunnel* ponttól-pontig (ppp) kapcsolatot épít ki. Így a támadó csak a teljes forgalom lementésével, és annak böngészésével tudja „kiválogatni" a fontos adatokat. A *tunnel* alkalmazása mára már elavultnak számít, bár manapság is alkalmazzák néhány helyen.



8. ábra: Hálózati alagút megoldás és támadási lehetősége

VPN és IPsec. A VPN & IPsec megoldások az eddigi legmegbízhatóbbak. Adataink nemcsak, hogy alagutakban (*virtuális magánhálózatban*) száguldoznak, hanem IP szinten titkosítva vannak.



9. ábra: VPN & IPsec megoldás és támadási lehetősége

Az egyetlen támadási lehetőség, ha a tűzfalat feltörik, és megszerzik a titkosított csomagok kikódolásához a jelszót. Viszont ezt nagyon nehéz véghez vinni, és csak úgy lehetséges, ha a tűzfalon olyan szolgáltatások futnak, amelyek biztonsági rést tartalmaznak. Ezért érdemes a szolgáltatásokat nyújtó szervereinket tűzfal mögé elhelyezni, és a szolgáltatások portjait úgymond *ki-forward*–olni (erről bővebben a következő fejezetben lesz szó).

A következőkben megnézzük, hogy miképp is kell egy ilyen rendszert megvalósítani.

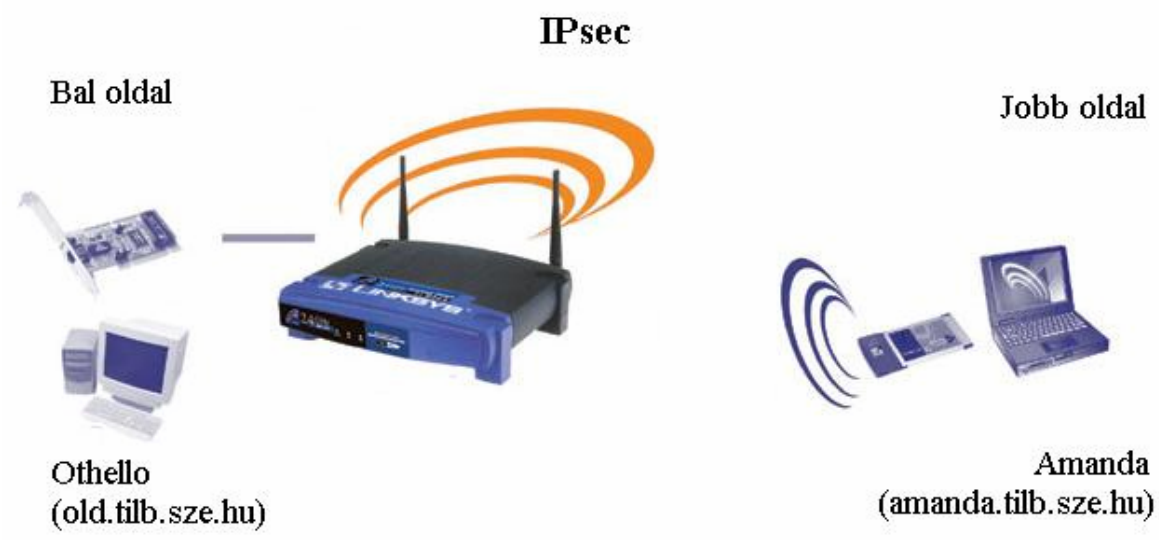
Megjegyzés: A vezeték nélküli hálózatok alkalmazása során is az IPsec használata az ajánlott vezeték nélküli összeköttetéseink védelmére.

8.5 IPsec megvalósítása FreeS/WAN-al.

A FreeS/WAN nyílt forráskódú, kiterjedt biztonságos számítógép-hálózat, a legnépszerűbb és az egyik legérettebb IPsec-megvalósítás, ami Linux rendszereken működik. Minthogy ez az IP-protokoll kiterjesztése, ez az Internet „hivatalos” VPN-protokollja.

A NetFilterhez hasonlóan a FreeS/WAN is a tényleges munkát végző rendszermagmodulból és a felhasználói felületekből áll. A NetFiltertől eltérően azonban a FreeS/WAN nem szerepel a Linux-rendszermag forráscsomagjai között, így Debian Linux alatt egy kernel-patch-elő csomagot kell letöltenünk, amivel patch-elve a kernelforrásunkat, új kernelt szükséges fordítanunk (*apt-get isnatall kernel-patch-freeswan*). Majd miután lefordult, és elindult az újonnan lefordított kernel (a FreeS/WAN támogatással) installáljuk fel a FreeS/WAN programot (*apt-get install freeswan*).

Miután felinstalláltuk a FreeS/WAN-t és kernelünk is vígan elboldogul az ipsec-es feladattal, nekiállhatunk bekonfigurálni rendszerünket. A konfigurációt két állományon kell elvégeznünk. Az első a */etc/ipsec.conf*, ami a főbb beállításokat tartalmazza, míg a második a */etc/ipsec.secrets*, ami a kulcstár szerepét tölti be. Vigyázzunk arra, hogy mindkét állománynak korlátozott engedélyekkel kell rendelkeznie; a 0600 megbízható választás. A */etc/ipsec.secrets* állományt különös gonddal kell védelmezni. Amennyiben ebből az állományból bármit ki kell másolni, mondjuk a gép nyílvános RSA kulcsát, akkor csak a szükséges dolgokat érdemes kimásolni egy külön fájlba. Soha ne engedjük meg, hogy az *ipsec.secrets* állomány kikerüljön a rendszerből!



10, ábra: Vezeték nélküli VPN&IPsec megvalósítása

Az *ipsec.conf* fájl a kliens felöl nézve.

```
# alapbeállítás
config setup
        interfaces=%defaultroute
        klipsdebug=none
        plutodebug=none
        plutoload=%search
        plutostart=%search
        uniqueids=yes

conn %default
        keyingtries=0
        authby=secret

conn othello-amanda
        authby=rsasig
        left=192.168.0.1
        leftid=@old.tilb.sze.hu
        leftsubnet=0.0.0.0/0
        leftrsasigkey=0pQr6587894f786rF7376vfF8r87Rh87Sq828g9
        right=%defaultroute
        rightid=@amanda.tilb.sze.hu
        rightrsasigkey=0xZ765s5sQ7fR46Ht79gY94jh47rhC83khJdi92
        auto=start
```

Az *ipsec.conf* fájl a szerver felöl nézve.

```
config setup
        interfaces=”ipsec0=eth1”
        forwardcontrol=yes
```

conn othello-amanda

authby=rsasig

left=192.168.0.1

leftid=@old.tilb.sze.hu

leftsubnet=0.0.0.0/0

leftrsasigkey=0pQr6587894f786rF7376vfF8r87Rh87Sq828g9

right=%any

rightid=@amanda.tilb.sze.hu

rightrsasigkey=0xZ765s5sQ7fR46Ht79gY94jh47rhC83khJdi92

auto=add

Az első konfigurációs fájl a kliens oldalt mutatja be (jobb oldali). A fájl három részből tevődik össze: alapvető telepítési kapcsolók (*config setup*), az előrre beállított alagútértékek (*conn %default*) és végül az alagút-meghatározások (*conn othello-amanda*). Ez a név az alagút nevét jelzi, ez szabadon választható. A *config setup* szakaszban található előrre beállított jellemzők az egykártyás rendszereken biztonsággal érintetlenül hagyhatók. A legfontosabb beállítás az *interfaces*, ami megadja, hogy melyik hálózati csatoló lesz az IPsec csatornánk helyi végpontja. Az előrre beállított érték a *%defaultroute* ennek az *ipsec=[csatoló]* a feloldása, ahol a *[csatoló]* a gép alapértelmezett hálózati csatolója. Ezennel máris az konfiguráiós fájl lényegéhez érkeztünk el: az alagút-meghatározáshoz. Az *othello-amanda* alagút meghatározásban az első sorban találjuk az authby beállítást, ami azt szabja meg, hogyan végezzék az egyes IPsec-gépek a saját hitelesítésüket egymás számára. Az alapérték a „*secret*”, amely az előzetesen megosztott titkos azonosítót jelenti. Ez a beállítás lehetővé teszi, hogy az előrre meghatározott titkos karaktersorozat legyen az azonosítási kulcs. Ez elsőre meggondolatlanságak tűnhet, de mégis biztonságos, ugyanis az azonosító sosem utazik a hálózaton. Most más a helyzet, hiszen az authby kapcsoló rsasig-re van állítva. Az RSA-hitelesítés (bővebben az ssh-nál lesz erről szó) nem szükségszerűen sokkal biztonságosabb, ezzel szemben sokkal kényelmesebb. Míg a megszokott azonosítót valamilyen biztonságos eszköz, mint például titkosított levél, vagy SSH segítségével előzetesen ki kell cserélni, az RSA-hitelesítésben használt

nyilvános kulcsok akár ország-világ előtt kicserélhetők, mi több, még a honlapon is közzétehetők.

A FreeS/WAN beállításoknál fontos a bal és jobb oldal megkülönböztetése: ezek a parancsokban és kapcsolókban az IP-alagutak végpontjának megjelölésére szolgálnak. Hogy melyik elnevezés melyik oldalt jelöli, tulajdonképpen lényegtelen, az a fontos, hogy a jelöléseket következetesen alkalmazzuk. Ugyanannak a gépnek mindkét alagút-meghatározásában ugyanazon az oldalon kell lennie, és egy rendszernek sem szabad megfordítania a másik gépen szereplő bal-jobb kiosztást.

A biztonság kedvéért, amikor egy gép a másik számára kiszolgálóként működik, ezt a gépet javasolt baloldalként feltüntetni. Jelen esetben az *othello* nevű gép kiszolgálóként működik a vezeték nélküli ügyfelek számára, így az *othello* a bal oldali, míg az *amanda* nevű számítógép a jobb oldali.

A „bal” alagútbeállítási értéke jelzi az alagút bal oldali végpontjának IP-címét, *othello: 192.168.0.1*. A jobb oldal természetesen a jobb oldal i végpont IP-címét jelöli ki. Szerverünk viszont az ügyfelek számára dinaikusan osztja ki az IP címeket. Ekkor a rögzített IP-cím megadása helyett a *%defaultroute* értéket fogjuk használni, amely a gép alapértelmezett útvonalát jelöli; ez a hálózati kártyához tartozó IP-címmé fog alakulni.

Az alagút *leftid* kapcsolója felesleges ismétlődésnek tűnhet, minthogy a *left*-tel már azonosítottuk a bal oldali IP-címet, ám ez valójában egy kicsit eltér attól. A *leftid* és *rightid* kapcsoló minden alagút végponthitelesítési azonosítóját határozza meg. Ez lehet egy IP cím, de lehet a ”@” jelet követő teljes név (FQDN). Minthogy az *amanda* ügyfél a dinamikus IP-cím kiosztáson keresztül kap címet így a *rightid* számára a @amanda.tilb.sze.hu az egyetlen használható érték. Ebben a példában ugyanakkor a *leftid* számára a @old.tilb.sze.hu és a 192.168.0.1 kölcsönössen felcserélhető értékek.

A *leftsubnet* beállítás azt adja meg, hogy a jobb oldali végpontok milyen cél-IP-címekről érkező csomagokat fogadhatnak el, illetve a jobboldali végpontok milyen célokhoz használhatják az alagutat. Mivel az *amanda* nevű gép az internetet éri el, ezért ezt 0.0.0.0/0-ra állítottuk, ami így az összes címet kifejezi.

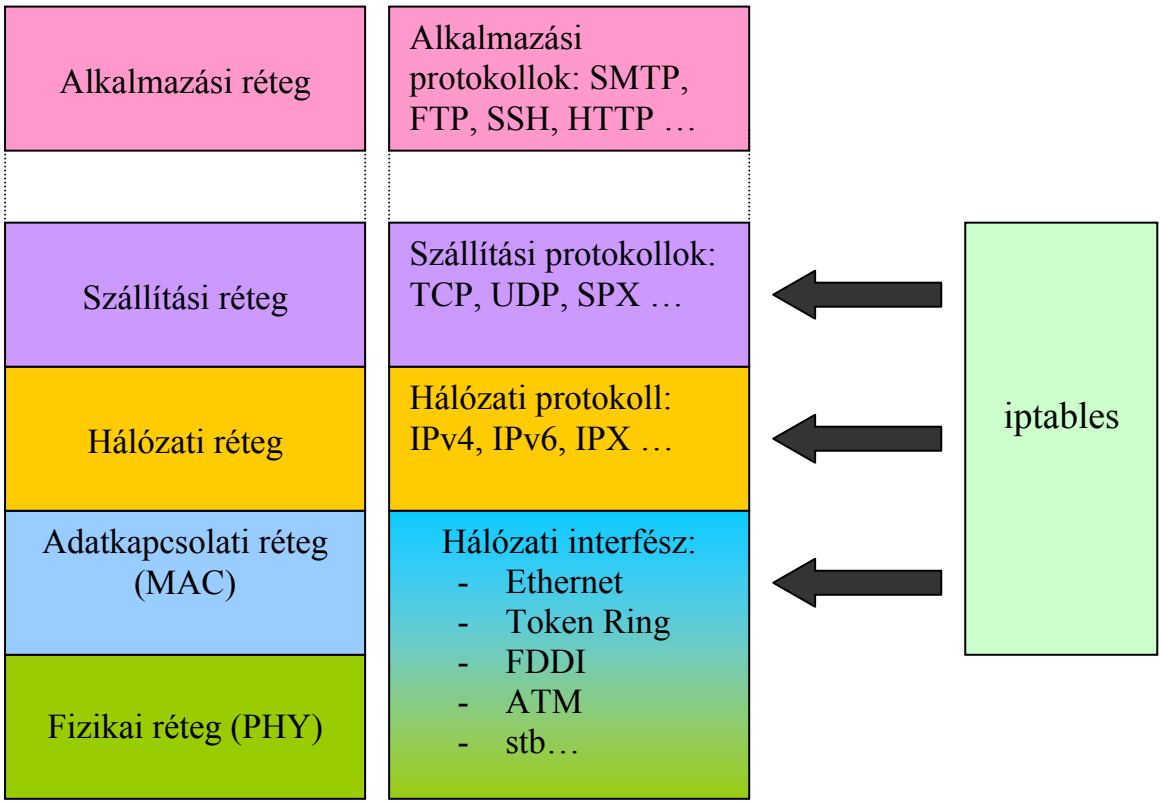
Létezik *rightsubnet* kapcsoló is, ezt azonban kizárólag a telephelyek közti kapcsolatokban szükséges beállítani. Távoli hozzáférés vagy egyéb kiszolgálóalapú megoldásokban a két kapcsoló közül csak az egyik beállítása szükséges.

Ha az alagút számára RSA-hitelesítést ítunk elő, akkor mind a leftrsasigkey, mind a rightrsasigkey megadása kötelező! Ezek az értékek az ipsec.secrets állományban, a #pubkey = kezdetű sorokban találhatók meg.

Az IPsec irányítására (Debian Linux alatt) a /etc/init.d/ipsec script használható.

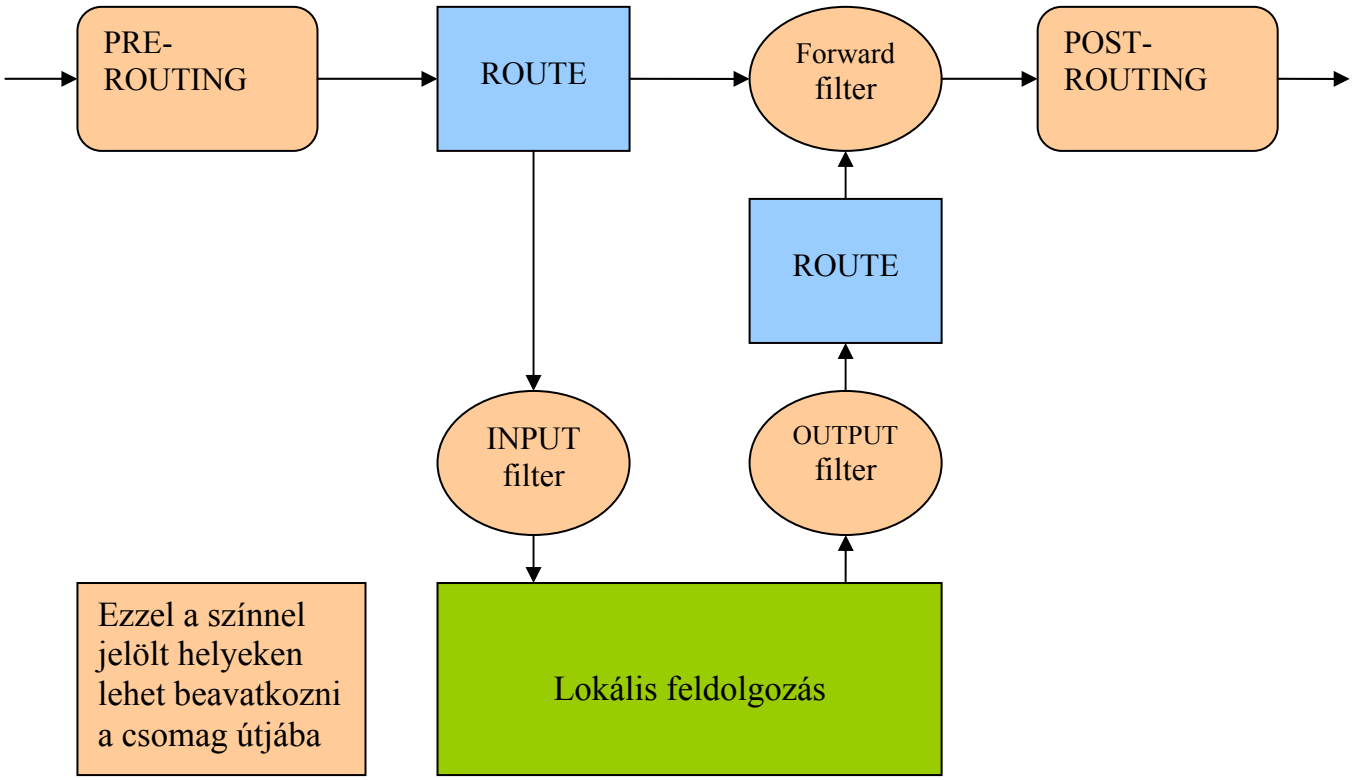
9 Netfilter

A Linux fejlődése során a kernel-be építették, és még ma is fejlesztik a hálózati forgalom figyelését, és irányítását. A 2.0.x kernelverziók esetén *ipfwadm* paranccsal lehetett úgynevezett *FORWARD* azaz csomagtovábbítási paramétereket adni. A 2.2.x verzió megjelenésével az *ipchains* már újabb funkciókat tudott az új kernelből kicsikarni. Ennek előnye, hogy könnyedén lehetett paraméterezni egy csomag útját, hátránya viszont, hogy átláthatatlanná, és kezelhetetlenné vált sok paraméter esetén.
A 2.4.x verzió megjelenése hozta az *iptables* megoldást. Az *iptables* segítségével egyszerűen lehet a csomagokat kezelni, és sok paraméter esetén is könnyen átlátható, állítható, módosítható, és elmenthető a konfiguráció.
Az *iptables* parancson kívül az *iptables-save* az éppen futó konfigurációt jeleníti meg a standard outputra. Így könnyedén elmenthetjük konfigurációnkat pl.: *iptables-save > my_ip_config*. Az *iptables-restore* paranccsal viszont könnyen visszaállíthatjuk elmentett beállításainkat pl.: *iptables-restore < my_ip_config*.
Vizsgáljuk meg, hogy az *iptables* OSI modell szerint milyen rétegekben tevékenykedik.



11. ábra: Az iptables hatásköre az OSI modell szerint

Mint azt később látni fogjuk az *iptables* egészen az adatkapcsolat rétegtől szállítási rétegig tudja befolyásolni a csomagok útját. Ahhoz, hogy az *iptables* hatékony legyen engedélyezni kell kernel szinten az IP csomag továbbítást, más néven a kernel routing-ot: *echo "1" > /proc/sys/net/ipv4/ip_forward.*



12. ábra: IP routing a 2.4.x kernelekben

9.1 Csomagszűrés működése és megvalósítása

A csomagszűrő (*packet filter*) figyeli a csomagok fejlécét miközben azok keresztül haladnak rajta és eldönti az adott csomag további sorsát. Ez lehet *DROP* mely a csomag eldobását jelenti, *ACCEPT*, mely a csomag elfogadását jelenti (hagyja továbbhaladni), és *QUEUE* azaz csomagkésleltetés. Az IP *QUEUE* egy kísérleti stádiumban lévő funkció. A kernel a csomagot egy *QUEUE* listában tárolja, majd innen a listából a megfelelő programok (pl.: forgalom korlátozó) kiveszik, és döntési mechanizmusuk szerint továbbítják egy részét, a többit eldobják.

Az *iptables* egy sor különböző funkciókkal rendelkezik (8.ábra). Három beépített lánccal indul, az *INPUT*, *OUTPUT*, és a *FORWARD* láncokkal, melyeket nem lehet

törölni. A láncok alapértelmezett módja az *ACCEPT*, ami azt jelenti, hogy minden csomag áthaladhat bármelyik táblán.
Nézzük a lehetséges műveleteket a láncokkal:

- Új lánc alkotása (–N)
- Üres lánc törlése (–X)
- Irányelv megváltoztatása beépített láncon (–P)
- Egy lánc szabályainak listázása (–L)
- A lánc összes szabályának törlése (–F)
- A csomag és byte számlálók nullázása a lánc valamennyi szabályában (–Z)
- Új szabály hozzáfűzése a lánchoz (–A)
- Új szabály beszúrása a láncba adott pozíción (–I)
- A adott pozíción lévő szabály cseréje újjal (–R)
- Adott pozíciójú szabály törlése a láncból (–D)
- Az első, erre illeszkedő szabály törlése a láncból (–D)

9.2 Műveletek egy egyszerű szabályon

A csomagszűrés alapja: szabályok alkotása és módosítása. A leggyakrabban a szabály hozzáfűzése (–A) és a szabály törlése (–D) parancsokat fogjuk használni. A többi parancs (–I a beszúrásra és –R a cserére) egyszerű kiterjesztése ezeknek.
Minden szabály meghatároz bizonyos tulajdonságokat, melyeknek illeszkedniük kell a csomagra, és persze meghatározza azt is, hogy mit kell tenni a csomaggal, ha a tulajdonság illeszkedik (ez a szabály ”célpontja, *TARGET*”). Például tételezzük fel, hogy minden a 127.0.0.1 címről érkező ICMP csomagot el akarunk dobni. Ez esetben a tulajdonságok közül kettőnek kell illeszkednie: a protokollnak ICMP-nek kell lenni, a csomag forráscímének pedig a 127.0.0.1-nek. A szabály célpontja pedig a ”*DROP*” lesz. A 127.0.0.1 a visszacsatolt interfész (*Loopback device*), mely akkor is működik, ha nincs tényleges hálózati kapcsolat. Adatcsomagok generálása a ”*ping*” paranccsal történik. Ez egy 8-as típusú (echo request) ICMP csomagot küld, melyre alapesetben a célállomás egy 0-ás típusú (echo reply) ICMP csomaggal válaszol.

root@pc0:/# ping –c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms

Szűrjük ki a 127.0.0.1 forráscímről érkező ICMP csomagokat, majd kíséreljük meg újra a *ping* parancsot.

root@pc0:/# iptables –A INPUT –s 127.0.0.1 –p icmp –j DROP
root@pc0:/# ping –c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss

Láthatjuk, hogy az első *ping* eredményes volt (a ” –c 1” azt jelenti, hogy csak 1 csomagot küld). Azután hozzáfűztük (–A) az ”INPUT” lánchoz egy szabályt, mely azt mondja, hogy a 127.0.0.1 forráscímről érkező (–s 127.0.0.1) ICMP protokollal rendelkező (–p icmp) csomagokat dobja el (–j drop). A második ping használatával leteszteltük, hogy érvényes e a szabály.
Két módon tudjuk a szabályokat törölni a láncból. Először is, mivel tudjuk, hogy ez az egyetlen szabály az *INPUT* láncban, használhatunk számozott törlést, mégpedig:
root@pc0:/# iptables –D INPUT 1
Ezzel az *INPUT* lánc első szabályát töröltük. A második mód a –A (hozzáfűzés) parancs tükörképe, de a –A parancsot –D -re cseréltük. Ezt akkor használhatjuk, ha komplexebb láncunk van. Ez esetben használjuk így:
root@pc0:/# iptables –D INPUT –s 127.0.0.1 –p icmp –j DROP

A –D szintaxisának pontosan egyeznie kell a –A (vagy –I, –R) szintaxisával. Ha több erre a szabályra illeszkedő szabály van a láncban, csak az első fog törlődni.

9.3 Forráscím és célcím meghatározása

Láthattuk, hogy a –p kapcsoló határozza meg a protokollt, és a –s való a forráscím meghatározására. Vannak azonban további opciók, melyekkel pontosabban meghatározhatjuk a csomagokat.

A forrás (–s, vagy –source, vagy –src) és a cél (–d, vagy –destination, vagy –dst) IP címeit négy módon adhatjuk meg. A legnépszerűbb a teljes domain név megadása, mint például a rs1.sze.hu . Második mód az IP cím megadása pl.: 193.224.128.1.

A harmadik és negyedik móddal az IP címek csoportjait tudjuk megadni. Pl.: 193.225.150.0/24 vagy másképpen: 193.225.150.0/255.255.255.0. Mindkettő meghatározás a 193.225.150.0 – 193.225.150.255 –ig terjedő IP címeket határoz meg. A ”/” jel utáni szám a netmaszkot jelenti. A ”24” például azt jelenti, hogy a netmaszkban 24 darab egymást követő bináris jelentésű 1-es van a maradék 0. Tehát a /24 egyenlő 255.255.255.0 –val, csak rövidítve fejezi ki.

Ha minden IP címre érvényes szabályt akarunk, használjuk a /0 meghatározást. Pl.:

root@pc0:/# iptables –A INPUT –s 0/0 –j DROP

Felhívnám a figyelmet, hogy a –s 0/0 teljesen felesleges, elég –A INPUT –j DROP használata!

Néhány kapcsoló, közöttük a ”–s” és a ”–d” flagek tartalmazhatják argumentumukat ”!” előtaggal is. Ez a logikai tagadásnak felel meg. Ez minden olyan csomagot meghatároz, amely az adott feltételnek nem felel meg. Pl.: -s !127.0.0.1 minden csomagra illeszkedik, amely nem a localhost-tól érkezik.

9.4 Protokoll meghatározása

A protokollt a –p (vagy a –protocol) kapcsolóval határozhatjuk meg. A protokoll lehet egy szám is (ha ismerjük a protokoll-számot) vagy névvel, mint például TCP, UDP,

ICMP. Kis és nagybetű nem számít. A protokoll elé is tehetünk ”!”-t, ami az illeszkedést megfordítja. Tehát a –p ! TCP minden olyan protokollra vonatkozik ami nem TCP.

9.5 Interfész meghatározása

A ”*–i*” (vagy –in-interface) és a ”*–o*” (vagy –out-interface) kapcsolók egy interfész nevével való egyezést határoznak meg. Az interfész fizikai eszköz, melyen keresztül a csomag bejön ”*–i*” vagy kimegy ”*–o*”. Az *INPUT* láncra érkező csomagoknak nincs kimeneti interfészük ”*–o*”, ezért az *INPUT* láncon ilyen szabály semmilyen csomagra nem fog illeszkedni. És hasonló képen az *OUTPUT* láncon kimenő csomagok bemeneti interfésszel nem rendelkeznek ”*–i*” ezért ezen a láncon a ”*–i*” kapcsolókkal meghatározott szabályokra nem fog illeszkedni a csomag. Csak a *FORWARD* láncon átfutó csomagok rendelkeznek mind kimeneti, mind bemeneti interfésszel.
Ha jelenleg nem működő interfészre határozunk meg szabályt, az helyes szabály lesz, csak akkor lép érvénybe, ha felkonfiguráljuk az interfészt. Ez igen jól használható például ppp-s interfészek esetén (telefon, adsl modem, általában *ppp0*).
Egy speciális opció, ha az interfész név után ”+” jelet teszünk az valamennyi interfészre illeszkedni fog (függetlenül attól, hogy az interfész fel van-e konfigurálva, vagy sem) melynek a neve a + előtti szöveggel kezdődik. Például egy olyan szabály, amely az összes Ethernet interfészre illeszkedni fog a ”*–i eth+*” kapcsolóval használhatjuk. Az interfész elé ”!” –t téve az összes csomagra illeszkedni fog, melyek nem az adott interfészre érkeznek, vagy nem az adott interfészről távoznak.

9.6 Töredékek meghatározása

Néha egy csomag túl nagy ahhoz, hogy egyszerre továbbítsuk. Ez esetben a csomagot töredékekre bontjuk és több különböző csomagban küldjük el. A végponton ezek összeállnak, és újból rendelkezésünkre áll az eredeti csomag. Ezzel a töredékekkel az a baj, hogy csak az első csomag tartalmazza a komplett fejlécmezőket (IP+ TCP, UDP, és ICMP) a többi csomag pedig csak egy kivonatát a fejlécnek (IP

hozzáadott protokoll mezők nélkül). Ezért ezeknek a csomagoknak a fejlécét nem tudjuk protokoll szempontjából vizsgálni.

NAT (Network Address Transalation, azaz hálózat cím fordítás) használatával kapcsolódunk a hálózathoz a csomagok újból összeállnak, mielőtt elérnék a csomagszűrőt. Ez esetben tehát nem kell a töredékektől tartani.

Minden más esetben fontos megérteni, hogy hogyan bánnak a csomagszűrők a töredékekkel. Minden olyan szabály, mely létező tulajdonságot vizsgál nem illeszkedőnek tekintendő. Ez azt jelenti, hogy – mivel az első csomagot ugyanúgy kezeli a csomagszűrő, mint a többit – a második és további töredékeket a csomagszűrő nem tudja vizsgálni. Így a szabály *(-p TCP –sport www*, forrás port meghatározása www-ként *"-sprot www*" azaz 80 port) sosem fog illeszkedni a töredékre (legalábbis az elsőtől különbözőre). Csakúgy, mint az ellentétes szabály *(–p tcp –sprot ! www*) sem. Mégis meg tudunk határozni szabályt a második és azt következő töredékekre a "–f" (vagy "–fragment") kapcsoló segítségével. Általában biztonságosnak tekintjük a második és további töredékek átengedését a csomagszűrőn, mivel a szűrés az első töredék alapján is egyértelműen meghatározza a csomag sorsát. Ennek ellenére vannak olyan hibák, melyek kihasználásával a töredékek alkalmasak lehetnek számítógépek feltörésére.

Példa: a következő szabály valamennyi 192.168.1.1 IP címre érkező töredéket eldob.

root@pc0:/# iptables –A OUTPUT –f –d 192.168.1.1 –j DROP

9.7 Kiterjesztések az *iptables*-höz: új illeszkedések

Az *iptables* kiterjeszthető, azaz mind a kernel, mind az *iptables* alkalmazás alkalmassá tehető újabb tulajdonságok vizsgálatának ellátására. A kernel kiterjesztések általában a kernel modulok alkönyvtárban találhatók (ha le lettek fordítva), mint például a */lib/modules/2.4.21/kernel/net/ipv4/netfilter* és a *modprobe* paranccsal tölthetjük be őket.

TCP kiterjesztések. A tcp kiterjesztések automatikusan betöltődnek amint a "*–p tcp*" ezt meghatározza. Ez a következő opciókat teszi lehetővé: (a töredékek kezelésének kivételével)

--tcp-flags esetleg ”!”-el utána majd két szöveg a flagekről lehetővé teszik, hogy speciális vizsgálatokat végezzünk egy TCP csomagon. Az első szöveg a flagekről a maszk: egy lista azokról a flagekről melyeket vizsgálni akarunk. A második szöveg a flagekről megmondja, melyik hogy legyen beállítva. Pl.:
root@pc0:/# iptables –A INPUT –p tcp –tcp flags ALL SYN,ACK –j DROP

Ez azt jelenti, hogy valamennyi flaget vizsgálunk (ALL ugyanazt jelenti, mint a SYN, ACK, FIN, RST, URG, PHS), de csak a SYN és az ACK flag lehet beállítva. Van még egy beállítási lehetőség: ”NONE” azt jelenti, hogy egy flag sincs beállítva.

--syn esetleg ”!”-el előtte, ez a rövidítés ugyanazt jelenti, mint a ”-tcp-flags SYN,RST,ACK SYN”.

--source-port esetleg ”!”-el utána majd vagy egy port, vagy egy TCP port tartomány. A portok megadhatók számmal vagy névvel, ahogy az a /etc/services fájlban le van írva. A tartomány két port név vagy port szám, egy ”:”-al elválaszva, vagy ha a porttal egyenlő vagy annál nagyobb portokat akarjuk megváltoztatni a port neve ”:”-al utána.

--sport ugyan az, mint a *--source-port*

--destination-port hasonló a feljebb, csak ez a csomag célját határozza meg

--dport ugyan az, mint a *--destination-port*

Néha hasznos, ha egyik irányban engedélyezzük a TCP forgalmat, a másik irányba pedig nem. Például ha el akarunk fogadni csomagokat egy külső szerver felől, de nem akarjuk, hogy a szerverről hozzánk kapcsolódjanak. Elsőre azt gondolnánk, elég blokkolni a szerverről felénk érkező TCP csomagokat. Azonban sajnos a TCP kapcsolatok működéshez mindkét irányban adatforgalmat igényelnek. A megoldás az, ha csak azokat a csomagokat blokkoljuk, melyek kapcsolat igénylésére szolgálnak. Ezeket a csomagokat SYN csomagoknak hívjuk. Ezen csomagok tiltásával meg tudjuk akadályozni a gépünkre való kapcsolódást. Példa: Tiltsunk le minden 192.168.1.1 –ről érkező kérelmet.

root@pc0:/# iptables –A INPUT –p TCP –s 192.168.1.1 --syn –j DROP
Ez a flag invertálható ha ”!”-t teszünk elé, ez minden olyan csomagra érvényes lesz, amely nem kapcsolat kezdeményezésére indul.

UDP kiterjesztések. Ezek a *–p udp* esetén automatikusan betöltődnek. *--sport, --dport* kiterjesztések hasonlóképpen működnek, mint TCP kiterjesztések esetében.

ICMP kiterjesztések. Ezek a kiterjesztések is automatikusan betöltődnek *–p icmp* esetén.
--icmp-type megadhatjuk az ICMP fajtáját névvel vagy akár a számával. Ez a mód is invertálható ”!”-el előtte.

9.8 MAC cím alapján való vizsgálat

Felmerülhet az a lehetőség, hogy az általunk üzemelt hálózatban vannak olyan gépek is, melyek számára nem akarunk routolni. Ezt ha IP cím alapján szeretnénk megtenni, könnyen kijátszható, mert csak az IP címet kell átállítani a gépen. Az iptables képes MAC cím szerint a kereteket, és az azokban utazó csomagokat vizsgálni. Ehhez a ”*–m mac*” kapcsoló szükségeltetik. A MAC kiegészítőt automatikusan betölti az iptables (amennyiben a *socket filtering* le van fordítva).

--mac-source forrás MAC címet lehet kijelölni
--mac-destination célzott MAC címet lehet kijelölni

Egy példa: Valósítsuk meg, hogy a belső hálózatunkon (eth1 csatoló) csak a 192.168.0.2 IP címről, és a 00:36:11:22:33:44 MAC című gépről fogadjon el csomagot a router.

root@pc0:/# iptables –A INPUT –i eth1 –s 192.168.0.0/24 –j DROP
Alapesetben minden kérést a 192.168.0.0/24 hálózatról eldob.

root@pc0:/# iptables –A INPUT –s 192.168.0.2 –m mac –mac-source 00:36:11:22:33:44 –j ACCEPT

A 192.168.0.2 IP című és 00:36:11:22:33:44 MAC című géptől fogadja a kéréseket.

9.9 Belső hálózatok routolása, NAT megvalósítása

A NAT jelentése: Network Address Transalation. Azaz címfordítást végzünk az IP csomagokban.

A datagramm cím mezői (forráscím) eredetileg nem változnak, a protokollok ezt is várják el. Miért van szükség rá?

Mert elegendő egy IP címet megrendelni, mégis több gépen tudunk internetezni. Ez úgy lehetséges, hogy a külső hálózatra rátesszük a routerünket, és a belső hálózaton a többi gép belső IP címeket kap (ilyenek a 10.x.x.x, vagy a 192.168.x.x IP tartományok). Ezek a belső IP tartományok nem route-olhatóak.

Megvalósítások:

- MASQUERADE (2.2.x kernel)
- NAT (2.4.x kenel)

Másik esetleges probléma, hogy a szolgáltatásaink különböző szervereken futnak, viszont mi csak egy IP címre fizettünk elő, ilyenkor a szolgáltatás portját továbbítani lehet az adott szerver belső IP címe felé. Ezt port forwardnak hívják.

SNAT azaz a Source NAT

- A kimenő forgalom a forrás IP címét cseréljük ki sajátunkra*: iptables –t nat –A POSTROUTING –o eth0 –j SNAT --to 10.0.0.2*
- Lehet több címet is megadni: *--to 10.0.0.2 10.0.0.8*
- Lehet portok alapján is: *iptables –t nat –A POSTROUTONG –o eth0 –j SNAT --to 10.0.0.2:1-1023*
- Pl.: Modemes kapcsolat megosztássa NAT és MASQUERADE funkciókkal: *iptables –t nat –A POSTROUTING –o ppp0 –j MASQUERADE*

- Ha változik az internetre kapcsolódó interfész (pl.: a modemes kapcsolat csak tartalék ha esetleg leáll a fővonal): *iptables –t nat –A POSTROUTING –s 10.0.0.0/24 –j MASQUERADE* . Ilyenkor a 10.0.0.0 – 10.0.0.255 IP címekre NAT-ol a routerünk.

DNAT azaz a Destination NAT

- Továbbítsa a router a 10.0.0.11 IP címre a datagramot: *iptables –t nat –A PREROUTING –j DNAT --to-destination 10.0.0.11*
- Küldjük a WEB kéréseket a 10.0.0.99 IP cím 8080 TCP portjára: *iptables –t nat –A PREROUTING –p tcp --dport 80 –j DNAT --to-destination 10.0.0.99:8080*

9.10 Protokoll segédek

A protokollok azt várják el, hogy a forráscím, és a célcím nem változik, ezért szükség van az ún. protokoll segédekre (protocol helpers). Ezek kernel modulként lefordíthatók, és megtalálhatók a */lib/modules/2.4.21/kernel/net/ipv4/netfilter* könyvtárban, és *modprobe* paranccsal betölthetőek. Ilyen például az FTP ahol két port funkcionál: 21-es port a kommunikációra, a 20-as port az adatátvitelre.

9.11 Tűzfal megvalósítások *iptables* segítségével

- NAT létrehozása a belső hálózatunk felé: *iptables –t nat –A POSTROUTING –s 192.168.0.0/24 –j MASQUERADE*
- Alapesetben minden TCP, és UDP portot szűrjünk ki: *iptables –A INPUT –j DROP*
- Csak azokat a portokat engedélyezzük, amire ténylegesen szükségünk van (pl: HTTP): *iptables –A INPUT –p tcp –dport 80 –j ACCEPT*

9.12 Feltöltési és letöltési sebesség korlátozás

A hálózati forgalom korlátazása többféleképpen lehetséges. A *shaperd*-vel rugalmasan kezelhetjük a kliens gépeink feltöltési és letöltési sebességeit. A *shaperd* Debian csomagként elérhető (*apt-get install shaperd*). A kernelünkhöz (modulként)

szükségeltetik egy fejlesztés alatt álló funkció lefordítása, ez a 6. fejezetben már említett *Userspace queueing via NETLINK* funkció. Továbbá szükséges még az *iptables* és támogatása (2.4.x kernel). A *shaperd* konfigurációs fájlja alap esetben a */usr/local/shaperd/shaperd.conf*. Ezt nyugodtan áthelyezhetjük a */etc* könyvtár alá, de ilyenkor a *shaperd –c /etc/shaperd.conf* –al indítsuk el a daemont. A konfigurációs fájlnak tartalmazni kell az alapvető beállításokat: logolási szint (*log level*), csomagtovábbítás fajtája (*divert* = 2.2.x, *ipq* = 2.4.x kernelek), daemonként való futás, pid fájlnév megadása (*pidfile*).

```
log level = warning
packet forwarding = ipq
daemon = yes
pidfile = /var/run/shaperd.pid
```

Ezek beállítása után tudjuk az ún. *class*–okat megadni. A *class*-okban megadható, hogy milyen forgalmat szeretnénk korlátozni, és hogy melyik IP című gép forgalmát. Itt megadható, hogy mekkora legyen a maximális adatátviteli sebesség (*bandwidth*), illetve letöltési vagy feltöltési korlátozás (*saddr* = forráscímről érkező csomagok, tehát feltöltés, *daddr* = célcímre érkező csomagok, tehát letöltés kliensgépre vonatkoztatva). Megadható, hogy milyen protokollok csomagjaira vonatkozzon a szabály (*IPv4 classifier proto = tcp*) . Illetve megadható a csomagméret, és hogy hány csomagot tároljon le a puffer tárban, így lehetőségünk van maximális csomagméretet is szabályozni (*queue limits*). Nézzünk meg egy példa konfigurációs fájlt, ahol a 192.168.1.20 –as IP című kliensgépünknek TCP (feltöltési, és letöltési) fogalmát korlátozzuk.

```
class client_20_up {
        bandwidth = 128 kbit/s
        ipv4 classifier proto=tcp saddr = 192.168.1.20
        queue limits = 200 kb 100 packets
}
```

class client_20_down {

bandwidth = 384 kbit/s

ipv4 classifier proto=tcp daddr = 192.168.1.20

queue limits = 200kb 100 packets

}

Ha elvégeztük a szükséges konfigurációs fájl módosítását, akkor indítsuk újra a *shaperd* daemont. Majd a bejegyzéshez megfelelő forgalmat irányítsuk a *QUEUE* listába.

root@tilb:~/# killall –HUP shaperd

root@tilb:~/# iptables –A FORWARD –s 192.168.1.20 –p tcp –j QUEUE

root@tilb:~/# iptables –A FORWARD –d 192.168.1.20 –p tcp –j QUEUE

10 Hálózati szolgáltatások UNIX alatt

10.1 Szolgáltatások indítása inetd segítségével

Ahhoz, hogy egy adott szolgáltatást nyújtsunk az Internet felé UNIX segítségével, futtatnunk kell egy olyan programot, mely az adott protokollal rendelkező kérésre válaszolni tud. Például, ha egy WEB szervert (HTTP kiszolgálót) szeretnénk üzemeltetni, állandóan futnia kell a *httpd* nevű programnak, mely figyeli a 80-as TCP porton érkező kéréseket, és képes feltölteni a kliens által kért fájlokat. Régebben nem voltak olyan nagy teljesítményűek a számítógépek, mint napjainkban, ezért UNIX esetében is igyekeztek erőforrás kímélő módon megoldani a szolgáltatások futtatását. A módszer lényege, hogy nem futtatunk minden szolgáltatáshoz külön programot, hanem csak akkor indítjuk el őket, amikor ténylegesen szükség van az adott szolgáltatásra. Ehhez semmi másra nincs szükségünk, csak egy úgynevezett „*szuperszerverre*”, amely az összes, általunk definiált szolgáltatásokhoz tartozó hálózati kérést figyel, és ha szükséges elindít egy daemon-t, ami végül lekezeli a kérést ténylegesen. A szuperszerver neve általában minden UNIX rendszerben: *inetd*. Az általa nyújtott szolgáltatásokat a */etc/inetd.conf* fájlban állíthatjuk be. Lássuk, hogy néz ki ez a fájl:

```
time    stream      tcp    nowait      root    internal
time    dgram       udp    wait        root    internal
```

Ezek az idő beállítására való szolgáltatások, melyek a rendszerórát szinkronizálják az Interneten lévő pontos idővel rendelkező szerverekhez. Ez az *inetd* belső szolgáltatása, nem kell külön program hozzá.

```
ftp     stream      tcp    nowait      root    /usr/sbin/tcpd          proftpd
```

A *proftpd* FTP szerver, melynek további konfigurációját a */etc/proftpd.conf* határozza meg.

telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd

A *telnet* távoli bejelentkezés daemon-ja.

comsat stream udp wait root /usr/sbin/tcpd in.comsat

A *comsat* egy belső szolgáltatás, mely a konzolon jelzi, ha e-mail érkezett.

shell stream tcp nowait root /usr/sbin/tcpd in.rshd –L
login stream tcp nowait root /usr/sbin/tcpd in.rlogind

A Berkley r* szolgáltatások daemonjai.

ntalk dgram udp wait root /usr/sbin/tcpd in.talkd

A *talk* egy interaktív program, Interneten keresztül lehet képernyőn billentyűzet segítségével beszélgetni.

pop3 stream tcp nowait root /usr/sbin/tcpd ipop3d
imap4 stream tcp nowait root /usr/sbin/tcpd ipop3d

Az e-mail-ek letöltésére szolgáló protokollok daemonjai.

finger stream tcp nowait nobody /usr/sbin/tcpd in.fingerd -u

A finger egy információs szolgáltatás, felhasználók adatait tudhatjuk meg a segítségével.

Az első oszlopban álló szolgáltatás-névhez a */etc/services* fájlban vannak hozzárendelve az adott protokoll port számai. Vegyük észre, hogy ha ott a például a POP3 protokollt ezután úgy neveznénk, hogy „kutya”, akkor az *inetd.conf*-ba is kutyát

kell írnunk! Ha szeretnénk megszüntetni egy szolgáltatást, annyit kell tennünk, hogy egy komment jelet („#”) írunk az inetd.conf megfelelő sorába, és újraindítjuk az *inetd* daemon-t:

root@tilb:/# killall –HUP inetd

Az *inetd.conf* hatodik oszlopában található tcpd daemon arra szolgál, hogy megszabhassuk, milyen IP címekről jelentkezhetnek be kliensek, hogy igénybe vehessék a szolgáltatásainkat:

/etc/hosts.allow – Ebben a fájlban azt mondhatjuk meg, kik azok, akik igénybe vehetik a szerver szolgáltatásait.
/etc/hosts.deny – Itt mondhatjuk meg, hogy *kik nem vehetik igénybe* a szolgáltatásokat.
/etc/hosts.equiv – Itt megmondhatjuk, melyek azok a gépek, amelyek bizonyos szolgáltatások (pl: nyomtatás) szempontjából egyenrangúnak számítanak gépünkkel.
/etc/hosts.lpd – Itt megadhatjuk, kik nyomtathatnak a szerverünkre installált nyomtatóra.
A különféle szolgáltatások *inetd*-ből történő indítása rendszerünket nehezen áttekinthetővé és megbízhatatlanná teszi, amihez ezen felül még biztonsági rések is társulnak. Emiatt mostanában a rendszergazdák szívesebben alkalmazzák azt a módszert, hogy minden szolgáltatás számára külön daemon-t futtatnak.

10.2 Szolgáltatások egyedi indítása

Amennyiben precízebben szeretnénk kontrollálni szolgáltatásainkat, egyenként kell elindítanunk a szolgáltatást végző daemon-okat. Ez persze nem azt jelenti, hogy kézzel kell elindítani őket, hanem pl. a rendszerindító szkriptek közül a */etc/init.d/bind* fájl, ami például a DNS szerver indításáért felelős.
Ha kíváncsiak vagyunk, hogy egy adott szolgáltatás (pl.: *named*) fut-e, a *ps* programmal ellenőrizhetjük:

```
root@tilb:/# ps ax |grep named
135   ?        S      0:07   /usr/sbin/named –u  daemon
6707  pts/1    S      0:00   grep named
```

Az, hogy egy adott daemon indításakor mit kell beírni a parancssorba, a program típusától függ, pl.: az NFS szerver szolgáltatásait a */etc/init.d/nfs-kernel-server* szkript segítségével ki-be kapcsolhatjuk.:

```
root@tilb:/# /etc/init.d/nfs-kernel-server start
Starting NFS services:
        /usr/sbin/exportfs –r
        /usr/sbin/rpc.rquotad
        /usr/sbin/rpc.nfsd 8
        /usr/sbin/rpc.mountd --no-nfs-version 3
        /usr/sbin/rpc.lockd
        /usr/sbin/rpc.statd

root@tilb:/# ps ax|grep nfs
6733  pts/1    SW     0:00   [nfsd]
6734  pts/1    SW     0:00   [nfsd]
6735  pts/1    SW     0:00   [nfsd]
6736  pts/1    SW     0:00   [nfsd]
6737  pts/1    SW     0:00   [nfsd]
6738  pts/1    SW     0:00   [nfsd]
6739  pts/1    SW     0:00   [nfsd]
6740  pts/1    SW     0:00   [nfsd]
6745  ?        S      0:00   /usr/sbin/rpc.mountd --no-nfs-version 3
root@tilb:/# /etc/init.d/nfs-kernel-server stop
Stopping NFS kernel daemon: rquotad nfsd mountd lockd statd
Unexporting directories for NFS kernel daemon… done
```

10.3 Szolgáltatások felderítése, rendszerbiztonság

Mint látható, a *ps* program segítségével könnyen ellenőrizhetjük, fut-e egy adott daemon. Ha ehhez hozzávesszük, hogy milyen szolgáltatásokat nyújt az *inetd*, nagyjából képet kapunk arról, hogy milyen szolgáltatásokat nyújt saját szerverünk. Ellenben mi van azokkal a programokkal, amelyek esetleg egyszerre több porton szolgáltatnak, vagy azokkal, amelyeket olyan szkript indít el, aminek a létezéséről nem is tudunk, és pl. holnap fogják vele feltörni a szerverünket? Ezeket csak bizonyos szolgáltatásokat felderítő programok segítségével „fülelhetjük” le.

Az egyik ilyen rendkívül hasznos program az *nmap*, melyet a Debian Linux csomagkészletében is megtalálhatunk, és könnyedén felinstallálhatunk (*apt-get install nmap*).

Grafikus módban a program kimenete színes, és egyből láthatjuk a kritikus beállítási pontokat, nézzük, például mit sikerült okoznunk az *inetd* átkonfigurálásával:

root@pc0:/# nmap –sS -0 pc0.tilb.sze.hu

Starting nmap V. 3.00 (www.insecure.org/nmap/)

Interesting ports on pc0.tilb.sze.hu (193.224.130.170):

(The 1586 ports scanned but not shown below are in state: closed)

Port	*State*	*Service*
21/tcp	*open*	*ftp*
22/tcp	*open*	*ssh*
23/tcp	*open*	*telnet*
25/tcp	*open*	*smtp*
37/tcp	*open*	*time*
79/tcp	*open*	*finger*
80/tcp	*open*	*http*
110/tcp	*open*	*pop-3*
111/tcp	*open*	*sunrpc*
113/tcp	*open*	*auth*
513/tcp	*open*	*login*
514/tcp	*open*	*shell*

515/tcp open printer

587/tcp open submission

6000/tcp open X11

Remote operating system guess: Linux 2.1.19 – 2.2.20

Uptime 0.171 days (since Mon Sep 23 11:30:48 2002)

Nmap run completted -- 1 IP address (1 host up) scanned in 6 seconds

A „State” oszlopban lévő open állapot azt jelzi, hogy szerverünk az adott porton vakon teljesít minden kérést, ezen felül, a színekből látszik, hogy melyik protokollokat tekinti a szoftver különösen veszélyesnek: ftp, ssh, telnet, login, shell. Gépünk tehát a támadási pontok tárháza. Hogy miért éppen ezek a protokollok? Mert ezek zöme (az SSH kivételével) **kódolatlanul küldik a jelszavakat a hálózaton!** Ha az SSH kódolja a jelszavakat és a forgalmat, miért veszélyes mégis? Oka bámulatosan egyszerű: az SSL kódolás nem használ olyan hosszú biztonsági kódot, amely ne lenne megfejthető néhány óra alatt, ezen felül maga az SSH szerver is rendelkezik biztonsági résekkel, melyet bárki megnézhet pl. a www.securityfocus.com weboldalon.
A legnagyobb hiba itt mégis az, hogy minden szolgáltatást engedünk igénybe venni bárhonnan, illetve futnak fölösleges szolgáltatások is, pl. ha van SSH akkor minek telnet, rlogin, és ftp? (Az ftp-t az SSL alatt működő scp programmal helyettesíthetjük, ezt is az sshd szolgálja ki.)
Emellett érdemes kitölteni a */etc/hosts.** fájlokat, hogy ne lehessen mindenhonnan például *telnet*-elni.

Figyelem!!! Az ilyen szolgáltatásokat felderítő programok használatát egyes, forrófejű és hozzá nem értő rendszergazdák támadásnak veszik, ezért idegen IP címeken csak akkor használjuk az nmap-ot, ha biztosak vagyunk benne, hogy nem fogunk lebukni!

10.4 Az SSH protokoll

Az SSH protokoll biztonságos távoli bejelentkezést biztosít UNIX alapú rendszerekbe. Mielőtt neki kezdenénk az *sshd* vizsgálatának, vegyünk néhány alapvető rendszerbiztonsági fogalmat és elméletet.

Kriptográfia – titkosítás, hitelesség

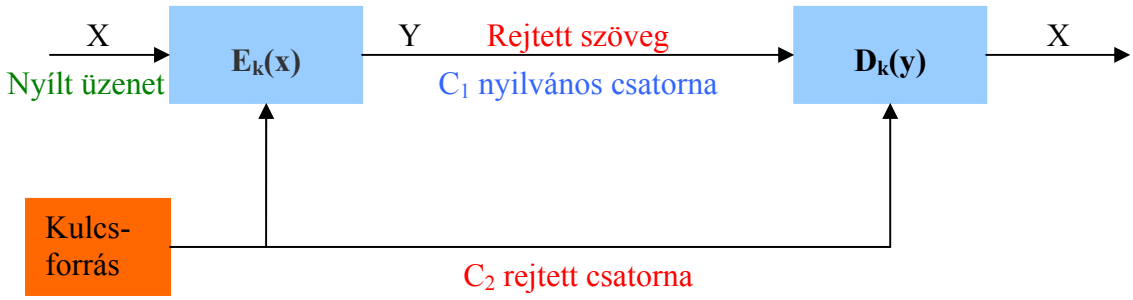
Kriptoanalízis – „feltörés”, illetéktelen hozzáférés

Ezt a két fogalmat átfogóan *kriptológiának* nevezzük.

Érzékeny információ – Olyan adat, amely eltulajdonítása vagy megsemmisítése nagy anyagi vagy erkölcsi kárt okoz a tulajdonosának.

Léteznek olyan protokollok, amelyek nyilvános csatornán kódolatlanul küldik a jelszavakat (pl.: telnet, ftp, pop3) és adatokat (pl.: http, imap4).

Ezen protokollok használatával egy támadó könnyen megszerezheti felhasználóink jelszavait.



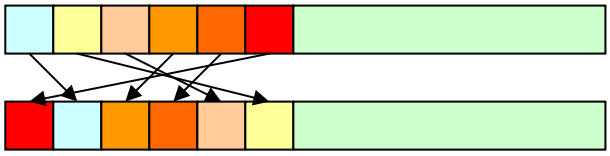
13. ábra: A kódolt kommunikáció konvencionális modellje

A támadó célja „k” kulcs és „X” üzenet megszerzése. A támadó a „k” kulcs kivételével az E( ) és D( ) függvényt ismeri, így kódkönyv használatával, kimerítő kereséssel feltörheti a kódolt üzenetet.

Néhány kódolási algoritmus:

Permutációs kódolás. Levelezéseknél alkalmazták, a betűk (vagy blokkok) sorrendjét egy algoritmus szerint felcserélték.

A	B	C	…	Z
D	H	G	…	E



14. ábra: Szöveg és blokk permutációs kódolása

Caesar titkosító.

A → 0 $\underline{X}$ (X_1, …, X_m) → Üzenet

B →1 $\underline{Y}$ (Y_1, …, Y_m) → Kódolt üzenet

Z → 26 $Y_i = (X_i + k_i) \bmod Z$

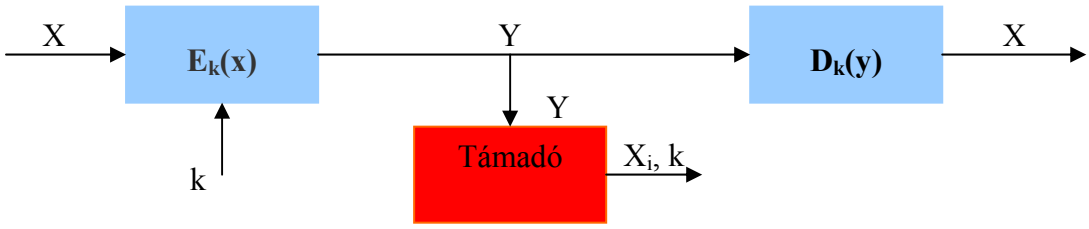
ahol „k_i” a kulcs.

Vernam titkosító. A Vernam titkosító a logikai kizárólagos vagy alkalmazásával titkosít. A beadott bináris számsorozatot *XOR*–olja a kulccsal. Visszaalakításnál elegendő újra XOR-olni a kulccsal a titkosított bináris számsorozatot, és máris visszakapjuk az eredeti számsorunkat.

X = 01000001
XOR
k = 11100110
Y = 10100111
XOR
k = 11100110
X = 01000001

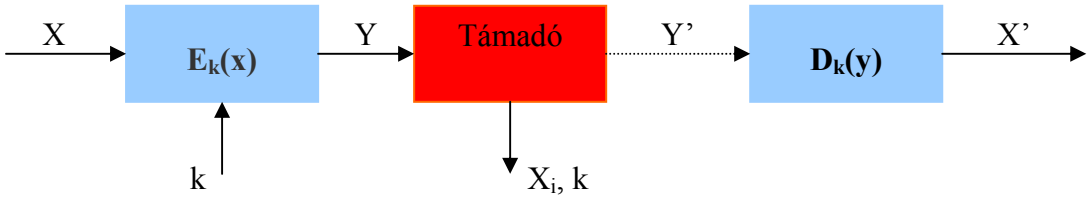
Algoritmikus támadások fajtái.

Passzív támadás (15. ábra): a támadó lehallgatja a nyilvános csatornán haladó kódolt üzeneteket.



15. ábra: Passzív támadás modellje

Aktív támadás (16. ábra): a támadó megszemélyesíti a címzettet, módosítja az üzenetet, majd visszajátssza a módosított üzenetet. Az aktív támadás fajtái: üzenetmódosítás, megszemélyesítés és visszajátszás.



16. ábra: Aktív támadás modellje

Akkor mondhatjuk, hogy az algoritmust feltörték, ha kulcs csere ellenére is gyorsan* megszerezhető Y –ból X.

Gyakorlati titkosság jelentése, hogy a feltörés túlságosan sok számítási kapacitást igényel (RC5→kb. 10000db Pentium II. kategóriájú számítógép a 64bit-es Vernam titkosítást 2 év alatt törte fel.)

** Gyorsan jelentése: a támadó fel tudja használni a szerzett információt*

Kriptográfiai protokollok. A Klasszikus kriptográfia célja a titkok megőrzése passzív támadásokkal szemben. A modern kriptográfiai protokollok védenek az aktív és passzív támadásokkal szemben is! Ezen protokollok fel vannak vértezve a következő tulajdonságokkal:

- integritás védelem
- partnerazonosítás
- lehalgathatatlanság
- kulcsgondozás*
- visszajátszás elleni védelem

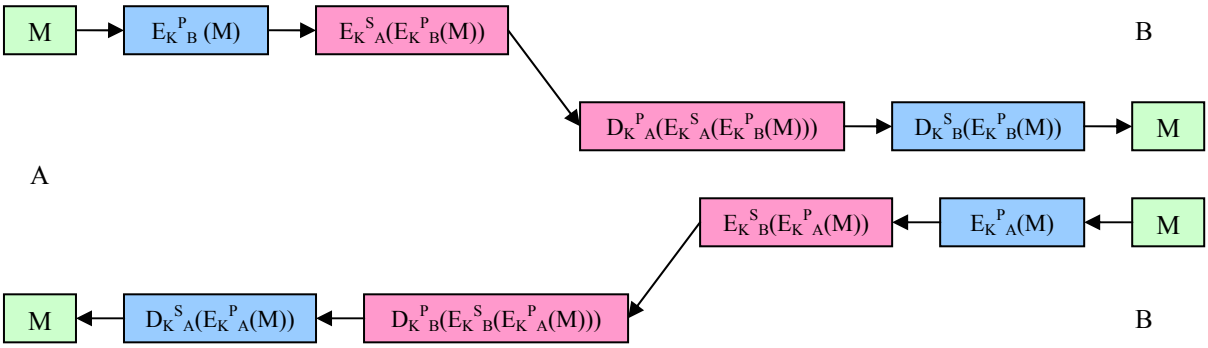
* Kulcsgondozás, másnéven kulcs management elemei:

- kulcsgenerálás
- kulcstárolás
- kulcstovábbítás, elosztás

Nyilvános kulcsú titkosítás.

Két kulccsal rendelkezem (K^p_a→nyilvános és K^s_a→titkos) A nyílvánosat közzéteszem egy nyilvános kulcstárba (léteznek ilyen kulcsszerverek, bővebb információ http://www.iif.hu). Az üzenetet küldő személy leszedi a nyilvános kulcsot, és egy csapda típusú egyirányú függvénnyel bekódolja az üzenetet a kulccsal. A csapda típusú egyirányú függvény sajátossága, hogy kiszámítása gyors, de visszafelé irreálisan lassú. A titkos kulcsommal viszont gyorsan visszafejthetem az üzenetet.

Az RSA titkosítás szimmetrikus, így a nyílvános kulccsal kódolt üzenet visszafejthető a titkos kulccsal, és visszafelé a titkos kulccsal kódolt üzenet visszafejthető a nyilvános kulccsal. Hogy miért is jó ez, azt a következő példa szemlélteti. A következő példa a megszemélyesítés ellen nyújt védelmet.

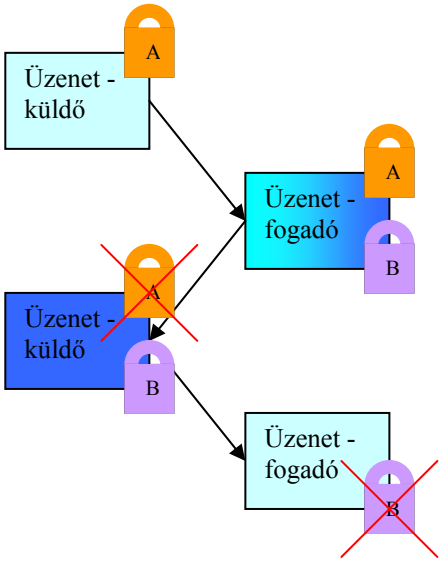


17. ábra: Nyílvános kulcsú titkosítás, megszemélyesítés védelemmel

Vizsgáljuk meg a 17. ábrát: M az átvinni kívánt üzenetet jelenti. A baloldal és jobboldal is rendelkezik egy nyílvános és titkos kulccsal (K_A^P=A nyílvános kulcsa, K_B^P=B nyílvános kulcsa, K_A^S=A titkos kulcsa, K_B^S=B titkos kulcsa, E() a kódoló, míg D() a dekodoló függvényt jelenti). A elkódolja B nyílvános kulcsával az üzenetet, majd a végeredményt elkódolja A titkos kulcsával. A az átviteli csatornára a kétszeresen elkódolt üzenetet adja ki. B visszafejti A nyilvános kulcsával az első kódolást, így biztos lehet benne, hogy A-tól jött az üzenet, mert A-nak a titkos kulcsát nem ismeri senki más. Majd a második kódot B titkos kulcsával könnyedén vissza lehet fejteni. A visszaüzenés is hasonló módszerrel működik.

Három utas üzenetváltással, nyilvános és előzetes kulcs csere nélküli titkosított üzenetátvitel.

Az ilyen típusú üzenetátvitel úgy oldható meg, hogy mindkét fél rendelkezik egy titkos kulccsal. Az üzenet küldő bekódolja az üzenetet a titkos kulcsával, elküldi. A címzett a kapott kódolt üzenetet bekódolja az ő titkos kulcsával is, majd visszaküldi. A válaszként kapott kétszeresen kódolt üzenetet az eredeti üzenetküldő visszakódolja a kulcsával, majd ismét elküldi a címzettnek. Így a címzett már dekódolni tudja, hiszen már csak az ő titkos kulcsával van kódolva az üzenet. Az ilyen kommunikáció viszont kijátszható megszemélyesítéssel. A megértést segíti a 18. ábra.



18. ábra: Három utas titkos üzenetátvitel

Az ilyen fajta üzenetátvitelhez a hatványozás tulajdonságain alapuló kódolást (RSA = Rivest-Shamir-Adleman) használják: $(X^d)^e = X^{de} = [X^e]^d$

Az sshd konfigurációja. Az ssh konfigurációs fájlja a */etc/ssh/sshd_config* alatt található.

drmomo@tilb:~$ more /etc/ssh/sshd_config
 # Package generated configuration file
 # See the sshd(8) manpage for defails

Kommunikációs port

Port 22

Titkosítási protokollok, amit az sshd használ. (verzió: 1, 2)

Protocol 2,1

Hosztkulcsok helye az egyes verziójú protokollokhoz.

HostKey /etc/ssh/ssh_host_key

RSA hosztkulcsok helye.

HostKey /etc/ssh/ssh_host_rsa_key

HostKey /etc/ssh/ssh_host_dsa_key

… nagyobb biztonság érdekében alapértelmezetten engedélyezve van.

UsePrivilegeSeparation yes

 # ...but breaks Pam auth via kbdint, so we have to turn it off

 # Use PAM authentication via keyboard-interactive so PAM modules can

 # properly interface with the user (off due to PrivSep)

PAMAuthenticationViaKbdInt no

 # Lifetime and size of ephemeral version 1 server key

Élettartama és mérete az egyes verziójú szerverkulcsnak

KeyRegenerationInterval 3600

ServerKeyBits 768

Logolás, logolási szint

SyslogFacility AUTH

LogLevel INFO

Azonosítással kapcsolatos beállítások (türelmi idő a csatlakozáshoz, ne engedjen root-ként belépést, …).

LoginGraceTime 600

PermitRootLogin no

StrictModes yes

RSA azonosítás

RSAAuthentication yes

RSA publikus kulcsok azonosítása

PubkeyAuthentication yes

RSA titkos kulcsok helye a hoszt gépeken

#AuthorizedKeysFile %h/.ssh/authorized_keys

Rhost azonosítás, alapesetben kikapcsolva van.

RhostsAuthentication no

 # Don't read the user's ~/.rhosts and ~/.shosts files

IgnoreRhosts yes

 # For this to work you will also need host keys in /etc/ssh_known_hosts

RhostsRSAAuthentication no

RhostsRSAAuthentication no

 # similar for protocol version 2

HostbasedAuthentication no

 # Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication

#IgnoreUserKnownHosts yes

 # To enable empty passwords, change to yes (NOT RECOMMENDED)

PermitEmptyPasswords no

 # Uncomment to disable s/key passwords

#ChallengeResponseAuthentication no

 # To disable tunneled clear text passwords, change to no here!

PasswordAuthentication yes

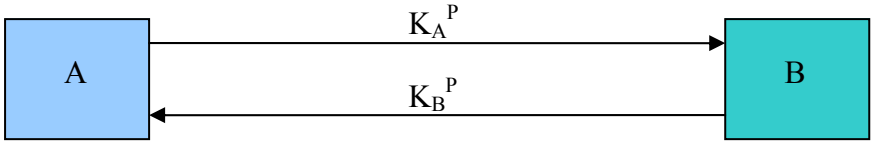
A */etc/init.d/ssh [start|stop|reload|force-reload|restart]* opciókkal irányíthatjuk az sshd futását.

Pár szó az ssh-ról. Az OpenSSH project lehetővé tette számunkra az ssh protokoll ingyenes használatát. Az ssh-nak két ismert verziója létezik: v.: 1.x, 2.x, de ezek nem összetévesztendőek az ssh protokoll verziókkal! (SSH Protocol v.1 = RSA, SSH Protocol v.2 = RSA/DSA)

Azonosítás folyamata:

- Session key (gépkulcs csere)

A eljuttatja B-nek a publikus kulcsát, és B visszaküldi A-nak a saját publikus kulcsát.



19, ábra: Session key

Majd mindkét fél generál egy véletlen számot (Unix rendszerekben a beépített véletlen szám generátor gondoskodik erről). A fél elkódolja ezt a véletlen számot B publikus kulcsával, majd átküldi B-nek. B fél is elkódolja ezt a váletlen számot A publikus kulcsával, majd átküldi A-nak. Mindkét oldalon előáll a $K_{SESSION}=f(r_a , r_b) = f(r_b , r_a)$. Ezután következik az authentikáció. Az authentikálás két módon futhat le. Az a, esetben erős (nyílvános kulcsú) titkosítással egy már előzetesen egyeztettet és megfelelő helyre beillesztett kódok alapján, illetve b, esetben a hagyományos Unix-os jelszó megadása során.

a, eset: Az ssh-keygen programmal készítünk egy publikus és egy titkos kulcsot. A szerver oldalon elhejezzük a publikus kulcsunkat a home könyvtárunk .ssh könyvtára alá (~/.ssh/[authorized_keys|authorized_keys2] – függően attól, hogy milyen ssh_protokollt használtunk). A titkos és a nyílvános kulcsot (Unix esetén) a kliens gépünk home könyvtárába szintén a .ssh könyvtár alá helyezzük el (~/.ssh/id_[rsa|dsa], ~/.ssh/id_[rsa|dsa].pub). Itt van lehetőségünk a titkos kulcsunkat elkódolni egy jelszóval. Ilyenkor a belépésnél ezt a jelszót kell megadnunk ahhoz, hogy kikódoljuk a titkos kulcsunkat. Soha ne felejtsük el az id_[rsa|dsa] fájlunkra 600 jogot adni!

Programok az ssh-hoz. Az ssh nem csak biztonságos távoli bejelentkezést tud nyújtani számunkra, hanem titkosított adatátvitelt két fél között. Unix rendszerekben erre szolgál az scp parancs. Szintaktikája megegyezik a már tanult Berkley r* rcp paranccal. Pl.:

root@teacher:~/# scp kep.jpg lencse@tilb.sze.hu:~/pic001.jpg

10.5 FTP szerver: *proftpd* és konfigurációja

A *proftpd* egy eléggé elterjedt FTP szerver program a Linux rendszerek világában. Debian Linux alá könnyedén felinstallálható: *apt-get install proftpd*.

A proftpd konfigurációs fájlja a */etc/proftpd.conf*, felhasználó-regisztrációs file: */etc/ftpusers*. Ebbe a fájlba azoknak a felhasználóknak a nevét kell bejegyezni, akiknek a hozzáférését le akarjuk tiltani.

A proftpd program /usr/sbin könyvtárban található.

Az anonymous hozzáféréshez létre kell hozni egy felhasználót, jelen esetben ftp nevűt, és adni kell neki egy home könyvtárat. Értelemszerűen csak ezt a home könyvtárat lehet majd anonymous –ként érni.

A többi felhasználó, akit a rendszerünkbe nyílván tartunk saját felhasználó nevével, és jelszavával tudja elérni az FTP szolgáltatásunkat.

/etc/proftpd.conf felépítése:

ServerName "Ez az ftp szerverünk neve"

ServerType inetd

A proftpd daemon futtatható a háttérben és inetd alatt is. Ha csak simán a háttérben szeretnénk futtatni akkor *standalone* -t kell megadni.

DefaultServer on

Ez az alapértelmezett szerver, ha esetleg van a rendszerünkön másik FTP szerver program.

Port 21

Adhatunk más portot is, de az alapértelmezett a 21-es TCP port.

Umask 022

Védelem a könyvtárakra, a chmod 022 (csoport, és mindenki által írható) jogokat maszkolja ki FTP alatt.

MaxInstances 30

Ezzel azt lehet beállítani, hogy maximum hány példányban fusson, tehát hány kérelmet lásson el egyszerre. Ez csak standalone módban működik.

User nobody
Group nobody

Milyen felhasználó és csoport jogaival induljon el a szerverünk.

SystemLog /var/log/proftpd.log
TransferLog /var/log/xferlog

Hová logoljon. System log a program futással kapcsolatos információkat logolja, míg a transfer log egy külön fájlba logolja, hogy kik mit szedetek le tőlünk, illetve raktak fel.

<Directory />*
AllowOverwrite on
</Directory>

Az összes file felülírható legyen, ha van rá engedély.

<Anonymous ~ftp>

RequireValidShell *on* csak olyan usereket enged be akinek van érvényes shellje.

User *ftp* milyen user névvel rendelkezzen az anonymous belépés (ezt a felhasználót létre kell hozni a /etc/passwd fájlban.)

Group *ftp* milyen group –hoz tartozzon az anonymous belépés (ezt is létre kell hozni a /etc/group fájlban, ha nincs ilyen csoport.)

UserAlias *anonymous ftp* hozzárendeli az anonymous hozzáférést az ftp felhasználóhoz

DisplayLogin *welcome.msg* üdvözlő fájl megadása (ez a file az anonymous home könyvtárában kell, hogy elhelyezkedjen)

DisplayFirstChdir *.message* minden könyvtárban megjelenő üzenet amit először nyit meg az anonymous felhasználó (ez a file az anonymous home könyvtárában kell, hogy elhelyezkedjen).

<Limit WRITE>	írás jog korlátozás anonymous –ként a gyökér könyvtárra.
DenyAll	alapértelmezetten tiltva van, engedélyezés: AllowAll
</Limit WRITE>	mint a HTML -nél le kell zárni a blokkot.

Ha például szeretnénk egy olyan könyvtárat is csinálni amibe bárki tud feltölteni a következő képen kell eljárnunk(ezt a fájlt létre kell hozni az anonymous home –jában */home/ftp/upload*):

```
<Directory upload/*>
    <Limit WRITE>
        AllowAll
    </Limit WRITE>
</Directory upload/*>
```

Erre a könyvtárra chmod 777 jogokat kell adni. Az anonymous felhasználónak nem lesz joga, hogy könyvtárakat hozzon létere illetve töröljön!

</Anonymous>

Itt ért véget az anonymous hozzáférés konfigurációja.
Minden átkonfigurálás után újra kell indítani a *proftpd* –t a *killall proftpd –HUP* parancsal ha nem *inetd*-ből fut.

10.6 DHCP szerver: *dhcpd* és konfigurációja

A DHCP szerver segítségével IP címeket lehet kiosztani dinamikusan, vagy Ethernet cím alapján. A kiosztás csak egy meghatározott időre szól, azután újra kell igényelni. Ez természetesen teljesen automatikusan zajlik, a felhasználó ebből nem vesz észre semmit. A szerver nem csak IP címek kiosztására szolgál, a hálózathoz tartozó beállításokat is képes átadni a kliens gépnek (hálózat cím, maszk, átjáró, DNS szerver). A protokoll leírása a 2131-es számú rfc-ben található meg.

A konfiguráció. A dhcpd program konfigurációs fájlja a */etc/dhcpd.conf* alatt található. A maximális frissítési (bérleti) idő megadása lehetséges a *max-lease-time*, az alap beállítás pedig a *default-lease-time* opcióval. Itt másodpercben kell megadni az időtartamokat. A leggyakrabban használt bejegyzések az *option, subnet, host*. Az *option* általános, illetve alhálózatra/gépre vonatkozóan adhat meg paramétereket:

option domain-name ”tilb.sze.hu”;
option domain-name-servers 193.224.130.161, 193.224.128.1;
option routers 193.224.130.161;

A *subnet* egy alhálózaton belüli paramétereket határoz meg. A címek dinamikusan kerülnek kiosztásra, a *range* által megadott tartományokból. Itt is használható az *option* kulcsszó, ez az alhálózatra adja meg az opciókat, amennyiben nincs megadva ilyen, akkor az általános részben definiáltak kerülnek értelmezésre, A tisztánlátás érdekében érdemes minden alhálózathoz megadni ezeket a paramétereket, amint az a példában is látható:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.20 192.168.1.30;
    option broadcast-address 192.168.1.255;
    option routers 192.168.1.1;
    option domain-name servers 192.168.1.1, 193.225.12.58, 193.224.128.1;
}
```

A host kulcsszóval egyetlen gépre vonatkozóan adhatjuk meg az IP címet, az átjárót, a DNS szervert. A hivatkozás Ethernet cím alapján történik:

```
host pc0 {
    hardware ethernet 00:50:04:34:A7:5F;
    fixed-address 192.168.1.7
}
```

A dhcpd indítása. A *dhcpd* megköveteli, hogy minden felkonfigurált interfészhez legyen a konfigurációs fájlban definíció. Ha nem kívánjuk az összes hálózaton elindítani a szolgáltatást, akkor is létre kell hozni mindegyikhez egy *subnet* hivatkozást, ám azt hagyjuk üresen. Indításkor adjuk meg a kiszolgálandó interfészeket. A szervert a *--help* paraméterrel elindítva megkapjuk a lehetséges paramétereket:

```
Internet Software Consortium DHCP Server
Copyright 1995, 1996, 1997, 1998, 1999 The Internet Software Consortium.
All rights reserved.

Please contribute if you find this software useful.
For info, please visit http://www.isc.org/dhcp-contrib.html

Usage: dhcpd [-p] [-d] [-f] [-cf config-file]
```

[-lf lease-file] [-pf pidfile] [if0 [...ifN]]

Példa az indításra:

root@tilb:~# dhcpd –q eth0

Ekkor csak az eth0-hoz tartozó hálózatokon tartózkodó gépeket fogja kiszolgálni a szerverünk.

A kliens oldalon operációs rendszertől függően kell beállítani, hogy DHCP szervertől kapjunk IP címet. Debian Linux alatt a */etc/network/interfaces* fájlban kell hivatkozni rá.

10.7 DNS szerver: named és konfigurációja

A DNS alapjai. A DNS (Domain Name System) feladata, hogy a hálózaton lévő számítógépek számára kijelölt neveket IP címre fordítsa, illetve az IP címeket nevekre fordítsa. Ezen funkciónak a felhasználó számára teljesen „átlátszónak" kell lennie. Azt a műveletet, amikor egy hálózati nevet (pl.: tilb.sze.hu) IP címre fordítunk, feloldásnak hívjuk (angolul: name resolving, mapping). Azt a műveletet, amikor egy IP címet host névre fordítunk magyarul csak körbeírni tudjuk: visszafelé feloldás (angolul: reverse mapping).

Host nevek, domain-ek.

Top level domain name (TLD): Ezek a hálózati név hierarchiában a legfelső helyen álló, gyökér domain nevek. (Pl.: .hu, .com, .net, .org, .tw, .it, stb…)

Domain name: Ezek a TLD-k alatt álló névrészletekkel együtt vett nevek, (Pl.: sze.hu, debian.org, stb…)

Host name: A domain névhez ragasztott név azonosít egy adott gépet. (Pl.: gwta.sze.hu, ip.sze.hu, stb…)

Fontos továbbá megérteni, hogy a host név nem befolyásolja, hogy milyen szolgáltatás fut az adott gépen. Attól, hogy egy gépet „www.valami.net" –nek hívnak, még futhat rajta ftp szerver is, illetve nem törvényszerű, hogy ha egy gépen WEB lap van, akkor a

nevének www-vel kell kezdődnie. (Tipikus példa: http://studio.sth.sze.hu, http://kerdit.sth.sze.hu)
Jegyezzük meg, hogy egy IP címhez akárhány host vagy domain nevet rendelhetünk, de egy host névre csak egy IP címet szabad visszakapnunk a DNS-ből.

Domain ←→ Zóna. Vegyük példának a sze.hu domain-t. A sze.hu egy zónaként is funkcionál, hiszen a www.sze.hu , illetve az rs1.sze.hu a sze.hu zonában található. Viszont a tilb.sze.hu már nem a sze.hu zónába tartozik, mert ő egy aldomain, ami saját zónát alkot. Az összes host ami tilb.sze.hu aldomain-be tartozik a tilb.sze.hu zónában van. De ha például létrehozunk egy proba.tilb.sze.hu zónát a tilb.sze.hu aldomain-ben, akkor az egy saját zónát alkotva host-okat jegyezhetünk be rá! (Pl.: gep1.proba.tilb.sze.hu)

Helyi feloldás. Ahhoz, hogy a DNS működjön, először bizonyos helyi szabályokat kell meghatároznunk, hogy mit milyen sorrendben, és honnan kérdezzünk le. A névszolgáltatások lekérdezésének sorrendjét a */etc/nsswitch.conf* fájl két sora adja meg:

hosts: files dns
networks: files dns

Ez annyit jelent, hogy a host és domain nevek feloldását előbb a helyi fájlokból, majd a DNS szerverből próbáljuk meg elkérni. A hostnév-IP cím megfeleltetéseket tartalmazó fájl a /etc/hosts, melynek egy sora itt látható:

193.224.130.189 dia.tilb.sze.hu dia

Az első oszlopba a gép IP címét, majd a teljes nevét, majd az utolsóba egy aliast írhatjuk. Az alias megkönnyíti gépre való hivatkozást. Pl.: ssh-nál nem kell beírni, hogy *ssh dia.tilb.sze.hu*, hanem egyszerűen csak *ssh dia*.

Ha a helyi fájlokban nem található meg a kért hostnévhez tartozó IP cím, egy DNS szerverhez kell fordulni. A */etc/resolv.conf* nevű fájl határozza meg a feloldás sorrendjét:

search tilb.sze.hu sze.hu
nameserver 193.224.130.161
nameserver 193.224.128.1

A *search* sorba írt domain nevekkel egészíti ki a rendszer a domain név nélkül beírt host neveket. A „*nameserver*” sorokban adhatjuk meg a DNS szerverek IP címeit, az első sorban megadott lesz az elsődleges.

A névfeloldás folyamata. Vegyünk egy egyszerű esetet. Otthoni internetes kapcsolatunkkal felkeressük a tilb.sze.hu gépet. Az otthoni bejegyzett DNS szerverünk (Pl.: a 62.112.192.4). Ez a szerencsétlenül járt névfeloldó szerver megkapja (a TCP és UDP protokollok 53-as portján) a kérést, hogy oldja fel a tilb.sze.hu nevet és adjon vissza nekünk egy IP címet. Mivel ez a szerver még hírből sem ismeri ezt a domaint, először lekérdezi a TLD szervertől, hogy ki a felelős a .hu feloldásáért. Majd erre a TLD válaszol hogy a .hu domainért az NIIF szervere felel. Ezután az általunk használt névfeloldó szerver (62.112.192.4) Megkérdezi az NIIF szerverétől, hogy ki felel a sze.hu domain feloldásáért. Az NIIF szerver boldogan választ ad, hogy a sze.hu-t az rs1.sze.hu (193.224.128.1) kezeli. Ezután a már hosszú utat bejárt névfeloldó szerverünk megkérdezi az rs1-től, hogy ki a felelős a tilb.sze.hu domain feloldásáért. Ő kidobja a választ, hogy a 193.224.128.28 IP című gép. Így megvan, hogy a keresett IP cím ez. A névfeloldó szerver (62.112.192.4) vissza is adja nekünk a választ, így mi el tudjuk érni a labor szervert. Mivan, ha másodszor is lekérdezzük ezt a nevet? Semmi baj, mert a 62.112.192.4 eltárolta (el cache-elte) magának ezt a domain – ip cím párost, így legközelebb már nem kell ezt a hosszú tortúrát végig szenvedni.

Caching-only name server. Ha egy host nevet beírunk például a WEB böngészőbe, elég sokat kell várnunk míg a kapcsolat felépül. Ennek oka, hogy a DNS feloldás

lassú, és esetleg több domain-utótag feloldását is ki kell várni, ami fel van sorolva a /etc/resolv.conf –ban. A cacheing-only name server ezen úgy segít, hogy miután már egyszer feloldotta egy host nevét, megjegyzi, és a következő kérés esetén már ez szolgál ki minket, rendkívül felgyorsítva a DNS feloldás folyamatát. Ez a módszer különösen hasznos, ha lassú vonalon keresztül kapcsolódunk az Internethez.
A caching-only name server megvalósítható a named nevű program segítségével, amely megtalálható a Debian Linux csomagkészletében. (Installálása: *apt-get install bind*)

A root DNS szerverek. Ahhoz, hogy a DNS szerverünk tudja, egy adott TLD feloldásához hová kell fordulni, meg kell adnunk neki az ún. root DNS szerverek IP címeit. A root DNS szerverek hatalmas teljesítményü gépeken futó DNS szerverek, melyek az Internet fő gerincvonalain ülve a Top level domain-ek feloldását végzik. Természetesen ezeket az IP címeket nem kell tudnunk fejből, le lehet tölteni egy megfelelő fájl formájában az ftp.rs.internic.net címről. A fájl neve named.root. Ezt a fájl helyezzük el a /var/named alkönyvtárban root.hints néven. Ez a fájl megtekinthető a tilb-en!

A named.conf fájl. A named alapkonfigurációs fájlja a */etc/bind/named.conf*.

```
options {
        directory ”/var/named”;
};
controls {
        inet 127.0.0.1 allow { localhost; } keys { rndc_key; };
};
key ”rndc_key” {
        algorithm hmac-md5;
        secret ”c3Ryb25nIGVu ... tYW4K”;
};
zone ”.” {
        type hint;
```

```
        file ”root.hints”;
};
zone ”0.0.127.in-addr.arpa” {
        type master;
        file ”mydomain/127.0.0”;
};
```

Az „options” részben a „directory” kulcsszó megadja, hogy melyik alkönyvtár legyen a named alapértelmezett gyökér könyvtára, innen indul ki minden további elérési út, melyet megadunk. A „controls” megadja az rndc szoftverrel való kapcsolat módját, ahol az „allow” kulcsszó azokat a gépeket jelenti, ahonnan a DNS szerver adminisztrálható. A keys megadja, hogy lesz egy titkosítási kulcsunk, melyet a „keys” részben meg is adunk. Ennek a kulcsnak egyeznie kell azzal a kulccsal, amelyet az adminisztrációs gép /etc/rndc.conf fájljában megadtunk. Ezután következik a root zóna megadása (zóna = domain), melyben a fentiekben említett root.hints fájl elérési útját kell megadni. A root zóna után meg kell adnunk a saját zónánkat leíró konfigurációs fájlokat, esetünkben ebből egy van, ez pedig a /var/named/mydomain/127.0.0 nevű fájl.

A zónaleíró fájl. Nézzük meg a zónaleíró fájlunkat, ha csak caching-only DNS –t építünk:

```
$TTL 3D
@       IN      SOA     ns.linux.bogus.      root.linux.bogus. (
                                1       ; Serial
                                8H      ; Refresh
                                2H      ; Retry
                                4W      ; Expire
                                1D      ; Minimum TTL
                        NS      ns.linux.bogus.
1                       PTR     localhost
```

Az első sor a Time To Live (TTL) értékét adja meg három napban, amely azt jelenti, hogy ez a fájl három nap alatt elévül. A második sorban az ns.linux.bogus a gépünk nevét jelent, a root.linux.bogus pedig a gép adminisztrátorát. Egyenlőre a többi beállításról nem esik szó, arra majd visszatérünk.
Az NS sor adja meg a domain DNS szerverének nevét, esetünkben ez ugyanaz, mint a 2. sorban lévő név. (Az utána álló pont fontos, mert így azt lezárja, és nem egészíti ki további DNS utótagokkal!)
Az utolsó sor megadja, hogy a (127.0.0).1 gép neve localhost.(linux.bogus).

Ahhoz, hogy a gépünkön lévő szoftverek a lokális DNS szervertől kérjék a nevek feloldását, meg kell adnunk a *resolv.conf* fájlban a következőket:

search linux.bogus
nameserver 127.0.0.1

Ezután már csak el kell indítanunk a DNS szervert, azaz a *named* programot:

root@ns:/# named –U daemon

A –U daemon paraméter azt jelenti, hogy a named „daemon” usernév alatt fog futni, nem root jogosultságokkal, erre biztonsági megfontolásokból van szükség.

Egy egyszerű domain megvalósítása. Nézzük meg, mi is szükséges ahhoz, hogy végre legyen egy igazi DNS szerverünk! Először is, szükségünk van egy domain névre, amit szeretnénk ezzel kiszolgálni. Legyen ez mondjuk a laborunk domain-je a „tilb.sze.hu” domian.
Ezután szükségünk van még egy IP tartományra, amit a hálózatcímmel és a netmask-al adunk meg: 193.224.130.160/27 (másképp 193.224.130.160/255.255.255.224). Így már mindent tudunk, hogy megírjuk a named.conf fájlt.
Először is, szükségünk van arra, hogy a szerverünk local domain-jét és saját nevét lekezeljük, ezt az előbb már megismertük a caching-only DNS beállításakor:

```
zone ”0.0.127.in-addr.arpa” {
        type master;
file ”tilb/127.0.0”;
};
```

Az „in-addr.arpa” egy speciális domain név, melynek segítségével egy gép nevét kerdezhetjük meg a DNS rendszertől, az IP cím ismeretében. A „zone” kulcszó megadja, hogy mi lesz a zóna fájlunk esetében a „gyökér” domain, azaz innen ered a hierarchiánkban majd minden név (ezután ezt nehezen körülírható fogalmat eredetnek fogjuk hívni.)
Ez tehát megmutatja, hogy a 127.0.0 domaint mi fogjuk kezelni, persze csak a saját hatáskörünkben, ez tulajdonképp a /etc/hosts fájl tilb.sze.hu bejegyzésének helyettesítésére szolgál. Lássuk a /var/named/tilb/127.0.0 fájl tartalmát:

```
$TTL 3D
@       IN      SOA     tilb.sze.hu.    root.tilb.sze.hu. (
                                        2003090102
                                        8H      ; Refresh
                                        2H      ; Retry
                                        4W      ; Expire
                                        1D)     ; Minimum TTL
                NS      tilb.sze.hu.
1               PTR     localhost.
```

Egy zóna fájl három darab úgynevezett erőforrás rekordot tartalmaz, ezek: A SOA, az NS és a PTR rekordok. A SOA a Start Of Authority rövidítése, ezzel indítjuk a DNS zónafájlt:

```
@       IN      SOA     tilb.sze.hu.    root.tilb.sze.hu. (
```

A „@” az eredet egy rövidítése, tehát a fenti sor tulajdonképp így néz ki:

```
0.0.127.in-addr.arpa    IN      SOA     tilb.sze.hu.    root.tilb.sze.hu. (
```

Az NS rekord megmutatja, hogy a tilb.sze.hu domain-hez tartozó name szerver a tilb.sze.hu gép. A „@” rövidítés hatás itt is érvényesül, már nem kell kiírni:

0.0.127.in-addr.arpa IN NS tilb.sze.hu.

Végül a PTR (Domain Name Pointer) rekord megmutatja, hogy az 1-re végződő IP címhez tartozó gépnév a hálózatban (0.0.127.in-addr.arpa, azaz 127.0.0.1) a localhost.
A SOA rekorddal kell kezdődnie a DNS szerverben minden zóna fájlnak, ebben vannak az alapvető információk a DNS zónáról. Minden zóna fájlban csak egy darab lehet belőle. A SOA kulcsszó után meg kell adni a zóna nevét, majd az adminisztrátor e-mail címét olyan formában, hogy a @ helyére is pontot írunk. Az ez alatti sorban lévő szám az úgynevezett sorozatszám (vagy serial) amely egy dátumból és az az napi változások számából áll. (Ha változtatunk a fájlon, ide beírjuk a mai dátumot, és egy 01-et, majd ha megint báltoztattunk, 02-re cseréljük a végét.) Ebből a számból tudja a többi DNS szerver, hogy változtattunk a zóna fájlon, és újból le kell tölteniük. A többi szám helyére (Refresh, Retry, Expire, Minimum TTL) írjuk a fent található számokat, ezek a biztonságos és szokásos értékek. A Refresh, Retry, Expire a másodlagos (kisegítő névfeloldó szervernek szóló információk), míg a minimum TTL egy elutasított kérés (pl.: olyan domain név lekérdezése, amely nem szerepel a nyílvántartásban) elévülését jelenti.
Most pedig adjunk egy új zóna bejegyzést a /var/named/named.conf fájlhoz.

```
zone ”tilb.sze.hu” {
        type master;
        file ”tilb/tilb.sze.hu”;
}
```

Ezzel létrehoztunk egy új zónát, mely a tilb.sze.hu névre hallgat, és a hozzá tartozó zóna file a /var/named/tilb/tilb.sze.hu, melynek a tartalma a következő:

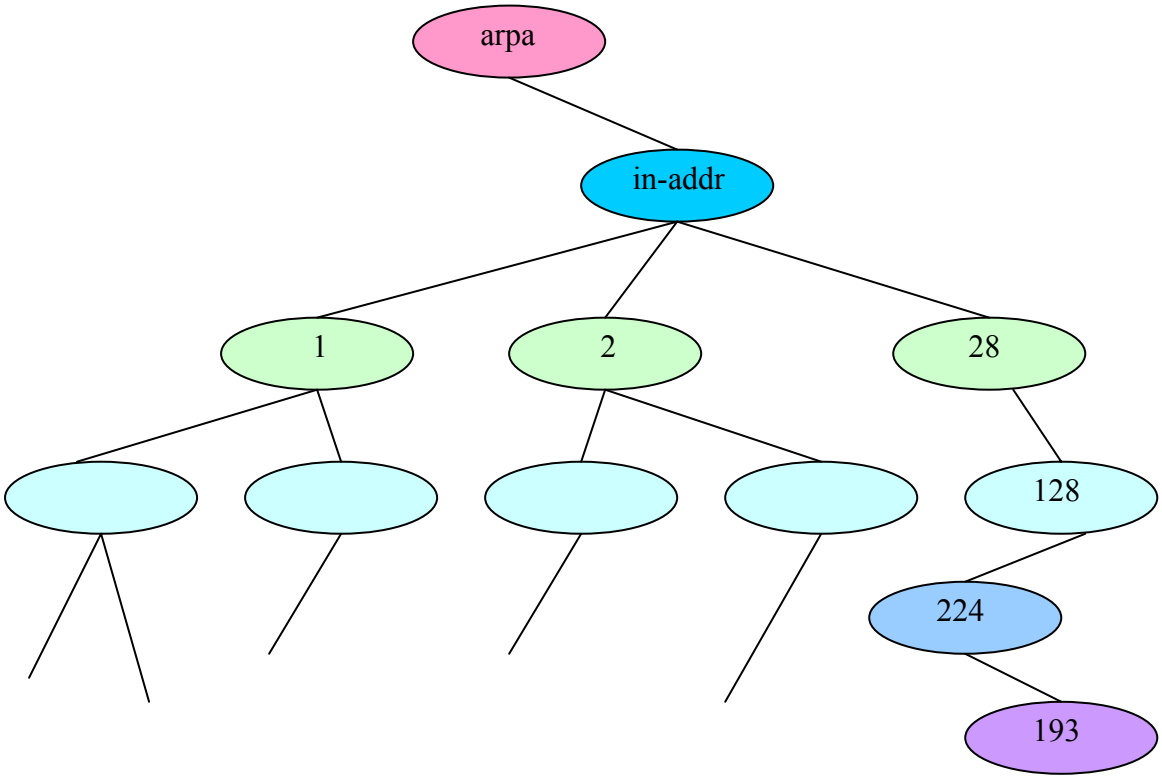
```
$TTL 3D
@       IN      SOA     tilb.sze.hu.    root.tilb.sze.hu. (
```

		2003090102
		8H
		2H
		4W
		1D)
	NS	*tilb.sze.hu.*
	MX	*10 tilb.sze.hu.*
localhost	*A*	*127.0.0.1*
tilb.sze.hu.	*A*	*193.224.128.28*
mmws	*A*	*193.224.130.168*
dia	*A*	*193.224.130.189*
pc0	*A*	*193.224.130.170*
pc1	*A*	*193.224.130.171*
pc2	*A*	*193.224.130.172*
pc3	*A*	*193.224.130.173*
pc4	*A*	*193.224.130.174*
pc5	*A*	*193.224.130.175*
pc6	*A*	*193.224.130.176*
pc7	*A*	*193.224.130.177*
pc8	*A*	*193.224.130.178*
pc9	*A*	*193.224.130.179*
teacher	*A*	*193.224.130.165*
ip	*A*	*193.224.130.166*
voip3	*A*	*193.224.130.181*
wlan	*A*	*193.224.130.162*
ras	*A*	*193.224.130.188*
gw	*A*	*193.224.130.161*
voip1	*A*	*193.224.130.163*
voip2	*A*	*193.224.130.164*
ede	*A*	*193.224.130.167*
old	*A*	*193.224.130.190*

ta65	*CNAME*	*tilb.sze.hu.*
ta13	*CNAME*	*tilb.sze.hu.*
ta37	*CNAME*	*tilb.sze.hu.*
ta67	*CNAME*	*tilb.sze.hu.*
ta68	*CNAME*	*tilb.sze.hu.*
ftp	*CNAME*	*tilb.sze.hu.*
www	*CNAME*	*tilb.sze.hu.*
ns	*CNAME*	*tilb.sze.hu.*
mail	*CNAME*	*tilb.sze.hu.*

Az MX rekord a default levelező szervert adja meg a hálózatunkon. Az „A” rekordok mindegyike egy IP címhez hozzárendelt domaint jelöl meg. Így érhetőek el doamin név alapján a zonafájlban megadott számítógépek. A „CNAME” rekordok a tilb.sze.hu IP címére hivatkozó bejegyzések, így például a ta65.tilb.sze.hu címre hivatkozva a bekonfigurált apache szerver a tantárgy honlapját adja be. Viszont ha a böngészőbe az ftp.tilb.sze.hu -t írjuk be akkor a tilb anonymous ftp szolgáltatása fog megjelenni. Látható tehát, hogy a virtuális doamin-ek mindegyike (amit a „CNAME” rekorddal hoztunk létre) a tilb.sze.hu –ra mutat.

A reverse DNS beállítása. A reverse DNS feladata, hogy IP címből visszaadja a domain nevet. Értelemszerűen egy IP címből csak egy doamin nevet adhat vissza, annak ellenére is, hogy egy IP címre több domain név is mutatjat!
A reverse DNS feloldását az arpa TLD végzi. Az itt található in-addr szerver decimális számok bejegyzéseit tartalmazza. Ezek a decimális számok az IP címek 8 bites számait jelentik. Ezek fastruktúra szerűen szétágazódnak (lásd 20. ábra). A feloldás menete lentről felfelé történik. Pl.: a 193.224.128.0/24 tartományhoz tartozó zónafájlt megfordítva kell bejegyezni a named.conf-ba: 0/24.128.224.193.in-addr.arpa.



20, ábra: reverse DNS feloldás

A példánkban a named.conf fájlba bejegyeztük a tilb.sze.hu.rev zónafájlt. Ez a zónafájl felel a reverse DNS beállításokért. A tilb.sze.hu.rev fájl a következő elemekből áll:

```
$TTL 3D
@       IN      SOA     tilb.sze.hu. root.tilb.sze.hu. (
                        2003090201
                        28800
                        7200
                        604800
                        86400)
              NS      tilb.sze.hu.
161     PTR     gw.tilb.sze.hu.
162     PTR     wlan.tilb.sze.hu.
163     PTR     voip1.tilb.sze.hu.
164     PTR     voip2.tilb.sze.hu.
```

165 PTR teacher.tilb.sze.hu.
166 PTR ip.tilb.sze.hu.
167 PTR ede.tilb.sze.hu.
168 PTR agytroszt.tilb.sze.hu.
169 PTR firewall.tilb.sze.hu.
170 PTR pc0.tilb.sze.hu.
171 PTR pc1.tilb.sze.hu.
172 PTR pc2.tilb.sze.hu.
173 PTR pc3.tilb.sze.hu.
174 PTR pc4.tilb.sze.hu.
175 PTR pc5.tilb.sze.hu.
176 PTR pc6.tilb.sze.hu.
177 PTR pc7.tilb.sze.hu.
178 PTR pc8.tilb.sze.hu.
179 PTR pc9.tilb.sze.hu.
181 PTR voip3.tilb.sze.hu.
182 PTR dhcp182.tilb.sze.hu.
183 PTR dhcp183.tilb.sze.hu.
184 PTR dhcp184.tilb.sze.hu.
188 PTR ras.tilb.sze.hu.
189 PTR dia.tilb.sze.hu.
190 PTR old.tilb.sze.hu.

Itt látható, hogy a reverse DNS zónafájl nem sokban különbözik a DNS zóna fájltól. Itt is megtalálható a ”@” karakter, amely minden bejegyzett sorhoz hozzárendeli 193.224.130. címet, így nekünk csak az utolsó számot kell beírni, és a gép teljes nevét. A teljes domain név után tegyünk mindig pontot.

Ha domain nevet szeretnénk regisztrálni, arra vannak cégek, akik azzal foglalkoznak, hogy pénzért domain nevet regisztrálnak. Regisztrálni csak olyan nevet lehet, ami nincs még felhasználva. Bővebb információ: http://www.iif.hu.

10.8 HTTP szerver: apache és konfigurációja

Ebben a fejezetben megtanuljuk, hogyan lehet WEB szervert építeni és konfigurálni az apache HTTP daemon segítségével.

Az apache legújabb változata mindig megtalálható valamely Debian tükörszerveren, így az *apt-get install apache* parancs megadásával azonnal telepíthetjük rendszerünkre. Az apache irányítására a következő parancs szolgál: *apachectl*. Az apachectl funkciói:

start – http daemon elindítása

stop – http daemon leállítása

restart – http daemon újraindítása

fullstatus – teljes státusz

status – rövid státusz

graceful – „kegyes” újraindítás SIGUSR1 küldésével, így a kapcsolatok nem szakadnak meg.

configtest – konfiguráció syntaxis ellenőrzés

help – ez a segítséget adó funkciólista

A Debian az apache konfigurációs fájljait a /etc/apache könyvtárba, míg a futtatható fájljait a /usr/sbin könyvtárba helyezi el. A számunkra elsősorban lényeges dokumentum fájlok a /var/www/htdocs alkönyvtár alatt vannak. Itt most az Apache help oldalait találjuk, de mivel ide szeretnénk tenni a saját WEB-lapunkat, nyugodtan tötöljünk le mindent ebből a könyvtárból.

Ha ide bemásoljuk a saját WEB-lapunk fájljait úgy, hogy a WEB-lap index.html-je a /var/www/htdocs alkönyvtárban van, és az összes többi tartalom az ez alatt lévő könyvtárakban, akkor honlapunk működőképes lesz.

Az apache konfigurációs fájljai a /etc/apache könyvtárban vannak. Ezek közül számunkra egyedül a httpd.conf érdekes, azonban a teljesség kedvéért megadjuk a többi fájl funkcióját is.

httpd.conf – Az apache fő konfigurációs fájlja, ebben van majdnem minden beállítás.

access.conf – Az apache-hoz történő hozzáféréseket lehet benne szabályozni.

srm.conf – Egyéb, egyéni konfigurációs utasításokat adhatunk itt meg.
magic, mime.types – A dokumentumtípusok és kezdőmoduljaik vannak itt felsorolva.
Az access.conf és az srm.conf alap esetben üresek, mert jobb minden beállítást a httpd.conf fájlban tartani. Ez a két fájl a httpd.conf megfelelő sorában lenne include-olva, azonban ez a két sor ki van kommentezve tehát ezt a két fájlt nyugodtan figylmen kívül hagyhatjuk. Web szerver farm esetén a gépenként eltérő beállításokat tehetjük az srm.conf-ba.

Az apache részltetes konfigurációja – a httpd.conf felépítése. A httpd.conf fájlt megnyitva láthatjuk, hogy alapvetően három fő részből áll:

I. A globális opciók, melyek a szerverprogram működését adják meg, pl.: hány child processz fusson, stb…
II. A „fő szerverre” vonatkozó opciók, melyek a nem-virtuális, tehát valódi névvel és IP címmel rendelkező WEB szerverek működését adják meg. (Ezt később részletesen is elmagyarázzuk)
III. A virtuális WEB szerverek konfigurációja, melyek alias-ként szerepelnek csak a DNS-ben és az alias ugyanerre a gépre mutat.

Amikor itt különféle „szerverekről” beszélünk, az nem azt jelenti, hogy ennyi gépünk van, hanem, hogy egy gépen belül több http szolgáltatást is indíthatunk.

Globális opciók

ServerType standalone
vagy
ServerType inetd
Itt megadhatjuk, hogy a httpd szerver állandó processzként fusson-e, vagy az inetd szuperszerver indítsa el. Ha az előbbit szeretnénk ide „*standalone*”-t, ha az utóbbit „*inetd*”-t kell írni. Ha *inetd*-vel akarjuk indítani, akkor az *inetd.conf*-ba be kell állítanunk ennek a portnak a figyelését!

Ezentúl, ha ilyen jellegű konfigurációs opció van, ahol több lehetőség közül kell választani, így fogjuk jelölni: *ServerType (standalone | inetd)*. Természetesen, csak egyiket szabad a fájlba beírni.

ServerRoot ”/etc/apache”

Megadja, hogy a httpd szerver számára mi legyen a root könyvtár. Ezután minden relatív útvonal ehhez képest értendő. Tehát ha a httpd.conf-ban egy fájlra hivatkozunk pl.: *logs/error.log* akkor ez valójában a */etc/apache/logs/error.log* fájlt fogja jelenteni. Természetesen továbbra is működnek az abszolút elérési utak, pl.: */var/log/apache/error.log*

PidFile /var/run/httpd.pid

Megadja azt a fájlt, ahol a szülő-httpd a saját processz-ID-jét tartja. Ne nyúljunk ehhez a beállításhoz, mert az apachectl indító szkript működését elronthatjuk vele.

Timeout 300

Egy kliensnek, csatlakozás után ennyi ideje van, hogy kérést adjon a szervernek, különben a szerver timeout üzenetet küld, és bezárja a TCP socketet.

KeepAlive (On|Off)

Ha ez „On”, akkor egy kliens egy TCP kapcsolat alatt több kérést is küldhet, ha „Off”, akkor a kapcsolat a kérés teljesítése után megszakad. Az „On” állapot akkor hasznos, ha a html oldalaink több részből állnak (pl.: képek, frame-k).

MaxKeepAliveRequests 100

Ha az előbbi „On” akkor ez adja meg a TCP kapcsolatonkénti maximális kérések számát. Ha teljesítményre kell optimalizálni egy szervert, akkor érdemes ezt viszonylag magas értéken tartani (néhány százas nagyságrend).

KeepAliveTimeout 15

Ennyi másodpercet várunk egy nyitott TCP kapcsolatban a következő kérésre.

MinSpareServers 5

MaxSpareServers 10

A „tartalék” szerver processzek számának minimális és maximális értéke. Kis szervereknél érdemes a minimumértéket 1-en tartani, ez indítás után 2 processzr fog eredményezni: 1db szülő és 1db tartalék (gyerek). Ha a WEB szerver terhelése növekszik a szülő httpd „ellik” még gyereket. A maximum értéket se tegyük 5-nél nagyobra, ha nincs extrém terhelésű szerverünk.

StartsServers 5

A szülővel együtt ennyi processzt fogunk indítani elsőre.

MaxClients 150

Ez az összes futó httpd kérések maximális száma. Ha ezt elérjük a WEB szerver nem szolgál ki több klienst. Nem érdemes túl alacsonyra állítani, mert szerverünk el fog utasítani klienseket. Ha túl nagyra vesszük, és jön egy túlterhelés a hálózat felől, rendszerünk alól elfogy a memória, és a szerver kiakad.

MaxRequestsPerChild 0

Ennyi klienst szolgálhat ki egy processz, utána kihal. Ha 0-ra van állítva, ez a szám korlátlan. Linux esetén nincs szükség erre az opcióra, de ha pl.: Solaris alatt futtatjuk az apache-ot akkor megelőzhetünk vele kóros memóriazabálási jelenségeket.

#Listen 3000

Itt beállíthatjuk, hogy a httpd szerver ne a 80 porton figyeljen. Csak akkor állítsuk át, ha kimondottan ilyen szándékunk van.

LoadModule vhost_alias_module libexec/modvhost_alias.so

AddModule mod_vhost_alias.c

Ez a két sor egy apache bővítőmodul betöltést végez el, számtalan ilyen van a httpd.conf-ban, csak akkor van rá szükségünk, ha egy új modult írunk vagy fordítunk az apache-hoz.

A fő szerver konfigurációja

Másnéven globális opciók. Ebben a részben adhatjuk meg az alapértelmezett WEB-szerver beállításait.

Port 80
Ezen a porton figyel a főszerver.

User nobody
Group nobody
Itt megadhatjuk, milyen UID és GID alatt fusson a szerver. Ez biztonsági szempontból fontos, ha a webszerver törhető, akkor sem tudnak csak nobody jogokkal garázdálkodni a rendszerünkben az illetéktelenek.

ServerAdmin drmomo@tilb.sze.hu
Azt adja meg, hogy ki a szerver adminisztrátora.

ServerName tilb.sze.hu
Ez a szerver neve, amire hallgat, emellett ennek nyilván egy érvényes, DNS-ben szereplő névnek kell lennie.

DocumentRoot ”/var/www/htdocs”
Itt megadhatjuk, melyik alkönyvtárban lesznek a honlap fájljai.

<Directory ”/var/www/htdocs”>
Itt ugyanannak az elérési útnak kell szerepelnie, mind a „*DocumentRoot*”-nál.

Ez a *Directory* tag a „*DocumentRoot*”-ot jelöli meg egésszen a */Directory* tag-ig. Itt az alapvető elérhetőségek és jogok beállítását tehetjük meg.

Options Indexes Includes FollowSymLinks MultiViews

Opciók megadása, pl.: Szimbólikus linkek követése.

AllowOverride None
Order allow,deny
Allow from all

Az alapvető logika: Elősször az engedélyt adunk meg, majd a tiltást.
Az *„allow from all”* megengedi mindenkinek a kapcsolódást ehhez a könyvtárhoz.
</Directory>
A *Directory*-t lezáró tag.

<IfModule mod_userdir.c>
UserDir public_html
</IfModule>

Ezek a sorok adják meg, hogy a felhasználók saját honlapjaik milyen nevű alkönyvtárban kell lenniük a home könyvtáron belül. Lehetőleg ne változtassunk rajta, ezzel sok kérdést előzhetünk meg a felhsználók irányából.

HostNameLookups (Off|On)
Ha bekapcsoljuk, a naplófájlokban megjelenik a hostoknak a neve is, amelyek kérést intéztek a szerverhez. Ha kikapcsoljuk, csak IP címet naplóz.

ErrorLog /var/log/apache/error_log
A hiba-naplófájl (*errorlog*) elérési útja. Ha a fájl nem létezik, létrejön, de az elérési útnak léteznie kell, a szerver nem hoz létre alkönyvtárat!
LogLevel (debug|info|notice|warn|error|crit|alert|emerg)

Azt a hibaszintet állítja be, amelynél már létrejön egy bejegyzés az *errorlog*-ban.

LogFormat <formátum-string>

Megadja a log fájlok egy sorának formátumát. Bővebb információk az apache dokumentációban (http://www.apache.org)

CustomLog /var/log/apache/access_log common

Megadja az eléréseket tartalmazó log fájl útvonalát. Itt a letöltött adatok tárolódnak el (ki, mikor, mit szedett le)

ServerSignature (On|Off|Email)

Ha bekapcsoljuk, akkor a szerver által generált (tehát nem az általunk feltöltött dir. List.) oldalak aljához hozzáírja a szerver verziót, illetve az Email opció esetén odatesz egy linket a ServerAdmin-ban megadott e-mail címre.

```
<IfModule mod_alias.c>
    Alias /icons/ ”/var/www/icons/”
    <Directory ”/var/www/icons”>
Options Indexes MultiViews
AllowOverride None
Order allow,deny
Allow from all
    </Directory>
    ScriptAlias /cgi-bin/ ”/usr/lib/cgi-bin/”
    <Directory ”/usr/lib/cgi-bin”>
        AllowOverride None
        Options None
        Order allow, deny
        Allow from all
    </Directory>
</IfModule>
```

Az Alias opció segítségével bemappolhatunk egy a rendszerben lévő alkönyvtárat a WEB szerver gyökeréhez viszonyított elérési útba. A ScriptAlias segítségével megadhatjuk, honnan lehet CGI szkriptet futtatni. A Directory környezetben megadhatjuk, hogy egy adott alkönyvtárra milyen elérési jogok vonatkozzanak. Pl.: Az „Indexes” kulcsszó engedélyezi egy alkönyvtár tartalmának listázását. A különféle opciók jelentését lást az apache dokumentációban.

Virtuális HTTP szerverek konfigurációja

Ha szeretnénk, hogy HTTP szerverünk több hostnévre is hallgasson, úgynevezett virtualhost-okat kell létrehoznunk. Ahhoz, hogy a www.tilb.sze.hu gépet pl.: ta65.tilb.sze.hu néven is nevezhessük, a tilb.sze.hu domain zóna fájlban a következő sornak kell állnia:

ta65 CNAME tilb.sze.hu.

Az apache konfigurációs fájljában a következőképp kell megadnunk a virtualhost-okat:

```
NameVirtualHost *
<VirtualHost *>
        ServerAdmin drmomo@tilb.sze.hu
        DocumentRoot /bigspace/htdocs/ta65/
        ServerName ta65.tilb.sze.hu
        ErrorLog /var/log/apache/ta65.error
        CustomLog /var/log/apache/ta65.custom
</VirtualHost>
```

A „*” a default beállítás, de megadhatnánk helyette a szerverünk ip címét és domain nevét is. (Pl.: *NameVirtualHost 193.224.128.28:80 <VirtualHost 193.224.128.28:80>*)

A „NameVirtualHost” kulcsszó bekapcsolja a virtuális webszervereket. Látható, hogy a virtuális szervereket ugyanúgy kell beállítani, mint a főszervert. A log-ok külön vannak választva, itt is érvényes, hogy a log-ot tartalmazó alkönyvtárnak léteznie kell.

10.9 Microsoft networks kezelése Linux-al: Samba és konfigurációja

Ha a Linux-ot M$ Windows gépekkel szeretnénk összekapcsolni, a samba szervert kell használnunk, mely a Server Message Block (SMB) protokoll Linux alatti megvalósítása. A Samba megtalálható a Debian Linux csomagkészleteiben, telepítése a következő képen lehetséges: apt-get install samba samba-common smbfs.
A konfiguráció megkezdéséhez a */etc/samba/smb.conf-sample* fájlt másoljuk át */etc/samba/smb.conf* névre. Ez tartalmazza a szerverrel kapcsolatos beállításokat.
Nézzük a beállításokat a konfigurációs fájlon végighaladva.
A globális beállítások a *[global]* pontnál találhatók.

workgroup = tilb
A workgroup beállítással adhatjuk meg a munkacsoportot (Win9x, ME) illeteve az NT Domain nevet (WinNT, 2000, XP).

server string = Samba Server on TILB
A server string tartalma a windows ”Számítógép leírása” mezőjében jelenik meg.

hosts allow = 193.224.130. 193.225.150. 193.224.128.
hosts deny = 193.224.130.99
A hosts opcióknál megadhatjuk, melyik IP tartományokból lehessen elérni a szerver szolgáltatásait és melyikből nem. (A host engedélyező fájlok érvényességi sorrendje a UNIX-ban: 1. hosts.allow, 2. hosts.deny, és ha egyikben sem szerepel a cím, akkor meg van engedve a hozzáférés.) Ezen kívül megadhatunk még netmaszkkal vagy az „except” kifejezéssel alhálózatot is például:
hosts allow = 193.224.130.160/27
hosts allow = 193.224.130. except 193.224.130.99

printcap name = /etc/printcap

loadprinters = yes

Ha szeretnénk, hogy a nyomtatónkat a /etc/printcap fájlból töltse be a Samba, és ne kelljen őket egyenként megadni, akkor ezt kell engedélyezni.

printing = bsd

A szerverünk nyomtatási rendszerét itt adhatjuk meg (nem szükséges beállítani). A következő nyomtatási rendszereket támogatja a samba: *bsd, sysv, lprng, aix, hpux, qnx*.

guest account = pcguest

Ha szeretnénk *pcguest* felhasználót, ezt engedélyezni kell, és felvenni a *pcguest*-et a */etc/passwd* fájlba, egyébként a nobody felhasználó jogait használja.

log file = /var/log/samba/log.%m

Minden gép (*%m*) logja külön log fájlba kerül. A „*%m*” a kliens gép netbios neve.

max log size = 100

A log fájlok maximális mérete Kbyte-ban.

security = user

Biztonsági szintet lehet megadni. Ha ezt ”*share*” értékre változtatjuk, akkor nem kell jelszó a megosztások eléréséhez, ”*user*” esetén kell. Ha a ”*server*” biztonsági módot választjuk, akkor külön NT szerver fogja a jelszavakat szolgáltatni.

password server = <NT-server-name>

Az NT-s password szerver nevét adhatjuk meg.

password level = 8

username level = 8

Ennyi darab kis és nagybetű különbséget tolerál a samba a jelszó és a felhasználónév megadásakor. (Pl. ha username level = 1 → fRef, Fred, frEd, freD = fred)

encrypt passwords = yes
smb passwd file = /etc/smbpasswd
Megadhatjuk, hogy a Samba a jelszavakat titkosítsa. Ha ezt használni szeretnénk, akkor bővebb információt az ENCRYPTION.txt, Win95.txt és WinNT.txt fájlokban a samba dokumentációjában találhatunk.

unix passwd sync = Yes
passwd program = /usr/bin/passwd %u
*passwd chat = *New*UNIX*passwd* %n\n *ReType*new*UNIX*passwd %n\n *passwd:*all*authentical*
Ez az opció engedélyezi, hogy Windows alól változtatni tudjuk a Linux-os rendszerjelszót is. Ezeket csak az ”*encrypt passwords*” és az ”smbpasswd” fájl opciókkal használjuk! Figyelem! Ha csak az Samba jelszavakat akarjuk Windows alól változtatni, akkor ezekre nincs szükség!

username map = /etc/smbusers
A Unix és samba felhasználónév lehet különböző is, ha az itt megadott fájlban bejegyezzük őket.

client code page = 852
character set = ISO8859-2
Ezzel a két sorral elérhetjük, hogy a Windows ékezetes fájl-nevei működjenek. A nemzetközi beállításokat részletesen a Samba dokumentációjában találjuk.

include = /etc/smb.conf.%m
Gépenként külön konfigurációs fájlt adhatunk meg a Samba számára, %m-et a gép Netbios nevével kell helyettesíteni.
socket options = TCP_NODELAY

Gyorsabb kiszolgálást érhetünk el ezzel az opcióval…

interfaces = 193.224.130.161/27 eth0

Több hálózati interfészen keresztüli kiszolgálás. Alapértelmezésben minden aktív interfészen fut a szolgáltatás, ami broadcast-elni képes, és nem loopback interfész. (Lehet eth*, ilyenkor minden *eth* interfészre illeszkedik a szabály.)

local master = yes

Tallózáskezelő opciók: állítsuk ezt ”no”-ra, ha nem szeretnénk, hogy szerverünk legyen a local master browser a hálózaton.

os level = 88

Ez adja meg a Master Browser kiválasztásánál a gépünk prioritását.

domain master = yes

Ezzel engedélyezhetjük, hogy a gépünk legyen a Domain Master a hálózatban. Ne használjuk ezt az opciót, ha van már NT Domain master a hálózatban.

A Local Master Browser és a Domain Master Browser az M$ Windows világban az úgynevezett tallózáskezelők, melyek a hálózaton lévő számítógépek neveit és egyéb adatait gyűjtögetik, ennek az az értelme (a M$ szakemberei szerint), hogy így (elvileg) nem lesz annyi broadcast csomag a hálózatban, amikor pl.: egy titkárnő megnyomja a „Teljes hálózat” ikont :). A Local Master csak egy adott alhálózaton gyűjtöget, a domain master pedig az egész NT domain-ben.

preferred master = yes

Ha ezt engedélyezzük, akkor a szerver indításakor egy elekciós kört (master browser election) kezdeményez, és nagyobb esélyünk van, hogy mi gépünk legyen az. Ez viszont veszélyes lehet, ha több gép is szeretne „*mester*” lenni. Ilyenkor a „*kűzdelem*” közben sok broadcast csomag lesz a hálózaton.

domain controller = <NT-Domain-Controller-SMBname>

Megadhatjuk, hogy melyik gép a domain vezérlő. Csak akkor használjuk, ha telepítéskor már van NT domain vezérlőnk.

domain logons = yes

Engedélyezzük, ha szeretnénk, hogy a Samba Domain logon controller legyen a Win95 kliensek számára.

logon script = /etc/smbscripts/%m.bat

Gépenként külön bejelentkező szkriptet engedélyezhetünk.

logon script = %u.bat

Felhasználónként is külön bejelentkező szkriptet engedélyezhetünk.

logon path = \\%L\Profiles\%U

Itt a felhasználói profilok elérhetőségét adhatjuk meg. A %L a szerver Netbios neve, %U a kliens által kért felhasználó név. (Alul a [Profiles] megosztást engedélyezni kell.

name resolve order = host lmhosts bcast

A Netbios neveket IP címekre kell fordítani, és a ”Name Resolve Order” segítségével megadhatjuk a feloldás sorrendjét. Az alap sorrend ”hosts lmhosts bcast”. A ”hosts” azt jelenti, hogy a Unix *gethostbyname()* rendszerhívással próbálja meg feloldani a nevet, melyet akár a */etc/hosts* vagy a DNS vagy a NIS feloldhat, a */etc/host.config*, */etc/nsswitch.conf*, */etc/resolv.conf* fájlok függvényében.

wins support = yes

Ezzel engedélyezhetjük az nmbd WINS szerverét.

wins server = w.x.y.z

Ha a samba csak WINS kliens lesz, mert már van WINS szerverünk, akkor azt állítsuk be, viszont az előbbi beállítást állítsuk ”*no*”-ra!

dns proxy = no

Ha ezt engedélyezzük a DNS által feloldott Netbios neveket eltárolja egy pufferbe, hogy gyorsabb legyen a feloldás.

short preserve case = no

Kis és nagybetűk megőrzése. Alapértelmezett beállítás a ”*no*”. (Ezt megosztásonként szabályozhatjuk.)

default case = lower

Kis és nagybetűk alapértelmezés.

case sensitive = no

Kis és nagybetűk megkülönböztetése, ha megkülönböztetve használjuk, egy-két megosztás nem működik!

Megosztások kezelése.

Ha szeretnénk a felhasználók home könyvtára automatikusan megosztásra kerüljön a homes név alatt, akkor ezt így jegyezzük be:

[homes]

comment = Home directories	#megjegyzés
browseable = yes	#tallózható
writeable = yes	#írható

Netlogon megosztás a netlogon kliensek számára

[netlogon]

comment = Network Logon Service
path = /home/netlogon
guest ok = yes
writeable = no
share modes = no

Felhasználói profilok könyvtára

[Profiles]

path = /home/profiles

browseable = no

guest ok = yes

Tegyünk a sorok elé „ *;* ” jelet, ha szeretnénk megadni külön könyvtárat a felhasználói profiloknak. Az alapértelmezés, hogy a felhasználó home könyvtárát használjuk.

Standard megosztások.

[bigspace]

comment = Home

path = /bigspace #A megosztott könyvtár elérési útja

valid users = lencse tbalazs drmomo #Kik használhatják

create mask = 700 #Fájlok létrehozási jogai

read only = yes #Csak olvasható

public = no #Mindenki láthatja?

printable = no #Lehet-e rá nyomtatni?

writeable = no #Lehet-e rá írni?

Nyomtató megosztása.

[HP_LaserJet]

comment = HP LaserJet 2200DN

printer name = HP LaserJet 2200DN #Nyomtató neve a Windowsban

path = /var/spool/lpd/ljet4-a4-auto-mono #Nyomtató spool elérési útja

browseable = yes #Tallózható

printable = yes #Lehet rá nyomtatni

writeable = yes #Lehet rá írni

gues ok = no #Guest user használhatja-e?

*print command = echo '/bin/date' %U nyomtatja a %s fajlt. >> *
/tmp/print.log; lpr %s; rm %s #Nyomtató parancssor, a UNIX hajtja végre %U user, %s fájl

Ez a három beállítás az NT kompatibilitás miatt kell.

default devmode = NULL
nt smb support = yes
nt status support = yes

Felhasználók kezelése. Ha már létezik egy felhasználó a Unix-ban a Samba-hoz akkor a következő paranccsal hozhatunk létre smb felhasználót:
smbpasswd –a <usernév>
Illetve az *smbpasswd <usernév>* paranccsal változtathatjuk meg a jelszavát.

A Samba indítása. A samba indítását az smbd és nmbd programok indításával végezhetjük el:
root@tilb:/# nmbd -D
root@tilb:/# smbd -D
Illetve *a /etc/init.d/rc.samba start|stop|restart* szkripttel.

A Samba parancsai. A Samba használatához tartoznak különböző utasítások.
smbmount <//Gépnév/Megosztás> <Könyvtár> – Windowsos megosztások mountolása (Kernelnek tudnia kell az smbfs-t kezelni)
smbclient <opció> <//Gépnév/Megosztás> – Windowsos megosztásokra csatlakozás. Ennek a programnak van egy parancspromptja, amiben a karakteres ftp kliensre emlékeztető parancsokat adhatunk ki. Opció: *–L* egy gép megosztásainak kilistázása.
smbmnt – Samba megosztáskezelő programja.
nmblookup <gépnév> – Samba névfeloldó parancsa. Ezt hívja meg a többi samba parancs a netbios nevek feloldásához.

10.10 Proxy szerver: SQUID és konfigurációja

A proxy szervereket többnyire hálózati cache-elés céljából alkalmazzuk. Két leggyakrabban használt konfiguráció:

I. A bejövő forgalmat cache-eljük és a kliensek innen érik el a WEB-et, így a már másodszor lekért oldal nem terheli a kimenő sávszélességünket.
II. A kimenő WEB szerverünket cache-eljük, így a többször lekért nagy kapacítást igénylő feladatok nem futnak le többször.

A squid beszerzése és installálása az *apt-get install squid* paranccsal lehetséges. A squid konfigurációs fájlja a /etc/squid.conf alatt található. Mivel a konfiguráció nagyon sok beállítási lehetőség van, mi csak a legalapvetőbbet tárgyaljuk. Ezeket a sorokat muszály megváltoztatni, hogy a squid működjön.

http_port 3128
HTTP Proxy portja (TCP, alapértelmezetten a 3128-at, vagy a 8080-as portokat szokták adni)

#cache peer
Ezt az opciót akkor hasznájuk, ha az úgynevezett parent cache (szülő cache) gép, akihez kapcsolódni tudunk, ha nincs ilyen, mint esetünkben, akkor ezt hagyjuk kikommentezve.

cache_mem 16 MB
Ez az opció adja meg, hogy mennyi memóriát szeretnénk adni a squid-nak, cache céljára. (A squid kb. háromszorosát fogja használni a megadottnak, úgyhogy adjuk a harmadát mint amennyit egyébként használni szeretnénk.) Nem szabad többet megadni, mint amennyi tényleges szabad RAM memóriánk van, mert semmi értelme, hogy a system swap-en legyen a cache (illetve ne vegyük el a memóriát a többi processztől)

cache_dir ufs /usr/local/squid/cache 100 16 256

Ez a bejegyzés a fájl cache alkönyvtárat adja meg, ide fogja a squid beszórni a becache-elt fájlokat. Az „ufs" a squid fájl-tárolási módja, ezt ne változtassuk. Az első szám a maximális cache méretet adja meg, a második kettő, pedig az alkönyvtár-rendszer méretét, itt tehát 16x256 könyvtárat fog a squid készíteni. (16 alkönyvtár lesz a */usr/local/squid/cache* alatt, és mindegyik alatt még 256 darab.) A */usr/local/squid/cache* alkönyvtárnak léteznie kell, és a squid számára írhatónak kell lenni, hogy a squid használni tudja.

icp_access deny all

Ezzel letilthatjuk a cache-peer-ként működő gépek hozzáférését, senkinek nem engedjük, hogy rajtunk keresztül cache-eljen.

acl labor src 193.224.130.160/27

Ez az úgynevezett Access Control List, ilyen acl kezdetű sorokkal csoportokat adhatunk meg, hogy honnan, mit lehessen elérni a proxy-n keresztül. Ha src-t adunk meg, akkor azokat soroljuk fel (esetünkben a labor gépeit), akik igénybe vehetik a proxy szolgáltatást, ha dst-t, akkor azt adjuk meg, hogy milyen hostokról tölthetünk le WEB lapokat. A dst listát érdemes az alapértelmezett beállításon hagyni, hogy a squid mindent beengedjen (Kivéve, ha cégünk csak néhány lapot enged böngészni). A minimum konfigurációt hagyjuk meg a config fájlban, ezt az egy sort adjuk hozzá a saját domain-ünk ip címeivel!

http_access allow labor
http_access deny all

Ezzel megadhatjuk, hogy a fentiekben elkészített listák közül milyen szabály vonatkozik. (Itt a labor gépeit, amit az acl sorral definiáltunk, engedjük a cache-hez hozzáférni.)

cache_mgr drmomo@tilb.sze.hu

A szerver gazdájának e-mail címe.

cache_effective_user nobody

cache_effective_group nogroup

Az a felhasználó, illetve csoport, ami alatt futni fog a squid. Ne használjunk root-ot, ha nincs nobody a rendszerünkben, hozzunk létre egy squid user-t. A */usr/local/cache* alkönyvtárat adjunk ennek a user-nek a tulajdonába.

visible_hostname cache.tilb.sze.hu

A név, ami látszani fog a cache neveként. (Pl.: a cache által generált WEB-oldalakon.)

cache_access_log /usr/local/squid/logs/access.log

cache_log /usr/local/squid/logs/cache.log

cache_storage_log /usr/local/squid/logs/storage.log

A log fájlok (hozzáférés, cache, tárolás) helyei. A logs alkönyvtárat is tudnia kell írni a squid-nak.

http_accel_uses_host_header off

Engedélyezzük ezt az opciót, ha transzparens proxyt szeretnénk építeni.

A squid indítása. Mielőtt elindítanánk a squid-ot még oly békés rendszerünkön, be kell állítani a jogosultságokat azokhoz az alkönyvtárakhoz, amiket a squid el fog érni:

root@tilb:/# mkdir /usr/local/squid/cache

root@tilb:/# chown nobody /usr/local/squid/logs

root@tilb:/# chmod 700 /usr/local/squid/logs

root@tilb:/# chown nobody /usr/local/squid/cache

root@tilb:/# chmod 700 /usr/local/squid/cache

Ezután a squid-al el kell készíttetni a könyvtárstruktúrát, ahová cachel-ni fog. Ezt csak most kell megtennünk, és soha többet!

root@tilb:/# squid –z

2003/09/15 15:31:28| Creating Swap Directories

Ezután, ha belenézünk a */usr/local/squid/cache* könyvtárba, láthatjuk, hogy ott a 16x256 db alkönyvtár. Ha ez lefutott, (jó sokáig tart), akkor a következő paranccsal indítjuk a squid-ot:

root@tilb:/# /usr/local/squid/bin/squid

A *ps aux* paranccsal ellenőrizhetjük a squid futását.

Transzparens proxy építése.

A transzparens proxy annyit jelent, hogy a kliensböngészők függetlenül a beállításaiktól rá vannak kényszerítve arra, hogy proxyn keresztül érjék el a WEB-et. Ez úgy lehetséges, hogy a 80-as TCP portot átküldjük a HTTP Proxy portjára, és az szolgálja ki a kéréseket (Ne felejtsük el a squid-ot felkészíteni erre a *httpd_accel_uses_host_header on* beállítással!):

root@tilb.sze.hu:/# iptables –t nat –A PREROUTING –s 193.224.130.160/27 –p tcp --dport 80 –j REDIRECT --to-ports 3128

10.11 Levelező szerver: Sendmail és konfigurációja

Rendszerszintű áttekintés. Miközben egy e-mail eljut a címzetthez, a következő folyamat zajlik le:

I. A felhasználó megírja a levelet az e-mail kliens programmal (pl.: pine)
II. Elküldés után az e-mail kliensben beállított SMTP szerverrel (outgoing mail server) kapcsolatot létesít a felhasználó gépe, és az SMTP szerver TCP 25-ös portjára bejelentkezve elküldi a levelet.
III. Az SMTP szerver a cél e-mail címhez a DNS-ből lekéri a cél domain leveleit kezelő gép címét, és a 25-ös TCP portjára bejelentkezve elküldi azt.
IV. A céldomain-hez tartozó SMTP szerver továbbküldi, vagy a megfelelő felhasználói fiókban elhelyezi a levelet.

Ahhoz, hogy tudjuk, hogy egy adott domain (pl.: tilb.sze.hu) leveleit ki kezeli, a domain DNS szerverében, a megfelelő zóna fájlban (/var/named/tilb/tilb.sze.hu) kell lenni egy bejegyzésnek, amely megmutatja, hogy a domain leveleit melyik gépnek kell

elküldeni, hogy eljussanak a címzetthez. A következő sorok a tilb.sze.hu domain DNS rekordját mutatják, a levelező szervert az ”MX” azonosítót tartalmazó sor adja meg:

```
$TTL 3D
@          IN      SOA     tilb.sze.hu.    root.tilb.sze.hu. (
                                           2003090102
                                           8H
                                           2H
                                           4W
                                           1D)
                   NS      tilb.sze.hu.
                   MX      10 tilb.sze.hu.
localhost          A       127.0.0.1
tilb.sze.hu.       A       193.224.128.28
```

Az MX rekord esetében megadhatunk egy referencia számot. Az a gép fogadja a leveleket alapértelmezetten, amelyik referenciaszáma kisebb.

Ahhoz, hogy ez a gép leveleket tudjon fogadni, futnia kell rajta egy SMTP szervernek. Esetünkben ez a sendmail program lesz, ennek beállításaival fogunk foglalkozni.
A sendmail-t egyszerűen feltelepíthetjük az *apt-get install sendmail* parancs kiadásával.
A sendmail-hez tartozó fájlok: (a sendmail konfigurációs fájljai a /etc/mail alatt találhatók)

sendmail.cf – Itt találhatók a sendmail beállításai.
aliases – Itt tudunk megadni a felhasználókhoz plussz neveket.
aliases.db – Ebben tárolódnak az előző pontban említett plusz nevek, az ún. aliasok.
helpfile – Ebben az SMTP protokollal kapcsolatos segítő szövegeket találjuk.

A küldésre váró kimenő leveleket a sendmail a */var/spool/mqueue* könyvtárból veszi ki, az érkezett leveleket a */var/spool/mail* könyvtárban helyezi el a felhasználó nevével megegyező nevű fájlban, innen tudja a levelező kliensprogram kiolvasni.

Mail alias-ok létrehozása.

Tegyük fel, hogy van egy a számítógépen, pl.: bela.toth néven. Ekkor az ő e-mail címe lehet pl. bela.toth@tilb.sze.hu. Szeretnénk, ha ezután lenne egy beci@tilb.sze.hu e-mail címe is, de természetesen ehhez nem akarunk új felhasználót felvenni. Az új e-mail címet, más néven alias-t a /etc/mail/aliases fájlban kell megadnunk:

beci: bela.toth

Miután beírtuk az alias-t a megfelelő helyre, le kell futtatni a newaliases nevű programot. Ez újratelepíti az aliases.db nevű fájlban lévő adatbázist, így a sendmail gyorsabban hozzáfér az aliasokhoz, mintha állandóan az aliases szövegfájlból kellene kiolvasnia. Ezek után, ha egy e-mail érkezik a beci@tilb.sze.hu címre, azt a bela.toth nevű felhasználó fogja megkapni, azaz hozzáfűződik a /var/spool/mail/bela.toth fájlhoz.

E-mail relay beállítása.

Ha szeretnénk, hogy a sendmail közvetlenül elfogadjon leveleket másik domain-ben lévő számítógéptől is, akkor meg kell adnunk, hogy melyik domainből engedélyezzük a levelek továbbítását. Egy gyakorlatias példán keresztül könnyebben megérthető ez a tulajdonság: a tilb.sze.hu domain leveleit kezelő tilb.sze.hu számítógép el kell, hogy fogadjon leveleket a sze.hu és a szif.hu domain-ekből is, ezen felül van Grúziában egy felhasználó barátunk, aki a tilb-et szeretné kimenő SMTP szervernek beállítani. A /etc/mail/sendmail.cf fájl első részében a következőt kell megadnunk:

FR-o /etc/mail/relay-domains

Ez annyit jelent, hogy lesz egy /etc/mail/relay-domains nevű fájlunk, és itt kell felsorolnunk azokat a domain-eket, amelyek számára hajlandóak vagyunk kimenő SMTP szolgáltatást nyújtani. A tartalma tehát a következő:

sze.hu

szif.hu

sanet.ge

Ezzel elértük, hogy pl.: a sanet.ge domain-ben lévő számítógép is bejelentkezhessen tilb.sze.hu TCP 25-ös portjára. (Mivel a tilb.sze.hu része a sze.hu-nak, így azt nem kell külön bejegyeznünk.) Ahhoz, hogy a változások érvénybe lépjenek, még rá kell vennünk a sendmail-t, hogy olvassa be újra a konfigurációs fájlt.

root@tilb:/# killall –HUP sendmail

Több doamin levelének kezelése.

Előfordulhat, hogy szeretnénk egyszerre több domain leveleit is kezelni (tehát a címzettek ezeken a domaineken tartózkodhatnak), pl. a tilb.sze.hu mellett pl.: az eskola.hu domain leveleit is elfogadjuk. Emelett szeretnénk, hogy a gépünk ne csak a tilb.sze.hu hanem a mail.tilb.sze.hu névre is „hallgasson”.

A */etc/mail/sendmail.cf* fájl első részében a következőt kell megadnunk:

Fw/etc/mail/lclhstnames

Ez azt jelenti, hogy a */etc/mail/lclhstnames* nevű fájl fogja tartalmazni a kezelt domain-ek listáját:

root@tilb:/# cat /etc/mail/lclhstnames
tilb.sze.hu
mail.tilb.sze.hu
eskola.hu
mail.eskola.hu

Ha ezután valaki a drmomo@eskola.hu címre fog levelet küldeni, a felhasználó úgyanúgy megkapja, mint ha az a drmomo@tilb.sze.hu címre jött volna. Természetesen ehhez is szükséges, hogy az eskola.hu domain DNS szerverén a tilb.sze.hu legyen megadva az MX rekordban.

E-mail fiókok letöltése távolról.

Amikor a szerveren számos felhasználó van felvéve, és e-mail-jeik ide érkeznek, lehetőséget kell biztosítani arra, hogy távolról is elérjék e-mail-jeiket, azaz egy számítógép és egy POP3 vagy IMAP levelező kliens segítségével letöltsék leveleiket.

Az IMAP és POP3 kliensek kiszolgálására szerver oldalon megfelelő programokat kell futtatnunk, melyek lekezelik a TCP 110-es POP3, illetve a TCP 143-as IMAP portokat.

UNIX alatt az Internet felé irányuló szolgáltatásokért az inetd szuperszerver felelős, amely figyeli a /etc/inetd.conf fájlban beállított portokat, és kérés esetén elindítja a megfelelő protokoll daemonját.

A következőkben látjuk az inetd.conf megfelelő sorait:

pop3	*stream*	*tcp*	*nowait*	*root*	*/usr/sbin/tcpd*	*popa3d*
imap4	*stream*	*tcp*	*nowait*	*root*	*/usr/sbin/tcpd*	*imap4d*

Az első oszlopban a szolgáltatás nevét, majd a hálózati csomagok típusát és szállítási protokollját látjuk. Ezután a várakozást beállító flag-ek, a daemont futtató felhasználót, majd a parancsot láthatjuk, amellyel a daemon elindul. Látjuk, hogy minden esetben a */usr/sbin/tcpd* nevű daemon indít el egy másikat, amely esetünkben a POP3 és az IMAP4 protokollokat lekezeli. Ez a popa3d és az imap4d nevű daemonok, melyek a debain csomagban megtalálhatók. (apt-get install popa3d mailutils-imap4d)

Irodalomjegyzék

Ez a jegyzet Kallai Péter UNIX című Hálózati operációs rendszerek I. óravázlat-segédanyag felhasználásával készült.

[7.2, 7.3, 9.9, 10.4] fejezetek Dr. Lencse Gábor előadás-vázlatából készültek.
[10.6] fejezet Büki Péter előadásanyagából készült.

További felhasznált segédanyagok forrásai:

[1] http://www.szabilinux.hu
[2] http://www.freeswan.org
[3] http://www.netfilter.org
[4] http://www.debian.org
[5] http://www.kernel.org
[6] http://www.linux.org
[7] Debian Linux Manual

A jegyzetet készítette: Molnár Zoltán Vilmos a Széchenyi István Egyetem – Informatikai és Villamosmérnöki Intézet – Távközlési Tanszék megbízásából.

Tartalomjegyzék