A GNU/Linux karbantartása
1.0.1
A Mithrandir Kft. nyelvi ellenőrzésével

Balsai Péter
Kósa Attila

2002. június 19.

A módosított változat közzétételéért felelős személyek:

Sári Gábor saga@lme.linux.hu

Javítások: Sári Gábor

Szerző

Vitéz Gábor gabor@swszl.szkp.uni-miskolc.hu

Szakmai lektor

Szalay Attila sasa@lme.linux.hu

Nyelvi ellenőrzés

Sári Gábor saga@tux.hu
Kósa Attila atkosa@shinwa.hu

Formázás (LaTeX)

Kósa Attila atkosa@shinwa.hu

Előzmények

A GNU/Linux karbantartása
A Mithrandir Kft. nyelvi ellenőrzésével

A kiadás éve: 2002.

Szerző

Vitéz Gábor gabor@swszl.szkp.uni-miskolc.hu

Szakmai lektor

Szalay Attila sasa@lme.linux.hu

Nyelvi ellenőrzés

Sári Gábor saga@tux.hu
Kósa Attila atkosa@shinwa.hu

Formázás (LaTeX)

Kósa Attila atkosa@shinwa.hu

Az LME által elkészíttetett Pingvin füzeteken a Mithrandir Kft. az olvashatóság érdekében nyelvi, helyesírási javításokat végzett.

A Mithrandir Kft. – valamint a nyelvi javítást végző természetes személyek – szakmai ellenőrzést, javítást nem végeztek.

Nem tették ezt (szakmai javítás), akkor sem – a szerzők és a szakmai lektorok munkája iránti tiszteletből –, ha a leírtak nem feleltek meg szakmai meggyőződésüknek.

A Mithrandir Kft. javítást végző szakemberei, illetve a Mithrandir Kft. mint jogi személy a leírtak helyességéért, esetleges avultságáért semmilyen felelősséget nem vállal.

Tartalomjegyzék

Ábrák jegyzéke

1. fejezet

A *GNU/Linux* rendszer karbantartása

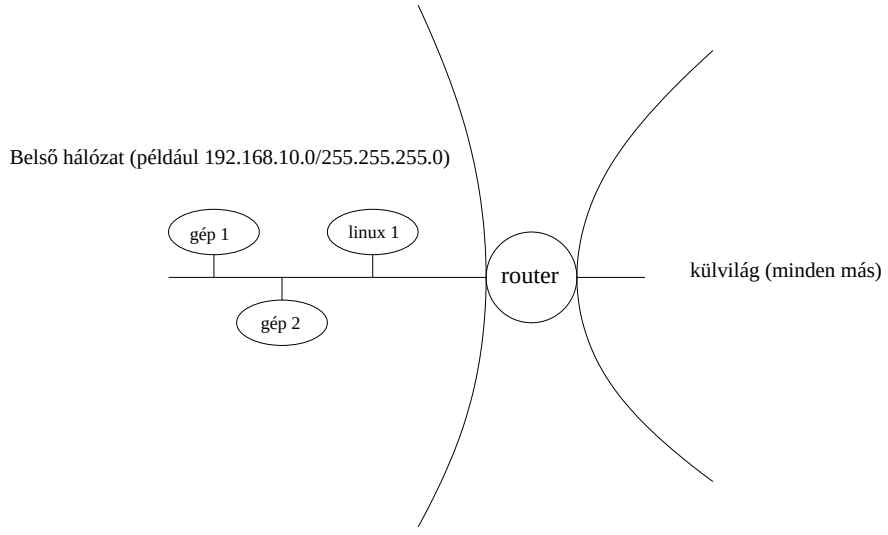
1.1. Hálózati beállítások

1.1.1. Hálózati interfészek konfigurálása

Linux alatt a számítógép hálózati csatlakozási pontjait hálózati interfészek reprezentálják. Az interfész neve, amit a kernel használ, egy szimbolikus dolog, mint például az eszközvezérlő fájlok, de érdemes megjegyezni, hogy a hálózati interfészek nem eszközvezérlő fájlok, ők „máshol élnek”. Az interfészeknek különböző tulajdonságai lehetnek, például a maximális csomagméret amit kezelni tud.

Most ezek beállításáról lesz szó. A beállításoknál TCP/IP hálózati protokollt és Ethernetes környezetet feltételezünk. Manapság ezek a legelterjedtebbek.

A hálózat felépítését az 1.1. ábra mutatja.



1.1. ábra. A hálózat felépítése

A hálózati interfészeket (többek között) az `ifconfig` paranccsal konfigurálhatjuk. Beállíthatjuk vele a hálózati címet amire az interfész hallgatni fog, a hálózati maszkot, és egyéb paramétereket, mint például a maximális csomagméretet, hogy pont-pont kapcsolatunk van-e, stb.

A konfiguráció lekérdezését egy paraméterezetlen `ifconfig` paranccsal oldhatjuk meg:

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:BA:EA:D6:FA
          inet addr:192.168.10.3  Bcast:192.168.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:60127 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64215 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:3 Base address:0xb800

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3904  Metric:1
          RX packets:144794 errors:0 dropped:0 overruns:0 frame:0
          TX packets:144794 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

A listán láthatjuk, hogy két interfészünk van: egy *eth0* nevű, ez az első ethernet kártya, az *lo* pedig a „loopback” interfész, ez egy tisztán szoftveres interfész, amivel egy gépen belül is lehet hálózati szolgáltatásokat használni. A listából egyszerű, forgalommal kapcsolatos mutatókat is kiolvashatunk, mint a küldött és fogadott csomagok száma, az ütközések száma, és a hibás csomagok száma.

A beállításokat szintén az `ifconfig` paranccsal végezhetjük, ez ethernet esetén tipikusan így néz ki:

```
ifconfig eth0 192.168.10.1 netmask 255.255.255.0 up
```

Ezzel felkonfiguráltuk az interfészünket, 192.168.10.1-es IP címet kapott, 255.255.255.0 hálózati maszkkal. Ez azt jelenti, hogy egy olyan ethernet hálózatban van a gép, ahol az IP címek 192.168.10.1-től 192.168.10.254-ig terjednek, a 192.168.10.0 cím magának a hálózatnak a címe, míg a 192.168.10.255 cím a *broadcast* cím, azaz ha erre a címre küldünk csomagokat, akkor azt mindegyik gép megkapja a hálózaton. Ezek után, hogy az interfésszel megvagyunk, jöhet a *routeolás*. Meg kell mondanunk a kernelnek, hogy egy célgép (vagy célhálózat) elérése érdekében melyik csomagot merrefelé továbbítsa (ez újabb rendszereken felesleges lehet, mert automatikusan megtörténik az `ifconfig` hatására). Ezt a `route` paranccsal tehetjük meg, ehhez hasonló módon:

```
route add -net 192.168.10.0 netmask 255.255.255.0 eth0
```

Ennek hatására a 192.168.10.0/255.255.255.0 hálózatra menő csomagokat a rendszer az *eth0* interfész felé fogja küldeni. Innentől már látjuk a lokális hálózatot, elkezdhetünk rajta forgalmazni, de a világba még nem látunk ki. Ehhez kell még egy *default route*, amin keresztül az összes, explicit módon meg nem adott hálózatot elérhetjük. Ehhez szükségünk lesz a mi hálózatunkat a világgal összekötő router IP címére. Ha ez mondjuk a 192.168.10.254, akkor a

```
route add default gw 192.168.10.254 eth0
```

paranccsal állíthatjuk ezt be.

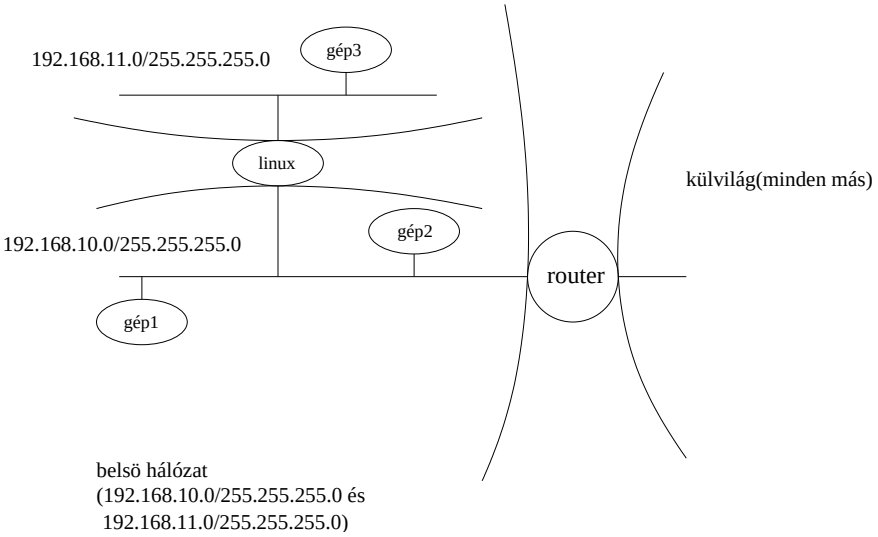
Ha több interfész van a számítógépünkben, akkor természetesen az összes interfészünket be kell így állítanunk. *Default route* csak egy kell.

Ha a gépünk *router*, akkor az

```
echo 1 >/proc/sys/net/ipv4/ip_forward
```

paranccsal be kell kapcsolnunk az interfészek közti csomagtovábbítást.

Az 1.2. ábra egy ilyen felállást mutat.



1.2. ábra. Egy másik hálózat felépítése

Ha úgy tűnik, hogy minden rendben van, akkor (vagy akár konfigurálás közben is) a `ping` paranccsal tesztelhetjük le a hálózatunkat. A `ping` paraméterként egy gép nevét, vagy IP címét várja. A célgépnek olyan (*icmp echo request*) csomagokat küld, amelyekre az válaszol (*icmp echo reply*ekkel), így le lehet tesztelni, hogy az adott gépet látja-e a miénk.

Elképzelhetőek bonyolultabb *routeolást* használó konfigurációk, például *dinamikus route*olás, *policy routing*, de ezekkel most terjedelmi okok miatt nem foglalkozunk.

1.1.2. Egyéb interfészekkel kapcsolatos beállítások

A `/proc` alatt található különleges *proc* fájlrendszer segítségével tovább finomíthatjuk beállításainkat. Ez egy olyan fájlrendszer, amiben a fájlok a rendszer állapotát tükrözik, különböző lekérdezéseket és beállításokat hajthatunk végre a segítségével, a szokásos fájlkezelő parancsokat használva. Most csak a hálózat szempontjából foglalkozunk ezzel. Itt beállíthatjuk, hogy a gépünk ne válaszoljon a *broadcast* címére küldött *ping* csomagokra (vagy hogy egyáltalán ne válaszoljon rájuk), megadhatjuk, hogy milyen sűrűn adjon választ a *ping* csomagokra (mondjuk századmásodpercenként kettőre válaszoljon), hogy ne *route*oljon az interfészei közt, és hogy próbálja kiszűrni a hamis IP csomagokat. Ezeket a következő parancsokkal állíthatjuk be:

- `echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts`

  A *broadcast*ra ne válaszoljon. 1 engedélyezi a tiltást, a 0 tiltja a tiltást, azaz engedélyezi a *broadcast ping*re történő választ.

- `echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all`

Egyáltalán ne válaszoljon a *ping*re. A számok jelentése ugyanaz, mint a *broadcast*os beállításnál.

- `echo 2 >/proc/sys/net/ipv4/icmp_echoreply_rate`
  Századmásodpercenként két *ping* csomagra válaszol.
- `echo 0 >/proc/sys/net/ipv4/ip_forward`
  A 0 letiltja a csomagok továbbítását az interfészek között.
- `echo 1 >/proc/sys/net/ipv4/conf/all/rp_filter`
  Az 1 engedélyezi a hamis csomagok szűrését.

Természetesen ezeken a beállításokon az igényektől függően változtatni kell, például egy *router*en nem jó ötlet az interfészek közti csomagtovábbítást letiltani. Mivel ezek a beállítások rendszerleállításkor elvesznek, érdemes őket valamelyik induláskor lefutó szkriptbe beletenni, a hálózati interfészeket bekonfiguráló szkript elé.

1.1.3. Egyéb, magasabb szintű beállítások

Hogy jól tudjuk használni a hálózatunkat, be kell állítanunk, hogy a gépünk milyen *nameserver*eket használjon a számítógépnév–IP cím feloldáshoz.

Az interneten a számítógépeknek az IP cím mellett nevük is van, ez a címtér hierarchikus felépítésű. A név – IP cím oda-vissza történő leképezést *nameserver*ek szolgáltatják. Ezek az adatbázisuk alapján meg tudják csinálni a leképezést, vagy ha nem, akkor a kérést továbbítják egy másik szervernek, ami válaszol majd. A gépek nevét gép (host) és domain nevekre lehet osztani. Például a www.ceg.hu névben a www a gép neve, a ceg.hu pedig a domain.

Ennek a szolgáltatásnak a használatához a `/etc/resolv.conf` fájlt kell megfelelően kitölteni. Itt a szerverek IP címe mellett azt is beállíthatjuk, hogy a gépünk milyen *domain*ekben keresse a gépeket, ha nem a teljes nevükkel keressük őket. Egy konkrét példa: a *nameszervereink* 192.168.100.22, 192.168.22.33; a domain nevünk *proba.hu*, és a *masik.hu* domain-beli gépeket is használni szeretnénk úgy, hogy nem a teljes (például *gepneve.masik.hu*) nevükkel, hanem csak a gép nevével hivatkozunk rájuk.

```
# /etc/resolv.conf
domain proba.hu
search masik.hu
nameserver 192.168.100.22
nameserver 192.168.22.33
```

Ezek után a hálózatos programokban például az *alma.proba.hu* gépre hivatkozhatunk *alma* néven, a *korte.másik.hu* gépre pedig *korte* néven.

1.2. Hálózati szolgáltatások

1.2.1. *Inetd*ből futó szolgáltatások

Az `inetd` egy különleges szerver program, egyetlen feladata, hogy más programokat indítson a megfelelő kérések hatására, például `ftp`, `telnet`, `pop3` szervereket lehet vele futtatni. A konfigurációját a `/etc/inetd.conf` fájlban tartja, ami – UNIX-os szokásokhoz híven – egy sima szöveges fájl.

A futtatandó szolgáltatásokat soronként lehet megadni a konfigurációs fájlban, a # jellel kezdődő sorok megjegyzést jelentenek, ezeket figyelmen kívül hagyja a program. Az `inetd` konfigurációs fájljában a sorok szerkezete a következő:

```
szolgáltatásport sockettípus protokoll wait/nowait szerver-program program argumentumok
```

Ahol a *szolgáltatásport* egy a `/etc/services` fájlban megtalálható port szimbolikus neve, a *sockettípus* tipikusan `stream` vagy `dgram` lehet aszerint, hogy az illető szolgáltatás *TCP* vagy *UDP* kapcsolatokat használ-e. A protokoll `tcp` vagy `udp` lehet, hasonlóan az előbbi meggondolásokhoz. A `wait`/`nowait` azt jelenti, hogy az `inetd` indítson-e új kiszolgáló processzt ha új kérés fut be, vagy a már futó szerver képes egyszerre több klienst kezelni. Ezek alapján néhány példa `inetd` szolgáltatásokra:

```
# Az ftp démon, az ftp portról rootként fut, egy
# példány csak egy kapcsolatot tud kiszolgálni.
ftp             stream  tcp     nowait  root    /usr/sbin/in.ftpd  in.ftpd
# Az ident démon identd felhasználói azonosítóval fut, és egy példány
# több kérést is ki tud szolgálni. Mindkét program TCP-t használ.
ident           stream  tcp     wait    identd  /usr/sbin/identd   identd
```

Manapság egyre kevesebb programot szoktak `inetd` segítségével futtatni, érdemes minden olyan szolgáltatást kikommentezni amire nincs szükségünk.

Ennek ellenére az `inetd` olyan szempontból érdekes, hogy jóval egyszerűbb egy `inetd`-t használó szerverprogramot megírni, mint egy önállót.

1.3. Egyéb szolgáltatások

1.3.1. *SSH*

Az SSH titkosított terminálkapcsolatot és sok minden mást tud szolgáltatni, ezért érdemes vele foglalkozni. Mi az *OpenSSH*val fogunk foglalkozni, mert ez ingyenes, szabadon felhasználható. Az `ssh` nagy vonalakban úgy működik, hogy a kliens és szerver oldal egy kétkulcsos titkosítás segítségével megegyezik egy egykulcsos titkosításban, aztán minden kommunikációt e felett a titkosított csatorna felett végeznek. Így az adataink biztonságban vannak a hálózat lehallgatásától, a TCP kapcsolatok eltérítésétől.

A szerver a konfigurációs fájlját általában a `/etc/ssh/sshd_config` fájlban, vagy a `/usr/local/etc/` alatt tartja. A konfigurációs fájl egyszerű opció-érték párokból áll. Példa egy felkommentezett konfigurációs fájlra:

```
# A 22-es alapértelmezett ssh porton fog hallgatni a szerver.
Port 22
# A kettes verziójú protokollt fogja először
# megpróbálni, aztán a régi egyest.
Protocol 2,1
# Alapértelmezés szerint az összes hálózati
# interfészen várja a kéréseket.
# ListenAddress 0.0.0.0
# A host kulcsok, ezek alapján tudnak a kliensek meggyőződni,
# hogy a szerver az, akinek gondolják. Ha a szerver kulcsa
# megváltozik, a kliensek panaszkodni fognak, mert lehet,
# hogy a szervert feltörték.
HostKey /etc/ssh/ssh_host_key
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
# A kapcsolatoknál használt RSA kulcs 768 bit hosszú.
# Aki paranoid, az magasabb értéket is megadhat itt.
ServerKeyBits 768
```

```
# A szerver lezárja a kapcsolatot, ha a kliensnek 600
# másodperc alatt sem sikerült az authentikáció.
LoginGraceTime 600
# A szerver 3600 másodpercenként generálja újra a
# kapcsolatokhoz használt RSA kulcsot. Esetleg le
# lehet csökkenteni kisebb értékre, a mai modern
# processzorok nagyon gyorsan tudnak kulcsot generálni.
KeyRegenerationInterval 3600
# Rootként nem lehet távolról bejelentkezni.
# Ez nem minden esetben szerencsés.
PermitRootLogin no
# A felhasználók fájljainak hozzáférési jogát
# szigorúan ellenőrizzük.
StrictModes yes
# Az X átirányítását bekapcsoljuk, ennek segítségével
# titkosított csatornán hozhatjuk át a grafikát a hálózaton.
X11Forwarding yes
X11DisplayOffset 10
# A nap üzenetét és az utolsó bejelentkezést nem írjuk ki.
# Ezt PAM segítségével is el lehet intézni, erről
# később lesz szó.
#PrintMotd no
#PrintLastLog no
KeepAlive yes

# Naplóírás
# A következő kettő az 1.11. rész
# elolvasása után teljesen érthető lesz.
SyslogFacility AUTH
LogLevel INFO

# A .rhosts-os authentikációt letiltjuk,
# mert nem biztonságos.
RhostsAuthentication no
#
# A .rhosts fájl és a távoli gépek host-kulcsaival
# történő authentikációt is letiltjuk.
RhostsRSAAuthentication no

# Az RSA kulcsos authentikációt engedjük, az jó.
# Ehhez kulcsokat kell generálni, erről is lesz szó.
RSAAuthentication yes

# Az egyszerű jelszavas authentikációt is engedjük.
PasswordAuthentication yes
PermitEmptyPasswords no

# A kerberosos beállítások most nem érdekelnek minket.
#KerberosTgtPassing yes

# Ezt kommentezhetnénk ki, hogy az sshd
# bejelentkezéskor kiírja, hogy jött-e levelünk.
# A PAM ezt úgyis elintézi nekünk.
#CheckMail yes
# A következő bekapcsolása esetén az interaktív
# bejelentkezésekhez a login programot használhatnánk.
#UseLogin no

# Az sftp-t szintén engedjük. Ezzel ftp-szerű, de
# titkosított adatátvitelt valósíthatunk meg.
Subsystem sftp /usr/lib/sftp-server
```

A konfigurálás után az `sshd` paranccsal indíthatjuk el az `ssh` démont. A fenti beállítások valószínűleg a legtöbb esetben jók lesznek, de azért érdemes jól átgondolni, hogy minden megfelel-e az igényeinknek.

Az `ssh` parancsot alapvetően az

```
ssh távolifelhasználó@távoli.gép [parancs]
```

módon használhatjuk, ennek hatására az `ssh` megpróbál bejelentkezni a távoli gépre, ott végrehajtani a parancsot úgy, hogy a távoli parancs kimenete és bemenete a lokális gépen futó `ssh` kimenetére és bemenetére kerül. Például:

```
# A debi nevű gépen az ls parancsot akarom futtatni
$ ssh debi ls
gabor@debi's password:
GNUstep
NetPIPE.out
...
$
```

Az `ssh` parancsnak rengeteg kapcsolója van, ezek közül egyelőre a `-v`-t érdemes megemlíteni, ennek hatására „bőbeszédű" lesz az `ssh`. Ez segíti a hibakeresést. A többiről menet közben lesz szó.

Az `ssh` csomagban van egy `scp` parancs, ezzel biztonságos távoli fájlátvitel valósítható meg. Az `scp` parancsot

```
scp távolifelhasználó@távoli.gép:távolifájl lokálisfájl
```

vagy a

```
scp lokálisfájl távolifelhasználó@távoli.gép:távolifájl
```

paraméterekkel hívhatjuk meg. Az utóbbi esetben több lokális fájlt is megadhatunk, de ekkor a távoli gépen egy könyvtárat kell megadnunk. Ha az `scp` parancsnak a `-r` kapcsolót adjuk meg, akkor rekurzív fájlmásolást fog csinálni.

Az alapvető

```
ssh távolifelhasználó@távoli.gép
```

és a

```
scp file felhasználó@távoli.gép:távoli/file
```

parancsokon kívül érdemes megismerkedni az *RSA* kulcsok használatával.

Az *RSA* kulcsok segítségével kényelmesebbé, és jobban automatizálhatóvá lehet tenni az `ssh`-t.

A kliensen az `ssh_keygen` parancs segítségével tudunk kulcsot generálni, majd a kulcs publikus felét a távoli gépen a home könyvtárunkban lévő `.ssh/authorized_keys` vagy `.ssh/authorized_keys2` fájlba másoljuk aszerint, hogy milyen verziójú `ssh` csomagot használunk. Ezek után az authentikációhoz tudjuk majd használni ezt a kulcspárt, azaz ezzel tudjuk azonosítani magunkat a szervernek.

```
# RSA típusú kulcsot generálunk.
[gabor@debi:~]$ ssh-keygen -t rsa
# Most dolgozik az ssh-keygen.
Generating public/private rsa key pair.
# Megkérdezi, hogy hova tegye le a kulcsot.
Enter file in which to save the key (/home/gabor/.ssh/id_rsa): /home/gabor/.ssh/ujkulcs
# Megkérdezi kétszer a jelszót a kulcshoz.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
# Elmondja, hogy hova fogja lerakni a kulcs
# titkos és nyilvános felét.
Your identification has been saved in /home/gabor/.ssh/ujkulcs.
Your public key has been saved in /home/gabor/.ssh/ujkulcs.pub.
The key fingerprint is:
5f:d2:e0:68:15:de:49:a7:3d:93:19:c7:45:9e:bf:9f gabor@tonhal
# Végül shell átirányításokkal és ssh használattal
# letesszük a kulcs nyilvános felét a szerveren.
[gabor@debi:~]$ ssh tonhal 'cat >>.ssh/authorized_keys2' <.ssh/ujkulcs.pub
gabor@tonhal's password:
[gabor@debi:~]$
```

Ezután már használhatjuk a kulcsunkat, vagy úgy, hogy az `ssh` parancsnak a `-i` kapcsoló után megadjuk a `~/.ssh/ujkulcs` fájlnevet, vagy hogy az `ssh-agent` programot használjuk.

Az `ssh-agent` egy olyan program, ami tárolni tudja a kulcsokat és a hozzájuk tartozó jelszót. Így a jelszót csak egyszer kell megadni (egy kulcshoz). Ugyanakkor érdemes belegondolni, hogy mivel a jelszót és a kulcsot is tárolja az ügynök, ezért ha a gépünket feltörik, és pont fut az `ssh-agent`, könnyen tovább tudnak menni a többi gépre is.

Az `ssh-agent` parancsot kétféleképpen lehet használni: vagy megadjuk neki egy futtatható program nevét, amit elindít, és addig fut amíg a program is, vagy pedig a `shell eval` parancsát hívjuk segítségül, de ekkor explicit módon `kill`-ezni kell az `ssh-agent`-et, és ezt könnyű elfelejteni. Példa a két használati módra:

- az első:

```
# Azt akarjuk, hogy az ssh-agent addig fusson, amíg az
# új zsh shellünk fut.
$ ssh-agent zsh -l
# Elindult a zsh, fut az ssh-agent,
# a fortune is lefutott.
"This generation may be the one that will face Armageddon."
-- Ronald Reagan, "People" magazine, December 26, 1985
# Ez a "belső" zsh promptja, innentől használhatjuk az
# ssh-agent szolgáltatásait.
# Ha kilépünk ebből a shellből, az agent is kilép.
$
```

- a második:

```
# Ilyenkor így kell elindítani.
$ eval ‘ssh-agent‘
# Kiírja, hogy elindult.
Agent pid 1312
# És innentől használhatjuk is.
$
# De el ne felejtsük lelőni!
$ eval ‘ssh-agent -k‘
Agent pid 1312 killed
```

Ezután egy összetettebb példa: elindítjuk az `ssh-agent`-et, megadjuk neki az új kulcsunkat (ezt az `ssh-add` paranccsal tehetjük meg), és megnézzük mi történik, ha a távoli gépen akarunk valamit futtatni:

```
# Szokásos ssh-agent indítás.
[gabor@debi:~]$ ssh-agent zsh -l
# Megadjuk neki a kulcsunkat.
[gabor@debi:~]$ ssh-add .ssh/ujkulcs
# Az ssh-add kéri a jelszót a kulcshoz.
Need passphrase for .ssh/ujkulcs
Enter passphrase for .ssh/ujkulcs:
# A kulcs átadása az ssh-agentnek sikeres.
Identity added: .ssh/ujkulcs (.ssh/ujkulcs)
# Kipróbájuk, mit csinál az ssh.
[gabor@debi:~]$ ssh tonhal uname -a
# Nem kér jelszót.
Linux tonhal 2.4.2 #5 Mon Feb 26 22:13:40 CET 2001 i686 unknown
```

Az *RSA* kulcsokat használhatjuk különböző automatikus dolgok végrehajtására is, ha nem adunk meg nekik jelszót a generáláskor. Ekkor minden jelszókérés nélkül lépkedhetünk a gépek között. Ez így persze veszélyes. Hogy a veszélyt csökkentsük, a

távoli gép `.ssh/authorized_keys2` fájljában megadhatjuk, hogy honnan használhatják azt a kulcsot, és hogy milyen programot futtathatnak (és még sok minden mást). Például az *ujkulcs* nevű kulcsunkat csak a *debi* nevű gépről engedjük be, és csak az `uptime` parancsot lehet vele futtatni:

```
# Megnézzük, hogy milyen az .ssh/authorized_keys2 fájl.
[gabor@tonhal:~]$ cat .ssh/authorized_keys2
from="debi",command="uptime" ssh-rsa AAAAB3N...
# Csak a debi gépről lehet használni, csak az uptime
# parancsot lehet futtatni. Utána az "ssh-rsa AAAAB3N.."
# már maga a kulcs.
[gabor@debi:~]$ ssh tonhal /bin/true
 10:51pm  up  1:40,  5 users,  load average: 0.02, 0.02, 0.00
```

Mint látjuk a `/bin/true` helyett az `uptime` parancs futott le. Egy összetett példa: a fentieket kombináljuk úgy, hogy az adott kulccsal csak egy fájlt tudjunk átmásolni a távoli gépre, ami a távoli gépen `/tmp/p` néven fog megjelenni. Első lépés: döntsük el, hogy milyen parancsnak kell lefutnia a távoli gépen. Erre egyszerűen rájöhetünk, ha az `scp` parancsot `-v` kapcsolóval futtatjuk, és egy *command* tartalmú sorra keresünk benne:

```
# Kipróbálás.
[gabor@debi:~]$ scp -v valami.ps tonhal:/tmp/p
# Itt van ami kell nekünk, a scp -v -t /tmp/p
# fut le a másik oldalon.
Executing: program /usr/bin/ssh host tonhal, user (unspecified), command scp -v -t /tmp/p
Sending file modes: C0664 26632 valami.ps
valami.ps              100% |****************************| 26632       00:00
```

Láthatjuk, hogy a távoli gépen az `scp -t /tmp/p` parancs fog lefutni. Ezután megfelelően átírjuk a távoli gépen a `.ssh/authorized_keys2` fájlt, és már működik is. A `-v` opciót kihagyjuk, azt a bőbeszédű mód miatt küldte.

```
# Leellenőrizzük a .ssh/authorized_keys2 tartalmát.
[gabor@tonhal:~]$ cat .ssh/authorized_keys2
from="debi",command="scp -t /tmp/p" ssh-rsa AAAAB3...
# Ezzel a kulccsal tényleg csak a megfelelő
# parancsot lehet használni.
# Most megpróbálunk mást futtatni a túloldalon,
# de ez nem fog sikerülni.
[gabor@debi:~]$ scp proba tonhal:/tmp/a
proba                  100% |****************************|   259       00:00
```

Nyilvánvalóan *tonhal* nevű gépen `/tmp/a` helyett `/tmp/p` fájlban fog megjelenni a *proba* fájl tartalma.

1.3.2. *Qmail*

Miért pont a `qmail`? A `qmail` egyszerűen konfigurálható, biztonságos, és eléggé jó teljesítményű. Ugyanakkor érdemes megjegyezni, hogy van egy-két hiányossága, például túl későn ellenőrzi, hogy van-e adott nevű felhasználó vagy alias a rendszerünkön; minden egyes kimenő levélhez új kapcsolatot nyit; a licence pedig lehetne kicsit barátságosabb. Érdemes megnézni a `postfix` programot (http://www.postfix.org/), de annak jóval bonyolultabb a konfigurálása.

A `qmail` biztonsági okok miatt több, különböző felhasználó azonosítóval futó processzre van felosztva, így ha netalán valamelyik részével valami baj történne, az akkor sem lesz komoly hatással a teljes rendszerre. Az *MTA*k közt ritka a hasonlóan biztonságos tervezés.

Telepítés: a `qmail` csomagot le lehet tölteni a http://www.qmail.org/ oldalról, vagy feltelepíthetjük disztribúciónk részeként, *.deb*, *.rpm* vagy egyéb csomagból. Ha van ilyen csomag, érdemesebb azt használni, mint magunknak szerezni. A `qmail` licence miatt valószínűleg csak forrásnyelvű csomagot fogunk találni, benne az utasításokkal a fordításhoz. Az egyetlen paraméter, amire a fordításkor érdemes odafigyelni, az a `conf-split` fájlban lévő szám, ez befolyásolja, hogy a `mail queue` hány könyvtárra lesz szétbontva. Nagyon nagy forgalom esetén ezt érdemes lehet felemelni. Mivel a `qmail` hasheléssel dönti el, hogy melyik levél melyik könyvtárba fog kerülni, ezért itt érdemes prímszámot megadni, a hashelés hatékonyságának növelésére. Ezek után, ha sikerült lefordítani és feltelepíteni, akkor a konfiguráció ellenőrzése következhet, mivel a konfigurációs állományok is a fordítás során, a helyi sajátosságoknak megfelelően jönnek létre.

A `qmail` a konfigurációját eredetileg a `/var/qmail/control` alatt tartja, mindent külön fájlban az egyszerűség kedvéért. A könyvtáron a disztribúciók változtatni szoktak, a *Debian* például a `/etc/qmail` könyvtárba teszi a konfigurációs fájlokat. A fontosabb konfigurációs fájlok sorban:

A `me` a legfontosabb konfigurációs fájl, ez állítja be a gép nevét, ezt a nevet fogja alapértelmezett értékként használni eléggé sok (de nem mindegyik) konfigurációs fájl helyett (tehát ennek a kitöltésével már szinte működőképes a levelező szerver).

A `defaultdomain` fájlba kell beletenni a gép domain nevét. Például ha a gép neve *mail.ceg.hu*, akkor ebbe a fájlba *ceg.hu* kerül.

A `plusdomain` fájlba a domain név domain nevét kell tenni, a fenti példánál maradva a *hu* szót.

Az `rcpthosts` fájlban felsorolt hostnevekre jövő leveleket fogja az smtp szerver processz elfogadni, hogy ezután a `queue`-ba bekerülve azokat valahogy kézbesítse a rendszer.

A `locals` fájlban felsorolt hostnevekre jövő leveleket fogja lokálisként kezelni a szerver, tehát ebben benne kell lennie az összes olyan névnek, amire lokális kézbesítést akarunk kérni.

A `qmail` a `concurrencyremote` fájlban megadott számú processzt fog futtatni a távoli levelek kézbesítéséhez, azaz ez határozza meg, hogy egyszerre hány levelet fog párhuzamosan távolra kézbesíteni. Ez nagyban befolyásolhatja a szerverünk teljesítményét. Érdemes kikísérletezni, hogy mennyi illik legjobban a hardverhez és a terheléshez, de hetven-nyolcvanat valószínűleg elbír a hardver.

A `qmail` a `concurrencylocal` fájlban megadott számú processzt fog használni a lokális kézbesítéshez. Ezzel már jobban kell vigyázni, mert például egy sok példányban futó nagy levelezőlistaszoftver erősen megfoghatja a gépet.

Jó esetben a disztribúciónkkal jövő csomagok közt volt a `qmail`, és akkor van hozzá indítószkript is, de ha nem, akkor nekünk kell megírnunk egyet. El kell indítanunk a szerver programot, és a hálózatra hallgató *SMTP démon*t is. Az elsőt a `/var/qmail/bin/qmail-start` programmal tehetjük meg, ami első paraméterként az alapértelmezett lokális kézbesítésnek megfelelő célt várja, míg a többi paraméter a logolást befolyásolja. Egy konkrét példa:

```
exec env - PATH="/var/qmail/bin:$PATH" \
qmail-start '|preline procmail' splogger qmail
```

Az első sor csak törli a felesleges környezeti változókat, a második sor elindítja a *qmail-start*ot úgy, hogy a lokális kézbesítéshez a *procmail*t fogja használni, a logoláshoz pedig az *splogger*t, ami a sima `syslog` programon keresztül logol. A *preline* azért kell, mert a `qmail` sok helyen környezeti változókat használ ahol más program

a standard inputról várná az adatot, a *preline* pedig a környezeti változók tartalmát megfelelően kiküldi a program (itt a *procmail*) inputjára.

Ezek után jön a `qmail-smtpd` futtatása. Ezt legegyszerűbben az `ucspi-tcp` csomag segítségével tehetjük meg. Ez is letölthető a http://www.qmail.org/ oldalról, de érdemesebb a disztribúciónk csomagkezelőjével feltelepíteni. Ez a csomag egyszerű hálózatos szolgáltatásokat kínál.

```
/usr/bin/tcpserver \
   -u ‘id -u qmaild‘ -g ‘id -g nobody‘ -x /etc/tcp.smtp.cdb 0 smtp \
   /usr/sbin/qmail-smtpd 2>&1 | splogger qmail -t qmail -p mail.notice &
```

Ez elindítja a *tcpserver* programot, megmondja neki, hogy futtassa a *qmail-smtpd*t a *qmaild* felhasználó azonosítójával, és *nobody* csoportazonosítóval, a */etc/tcp.smtp.cdb* fájlban lévő szabályokat használja, a gép bármelyik IP címére fogadjon el kapcsolatot (a 0 az), és az *smtp* porton hallgasson.

A `relay`-ezést a `qmail-smtpd` programnak a `RELAYCLIENT` környezeti változó beállításával engedélyezhetjük, akár `inetd`-ből `tcpwrapper` segítségével, akár a `tcpserver` segítségével. Az utóbbi használatához ki kell tölteni egy tetszőleges fájlt, például így:

```
127.0.0.1:allow,RELAYCLIENT=""
192.168.20.:allow,RELAYCLIENT=""
```

Ennek hatására a lokális gépről és a 192.168.20.0/255.255.255.0 hálózaton lévő gépekről küldhetnek levelet a szerveren keresztül, a szerver `rcpthosts` fájljában nem szereplő gépnek. Ebből a fájlból még egy bináris *.cdb* adatbázist kell csinálni:

```
tcprules /etc/tcp.smtp.cdb temporaryfile < szovegesfile
```

Ahol a *temporaryfile* egy ideiglenes fájl neve, a *szovegesfile* pedig annak a fájlnak a neve, amibe a fenti két sort betettük.

Aliasok kezelése: a `qmail` az aliasokat a `/var/qmail/aliases` könyvtárban tartja, mindegyiket külön fájlban. A fájlok neve `.qmail-aliasneve`, tehát a `/var/qmail/aliases/.qmail-postmaster` fájlba kell tennünk, ha például van egy *postmaster* aliasunk. A `.qmail` fájlokban a pipe jellel kezdődő sor hatására a `qmail` a pipe jel után lévő program standard inputjára fogja küldeni a bejövő leveleket. „/”-rel kezdődő sor hatására az adott *mailbox*ba vagy *Maildir*be fogja tenni a bejövő leveleket. Az utóbbiba akkor, ha megint csak egy „/” jel van a sor végén.

Ide jönne egy példakonfiguráció, ha szükséges.

1.3.3. *vsftpd*

Az `ftp` nem egy biztonságos szolgáltatás, de sajnos a felhasználók nagyon ragaszkodnak hozzá annak ellenére, hogy az `ssh` csomagnak van *Secure FTP*je, titkosított adatávitellel, grafikus klienssel *Windows*hoz. Az `ftp` problémája, hogy bizonyos műveletekhez az egyszerű felhasználótól magasabb privilégium-szint kell, és ráadásul a forgalmat sem titkosítja (azaz a hálózat lehallgatásával össze lehet gyűjteni a felhasználók jelszavait). További probléma, hogy kevés igazán biztonságos `ftp` szerver létezik.

Egy új `ftp` szerver a `vsftpd`, a *Very Secure FTP Daemon*. A név természetesen semmit sem garantál, csak azt, hogy az írója mindent megtesz a biztonság érdekében. Számomra ez a szerver biztonságosabbnak tűnik a többinél, ezért ezt ismertetem.

A szerver `inetd`-ből fut, minden kapcsolathoz két processzt indít, egy privilégizáltat a különböző hálózati műveletek elvégzéséhez (amikhez a normál privilégiumszint kevés lenne), és egy másikat, ami az épp bejelentkezett felhasználó azonosítójával fut. A privilégizált processz sem *root*ként fut, hanem a Linux *capabilities* nevű mechanizmusát használva egy egyszerű felhasználóként úgy, hogy közben a jogot az 1024-től kisebb hálózati portok megnyitására megtartja. Ráadásul egy másra nem használt könyvtárba `chroot`-olja magát, így onnan „nem lát ki”. A két processz *Unix Domain Socket*en keresztül kommunikál egymással úgy, hogy a privilégizált processz nem bízik a privilégizálatlan által küldött üzenetekben. Ehhez hasonlóan tervezett `ftp` démonok ritkák.

Egy két klienst kiszolgáló szervert így képzelhetünk el:

Telepítés: a `vsftpd` csomagot a ftp://ferret.lmh.ox.ac.uk/pub/linux/ oldalon lehet megtalálni, de érdemesebb a disztribúciónkból feltenni, ha ott van a csomagok közt. A szoftver például része a *Debian Woody* disztribúciónak. Hozzá képest a kézzel fordított, vagy más disztribúciók részét képező változatok esetében csak a fájlok elérési útvonalában várható változás.

`Inetd` beállítása: a `/etc/inetd.conf` fájlba írjuk be a következő sort, majd a `killall -HUP inetd` parancsot kiadva olvastassuk újra az `inetd`-vel a konfigurációs fájlját.

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

A program egyetlen paramétert vár, a konfigurációs fájljának az elérési útvonalát, ha az nem a `/etc/vsftpd.conf` fájlban van. Ezek után természetesen a `tcpwrapper` csomagot is be kell konfigurálni, a `/etc/hosts.allow` és a `/etc/hosts.deny` fájlok megfelelő kitöltésével.

A `/etc/hosts.allow` fájlba például *anonymous ftp* esetén ilyesmit lehet írni:

```
# Beengedünk mindenkit, akinek feloldható az IP címe, kivéve ha a
# a név visszaellenőrzésénél nem egyeznek az IP címek.
vsftpd: KNOWN EXCEPT PARANOID
```

A konfigurációs fájl: `/etc/vsftpd.conf`. Ez egyszerű `opció=érték` párosokból áll. Fontos tudni, hogy nem lehet szóköz az egyenlőségjel két oldalán. A fájlban a megjegyzéseket a szokásos soreleji # jel jelzi. Elég sok opció logikai értékű, ezeket `YES`-re illetve `NO`-ra állíthatjuk, ezt nem fogom mindegyiknél külön megemlíteni.

anonymous_enable Ezzel lehet engedélyezni az *anonymous ftp*t. Az *anonymous ftp*hez szükség van egy *ftp* nevű felhasználóra. Ennek a felhasználónak a *home* könyvtára lesz az *anonymous ftp* terület.

local_enable A normál felhasználókat engedjük-e bejelentkezni vagy sem. A fenti opcióval együtt lehet csak *anonymous*, csak normál felhasználókat beengedő, vagy mindkettőt engedőre konfigurálni az `ftp` szervert.

write_enable Engedi-e az írást az `ftp` szerver (feltöltés, stb.).

local_umask Ez nem logikai értékű, azt mondja meg, hogy milyen *umask*ot használjon az új fájlok létrehozásakor. Az alapértelmezett 077 esetén például csak a tulajdonos fogja tudni elérni az általa letett fájlokat.

anon_upload_enable *Anonymous*ként lehet-e feltölteni a szerverre. Kell hozzá a `write_enable` is.

anon_mkdir_write_enable *Anonymous*ként lehet-e új könyvtárat létrehozni. Kell hozzá a `write_enable` is.

xferlog_enable Az `xfer log`ot írja-e a démon.

connect_from_port_20 A szokásos 20-as (`ftp-data`) portról menjen-e ki az adatkapcsolat. Erre érdemes igent mondani.

chown_uploads Az *anonymous*ként feltöltött fájlok tulajdonosát megváltoztassa-e a szerver.

chown_username Kinek a tulajdonává `chown`-olja a szerver az *anonymous*ként feltöltött dolgokat. Ez nem logikai típusú, az alapértelmezett érték a *root*.

idle_session_timeout A szerver mennyi idő múlva bontsa a tétlen kapcsolatokat. Nem érdemes se túl nagyra, se túl kicsire állítani. A használható értékek valahol 60 és 600 másodperc közt lehetnek.

data_connection_timeout Az adatkapcsolatokat mennyi idő múlva bontsa a szerver, ha nincs rajta forgalom. (Az utóbbi két paramétert másodpercben kell megadni.)

nopriv_user Egy nem privilégizált, az `ftp` démon működtetéséhez fenntartott felhasználó neve. Nem érdemes *nobody*t megadni, inkább új felhasználót kell hozzá létrehozni.

ascii_upload_enable, ascii_download_enable Engedélyezzük-e az *ASCII* fel/letöltést. Ez jó ötletnek tűnik, de például egy ilyen módban történő fájlméret-lekérdezéshez karakterenként végig kell olvasnia a szervernek a fájlt.

ls_recurse_enable Engedélyezzük-e a rekurzív könyvtárlistázást. Ez nagyon megterhelheti a szervert, de ugyanakkor a mirrorozó programoknak szükségük lehet rá. Némelyik így gyűjti össze a lehozandó fájlok listáját.

1.3.4. *stunnel*

Az `stunnel` segítségével titkosított *SSL* (*Secure Socket Layer*) kapcsolatba csomagolhatunk különböző protokollokat, mint például *POP3*, vagy *IMAP*. Így például *SSL*es *POP3* szerver valósítható meg, egy *SSL*t nem ismerő szerverprogram segítségével. Ezzel lehet védekezni a hálózatlehallgatás ellen.

Az `stunnel`-t lehet `inetd`-ből vagy önmagában futtatni. A két mód csak egy kapcsolóban tér el egymástól.

Az `stunnel` leggyakrabban használt kapcsolói:

-l segítségével lehet megadni, hogy milyen programot futtasson, amikor csatlakoznak hozzá. Itt egy `inetd` segítségével is futtatható programot vár (például egy `pop3` szerver).

-N segítségével adhatjuk meg, hogy a `hosts.allow` és a `hosts.deny` fájlt milyen szolgáltatásnév szerint vizsgáljon meg.

-p után adhatjuk meg az `stunnel`-nek a *PEM* fájlját. Ezzel igazolja a szerver, hogy ő valóban az, akinek állítja magát.

-d segítségével állíthatjuk át az `stunnel` működési módját `inetd`-sről különállóra. Ez egy hálózati port nevét várja paraméterként.

Ezek után már ki is lehet próbálni az `stunnel`-t, például `inetd`-ből, így:

```
# /etc/inetd.conf részlet
# A pop3s porton stunnelt futtatunk. A processz neve pop3d lesz.
# A tcpwrapper szolgáltatásnév solid-pop3d-s, a PEM fájl a
# /etc/ssl/certs/pop3.pem, a futtatott program pedig
# /usr/sbin/solid-pop3d.
pop3s  stream  tcp  nowait  root.mail  /usr/sbin/stunnel pop3d -N solid-pop3d-s \
-p /etc/ssl/certs/pop3.pem -l /usr/sbin/solid-pop3d
```

Ha ugyanezt önállóan futó `stunnel`-lel szeretnénk megoldani, akkor egy ilyen paranccsal indíthatjuk el az `stunnel`-t:

```
/usr/sbin/stunnel -d pop3s -N solid-pop3d-s \
-p /etc/ssl/certs/pop3.pem -l /usr/sbin/solid-pop3d
```

Mindkét esetben a `hosts.allow` fájlban egy hasonló bejegyzésre lesz szükségünk:

```
solid-pop3d-s: localhost, .domain.hu
```

Végül `telnet-ssl` segítségével letesztelhetjük, hogy jól működik-e:

```
$ telnet-ssl -z ssl szerverneve pop3s
+OK Solid POP3 server ready <22458.996136614@szerverneve>
USER gabor
+OK username accepted
PASS titok
LIST
+OK scan listing follows
.
QUIT
+OK session ended
Connection closed by foreign host.
```

A *PEM* fájlt az azonosságának az igazolására használja az `stunnel`. Telepítéskor generál egyet, de lehet, hogy később szükségünk lesz arra, hogy magunknak generáljunk egyet. Ezt az

```
openssl req -new -x509 -days 365 -nodes -config stunnel.cnf \
-out stunnel.pem -keyout stunnel.pem
```

paranccsal tehetjük meg. Ennek hatására az *openssl* egy jelszó nélküli, 365 napig érvényes *PEM* fájlt generál, az *stunnel.cnf* konfigurációs fájl segítségével. A kimenetet az *stunnel.pem* fájlba fogja tenni.

Amikre oda kell figyelni: az `openssl` *Common Name:* kérdésére a szerver gépnevét kell megadni.

Némelyik disztribúció elfelejti az `stunnel.cnf` fájlt az `stunnel` csomagba beletenni, ekkor letölthetjük magunknak a http://www.stunnel.org/ oldalról.

Lehet, hogy szükségünk lesz *Diffie–Hellman* paraméterekre a *PEM* fájlunkhoz, ezt az

```
openssl gendh 512 >> stunnel.pem
```

paranccsal generálhatunk hozzá.

1.4. Behívószerver konfigurálása

Behívószerverre akkor lehet szükségünk, ha valakiknek modemes internet-elérést szeretnénk biztosítani.

Behívószerver felállításához szükségünk lesz a `pppd` és az `mgetty` csomagokra. Ezek általában a disztribúciók részét képezik, nem lesz szükségünk arra, hogy magunknak fordítsuk le őket. Először az `mgetty` konfigurációjával fogunk foglalkozni.

1.4.1. *Mgetty*

Az `mgetty` egy `getty`-szerű program, de modemekhez tervezték. Valamelyik termináleszközön egy felhasználónévre vár, és utána meghívja a `login`, vagy valamilyen más, bejelentkezést kezelő programot. Ezen kívül kezelni tudja a modemek különböző funkcióit, például modem inicializálás, kicsörgések száma mielőtt felvenné, fax fogadása.

Célunk az lesz, hogy a modem valahány csörgés után fogadja a hívást, majd az `mgetty` elindítson egy `pppd`-t a soros vonalon.

Az `mgetty`t a `getty` parancshoz hasonlóan `init`-ből szokták futtatni, ehhez egy hasonló sor kell a `/etc/inittab` fájlba:

```
T1:23:respawn:/sbin/mgetty -x0 -s 115200 ttyS1
```

Ez azt jelenti, hogy az `init` processz a kettes és hármas futási szinten mindig indít egy `mgetty` programot a `ttyS1` soros portra, ha az előző `mgetty` kilép.

Látható, hogy az `mgetty` parancsnak különböző parancssori paraméterei lehetnek, de ezek közül csak azokkal fogunk foglalkozni, amelyek nekünk fontosak. Ezek a következők: `-s`, ezzel adhatjuk meg a modem sebességét, ezt érdemes 115200-ra állítani; a `-n`, ezzel lehet majd megmondani az `mgetty`nek, hogy hány kicsörgés után fogadja a hívást; a `-x`, ezzel állíthatjuk a *debug level*t, ami fontos lehet a beállítás közben. Ezen kívül még van néhány konfigurációs fájlja, amik szintén fontosak.

`/etc/mgetty/login.config`: ebben a fájlban állíthatjuk be, hogy milyen felhasználó névhez milyen `login` programot futtasson az `mgetty`. Ennek a formátuma

```
felhasználónév felhasználóazonosító utmpbejegyzés loginprogram argumentumok...
```

Így ha az adott nevű felhasználó jelentkezik be, akkor az adott felhasználóazonosítóval az adott program fog lefutni, és az `mgetty` lerak egy megfelelő bejegyzést az `utmp` fájlba.

Az `mgetty` programnak van egy *autoppp* módja, ez hasznos, mert észre tudja venni, ha *ppp démon* van a vonal másik oldalán, és el lehet vele indíttatni a *ppp démon*t ezen az oldalon. Ez a sor körülbelül így néz ki:

```
/AutoPPP/ -     a_ppp   /usr/sbin/pppd auth -chap +pap login debug
```

Ennek hatására az `mgetty` az */AutoPPP/* felhasználó érzékelésekor (azaz ha *ppp démon* van a másik oldalon) letesz egy `a_ppp` bejegyzést az `utmp` fájlba, és meghívja a `pppd` parancsot a megadott paraméterekkel.

A `-` felhasználóazonosító hatására az `mgetty` nem fog felhasználóazonosítót váltani, tehát a `pppd` is `root`-ként fog futni, mint az `mgetty`.

`/etc/mgetty/mgetty.config`: itt körülbelül ugyanazokat lehet beállítani mint a parancssorban, megadhatjuk a *debug level*t, a modem sebességét, stb. Érdemes odafigyelni, hogy a modemünket csak adatmodemként konfiguráljuk, és a *fax-id*t kapcsoljuk ki. Ha van *fax-id* sorunk, kommentezzük ki. A *debug levelt* egy `debug X` sorral állíthatjuk, 0 és 9 között. `speed XXXX` opcióval állíthatjuk be a modem sebességét, a `data-only yes` vagy `modem-type data` opcióval állíthatjuk be, hogy nem akarunk faxot kezelni. Ha különböző típusú modemjeink vannak, akkor portonként beállíthatjuk ezeket, `port portneve` (például `port ttyS1`) sorokkal választhatjuk el egymástól a különböző portokra vonatkozó bejegyzéseket.

Példa egy minimális `mgetty.config` fájlra:

```
# Négyes szintű nyomkövetés.
debug 4
# 115200 bauddal kommunikálunk a modemmel.
speed 115200
# A bejelentkezés előtt a /etc/issue.mgetty
# fájlt küldi ki a soros vonalon.
issue-file /etc/issue.mgetty
# Adatmodemünk van, faxot nem akarunk.
modem-type data
```

Ha azt akarjuk, hogy az `mgetty` ne mindig vegye fel a telefont, `cron` segítségével egy `/etc/nologin.ttyS1` (vagy más, a soros portnak megfelelő nevű) fájl létrehozásával és törlésével ezt szabályozhatjuk. Ehhez, és a kicsörgések számának megadásához a modem automatikus hívásfogadásának a kikapcsolásához lehet, hogy át kell konfigurálnunk a modemet. Ehhez fel kell raknunk valamilyen terminálkezelő szoftvert, mint például a `minicom`. Ha a `minicom` csomagot sikerült telepítenünk és beállítanunk, akkor a

```
# A modemet bőbeszédűre állítjuk.
ATV1
# Nem akarjuk, hogy a modem automatikusan
# felvegye a telefont.
AT&M1
# Elmentjük a modem konfigurációját.
AT&W
```

parancsokkal ki tudjuk kapcsolni az automatikus telefonfelvételt, be tudjuk kapcsolni a modem „bőbeszédű” módját, hogy az `mgetty` tudja, hogy mikor csörög ki a modem, és végül elmentjük a beállításokat.

Az `AT&M1` helyett lehet, hogy az `AT&M3` parancsra lesz szükségünk az automatikus telefonfelvétel letiltására.

1.4.2. *PPPD*

A *PPP* (*Point to Point Protocol*) segítségével lehet soros, vagy más pont-pont felépítésű vonalon hálózati kommunikációt megvalósítani. A `pppd` egy *PPP* vonal menedzseléséhez szükséges különböző szolgáltatásokat nyújtja, mint például a végpontok közti authentikáció, kommunikációs paraméterek, tömörítés, stb. beállítása.

Ahhoz, hogy a modemes behívószerverünk működjön, minimálisan a következő szoftverekre lesz szükség: egy kernel soros port és *ppp* támogatással, `mgetty` és `pppd`. Az utóbbi két szoftver verzióját érdemes a `kernel-forrás/Documentation/Changes` fájlban leírt verziószámhoz igazítani, különben problémákba ütközhetünk.

A `pppd` a konfigurációját a `/etc/ppp/options`, a `~/.ppprc` és a `/etc/ppp/options.terminálnév` fájlokból szedi, ahol a *terminálnév* a terminál neve, amit a `pppd` parancsnak a parancssorában megadtunk. A fájlok megvizsgálása után a parancssort nézi meg.

A konfigurálás eléggé egyszerű lesz. Kitöltjük a `/etc/ppp/options` fájlt, jelszavakat adunk a felhasználóknak, körülbelül ennyi az egész.

```
# Az /etc/ppp/options körülbelül így néz ki:
###
# Sima soros vonalunk van, nem kell semmilyen karaktert
# különlegesen kezelni.
asyncmap 0
# A kliensnek authentikálnia kell magát, ha pppt akar
# használni.
auth
# Csak chap ("kihívásos") authentikációt használunk,
```

```
# a papot ("jelszó authentikáció") tiltjuk.
# Ezen persze az igényeknek megfelelően lehet
# változtatni.
refuse-pap
require-chap
# Hardware flow controlt használunk.
crtscts
# Uucp-s lockolást használunk a  sorosporti
# eszközön, nehogy más is használni próbálja.
lock
# Nem akarjuk, hogy a naplófájlokban
# megjelenjenek a jelszavak.
hide-password
# A sorosvonalon egy modem ül.
modem
# Proxyarp-ot használunk a csomagok routolásánál,
# erről később még lesz szó.
proxyarp
# Echo-requesteket küldünk a másik
# oldalnak 30 másodpercenként, ha
# nem jön válasz, valami baj van.
lcp-echo-interval 30
# Ha a másik oldal 4 ilyenre nem válaszol, mert valami
# rossz történt vele, akkor bontjuk a kapcsolatot.
# Nagyon rossz vonalnál lehet, hogy megéri nagyobbra venni.
lcp-echo-failure 4
# Ipxet nem akarunk.
noipx
# Megadjuk a ppp link távoli és lokális felének IP címét.
# A lokális fele a 192.168.10.10, ez egy tetszőleges,
# az adott szerver gépen és a hozzá kapcsolódó
# hálózatokon nem használt cím lehet, ezzel nem
# fognak csomagok kikerülni a hálózatra.
#
# A távoli felének egy olyan IP címnek kell lennie, ami
# a szerver gép belső lokális hálózatának egy nem
# használt IP címe. Ebben az esetben például a szerver
# eth0 interfésze a 192.168.22.0/255.255.255.0
# hálózatra csatlakozik, és 192.168.22.33 egy nem
# használt IP ezen a hálózaton.
#
# Ez a proxyarphez kell.
#
# Több ilyen IP cím párost is megadhatunk itt.
#
# Ha több modemünk van, akkor érdemes vonalanként
# egy-egy egyéni IP cím párost megadni, a
# megfelelő options.ttySxx fájlokban.
192.168.10.10:192.168.22.33
# A következő két sorban szinte az összes
# lehetséges tömörítést bekapcsoljuk. Ez
# nagyban megyorsítja például a html fájlok
# letöltését.
bsdcomp 15,15
deflate 15,15
```

Ezek után fel kell vennünk a felhasználókat a `/etc/ppp/chap-secrets` fájlba:

```
kliens  szerver   titkosjelszó IPcím
```

Ahol értelemszerűen a *kliens* a kliens neve, a *szerver* a szerverünk neve, a *titkosjelszó* a kliens jelszava, az *IPcím* pedig az az IP cím, amit a kliens használhat, vagy *, ha bármilyen IP-t használhat, amit megadtunk a `pppd options` fájljaiban. Itt így felvehetjük a felhasználóinkat.

Mit csinál a `proxyarp`? A `proxyarp` kulcsszó hatására a `pppd` tesz egy bejegyzést a szerver első ethernet interfészének `arp` táblájába, hogy ha a kliens hardvercímét szeretné tudni valamelyik gép a szerver hálózatán, akkor a szerver hardver címét

kapja meg hozzá. Ennek hatására a kliensnek szánt hálózati csomagok először eljutnak a szerverhez, majd az továbbroutolja őket a klienshez. A routolás jól fog működni, mert a kernel mindig a „pontosabban” megadott útvonal felé fogja küldeni a csomagokat, tehát a 255.255.255.255-ös hálózati maszkú *ppp* linkról a csomagok jól átjutnak a (mondjuk) 255.255.255.0-s külső hálózat felé, annak ellenére, hogy a kliens IP címe a külső hálózatnak is részét képezi.

IP cím hozzárendelése soros vonalakhoz: tegyük fel, hogy két soros vonalunk van, `ttyS0` és `ttyS1`. Ekkor az `inittab` például így fog kinézni:

```
...
T0:23:respawn:/sbin/mgetty -x0 -s 115200 ttyS0
T1:23:respawn:/sbin/mgetty -x0 -s 115200 ttyS1
...
```

Ekkor a megfelelő opció fájlok kitöltésével elérhetjük, hogy a két soros vonalon futó `pppd` más IP címet kapjon:

```
#/etc/ppp/options.ttyS0
#
192.168.10.10:192.168.22.33
#/etc/ppp/options.ttyS1
#
192.168.10.11:192.168.22.34
```

A `pppd` különböző szkripteket futtat a *ppp* kapcsolat kezdetekor és végekor, ezekkel sok mindent meg lehet oldani, például tűzfal átkonfigurálás, forgalommérés. A `pppd` a `/etc/ppp/ip-up` szkriptet futtatja amikor a *ppp* link használhatóvá válik, és az `/etc/ppp/ip-down` szkriptet, amikor a kapcsolatnak vége szakad. Hasonlóan futtatja le az `/etc/ppp/auth-up` és a `/etc/ppp/auth-down` szkripteket, ha volt authentikáció. Ezeket a programokat különböző paraméterekkel és környezettel futtatja le, amikből a *ppp* link különböző paramétereit tudhatjuk meg:

Környezeti változók, amiket a szkriptek megkapnak:

DEVICE A terminále szköz neve amit a `pppd` használ.

IFNAME Az interfész neve amit a `pppd` használ. Például `ppp0`, `ppp1`.

IPLOCAL A link lokális felének IP címe.

IPREMOTE A link távoli felének IP címe.

PEERNAME A másik oldal neve, ha történt authentikáció.

SPEED A vonal sebessége.

ORIG_UID, PPPLOGNAME A *ppp démon*t futtató felhasználó eredeti felhasználóazonosítója és felhasználóneve.

Az `ip-down` és `ip-up` szkriptek még négy változót kapnak:

CONNECT_TIME A kapcsolat hossza másodpercben.

BYTES_SENT A soros porton elküldött byte-ok száma.

BYTES_RCVD A soros vonalon fogadott byte-ok száma.

LINKNAME A link logikai neve. A `linkname` kulcsszóval lehet valamelyik opciófájlban beállítani.

Parancssori argumentumok, amiket a szkriptek a `pppd` parancstól kapnak: a `/etc/ppp/auth-up` és a `auth-down` szkriptek az *interfész-név*, *másikoldal-neve*, *felhasználó-név*, *terminál-eszköz*, *sebesség* paramétereket kapják, a `/etc/ppp/ip-up` és `ip-down` szkriptek pedig az *interfész-név*, *terminál-eszköz*, *sebesség*, *lokális IP cím*, *távoli IP cím*, *ipparam* paramétereket kapják meg. Az `ipparam` egy sztring, amit valamelyik opciófájlban adhatunk meg, de ez nem kötelező.

1.4.3. *PPP ssh* felett

Néha szükség lehet arra, hogy két vagy több távoli gép között biztonságos adatforgalmat bonyolíthassunk úgy, hogy az egy nem megbízható hálózaton megy át. Ezzel úgynevezett *VPN*t (*Virtual Private Network*) alakíthatunk ki például két iroda közt.

Ilyenek kialakítását sokféleképpen meg lehet oldani, és sok szoftvermegoldás létezik erre, például a `tunnelv`, a `vpnd`, a `vtun`, vagy akár *ppp*t is átküldhetünk `ssh` felett.

Mivel már a *ppp*ről és az `ssh` csomagról is volt szó, most ezt ismertetem. A konfiguráció jóval egyszerűbb lesz, mint a behívószerver esetében. Nem kell törődni a *ppp* authentikációval, vagy hogy melyik soros vonalat használja majd a *ppp*.

A link két végét az érthetőség kedvéért hívjuk kliensnek és szervernek. A kliens oldal fog `ssh` parancsot hívni, amit a szerver oldali `sshd` fogad.

A dolog úgy fog működni, hogy a `pppd` a kliensen induláskor `ssh` segítségével bejelentkezik a szerverre, és ott elindít egy másik `pppd` parancsot. Ezzel meg is van a *ppp* link két vége. A routoláson, csomagszűrésen (ha erre szükség van) a két oldalon a `/etc/ppp/ip-up` és a `/etc/ppp/ip-down` segítségével lehet még finomítani.

Először mindkét oldalon be kell konfigurálni a `pppd` csomagot. Ehhez egy `/etc/ppp/peers/partnerneve` opció-fájlt kell kitöltenünk, ehhez hasonlóan:

```
# A kliens konfigurációja.
#
# Amíg teszteljük, addig érdekelnek az üzenetek.
debug
# Authentikáció nem kell.
noauth
# Nem akarjuk, hogy a pppd a háttérbe forkoljon.
nodetach
# A "soros vonalon" nem modem ül, hanem egy ssh.
local
# Megadjuk a link lokális és távoli IP címét.
192.168.22.23:192.168.23.23
# Kikapcsoljuk az összes tömörítést.
bsdcomp 0
novj
novjccomp
deflate 0
nopredictor1
# Kikapcsoljuk a proxy arp-t.
noproxyarp
# Terminálnév helyett a pty opció
# segítségével egy program nevét adjuk meg.
#
# Ennek hatására a pppd lefoglal egy terminált úgy, hogy
# a terminál egyik végén a pppd, a másikon a program
# (itt az ssh) lesz. Így a pppd az sshn keresztül
# fogja a Point-to-Point protokollt megvalósítani.
pty "/usr/bin/ssh -e none -t -i /root/.ssh/szerverppp root@szerver"
# Az ssh a -t kapcsoló hatására lefoglal egy terminált
# a másik oldalon, a -e none hatására pedig letiltja
# a  ~.  karakterkombinációval történő kapcsolatbontást.
```

A fájl neve legyen mondjuk `/etc/ppp/peers/proba1`.

A szerveren a `pppd` konfigurációja ehhez nagyon hasonló. A különbség annyi, hogy az IP címeket fell kell cserélni, és nem kell a `pty...`sor. A fájl neve itt `/etc/ppp/peers/proba2` legyen.

Ezek után jöhet az `ssh` konfigurálása. Generálnunk kell egy jelszó nélküli *RSA* kulcsot a kliensen, legyen ennek a neve `/root/.ssh/szerverppp`. Ha ez megvan, a kulcs publikus felét át kell vinni a szerverre.

A szerveren a kulcs publikus felét le kell tenni a `/root/.ssh/authorized_keys2` fájlba a már korábban ismertetett módon. A fájlt ezután kézzel módosítjuk, hogy csak *ppp*t lehessen futtatni ezzel a kulccsal:

```
from="szerver_neve",command="/usr/sbin/pppd call proba2" AAAAB3NzaC1yc2EAAA...
# Ezt a kulcsot csak a szerverről engedjük,
# csak /usr/sbin/pppd call proba2-t futtathat.
```

A szerveren a *PAM*ot érdemes úgy beállítani, hogy bejelentkezéskor semmit se írjon ki, mert az megzavarja a *ppp*t. Ehhez a `/etc/pam.d/ssh` fájlból ki kell kommenteznünk a

- `session optional pam_lastlog.so`,
- `session optional pam_motd.so`, és a
- `session optional pam_mail.so`

sorokat, ha van benne ilyen. Ezeknek a jelentéséről később lesz szó.

Végül, ha minden megvan, a kliensen `pppd call proba1` paranccsal indíthatjuk a *ppp*t. Ennek hatására a kliensen a `pppd` a `/etc/ppp/peers/proba1` fájlból fogja felszedni a parancsokat, meghívja az `ssh` parancsot, és a szerver oldalon elindítja a `pppd call proba2` parancsot. A kapcsolat bontásához *HUP* szignált küldhetünk a `pppd`-nek, a `kill` parancs segítségével.

Ha azt akarjuk, hogy a *ppp* linkünk mindig biztosan menjen, a `/etc/inittab` fájlba tehetünk egy bejegyzést, hogy az `init` mindig újraindítsa, ha leáll. Ez a bejegyzés például így nézhet ki:

```
# A 2-es és 3-as futási szinten, P0 azonosítóval
# menjen a pppd, és induljon újra ha meghal. A P0
# helyett más, tetszőleges azonosítót is használhatunk.
P0:23:respawn:/usr/sbin/pppd call proba1
```

Ezután egy `killall -HUP init` parancsot kiadva már megy is a `pppd`.

1.5. Felhasználói erőforrások korlátozása

1.5.1. Ulimit

Az `ulimit` egy shell interfész a kernel erőforrás-korlátozó hívásaihoz.

Az `ulimit` segítségével különböző erőforrásokat korlátozhatunk, mint például a maximálisan felhasználható CPU időt, maximálisan felhasználható memóriát. Az `ulimit` funkcionalitását vagy shellből az `ulimit` hívással lehet használni, vagy `setrlimit()` rendszerhívással a programjainkban. Mivel az `ulimit` `shell` parancs, shellenként változhat a szintaxisa.

`Bash` `ulimit` használata: az `ulimit` paranccsal lehet beállítani a különböző paramétereket. „Kemény” és „puha” korlátokat különböztethetünk meg, a „keményet”

nem lehet nagyobbra módosítani ha egyszer beállítottuk, a „puhát” a „kemény” határ eléréséig módosíthatjuk felfelé is.

A legfontosabb határok valószínűleg a használható memória mennyiségét korlátozzák, ezek a következők:

-l segítségével megadhatjuk, hogy mennyi memóriát jelölhet nem swapelhetőnek a processz;

-m segítségével megadhatjuk a munkahalmaz maximális méretét (azaz körülbelül azt, hogy egy processznek mennyi fizikai memória jut);

-s segítségével a maximális stack méretet adhatjuk meg;

-v a maximális virtuális memória méretet szabályozhatjuk.

Ezek közül sajnos nem mindegyik működik úgy, ahogy az ember várná, jó vigyázni velük, és letesztelni őket, mielőtt élesben használnánk. Ennek ellenére érdemes foglalkozni velük, ilyen korlátozások nélkül lokális felhasználók könnyen lefoghatnak egy gépet.

Másik három fontos kapcsoló a

-n ezzel korlátozhatjuk a nyitva tarható fájlok maximális számát;

-t ezzel korlátozhatjuk, hogy egy processz mennyi gépidőt kap, ha ezt túllépi, akkor egy szignált kap, amitől kilép;

-u az egy felhasználóra jutó processzek maximális számát határozhatjuk meg.

Egy példa a CPU idő korlátozására:

```
# Indítunk egy bash shellt.
$ bash
# 10 másodperc gépidőt adunk neki.
$ ulimit -t 10
# Shell végtelen ciklus, hogy felzabálja
# a 10 másodpercét.
$ while :; do :; done
# Kifutott az időből, a rendszer lelőtte.
zsh: killed     bash
```

1.5.2. PAM

A *PAM* (*Pluggable Authentication Modules*) egy szoftverrendszer, aminek a segítségével rugalmas felhasználó-authentikációt lehet megvalósítani Linuxon (és más rendszereken). A hangsúly a rugalmasságon van, *PAM* segítségével ugyanaz az alkalmazás authentikálhat a szokásos password fájlból, `LDAP` adatbázisból, `mysql` adatbázisból, stb. anélkül, hogy az alkalmazáson bármit is változtatni kellene.

Például az `ftp`, `pop3`, `ssh` démonok általában mind *PAM*ot használnak az authentikáció elvégzéséhez.

A *PAM* a bejelentkezéssel kapcsolatos különböző funkciókat valósítja meg, ezek az authentikáció, a felhasználó kilétének megállapítása, az „account” menedzselés (ezzel egyéb korlátozásokat lehet még az authentikáció mellé definiálni), a „session” menedzselés (ez a session megkezdése előtt és után fut le, és különböző adminisztrációs funkciókat lát el, mint például naplóbejegyzések írása). A negyedik ilyen funkció a jelszómenedzselés.

A *PAM* rendszer a konfigurációját a `/etc/pam.d` könyvtárban tartja, minden szolgáltatáshoz egy megadott nevű fájl tartozik. Hogy egy szolgáltatáshoz mi tartozik, az általában fordítási időben dől el.

A fájl felépítése:

```
modultípus   fontosság-jelzõ   modulneve   modulparaméterek
```

A *modultípus* a fent említett funkciók szerint `auth`, `account`, `session`, `password` lehet. Egy modultípushoz több ilyen sor is tartozhat, ezeket a fájl elejétől a vége felé haladva fogja végigjárni a *PAM*, a később leírt szabályoknak megfelelően. Ez fontos a *PAM* működése szempontjából.

A *fontosság-jelző* azt jelzi, hogy mi történjen, ha a *PAM* az adott sornál sikeresen vagy sikertelenül hajtotta végre a feladatát. Ez a következők közül lehet egy:

required Az adott modul sikertelensége esetén a művelet (amit a modultípus határoz meg) sikertelen lesz, de a rendszer a többi modult is végig fogja próbálni.

requisite Az adott modul sikertelensége esetén a művelet sikertelen lesz, de a rendszer a többi modult nem fogja végigpróbálni, egyből visszatér a *PAM*ot használó alkalmazáshoz.

sufficient Ha az így jelölt modul sikeresen elvégzi a funkcióját, akkor a teljes művelet is sikeres lesz, és a többi `required`-ként megjelölt modult nem fogja végigjárni (ha egy másik `required` sem volt sikertelen e modul előtt). Ha a modul nem végezte el sikeresen a funkcióját, azt a *PAM* rendszer nem tekinti fatális hibának.

optional Ez egy opcionális elem a listában, sikeressége vagy sikertelensége nem befolyásolja a teljes művelet sikerességét, kivéve, ha ez az egyetlen modul ami alapján el lehet dönteni a teljes művelet sikerességét.

Van egy újabb mód is a modulok fontosságának jelzésére, de ez nincs elég jól dokumentálva, és jóval összetettebb a fent leírt módnál.

Modulparaméterek: a moduloknak különböző paramétereket lehet átadni. Vannak modulonként változó és általános paraméterek, ezek közül a leggyakoribbak:

debug A modul hibakeresést segítő üzeneteket fog küldeni a `syslog` démonon keresztül.

no_warn A modul nem fog figyelmeztető üzeneteket küldeni az alkalmazásnak.

use_first_pass A modul a már egy korábbi modulnak megadott jelszót fogja használni az authentikációhoz. Ezt az opciót csak az `auth` és a `password` funkcióknál lehet használni.

try_first_pass A modul a már egy korábbi modulnak megadott jelszót próbálja használni az authentikációhoz. Ha sikertelen a felhasználó azonosítása, akkor új jelszót fog kérni. Ezt az opciót csak az `auth` funkcióval lehet használni.

expose_account A modul a felhasználóhoz kapcsolódó különböző információkat tesz elérhetővé, például kiírja a nevét amikor a jelszót kéri. Ez barátságos, de nem túl biztonságos.

Alapértelmezett beállítások: a *PAM* az alapértelmezett beállításait a `/etc/pam.d/other` fájlban tartja. Ha egy szolgáltatásnak nem adunk külön konfigurációs fájlt, akkor ezt fogja használni. Ezt érdemes úgy beállítani, hogy senkit se engedjen be:

```
auth           required     pam_deny.so
account        required     pam_deny.so
session        required     pam_deny.so
password       required     pam_deny.so
```

Majd ki lehet egészíteni `pam_warn.so` modulokkal, mert a `pam_deny` nem küld semmilyen hibaüzenetet.

Ha kizártuk magunkat. . . , a számítógépet *single user* módban újraindítva ki tudjuk javítani a hibát.

A leggyakrabban használt modulok:

pam_access Ez a modul a `/etc/security/access.conf` konfigurációs fájl alapján, a felhasználói név és a terminál név, vagy a távoli gép (ahonnan például az `ssh` parancsot indították) alapján enged vagy nem enged be felhasználókat. Tipikus felhasználás:

```
account     required    pam_access.so
```

pam_cracklib Ez a modul a *cracklib* könyvtár segítségével megpróbálja feltörni a jelszót, hogy ellenőrizze, elég erős-e. Különböző ellenőrzéseket hajt végre a jelszón. Tipikus használata:

```
password   required       pam_cracklib.so retry=3 minlen=6 difok=3
password   required       pam_unix.so use_authtok nullok md5
```

Paraméterei: `retry=` megadja, hogy hányszor fog megpróbálni új jelszót kérni a felhasználótól. `minlen=` a jelszó minimális hosszát adja meg. `dcredit=`, `ucredit=`, `lcredit=`, `ocredit=`: ezek azt adják meg, hogy mennyit számít, ha számjegy, nagybetű, kisbetű, vagy egyéb karakter van a jelszóban. Például nagybetűk esetében plusz eggyel számít az összes nagybetű, amíg a megadott számtól kevesebb nagybetű van a jelszóban. A `difok=` paraméter azt mondja meg, hogy legalább hány karakternek kell másnak lennie az új és régi jelszóban (de ha a karaktereknek legalább a fele különböző, azt mindenképpen elfogadja). Ez a modul támogatja az erősebb *MD5*ös jelszavak használatát.

pam_deny Ez a modul minden kérést visszautasít, nem enged bejelentkezni, jelszót változtatni, stb. Használata:

```
account     required    pam_deny.so
```

pam_env Ez a modul környezeti változókat tud beállítani. Ehhez a `/etc/environment` vagy a `/etc/security/pam_env.conf` fájlt használja. A `/etc/environment` egyszerű `változónév=érték` párokból áll, ezzel szemben a `/etc/security/pam_env.conf` összetettebb változóbeállítást enged meg: egy környezeti változóhoz megadhatunk `DEFAULT` és `OVERRIDE` értékeket. A változó a `DEFAULT` értéket veszi fel, kivéve, ha az `OVERRIDE`ként megadott sztring nem üres. Ebben az esetben a változó értéke az adott sztring lesz. Tipikus használata:

```
auth     required    pam_env.so
```

Minimális példa konfigurációs fájl:

```
# /etc/security/pam_env.conf
DISPLAY               DEFAULT=${REMOTEHOST}:0.0 OVERRIDE=${DISPLAY}
```

Megjegyzések: megadhatunk más fájlokat a `/etc/environment` és a `/etc/security/pam_env.conf` fájl helyett az `envfile=fájlnév` és a `conffile=fájlnév` modulparaméterekkel.

pam_ftp Ezen modul segítségével *anonymous ftp*, vagy más hasonló szolgáltatást valósíthatunk meg. Ha *anonymous* vagy *ftp* felhasználóként próbálunk belépni, akkor a felhasználónevet *ftp*re állítja és beenged. Egyébként a megadott felhasználónevet és jelszót békén hagyja, és hibajelzéssel tér vissza (aminek hatására egy másik modul folytathatja az authentikációt mondjuk a szokásos `/etc/password` fájlból.) Tipikus használata:

```
# /etc/pam.d/ftp
auth    sufficient    pam_ftp.so
auth    required      pam_unix.so use_first_pass
```

pam_group Ez a modul a felhasználó által használt terminál, a felhasználó neve és az idő alapján extra csoportazonosítókat ad a felhasználó processzeinek. A konfigurációs fájlja a `/etc/security/groups.conf`. A fájl szerkezete:

```
szolgáltatásnév;terminálok;felhasználók;idő;csoportok
```

Ahol a *szolgáltatásnév* a *PAM*ot használó szolgáltatás neve, a *terminálok* a használt terminál neve, a *felhasználók* a felhasználók listája, a *csoportok* pedig az extra csoportazonosítók, amiket a felhasználó megkap, ha az előbb említett feltételek teljesülnek. A listák „logikai listák”, az időt kivéve &, |, ! operátorokkal lehet belőlük logikai kifejezéseket alkotni. Az időt intervallumként lehet megadni, ahol a napokat az angol nevük szerint „Mo Tu We Th Fr Sa Su” jelöli, „Al” jelenti a hét minden napját, „Wk” a hétköznapokat, „Wd” a hétvégét. Ha egy napot kétszer adunk meg, akkor az tagadást jelent. Például ha minden lokális felhasználónak floppy és hangkártya használati jogot akarunk adni akkor a `/etc/security/groups.conf` fájlba egy ilyen bejegyést tehetünk:

```
login|gdm;tty*&!ttyp*;*;Al0000-2400;floppy, audio
```

Ez azt jelenti, hogy a *gdm* és *login* szolgáltatásokra, a *tty** portról (de nem a *ttyp** portokról) jövő felhasználók mindig megkapják az extra *floppy* és *audio* csoportazonosítókat. Tipikus *PAM* bejegyzés:

```
auth     required    pam_group.so
```

pam_issue Ez a modul hozzáadja a `/etc/issue` fájlt a felhasználónevet kérdező prompthoz. Használata:

```
auth     required    pam_issue.so
```

pam_lastlog Ez a modul kezeli a `/var/log/lastlog` fájlt, a sessionök elején és végén letesz egy-egy bejegyzést ebbe a fájlba, illetve bejelentkezéskor ki tudja írni, hogy honnan történt az utolsó bejelentkezés. Használata:

```
session     required    pam_lastlog.so
```

Paraméterei:

debug debug információ írása a `syslog` démonon keresztül;

nodate, noterm, nohost ne írja ki az utolsó bejelentkezés idejét, a hozzá tartozó terminált, és/vagy a távoli hostot;

silent semmit se írjon ki;

never ha a lastlog fájl alapján úgy látja, hogy a felhasználó még egyszer sem volt bejelentkezve, akkor egy üdvözlő szöveget fog kiírni neki.

pam_limits Ezzel az `ulimit` parancsnál megismert korlátozásokat lehet felhasználónként automatikusan beállítani. Használata:

```
session     required    pam_limits.so
```

Ez a modul a konfigurációját a `/etc/security/limits.conf` fájlban tartja. A fájl szerkezete:

```
kit_korlátozunk               hard_vagy_szoft  mit_korlátozunk  korlát_értéke
felhasználónév/@csoportnév/*  soft/hard/-      korlátozandók    érték
```

Ahol a *kit_korlátozunk* egy felhasználó neve, egy @csoportnévvel megadott csoport neve, vagy * esetén mindenki lehet, a *hard_vagy_szoft* hard, soft vagy `--` lehet, ha a két korlátot egyszerre akarjuk beállítani, a *mit_korlátozunk* pedig *core*, *data*, *fsize*, *memlock*, *nofile*, *rss*, *stack*, *cpu*, *nproc*, *as*, *maxlogins*, *priority* közül lehet egy aszerint, hogy mit akarunk korlátozni. A *korlát_értéke* az erőforrás-használat felső korlátját adja meg. Tehát az adott nevű vagy adott csoportba tartozó felhasználó (vagy * esetén mindenki) soft, hard limitjét (vagy mindkettőt a `--` esetén) állíthatjuk így be. Az egyéni limitek felülbírálják a csoportlimiteket. Ha nem adjuk meg a korlátozandó erőforrás nevét és értékét, és a soft vagy hard helyén egy `--` van, akkor az adott felhasználók semmilyen korlátozást nem fognak kapni. Fontos tudni, hogy ezek a limitek bejelentkezésenkénti limitek, ha kétszer jelentkezik be a felhasználó párhuzamosan, akkor például dupla annyi processzt futtathat egyszerre.

pam_nologin A `/etc/nologin` fájl létezése esetén csak a `root` felhasználót fogja beengedni, a többieknek csak megmutatja a `/etc/nologin` tartalmát. A helyes működés érdekében ennek a modulnak az `auth sufficient ...` modulok előtt kell szerepelnie. Használata:

```
auth     required    pam_nologin.so
```

pam_securetty Ez a modul a `root` felhasználót csak a `/etc/securetty` fájlban felsorolt terminálokról engedi bejelentkezni. Használata:

```
auth     required    pam_securetty.so
```

pam_time Ennek a modulnak a segítségével a felhasználók bejelentkezését időhöz, terminálhoz és szolgáltatáshoz köthetjük. A modul konfigurációs fájlja a `/etc/security/time.conf`, melynek hasonló a szerkezete, mint a `/etc/security/groups.conf` fájlé. A különbség annyi, hogy itt nem adhatunk meg extra csoportokat, és ha a feltételek nem teljesülnek, akkor a felhasználó egyáltalán nem tud bejutni. Egy példa a konfigurációs fájlból:

```
su;tty* & !ttyp*;you|me;!Al0000-2400
```

Ezzel a *you* és *me* felhasználók a *tty** terminálokon sosem használhatják a *su* szolgáltatást. Használata:

```
account     required    pam_time.so
```

pam_unix Ez a modul a szokásos UNIX `/etc/password` és `/etc/shadow` fájlból authentikálja a felhasználókat; ezt minden rendszeren használni szokták. Bejelentkezéskor ellenőrizni tudja a különböző shadow-paramétereket, mint például utolsó jelszó váltás. Ehhez az account ellenőrzéshez tipikusan egy ilyen bejegyzés kell:

```
account     required    pam_unix.so
```

Authentikáció esetén a modul kezelni tudja a már korábban említett `try_first_pass` és `use_first_pass` paramétereket, illetve van még egy `nodelay` paramétere, aminek hatására nem fog várni az ismételt próbálkozással, ha a felhasználó jelszava vagy felhasználói neve nem volt megfelelő. Authentikációnál tipikusan így szokták használni: `auth required pam_unix.so` Jelszóváltáskor a modulnak elég sok paramétere lehet, köztük ott van a szokásos `try_first_pass` és a `use_first_pass`, a többi fontosabb paraméter: `md5` – ennek hatására a modul *md5*tel fogja titkosítani a jelszót, a gyengébb UNIX-os *crypt()* helyett. Ezt érdemes használni, de kompatibilitási problémákat okozhat például heterogén *NIS* hálózat esetén. A `use_authtok` hatására a modul minden esetben az őt megelőző modul által beállított jelszót fogja használni. A `not_set_pass` hatására a modul figyelmen kívül hagyja a korábbi modulok által beállított jelszót, és az általa beolvasott jelszót sem fogja továbbküldeni a következő moduloknak. A `remember=` paraméter hatására a modul a `/etc/security/opasswd` fájlba visszamenőleg adott számú jelszót eltárol, és megakadályozza, hogy a felhasználó ezek közül válasszon újra egyet. Végül a `nullok` paraméter hatására engedni fogja a jelszóváltoztatást akkor is, ha a régi jelszó üres volt. A jelszómenedzselő mód használata ezek után valami ilyesmi lehet:

```
password    required    pam_unix.so  md5  nullok
```

Végül a session menedzsmentről: a `pam_unix` modul egyszerűen csak logolja a session elejét és végét a `syslog` démonon keresztül. Használata:

```
session     required    pam_unix.so
```

Álljon itt egy példa egy egyszerű *PAM* beállításra az `ssh` parancshoz:

```
#%PAM-1.0

### Authentikáció.
# A /etc/nologin ellenőrzése.
auth       required     pam_nologin.so
# Authentikáció a szokásos UNIX-os password fájlokból.
auth       required     pam_unix.so
# Környezeti változók beállítása.
auth       required     pam_env.so

### Account ellenőrzés.
# Jelszó és account lejárat ellenőrzése.
account    required     pam_unix.so

### Session adminisztráció.
# A bejelenetkezés logolása.
session    required     pam_unix.so
# A következők sikeressége nem olyan fontos:
# utolsó bejelentkezés kiírása
session    optional     pam_lastlog.so
# a "nap üzenetét" kiírja,
session    optional     pam_motd.so
# kiírja, hogy jött-e a felhasználónak levele,
session    optional     pam_mail.so standard
# végül beállítja a különböző erőforrás-korlátokat.
session    required     pam_limits.so

### Jelszómendzselés.
# Cracklib segítségével ellenőrizzük az új jelszó
# erősségét,
password required       pam_cracklib.so retry=3 minlen=6 difok=3
# ha az sikeres volt, a pam_unix leteszi az új bejegyzést a
# password fájlba.
password required       pam_unix.so use_authtok nullok md5
```

1.5.3. Quota

Mi a quota?

A `quota` egy olyan szolgáltatás Linux és más UNIX rendszerek alatt, mely megakadályozza, hogy a felhasználó (vagy csoport) egy bizonyos fájl- (inode), vagy tárolóhely-korlátot (block) túllépjen. Így a felhasználók csak a saját `quota`-jukon belül tudnak adatokat tárolni, ellenőrzött körülmények között tevékenykedhetnek, nem tudják teletölteni a fájlrendszereket, nem veszik el egymástól a helyet. A `quota` lehetővé teszi, hogy a felhasználók vagy csoportok ezen beállításai fájlrendszerenként különbözőek legyenek, tehát személyenként (csoportonként) változhat az egy fájlrendszeren belüli *block*, illetve *inode* korlát.

A quota beállítása

A `quota` működéséhez először is meg kell győződnünk róla, hogy a kernelünkbe van-e *quota támogatás* fordítva, ha nincs, ezt meg kell tennünk.

Ezek után disztribúciónk csomagkezelőjének segítségével fel kell telepítenünk a `quota` csomagot, majd a `/etc/fstab` fájlban át kell írnunk a quotázni kívánt fájlrendszerek bejegyzését:

```
# block eszköz   könyvtár   fs_típus   opciók
/dev/hda7        /home      ext2       defaults,rw,usrquota,grpquota         0       2
```

Az `usrquota` és a `grpquota` közül csak az egyikre lesz szükségünk, ha csak felhasználó-quotát, vagy csak csoport-quotát akarunk kezelni. Hogy biztosra menjünk, hozzuk létre a megfelelő fájlrendszereken a `quota.user` és `quota.group` fájlokat:

```
touch /home/quota.user /home/quota.group; chmod 600 /home/quota.user /home/quota.group
```

Ezután ha minden megvan (a kernel tudja a quotát, az `fstab` rendben van, és a quota.* fájlokat is megcsináltuk), elindíthatjuk a fájlrendszerek *quota* ellenőrzését a `quotacheck -avug` paranccsal. Ez kitölti a `quota.user` és `quota.group` fájlokat. Ezt érdemes olyankor futtatni, amikor a rendszer másra nem használja az adott fájlrendszereket, különben az adatok pontatlanok lesznek. Ezek után bekapcsolhatjuk a quotát a `quotaon -avug` paranccsal.

Ezeknél a parancsoknál a kapcsolók a következőket jelentik:

-a az adott parancs nézze végig az összes quotázott fájlrendszert;

-v legyen bőbeszédű, futás közben informáljon arról, hogy mit csinál;

-u a felhasználók quotájával (is) foglalkozzon (kapcsolja ki, be, vagy pedig ellenőrizze azt);

-g a csoportok quotájával (is) foglalkozzon.

Shutdownkor a rendszer automatikusan le fogja futtatni a `quotaoff -aug` parancsot, hogy kikapcsolja a quotát. Enélkül a *quota.** fájlok lehet, hogy nem a legfrissebb adatokat fogják tartalmazni. Ezt a `quotacheck` futtatásával korrigálhatjuk.

A felhasználók quotájának beállítása

A felhasználók quotáját az `edquota` paranccsal állíthatjuk, interaktívan vagy automatikusan. Interaktív beállítás esetén `edquota -u felhasználónév ...` segítségével állíthatjuk be a felhasználók quotáit, `edquota -g csoportnév ...` segítségével pedig a csoportok quotáit. Több felhasználó és csoportnevet is megadhatunk egy parancssorban.

Megadhatjuk a *soft* és *hard* határokat, illetve a *grace time*ot. A „soft limit” átlépése esetén a rendszer figyelmezteti a felhasználókat, hogy túl sok helyet foglalnak, de engedi, hogy tovább folytassák a helyfoglalást egészen a „hard limitig”, vagy amíg „grace time”-nyi idő el nem telik a „soft limit” átlépésétől kezdve.

A határokat megadhatjuk külön a blockok számára (egy block 1024 byte), illetve a fájlok (inodeok) számára. Interaktív módban az `EDITOR` környezeti változó megfelelő beállításával választhatunk szövegszerkesztőt a munkához.

Nem interaktív módban az

```
edquota -p prototípusfelhasználó -u felhasználónév ...
```

és az

```
edquota -p prototípuscsoport -g csoportnév ...
```

parancsokkal egy prototípus felhasználó vagy csoport quotájával inicializálhatjuk a többi felhasználó vagy csoport quotáját.

Quota lekérdezések

A `quota` paranccsal le lehet kérdezni a felhasználók vagy csoportok quotáját, `-u` kapcsolóval a felhasználókét, a `-g`-vel a csoportokét. A `repquota` paranccsal fájlrendszerenként lehet lekérdezéseket csinálni; ez jóval gyorsabb, mint ha a felhasználókat egyesével kérdezgetnénk le.

Egyéb megjegyések

Jelenleg quotát a (használható minőségű) linuxos fájlrendszerek közül csak az *XFS* és az *ext2* támogat, *ReiserFS*hez elérhetők kiegészítések.

1.6. Teljesítmény javítása

1.6.1. A teljesítmény mérése

Mielőtt elkezdenénk hangolni a rendszerünket, fell kell deríteni, hogy mivel lehet (majd) probléma.

Ehhez vannak különböző benchmark programok, amivel jó esetben hasonló terhelést tudunk előállítani, mint amilyennel a rendszerünknek meg kell majd birkóznia. `Apache`hoz van `Apachebench` (`ab`, általában az `apache` csomag része), a `Postfix` csomaggal is jön benchmark program, illetve ilyeneket a hálózatról is le lehet tölteni.

Alacsonyabb szintű merevlemez benchmark rengeteg van, ilyenek a `bonnie`, `bonnie++`, `iozone`. Velük vigyázni kell, nem biztos, hogy a rendszer azon paramétereit mérjük meg, amelyekre kíváncsiak vagyunk.

Miközben benchmarkoljuk a rendszerünket, `vmstat`-tal, `ps`-sel illetve `top`-pal figyelhetjük, hogy mit is csinál, milyen célra mennyi gépidőt használ el.

A `vmstat` segítségével elég jól el lehet dönteni, hogy a rendszerünknek hol van a szűk keresztmetszete, mit érdemes bővíteni ahhoz, hogy javítsunk a teljesítményén. A hálózat terhelését akár `ifconfig` segítségével is nézhetjük, vagy inkább a `/proc/net/dev` elemzésével. Ha a hálózat 100%-ig terhelve van, a processzor és a diszk viszont nincs annyira, akkor azon kell változtatnunk, stb. Amire érdemes még odafigyelni, az a swap. Egy fájlszervernek nem szabad aktívan swapelnie.

1.7. Megfelelő hardver

1.7.1. A merevlemezek

Ha a szerverünk I/O intenzív feladatokat fog végezni, mint például egy fájlszerver (ebbe a kategóriába sorolható a statikus lapokat kiszolgáló www szerver, ftp szerver, samba és az elektronikus levelezés is), akkor a várt forgalomtól függően érdemes *SCSI* merevlemezeket használni.

Az *IDE* és *SCSI* merevlemezek sebessége közt manapság egyre csökken a különbség, de alapvetően más architektúrájuk miatt az *IDE* merevlemezek kezelése több gépidőt eszik, mint az *SCSI*ké, ami egy nagy forgalmú szerveren problémát jelenthet.

A forgalmat nagyon nehéz előre megbecsülni, nem lehet tudni, hogy például egy www szerver sikeres lesz-e, ezért ha csak a legkisebb esélyét is látjuk, hogy a forgalom megnőhet, vegyünk *SCSI*t.

Az *SCSI* további előnye, hogy egy *SCSI* vezérlő több eszközt tud meghajtani mint egy *IDE* vezérlő, ezt is érdemes figyelembe venni. Ha még több pénzünk van, vehetünk üvegszálon kommunikáló, *Fibre Channel* merevlemezeket, de ezek nagyon drágák.

Ha lehet több merevlemezt vegyünk, és szervezzük úgy az adatainkat, hogy a terhelés megosztódjon a lemezek közt. Legalább három részre szétbonthatjuk az adatainkat: a „/” és egyéb olyan fájlrendszerekre, amikhez nem nagyon nyúl hozzá a rendszer, a fájlrendszerre, ahová a naplóbejegyzések kerülnek, és végül a fájlrendszerre, ahonnan

az adatokat szolgáltatjuk. Ezek közül az utóbbi két fájlrendszer lesz I/O intenzív, érdemes őket külön merevlemezekre tenni.

1.7.2. Hálózati kártya

Hálózati kártyáknál nagyon fontos a megbízhatóság. Mindenképpen olyan kártyát vegyünk, aminek jó neve van a linuxozók körében. Nem érdemes NE2000 klónnal nagy terhelésű szervert üzemeltetni.

Érdemes olyan kártyát venni, ami tud hardveres ellenőrzőösszeg-számítást csinálni, és ezt a linux kernel ki is használja, mert nagy forgalom esetén ez komoly terhet vesz le a processzorról. Ezekkel megvalósítható olyan halózatkezelés, hogy a hálózatra kikerülő adatok egyszer sem mennek át a processzor buszán. Ilyen például a 3Com 905C, és a gigabit ethernet kártyák.

A hálózati kártyák ezen képességeinek kihasználásához megfelelő, a *sendfile* rendszerhívást támogató szoftver kell. Az `ftp` szerver szoftverek közül ilyen például a `vsftpd` és a `proftpd`.

1.7.3. Processzor

Terheléstől függően erős processzorra lehet szükség. Sima statikus fájlszerver esetén, a megfelelő hardveres körítéssel egy gyengébb processzor is nyugodtan bírni fogja a terhelést, de például dinamikus lapokat kiszolgáló www szerver esetén a határ a csillagos ég. Érdemes lehet több processzorban gondolkozni.

Oda kell figyelni a processzor cache méretére, és a processzorbusz sebességére. Ha ezek túl kicsik, a rendszer nem lesz jó teljesítményű. Negatív példaként érdemes megemlíteni a régi, másodszintű cache nélküli Celeronokat.

1.7.4. Memória

Memóriából érdemes annyit venni, hogy bőven elférjenek benne a futtatott alkalmazások, fájlszerver esetén pedig rengeteg fájlt tudjon a memóriában tartani, a gyors elérés érdekében.

A memória sebességét érdemes a processzoréhoz igazítani. Egy 1 Gigabyte/másodperces processzor-busszal rendelkező (Intel Pentium III) processzorhoz teljesen felesleges méregdrága, 3.2 Gigabyte/másodperces *Rambus RAM*ot venni.

1.8. Szoftveres teljesítményjavítás

1.8.1. Megfelelő fájlrendszer-típus kiválasztása

Elég sok fajta linuxos fájlrendszer van, ezek közül legalább három használható minőségű: a régi `ext2`, a `Reiserfs`, és az *SGI-féle* `XFS`.

Az `ext2` nagyon jó teljesítményű nagy fájlok esetén, illetve amíg a könyvtárakban kevés fájl van. Nagyon kevés gépidőt fogyaszt. Jó blokklefoglaló algoritmusa miatt nem nagyon töredeznek a fájlok. Hátránya, hogy ha sok fájl kerül egy könyvtárba nagyon lelassul, illetve nem naplózott, azaz áramszünet vagy rendszerösszeomlás esetén `fsck`-zni kell. Adatbázisokhoz valószínűleg ez a legjobb.

A `Reiserfs` egy újabb fájlrendszer, nagyon jól kezeli a „sok fájl egy könyvtárban” esetet, naplózott, rendszerösszeomlás után nem kell `fsck`-zni (ennek ellenére

van hozzá `fsck`, arra az esetre, ha szoftverhiba miatt korruptálódna a fájlrendszer). Hátránya, hogy több gépidőt eszik az `ext2` fájlrendszernél. Előnyös lehet levelező szervereknél.

1.8.2. Megfelelő szerverszoftver használata

Érdemes olyan szoftvert keresni, ami minél kevesebb processz felhasználásával oldja meg a feladatot, mert a processzek közti kapcsolgatás költséges.

Statikus lapok kiszolgálására jó például a `boa`, vagy a `thttpd`, ami egy szálon fut, vagy a `Tux` nevű kernelben futó www szerver. Ez utóbbi már tud *anonymous ftp*t is kiszolgálni.

1.8.3. Kernel paraméterezés

A kernelnek rengeteg menet közben állítható paramétere van, ezekkel nagyban javítható a rendszer teljesítménye.

Ezeket a kernelparamétereket vagy a `sysctl` paranccsal, vagy a `/proc` alatt lévő fájlok írásával lehet állítani. Mi ez utóbbival fogunk foglalkozni.

Általában ezekből a fájlokból `cat` segítségével kiolvashatók az érvényben lévő értékek, és egy átirányított `echo` paranccsal egyszerűen megváltoztathatók.

Megnyitható fájlok maximális száma: ha a rendszerünk nagyon sok kérést szolgál ki egyszerre, szükség lehet ennek a megemelésére. Ehhez az

```
echo 16384 > /proc/sys/fs/file-max
```

parancsot adjuk ki, ha például 16384-re szeretnénk állítani a nyitva tartott fájlok maximális számát. Hogy szükségünk van-e ennek a megemelésére, azt a `/proc/sys/fs/file-nr` fájlból tudhatjuk meg. Ebben az első szám a lefoglalt fájl-azonosítók számát jeleneti; ha ez kezd közel kerülni a harmadik számhoz (ami ugyanaz, mint ami a `file-max` fájlban van), akkor ideje megemelni a megnyitható fájlok számát.

Mountolható fájlrendszerek száma: ha nagyon sok fájlrendszert akarunk mountolni, az

```
echo 512 > /proc/sys/fs/super-max
```

paranccsal fel tudjuk emelni például 512-re a mountolható fájlrendszerek számát.

Maximális processz-szám: ha nagyon sok programot akarunk egyszerre futtatni, akkor az

```
echo 16384 > /proc/sys/kernel/threads-max
```

paranccsal 16384-re emelhetjük ezt a korlátot.

Hálózati kapcsolatok száma: ha sok kapcsolatot szeretnénk nyitni, akkor érdemes megnövelni a `/proc/sys/net/ipv4/ip_local_port_range` „fájlban” található hálózati-port intervallum felső határát.

Vannak egyéb, a memóriamenedzselést befolyásoló paraméterek, de ezek nincsenek elég jól dokumentálva.

1.8.4. *IDE* merevlemez finomhangolása

A `hdparm` program segítségével sok beállítást elvégezhetünk *IDE* merevlemezeken. Bekapcsolhatjuk a *DMA* módokat, stb.

Ugyanakkor elrontott beállítás adatvesztést okozhat, óvatosnak kell lenni vele.

A `hdparm -Tt /dev/hda` parancs segítségével lemérhetjük a merevlemez átviteli sebességét, ezt használhatjuk változtatásaink hatásának méréséhez. Az első szám a rendszer fájl-cachének a sebessége, ez nem fog változni. A második a merevlemez maximális átviteli sebessége, ezen tudunk változtatni. A kimenete várhatóan így fog kinézni:

```
$ hdparm -Tt /dev/hdc

/dev/hdc:
 Timing buffer-cache reads:   128 MB in  0.77 seconds =166.23 MB/sec
 Timing buffered disk reads:  64 MB in  3.46 seconds = 18.50 MB/sec
```

A `hdparm` segítségével a merevlemez beállításait, és egyéb merevlemez információt is lekérdezhetünk. Ha a `hdparm` kapcsolóihoz nem adunk paramétert, akkor az általában a beállítások lekérdezését jelenti. A `-i` kapcsolóval a merevlemez bootoláskor kiolvasott beállításait nézhetjük meg, ennek segítségével dönthetjük el, hogy egyáltalán mit tud a merevlemezünk.

```
$ hdparm -i /dev/hda

/dev/hda:

 Model=QUANTUM FIREBALLlct10 10, FwRev=A03.0900, SerialNo=872012259666
 Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs }
 RawCHS=16383/16/63, TrkSize=32256, SectSize=21298, ECCbytes=4
 BuffType=3(DualPortCache), BuffSize=418kB, MaxMultSect=16, MultSect=16
 DblWordIO=no, OldPIO=2, DMA=yes, OldDMA=2
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=20044080
 tDMA={min:120,rec:120}, DMA modes: mword0 mword1 mword2
 IORDY=on/off, tPIO={min:120,w/IORDY:120}, PIO modes: mode3 mode4
 UDMA modes: mode0 mode1 mode2 mode3 *mode4
```

Az így kapott információt felhasználhatjuk a későbbi beállításoknál.

A `-m` kapcsolóval beállíthatjuk, hogy hány blokkot olvasson be a merevlemez egyszerre. A maximális beállítható értéket kiolvashatjuk a `hdparm -i` parancs kimenetének *MaxMultSect* mezőjéből. A lineáris átviteli sebességet növelhetjük ennek a kapcsolónak a használatával.

A `-c` kapcsoló segítségével lehet az I/O módot 32 bitesre állítani. Ez megduplázhatja az átviteli sebességet. A `-c` után „1”-et írva bekapcsolhatjuk ezt az üzemmódot, „3”-mat írva ehhez még egy szinkronizáló szekvenciát kérhetünk, ha az „1”-gyel nem működne jól a merevlemezünk.

A `-u` kapcsolóval engedélyezhetjük, hogy a kernel más megszakításokat is kiszolgáljon az *IDE* megszakítás közben. Ez javíthat a hálózat és a soros port teljesítményén, és jobb interaktivitást biztosít.

A `-d` kapcsolóval bekapcsolhatjuk a merevlemez *DMA* módját. Ezek együtt:

```
$ hdparm -u1 -m8 -c1 /dev/hdc

/dev/hdc:
 setting 32-bit I/O support flag to 1
 setting multcount to 8
 setting unmaskirq to 1 (on)
 multcount    =  8 (on)
 I/O support  =  1 (32-bit)
 unmaskirq    =  1 (on)
```

Fontos tudni, hogy a kapcsolók némelyike hibás hardverrel adatkorrupciót okozhat. Ne éles rendszeren kísérletezgessünk a `hdparm` használatával.

1.9. *Logical Volume Manager* és naplózott fájlrendszer

1.9.1. A *Logical Volume Manager*

A *Logical Volume Manager* még két absztrakciós szintet tesz a szokásos merevlemez partíciók fölé.

Az első, a *Logical Volume Group* szint a partíciók (vagy egyéb blokkos eszközök) felett egy egyesített réteget képez, azaz több blokkos eszközt összevonhatunk egy *Volume Group*ba (ezentúl csak VG-be).

A *VG*ket menet közben lehet újabb blokkos eszközökkel bővíteni, így átméretezni, stb.

A *VG*ket tovább lehet bontani *Logical Volume*okra (innentől csak *LV*ként fogunk hivatkozni rájuk), ezekre lehet majd fájlrendszert tenni, vagy máshogy blokkos eszközként használni. Az *LV*ket is át lehet méretezni menet közben, és ha megfelelő fájlrendszert teszünk rá, ami szintén támogatja a menet közbeni átméretezést, akkor egy rugalmas, menet közben átméretezhető háttértár-rendszert kap az ember.

A *Logical Volume Manager* (*LVM*) használatához a kernelünket *Logical Volume Manager* támogatással kell fordítani, illetve szükségünk lesz az *LVM* segédprogramjaira. Ezeket például a http://www.sistina.com/lvm/ oldalon meg lehet találni, de az *LVM* használatához szükséges programok már megtalálhatók a legtöbb disztribúcióban. A használt segédprogramok verzióját a kernelben használt meghajtó verziójához kell igazítani.

A *Logical Volume Manager* használatához először elő kell készíteni a partícióinkat erre: `fdisk` segítségével (vagy valami más partícionáló programmal) a partíció típusát hexadecimális `8e`-re kell állítani. Ezek után a

```
pvcreate /dev/blokkeszköznév
```

segítségével úgynevezett *Physical Volume*ot (*PV*) lehet belőle készíteni, majd a *PV*kből *VG*t:

```
vgcreate proba /dev/blokkeszköznév1 ...
```

Ennek hatására létrejön egy *proba* nevű *VG*, aminek a mérete megegyezik az alatta lévő partíció méretével. Ezek után egy *LV*t kell létrehoznunk rajta, és már használhatjuk is:

```
lvcreate -L200M -n elsolv proba
mkfs.ext2 /dev/proba/elsolv
mount ...
```

Ennek hatására a rendszer létrehoz egy 200 MB-os *elsolv* nevű *LV*t, a *proba* nevű *VG*re.

Ezek után jöhetnek az érdekesebb dolgok: az `lvextend` segítségével átméretezhetjük az *LV*nket. Arra vigyázzunk, hogy ha nagyobbra méretezzük, akkor mindig először az *LV*t állítsuk nagyobbra, és csak aztán a fájlrendszert rajta, ha pedig kisebbre vesszük, akkor először a fájlrendszert, aztán az *LV*t módosítsuk. Az *LV* módosításához az `lvextend` és az `lvreduce` programokat használhatjuk. Például egy *LV* nagyobbra vételéhez (250 MB-ra) az

```
lvextend -L250M /dev/proba/elsolv
```

parancsot használhatjuk.

Egy *VG* alá a `vgextend` paranccsal rakhatunk be újabb *PV*ket, illetve a `vgreduce` paranccsal vehetünk ki alóla *PV*ket. A `vgreduce` parancsot csak akkor használhatjuk, ha az adott *PV*t nem használja a rendszer. A `pvmove` parancs segítségével egy *PV*ről máshová költöztethetjük az adatokat, így akár egy merevlemezről egy másikra átköltözhetünk anélkül, hogy a rendszert e miatt le kellene állítani. Persze merevlemez cseréjéhez, vagy más konfigurációval kapcsolatos munkák elvégzéséhez ettől még lehet, hogy le kell állni. Ha például a `/dev/hda3` partíción lévő *proba* nevű *VG*nket át akarjuk költöztetni a `/dev/hdc3` partícióra, akkor azt a következőképpen tehetjük meg:

```
# Kibővítjük a VG-t a hdc3-mal.
vgextend proba /dev/hdc3
# A hda3-ról átvisszük az adatokat a hdc3-ra.
# (Ez sokáig tarthat.)
pvmove /dev/hda3 /dev/hdc3
# A proba VG alól kivehetjük a hda3-mat.
vgreduce proba /dev/hda3
```

Bootolás, shutdown

Bootoláskor a

```
vgscan
vgchange -a y
```

parancsokkal lehet detektálni és aktiválni, shutdownkor pedig a

```
vgchange -a n
```

paranccsal lehet deaktiválni a *VG*ket.

1.10. Naplózott fájlrendszerek

A naplózott fájlrendszerek olyan fájlrendszerek, amelyeknél tranzakciókezelési technikák segítségével mindig konzisztens adat van a diszken.

Ez nem azt jelenti, hogy áramszünet esetén nem vesztünk adatot, hanem azt, hogy nem kell sokáig tartó *fsck*ra várni.

A naplózott fájlrendszerek nem védenek a merevlemezek meghibásodásától sem.

Linux alatt egyre több naplózó fájlrendszer vált elérhetővé az utóbbi években, mindegyik más minőségű, más tudású. Említés szintjén ezek a `Reiserfs`, az *SGI-féle* `XFS`, az *IBM-féle* `JFS`, az `ext2` leszármazottja az `ext3`, illetve a `Tux2` nevű, szintén `ext2` fájlrendszerből származó „failsafe” fájlrendszer.

Ezek közül a leghasználhatóbb a `Reiserfs`, és az `XFS`. Itt az elsővel fogok foglalkozni.

A `Reiserfs` egy kiegyensúlyozott fákon alapuló fájlrendszer, emiatt főleg kis fájlok kezelésénél gyorsabb társainál. Ezen tulajdonságát kihasználva nagy teljesítményű fájl, és levelezőszerver építhető vele. További előnyös tulajdonsága, hogy támogatja a lemountolás nélküli átméretezést, ami jól párosítható az *LVM LV* átméretezésével.

A `Reiserfs` hátránya, hogy több gépidőt eszik, mint például az `ext2`, illetve 32 MB-ot lefoglal magának a naplózáshoz. Ezért ha egy 32 MB-os fájlrendszerre van szükségünk, akkor 64 MB-os partíció kell hozzá.

A `Reiserfs` a 2.4-es kernelsorozat részét képezi, esetleg hibajavításokért érdemes elnézni a http://www.namesys.org/ oldalra. Onnan letölthetjük a `Reiserfs`-hez tartozó programokat is, de valószínűleg kedvenc disztribúciónkban is megtaláljuk ezeket.

Ha sikerült a kernelünket `Reiserfs` támogatással lefordítani és a segédprogramok is megvannak, akkor az `mkreiserfs` vagy `mkfs.reiserfs` paranccsal hozhatunk létre ilyen fájlrendszert egy blokk-eszközre, amit aztán a szokásos `mount` paranccsal kezdhetünk el használni.

A `reiserfs` extra `mount` opciói a `notail` és a `resize=`. Az első segítségével kikapcsolhatjuk a kis fájlok és a fájlvégek különleges kezelését. Ez gyorsíthatja a fájlműveleteket, illetve régi `lilo`-nál szükség van erre, különben nem fogunk tudni bootolni a fájlrendszerünkről. A helyes bootoláshoz ezt az opciót a kernel helyretétele és a `lilo` futtatása előtt kell bekapcsolni.

A `resize` opció segítségével menet közben átméretezhetünk egy fájlrendszert, nem kell lemountolni se. Arra kell figyelni, hogy a fájlrendszer új mérete ne legyen nagyobb az alatta lévő blokkos eszköztől. A `resize` használatával csak növelni lehet a fájlrendszer méretét. Ha csökkenteni akarjuk, akkor a `resize_reiserfs` programot kell használnunk, miután lemountoltuk a fájlrendszert.

A `resize` opció 4 kB-os blokkméretben várja az új méretet.

Ezek alapján egyszerűen megírhatunk egy olyan szkriptet, aminek a segítségével egyszerre méretezhetjük át a fájlrendszert, és az *LV*t, amin a fájlrendszer ül:

```
#!/bin/sh
# használat  resize   blokkeszköz-útvonala újméret (MB-ban)
#
#         $0     $1         $2
#
# Hiba esetén egyből kilépünk.
set -e
# Ha pontosan két paraméterünk van, akkor megyünk tovább.
if [ $# = 2 ] ; then
        # Nagyobbra húzzuk át a LV-t.
        /sbin/lvextend -L${2}M "${1}"
        # Átszámítjuk a MB-ot négy kB-os blokkokra.
        FOURKBLOCKS=‘/usr/bin/expr ${2} ’*’ 256 ‘
        # Megcsináljuk az átméretezést.
        /bin/mount -n "${1}" -oremount,resize=${FOURKBLOCKS}
        # Ha eddig eljutottunk, akkor minden jól ment.
        /bin/echo ’++++ Átméretezés sikeres ++++’
        # Még kiírjuk, hogy mennyi helyünk is van az átméretezés után.
        /bin/df -k "${1}"
else
        /bin/echo "Használat: ${0} blokkeszköz-útvonala új méret (MB-ban)"
fi
```

A `mount` opciókkal kapcsolatban még annyit érdemes megjegyezni, hogy ha csak olvashatóként mountoljuk a fájlrendszert, a `Reiserfs` akkor is írni fog rá, amikor a naplóját ellenőrzi.

1.11. Naplóelemzés

1.11.1. A *syslog* megfelelő beállítása

A `syslog`, a rendszer naplóíró démonjának helyes beállítása nagyon fontos a megfelelő naplózáshoz.

`Syslog` démon többféle van, ilyenek a régi `syslog`, az újabb, többet tudó `syslog-ng`, illetve vannak adatbázisba naplózó démonok is.

A régi unixos `syslog` démonnal fogunk foglalkozni, az található meg a legtöbb rendszeren.

A UNIX-os `syslog` démonnál a naplóüzeneteket kétféle szempont szerint szokták csoportosítani: az egyik a `facility`, ez körülbelül meghatározza, hogy ki küldte az üzenetet, a másik a `priority`, ami az üzenetek fontosságát határozza meg.

A `facility` lehet

auth, authpriv biztonsággal kapcsolatos bejegyzések esetében;

cron a `cron` és `at` démon bejegyzéseinek;

daemon az egyéb démonok bejegyzéseinek;

kern a kernel bejegyzéseinek;

local0, local1, . . . local7 tetszőleges lokális bejegyzéseknek;

lpr a nyomtató-alrendszer bejegyzéseinek;

mail a levelező szerver bejegyzéseinek;

news a news szerver bejegyzéseinek;

user általános, felhasználói bejegyzéseknek.

A `priority` lehetséges értékei, jelentésük:

emerg valami komoly problémát jelez, például valamelyik alrendszer használhatatlanná válását;

alert olyan hibát jelez, amit azonnal korrigálni kell;

crit valamelyik alrendszer állapota kritikus;

err valamelyik alrendszer hibáját jelzi;

warning mindössze figyelmeztetést jelent;

notice ezt a prioritást normális, de fontos események esetén szokták használni;

info információközlésre szokták használni, nem jelez problémát;

debug a programok a hibák megtalálását segítő üzeneteknél szokták használni.

A naplóbejegyezések szétválogatásánál az üzenet ezen két tulajdonságát lehet használni.

A `syslog` démon támogatja az üzenetek fájlba, *FIFO*ba, terminálra, távoli gépre vagy lokálisan bejelentkezett felhasználóknak történő továbbítását.

A `syslog` démon konfigurációs fájlja a `/etc/syslog.conf`, ez szelektor-üzenet cél párosokat tartalmaz, ahol a szelektor az üzenet típusát határozza meg, a cél pedig azt, hogy hova kerüljön az üzenet.

A szelektorok alapvetően `facilitymeghatározás.prioritásmeghatározás` felépítésűek, ahol a *facilitymeghatározás* (egy vagy) több `facility` vesszővel elválasztott listája, ez *vagy* kapcsolatot jelez köztük, illetve a `*` lehet, ez illeszkedik minden `facility`-re.

A *prioritásmeghatározás* egy egyedülálló prioritást tartalmazhat, ebben az esetben az összes, az adottal egyező, és attól nagyobb prioritású üzenetre illeszkedni fog. A formája lehet továbbá `=prioritás`, ez értelemszerűen pontos egyezést jelent. A `!` segítségével lehet tagadást kifejezni. Ha prioritásként `none`-t adunk meg, az semmire sem fog illeszkedni.

Az üzenet céljának meghatározása ettől jóval egyszerűbb. Egy `/` jellel kezdődő cél merevlemezen lévő fájlt jelent, `-/` szintén fájlt jelent, de ebben az esetben az üzenetet nem szinkron módon írja ki a diszkre, hanem hagyja, hogy a kernel azt majd később elintézze (ez gyorsabb mint az előző módszer, de összeomlás esetén adatot veszthetünk). A `|` jellel kezdődő cél merevlemezen lévő *FIFO*t jelent, ezzel akár saját kezűleg is feldolgozhatjuk az üzeneteket. A `@` jel után egy távoli gép nevét várja a `syslog`, oda fogja tovább küldeni a naplóbejegyzéseket (hogy ez működjön, a távoli rendszeren `-r` kapcsolóval kell indítani a `syslog` démont). A `*` jellel, illetve az egyéb karakterekkel kezdődő célkijelölés hatására a rendszer minden felhasználónak, vagy az adott felhasználók termináljára fogja küldeni az üzenetet.

Példaként egy részlet egy `syslog` konfigurációs fájlból:

```
*.=debug;\
        auth,authpriv.none;\
        news.none;mail.none     -/var/log/debug
```

Ennek hatására tehát a `syslog` démon az *auth*, *authpriv*, *news*, *mail facility*ken kívül minden *debug* prioritású üzenetet aszinkron módon a */var/log/debug* fájlba ír.

További meggondolások: a biztonságos naplózás érdekében érdemes egy `*.debug` bejegyzést is használni, hogy biztosan ne hagyjunk ki egy *facility-prioritás* párost se. Ugyancsak érdemes egy távoli gépre is küldeni a logokat, hogy például egy betörés, vagy komoly rendszerösszeomlás esetén megtudhassuk mi történt.

1.11.2. A naplófájlok elemzése

A `syslog` rendszer egyik problémája, hogy az üzenetek, amiket a programok küldenek, bizonyos szempontból nem standardizáltak, például az `ssh`, `ftp`, `telnet`, stb. démonok máshogy naplózzák, ha egy bejelentkezés sikertelen volt. Ezért lesz szükség külön naplóelemzésre.

Bizonyos forgalomig „kézzel” is végignézhetjük a naplóbejegyzéseinket, de ez például napi 10 MB-nyi naplónál is lehetetlen. Ilyenkor valamilyen programot kell használnunk.

A kész programok közül valószínűleg a `logcheck` nevű a legelterjedtebb, de írhatunk magunknak is egyet.

Ha magunknak akarunk írni logellenőrző programot, akkor azt egyszerű shell szkriptekkel megtehetjük. Nagyon jól lehet együtt használni a `cut`, a `sed`, a `sort` és a `uniq` segédprogramokat. Kiindulásként használhatunk valami ilyesmi rendszert: a `cut` segítségével levághatjuk a dátum mezőt a naplóbejegyzésekről (az az első 16 karakter), aztán a `sed` paranccsal további feldolgozást csinálhatunk (felesleges sorok törlése, a naplóba író processzek processzazonosítójának törlése, stb.), végül – a `sort` és a `uniq` segíségével – ha több megegyező sorunk van, abból egyet csinálhatunk.

Tehát a shell szkriptben valami ilyesmi fog szerepelni:

```
cut -b16- /var/log/syslog |sed -f sedparancsfile |sort |uniq
```

A `sed` parancsfájlja pedig ilyesmi lesz (ez csak egy példa!):

```
/^ debi pppd\[.*$/d
/^ debi automount\[[0-9]*\]:.*$/d
s/CRON\[[0-9]*\]:/CRON:/
s/watchdog\[[0-9]*\]:/watchdog:/
...
```

Azaz a *debi* nevű gépről jövő *pppd* illetve *automount* üzeneteire nem vagyunk kíváncsiak, töröljük őket, a *cron* és *watchdog* démon üzeneteiről pedig leszedjük a processzazonosító számot, hogy majd a `sort` és a `uniq` több sort össze tudjon egyre húzni. Ezt a programkát hasonló elveket követve kibővíthetjük. Ezek után a feldolgozott naplóbejegyzéseket elküldhetjük magunknak levélben, kinyomtathatjuk, stb.

Ez természetesen csak egy példa volt, vannak hiányosságai, de kezdetnek nem rossz. Innen továbblépve jóval bonyolultabb naplóelemző programokat is megírhatunk magunknak.

Ha nem akarunk magunknak naplóellenőrző programot írni, akkor használhatjuk például a `logcheck` nevű programot. Ezt telepíthetjük csomagból, ha a disztribúciónk részét képezi, vagy letölthetjük a http://www.psionic.com/tools/ címről. Ez a program mintaillesztés segítségével próbálja meg megmondani egy-egy naplóbejegyzésről, hogy betörési kísérlet-e, rendszerbiztonság sérülését jelző bejegyzés-e, vagy csak valamilyen szokatlan sor a naplófájlban.

A program egyszerűen konfigurálható, a `logcheck.sh` fájlban kell átírni a megfelelő útvonalakat, amik a különböző mintákat tartalmazó fájlokra mutatnak. A `HACKING_FILE=`, `VIOLATIONS_FILE=`, `VIOLATIONS_IGNORE_FILE=` és az `IGNORE_FILE=` sorokat kell átírni aszerint, hogy hová tettük a `logcheck.hacking`, `logcheck.violations`, stb. fájlokat.

Ezek után a

```
$LOGTAIL /var/log/messages >> $TMPDIR/check.$$
```

sorokat kell átírni, hogy megmondjuk neki, hogy melyik fájlokban ellenőrizze a naplóbejegyzéseket. Ha van olyan fájlunk ahová a `*.debug` bejegyzéseket küldjük, érdemes azt megadni.

Most már akár le is futtathatnánk a `logcheck.sh` fájlt, de előbb nézzük át a mintákat tartalmazó fájlokat.

A `logcheck.hacking` fájl tartalmazza a betörési kísérletekre utaló mintákat. Ha ezek közül egy is illeszkedik a naplófájl egy sorára, akkor azt a program egy ideiglenes fájlban eltárolja. A `logcheck.violations` és `logcheck.violations.ignore` fájlt együtt használja a program; ha a naplófájl egy sora illeszkedik egy mintára a `logcheck.violations` fájlból, de egyre sem a `logcheck.violations.ignore` fájlból, akkor ezt a sort is beteszi egy átmeneti fájlba. Végül az összes olyan sort, ami nem illeszkedik a `logcheck.ignore` mintáira, szintén eltárolja. Ha talált támadásra utaló jeleket, az elmentett adatokat egy *gépnév dátum ACTIVE SYSTEM ATTACK!* című levélben elküldi a rendszergazdának. Ha nem talált támadásra utaló bejegyzéseket, akkor a levél címe egyszerűen *gépnév dátum system check* lesz.

Ha úgy látjuk, hogy minden rendben van, akkor a `cron` démonra bízhatjuk a `logcheck.sh` rendszeres lefuttatását.

1.12. Alapvető biztonsági beállítások

1.12.1. Szolgáltatások

Ne futtassunk felesleges szolgáltatásokat. A futó processzek listáját például a `ps axu` paranccsal nézhetjük meg. Nézzük végig a listát, és próbáljuk minden processzről megtudni, hogy mit csinál, a `man` parancs és a disztribúcióval jött egyéb dokumentációk alapján. Ha úgy döntöttünk, hogy az adott processzre nincs szükségünk, akkor érdemes azt leszedni disztribúciónk csomagkezelőjével. Ilyen tipikusan felesleges szolgáltatások közé tartozhat például a `portmap` és az `lpd`.

Ezek után nézzük végig a nyitott hálózati portokat. Ezt a `netstat -a` parancs segítségével tehetjük meg. Az outputból minket most csak a `LISTEN`, azaz hálózati kérésre váró portok érdekelnek. Ezt a listát például a

```
$ netstat -a |fgrep LISTEN
```

paranccsal kérhetjük a rendszertől. Példa kimenet:

```
tcp        0      0 *:shell                 *:*                     LISTEN
tcp        0      0 *:ldap                  *:*                     LISTEN
tcp        0      0 *:time                  *:*                     LISTEN
tcp        0      0 *:discard               *:*                     LISTEN
tcp        0      0 *:font-service          *:*                     LISTEN
...
```

A példán látszik, hogy a *shell*, *ldap*, *time*, *discard*, *font-service TCP* portok vannak nyitva. Azt, hogy melyik port milyen processzhez tartozik, az `fuser` paranccsal tudhatjuk meg. Példa:

```
# A font-service tcp port használójára vagyunk
# kíváncsiak, bőbeszédű kimenetet kérünk.
$ fuser -v -n tcp font-service

                     USER        PID ACCESS COMMAND
font-service/tcp     root        323 f....  xfs
```

Tehát a `font-service` porton az `xfs` processz várja a beérkező kéréseket. Most, hogy megvan a processzünk, a fentebb leírtak alapján dönthetünk arról, hogy szükségünk van-e rá.

Nyitott *UDP* portok: ezek a `netstat -a` kimenetében így látszanak:

```
udp        0      0 *:talk                  *:*
```

Ezekre is működik az `fuser` parancs, a `tcp` kulcsszót értelemszerűen `udp`re cserélve.

1.12.2. *inetd*

Az `inetd` az *Internet Superserver Daemon*, igény szerint többfajta hálózati szolgáltatást tud nyújtani segédprogramok felhasználásával. Konfigurációs fájlja a `/etc/inetd.conf`. Róla már volt szó korábban.

Ha nincs szükségünk egy `inetd`-ből futó szolgáltatásra, akkor a sor elejére # jelet írva letilthatjuk azt. Ezután az `inetd` programnak *HUP* szignált kell küldeni, hogy újraolvassa a konfigurációs fájlját. Példa letiltott `telnet` szolgáltatásra:

```
#telnet  stream  tcp  nowait  telnetd.telnetd  /usr/sbin/tcpd  /usr/sbin/in.telnetd
```

1.12.3. /etc/hosts.allow, /etc/hosts.deny

A `/etc/hosts.allow` és a `/etc/hosts.deny` fájlokkal tovább lehet finomítani a hálózati szolgáltatások listáját. A `host.allow/deny` fájlokat a `tcpwrapper` libraryval fordított programokkal vagy a `tcpd`-t használó `inetd`-s programokkal használhatjuk. A `hosts.deny` fájlba kerül bele a tiltandó szolgáltatások listája, a `hosts.allow` fájlba pedig az engedélyezetteké.

A fájlok formája durván:

```
szolgáltatás1,szolgáltatás2 : host1, host2, .domain.pelda, ipcím/netmask
```

Kiértékelési sorrend `hosts.allow`, `hosts.deny`. Ha egy szolgáltatás-kliens páros egyikben sem szerepel, akkor a hozzáférést engedélyezni fogja. Ajánlott a `host.deny` fájlba a `ALL: ALL` sort írni, és a `hosts.allow` fájlban engedélyezni a megfelelő hostokat. Példa `hosts.allow` fájlra:

```
ALL: 192.168.10.0/255.255.255.0
in.telnetd: .ilyen.nincs.hu
```

Ennek hatására a *192.168.10.0/255.255.255.0* hálózatról minden *tcpwrapper*es szolgáltatás, és az *.ilyen.nincs.hu* domainból a *telnet* szolgáltatás lesz elérhető.

1.12.4. *Mount* opciók

Megfelelően megválasztott `mount` opciókkal nagyban javíthatjuk rendszerünk biztonságát. Amit `mount` opciókkal befolyásolhatunk: írható-e az adott fájlrendszer, futtathatunk-e róla programokat, lehet-e használni eszközfájlokat rajta, illetve használhatunk-e setuid bitet.

Ennek megfelelően az érdekes `mount` opciók:

ro read only, csak olvasható lesz az adott fájlrendszer;

nodev eszközfájlokat nem lehet használni az adott fájlrendszeren;

nosuid `setuid`-os fájlokat nem a tulajdonos felhasználói azonosítójával fogja futtatni, hanem a hívó processzével, mintha a `setuid bit` nem lenne rajta beállítva;

noexec az adott fájlrendszeren levő fájlokat nem lehet futtatni (még ha különben futtathatók is lennének). Ez a kapcsoló jelen pillanatban megkerülhető az alábbi módon:

```
ldd /mnt/noexec_drive/bin/bash
```

A fenti parancs elindítja a szándékunk szerint nem futtatható állapotú partíció `/bin/` könyvtárából a `bash` binárist. Emiatt bánjunk óvatosan a `noexec` kapcsolóval!

A `mount` opciókat a `remount mount` opcióval változtathatjuk meg. Példák:

```
mount /dev/hda3 /tmp -onosuid,nodev
mount /dev/hda2 /usr -oro
```

Remountra:

```
mount /tmp -oremount,nosuid,nodev,noexec
```

Természetesen itt is érdemes csak a minimálisan szükséges jogokat meghagyni (`/usr read-only`, `/tmp` és `/home nosuid`, `nodev` és esetleg `noexec`).

1.12.5. Egyebek: *lilo*, jelszó biztonság

A kernelnek `lilo`-n keresztül meg lehet adni, hogy az `init` helyett milyen programot indítson elsőként. Például:

```
LILO: linux init=/bin/bash
```

Ez azonnali `root` shellt eredményez a számítógéphez közvetlenül hozzáférő felhasználóknak. Ez ellen segít a `restricted` `lilo.conf` opció, ennek hatására a `lilo` jelszót fog kérni, ha a kernelnek valamilyen paramétert akarunk átadni a `lilo` parancssorban. Példa:

```
# /etc/lilo.conf
# ...
image=/boot/vmlinuz
restricted=Ezittajelszo!
        label=Linux
        read-only
```

Természetesen kell még egy `chmod 600 /etc/lilo.conf` is, illetve a `lilo` program futtatása.

Általános jelszóbiztonság: sose használjunk szótári szót jelszóként, inkább generáljunk egyet a `pwgen` programmal. A jelszavakat cseréljük rendszeresen. Érdemes lehet néha a *John the Ripper* programot futtatni a jelszófájlra (letölthető a http://www.openwall.com/ oldalról).

1.13. Mentés és helyreállítás

1.13.1. *tar*

A *GNU* `tar` segítségével teljes könyvtár-hierarchiákat lehet egy vagy több eszközre összecsomagolni, így például egy fájlba lerakni, vagy szalagra írni.

A `tar` a parancsokat a parancssorában várja. Egy, és csak egy parancsot meg kell neki adni.

A parancsokon kívül vannak még nem kötelező jellegű paraméterei.

A `tar` leggyakrabban használt parancsai: a `--create` egy új archívum létrehozásához, a `--extract` egy létező archívum kibontásához, azaz a fájlok visszállításához. Egy archívum tartalmát a `--list` paranccsal tudjuk kilistázni. Ha új archívumot hozunk létre, akkor a `tar` parancsnak meg kell mondanunk, hogy miket tegyen bele. Ehhez az archiválandó fájlokat fel kell sorolni a `tar` parancsai és kapcsolói után. Archívum kibontása esetén hasonlóan adhatjuk meg az archívumból kibontandó fájlok listáját.

Gyakori opcionális kapcsolók a `--file=`, az archív fájl nevét lehet így megadni; a `tar` ezt fogja használni a parancsok elvégzésénél, például ebbe fogja tenni az archiválandó fájlokat. Ha nem használjuk a `--file=` kapcsolót, akkor a `tar` a standard outputot fogja használni. Fontos kapcsoló még a `--verbose`, ennek hatására a `tar` bőbeszédű lesz.

Ezek után már ki is lehet próbálni a `tar` parancsot:

```
# Csinálunk egy könyvtárstruktúrát a próbához.
$ mkdir proba
$ echo hello >proba/a
$ echo hello2 >proba/b
# A proba nevű könyvtárat elmentjük az
# elso.tar nevű fájlba.
```

```
$ tar --create --file=elso.tar --verbose proba
proba/
proba/a
proba/b
# Kilistázzuk az archívum tartalmát.
$ tar --list --file=elso.tar --verbose
drwxrwxr-x root/root       0 2001-07-19 13:55:07 proba/
-rw-rw-r-- root/root       6 2001-07-19 13:55:02 proba/a
-rw-rw-r-- root/root       7 2001-07-19 13:55:07 proba/b
# Elvesztjük a proba könyvtárunkat.
$ rm -rf proba
# Aztán a mentésből visszaállítjuk.
$ tar --extract --verbose --file=elso.tar
proba/
proba/a
proba/b
# Aztán még egyszer kibontjuk, de most csak a
# proba/a nevű fájlra vagyunk kíváncsiak.
$ tar --extract --verbose --file=elso.tar proba/a
proba/a
```

Ezek után az egyszerű példák után jöhetnek az érdekesebb dolgok: a `tar --update` parancsa segítségével egy archív fájl tartalmát frissíthetjük. Ez egyszerűen úgy működik, hogy a `tar` végignézi az archív fájlt, és az archiválandó könyvtárstruktúrát, és ha valamelyik fájl újabb, mint az archívumban található, akkor az új fájlt hozzáfűzi az archívumhoz. Emiatt az archív fájl mérete állandóan nőni fog. Az archív fájl kibontása esetén csak az utoljára hozzáfűzött fájl fog a lemezen maradni. Szalagos egységeken ezt nem lehet használni. Például:

```
$ echo megegy >proba/k
$ tar --update --file=elso.tar --verbose proba
proba/
proba/k
```

A `tar --delete` parancsa segítségével törölhetünk egy fájlt az archívumból. Ez nem működik szalagos egységek esetén.

```
# Kitöröljük a proba/a fájlt az archívumból.
$ tar --delete --file=elso.tar proba/a
```

A `tar` segítségével inkrementális mentések is készíthetők, ehhez a `--listed-incremental=` opciót kell használni. Ez az opció egy fájl nevét várja, a `tar` ezt a fájlt fogja használni a könyvtár-hierarchiában történt változások követésére, tehát más-más könyvtárak mentéséhez más fájlt kell megadnunk. Például:

```
# Csinálunk egy új mentést.
$ tar --create --file=masodik.tar --verbose --listed-incremental=/tmp/lista proba
proba/
proba/a
proba/b
proba/k
# Ezután létrehozunk egy új fájlt a proba könyvtár alatt,
# és arról csinálunk egy inkrementális mentést.
# Az inkrementális mentés új tar fájlba kerül.
$ touch proba/ujfile
$ tar --create --file=harmadik.tar --verbose --listed-incremental=/tmp/lista proba
proba/
proba/ujfile
# Ezek után, ha vissza akarjuk állítani a proba könyvtár
# tartalmát, akkor azt két tar segítségével megtehetjük.
$ tar --extract --file=masodik.tar
$ tar --extract --file=harmadik.tar
```

A `tar` parancsnak van még egy fontos opciója, a `--multi-volume`, ennek segítségével például több szalagra lehet menteni az adatokat. A kazettaváltások közt a `tar` jelez.

A `tar` különböző eszközökre tud menteni, ezek tipikusan fájl és mágnesszalag (a `tar`-t eredetileg szalagos eszközökhöz tervezték), de tud távoli eszközökre is mentést végezni `rsh` vagy `ssh` használatával. Ha távoli eszközt szeretnénk használni, akkor a `--file=` opciónál ezt a `gépnév:útvonal` vagy a `felhasználó@gépnév:útvonal` paraméterekkel érhetjük el.

```
$ tar --create --verbose --file=debi:/tmp/proba.tar --rsh-command=/usr/bin/ssh .
./
./index.html
...
```

A távoli gépen kell lennie egy `/etc/rmt` programnak, ez biztosítja a távoli eszközök elérését.

A visszaállítás távolról hasonlóképpen történik.

A `tar` tud különböző tömörítéseket kezelni, de érdemesebb egyszerűen pipe-ot használva ráküldeni egy tömörítőprogramra, mert azt így könnyebb paraméterezni. Például ha erősebb tömörítést szeretnénk alkalmazni, hogy kisebb állományt kapjunk, akkor azt így oldhatjuk meg:

```
$ tar --create --verbose . | gzip -9 >/tmp/proba.tar.gz
./
./index.html
...
```

1.13.2. *rsync*

Az `rsync` egy főleg mirrorozásra használt program, így például két gép (vagy csak egy gépen belül) között tud könyvtár-hierarchiákat mozgatni.

Az `rsync` előnye például az `ftp` parancsot használó mirrorokkal szemben az, hogy eleve mirrorozásra tervezték. A két könyvtár-hierarchia közt csak a különbségeket viszi át, ezekkel együtt alkalmas például egy szerver és egy tartalék szerver közti adatszinkronizálásra.

Az `rsync` parancsot általában

```
rsync opciók forrásmeghatározás célmeghatározás
```

módon szokták használni, ahol az *opciók* határozzák meg a fájlátvitel különböző paramétereit, a *forrás* és a *cél* pedig értelemszerűen a forrás- és célkönyvtárat határozzák meg.

Az `rsync` eléggé sok opciót ismer, ezek közül a legfontosabbakkal foglalkozunk itt.

A `-v` és a `-q` opciók a bőbeszédű és a csendes üzemmód közt váltanak. Több `-v` hatására az `rsync` egyre több üzenetet fog kiírni, ezzel segíti a hibakeresést.

A `-a` opció az archív módot kapcsolja be. Ez azt jelenti, hogy az `rsync` rekurzívan másolja a fájlokat és a könyvtárakat, és megtartja a tulajdonosokat, a fájlok jogosultságait, linkeket, és az eszközfájlokat, feltéve, hogy van ehhez joga. Ezek után már ki is lehet próbálni az `rsync`-et:

```
# Létrehozunk egy könyvtárat a próbához.
$ mkdir /tmp/proba
# Elindítjuk az rsyncet, a public_html könyvtárat
# visszük át a /tmp/proba könyvtárba.
$ rsync -av public_html/ /tmp/proba
building file list ... done
./
index.html
...
```

```
wrote 4782961 bytes  read 576 bytes  9567074.00 bytes/sec
total size is 4780153  speedup is 1.00
# Végül megnézzük az eredményt.
$ ls /tmp/proba
index.html ...
```

Ezek után, ha megint `rsync`-et futtatnánk, akkor az már csak a különbségeket vinné át.

Oda kell figyelni a forrás meghatározásánál a forrás végén álló „/” jelre; ez befolyásolja, hogy az `rsync` hogyan építi fel a könyvtárstruktúrát a célon. Ezt egy példán keresztül lehet a legjobban megérteni:

```
# Létrehozunk egy könyvtárat a próbához.
$ mkdir /tmp/proba
# Elindítjuk az rsyncet, a public_html könyvtárat
# visszük át a /tmp/proba könyvtárba.
$ rsync -av public_html /tmp/proba
building file list ... done
public_html/
public_html/index.html
...
wrote 4782998 bytes  read 576 bytes  9567148.00 bytes/sec
total size is 4780153  speedup is 1.00
# Megnézzük mit tett le.
$ ls /tmp/proba
public_html
```

Egy ilyen mentésből a helyreállítás értelemszerűen a forrás és a cél megfelelő felcserélésével elérhető.

Mint már korábban volt róla szó, az `rsync` használható távoli szinkronizálásra vagy mentésre is.

Ez kétféleképpen valósítható meg: vagy egy `inetd`-ből futtatott `rsync` szerverrel, vagy pedig `rsh` vagy `ssh` segítségével. Biztonsági okok miatt az utóbbi lehetőséggel fogunk foglalkozni. Ezzel ötvözni lehet az `rsync` és az `ssh` képességeit.

Ezt a módot úgy lehet használni, ha az `rsync` parancsnak célként vagy forrásként egy távoli gépen található könyvtárat adunk meg, `távolifelhasználó@távoligép:távolikönyvtár` formában, és a `-e` kapcsoló segítségével megmondjuk neki, hogy `ssh` parancsot használjon az alapértelmezett `rsh` helyett.

```
# Elindítjuk az rsync-et ssh-n keresztül.
$ rsync -e "ssh" -av public_html tonhal:/tmp/proba
# Az ssh kéri a jelszót.
gabor@tonhal's password:
# Ezután a szokásos rsync kimenetet láthatjuk.
building file list ... done
created directory /tmp/proba
public_html/
public_html/index.html
...
wrote 4782998 bytes  read 576 bytes  289913.58 bytes/sec
total size is 4780153  speedup is 1.00
```

Ezt persze kombinálni lehet az `ssh` különböző kapcsolóival, például ha más porton (mondjuk az 1678-ason) fut a távoli `ssh` szerver, akkor a parancssor ilyen lesz:

```
$ rsync -e "ssh -p 1678" -av public_html tonhal:/tmp/proba
```

Egyéb fontos `rsync` opciók: a `-z` kapcsoló hatására az `rsync` röptömöríteni fogja az átjövő adatokat. Ez csökkenti a hálózat terhelését, de növeli a két gép processzorterhelését. A `--delete` kapcsoló hatására az `rsync` törölni fogja a cél oldalán azokat a fájlokat, amelyek a forrás oldalon (már) nincsenek meg. A

`--numeric-ids` hatására az `rsync` numerikusan fogja átvinni a felhasználó és csoportazonosítókat. Ez előnyös lehet, például ha a *home* könyvtárakat mentjük másik gépre, és a távoli gépen a felhasználóink nincsenek felvéve.

GNU Szabad Dokumentációs Licensz 1.1 verzió, 2000 március

0. ELŐSZÓ

Jelen Licensz célja egy olyan kézikönyv, tankönyv, vagy effajta írott dokumentum megalkotása, mely a szó szoros értelmében „szabad”: annak érdekében, hogy mindenkinek biztosítsa a szöveg sokszorosításának és terjesztésének teljes szabadságát, módosításokkal, vagy anélkül, akár kereskedelmi, akár nem-kereskedelmi úton. Másfelől, e Licensz megőrzi a szerző, vagy kiadó munkája elismeréséhez fűződő jogát, s egyúttal mentesíti őt a mások által beiktatott módosítások következményei alól.

Jelen Licensz egyfajta „etalonnak” tekinthető, ami nem jelent mást, mint hogy a dokumentumból származtatott munkák maguk is szabad minősítést kell, hogy kapjanak. E dokumentum egyben a GNU Általános Felhasználói Licensz kiegészítőjeként is szolgál, mely egy a szabad szoftverekre vonatkozó etalon licensz.

E Licenszet a szabad szoftverek kézikönyveiben való használatra alkottuk, hiszen a szabad szoftver egyben szabad dokumentációt is igényel: egy szabad programot olyan kézikönyvvel kell ellátni, mely ugyanazon szabadságokat biztosítja, mint maga a program. Jelen Licensz, mindazonáltal, nem korlátozódik pusztán kézikönyvekre; feltételei tetszőleges tárgykörű írott dokumentumra alkalmazhatók, függetlenül attól, hogy az könyvformában valaha megjelent-e. Mindamellett e Licenszet főként olyan munkákhoz ajánljuk, melyek elsődleges célja az útmutatás, vagy a tájékoztatás.

1. ALKALMAZHATÓSÁG ÉS DEFINÍCIÓK

E Licensz minden olyan kézikönyvre, vagy más jellegű munkára vonatkozik, melyen megtalálható a szerzői jogtulajdonos által feltüntetett figyelmeztetés, miszerint a dokumentum terjesztése jelen Licensz feltételei alapján lehetséges. A „Dokumentum” alább bármely ilyen jellegű kézikönyvre, vagy egyéb munkára vonatkozik. A lakosság minden tagja potenciális licensztulajdonosnak tekinthető, és mindegyikük megszólítása egyaránt „ön”.

A Dokumentum „Módosított Változata” bármely olyan munkára vonatkozik, mely tartalmazza a Dokumentumot, vagy annak elemeit akár szó szerint, akár módosításokkal, és/vagy más nyelvre lefordítva.

A „Másodlagos Szakasz” egy egyedi névvel bíró függelék, esetleg a Dokumentum egy megelőző szakasza, mely kizárólag a kiadóknak, vagy az alkotóknak a Dokumentum átfogó tárgyköréhez (vagy kapcsolódó témákhoz) fűződő viszonyáról szól, és nem tartalmaz semmi olyat, ami közvetlenül ezen átfogó témakör alá eshet. (Ha például a Dokumentum részben egy matematika tankönyv, úgy a Másodlagos Szakaszban nincs lehetőség matematikai tárgyú magyarázatokra.) A fenti kapcsolat tárgya lehet a témakörrel, vagy a kapcsolódó témákkal való történelmi viszony, illetve az azokra vonatkozó jogi, kereskedelmi, filozófiai, etikai, vagy politikai felfogás.

A „Nem Változtatható Szakaszok” olyan speciális Másodlagos Szakasznak számítanak, melyek ilyetén való meghatározását az a közlemény tartalmazza, miszerint a Dokumentum jelen Licensz hatálya alatt lett kiadva.

A „Borítószövegek” olyan rövid szövegrészek, melyek Címlap-szövegként, illetve Hátlap-szövegként kerülnek felsorolásra abban a közleményben, miszerint a Dokumentum jelen Licensz hatálya alatt lett kiadva.

A Dokumentum „Átlátszó” példánya olyan géppel-olvasható változatot jelöl, mely a nyilvánosság számára hozzáférhető formátumban kerül terjesztésre, továbbá melynek tartalma szokványos szövegszerkesztő-programokkal, illetve (pixelekből álló képek esetén) szokványos képmegjelenítő-programokkal, vagy (rajzok esetén) általánosan hozzáférhető rajprogramok segítségével azonnal és közvetlenül megtekinthető, vagy módosítható; továbbá olyan formátumban mely alkalmas a szövegszerkesztőkbe való bevitelre, vagy a szövegszerkesztők által kezelt formátumokba való automatikus átalakításra. Egy olyan, egyébként Átlátszó formátumban készült példány, melynek markupja úgy lett kialakítva, hogy megakadályozza, vagy eltántorítsa az olvasókat minden további módosítástól, nem tekinthető Átlátszónak. A nem „Átlátszó” példányok az „Átlátszatlan” megnevezést kapják.

Az Átlátszóság kritériumainak megfelelő formátumok között megtalálható például a markup nélküli egyszerű ASCII, a Texinfo beviteli formátum, a LaTeX beviteli formátum, az SGML vagy az XML egy általánosan hozzáférhető DTD használatával, és a standardnak megfelelő, emberi módosításra tervezett egyszerű HTML. Az Átlátszatlan formátumok közé sorolható a PostScript, a PDF, a szabadalmaztatott és csak fizetős szövegszerkesztőkkel olvasható formátumok, az olyan SGML vagy XML, melyhez a szükséges DTD és/vagy egyéb feldolgozó eszközök nem általánosan hozzáférhetők, és az olyan gépileg-generált HTML formátum, melyet egyes szövegszerkesztők hoznak létre, kizárólag kiviteli célra.

Egy nyomtatott könyv esetében a „Címlap” magát a címlapot, illetve bármely azt kiegészítő további oldalt jelöl, amely a jelen Licenszben definiált címlap-tartalmak közzétételéhez szükséges. Az olyan formátumú munkáknál, melyek nem rendelkeznek effajta címlappal, a „Címlap” a munka címéhez legközelebb eső, ám a szöveg törzsét megelőző szövegrészeket jelöli.

2. SZÓ SZERINTI SOKSZOROSÍTÁS

Önnek lehetősége van a dokumentum kereskedelmi, vagy nem-kereskedelmi jellegű sokszorosítására és terjesztésére, bármely médiumon keresztül, feltéve, hogy jelen Licensz, a szerzői jogi figyelmeztetés, továbbá a Dokumentumot jelen Licensz hatálya

alá rendelő közlemény minden példányban egyaránt megjelenik, és hogy e feltételeken kívül semmi mást nem tesz hozzá a szöveghez. Nem alkothat olyan technikai korlátokat, melyek megakadályozhatják, vagy szabályozhatják az ön által terjesztett példányok elolvasását, vagy sokszorosítását. Mindazonáltal elfogadhat bizonyos összeget a másolatok fejében. Amennyiben az ön által terjesztett példányok száma meghalad egy bizonyos mennyiséget, úgy a 3. szakasz feltételeinek is eleget kell tennie.

A fenti kritériumok alapján kölcsönbe adhat egyes példányokat, de akár nyilvánosan is közzéteheti a szöveget.

3. SOKSZOROSÍTÁS NAGYOBB MENNYISÉGBEN

Amennyiben 100-nál több nyomtatott változatot tesz közzé a Dokumentumból, és annak Licensze feltételül szabja a Borítószövegek meglétét, úgy minden egyes példányt köteles ellátni olyan borítólapokkal, melyeken a következő Borítószövegek tisztán és olvashatóan fel vannak tüntetve: Címlap-szövegek a címlapon, illetve Hátlap-szövegek a hátlapon. Mindkét borítólapra egyértelműen és olvashatóan rá kell vezetnie a kiadó, vagyis jelen esetben az ön nevét. A címlapon a Dokumentum teljes címének jól láthatóan, továbbá minden egyes szónak azonos szedésben kell megjelennie. Ezen felül, belátása szerint, további részleteket is hozzáadhat a borítólapokhoz. Amennyiben az esetleges módosítások kizárólag a borítólapokat érintik, és feltéve, hogy a Dokumentum címe változatlan marad, továbbá a borítólapok megfelelnek minden egyéb követelménynek, úgy a sokszorosítás ettől eltekintve szó szerinti reprodukciónak minősül.

Abban az esetben, ha a borítólapok bármelyikén megkövetelt szövegrészek túl hosszúnak bizonyulnának az olvasható közzétételhez, úgy csak az elsőként felsoroltakat kell feltüntetnie (amennyi józan belátás szerint elfér) a tényleges borítón, a továbbiak pedig átkerülhetnek a következő oldalakra.

Amennyiben 100-nál több Átlátszatlan példányt tesz közzé, vagy terjeszt a Dokumentumból, úgy köteles vagy egy géppel-olvasható Átlátszó példányt mellékelni minden egyes Átlátszatlan példányhoz, vagy leírni minden egyes Átlátszatlan példányban egy a módosítatlan Átlátszó példányt tartalmazó nyilvános hozzáférésű számítógéphálózat elérhetőségét, ahonnan bárki, anonim módon, térítésmentesen letöltheti azt, egy közismert hálózati protokoll használatával. Ha az utóbbi lehetőséget választja, köteles gondoskodni arról, hogy attól a naptól kezdve, amikor az utolsó Átlátszatlan példány is terjesztésre került (akár közvetlenül ön által, akár kiskereskedelmi forgalomban), a fenti helyen közzétett Átlátszó példány még legalább egy évig hozzáférhető legyen a felhasználók számára.

Megkérjük, ámde nem kötelezzük önt arra, hogy minden esetben, amikor nagyobb példányszámú terjesztésbe kezd, már jóval ezt megelőzően lépjen kapcsolatba a Dokumentum szerzőivel, annak érdekében, hogy megkaphassa tőlük a Dokumentum esetleges felújított változatát.

4. MÓDOSÍTÁS

Önnek lehetősége van a Dokumentum Módosított Változatának sokszorosítására és terjesztésére a 2. és 3. szakaszok fenti rendelkezései alapján, feltéve, hogy a Módosított

Változatot kizárólag jelen Licensz feltételeivel összhangban teszi közzé, ahol a Módosított Változat a Dokumentum szerepét tölti be, ezáltal lehetőséget biztosítva annak terjesztésére és módosítására bárkinek, aki csak hozzájut egy példányához. Mindezen felül, a Módosított Változat az alábbi követelményeknek is meg kell, hogy feleljen:

- A Címlapon (és ha van, a borítókon) tüntessen fel egy a Dokumentumétól, illetve bármely korábbi változatétól eltérő címet (melyeknek, ha vannak, a Dokumentum Előzmények szakaszában kell szerepelniük). Egy korábbi változat címét csak akkor használhatja, ha annak szerzője engedélyezte azt.
- A Címlapon szerzőkként sorolja fel a Módosított Változatban elvégzett változtatásokért felelős személyeket, vagy entitásokat, továbbá a Dokumentum fő szerzői közül legkevesebb ötöt (vagy mindet, ha nincsenek öten).
- A Címlapon a Módosított Változat közzétételéért felelős személyt tüntesse fel kiadóként.
- A Dokumentum összes szerzői jogi figyelmeztetését hagyja érintetlenül.
- Saját módosításaira vonatkozóan is tegyen közzé egy szerzői jogi megjegyzést, a többi ilyen jellegű figyelmeztetés mellett.
- Rögtön a szerzői jogi figyelmeztetéseket követően tüntessen fel egy közleményt, az alábbi Függelék mintájára, melyben engedélyezi a Módosított Változat felhasználását jelen Licensz feltételei alapján.
- A fenti közleményben hagyja érintetlenül a Nem Változtatható Szakaszok és a szükséges Borítószövegek jelen Dokumentum licenszében előírt teljes listáját.
- Mellékelje jelen Licensz egy eredeti példányát.
- Az „Előzmények” szakaszt, illetve annak címét szintén hagyja érintetlenül, emellett adjon hozzá egy új elemet, amely minimálisan tartalmazza a Módosított Változat címét, kiadási évét, továbbá az új szerzők, illetve a kiadó nevét, a Címlapon láthatókhoz hasonlóan. Amennyiben a Dokumentum nem tartalmaz semmiféle „Előzmények” elnevezésű szakaszt, úgy hozzon létre egyet, mely tartalmazza a Dokumentum címét, kiadási évét, továbbá a szerzők, illetve a kiadó nevét, a Címlapon láthatókhoz hasonlóan; majd ezt követően adjon hozzá egy új, a Módosított Változatra vonatkozó elemet, a fentiekkel összhangban.
- Ne tegyen változtatásokat a Dokumentumban megadott Átlátszó példány nyilvános hálózati elérhetőségét (ha van ilyen) illetően, vagy hasonlóképp, a Dokumentum alapjául szolgáló korábbi változatok hálózati helyére vonatkozóan. Ezek az „Előzmények” szakaszban is szerepelhetnek. Csak abban az esetben hagyhatja el egyes korábbi változatok hálózati elérhetőségét, ha azok legkevesebb négy évvel a Dokumentum előtt készültek, vagy ha maga az alkotó engedélyezi azt.
- Bármely „Köszönetnyilvánítás”, vagy „Ajánlások” szakasz címét hagyja érintetlenül, továbbá gondoskodjon arról, hogy azok tartalma és hangvétele az egyes hozzájárulókat, és/vagy az ajánlásokat illetően változatlan maradjon.
- A Dokumentum összes Nem Változtatható Szakaszát hagyja érintetlenül, úgy címüket, mint tartalmukat illetően. A szakaszok számozása, vagy bármely azzal egyenértékű jelölés nem tartozik a szakaszcímek közé.

- Töröljön minden „Jóváhagyás” elnevezésű szakaszt. Effajta szakaszok nem képezhetik részét a Módosított Változatnak.

- Ne nevezzen át semmilyen létező szakaszt „Jóváhagyás”-ra, vagy olyasmire, mely címében a Nem Változtatható Szakaszokkal ütközhet.

Ha a Módosított Változat új megelőző szakaszokat tartalmaz, vagy olyan függelékeket, melyek Másodlagos Szakasznak minősülnek, ám nem tartalmaznak a Dokumentumból származó anyagot, abban az esetben, belátása szerint, e szakaszok némelyikét, vagy akár az összeset nem változtathatóként sorolhatja be. Ehhez nem kell mást tennie, mint felsorolni a szóban forgó címeket a Módosított Változat licenszének Nem Változtatható Szakaszok listájában. E címeknek határozottan el kell különülnie minden egyéb szakaszcímtől.

„Jóváhagyás” elnevezésű szakaszt csak akkor adhat a Dokumentumhoz, ha az kizárólag a Módosított Változatra utaló megjegyzéseket tartalmaz – például mások recenzióira vonatkozóan, vagy hogy egy szervezet a szöveget egy standard mérvadó definíciójaként ismerte el.

Címlap-szöveg gyanánt egy legfeljebb öt szóból álló szövegrészt adhat meg, a Hátlap-szöveg esetén pedig 25 szót fűzhet a Módosított Változat Borítószövegeinek végéhez. Bármely entitás csak és kizárólag egy Címlap- és egy Hátlap-szövegrészt adhat (akár közvetítőn keresztül) a Dokumentumhoz. Ha a dokumentum már eleve rendelkezik Borítószöveggel, akár azért, mert azt korábban ön adta hozzá, vagy mert valaki más önön keresztül gondoskodott erről, abban az esetben nincs lehetőség újabb Borítószöveg hozzáadására; a régit mindazonáltal lecserélheti, abban az esetben, ha annak kiadója egyértelműen engedélyezi azt.

A Dokumentum szerzője/i és kiadója/i jelen Licensz alapján nem teszik lehetővé nevük nyilvános felhasználását egyetlen Módosított Változat támogatása, vagy támogatottsága érdekében sem.

5. KOMBINÁLT DOKUMENTUMOK

Önnek lehetősége van a Dokumentum egyéb, e Licensz hatálya alatt kiadott dokumentumokkal való kombinálására a 4. szakasz módosított változatokra vonatkozó rendelkezései alapján, feltéve, hogy a kombináció módosítás nélkül tartalmazza az eredeti dokumentumok összes Nem Változtatható Szakaszát, és hogy azok mind Nem Változtatható Szakaszként kerülnek felsorolásra a kombinált munka licenszében.

A kombinált munkának jelen Licensz mindössze egy példányát kell tartalmaznia, az egymással átfedésben lévő Nem Változtatható Szakaszok pedig kiválthatók egy összegzett példánnyal. Amennyiben több Nem Változtatható Szakasz szerepelne ugyanazon címmel, ám eltérő tartalommal, úgy alakítsa át minden egyes szakasz címét olyan módon, hogy mögéírja zárójelben az eredeti szerző és kiadó nevét (ha ismeri), vagy egy egyedi sorszámot. Ha szükséges, a Nem Változtatható Szakaszok címeivel is végezze el a fenti módosításokat a kombinált munka licenszében.

A kombinált munkában az eredeti dokumentumok összes „Előzmények” elnevezésű szakaszát össze kell olvasztania, miáltal egy összefüggő „Előzmények” szakasz jön létre; hasonlóképp kell eljárnia a „Köszönetnyilvánítás”, illetve az „Ajánlások” szakaszok tekintetében. Ugyanakkor minden „Jóváhagyás” elnevezésű szakaszt törölnie kell.

6. DOKUMENTUMGYŰJTEMÉNYEK

Önnek lehetősége van a Dokumentumból, illetve bármely egyéb, e Licensz hatálya alatt kiadott dokumentumból gyűjteményt létrehozni, és az egyes dokumentumokban található licenszeket egyetlen példánnyal kiváltani, feltéve, hogy a gyűjteményben szereplő összes dokumentum esetén minden más tekintetben követi jelen Licensz feltételeit, azok szó szerinti sokszorosítására vonatkozóan.

Tetszése szerint ki is emelhet egy meghatározott dokumentumot a gyűjteményből, továbbá terjesztheti azt jelen Licensz feltételei alapján, feltéve, hogy a szóban forgó dokumentumhoz mellékeli e Licensz egy példányát, és minden egyéb tekintetben betartja jelen Licensz előírásait a dokumentum szó szerinti sokszorosítására vonatkozóan.

7. ÖSSZEFŰZÉS FÜGGETLEN MUNKÁKKAL

A Dokumentum és annak származékainak különálló, vagy független dokumentumokkal, illetve munkákkal való összefűzése egy közös tárolási, vagy terjesztési egységen, egészében nem tekinthető a Dokumentum Módosított Változatának, feltéve, hogy az összefűzés nem lesz szerzői jogvédett. Az effajta összefűzés eredményeként „összegzés” jön létre, ám jelen Licensz nem érvényes az abban a Dokumentummal együtt szereplő önálló munkákra, hacsak azok nem a Dokumentum származékai.

Amennyiben a 3. szakasz Borítószövegekre vonatkozó rendelkezései alkalmazhatók a Dokumentum e példányaira, és a Dokumentum a teljes összegzésnek kevesebb, mint egynegyedét teszi ki, úgy a Dokumentum Borítószövegeit olyan módon is el lehet helyezni, hogy azok csak magát a Dokumentumot fogják át. Minden más esetben a teljes összegzés borítólapjain kell feltüntetni a fenti szövegeket.

8. FORDÍTÁS

A fordítás egyfajta módosításnak tekinthető, így hát a Dokumentum lefordított példányai a 4. szakasz rendelkezései alapján terjeszthetők. A Nem Változtatható Szakaszok lefordítása külön engedélyt igényel a szerzői jogtulajdonostól, mindazonáltal közzéteheti a lefordított változatokat is abban az esetben, ha az eredeti Nem Változtatható Szakaszokat is belefoglalja a munkába. E Licensz lefordítására ugyanezek a feltételek érvényesek, vagyis a lefordított változat csak akkor jelenhet meg, ha mellette ott van az eredeti, angol nyelvű Licensz szövege is. Amennyiben eltérés mutatkozna az eredeti változat, illetve a fordítás között, úgy a Licensz angol nyelvű eredetije tekintendő mérvadónak.

9. MEGSZŰNÉS

A jelen Licenszben egyértelműen kijelölt kereteken kívül tilos a Dokumentum bárminemű sokszorosítása, módosítása, allicenszelése, vagy terjesztése. Minden ezzel szembeni sokszorosítási, módosítási, allicenszelési, vagy terjesztési kísérlet a jelen Licenszben meghatározott jogok automatikus megszűnését vonja maga után. Azok a fe-

lek, ugyanakkor, akik önön keresztül jutottak másolathoz, vagy jogosultságokhoz, nem veszítik el azokat, amíg maradéktalanul betartják e Licensz előírásait.

10. JELEN LICENSZ JÖVŐBENI JAVÍTÁSAI

Megtörténhet, hogy a Szabad Szoftver Alapítvány időről időre felülvizsgált és/vagy új verziókat bocsát ki a GNU Szabad Dokumentációs Licenszből. E verziók szellemisége hasonló lesz jelen változatéhoz, ám részleteikben eltérhetnek, új problémák, új aggályok felmerülése okán. Vö.: http://www.gnu.org/copyleft/

A Licensz minden változata egyedi verziószámmal van ellátva. Ha a Dokumentum jelen Licensz egy konkrét, számozott verziójára, „vagy bármely újabb verzióra” hivatkozik, úgy önnek a szóban forgó változat, vagy bármely újabb a Szabad Szoftver Alapítvány által (nem vázlatként) publikált verzió feltételeinek követésére lehetősége van. Ha a Dokumentum nem ad meg semmilyen verziószámot, úgy bármely a Szabad Szoftver Alapítvány által valaha (nem vázlatként) publikált változat megfelel.

FÜGGELÉK: A Licensz alkalmazása saját dokumentumaira

Ha e Licenszet egy ön által írt dokumentumban kívánja használni, akkor mellékelje hozzá a Licensz egy példányát, továbbá vezesse rá az alábbi szerzői jogi és licensz közleményeket, rögtön a címlapot követően:

> Copyright © ÉV AZ ÖN NEVE.
>
> E közlemény felhatalmazást ad önnek jelen dokumentum sokszorosítására, terjesztésére és/vagy módosítására a Szabad Szoftver Alapítvány által kiadott GNU Szabad Dokumentációs Licensz 1.1-es, vagy bármely azt követő verziójának feltételei alapján. A Nem Változtatható Szakaszok neve SOROLJA FEL A CÍMÜKET , a Címlap-szövegek neve LISTA, a Hátlap-szövegek neve pedig LISTA. E licensz egy példányát a „GNU Szabad Dokumentációs Licensz” elnevezésű szakasz alatt találja.

Ha a szövegben nincsenek Nem Változtatható Szakaszok, úgy írjon „nincs Nem Változtatható Szakasz”-t, ahelyett, hogy egyenként felsorolná azokat. Ha nincsenek Címlap-szövegek, akkor írjon „nincs Címlap-szöveg”-et, ahelyett, hogy „a Címlap-szövegek neve LISTA”, és hasonlóképp járjon el a Hátlap-szövegek esetében is.

Amennyiben a dokumentum haladó programkód-példákat is tartalmaz, úgy azt javasoljuk, hogy e példákat egy választása szerinti szabad szoftver licensz alatt közölje – mint például a GNU Általános Felhasználói Licensz –, hogy lehetővé tegye a kódok szabad szoftverekben való alkalmazását.

Hátlapszöveg