Biztonság kívül-belül

Általában igaz, hogy sokat foglalkozunk a biztonsággal. Egyszerű az indok: nagyon fontos ez a témakör. E hónapban rátesszünk még egy lapáttal, és körbeszaglásszuk a jelszavakat, az azonosítást, a titkosítást (sőt, még a `tar` egy olyan változatára is kitérünk, mely titkosítja a tárolandó fájlokat), valamint a tűzfalak témakörét (egy cikk erejéig igyekszünk összeállítani egy kezdetleges, Netfilteren alapuló tűzfalat). Ha van egy kis szabadidőd, kedves olvasó, hasonlítsd össze az amerikai és a magyar szerzők cikkeit! Például ***Bruce Byfield*** a 36. oldalon ír a jelszavakról általában, és kitér a PAM-ra. *Mátó Péter* cikkében is szóba kerül a PAM, igaz kicsit más szemszögből. Vagy ott van a GPG, amelyről ***Mick Bauer*** közöl egy cikket a 42. oldaltól, majd szintén előkerül ***Varga Csaba*** írásában, ahol a Mutt levelezőprogram kerül terítékre.

Szóval titkosíthatunk, biztonságossá tehetünk bármit. Csak nehogy úgy járjunk, ahogy én szoktam, nevezetesen, hogy a jelszót másnapra elfelejtem, és az anyag onnantól kezdve a teljesen titkos kategóriába kerül. Találkoztam már olyan rendszergazdákkal is, akik saját maguknak küldözgettek levelet, „akkor biztos nem felejtem el” felkiáltással. Ez sem rossz módszer, de a jelszavakat tartalmazó leveleket megint csak titkosítani kell, ugye.

De lépjünk tovább. A múlt hónapban írtam, hogy mennyire fontosnak tartom, hogy a nyílt és a szabad programok követői összefogjanak. Az írás kapcsán számos véleményt kaptam, természetesen akadtak köztük rosszalló levelek: „miért kellene szeretnünk a mások munkájából meggazdagodó ingyenélőket”, illetve „miként képzeled, hogy pénzt tudok keresni, ha ingyen adom a programom”, és így tovább. Időközben volt szerencsém egy hosszabb riportot készíteni ***Sebastien Blondeel***-lel, az FSF franciaországi részlegének egyik kiemelkedő alakjával (10. oldal), akinek záró gondolatát mindenkinek szíves figyelmébe ajánlom!

De akad más nagyon izgalmas hír is! Az LME épphogy kezdett magára találni, máris komoly döntés elé állítottak. Történt ugyanis, hogy megjelent egy hír, miszerint a Microsoft és a Miniszterelnöki Hivatal között megállapodás jött létre, a MeH a Microsoftnak nagy kalap pénzt fizet, ezért cserébe a diákok ingyen használhatják a Microsoft termékeit. Erről gyorsan kiderült, hogy ***nem egészen így van***. A Microsoft honlapján, a ***Sajtóközlemények*** alatt jelent meg ➲ http://www.microsoft.com/hun/news/default.asp?Type=Doc&SubPage=010920_edu (a közzé tételétől azonban el kellett tekintenünk, ugyanis még a sajtóközlemény alatt is az szerepel, hogy csak akkor tölthető le, ha a mellékelt szerződést – ez több oldalt tenne ki – is megjelentetnénk és a teljes szöveget közölnénk, valamint ha kizárólag tájékoztatási célból, szigorúan csak személyes használatra szeretnénk letölteni…), hogy mégsem ingyen és nem mindet.

Akkor mi is a megállapodás lényege? A felsőoktatásban résztvevő hallgatók és oktatók a ***megállapodás tárgyát*** képező termékeket tanulási célra használhatják. Ehhez egy Campus szerződést kell kötni, ennek keretén belül a Microsoft hitelesített viszonteladója kedvezményes vásárlási díjat ajánl fel a vásárlónak. Nem arról van szó tehát, hogy a MeH óriási összeget fizetett, és ezért cserébe a hallgatóknak és oktatóknak egy évig nem kell félniük a BSA fenyegetésétől. Inkább arról, hogy bizonyos termékek oktatási változatát kedvezményesen vásárolhatják meg. Az óriásvállalat egyébként éves szinten egymilliárd forintot kér, az első év költségeit a MeH állja, a további két évben pedig a MeH és az Oktatásügyi Minisztérium közösen fizeti. Az MTI honlapján megjelent változat szerint is ingyen juthatnak a termékekhez az oktatók és a diákok. Mások attól félnek, hogy így a Microsoft egy olyan címlistát gyűjt össze, amellyel azután bármilyen fejvadász vagy reklámcég igényét ki tudja elégíteni. Szóval a helyzet zavaros.

Az LME is közzétett egy nyilatkozatot – a Magyarországi BSD Egyesülettel közösen –, ezt a 14. oldalon mi is megjelentetjük.

Engem leginkább egyik olvasónk véleménye gondolkodtatott el, aki azt írta, hogy ez a lépés a Microsoft számára több szempontból is piaci helyzetének megerősödését jelenti, hiszen biztosítja, hogy oktatási termékeket vásároljanak (számos rendszer esetén ingyenes az oktatási változat), így további bevételhez jut (az éves egymilliárd mellett). Másrészt pedig annak feltételeit is megteremti, hogy az oktatás a lehető legnagyobb arányban MS-termékeken folyjon, ez pedig hosszútávon azt eredményezi, hogy a pár év múlva végző diákok továbbra is csak az óriásvállalat termékeihez értenek majd.

Akárhogy is nézzük, egy történetnek mindig legalább két oldala van. Ha arra gondolunk, hogy a MeH már több olyan pályázatot és tervet is támogatott, ami egyértelműen a nyílt és szabad rendszerek mellett foglalt állást, feltételezhetjük, hogy a jó szándék vezérelte őket ebben az esetben is. Remélem, hogy sikerül a MeH oldaláról egy illetékes munkatárssal riportot készítenünk, és tisztáznunk minden kérdést. Tőled, kedves olvasó, azt kérem, ha akad olyan gondolatod, kérdésed, melyet a témában szívesen megosztanál mindenkivel, írd meg nekem!

E bonyodalmas téma után levezetőként evezzünk boldogabb vizekre! A GNU/Linux-rendszerek egyre több helyen bizonyítják, hogy az üzleti életben is megállják helyüket. E hónapban ***Kósa Attila*** egy állami intézménybe kalauzol el bennünket, ahol az informatikai részleg meglepően értelmes és előrelátó döntésekkel már hosszú évek óta használja a szabad rendszereket. A Vas Megyei Bíróságról van szó. Ezt az érdekes és tanulságos cikket a 18. oldalon olvashatjuk.

Bár az, hogy milyen Linux-változat kerüljön az ügyfélgépekre, mindig is komoly vitát jelentett. Mindenesetre érdemes minél több rendszert megismerni. E hónapban a Mandrake Linux új, 8.1-es változatáról írunk, amely megtalálható lemezmellékletünkön is. Kellemes linuxozást kívánok!

Szy György
a Linuxvilág főszerkesztője, a Kiskapu Kiadó vezetője. Mindenki véleményét és levelét örömmel várja az alábbi levélcímen: Szy.Gyorgy@linuxvilag.hu

A LILO és korlátai

Az előző számunkban megjelent ***Gumimatraccal vagy légykukaccal indítsuk a Linuxot?*** című írásunkban benn maradt egy hiba a LILO-val kapcsolatban, amelyre ***Lakos Imre*** hívta fel szerzőnk és természetesen a mi figyelmünket is. Levelét az alábbiakban teljes terjedelemben közöljük:
Feladó: Lakos Imre ➲ lakosimi@freemail.hu
Címzett: ➲ ratiosoft@freemail.hu
Tárgy: Gumimatractúra

Tisztelt Szaló István!
Szeretettel meghívom egy gumimatractúrára. Nem, nem kell sehova sem elutaznia, csak itt helyben maradunk a gép előtt, ez is elég! Örömmel fogadtam cikkét a ***Linuxvilág*** szeptemberi számában, és érdeklődéssel olvastam a GRUB-ról írt sorokat, melyben a LILO-ról is ejtett néhány szót. Ez az, ami arra indított, hogy billentyűzetet „ragadjak".
A LILO, ez a jó kis öreg program, mely egy maroknyi csapatnak hála – ***Werner Almesberger***-rel az élen – napról napra megújul. Majdhogynem azt mondhatom: két dolog miatt (bár több oka is van) választottam, illetve maradtam a LILO-nál:
1. Nem szükséges megadni a `mem=256M` sort a /etc/lilo.conf-ban.
2. Akár a 2030. cilindertől is lehet indítani a Linux-rendszert. Nem, nincs itt szó semmilyen félreértésről! Nem is valamilyen szándékos kötözködés indított, csak az 1024. cilinder alatti időszaknak rég vége!
Amikor megjelent a Linux című könyv újabb „átdolgozott(?), javított" kiadása, én már akkor jeleztem a kiadónak, hogy ugyan már, ne poroljuk le ezt a régi dolgot és ne vegyük elő újra! Ezzel együtt a többi hiba kijavítására is megkértem őket, illetve a lapjukban egy „hibajavítás" megjelentetésére, de erre azóta sem került sor.
Tehát itt mellékelném a merevlemezem felosztási táblájáról (partition table) származó adatfájlt. Íme:

```
Disk /dev/hda: 255 heads, 63 sectors, 2482
cylinders
Units = cylinders of 16065 * 512 bytes
Device    Boot   Start     End    Blocks   Id
System
/dev/hda1  *    1      4      32098+   83     Linux
/dev/hda2       5    820    6554520   83     Linux
/dev/hda3     821    951    1052257+  83     Linux
/dev/hda4     952   2482   12297757+   5   Extended
/dev/hda5     952   1082    1052226   83     Linux
/dev/hda6    1083   1898    6554488+  83     Linux
/dev/hda7    1899   2029    1052226   83     Linux
/dev/hda8    2030   2482    3638691   83     Linux
```

Ez egy 20 GB-os Quantum Lct20-as merevlemez. A ***/dev/hda8***-as lemezterületen szoktam az új Linux-terjesztéseket kipróbálni, és itt található a ***/boot*** könyvtár és a rendszermag is.
A ***/dev/hda1***-es lemezterület kizárólag a Reiser FS használatához lett kialakítva, mivel a LILO nem szerette, ha Reiser FS-sel „kezdődik" a merevlemez, emiatt valahogy nem minden esetben írta be az új LILO-bejegyzést.
És itt van az említett lemezterületről a ***/etc/fstab*** fájl:

```
/dev/hda8  /reiserfs  defaults 1                    1
/dev/hdb   /cdrom     auto     ro,noauto,user,exec  0  0
/dev/fd0   /floppy    auto     noauto,user          0  0
proc       /proc      proc     defaults             0  0
# End of YaST-generated fstab lines
```

A titok nyitja továbbá a ***/etc/lilo.conf*** fájl egy részlete:

```
# Ez a LILO konfigurÆci s fÆjl
#
# Figyelem! Az append="mem=256MB" opci  csak
# a 2.2.16-os rendszermagig kell!!!
#
#*** A YAST-tal ne piszkÆlj bele, mert
#  sszekavarja!! ***
#
# Csak a "#" karaktert kell eltÆvol tanod, Øs
# a lilo parancsot lefuttatnod, Øs mßk dik az
# œjabb opci .
#restricted # Csak single user m dban kØr
# jelsz t ezzel az opci val.
#append="mem=256M" #ezt itt is hozzÆ lehet
# adni, ha kell. LÆsd fent!
boot=/dev/hda
#compact  # ´gy gyorsabb az ind tÆs, de nem
# mßk dik minden rendszeren,
# Øs  tk zik az lba32 opci val.
#vga=ASK # VESA m dban indul, Øs megadhat ,
# pl.: 0301-es opci  (640 480)
# vagy akÆr mÆs felbontÆs is vÆlaszthat ....
#vga=0x312  # Ezzel mindig 640 480-as
# felbontÆsban 32 bit sz nmØlysØggel indul.
lba32
# Ezzel mØg a 2330. cilinder f l l is lehet
# ind tani<<!!
message=/boot/message
======================================
read-only
prompt
timeout=150
default=Xwindow
menu-title=" dv zli a SuSE Linux 7.1"
# Felirat a LILO ablakÆban (DOS karaktertÆbla)
# End LILO global Section
# A grafikus bejelentkezı fel lettel indul.
image = /boot/linuxScsi
  root = /dev/hda8
  label = Xwindow
  append="5"

# CD-lemez  rÆsa SCSI-emulÆci val.
image = /boot/linuxScsi
  root = /dev/hda8
  label = CD  rÆs
  append="1 hdc=ide-scsi"
  vga=0x312

# Floppyr l ind tunk.
other = /dev/floppy
  label = Floppy
  unsafe
```

Tehát ez egy (az említett lemezterületen) használatban lévő fájl, amelyet jegyzetekkel láttam el, mivel másokra is gondoltam, azokra, akiknek feltettem a Linuxot.
Elérkeztünk gumimatractúránk végére. Nem vagyok egy szőrszálhasogató típus, viszont szeretem a pontosságot.
További jó munkát kívánok!
Üdvözlettel: Lakos Imre
Köszönjük az észrevételeket!

Programvadászat

Mandrake Linux 8.1

Olvasóink e havi CD-mellékletünkön a Mandrake Linux legfrissebb változatát találhatják. Ez a rendszer érdemli meg leginkább a „felhasználóbarát” jelzőt. Mellőzi az összes olyan „félmegoldást”, amit a fejlesztők a felhasználóbarát kiszolgálókba azzal a címszóval építenek be, hogy mindenki kedve szerint használhatja akár kiszolgálóként, akár munkaállomásként. A Mandrake kifejezetten munkaállomás, magyar nyelvű telepítője pedig elég fejlett ahhoz, hogy az is elboldoguljon vele, aki először találkozik Linux operációs rendszerrel. A rendszer beállítása is nagyon egyszerű. Memóriaigénye elég nagy, a számos cirkalmas szolgáltatást látva azonban érthető, hogy miért is lassabb ugyanazon a gépen, amelyen a Debian-rendszer könnyedén és zökkenőmentesen fut. Nagyon sok olvasói levelet és telefont kapok a „melyik Linuxot válasszam?” kérdéssel kapcsolatban. A saját tapasztalataim alapján csupán azt tudom javasolni: otthoni gépnek mindenképpen a Mandrake-et, kiszolgálónak pedig a Debiant választanám. Ezzel természetesen vitába lehet szállni, de mindenkinek megvan a saját kívánságlistája, így hangsúlyozom, ez csupán az én véleményemet tükrözi. Biztosan akadnak olyanok is, akinek a SuSE, esetleg a RedHat vagy bármely más Linux-terjesztés a kedvence, én mindenütt mindenre Debiant használok. Arra azonban mindenkinek felhívom a figyelmét, hogy mindenkor a feladat szabja meg, milyen Linuxot válasszunk.

Milyen is a Mandrake valójában?

Nos, szerintem könnyűszerrel telepíthető, beállítható és használható rendszer. A Linuxszal éppen csak ismerkedőket nem nehéz elriasztani, ugyanakkor az új rendszert könnyedén meg is lehet szerettetni. Debian-telepítés közben egy avatatlan szemlélő azt mondta, hogy kíváncsi lenne, az agyam hány százalékát foglalják el a különböző varázsszavak, amelyekkel életet lehelek a rendszerbe. A Mandrake Linuxnál erről egyáltalán nincs szó, itt minden egérkattintással és néhány billentyű lenyomásával „elintézhető”. A teljes telepítés mindössze tizenöt percet vett igénybe, a hálózat, a nyomtató és az X beállításával együtt. Miután újraindítottam a gépemet, máris kellemes grafikus környezetben találtam magam. A menük és az egész felület szinte teljesen

magyarul szól hozzánk, így még a kezdő angoltudással sem rendelkezők is játszi könnyedséggel eligazodhatnak az „új” felületen. Mivel a gépem hátulján egy Matrox G450-es grafikus kártya két feje kukucskál ki, gondoltam, kipróbálom, milyen is a xinerama és a két különálló munkafelület. Kellemes élményként említem ismét az egérkattintgatást, mert a beállítások egy perc múlva már működtek is. A xinerama-beállításként valójában egy 2048×768-as képernyőt kaptam, amelyen a programokat egyikről a másikra vontathattam. A képeken látható, hogy míg az egyik képernyőn gimpeltem, a másikon szöveget szerkesztettem. A 3D-beállítást a beállítási programban választhatjuk ki (aki játszani szeretne, mindenképpen ezt válassza!). Azt azonban nem tudtam beállítani, hogy az egyik monitoron más felbontásban és más frissítéssel működjön a grafikus felület. Ezután nekiálltam a hálózatban a megosztott könyvtárak feltérképezésének, ez is sikerült, úgyhogy a könyvtárszerkezetbe a sambás megosztások is bekerültek.

Mindent összevetve egy nagyon összefogott és kellemes rendszer lett a 8.1-es Mandrake. Mindenkinek ajánlom, aki még csak most kezd Linuxszal foglalkozni, sőt még azok számára is, akik már letették a szavazatukat valamely másik Linux mellett. Egyedüli zavaró hibájaként az róható fel, hogy nálam a csomagkezelőben az internetes forrás kiválasztása nem működött.

A Mandrake telepítéséről és beállításáról az 74. oldalon kezdődő írásunkban olvashatnak.

Kapcsolódó címek

- http://www.linux-mandrake.com/
- http://www.mandrakesoft.com
- http://www.mandrakestore.com
- http://www.mandrakeexpert.com
- http://www.mandrakecampus.com
- http://www.mandrakebizcases.com
- http://www.mandrakeforum.com
- http://www.mandrakeuser.org

A 20. CD-n természetesen a magazinban megjelent cikkekhez tartozó listák, illetve képek is megtalálhatók.

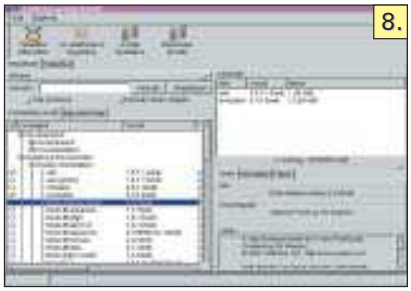
A rendszerben egyszerű felhasználóként nem tehetünk meg mindent, ami természetesen rendjén is van, hiszen rendszergazdánk valószínűleg a haját tépné, ha rendszerünket naponta kellene bütykölnie, mert mi azt találtuk mondani: „csak úgy kipróbáltam ezt meg azt!”. Ez nem áll az otthoni rendszerekre, mert ott valószínűleg ismerjük a rendszergazda (root) jelszavát, tehát szabadon garázdálkodhatunk benne.

Az erre alkalmas eszközök egyike a ***Vezérlőközpont*** (Control Center ***3. kép***) – a munkafelületre kattintva előugrik egy ablak (***4. kép***), ahol meg kell adnunk a rendszergazda jelszavát, és ha ezen sikeresen túljutottunk, a Vezérlőközpont barátságos felülete jelenik meg.

Nézzük, mit is lehet innen „vezérelni”! Az első menüpont a ***Rendszerindítás***, ahol indítólemezt készíthetünk frissen telepített rendszerünkhöz, beállíthatjuk a rendszerindításhoz használt programot (LILO/GRUB), az induláskor megjelenő kinézetet, továbbá az X rendszer önműködő indítását. Ha szeretnénk, egy olyan felhasználót is megadhatunk, aki az alapértelmezett bejelentkező lesz, így nem kell a felhasználó nevének és jelszavának begépelésével bajlódni

(kényelmes megoldás ugyan, de én nem ajánlom, mivel ha egynél több ember használja ugyanazt a gépet, az adatok és egyéb személyes anyagaink szabad prédának bizonyulhatnak mindenki számára). Végül pedig önműködő telepítőlemezt készíthetünk, amelyen rendszerünk beállításait tárolhatjuk (így készíthetünk másolatokat Mandrake Linuxunkról).
A következő menüpont a ***Hardver*** (***6. kép***), itt állíthatjuk be a gépünkben található alkatrészeket. Először az X beállítását tudjuk finomhangolni.
A ***Hardver*** almenüben megnézhetjük,

milyenek is a gépünkben lévő alkatrészek. Az ***Egér***-nél mutatóeszközünket tudjuk kezessé tenni, továbbá lehetőségünk nyílik a ***Nyomtató***, a ***Billentyűzet*** és a ***Csatlakoztatási pontok*** (befűzési pontok) beállítására is. Nagyon kellemes szolgáltatás a Samba önálló megosztáskeresője, amely az összes megosztást hibátlanul felismerte, így könnyűszerrel használatba vehettük őket.



Lássuk, mit is csinálhatunk a ***Hálózat és Internet*** menüben. A ***Kapcsolat*** almenüben varázsló segítségével tudjuk a gépünkhöz csatlakoztatott összes hálózati eszközt beállítani, amelyek hagyományos modemes, ISDN, ADSL, kábeles vagy helyi hálózati kapcsolatok lehetnek. A ***Kapcsolatmegosztás***-nál pedig rendszerünk hálózati erőforrásait oszthatjuk meg a többi számítógéppel. A beállítás után rendszerünk DHCP segítségével képes címet osztani.
A ***Biztonság*** szintén testreszabható (valamilyen mértékben) a Vezérlőközpontból. A ***Biztonsági szint*** beállításánál a következőket olvashatjuk: ***Válassza ki a biztonsági szintet!***
Alacsony – ezen a szinten néhány biztonsági kiegészítés lép életbe, több a biztonsági ellenőrzés és a figyelmeztetés.
Közepes – ezt a biztonsági szintet javasoljuk, ha a gép ügyfélként csatlakozik az Internethez. A biztonsági ellenőrzések életbe lépnek.
Magas – ezzel a biztonsági szinttel használhatjuk gépünket kiszolgálóként. Ez a biztonsági szint elég magas ahhoz, hogy a rendszer számos, hálózaton keresztül csatlakozó ügyfelet kiszolgáljon.
A ***Tűzfal*** almenünél a TinyFirewall programot szabhatjuk testre.

Már csak egyetlen menüpont maradt, a ***Rendszer***, ami a következő, feladatukról nevükben is sokat elárulo menüpontokat tartalmazza: ***Menük***, ***Szolgáltatások***, ***Betűtípusok***, ***Dátum és idő***, ***Szoftverkezelő***, ***Naplók*** és ***Konzol***. Ezek közül csak néhányat tekintünk át. A ***Szolgáltatások*** pontnál a rendszer indulásánál elindítandó szolgáltatásokat lehet beállítani (ilyen lehet például az Apache, a DHCPés az IP Tables).
A ***Szoftverkezelő***-ben programokat adhatunk hozzá és távolíthatunk el. Remek függőségkezeléssel rendelkezik, így ha

egy csomagot telepíteni akarunk, ez azonban egy olyan másik csomagtól függ, amely hiányzik a rendszerünkről, akkor azt is telepíti. A ***Konzol*** menüpont pedig rendszergazdai héjat ad, ahol mindenféle építő és romboló kedvünket kiélhetjük.
Ha a megfelelő programcsomag esetleg nincs telepítve, a rendszer a megfelelő számú CD-lemezt kéri, és pillanatok alatt elvégzi a telepítést.

Remélem, írásunkkal a kezdő linuxosok életét sikerült egy kicsit megkönnyíteni. Az egyik felhasználó „támadási felületet”talált a Mandrake 8.1 devfs megvalósításában. A hiba elhárításáig a gyártó azt kéri, hogy `devfs=nomount` kapcsolóval indítsa mindenki a rendszert.

Csontos Gyula (Csontos.Gyula@linuxvilag.hu) a Linuxvilág szakmai, hír- és CD-szerkesztője. Szabadidejében szívesen mászik hegyet és kerékpározik.

Linux franciául

Ha valakit a Linux használatától esetleg az riasztott el, hogy a legtöbb leírás és segítség csak angol nyelven érhető el, ugyanakkor ért franciául, annak megoldást jelenthetnek az alábbi francia linuxos oldalak címei. Számos HOGYAN is hozzáférhető, tudomást szerezhetünk a legfrissebb hírekről, valamint betekintést nyerhetünk a francia linuxosok életébe. Bár az én francia nyelvtudásom egyenlő a nullával, mégis egyszerűen elbolyongtam ezeken az oldalakon, ami számomra azt mutatja, hogy a felépítésük logikus és jól használható.

Kellemes vadászatot!

➲ http://glinux.tuxfamily.org/
➲ http://www.linux-france.org/article/cesar/
➲ http://www.linuxfr.org

Képkereső a Google-on

A Google internetes kereső új szolgáltatással bővíti kínálatát, ez a szolgáltatás a képkereső. Bármilyen szóra kereshetünk ugyanúgy, mint más böngészőkben, viszont az eredmény mindenképpen képi anyag lesz. Teljesítményét és tudását tekintve igen kiemelkedő: a ***Linuxvilág*** szóra rákeresve pár ezredmásodperc alatt eredményt kaptam. A keresési feltételek ugyanúgy működnek, mintha egyszerű keresést hajtanánk végre. Működnek az ékezetek, úgyhogy nyugodtan kereshetünk akár ***árvíztűrő tükörfúrógépeket*** is. Kellemes tulajdonságai között megemlíteném, hogyha a képre kattintunk, a kép nemcsak egyszerűen megjelenik, hanem egy kettéosztott ablakot kapunk felülső részében a képpel, az alsó ban pedig azzal a honlappal, ahonnan a kép származik. Nagyon kényelmes és hasznos szolgáltatásnak tartom azt is, hogy a keresési feltételt megőrizvén bármely másik keresőrészre kattintva a keresés ismét lefut , így egy adott témában a weboldalakon, a képek között, a hírcsoportokban és a „könyvtárakban" is körülnézhetünk. Mindenkinek ajánlom a kipróbálását!

➲ http://images.google.com

Védjük hazánkat szabadon!

A Magyar Honvédség is a Linux operációs rendszer mellett döntött. Linuxos számítógépekkel támogatja a Miniszterelnöki Hivatal Informatikai Kormánybiztossága a Magyar Honvédség 11 helyőrségét. A Miniszterelnöki Hivatal összesen 352 számítógépet, kabinetenként egy kiszolgálót és egy kivetítőt, a szükséges Linux operációs rendszert, valamint a kapcsolódó irodaautomatizálási programokat és a Microsoft programkörnyezet-felhasználási szerződéseit biztosítja. Az oktatókabinetekben, melyek napi 8 órában várják a hallgatókat, 12 ezer sorkatona tanulhat számítógép-használati ismereteket.

➲ http://www.meh.hu

PicoGUI

A világ legkisebb grafikus felülete a PicoGUI. Az állítás minden tekintetben megfelel a valóságnak, mert mind az eszközök, mind a saját méretére is igaz. Elsősorban beágyazott rendszerekhez és kézi gépek grafikus felületeként használható. Felépítése nagyon hasonlít az X Window rendszeréhez, nagy rugalmasságú a kiszolgáló-ügyfél kapcsolatban, de az X-szel ellentétben a betűk,

a bittérképek és minden egyéb, amire az alkalmazásoknak szükségük lehet, be lettek építve a kiszolgálóba. Természetesen „bőrözhető", miáltal mindenki „kiválaszthatja" a neki tetsző felhasználói felületet. Az elf formátumban lévő program mérete száz kilobájt körüli. Ebben már megtalálható a kiszolgáló, továbbá a betűk és minden más adat – bár a fordítás számos beállítási lehetősége miatt a méret változhat. A témák mérete pedig a bittérképekétől függ, ami lehet néhány, de akár több száz kilobájt, az ***Aqua*** téma mérete például húsz kilobájtnyi. Képes 1×1 képpontos képernyők kezelésére is, aminek ugyan nem sok értelmét látom, de jól jellemzi a rendszer rugalmasságát.

➲ http://www.picogui.org

Novell-Microsoft per

A Novell Inc. bejelentette, hogy Salt Lake City kerületi bíróságán eljárást kezdeményezett a Microsoft Corporation ellen, mert az az állami és szövetségi törvényeket sértő módon valótlan és

félrevezető kijelentéseket tett a Novellről és termékeiről. A Novell intézkedéseket kezdeményez annak érdekében, hogy a Microsoft beszüntesse a minden alapot nélkülöző és félrevezető állításainak terjesztését, valamint helyreigazítást kér. Továbbá felszólítja a Microsoftot a jogsértő reklámozás beszüntetésére. Ezenkívül azt szeretnék, hogy a céget az okozott károk megtérítésére kötelezzék. A Microsoft valótlanságokra épülő és a Novell lejáratását célzó kampánya a NetWare 6 megjelenésére adott válasznak tűnik. „Megítélésünk szerint Magyarországon ilyenre nem kerülhet sor, mert mint az minden bizonnyal sokak számára emlékezetes, 1997-ben a Novell Magyarország fellépett a Microsoft Magyarország minden olyan hirdetése ellen, amely vitatható volt" – mondta ***Szittya Tamás***, a Novell Magyarország ügyvezető igazgatója. „E fellépés következményeként a Novell Magyarország és a Microsoft Magyarország között megállapodás jött létre, hogy a Microsoft tartózkodni fog a magyar piacon a jogsértő reklámtól. Például az olyan reklámoktól, amelyek akár a Novellt, akár más versenytársat mint céget vagy a termékeiket minősítené. Mivel a Microsoft Magyarország ez idáig tartotta magát ehhez a megállapodáshoz, nem számítunk az amerikaihoz hasonló eset előfordulására Magyarországon."

A Microsoft hamis kijelentéseit egy müzlis dobozhoz hasonló marketing-kiadványban juttatta el számos Novell vásárlóhoz az Egyesült Államok területén. A terméknév helyén a „Microsoft Server Crunch" felirat olvasható, a dobozon pedig a következő a Novellt érintő kifogásolt állítások szerepeltek: „Mikor is jár le a NetWare szavatossága? A Cambridge Technology Partners cég-

gel történő egyesülés után a Novell a programfejlesztés helyett a tanácsadásra összpontosít. Olyan kiszolgálófelületet használ, melyet a gyártó nem támogat teljes mértékben. Ez növekvő költségeket jelent, mivel a felület gyorsan elavulttá válik, és időrabló átalakításokat tesz szükségessé."

„A kérdések és állítások minden alapot nélkülöznek és félrevezetők" – mondta ***Stewart Nelson***, a Novell alelnöke. „A Microsoft a NetWare halálhírét költi, hogy félelmet és bizonytalanságot keltsen a Novell vásárlói körében, valamint a leendő vásárlók elriasztására törekszik, hogy ne Novell-alapokra építsék a rendszerüket. Az amerikai üzleti világ hozzászokhatott a Microsoft erőszakos üzleti stratégiájához, ezzel a kampánnyal azonban a Microsoft elvetette a sulykot. A Novell-lel, termékeivel és terméktámogatásával kapcsolatos állítások túlmennek az összehasonlító reklámok határain, és véleményünk szerint sértik az állami és szövetségi törvényeket."

A müzlisdoboz-kampány nem az első próbálkozás, amellyel a Microsoft bizonytalanságot kíván kelteni a Novell vásárlóiban és a piac résztvevőiben. Ez év áprilisában a Microsoft egy cikket tett közzé az MSN Money Centralban – ez azután a TheStreet.com webhelyre is felkerült –, amely a következőket tartalmazta: „A Novell nemrég bejelentette, hogy kilép a szoftverüzletből, és tevékenységét a hálózati tanácsadás és szolgáltatások területén folytatja."

A Novell felszólítására a Microsoft kénytelen volt módosítani az állításán.

A NetWare-rendszert érintő tájékoztatás és az iparági sajtó NetWare 6 operációs rendszerről szóló cikkei a következő címeken érhetők el:

➲ http://www.novell.com/netware6/ext_news.html
➲ http://www.novell.hu/news/
➲ http://www.novell.hu
➲ http://www.novell.com

Sun-újdonságok

StarOffice 6 próbaváltozat

Letölthető a StarOffice 6 próbaváltozata. Remélem, minél többen le is töltik, és ha hibákat észlelnek, értesítik ezekről a Sun csapatát is! Így végleges változatként jó minőségű irodai csomagot kapunk majd a Linux (Windows, Sun) operációs rendszerre. A csomag szövegszerkesztőt, táblázatkezelőt, webes kiadványszerkesztőt és adatbázismodult egyaránt tartalmaz. Ha valaki nem tudná letölteni, megrendelheti a posta- és az előállítási költség kifizetésével a ➲http://www.amazon.com oldalon.

A Sun nagy erőkkel kezdte meg a MacOS X-változat fejlesztését is, ezt a munkálatot azonban felfüggesztették. Ebben a változatban szerencsére már

megszabadultak a beépített ***Start*** menütől, a levelezőtől, a programok pedig külön-külön „dolgoznak", meggyorsítva a működést. Biztosan sok minden ismerős lesz azok számára, akik már használták az OpenOffice valamelyik példányát.

➲ http://www.sun.com/staroffice/6.0beta/
➲ http://www.sun.com/software/star/staroffice/6.0beta/get.html

Solaris 9 EA (Early Access) próbaváltozat

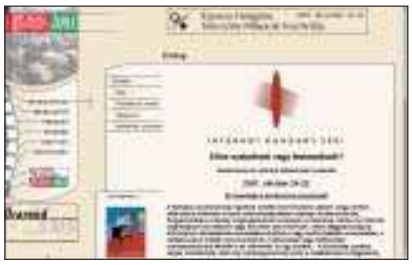
A Solaris 9 EA (Early Access) próbaváltozat szintén letölthető. A fájlokat az `unwrap_s9ia.sh` parancsfájllal lehet összefűzni.

➲ http://www.sun.com/software/solaris/programs/solaris9ea/
➲ ftp://ftp.fsn.hu/pub/CDROM-Images/solaris9/

Internettanácskozás Tihanyban

A hazai internetes világ legnagyobb szakmai tanácskozását október 24–25-én, Tihanyban tartják. „Előreszaladtunk vagy lemaradtunk?" a mottója az Internet Hungary 2001 elnevezésű szakmai tanácskozásnak, amelyet az internetes szakterületek képviselőinek és a Világháló felhasználói számára rendeznek – jelentette be ***Csermely Ákos***, a konferencia rendezőségének társelnöke.

A kétnapos rendezvényen a résztvevő szakemberek és érdeklődők elsősorban azokra a kérdésekre keresik a választ, hogy a tavalyi találkozó óta milyen változások történtek a hazai internetes szakmában. Az alábbi kérdéseket szándékoznak felvetni, amelyre a szakemberektől várják a feleletet: beigazolódtak-e az elmúlt évben megfogalmazott remények és félelmek? Milyen helyet tölt be az Internet a gazdaságban? Bekerülhet-e Magyarország az információs társadalmak élvonalába vagy behozhatatlan lemaradással küszködik? A vállalkozások milyen szerepet szánnak a Világhálónak: munkaeszközt vagy inkább kommunikációs csatornát látnak benne? ***Sükösd Miklós***, a rendezvény másik társelnöke az előadások sorából a kormányzati portállal és az önkormányzati internetoldalakkal foglalkozó, valamint az internetszabályozásról és a szerzői jogokról szóló vitát, továbbá a Világháló biztonsági kérdéseiről szóló beszélgetést emelte ki. Emellett szó lesz a média internetes megjelenéséről, a hírszolgáltatás és a szórakoztatás ötvöződéséről, a televíziós műsorok internetes felbukkanásáról is. Felhívjuk leendő résztvevő olvasóink figyelmét az izgalmasnak ígérkező záróvitára, amelyet „Ki a jobb?" címmel a Linux és a Microsoft képviselői



között rendeznek meg. A szervezők tájékoztatása szerint az esemény megnyitója és fővédnöke ***Stumpf István*** kancelláriaminiszter lesz.

A programok részletes listáját a

➲ http://www.mediahungaria.hu/aktualis/cimlap/ oldalon tekinthetik meg.

Beszélgetés az FSF Franciaország egyik képviselőjével

A szeptemberben megrendezett Linux Konferencia meghívott vendége *Richard M. Stallmann* volt, aki azonban a sajnálatos terrortámadás miatt kialakult helyzetben nem tudott Magyarországra repülni. Szerencsére a Free Software Foundation egyik francia vezéregyéniségét,

Sebastien Blondeel-t sikerült rábeszélni, hogy helyettesítse őt. Mint kiderült, Sebastien éppúgy fel tudta korbácsolni a kedélyeket, ahogyan Richard Stallmann tette volna. Mielőtt visszatért volna Franciaországba, sikerült egy rövid riportot készítenem vele.

Szy György: *Az előadásodon említetted, hogy Stallmann az LME képviselője felé írt első válaszlevelében megkérdezte, hogy az egyesület miért nem változtatja meg a nevét GNU/Linux-felhasználók Magyarországi Egyesületére. Miért tartjátok ennyire fontosnak, hogy kihangsúlyozzátok a GNU szerepét?*

Sebastien Blondeel: A Linux szó önmagában csak a rendszermagot jelenti, és az emberek sajnos hajlamosak rá, hogy elfeledkezzenek arról a rendkívül sok emberről, akik az egész környezetet kialakították. Állíthatom, hogy a GNU-termékek nélkül nem létezne olyan, hogy „Linux-rendszer". Gondolok itt a fordítókra, a héjakra, a rengeteg segédprogramra, valamint az egyéb külön futó programcsomagokra. Ezért kérünk mindenkit, hogy ne Linuxnak hívja az egész rendszert, hanem GNU/Linuxnak. Ugyancsak sajnálatosan sokszor fordul elő, hogy amikor a GNU szóba kerül, a felhasználó így válaszol: „Én Linuxot használok, mi az a GNU?" Úgy érezzük, a GNU csapata megérdemli, hogy a rendszer nevében is feltüntessük őket.

Sz. Gy.: *Úgy vélem, hogy a jelenlegi helyzetet elsősorban a Linux-változatok forgalmazóinak köszönhetjük, hiszen ha bemész egy boltba, akkor nem RedHat GNU/Linuxot, hanem RedHat Linuxot vásárolsz, nem?*

S. B.: Pontosan. Ezért is fontos számunkra, hogy a nagy cégek is használják a GNU nevét. Nálunk Franciaországban jó kapcsolatot építettünk ki a MandrakeSoft céggel, akik az új változatban már Mandrake GNU/Linuxnak hívják a rendszerüket. Ez komoly elismerés azon programozók számára, akik a rendszer elemeit már *Linus Torvalds* megjelenése előtt is hosszú évek óta fejlesztették.

Sz. Gy.: *Mi a helyzet a többi nagycéggel?*

S. B.: Hadd vegyem előre a Debian-rendszert. Már közhely, hogy a Debian GNU/Linux mögött nem áll egyetlen olyan cég, mint a Caldera, a SuSE vagy a RedHat. Nos, a Debian-rendszer nevében természetes elem a GNU. A nagycégek helyzetében azért egy kicsit nehezebb a kérdés, az okok között pedig az első helyen általában az eladhatóság szerepel. Egy ilyen változtatás például a SuSE vagy a RedHat háza táján a cégek marketingesei szerint komolyan rontana a termék eladhatóságán. A SuSE egyébként is különcnek számít, gondolok itt például a YAST telepítőrendszerre, amely a saját fejlesztésük. Nálunk fontos, hogy a rendszerek között felügyeleti szinten is átjárhatóság legyen.

Sz. Gy.: *Ha jól tudom, a linuxos berkeken belül más területeken is vannak viták, értem ez alatt például a Szabad Programok és a Nyílt Forrás elméletek különbözőségét.*

S. B.: Igen, a két elmélet között a fő különbség, hogy míg a Szabad Programok (Free Software) elmélet egyértelműen Richard M. Stallmann nevéhez kötődik és igazi szabadságot jelent, addig a Nyílt Forrás elmélet egy üzletemberek által kitalált rendszer. A Szabad Programok elmélet mögött nincsen cég, amely a termék bevételéből akarna megélni, csupán Richard M. Stallmann, és a hozzá kapcsolódó emberek. A Stallmann által képviselt irányvonal egyértelműen arra törekszik, hogy a felhasználó érdekeit védje. A GPL jogszerződés például lényegesen komolyabb előírásokat tartalmaz a továbbfejlesztés és a forgalmazás terén, melyek elsősorban az ingyenességet, valamint a használhatóságot kívánják védeni.

Sz. Gy.: *Stallmann nagyon komoly szerepet játszik ebben a mozgalomban. Mi vonzza hozzá az embereket?*

S. B.: Richard nagyon határozott egyéniség, ha valamit a fejébe vesz, azt végrehajtja. A „kezdetek kezdetén" Richard munkatársaival együtt szabadideje feláldozásával vágott bele az új ingyenes rendszer fejlesztésébe. Egyikőjük sem kapott érte egy huncut krajcárt sem. Egy idő után viszont mindegyik munkatársa elszegődött cégekhez dolgozni és Richard egyedül folytatta a kódolást. A történet szerint annyira jó programozó, hogy amit számos jól képzett programozó céges megbízásból készített, ő az ugyanazokat a feladatokat ellátó programokat ugyanannyi idő alatt egyedül alkotta meg. Richard a saját maga megbízója, és önmagával szemben is komolyan érvényesíti az elveit: a könyveket és programokat ingyen írja. A történethez hozzátartozik, hogy egyedülálló. A csoport növekedésével azonban szükségessé vált egy komoly szervezet kialakítása. A szervezet tagjai között ma már nemcsak önkéntesek vállalnak szerepet, az FSF-nek főállású és megbízásos alapon dolgozó munkatársai is vannak.

Sz. Gy.: *Az előadáson is felvetődött a kérdés: hogyan él meg egy programozó, ha ezt az elméletet követi?*

S. B.: Ez valóban gyakori kérdés. Itt két külön vonalat kell ismét szétválasztani. Az elsőhöz azok a programozók

tartoznak, akik a szabadidejüket feláldozva segítenek. Ezek az emberek azért dolgoznak, hogy jól és szabadon használható rendszerek legyenek. Nem pénzért munkálkodnak, az előállított terméket bárki használhatja. Az ő legnagyobb fizetségük, hogy létezik a GNU/Linux, amit bárhol használni tudnak, valamint egy-egy ilyen programban való részvétel komoly referenciát jelent. Természetesen sokan a megélhetésért programoznak. Itt elsősorban azt tartom fontosnak, hogy megkülönböztessük a „szokásos" kereskedelmi rendszert az új elméletektől. A jelenleg legtöbb helyen működő változat szerint egy cég megírat egy programot mondjuk ezer dollárért, majd százért árusítja. Ezután, ha a rendszer ötven példányban fogy el, a forgalmazó busás haszonra tett szert. Szerintem ebben az esetben a vállalat lényegesen több pénzt kért el, mint amennyit megérdemelt. Jobbnak tartom, ha a programozó megkapja az ezer dollárt, a program pedig ingyenes, bárki szabadon használhatja. Természetesen komoly kérdés, hogy honnan kapja meg a „munkás" az említett összeget.
Jelenleg legtöbbször az első megbízó fizeti a rendszer árát. Mivel neki a rendszer amúgy is kell, ezért akár ilyen, akár olyan csapatnak, de fizetni fog érte. Ez az a pont, ahol a cég dönt, nyílttá teszi-e a fejlesztés eredményét. Ha bárki számára elérhetővé teszi, más cégek költséget takaríthatnak meg, az eredeti megbízónak pedig nem lesz miatta vesztesége (kivéve természetesen, ha például egy olyan rendszerről van szó, amely a versenytársakat komoly ellenféllé teheti).

Sz. Gy.: *Rengeteg olyan rendszer van viszont, amely fizetős, és nem lehet megkerülni őket.*

S. B.: Igen, de a fizetős rendszereket viszont egyre többször írják újra a Szabad Programok elméletét követve. Hadd említsem az OpenSSH és GPG (az SSH és a PGP „újraírásai") példáját, amely azt mutatja, hogy az eredmény sokszor még az eredeti rendszernél is jobb és hatékonyabb lesz. Ez egyelőre még lassú, de fokozatosan gyorsuló folyamat, és minél drágábban árusítja valaki valamelyik termékét, annál többen akarják majd segíteni ezt a fajta „újraírást".

Sz. Gy.: *Tehát ha egy cég programot írat, készüljön fel rá, hogy a piacon hamarosan megjelenik ingyenes vetélytársa?*

S. B.: Ez természetes folyamat, de nem hiszem, hogy olyan világvége-hangulatra adna alapot, amilyennel a nagyobb cégek rémisztgetik a felhasználókat. Szerintem nem így kellene gondolkodni. A kódolás csupán töredéke a munkának. Egy rendszer akkor válik igazán értékessé, ha sokan használják. Egy-egy program kapcsán egyéb feladatok is jelentkeznek. Az új felhasználókat be kell tanítani, a rendszert üzemeltetni kell (szerencsére a GNU/Linux-rendszereknél ez kevesebb gondot jelent, mint sok más esetben), az új igényeknek megfelelő módosításokat üzembe kell állítani és a többi. Nem látom igazi értelmét, hogy ötezerszer kelljen megírni ugyanazt a számlázóprogramot. Olyan változásra számítok, mint amilyen például az ipari forradalom volt. Mindenki félt a szövőszéktől, de utólag visszatekintve komoly lépés volt az iparosodás folyamatában.

Sz. Gy.: *Ezek szerint az általatok épített jövőben minden program szabadon elérhető és ingyenes, csak összeállítom a rendszert és máris működik? Egy idő után nem is lesz szükség kódolókra?*

S. B.: Közel jársz az igazsághoz, de az, hogy ne lenne szükség igazi szakemberekre, nem igaz. *Erik S. Raymond* például azt írja, hogy a programozók kevesebb mint öt százaléka dolgozik a „dobozos" termékek előállításán, a többiek egyedi megbízások alapján tevékenykednek.

Sz. Gy.: *Térjünk vissza egy pillanatra az előadásodra. Az egyik fő mondandó az volt – ha jól értettem –, hogy ne vásároljunk programokat. Mi a helyzet, ha komoly határidővel kell számolni, és egy dobozos termék másnapra megoldaná a gondot?*

S. B.: Nagyon fontos, hogy ez az egész elmélet csak akkor működik, ha az emberek is elfogadják. Hiába írnak egy rendkívül jó Gimpet, ha a grafikusok egyszerűen ki sem próbálják. Nem azt állítom, hogy minden helyzetet meg lehet a szabad programokkal oldani, de elvárható, hogy adjunk nekik egy esélyt. Számos esetben a fizetős rendszerek megvásárlása helyett szabad termékeket próbáltak ki, és ezek a rendszerek beváltak; vagy olyan esetek is előfordulnak, amikor egy-egy drága rendszer megvásárlása helyett annak újraíratása mellett döntött a cég (vagy a lehetséges vásárlók egy csoportja), és a végeredmény nemhogy kevesebb költséggel készült el, de ráadásul hatékonyabb volt, mint fizetős testvére.
Az tény, hogy ha valaki átállt a szabad termékekre, szinte sosem tér vissza a fizetős termékek világába. Hadd említsek egy példát: a marokgépek, a PDA-k világát. Ezek a kis gépek régen 250 dollár körül mozogtak, WinCE rendszert futtattak, mely rendszer ára – mondjuk – harminc dollár körül mozgott. Nincs messze az idő, hogy a jobb marokgépek ára is akár harminc dollár alá csökken, és természetes, hogy ebből a bevételből a gyártók nem kívánnak rendkívül nagy részesedést a Microsoftnak továbbadni.

Sz. Gy.: *A programok világában tehát megerősödött, elismert a szervezet. Említetted, hogy az oktatás területén is komoly terveitek vannak. Mondanál erről néhány szót?*

S. B.: Igen, mint mondtam, az elmélet nemcsak a programokra érvényes, de a megvalósításhoz szükséges leghatékonyabb eszköz; az Internet a programozók mindegyike számára rendelkezésre áll, természetes tehát, hogy itt indult el. Az elv viszont, hogy aki ingyen szeretne segíteni, az segíthessen, az oktatásban is használható. A tankönyvek világában például évről évre megismétlődik, hogy egy-két ember (most ne menjünk bele a részletekbe) felkérést kap például egy történelem tankönyv megírására. Jó esetben az adott szerző ért is a témához, nekiáll, és ír egy négyszáz oldalas könyvet, amiben tíz fő témát tárgyal. A történészek (de más területek képviselői is) általában az anyag egy-egy részében érzik magukat igazán otthon. Lehet, hogy valaki a „sötét" középkorral foglalkozik húsz éve, míg egy másik

történész az első világháborút ismeri elölről-hátulról. Előfordul, hogy az, aki húsz éve a középkorral foglalkozik, akár ingyen is nagyon szívesen elkészítené az említett tankönyv egy fejezetét, egyrészt a neve feltüntetéséért, másrészt azért az elvért, hogy a diákok minél színvonalasabb tananyagot kapjanak.

Fut már egyébként egy nagy projekt, a GNUpedia, mely a fenti elv alapján igyekszik nemzetközileg használható, magas színvonalú enciklopédiát létrehozni. Mint látható, itt is kulcskérdés, hogy az oktatók internetközelben legyenek, tudják és akarják használni az adott lehetőségeket, eszközöket.

Sz. Gy.: *Már csak egy rövid kérdésre maradt időnk. Kérlek, engedj meg egy kérdést a magyarországi helyzettel kapcsolatban. Bizonyára hallottál róla, hogy a pártoló tábor növekedésével párhuzamosan nálunk is komoly személyes ellentétek ütötték fel a fejüket a hazai egyesületek és csoportok tárgyai között. Arra vagyok kíváncsi, hogy az ilyen típusú ütközéseket ti hogyan oldottátok meg?*

S. B.: Szó sincs róla, hogy nálunk ne lennének nagyon komoly ellenérdekek. Ez természetes. Az April például folyamatos vitában áll a Nyílt Forrás elméletet követő nagycégekkel. A kulcs talán az, hogy megértsük egymást és elfogadjuk, hogy a mi céljainkat nem kell mindenképpen ráerőltetni az egész világra. Annak ellenére, hogy vitáink vannak, egy oldalon állunk, és meg kell tanulnunk elfogadni, hogy mások is járhatják a saját útjukat. Megfontolt és hosszú távú célokat csak összefogással, egymás munkájára építve érhetünk el!

Sz. Gy.: *Köszönöm a beszélgetést, és további sikereket kívánok!*

Szy György

A Caldera Magyarországon

Magyarországon kevésbé ismert, pedig rendkívül jó nemzetközi visszhangnak örvend a Caldera Linux. A Caldera cégtől *Greg Bogochwalski*-val (Kelet-Európa és Közép Ázsia területi igazgatója) beszélgettünk a változásokról és a jövőről.

Szy György: *Magyarországon nem cseng annyira ismerősen a Caldera név. Pár szóval bemutatná, hogy a Caldera Linux milyen vásárlói közönséget céloz meg, illetve miben különbözik a többi, hozzá hasonló terméktől?*

Greg Bogochwalski: A Caldera jelenleg két fő vonalon tevékenykedik. Az első vonal a Caldera Linux, amelyet főleg fejlesztői felületek kialakításához, valamint kisebb kiszolgálókhoz ajánlunk. A Caldera Linux elsősorban az üzembiztosságot és a megbízhatóságot tartja szem előtt, olyan rendszert biztosítva, mely hosszú távon nem változik. Ha egy változatot összeállítottunk, azt szigorú ellenőrzés után „fagyasztjuk" fél évre. Ez idő alatt a rendszer főbb elemei nem módosulnak (természetesen hibajavításokat folyamatosan készítünk).

Sz. Gy.: *Ezzel a termékkel melyik vásárlói réteget célozzák meg?*

G. B.: Elsősorban a közép- és nagyvállalatokat, a bankokat, valamint a közigazgatást. Ezeken a területeken különösen fontos az egységes, könnyen kezelhető rendszer. Nagyon szép eredményeket értünk el Koreában, Ausztráliában, az Egyesült Államokban és további országokban is. Különösen fontos számunkra az is, hogy felhasználóink megfelelő támogatást kapjanak, ezért mindenhol komoly támogatási központok kialakítására törekszünk.

Sz. Gy.: *Magyarországon is létezik ilyen támogatási központ?*

G. B.: Jelenleg Magyarországon az ez irányú feladatokat az Areco szakemberei látják el, és meg kell mondanom, nagyon elégedettek vagyunk a szakmai felkészültségükkel. Ez nem mindig egyszerű feladat, hiszen a Caldera számos alapvető kiszolgálóprogram hitelesített felülete, ilyen például az Oracle 9i, sőt hamarosan az 11i. Ez nagy horderejű hír, hiszen egy ilyen fontos alkalmazás futtatása során egyetlen érthetetlen hiba is nagyon komoly következményeket vonhat maga után.

Sz. Gy.: *Az SCO beolvadt a Calderába. A másik fő feladatkör tehát a nagygépek területe?*

G. B.: Igen, pontosan. Az egybeolvadással a Caldera számos új ügyfelet szerzett, akik igényeinek továbbra is eleget kíván tenni. A cég az új Caldera Unix-rendszerrel kíván választási lehetőséget teremteni a vevők számára. Amennyiben egy feladat ellátására nem elég az egyprocesszoros, Linux-alapú rendszer, ott erős, méretezhető Unixot ajánlunk, amely pillanatok alatt feldolgozza a legnehezebb feladatokat is.

Sz. Gy.: *Térjünk vissza a Linuxhoz – sokszor hallom, hogy támadják a kereskedelmi cégeket, amiért egy ingyenes rendszert pénzért kínálnak. Mi a Caldera álláspontja, mely területeken teremti meg a bevételét?*

G. B.: Felmerül néha ilyen vélemény is, de a Caldera nagyon sokat tett a GNU- és a linuxos világ érdekében. Nagyon sok program eredetileg a Caldera védőszárnyai alatt készült, majd hasznos részévé vált a rendszereknek. Ilyen például az oly sokat használt RPM program is.

Mi elsősorban nem a vásárlókról kívánunk minél több bőrt lenyúzni. Fő tevékenységi területeink a nagyvállalatok. Egy olyan cégnek, mint például a McDonalds vagy a BMW, nem az a fontos, hogy mennyibe kerül egy termék, hanem az, hogy működjön. Mi arra szerződünk, hogy működni fog, és ha a termék nem állja a sarat,

hatalmas pénzektől esünk el. Ezért fontos számunkra, hogy a rendszer minél tökéletesebb legyen.

Sz. Gy.: *Az óriáscégeknél is alapvető kérdés az irodai munkaállomás. Milyen terveik vannak e téren?*

G. B.: Terveink között szerepel, hogy ne egy egyszerű munkaállomást biztosítsunk, hiszen ez kevés ahhoz, hogy erős vetélytársként induljunk olyan nagycégek ellen, mint például a Microsoft. Olyan átfogó programcsomagra van szükségünk, amely az összetett irodai feladatok elvégzését is lehetővé teszi, s nem csupán a szövegszerkesztést, táblázatkezelést, hanem a csoportmunka-támogatást is magában foglalja. Ebbe az irányba kacsintgat a Caldera Messaging Center, amely képes együttműködni a Microsoft Outlook-rendszerekkel és természetesen a StarOffice-szal is. Régen a Microsoft azért tört be a piacra, mert választási lehetőséget teremtett a Unixszal szemben. Most a Linux új lehetőséget kínál a Microsofttal szemben.

Sz. Gy.: *E területen melyek az akadályai a Linux térhódításának?*

G. B.: Ez lassú folyamat. Szerencsésnek tartom viszont az új XP-k megjelenését, így ugyanis mindenki újragondolhatja ezeket a kérdéseket. Magyarországon (és egyéb országokban is) alapvető kérdés viszont a honosítás, a termékek anyanyelvre történő átültetése. Ebből a szempontból még komoly munka vár a Linux-közösségre.

Sz. Gy.: *Itt szintén előfordulhat, hogy a Caldera egy-egy honosítást akár támogatna is?*

G. B.: Nem kizárt. Ha olyan csomagot tudunk összeállítani, amely tényleges munkakörnyezetet rejt magában, akkor ezért természetesen áldozni is hajlandók vagyunk. Sőt, akad olyan eset, hogy mi magunk is külsős munkatársakat keresünk, akik megbízásos alapon fordítanak. A Calderát egyáltalán nem zavarja, ha az elkészült termék szabad programként él tovább, a fontos az, hogy Caldera alatt is megbízhatóan működjön, továbbá a vevőink igényeinek megfeleljen.

Sz. Gy.: *Mely területen nyitnak még, milyen partnereket keresnek?*

G. B.: Az oktatás területén is sorsdöntő lépéseket szeretnénk tenni. Van ugyan angol, számítógépes alapú oktatási rendszerünk, de ezt Magyarországra is át szeretnénk ültetni. Itt sem elsősorban az a fontos, hogy nagy pénzeket „akasszunk le", hanem hogy Magyarországon is megbízható, erős és szakmailag felkészült piac alakuljon ki. Természetesen forgalmazókat is keresünk.

Sz. Gy.: *Még egy utolsó kérdés. Említette a Microsoft jelenlegi piaci helyzetét. Hogyan értékeli, mennyire veszélyes egymásra a Microsoft és a Linux-közösség?*

G. B.: Véleményem szerint a Microsoftra a legnagyobb veszélyt maga a Microsoft jelenti. Ahogy mondani szokták: maga alatt vágja a fát, és ez a fa hamarosan ki fog dőlni. A Linux csupán választási lehetőséget kínál a felhasználók számára.

Sz. Gy.: *Köszönöm a beszélgetést, és további sikereket kívánok!*

Szy György
a Linuxvilág főszerkesztője, a Kiskapu Kiadó vezetője. Mindenki véleményét és levelét örömmel várja az alábbi levélcímen: Szy.Gyorgy@linuxvilag.hu

Bejelentették a Samba 2.2.2 kiadást

Október 13-án, szombaton hivatalosan is bejelentették a Samba kiszolgáló 2.2.2-es változatát. Ez a legfrissebb megbízható változat, a frissítést minden termelésben résztvevő kiszolgálónál javasolják. Különösen fontos a frissítés, hiszen az új változatban több komoly hibát is kijavítottak. Az újdonságokról röviden:

Új démon: winbindd

Az új démon segítségével nss (Name Service Switch) szolgáltatást lehet megvalósítani, így részt vehet a rendszer a Windows NT/2000 tartományokban. Ezzel a rendszer képes fájl- és nyomtatómegosztást biztosítani a */etc/passwd*-ben nem szereplő felhasználók számára is. Még akadnak azonban befejezetlen részei is.

Új kapcsolók

Az *smb.conf*-ban használatos kapcsolók némelyike megváltozott, és újabbak is bekerültek. Többek között tovább javítják a Windows 2000-es rendszerekkel való együttműködést. Külön érdekes, hogy a megosztott nyomtatókat Windows 2000 alól helyi nyomtatóként (helyi vezérlővel) használhatjuk. Az LDAP-n tárolt SAM-adatbázis támogatása is bővült (ezért újrafordítást igényel). Az SSL-támogatás hibáit is (fordítás `--with-ssl` lehetőséggel) kijavították, és három új kapcsolóval bővítették tovább az amúgy sem rövid listát.

Új leírások

Két nagyon érdekes új leírással bővült a *docs/* könyvtár. Az első (README.Win2kSP2) a vándorló profilok Windows 2000 alatti támogatását tárgyalja, a második (README.Win32-Viruses) pedig ötleteket, tanácsokat ad, hogy miként harcolhatunk a vírusok ellen a Samba segítségével.

Emellett rengeteg egyéb hibajavítás és újraírás is történt, de ami külön figyelmet érdemel, hogy még júniusban felfedeztek egy hibát, amelyet kihasználva a támadó rendszergazdai jogosultságot szerezhet. A hiba lényege a %m makró használatából származott. Szerencsére, ritkán használják ezt a lehetőséget, ezért nem hallhattunk olyan komoly betörésről, amelyik ezt a gyenge pontot használta volna ki.

Az LME és a MBE közös nyilatkozata

A tervek szerint a lapzárta előtt napvilágot látott volna a Linux-felhasználók Magyarországi Egyesülete valamint a Magyar BSD Egyesület által közösen fogalmazott nyilatkozat. A nyilatkozat kapcsolatos Miniszterelnöki Hivatal és a Microsoft között szeptember 20-án létrejött szerződéssel, mely keretében a MeH első évben önállóan, majd a két következő évben az Oktatási Minisztériummal évenként egymillió forintot fizet a Microsoftnak, cserébe a hazai oktatók és hallgatók kedvezményes áron juthatnak hozzá az óriáscég több termékéhez.
Sajnos a nyilatkozattal kapcsolatos egyeztetések, illetve az elnökségi döntés október tizennyolcadikáig nem született meg, ezért a végleges szöveget nem tudjuk leközölni. A helyzetet bonyolítja, hogy a szerződésről több hírforrás fontos részletekben eltérő módon beszélt, nem volt például tisztázva, hogy pontosan milyen termékekről van szó, és hogy a kedvezményes ár milyen módon, milyen elmélet szerint számítandó.
Remélem, hogy a lap piacra kerülésekor a nyilatkozat már napvilágot látott. Amennyiben így van, a hivatalos változatot mindkét egyesület honlapján, valamint a ➲ linuxvilag.hu alatt is megtalálhatjuk.

Szy György

Szemünk fénye

Aki a szakmában van már egy jó évtizede, még emlékszik azokra az időkre, amikor csodájára jártunk, ha egy monitoron nem csak tizenhat szín jelent meg. A vágyak netovábbja a VGA monitorok voltak, még abba is beleegyeztünk, ha le kell mondani a színekről, hiszen a monokróm VGA két kategóriával jobb volt, mint a CGA. Azóta a világ sokat változott. Ma már nem is lehet ilyen ősállatokat kapni. Akkoriban még természetes volt, hogy a számítástechnika felkent papjai mind egytől egyig komoly szemkárosodással élnek együtt, ahogy egyik ismerősöm mondta, már a szemük is 60 Hertzen ketyegett.
Ma már szerencsére más a helyzet, komoly szabványok vonatkoznak a monitorokra, sőt az új lapmonitorok nem is „villognak”, teljesen más elméleten alapul működésük. Aki viszont nem olyan szerencsés, hogy egy ilyen többszázezer forint értékű falraakasztható képkeretet használjon, annak fontos, hogy figyeljen a szeme egészségére.
Mire is figyeljünk? Elsősorban a frissítési gyakoriságra. Ez az adat azt mondja meg, hogy a monitoron megjelenő kép másodpercenként hányszor „rajzolódik újra”. Ebből a szempontból a televízió is lehet viszonyítási alap. A tévékészülék általában 50 Hz-en dolgozik, egy-két óra tévézés után érzi is az ember, hogy villog a kép, ég a szeme stb. A monitorokra jellemző, hogy minél jobb egy termék, annál nagyobb felbontást és annál nagyobb gyakoriságot (60, 70, 75, 85, 100, 120 Hz, vagy még magasabb) tud kezelni. Mint említettem, a 60 Hz még bántja a szemet, a 75 a mai mércék szerint éppenhogy elfogadható, a 85 már szép képet biztosít. Az ennél magasabb frissítések csak különleges esetben (például 3D szemüveg) szükségesek.
Ennyit a gépi oldalról, de figyelnünk kell a szemünkre is! A programozó hajlamos arra, hogy alkotás közben „kikapcsolja” a testét, nem figyel rá. Pár óra munka (vagy játék, ugye…) után viszont azt érezzük, hogy fáj a szemünk, a fejünk, meg egyáltalán, mindenünk. De mi is történik?
A szem nagyon érzékeny, a szemgolyók mögött sok apró izom gondoskodik a mozgásról és az éleslátásról. Bármit is végzünk, általában nem mereven egy kis téglalapot nézünk, ezért szemizmaink mozognak, frissek, a vérellátás is rendesen működik. Amikor viszont elkezdjük folyamatotsan bambulni a monitort az izmok megmerevednek, és mivel a szemizmok állapotát nem érezzük olyan erősen, mint mondjuk a combunk izmait, észre sem vesszük, hogy az izmok számára természetellenes állapotban vannak. Ennek következménye rövidtávon szemégés, fejfájás, az összpontosítási képesség gyengülése, hosszú távon látásromlás és egyéb látási zavarok.
Ezért nagyon fontos, hogy vigyázzunk szemünkre! Ha nem megy másként, az elején folyamatosan figyeljük az órát és félóránként rövid szemtornát tartsunk, (legkésőbb) két-három óránként pedig iktassunk be negyed óra pihenőt.
Miből is áll egy ilyen szemtorna? Először is lazítsunk. Üljünk egyenesen, lazítsuk el a vállakat, majd csukjuk be a szemeket, és vegyünk két-három mély lélegzetet (nem baj, ha szemünk előtt ilyenkor színes ábrák vibrálnak, ez természetes). Ha nagyon nyúzott a szemünk, és már le sem akar csukódni, a két tenyerünket az arcunk elé rakva takarjuk be szemeinket, csak arra összpontosítva, hogy a szemizmokat ellazítsuk. Ezt a lazítást legalább fél percig végezzük. Itt is természetes jelenség, hogy a szem „ugrál”, hagyjuk, idővel kimegy a feszültség, a szemizmok újra megnyugszanak.
Lazítás után jön a torna! Nézzünk föl a fej mozgatása nélkül, amennyire tudunk, majd lassan le, az állunk felé. Ezt ismételjük meg tízszer. Majd ugyanígy balra és jobbra is. Most szép lassan föntről kezdve vezessük körbe szemünket a látóterünk szélén. Ebből is, majd a másik irányból is tíz kört. Ezután keressünk egy minél messzebb lévő, még élesen látható tárgyat, például a szemközti ház egy erkélyét. Emeljük fel a mutatóujjunkat az orrunk elé körülbelül egy arasszal, majd nézzünk először az ujjunk hegyére, majd az erkélyre. Ezt ismételgessük pár tucatszor.
Máris készen vagyunk, egy kis lazítás után jöhet a munka! (Vagy a SimCity.)

Szy György

III. GNU/Linux Szakmai Konferencia – 2001

Immáron harmadik alkalommal rendezte meg a Linux-felhasználók Magyarországi Egyesülete (LME) a GNU/Linux szakmai tanácskozást, amely a közkedvelt operációs rendszer mellett a nyílt forráskódú programfejlesztést is előtérbe helyezte. A neves magyar és külföldi előadókat is felvonultató, mára fontos szerepet betöltő rendezvénynek idén a Közép-Európai Egyetem (CEU) Konferencia Központja adott otthont. Az előadások mellett természetesen kiállítás és vásár is helyet kapott, ahol a legnagyobb hazai linuxos cégek is képviseltették magukat.

Biztonság mindenek felett...

Elmondhatjuk, hogy folytatódott az előző tanácskozás irányvonala. Az ebédszünetet leszámítva nem volt a napnak olyan időszaka, hogy ne lett volna legalább egy biztonságtechnikai témájú előadás. Ennek oka talán az lehet, hogy valóban nagy igény mutatkozott iránta.
Az utóbbi időben a Linux hódítása nem várt méreteket öltött, a gazdasági élet szinte minden területén alkalmazzák, ám ezzel együtt az emberek szemlélete nem változott, tehát nem fordítanak kellő figyelmet a biztonságra. A nyílt forrású, szabadon felhasználható programok elterjedése azonban azáltal segíthet kiküszöbölni ezt a folyamatot, hogy mindenki számára biztosítják a felhasználás és az ellenőrzés lehetőségét. Az előadók ezekre kívánták felhívni a hallgatóság figyelmét, majd mondandójukat kész megoldások ismertetésével szemléltették. Elsőként *Szentiványi Gábor*, a SuSE Magyarország Kft. munkatársa tartott előadást a tűzfalakról, a biztonsági stratégiákról és az alkalmazandó biztonságpolitikai alapszabályokról. Ezután részletesen, de közérthetően kifejtette a helyes fizikai felépítéssel és az általánosan használt programbéli megoldásokkal kapcsolatos álláspontját, majd arról is beszélt, hogy ezeket a feltételeket hogyan biztosítja számunkra egy Linux-rendszer.
Az előadásra természetesen nem bizonyult elegendőnek a tanácskozás programjában előirányzott 45 perc, a program közel két óra elteltével fejeződött be. Ezalatt egy másik helyszínen *Scheidler Balázs*, a Balabit Biztonságtechnikai Kft. előadója kötötte le a látogatók figyelmét, kicsit gép- és programközelibb, de még mindig védelemmel kapcsolatos előadásával, amelyben kész megoldásokat vázolt mindenki számára.
Sokan a szabad programoktól várják a ma még nem, vagy csak kisebb-nagyobb hibákkal működő eljárások rögös útjának kisimítását is. Erre utalt az a digitális aláírásokról tartott tájékoztató, amely a jelenlegi megoldások hiányosságaira mutatott rá, valamint mindezek elkerülésének módjait villantotta fel. A válasz természetesen a nagy tömegeket megmozgató, nyílt forráskódú fejlesztés. Ezeken túl szó esett még a biztonságos `chroot`-környezet kialakításáról, valamint a hasonló tulajdonságokkal bíró levelező- és webkiszolgálók építéséről és védelméről. Itt azonban már nem csak az adatbiztonság volt a fontos: a szolgáltatások célszerű felhasználásának igénye is előtérbe került. Ezek között szerepelt például a levélkiszolgálók esetében az adatforgalom mérséklésével előálló költségcsökkentés, valamint az is, hogy mindezek figyelembevételével miként kell felépítenünk az adott szolgáltatásokat nyújtó gépünket – természetesen GNU/Linux-rendszer felhasználásával.

Linuxot mindenféle géptípusra!

A nap egyik érdekes színfoltjának számított egy IBM rendszeren működő Linux bemutatása. Az előadást *Angyal László* és *Kolb Zoltán*, a Polygon Informatikai Kft. munkatársai tartották. Az első élő magyarországi bemutató során megismerkedhettünk a GNU/Linux alkalmazkodási képességeivel. A középpontban egy IBM iSeries (AS/400-típusú) kiszolgáló állt, amelyre előre felvarázsoltak egy Linuxot. Ezután a hallgatóságnak bemutatták a gépet, jellemezték főbb tulajdonságait, kiemelték előnyeit, majd a legelső lépésektől kezdve elmagyarázták a telepítési folyamatot: virtuális gépet alakítottak ki, lemezterületekre bontották, erőforrásokat foglaltak le, és beállították a virtuális eszközöket. A gépen már futó Linux-rendszeren könnyedén szemléltethették élő példával az összes lépést.

Egyre több területen hódít a Linux

Számtalan beszámoló szólt arról, hogy a Linux a gazdasági és társadalmi élet szinte minden területén használható. Ezt felismerve egyre többen kezdik alkalmazni mindennapos használatra, irodai- és multimédiás alkalmazások futtatására, de akár bonyolult matematikai műveletek számítására is, a fürtözés módszerét segítségül hívva. Az ezekről szóló beszámolók legalább annyi embert vonzottak, mint biztonságról szóló társaik.

Szabad programáramlás

Az eredeti program szerint a konferencia díszvendége *Richard M. Stallman*, a Szabad Szoftver Alapítvány (FSF) elnöke lett volna, aki a New York-i események miatt nem tudott eljönni. Helyette *Sebastien Blondeel*, az FSF európai képviselője tartotta az összevont fő előadást, amelyben a nyílt forráskódú programozásról és a GNU Linux operációs rendszerről beszélt. Minderről bővebben a vele készült riportban olvashatnak a 12. oldalon.

Hogyan tovább?

Megállapíthatjuk, hogy az idei tanácskozás hűen tükrözte a Linux és a nyílt forrású programok magyarországi helyzetét, erőviszonyait. Az ide látogatók képet kaphattak az új irányzatban rejlő lehetőségekről, kihasználásuk módszereiről. A tanácskozás évről-évre nagyobb méretű, és ez minden Linux-rajongó számára öröm.

Komáromi Zoltán
(komi_@freemail.hu) 21 éves, a BME hallgatója, mellette PHP-programozóként dolgozik. Kedvenc területe a multimédia. Kedveli a nagy társaságot, az érdekes embereket, a jó filmeket és mindent, ami mozgalmas. Szabadidejében röplabdázik.

Algoritmusok Afrikában (2. rész)

Afrika csodálatos és összetett világ! Ha Conakry utcáin sétálok, vagy egy eldugott kis faluba jutok el, körülölel az emberiesség melege, elárasztanak a színek, az életerő és az élet teljes sávszélessége.

Amikor Afrika kerül szóba, az úgynevezett digitális szakadék egyszerű szakadékká változik át, nem lesz benne semmi digitális. A szakadék egyrészt földrajzi, mert Afrika a messzi távolban fekszik, másrészt kulturális, mert az afrikai hagyományok és történelem az európaitól és az amerikaitól egyaránt függetlenül jött létre. És mintegy „mellékesen" megjegyezve: a szakadék gazdasági is, amennyiben a készpénzes erőforrásokat nézzük, vagy azt, hogy mi az emberek elképzelése a gazdagságról – ne feledkezzünk azonban meg róla, hogy a földrész természeti erőforrásokban gazdag. A szakadék végeredményben főként a tudatlanságból ered, amely Amerikánál a legszélesebb.

Az amerikaiak általában nagyon keveset tudnak Afrikáról, és ami keveset tudni vélnek, az is általában előítéletekből táplálkozik és téves. Ha Florida államot csak a hírműsorokból ismerném, azt gondolnám, hogy egy nagy, lakókocsikkal teli park, amelyekben kövér rózsaszínű emberek élnek, akik állandóan hurrikánok elől menekülnek. Hasonlóképpen a legtöbb amerikai számára Afrika csak egy katasztrófaövezet járványokkal, éhezéssel és népirtással. A képen, ami az átlagos amerikai képzeletében él az afrikai segélyezésről, általában egy bátor, fiatal (fehér) nő – ápolónő vagy önkéntes – szerepel, amint egy gyámoltalan fekete csecsemőt tart a kezében, mögötte a háttérben hálás és csodáló arckifejezésű afrikaiak állnak. Afrika természetesen egyáltalán nem ilyen, és a szakadékon átvezető hídon megtett első lépéssel együtt száműznünk kell ezeket a sértő közhelyeket, és mindent meg kell tennünk azért, hogy megtudhassuk, milyen is a *valódi* Afrika. Nem tenne jó szolgálatot a rajta élő embereknek, ha általánosítanék a földrésszel kapcsolatban, mintha csak egyetlen helyről lenne szó. Afrika csodálatos és összetett világ! Ha Conakry utcáin sétálok, vagy egy eldugott kis faluba jutok el, körülölel az emberiesség melege, elárasztanak a színek, az életerő és az élet teljes sávszélessége. Ez sokkal több, mint amit valaha – akár a legnagyobb képernyőn is, akár a leggyorsabb DSL-kapcsolattal – ki lehetne fejezni. Afrika az igazán széles sávú: kultúrák és sokszínűség, a nemzeti és az emberi tartalom teljes tárházát kínálja. Lehet, hogy a laposképernyős „valóság" nagy jelentőséggel bír az amerikai alkalmazott számára szánalmasan sivár dobozirodájában, de Afrika a személyes párbeszédről szól, valós időben.

A nyílt forráskód támogatói biztosak lehetnek benne, hogy az afrikaiak tudják, mi is a közösség. Afrikában létezik a bazár. Ezek a fogalmak az egész földrészen átitatják a kultúrákat és a hagyományokat – az afrikai társadalmak már jóval az RJ45-ös dugó feltalálása előtt a hálózatok mesterei voltak. Történelmük során mindig befogadták, sokszor vesztükre, a kultúrák között áramló vagy a külső népek által behozott nézeteket és újításokat. Általánosságban elmondhatjuk, hogy Afrika már korán és lelkesen befogadta az új távközlési eljárásokat, főleg amikor ezek célszerűek és olcsók voltak. Tíz évvel ezelőtt megdöbbentett a Botswanában egy főre eső faxgépek száma, ma pedig Guineában a mobiltelefonok egyre növekvő törvényes és feketepiacát látom. A közelmúltban meglátogattam az ország belsejében egy mecsetet, ahol is az ősz szakállú, bölcs müezzin kifejezetten azért vitt fel a minaret tetejére, hogy megmutassa a napenergiával működő erősítő- és hangszóró rendszert, amivel a híveket imára hívja.

Kevesebb, mint 90 éve történt – írta *Gina Kolata*, a New York Times tudományos újságírója Nátha című, nemrég megjelent könyvében –, hogy a Ladies Home Journal (hölgyeknek szóló lakberendezési magazin) a „living-room" kifejezés használatát javasolta a nappali megjelölésére, mivel a nappali az a helyiség, ahol az élet zajlik. A sors iróniája, hogy röviddel e kijelentés után a spanyolnátha, egy máig ismeretlen eredetű, világméretű influenzajárvány beköltözött az otthonokba és 1,5 millió amerikai halálát okozta, világszerte pedig negyvenmillió áldozatot szedett. Ez nem valami sötét középkori történet, 1918-ban történt. Napjainkban, amikor korunk pestise, az AIDS pusztít, újra felbukkan a tuberkulózis, és olyan új rejtélyek jelennek meg, mint például az embereknél tapasztalt Creutzfeldt-Jakob szindróma és a szarvasmarhákat megfertőző kergemarhakór összefüggése. Vajon elégséges lesz-e az orvostudomány ahhoz, hogy megmaradjunk, akárcsak mi, a legszerencsésebbnek nevezhető kisebbség?

Mindenesetre azoknak, akik másokon szeretnének segíteni, legalább a múltbeli hibáikból okulhatnának, és meg kellene kérdezniük maguktól, hogy a városszervezési modellek, a túlzsúfoltság, az erőforrás-felhasználás módja, a környezet kiaknázása vajon mások számára is fenntartható, kívánatos vagy kimondottan jó exportcikk-e ebben a világban.

A gondolatmenet során ekkor jutunk el odáig, hogy el tudjuk fogadni a következőt: elképzelhető, hogy az Internet nem jelent megoldást az emberiség minden gondjára-bajára. Még az is előfordulhat, hogy elegendő a türelmünk annak megértéséhez, miszerint a nagy változások véghezviteléhez több nemzedéknyi időre és türelemre lesz szükség. Mostanra világossá vált, hogy a Linux és a nyílt forrás fejlesztői azzal teszik a legtöbbet Afrikáért, amivel eddig is. Azok, akik az Internet lelkét adó, világraszóló szabad programokat írják és telepítik, fontos és mélyreható segítséget nyújtanak felhasználási szerződéstől függetlenül. A GNU és a nyílt forráskódú programok tökéletesen megfelelnek a felfutó afrikai nemzeteknek és a világ többi részének is – nemcsak azért, mert e rendszerek műszaki adottságai rendkívül jók, hanem legfőképpen ama értékeknek köszönhetően, amelyeket a fejlesztés során szem előtt tartanak.

Wayne Marshall
(guinix@yahoo.com)
Unix-programozó és technikai tanácsadó. A nyugat-afrikai Guineában él.

Szerzői jogi kérdések

A *Linux Journal* jogi szakértője most a szerzői jogok útvesztőjében vállal rövid túravezetést.

Szeretném elkészíteni egy kereskedelmi program nyílt forráskódú változatát. Megtehetem-e úgy, hogy ne pereljen be a program tulajdonosa?

Andre Durand, a Jabber.Com alapítója

Kizárólag a példa kedvéért azt fogom feltételezni, hogy egy olyan programnak szeretné elkészíteni a nyílt forráskódú változatát, mint amilyen a Windows. A legtöbb kereskedelmi program tulajdonosa (mint például a Windowsé) sajnos nem engedi meg, hogy a programkódot visszafejtsük vagy lemásoljuk. Ez azonban nem jelenti azt, hogy a saját tehetségünkre építve ne hozhatnánk létre egy kereskedelmi program akár javított és továbbfejlesztett változatát.

A kérdésre a rövid válasz: igen – mindaddig, amíg saját változat készül a programból, és nem sérti meg a kereskedelmi program szerzői jogait és szabadalmait.

A bíróságok egységesen úgy vélik, hogy a szerzői jog megsértését a szerzői joggal védett mű szándékos, illetve nem tudatos lemásolása meríti ki. Ezzel ellentétben, ha valaki a már meglévő védett anyagok felhasználása nélkül a saját munkájával állít elő valami hasonlót, nem történik jogsértés.

Ahogyan *Learned Hand* bíró fogalmazott: ahhoz, hogy egy szerzői jogi per sikeres legyen, „többet kell felmutatni, mint a feltételezett másolat és a kérdéses rész közti puszta hasonlóságot vagy akár azonosságot" (Fred Fisher Inc. kontra Dillingham D.C.N.Y. 1924 298 F.145, 147). Így folytatja:

„Két vagy több ember írhat ugyanarról a témáról, kezelheti azt hasonlóképpen, illetve használhatja ugyanazokat az anyagokat hasonló módon és ugyanazon célra. Műveik tartalmazhatják ugyanazokat a gondolatokat, érzéseket, elképzeléseket; akár azonosak is lehetnek. Az ilyen hasonlóság vagy egyezés csak annyiban lényeges, hogy megmutathatja, történt-e törvénytelen másolás."

A másolás tényét bizonyítékkal kell alátámasztani. A törvény megköveteli annak bizonyítását, hogy kifejezetten másolás történt, és amennyiben a bizonyítékok nem meggyőzőek, annak bizonyítását, hogy az eredeti hozzáférhető volt és „lényegbevágó hasonlóság" áll fenn. A bizonyítékokat többféleképpen lehet értelmezni, ami kockázatot rejt magában, ezért aki arra adja a fejét, hogy saját, független változatot készítsen egy kereskedelmi program alapján, az jobban teszi, ha pontos feljegyzést készít mindenről, amit tesz.

Amikor az újra megvalósítandó programot megvizsgáljuk, korlátozzuk magunkat a program feladatának és eljárásainak külső megfigyelésére. Emlékezzünk rá, hogy amennyiben a programunk túlságosan is hasonlít a kereskedelmi programhoz, a bíróság nem feltétlenül hiszi el, hogy nem láttuk és nem másoltuk le a szerzői jogvédelem alatt álló kódot. Minél kevesebbet látunk, annál valószínűtlenebb, hogy „lényegbevágóan hasonló" legyen a kódunk.

Ne másoljuk le a kereskedelmi program felhasználói leírását még akkor sem, ha az új program szándékaink szerint azonos módon fog működni. A leírás ilyesfajta másolása jogsértő, akkor is, ha maga a program nem sért szerzői jogokat.

Ha a forráskódot vagy a kereskedelmi program megvalósításának egyéb részleteit korábban megismerhettük, szükség lehet rá, hogy kizárjuk magunkat a nyílt forráskódú változat fejlesztéséből. Ennek egyik megvalósítási módja a nagy cégeknél a programmal szembeni követelmények rögzítése, és egy olyan fejlesztőkből álló csoportnak történő átadása, akik sohasem látták az eredeti kódot. Védekezésképpen érdemes ezt az eljárást kimerítően leírni.

Egy utolsó figyelmeztetés: az eredeti kereskedelmi program szerzői jogai nem terjednek ki a „gondolatokra," amelyeket a program megvalósít, csakis ezek „kifejezésére." Jogunk van a program gondolatainak újbóli megvalósítására anélkül, hogy a szerzői jogok megsértésétől tartanunk kellene. Ugyanakkor a gondolatokat szabadalmak védhetik. A szabadalommal védett megoldás puszta megalkotása, használata vagy árusítása kimerítheti a szabadalmi jog megsértését. Egy szabadalmat megsérthetünk anélkül, hogy lemásolnánk, még akkor is, ha független módon és jóhiszeműen jutunk el pontosan ugyanahhoz a gondolathoz; ha tehát a kereskedelmi program bizonyos részeit szabadalom védi, akkor meg kell találni annak a módját, hogy ezeket a részeket a szabadalmazott eljárás bárminemű fölhasználása nélkül valósítsuk meg.

Lawrence Rosen (www.rosenlav.com) magángyakorlatot folytató jogász a kaliforniai Redwood Cityben. A Nyílt Forrás Kezdeményezés (Open Source Initiative ügyvezető igazgatója és jogtanácsosa (➲ www.opensource.org).

A bíróságok egységesen úgy vélik, hogy a szerzői jog megsértését a szerzői joggal védett mű szándékos, illetve nem tudatos lemásolása meríti ki.

Ne másoljuk le a kereskedelmi program felhasználói leírását még akkor sem, ha az új program szándékunk szerint azonos módon fog működni.

Az eredeti kereskedelmi program szerzői jogai nem terjednek ki a „gondolatokra", amelyeket a program megvalósít, csakis ezek „kifejezésére."

Cégcsokor (2. rész)

Sorozatunkban olyan cégeket gyűjtünk csokorba, amelyek huzamosabb ideje számos területen Linuxot alkalmaznak. Lássuk, mi a helyzet a hazai bíróságokon.

Az informatika a kilencvenes évek elején jelent meg a bíróságokon. 1998-ig – az Országos Igazságszolgáltatási Tanács megalakulásáig – a bírósági informatikai fejlesztést a megyei bíróságok többé-kevésbé önállóan végezték. Általában a maradékelvet követték: ha maradt rá pénz, volt fejlesztés. (A szabályt erősítő kivételt a cégbíróságok országos hálózatának központi fejlesztése jelentette.)
1992 végétől a Vas Megyei Bíróságon – az akkortájt megszokott DOS-NetWare hálózat helyett – SCO Unix-alapú hálózatot helyeztek üzembe. Azért döntöttek így, mert a DOS-os világ korlátai már kezdtek kirajzolódni, ugyanakkor megfigyelhetők voltak az Internet kibontakozására utaló első jelek. Célszerűnek látszott az eleve nagyobb lehetőségeket engedő, egyszersmind komoly múlttal és referenciákkal rendelkező, de elérhető árú rendszer megismerése és alkalmazása. Természetesen követelmény volt a már meglévő programok illeszthetősége is. Úgy ítélték meg, hogy választásukkal a mozgásterüket nem szűkítették le: mind a DOS, mind Unix világa nyitott maradt.
A DOS-Windows 3.1-alapú ügyfeleket eleinte SCO-megfelelő program (PC-Interface) kapcsolta a kiszolgálóhoz. Később az ügyfelek számának növekedésével ingyenes (részben nyitott forráskódú) programok használatára tértek át. 1996-ig üzemelt ez a rendszer, az SCO lényegében fájlkiszolgáló, nyomtatókiszolgáló és CD-kiszolgáló (Complex Jogtár) feladatkört látott el. Időközben hálózatbővítés eredményeként egy másik SCO-kiszolgálót is üzembe helyeztek: a két kiszolgáló útválasztóként is működött, három koax ethernetszakaszt ellátva.
Linuxszal 1994-től kezdtek el foglalkozni egy magánúton beszerzett Yggdrasil-változatot tartalmazó CD segítségével (0.99.13-as rendszermag, X11R5, Postgres 4.1). Eleinte ügyfélként próbálgatták, de hamar nyilvánvalóvá vált, hogy kezesebb, mint az SCO. 1996-ban jött el az ideje, hogy a kiszolgálókra Linux kerüljön: Slackware-terjesztés 1.2-es sorozatú rendszermaggal. (Mindeközben a bíróság gazdasági hivatalán belül egy, az idő múlásával egyre „szürkülő” NetWare-hálózat működött nyolc géppel.)
Gyökeres változás 1998 őszén következett be. Ekkor minden megyei bíróság (az addig informatikára fordított összegekhez képest) komoly központi forrásokhoz jutott, amelyet a (megye)székhelyi bíróság informatikai fejlesztésére kellett fordítaniuk. Úgy döntöttek, hogy a legjobb költséghatékonyság elérése érdekében alapvetően linuxos rendszert alakítanak ki. Ekkorra már kezdett beindulni a „Linux-úthenger”, egyre több mértékadó folyóirat, cég, fórum volt kénytelen válaszolni a Linux-jelenségre, miáltal igazolva kezdték látni korábbi Unix-irányú döntésük helyességét. Végül a rendelkezésre álló tízmillió forintból a következő rendszert sikerült kialakítani:

- 36 végpontos, tisztán UTP-hálózat (addig mindenhol házilag barkácsolt koax ethernetet használtak);
- egy kiszolgáló (10 GB UWSCSI merevlemez, CD-író, CD-olvasó, 512 MB RAM, 24 GB DAT);
- harminc merevlemez nélküli (diskleses) PC (K6-200, illetve 300, 64 MB RAM, boot epromos hálózati kártya, 15 hüvelykes monitor);
- egy hálózati nyomtató (PostScript, A3, illetve A4, 32 lap/perc, kétoldalas, 3000 lapos adagoló);
- tíz személyi lézernyomtató (8 lap/perc).

A rendszert úgy alakították ki, hogy a leíró-, illetve kezelőirodákban az ott dolgozók 1–1 nyomtatót közösen használnak. A bírói szobákban nincs nyomtató, de hálózati változata mindenki számára elérhető (elsősorban a jogszabályok CD Jogtárból történő nyomtatására).
Operációs rendszerként a SuSE terjesztését alkalmazták, két okból is: egyrészt mert megítélésük szerint ez támogatta leginkább az új videovezérlőket, másrészt ezt kapták a bevezetett ApplixWare irodai program mellé.
Az ügyfél alapvetően kétféle lehet: X-terminál, illetve X-es munkaállomás, attól függően, hogy a felhasználói programok a kiszolgálón vagy az ügyfélen futnak. Ők ez utóbbit választották: így egyrészt az ügyfél processzorának teljesítménye megfelelően kihasznált, másrészt csökken a kiszolgáló terhelése, harmadrészt a helyi nyomtatás nem jelent nehézséget. Minthogy még DOS-os programokkal is rendelkeznek, kénytelenek DOSemut használni, ami szintén ilyen elrendezést kíván. Hátrányként el kell viselni a csereterület (swap) hiányát és a nagyobb helyi hálózati sávszélességigényt.
A következő (több részből álló) nagyobb beruházásra 1999 őszén került sor. Ekkor már a bíróság – a központosított közbeszerzés keretén belül – közvetlenül jutott eszközökhöz. A szállító céget megkérték, hogy a működő rendszerben próbálhassák ki az általuk felkínált gépet. Ennek során kiderült, hogy az Intel alaplap (valószínűleg BIOS-gondok miatt) nem kezelte a boot epromos hálózati kártyát. Az eredeti kiépítésen tehát módosítani kellett: egy másik (olcsóbb) alaplapot kértek a gépekbe merevlemez, hajlékonylemezes meghajtó, CD-olvasó és Windows nélkül, viszont gyorsabb processzorral, nagyobb memóriával, sőt – hogy a forintkeretet kitöltsék – még két további gépet és tíz lézernyomtatót is vásároltak.
Érdekes történet ebből az időből: a szállító értékesítési menedzsere eleinte nem tudta elhinni, hogy nem minden (lemez nélküli) géphez kell OEM Linux…
Amikor kiderültek az alaplappal kapcsolatos gondok, a szállítót nehéz feladatnak bizonyult rábeszélni a cserére. (Felmerült, hogy amennyiben a Boot Eprom nem megy, hajlékonylemezes meghajtót kérnek a gépekbe, de nyílásával a gép belseje felé fordítva, egyszer s mindenkorra benne hagyva az indítólemezt.)
Összesen hat kiszolgálót is kaptak volna a vidéki bíróságok ellátására (egy nagyobbat és öt kisebbet). Mivel csak két vidéki bíróságon van hálózat, az ötből hármat „becseréltek” a szállítónál, ezek helyett a bíróság két szombathelyi épülete között mikrohullámú összeköttetés kiépítését kérték. Az útválasztók természetesen kiöregedett 486-osok voltak, Linuxszal… A „nagy” kiszolgálót a megyei bíróságon állították munkába, az addigi kiszolgáló tartalékként maradt, illetve a mentéseket végzi (naponta DAT-ra, nagyobb időszakok végén, programfris-

sítés előtt CD-re is). Minthogy minden adat a kiszolgálón található, a napi mentést egy lépésben, egyszerű parancsállománnyal el lehet elvégezni.
Ezzel a beruházással lényegében Linux-alapon áll a bírósági informatikai rendszer, és egyúttal az egyre kínosabbá váló jogtisztasági kérdést is megoldották (az ApplixWare mellett fokozatosan bevezetik a StarOffice-t). Időközben saját erőből valósították meg az internetelérést ISDN-en keresztül, sőt, a körmendi és a sárvári bíróságok felé is közvetlen ISDN-adatkapcsolat van.
1999 végén egy Phare-projekt keretén belül kiépült a bíróságok országos hálózata is: minden megyei bíróság 16 kbit/s garantált sávszélességű kapcsolatot (Frame Relay) kapott az OITH irányába, valamint egy RS/6000 kiszolgálót (*AIX+Apache+Samba+Oracle InterOffice*, 14 felhasználó) a belső levelezés számára, és 14 IBM PC-t Windows NT-vel. Mivel a levelezőrendszer működése és kényelme némi kívánnivalót hagyott maga után, a 14 PC-t is inkább a helyi rendszerbe illesztették, kiiktatva ezáltal az NT-ket. Az országos hálózat IP-címeivel való ütközés miatt a helyi rendszert az RS/6000-ről (természetesen linuxos) tűzfallal választották le úgy, hogy az azon még futó InterOffice szükség esetén elérhető legyen.
Jelenleg a kiszolgálón a „szokásos" kiszolgálóprogramok futnak: *bootpd*, *NFS-kiszolgáló*, *Apache* (belső használatra), *Squid*, *Bind*, *Qmail*, *PostgreSQL*, *mars_nwe* stb.
A közeljövőben vírusirtó programot is kapnak, pontos szerepe még nem tisztázott… A kiszolgáló szolgáltatja a Jogtárat is, de nem CD-ről, hanem a gyorsaság kedvéért a merevlemezről. Az 57 feliratkozott felhasználó közül egyidejűleg általában 30–35-öt szolgál ki, kellemesen alacsony (0,1 körüli) terheléssel. Folyamatosan zajlik az internetes ismeretek oktatása a felhasználók számára, továbbá a Fővárosi Bíróság jóvoltából valamennyi felhasználó külön költség nélkül levelezhet.
Különlegességként: a titkos anyagokat *cfs* (crypto file system) segítségével kódolt formában a kiszolgálón tárolják. (Kódolás hiányában mobil adathordozót lenne szükséges használni és páncélszekrényben tárolni.)
Az ügyfeleken elsősorban szövegszerkesztés folyik, valamint nagyrészt Clipper-alapú nyilvántartásokat vezetnek. Ez utóbbiak továbbra is DOSemuból érhetők el, de – egy-két program esetén – mód van az ügyfél közvetlen DOS-os indítására is. Mindkét esetben az ügyfélgép egyúttal mars_nwe-ügyfél is. Néhány programjuk már linuxos környezetre készült, ezek egy része önálló fejlesztés, másikuk meglévő DOS-program helyettesítője, illetve kiegészítője. (Ez utóbbi esetekben a .DBF formátumú adatokat ültetik át PostgreSQL alá és úgy használják fel.)
Terveik között szerepel az LDAP bevezetése, például azonosítás céljára. Minthogy a lemez nélküli ügyfelek mind ugyanazt a *passwd*- és *shadow*-fájlt látják, az LDAP alkalmazása valójában nem sürgős, de szeretnének fölkészülni az egyéb LDAP-t igénylő programok fogadására, illetve készítésére. Tervezik további DOS-alapú programok kiváltását linuxos, illetve többfelületes programokkal; és újak készítését a bírósági ügyvitel még ellátatlan területeire; ezenkívül a vidéki bíróságok adatkapcsolatának kihasználását (levelezés, adatbázis-elérés). Szeretnének bootp-ről DHCP-re áttérni. A fejlesztések során elsősorban a Python–PostgreSQL párost használják. Tekintettel a roppant egyszerű és könnyen áttekinthető felépítésre hálózatfelügyelő programot egyelőre nem alkalmaznak.
A felhasználók többsége csak annyit tud a Linuxról, hogy más, mint a Windows. Korábban már többen használtak Windowst, esetükben a grafikus felület használata nem jelentett gondot, viszont az alkalmazások „mássága" igen. Az első időkben előfordultak véletlen dokumentumtörlések, mikor is a felhasználók első válasza többnyire az volt, hogy „rossz a gép". Akiknek nem volt számítógépes tapasztalatuk, azoknál a legnagyobb nehézséget az egér kezelésének megtanulása és az írógépes múlt során rögzült rossz szokások levetkőzése (az „1" és „l" karakterek, valamint a „0" és az „o" karakterek összekeverése, a *szóköz* billentyűvel végzett szövegformázás) jelentette. Ők alapszintű oktatásban részesültek, és néhány nap alatt belejöttek a gyakorlatba. A „fortélyokat" mindig egy-egy valódi eset kapcsán tanulták meg. Mivel korábban ugyanezek a nehézségek jelentkeztek a windowsos környezet kapcsán is, úgy tűnik, hogy a kérdések nem egyetlen rendszerre jellemzőek.

Az operációs rendszerrel magával csak az informatikusok találkoznak. Problémák abból szoktak adódni, hogy a .DOC- vagy .XLS-állományok bevitel után nem pontosan úgy néznek ki, mint egy windowsos rendszerben. Ha az azonosság mégis fontos, akkor általában más formában (például .RTF) kérik a küldőtől az adatokat. Nem okoz különösebb gondot a hajlékonylemezes meghajtók hiánya sem, mert kérésre bárki a hajlékonylemezen szállított dokumentumait felteszik a rendszerre.
Az irodai rendszerként használt Applixware menürendszere magyarított, ezért a használata egyszerű. Ha szokatlan üzenet megjelenésekor (mindegy, hogy angol vagy magyar nyelvű) a felhasználó bizonytalan a helyes válaszban, inkább az informatikusokhoz fordul.

Kósa Attila
(atkosa@shinwa.hu) informatikus mérnök. Egy japán cégnél dolgozik rendszergazdaként. 1995-ben találkozott először a Linuxszal. Amikor csak teheti, két kisfiával játszik.

Agenda – az első linuxos PDA

A személyi titkárok piacán legalább akkora lépés volt az első teljesen linuxos titkár megjelenése, mint a PC-piacon a Linux felbukkanása. Majd egy héten keresztül dobogó szívvel vártam, hogy a vámosok végre eldöntsék: a kezükben tartott csomag tartalmaz-e valami törvényelleneset, és ha nem, vámkezeljék. A mondás szerint az első találkozás mindig meghatározó, ami ebben az esetben valóban igaznak bizonyult. Az Agendán semmi sem ott van, ahol az ember először keresné: például a bekapcsológombot is az oldalán rejtették el. A többi gombot szintén itt helyezték el – ezek első látásra feleslegesnek tűnhetnek, ám hamar megtapasztalhatjuk, milyen hasznos szolgáltatásokat takarnak. A kezdeti furcsaságokat továbbiak követték: az Agenda felépítése hasonló a Palméhoz, a felső része (hátul) azonban sokkal szélesebb, hiszen ide kerültek a ceruzaelemek. Nagy örömmel fogadtam az elemes tápellátást, még ha azok a jóval drágább AAA-osztályból kerülnek is ki, de a titkár ettől válik igazán hordozhatóvá és bárhol utántölthetővé. A versenytárs termékekhez képest további ügyes újítás, hogy a ceruzát (mutatóeszközt) kihúzva a gép önműködően bekapcsol, a helyére illesztve pedig kikapcsol, így akár egy kézben tartva is könnyen tudjuk használni, például unalmas sorban állás közben. A képernyőt védő fedél három helyzetben is hasznos:

1. A fedél lecsukott állapotában védi a képernyőt a karcolásoktól.
2. A fedél nyitott helyzetében (90 fokban állva) eltakarja a fényt, ezzel is elősegíti a munkánkat. Ha azonban nem megfelelően egyensúlyozunk vele, sajnos visszacsukódik.
3. A fedelet 360 fokban hátrahajtva a titkár hátlapjához foghatjuk, és szinte jegyzettömbként használhatjuk.

A készüléken kétoldalt (pontosan a fogási pontokon) 1-1 SHIFT billentyűt helyeztek el, amelyek nagyon hasznosak a nagybetűk bevitelénél. A képernyő borotvaéles és kalibrálás után pontosan „fog”. Természetesen a háttérvilágítás itt is megtalálható, mindössze a bekapcsoló gombot szükséges hosszan lenyomva tartani.

A kivitelezésen látszik, hogy az Agendát olyan emberek tervezték, akik használták a versenytársak termékeit, és azok hibáit már az első Agenda-sorozatban megpróbálták kiküszöbölni. Ilyen apró, de sok pénzt megtakarító ötlet, hogy a dokkoló szétválasztható, tehát az így kapott soros kábelt külön is használni tudjuk. A soros kábel csatlakozóján szabványos hálózati adapteres csatlakozási lehetőség is van. Az alapcsomagolásban továbbá bőrtok és fülhallgató található mikrofonnal.

Az első bekapcsolásnál szívet melengető látvány végignézni a kis képernyőn a Linux betöltődési folyamatát: elsőként a démonok indulnak el, majd az utolsó lépések egyikeként a beviteli mutatóeszközt kell kalibrálnunk. Ezután a már jól ismert xdm bejelentkező ablakával találjuk magunkat szemben, amit később ki lehet iktatni. Külön felhasználókat és külön arculatokat hozhatunk létre akár egy átlagos otthoni PC-n is. Az alapprogram viszont, amelyet a géphez adnak, leginkább használhatatlan. Egyszerűen képtelenség folyamatos működésre bírni, állandóan fagy, vagy 5–10 percet gondolkodik egy-egy alkalmazás elindítása előtt, után és közben. E ponton jelentkezik a Linux óriási előnye: míg más eszközöknél a használat itt véget is ér, és az eszközt visszaküldjük a gyártónak továbbfejlesztésre, az Agenda esetében a legfrissebb rendszermagot és a legújabb rendszerlemezt egyszerűen letölthetjük az Internetről. A gyári CD-n adott tájékoztatás segítségével akár egy

Előnyök

- Elemmel üzemel – akár tölthető elemeket is használhatunk, amelyeket bármikor cserélhetünk, tehát nem függünk elektromos hálózati csatlakozótól.
- Nagyon jól látható a képernyő.
- A soros kábel és a dokkoló egyben található.
- Alapfelszerelésben adnak hozzá szinte mindent.
- Infrakapuja ismeri a GSM-modemmel rendelkező infrás telefonokat és kapcsolatot tud velük létesíteni (például Nokia 6210-sel).
- Szép és jól súlyozott ceruza (mutatóeszköz).
- A bekapcsolása ötletes.
- Az oldalt és alul elhelyezett nyilak kellemes mozgást tesznek lehetővé az alkalmazásokban.
- Magyarítható (magyar billentyűzet).
- Linuxot futtat és a frissítések nagyon gyakoriak.

Hátrányok

- Az elemek üzemideje csekély, gyakori használat esetén 2–3 nap.
- Az alkalmazások egyelőre nehezen használhatók.
- A soros adatszinkronizáció kidolgozatlan és nehézkes.
- A felhasználóra hárítják a Linux-frissítés terhét, máskülönben néhány alapszolgáltatás sem működik (például az ébresztő nem kapcsolja be a gépet, csak akkor riaszt, ha mi bekapcsoljuk).
- 8 MB RAM van beépítve, amelynek nagy részét a futó rendszermag azonnal elfoglalja.

kezdő Linux-felhasználó is betöltheti az új flasht a gépre. Érdemes figyelni az elemtöltöttségre, mivel a betöltés és a flash megírása körülbelül 15 percet vesz igénybe, ami soros összeköttetésnél nagyon megterheli az elemeket (tehát ajánlatos teljesen feltöltött elemekkel belevágni). A feltöltést követően a titkárt az új rendszermaggal el kell indítanunk, és ismét végre kell hajtanunk a kalibrálási folyamatot. Megéri, hiszen cserébe teljesen új rendszert kapunk, amely jóval használhatóbb lesz és nem olyan lassú, mint az elődje. A fejlesztők természetesen az MS-felhasználókra is gondoltak, és a CD-n az adatátvitelhez szükséges összes program windowsos változata is megtalálható.

Az új programcsomag rengeteg alkalmazást rejt, főleg a játékok nagy száma lepett meg, a 47-es villamoson például remekül szórakoztam az Aliensszel. Az X-munkafelületet igényeinknek megfelelően állíthatjuk, eldöntve, hogy ikonos (Palm-stílus) vagy általános X-felületet kívánunk-e magunk előtt látni. Hasznos a *Lunch Pad* bekapcsolása, ekkor a legfelső sorban az elem töltöttségét és a pillanatnyi időt vagy a rendszer elindítása után eltelt időt látjuk. A képernyő legalsó során elhelyezett érintőgombok eléggé értelmetlenre sikerültek, ugyanis a rájuk rajzolt ábrákból nehéz rájönni, milyen szolgáltatást is rejtenek. A határidőnaplót például két D betű jelképezi. A legnagyobb gondot a rögzített beviteli mező hiánya okozza, itt ugyanis nincs egy mindig készenlétben lévő billentyűzet, illetve graffiti, amin írni vagy rajzolni lehetne. A szolgáltatás eléréséhez meg kell érinteni a billentyűzetábrát, és csak megközelítőleg három másodperc eltelte után érhető el a képernyőn a billentyűzet. Sajnos rosszul felépített, ugyanis elsőre csak igen kevés karakterhez férhetünk hozzá. A számok vagy a különleges karakterek használatához rá kell böknünk az 123 feliratú gombra, és ezután töltődik be a számbillentyűzet, ahonnan a HWR gomb megnyomásával nyílik lehetőség a kézírás-felismerő program elérésére. Ez nagyon komoly nehézség azok számára, akik a Palmnál megszokták, hogy a felület mindig rendelkezésre áll, és a váltás is jóval egyszerűbb.

Az adatszinkronizálás egyelőre még igencsak gyerekcipőben jár. A CD tartalmaz ugyan `rpm` és `tgz` formátumú GUI-t, amit szó szerint fel kell szenvedni a Linuxra, azonban semmi mást nem tesz, mint értékekkel lát el egy `pppd`-t, és létrehoz egy hurokcsatolót (így akár telnetelhetünk az Agendára, bár a fejlesztők az *rsync*-et ajánlják adatátvitelre). Az Agendán levő pppd azonban időnként lefagy, valamint a syslogd felemészti a memóriát. Ilyenkor csak az újraindítás segíthet, ám mire ez a cikk megjelenik, a hibát valószínűleg már egy újabb rendszermagváltozat orvosolja.

Mindent összevetve az Agenda egy nagyon jól összeállított készülék, amelynek óriási szerencséje, hogy Linuxot futtat, hiszen napok alatt javítják a hibákat, és a rajta futtatható programok száma 1–2 hónap alatt megkétszereződik. Nemrég jelent meg rá a keresztfordító, így akármelyik kedvenc alkalmazásunkat is lefordíthatjuk erre a rendszerre.

Egyelőre azonban piacot igazából csak azok számára jelenthet, akik a már létező, például ipari alkalmazásukat nem akarják egy tenyérgépre újraírni, hanem elegendő lefordítaniuk. Ezenkívül a Linux-őrültek is szívesen látják, de sok rendszermagnak kell még átfolynia a soros kábelen, hogy az Agenda legyen az egyetlen társunk a mindennapokban.

Adatok

Processzor: 66 MHz NEC-processzor (32 bites)
Memória: 8 MB RAM + 16 MB flash
Képernyő: 160×240 LCD
Infravörös (Irda) és soros (RS232) kapu

További tájékoztatást a Psion Rendszerház Kft. nyújt.
telefon: 209-3804
e-mail: psion@psion.hu
➲ http://www.psion.hu
➲ http://www.agendacomputing.com

Varga S. Csaba (guska@guska.hu) Az 1.1-es Slackware óta linuxozik. Kedvtelései közé tartozik a fotózás és Linux telepítése PDA-kra. Legszívesebben a Gerecsében túrázik barátaival.

Linux-index

1. A számítógépek ekkora százalékát bántalmazzák a felhasználók: 25
2. A Microsoft készpénztartaléka 2001 májusában (milliárd dollár): 30
3. Havonta ennyi milliárd dollárral nő a Microsoft készpénztartaléka: 1
4. 2001 folyamán (május 30-ig) a Microsoft-részvények ára ennyi százalékkal emelkedett: 62
5. Az egyidejűleg lefolytatható beszélgetések száma Marconi idejében: 1
6. Ugyanez a szám ma: 1 trillió
7. Ennyi évente kétszereződik meg ez a szám: 2,5
8. A föld felülete négyzetkilométerben: 148 940 000
9. Ennyi beszélgetést lehet folytatni négyzetkilométerenként: 6 714
10. A Google ennyi találatot ad az „open source” kifejezésre: 1 930 000
11. A Google ennyi találatot ad a „free software” kifejezésre: 1 150 000
12. A Google ennyi találatot ad az „Eric Raymond” névre: 26 100
13. A Google ennyi találatot ad az „Eric S. Raymond” névre: 30 100
14. A Google ennyi találatot ad a „Richard Stallman” névre: 54 300
15. A Google ennyi találatot ad a „Richard M. Stallman” névre: 11 600
16. A Google ennyi találatot ad a „Copyleft” kifejezésre: 176 000
17. A Google ennyi találatot ad a „GNU/Linux” kifejezésre: 446 000
18. A Google ennyi találatot ad a „Linux” kifejezésre: 26 500 000
19. A Gartner felmérése szerint a 2000-es év harmadik negyedévében szállított kiszolgálók között ennyi százalék használt Linuxot: 9
20. A Linux piaci részesedése a kiszolgálók területén az IDC felmérése alapján: 27
21. Az AllNetSearch felmérése szerint a válaszadók ennyi százaléka használ Linuxot: 39
22. Az IBM Linux bevételének növekedése: 128%
23. A Linux szállításából származó bevételnövekedés 1999–2000: 28%
24. A telepített Linux-kiszolgálók várható száma 2005-re: 21 006 000

Források
1: Wired News
2–4: TIME Magazine
5–7: Martin Cooper, CEO, ArrayComm és a mobiltelefon feltalálója
8: CIA World Factbook
9: matematika
10–18: Google, 2001. június 11.
19: Gartner Group
20: International Data Corp.
21: Internet.com
22: IBM
23–24: International Data Corp.

Ők mondták

Azt feltételezni, hogy a szerző tudja a legjobban, mit szeretne az olvasók mindegyike olvasni: butaság, márpedig ez derül ki az álláspontjából.
(*John Wilcox, Microsoft-alkalmazott a Smart Tags „okos címkék” védelmében*)

Ha a Smart Tags „okos címkék” a másolási jogokat vagy az álnok kereskedelmi törvényeket nem is sértik, a Web sértetlenségét mindenképpen érintik. A Világhálóban részben az a vonzó, hogy bárki számára lehetővé teszi gondolatainak, érzéseinek és véleményének közzétételét cenzúra és a céges érdekek figyelembevétele nélkül. Azáltal, hogy a Microsoft a Smart Tagset hozzáadja a weblapokhoz, az írók és az olvasók közé áll. A Microsoft a következőt mondta *Walter Mossbergnek*: „Ez a lehetőség megkíméli a felhasználókat a hivatkozásokban szegény weblapoktól”.
A Microsoft valójában azt dönti el, hogy a szerzők miként írjanak és a fejlesztők hogyan építsék fel a weblapjaikat.
(*Chris Kaminski*)

Józan nemzetek váltak kétségbeesett szerencsejátékossá és tették kockára még a létezésüket is egy darabka papír(pénz) miatt. Ez a mű a leghíresebb csalódások történetének kutatását tűzi ki céljául. Az emberekre – ahogyan mondani szokás – jellemző a csordaszellem. Tapasztalhatjuk, hogy csoportosan megőrülnek, és csak lassan, egyenként nyerik vissza az értelmüket. (*Charles MacKay, 1841*)

Nem lehet felébreszteni azt, aki úgy tesz, mintha aludna. (*Navajo közmondás*)

Ha a „legjobb gyakorlatok” üzleti fogalmát már a civilizáció hajnalán alkalmazták volna, az emberiség sohasem jutott volna el a civilizáltságig. A művészettörténet foglalkozhatna a rómaiak által kőbevésett mosóporreklámokkal, a sixtusi kápolna mennyezetén található freskó azt hirdetné, hogy „Kölcsönt a Mediciktől!”, és a Szabadság-szobor fáklya helyett egy tubus fogkrémet tartana a kezében.
(*Christopher Locke*)

Az első napi zárás után a pénztárban 24,67 dollár volt. 24 dollárt költöttünk el hirdetésre, 67 centet tartottunk meg másnap reggelre váltópénznek. (*John Wanamaker, a világ első nagyáruházának megnyitása után, Philadelphiában 1861-ben*)

Új termékek

Pervasive.SQL 2000i

A Pervasive Software Inc. kiadta Pervasive.SQL 2000i névre hallgató beágyazott adatbázis-kezelő prog-

ramját, melynek segítségével olyan alkalmazások építhetők, amelyek a Világhálón a felületek között a kód megváltoztatása nélkül vándorolhatnak. A több felületet támogató adatbázismotor a rendszermag 2.2-es és frissebb változataival használható. A 2000i visszamenőleg támogatja a létező API-alapú alkalmazásokat Linux, NetWare és NT alatt. Továbbá független a kiszolgáló, illetve lekérdező változatától, és a beállítások megváltoztatása nélkül képes beépülni. A Pervasive.SQL támogatja az ODBC, OLE DB 2.5 és JDBC 2.0 szabványokat, valamint a Java, a Perl, a PHP és más nyelveket.
Adatok: Pervasive Software, Inc., 12365 Riata Trace Pkwy., Bldg. II, Austin, Texas 78727,
telefon: 800-287-4383,
e-mail: info@pervasive.com,
➲ http://www.pervasive.com/

Sistina GFS 4.1.1

A Sistina GFS fürtözött fájlrendszere lehetővé teszi, hogy SAN-ban elhelyezett kiszolgálók is írhassák, illetve olvashassák a megosztott SAN-eszközön található fájlrendszert. A 4.1.1 változat új tulajdonságai között található: GFS rendszermagfolt Linux 2.4.5-höz, egy új I/O-védelmi módszer (apc_ms hálózati kapcsoló) és frissített indítóparancsfájlok. A GFS 4.1.1-re a GNU GPL felhasználói szerződés vonatkozik. Letölthető a
➲ http://www.sistina.com/gfs/software/index.html címről.
Adatok: Sistina Software, Inc., 1313 Fifth Street SE, Minneapolis, Minnesota 55414,
telefon: 612-638-0507,
➲ http://www.sistina.com/

ClusterWorX 2.0

A ClusterWorX 2.0 a Linux NetworX kezelőprogramja, amely a rendszergazdák számára lehetővé teszi, hogy a géptelepet egyetlen rendszerként vezéreljék. A felhasználók távolról figyelhetik a géptelepet, egészen a csomópontok szintjéig. Testreszabható és bővíthető. A ClusterWorX olyan eseménymotort használ, amelyben sokféle érték beállítható. A program tulajdonságai többek között: a látványos megfigyelés lehetősége, a bővítmények támogatása, a háromrétegű alkalmazás, a lemezmásolás és a lemezképek kezelése, valamint a csomópontok áramellátásának megfigyelése.
Adatok: Linux NetworX, 8689 South 700 West, Sandy, Utah 84070,
telefon: 801-562-1010,
e-mail: info@linuxnetworx.com,
➲ http://www.linuxnetworx.com/

Replicator

Az everStor Inc. bejelentette Replicator nevű kezelő-, másoló- és mentéskezelő programjának a megjelenését. A Replicator központilag intézi bármely kiszolgálón vagy munkaállomáson a megadott lemezek, könyvtárak és fájlok kezelését vállalaton belül, helyi hálózaton (LAN), WAN-on vagy internetkapcsolaton keresztül. Mivel on-line adatelérést biztosít, a Replicator az összeomlás utáni helyrehozatal részeként is használható. A Replicator a kiszolgálón vagy hálózati eszközön helyezkedik el, és nem igényli ügyféloldali program telepítését. A fájlátvitel TCP/IP-kapcsolaton keresztül valósul meg.
Adatok: everStor, Inc., 1240 N. Lakeview Avenue, Suite 150, Anaheim, California 92807,
telefon: 714-970-7511,
e-mail: sales@everstor.com,
➲ http://www.everstor.com/

Javlin

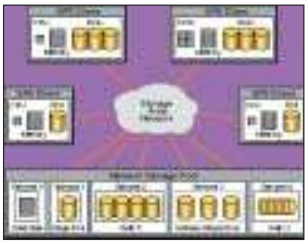
A Javlin, az Object Design középszintű vállalati JavaBean adatgyorstár-kezelője már i386 Linux-környezetben is elérhető. Támogatja a 2.4-es rendszermagot. A Javlin az adatokat elosztott állandó gyorstárak alkalmazásával tárolja és szállítja az alkalmazáskiszolgálókhoz. Lehetővé teszi továbbá, hogy a működő rendszerhez menet közben kategóriákat (categories) és tulajdonságokat (attributes) adjunk. Ráadásul a Javlin Java-dinamikus adatkeretrendszert nyújt.
Adatok: Object Design, 25 Mall Road, Burlington, Massachusetts 01803,
telefon: 800-424-9797,
e-mail: info-objectdesign@objectdesign.com,
➲ http://www.objectdesign.com/

PBS Pro 5.1

Megjelent a Veridian PBS Pro programjának 5.1-es változata. A programmal a géptelepek munkaterhelésének kezelése és a munkafolyamatok sorokba rendezése végezhető el, és minden Unix-változaton alkalmazható. Az 5.1-es változat tulajdonságai többek közt: dinamikus terheléselosztás, rendszereken átívelő ütemezés, a feladatok függetlensége és láncolása, preemptív feladatütemezés, a kötegelt és interaktív munka kezelése, valamint grafikus és parancssoros felület.
Adatok: Veridian Systems, PBS Products Dept., 2672 Bayshore Parkway, Suite 810,
Mountain View, California 94043,
telefon: 877-905-4,
e-mail: sales@pbspro.com,
➲ http://www.pbspro.com/

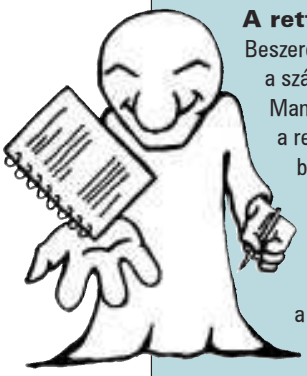
MandrakeSecurity

A MandrakeSoft, a Linux-Mandrake kiadója bejelentette a Single Network Firewall nevű programot, amely az első a MandrakeSecurity márkanév alatt megjelenő termékek közül. A kifejezetten kis- és középméretű vállalatok számára tervezett program a következő tulajdonságokkal rendelkezik: könnyen használható webes felület, biztonságos hálózati kapcsolat, betörésérzékelés és -szűrés. A programra a GNU GPL felhasználói szerződés vonatkozik. Letölthető a
➲ http://www.linux-mandrake.com címről.



Adatok: MandrakeSoft Inc., 2400 N. Lincoln Avenue, Altadena, California 91001,
telefon: 626-296-6290,
➲ http://www.mandrakesoft.com/

A hónap szakmai tanácsai



A rettegett "LI" felirat

Beszereltem egy második IDE-merevlemezt a számítógépembe, és telepítettem a Linux Mandrake 8.0-t. Újraindítás után megjelent a rettegett „LI" felirat, tehát a LILO négy betűjéből csak az első kettő bukkant fel. Úgy vélem, a gond forrása az, hogy a második merevlemezt szolgaként telepítettem a másodlagos IDE-sínre. A rendszerindító lemezrész a (/-ként csatolt) hdd6.

Frederic Mora, fmora@attglobal.net

Ha a LILO csak a „LI" feliratot jeleníti meg, azt jelenti, hogy a rendszerindító folyamat első lépcsője betöltötte a második lépcsőt, de nem tudta futtatni. Ezt vagy a lemez fizikai beállításainak összeférhetetlensége okozza, vagy hogy elmozdítottad a */boot/boot.b*-t a leképezés telepítőjének lefuttatása nélkül. A megoldás: a */etc/lilo.conf* fájlban meg kell adni, hogy a rendszer hol (melyik lemezen és melyik lemezrészen) helyezkedik el. Indítsd a rendszert a vészindító lemezről, lépj be rendszergazdaként, hogy szerkeszthesd a lilo.conf-ot, és minden elemet oda állíts be, ahol fizikailag elhelyezkedik (úgy, ahogy a lemezeidet azonosítja a rendszer: */dev/hdaX, /dev/hdbY, /dev/hddZ* stb.). A /etc/lilo.conf szerkesztése után futtasd le a `/sbin/lilo -v` parancsot, amely mindent a lemezre ír, és indítsd újra a számítógépet.

További részletekért nézd meg a ➲ http://www.fan.nb.ca/~aa126/troubleshoot-LILO.html lapot.

Felipe E. Barousse Boué, fbarousse@piensa.com

A wu-ftp nem engedi be a felhasználókat

A munkahelyemen gondjaim adódnak az FTP-kiszolgálóra történő belépéssel. A kiszolgálóra a `wu-ftpd`-t és az `anonyftp`-t telepítettem. A vendég (guest) azonosítót létrehoztam, de egy felhasználó sem tud belépni az FTP-kiszolgálóra. A jelszó biztosan jó.

Alan Lim, alan_lim@astro.com.my

Nézd meg a */var/log* alatti naplófájlokat, hátha mondanak valamit. Használhatod a `wu-ftpd -d` kapcsolóját, ekkor több mindent naplóz a program. Ha az FTP-démont saját magad fordítottad, ellenőrizd, hogy nem a PAM vagy az árnyékjelszavak használatával van-e gond. Fordítás közben talán nem derült ki, hogy erre is szükség van. Mindenesetre jegyezd meg: nagy biztonsági kockázatot vállalsz. A `wu-ftpd` nem tudja végrehajtani a `chroot()` hívást a közönséges felhasználók bejelentkezésekor, ami azt jelenti, hogy a nem névtelen (nem anonymous) felhasználók belépés után a teljes rendszert elérhetik. Mindkét gondot egyszerre megoldhatod, ha megfontolod egy biztonságosabb FTP-démon telepítését, például a ProFTPD-t vagy az NcFTPD-t, és a `chroot()`-ot beállítod a felhasználó saját könyvtárába.

Chad Robinson, crobinson@rfgonline.com

Alan, azt is ellenőrizd, hogy a felhasználók által használt héj fel van-e sorolva a `wu-ftpd` beállításai között. Ha nem, úgy a felhasználó akkor sem léphet be, ha a jelszó helyes.

Mario Neto, mneto@argo.com.br

Nincs forrásom, de fordítanom kell

Telepítettem vagy megpróbáltam telepíteni majdnem minden elérhető terjesztést. A legjobb telepítő szerintem a Mandrake 7.0, 7.1 és 7.2. A gondom az, hogy nem tudom kiválasztani a forráskód telepítését, amikor a beállításokat végzem. A rendszer vagy hibát ad, vagy fájlok hiányoznak. Elolvastam a readme-fájlokat, több könyvet, de még mindig nem jöttem rá, mi a hiba.

Bill York, bill_york@pipeline.com

A forrásfájlokat (például a Linux-rendszermag forrását) RPM-ből telepítheted. Fűzd be a forrásokat tartalmazó CD-t, lépj be a forrás RPM-ek könyvtárába, és add ki az `rpm -i kernel-source-file.rpm` parancsot. Ennyi az egész. A telepítés az RPM-mel (a Mandrake esetében) meglehetősen egyszerű, a más fájloktól való függőségek a legtöbb esetben szintén figyelembe vannak véve. Ha hibaüzenetet kapsz, valószínűleg más összetevőket is telepítened kell, mielőtt megpróbálnád újratelepíteni a csomagot.

Felipe E. Barousse Boué, fbarousse@piensa.com

2.4.2-pánik!

Megpróbáltam a rendszermagot a 2.4.2-változatra frissíteni. Újraindítás után rendszermagpánik következett:

```
root fs not mounted
cannot open root device "301" of 03:01
Please append a correct root = "boot
                                option"
kernel panic vfs:
unable to mount root file system on 03:01.
```

Ugyanazt az eszközt használtam, mint a régi rendszermagnál: a /dev/hda1-et. Ellenőriztem az `rdev`-vel, és rendben volt. Még azt is megvizsgáltam a `make xconfig`-ban, hogy az ext2 nem modulként a rendszermagba van-e fordítva.

Michael Diaczyk, mdiaczyk@tampabay.rr.com

Ha valamiképpen nem tetted tönkre a felosztási táblát vagy a lemez adatait, el kell tudnod indítani a rendszert úgy, hogy a LILO-nak rendszerindítás közben adsz át paramétereket. A LILO parancssorba próbáld beírni: `linux root=/dev/hda1`

Ha működött, nyisd meg a /etc/fstab fájl és ellenőrizd, hogy a sajátkönyvtár bejegyzése (a /) helyes-e. Arról is győződj meg, hogy a `root=` sor a /etc/lilo.conf-ban a megfelelő lemezrészre mutat-e.

Scott Maxwell, maxwell@ScottMaxwell.org

Szolgáltatás hozzáadása a Kickstartból

A RedHat 7.1 Kickstart fájljának telepítés után lefutó részében meg szeretném adni, hogy bizonyos szolgáltatások, például az `ypserv` és az `autofs` maguktól induljanak el. Hogyan lehetséges ez?

Sowmya, sowms@yahoo.com

A `chkconfig` a megoldás. Add ki a `chkconfig -help` parancsot, és nézd meg a lehetőségeket: például az NFS önműködően elindul a 3-as futási szinten, ha kiadod a `chkconfig --level 3 nfs on` parancsot.
Felipe E. Barousse Boué, fbarousse@piensa.com

Elégedj meg ennyi lemezhellyel!

Lehetséges-e határt szabni egy könyvtár méretének? Webkiszolgálót szeretnék létrehozni, és bizonyos felhasználók legfeljebb 100 MB-ot foglalhatnának el. Hogyan valósítható ez meg?
Jason Sidabras, sidabraj@msoe.edu

Tanulmányozd a Linux `quota` csomagját. Az alkalmazás egyik része a rendszermag részeként fut, ezért a rendszermagba vagy a modulba kell fordítani. A másik rész a felhasználói térben fut, ezzel vezérelhető a működés és felelős a figyelmeztetésekért.
Chad Robinson, crobinson@rfgonline.com

Olvasd el a Quota mini HOGYAN-ját:
➲ http://www.linuxdoc.org/HOWTO/mini/Quota.html.
Marc Merlin, marc_bts@valinux.com

BOOTP-kérések figyelmen kívül hagyása

Nemrég telepítettem egy DHCP-kiszolgálót, és megfigyeltem, hogy egy régi VAX-rendszer, amely a rendszerindítási adatokat az útválasztó túloldalán elhelyezkedő gépről szerzi be, BOOTP-kéréseket küld a kiszolgáló kapujára. Létezik valami, amit kikapcsolhatnék a Linuxon, vagy olyan beállítás, amely figyelmen kívül hagyja ezeket a kéréseket?
Pat Derosa, pderosa@ap.org

Okoznak-e a BOOTP-kérések valamilyen zűrt, vagy egyszerűen csak zavarnak a naplófájlban? Az ISC DHCP démonnak megadhatod a `deny bootp` beállítást, melynek hatására figyelmen kívül hagyja a BOOTP-ügyfeleket, de ettől még a kérés naplózódik a rendszeren. Ebben az esetben nincs túl sok választásod. Együtt kell élned ezekkel az üzenetekkel, hacsak nem érzel magadban elég bátorságot ahhoz, hogy megkeresd az ezért felelős sort a forráskódban, kitöröld, és újrafordítsd a démont.
Chad Robinson, crobinson@rfgonline.com

Ha az `allow-unknown-clients` nincs beállítva, a DHCP-kiszolgáló nem szolgálja ki azokat a gépeket, amelyek nem kifejezetten a MAC-címük szerint vannak megadva. Ha a `dynamic-bootp` beállítást hagyod el, a DHCP-kiszolgáló nem válaszol a BOOTP-kérésekre.
Marc Merlin, marc_bts@valinux.com

Lassú levelezőkiszolgáló

Levélellenőrzéskor azt tapasztaljuk, hogy a kérés feloldása a kiszolgálón hosszú késleltetéssel megy végbe. Néha azonnal sikerül, de legtöbbször túllépi az időkeretet (több mint négy percig tart). A „hosts,DNS” és „DNS,hosts” többféle beállítását is kipróbáltunk, mert feltételeztük, hogy a kérdező IP-címét próbálja feloldani. A behívó kapcsolat felépítésekor IP-címet és DNS-kiszolgálót adunk át.
Az Internet (Squid) gond nélkül működik. A tapasztalatok szerint a levélellenőrzési kapcsolat felépülése után a következő lekérdezés azonnal sikerül.
Kevin, kevin@atom.co.za

Megnézhetitek a levélkiszolgáló gépen, hogy a késleltetést a DNS okozza-e. Használhatjátok a `tcpdump`-ot, és megfigyelhetitek a ki- és bemenő forgalmat. Így kiderül, hogy mi okozza a késleltetést (feltéve, hogy nem a DNS).
Christopher Wingert, cwingert@qualcomm.com

Futtassátok a `tcpdump`-ot, vagy ha van, az `ethereal`-t, és hallgassátok le a kapcsolatot. Ebből megállapíthatjátok, hogy a DNS késleltetése a ti oldalatokon vagy esetleg a levélkiszolgáló oldalán van-e.
Marc Merlin, marc_bts@valinux.com

Írnom kell a Windows-lemezrészekre

Bár mindhárom Windows-lemezrészem be van fűzve Linux alatt, csak rendszergazdaként van rájuk írási jogom. Szeretnék saját magamnak is írási jogot, hogy futtathassam a VMware-t. Megpróbáltam ezeknek a lemezrészeknek a jogosultságait a `chmod` paranccsal rendszergazdaként átállítani, de semmi sem változott.
Bill Freeto, wfreeto@earthlink.net

Használd a `mount uid` és a `gid` beállításait (általában a `quiet`-et is érdemes beállítani). Például egy vfat-lemezrész fstab-bejegyzése így nézhet ki:

```
/dev/sda3       /drv/c
➥vfat user,umask=
➥002,uid=500,gid=500,quiet,low
```

További adatokért fordulj a `mount` súgóoldalához.
Marc Merlin, marc_bts@valinux.com

Ejnye-bejnye, rossz modul!

Lehetséges-e, hogy egy hibás modul úgy tönkretegye a */proc* fájlrendszert, hogy senki sem nézheti meg „oops” okozása nélkül? Ha a modul elfelejti felszabadítani a megszakítást, amikor kikerül a memóriából, a */proc/interrupts* tönkremegy. Ha elfelejti elengedni az I/O-teret, a */proc/ioports* nem működik többé. Ha elgépelt modulnévvel meghívja az `unregister_chrdev`-et (senki nem tenne ilyet, igaz?), a /proc/devices megsemmisül. Amikor ilyen események történnek, megmenthető-e a rendszer, vagy az újraindítás az egyetlen lehetőség?
Bill McConnaughey, mcconnau@biochem.wustl.edu

A hibás modul bármit megtehet – végtére is a rendszermag részeként fut. Ha olyasmivel találkoznék, amiket te írtál le, azonnal újraindítanám a rendszert. Elméletileg lehetséges a hiba kijavítása újraindítás nélkül: olyan modult kell írni és telepíteni, amely helyreállítja az első által okozott károkat. Ez azonban gyakorlatilag lehetetlen, hacsak nem tudod pontosan, hogy mi történt, és még akkor sem feltétlenül egyszerű.
Scott Maxwell, maxwell@ScottMaxwell.org

A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tüköroldalait, a gyakran feltett kérdéseket és az egyéb útmutatásokat a ➲ www.linuxjournal.com honlapon olvashatjátok el. A rovatban közzétett válaszokat Linux-szakértők kis csapata készítette el. További kérdéseiteket szívesen fogadják (angol nyelven) a ➲ www.linuxjournal.com/lj-issues/techsup.html címen, ahol csak egy kérdőívet kell kitöltenetek, de a bts@ssc.com címre levelet is írhattok. A levél tárgyában szerepeljen a „BTS” kulcsszó.

A biztonság

Napjaink egyik legfontosabb kérdését járjuk körül: mit tehetünk munkánk, levelezésünk biztonsága érdekében.

A számítógépek biztonsága növelésének kérdése egészen a kezdetekig nyúlik vissza, amikor a gépek még szobányi méretűek voltak. A fejlesztők elsősorban üzembiztonságra törekedtek, ha valami mégis elromlott, az alkatrészeket kipakolták fehér lepedőkre és megpróbálták megtalálni a hibát. Ebben az időben a gépek állásideje még több volt, mint a hasznos számításokkal töltött idő. Miután az üzembiztonságot nagymértékben sikerült megemelni (több időt dolgozott a gép, mint amennyit állt), az adatok és a folyamatok biztonságát kellett megoldani. Mivel egyre több felhasználót kellett kiszolgálni, a felhasználók biztonsága szintén égető feladattá vált. Lassan elterjedtek a PC-k (Personal Computer – személyi számítógép), amelyek egyszerre csak egy embert szolgáltak ki, és hálózati kapcsolat nélküli különálló egységek voltak. Ezeknél – viszonylag kis méretük következtében – az adatok védelme mellett a biztonság már fizikai biztonságot is jelentett. Nem is olyan régen kezdődött el a PC-k hálózatba kapcsolása, amivel ismét rengeteg, a régi szép nagygépes időkben már felfedezett és megoldott gonddal találhattuk újra szemben magunkat. Rendszerünk, adataink, felhasználóink védelmét úgy kellett megoldani, hogy személyes adataikhoz rajtuk kívül senki se férhessen hozzá, a rendszerben azonban olyan részek is legyenek, amelyeket mindenki használhat. Egy helyi hálózaton, feltéve, hogy a rendszer minden biztonsági követelménynek megfelel (ilyen azért nem nagyon akad, kivétel, ha gépünket kikapcsolt állapotban a pincében befalazva tartjuk!), a felhasználó és jelszava a „leggyengébb láncszem". A felhasználók sokszor nagyon egyszerű jelszavakat használnak, például a kedvesük, gyermekük, aranyhaluk nevét, esetleg a születési dátumokat. Így aztán ne is csodálkozzon senki, hogy a hálózaton egy másik ember is hozzáférhet a legtitkosabb adataihoz, és ez akár az állásába is kerülhet, például mert nem védte megfelelően a cége adatait. Ezekután megjelent az Internet és elszabadult a pokol. Az Internet sok-sok alhálózatból összeálló nagyhálózat. Míg helyi hálózatunkon öt, tíz, esetenként több száz gépet kellett csak kordában tartani, úgy az Internet megjelenésével számtalan gép kapcsolódhatott egymáshoz. Ennek a legnagyobb hátránya az, hogy a különféle rendszerek gyenge pontjait kihasználva szinte bármit megtehetünk, amit nem szégyellünk. Nap mint nap hallani a híreket különféle rendszerfeltörésekről, honlapcserékről, vírusfertőzésekről. A hibákat az emberek elsősorban ott követik el, hogy nem törődnek a biztonsággal, vagy ha törődnek is vele, csak másodlagos szempontként veszik figyelembe. Mennyi kellemetlenségtől kímélhetnénk meg magunkat, ha alkalmaznánk az alapvető megoldásokat!

Csontos Gyula

A Netfilter megszelídítése

Meg kellene erősíteni várkastélyunk árkait? Nézzük meg, hogyan használhatjuk a Netfiltert és az IP Chainst otthoni rendszerünk biztonságának növelésére.

Gratulálok azoknak, akik a mélyvízbe fejest ugorván a 2.2.x (netán 2.0.x) rendszermagról a 2.4.x-re váltottak. Ha mi, akárcsak sokan mások, valamilyen tűzfalat futtatunk az ***IP Chains*** vagy az ***ipfwadm*** segítségével, parancsfájljaink vélhetően egyelőre megfelelőek. Ám előbb vagy utóbb valószínűleg tovább szeretnénk lépni.

A 2.4.x változatszámú rendszermagokba ***Rusty Russell***, a Linux csomagszűrő guruja és programozócsapata beépítette a Netfiltert, ami az IP Chains vagy ipfwadm kiváltására készült. Szerencsére a Netfilter továbbra is lehetővé teszi, hogy IP Chainst vagy `ipfwadm`-ot használjunk, amíg meg nem szokjuk az ***IP Tables*** használatát. Mindezt egy modulba épített rétegen keresztül valósítja meg. A Netfilter azonban olyan sok érdekes újítást tartalmaz, hogy valószínűleg mi is minél előbb a szabályok átalakításán leszünk. Egy dologra viszont oda kell figyelnünk: amennyiben az `ipchains` vagy `ipfwadm` modulokat betöltjük, ezt nem tehetjük meg az `ip_tables`-szel (és vice versa). Szóval mindent vagy semmit. Mindenesetre ennek a cikknek az elolvasása után a változtatások elvégzése magától értetődő lesz.

Azok, akik csak most ismerkednek a csomagszűréssel, nyugodtan hagyják ki az IP Chains-fordításokat, és csak az IP Tables-példákkal ismerkedjenek. Bár nem fogjuk megadni az összes IP Chains parancs és lehetőség IP Tables-fordítását, de írásunk talán jó ötleteket adhat, ha a csomagszűrő tűzfal összeállításához IP Chains parancsokat szeretnénk IP Tables parancsokká alakítani.

Az egyik ok, amiért érdemes Netfilterre váltani, az, hogy (az IP Chainsszel vagy az ipfwadm-mal ellentétben) állapotfüggő (stateful). Hogy ez mit jelent? Azt, hogy képes nyomon követni a kapcsolatokat és engedélyezni a kimenő kérelmekre érkező bejövő válaszokat anélkül, hogy eközben réseket ütne a tűzfalon. A kapcsolat követése itt mindig csak egy bizonyos, ideiglenes lyukat készít a válasz számára, amit kizárólag az adott kiszolgáló használhat. Nemsokára megismerhetjük a működését. A dolog hátulütője, hogy a kapcsolatkövetés ideje alatt a Netfilter egy kicsit több memóriát fogyaszt, mivel a kapcsolatokat a memóriában követi nyomon. Így elképzelhető, hogy 4 MB memóriájú 386-16 gépünk (a szűrési követelmények függvényében) már nem lesz elegendő a feladathoz.

Háttér

A jelenlegi Netfilter-megvalósítás két részre bomlik: a Netfilterként ismert rendszermagrészre, illetve a felhasználói eszköztárra, amely kapcsolatot tart a Netfilterrel, és elkészíti a szabálygyűjteményeket, valamint az IP Tables-adatokat. A csomagszűrő tűzfalhoz mindkettőre szükségünk lesz.

Először összpontosítsunk a rendszermagrészre. A Netfilter támogatja az IPv4 és IPv6 protokollokat, ugyanakkor másféléket nem képes szűrni, ezért tűzfalunk nem futtathat IPX, AppleTalk vagy más efféle protokollt, amit esetleg IP Tables-szabályok kijátszására fel lehetne használni. Továbbá nem szabad engedélyeznünk a gyorskapcsolás (Fast-switching) lehetőséget sem. Ezt a kapcsolót a rendszermag ***Beállítás*** menüjében hálózati lehetőség almenüjének vége felé találjuk. A kód a gyorskapcsolást az IP-verem alacsony szintjén engedélyezi. A Netfilter-kód ennél sokkal magasabb szinten helyezkedik el, így a gyorskapcsolás tulajdonképpen egyszerűen megkerüli őt.

A rendszermag beállítása

Ahhoz, hogy elkezdhessünk dolgozni a Netfilterrel, először is a rendszermagot Netfilter-támogatással le kell fordítanunk. A legtöbb terjesztés alapértelmezés szerint támogatja, de előbb tegyünk egy rövid próbát. Ha az `ip_tables` modult be tudjuk tölteni, ezzel a fejezettel nem kell foglalkoznunk. Rendszergazdaként futtassuk a következő parancsot:

```
modprobe ip_tables
```

Ezután gépeljük be:

```
lsmod | grep ip_tables
```

Ha az `ip_tables` feltűnik, rendben vagyunk. Ha nem, akkor sem kell megijednünk, a rendszermag újrafordítása rendkívül egyszerű. Írásunk ugyan nem fogja ismertetni a teljes újrafordítási folyamatot, de számos forrást találhatunk, ami segíthet bennünket ebben a lépésen. Ha kiderül, hogy rendszermagunkat újra kell fordítanunk, a következő oldalon lévő széljegyzet nyújthat segítséget abban, hogy mit is illesszünk be a rendszermagba.

Netfilter-modulok

Ha az összes modult lefordítottuk és feltelepítettük, valamennyi önműködően be fog töltődni, ha beviszünk valamilyen szabályt, kivéve az `ip_tables`, `ip_nat_ftp` és `ip_conntrack_ftp` csomagokat. Ezeket vagy kézzel vagy IP Tables indító parancsfájlunk részeként tölthetjük be.

A Netfilter teljes fordítása és telepítése rengeteg modult készít, amiből egy átlagos tűzfal azonban csak keveset használ. Azok a modulok, amelyek nem töltődnek be, természetesen memóriát sem fogyasztanak, így nem kell aggódnunk miattuk.

Az IP Tables beszerzése és telepítése

Elképzelhető, hogy terjesztésünk már eleve tartalmazza az IP Tablest, sőt, amennyiben a rendszermag Netfilter-támogatással bír, ez majdhogynem biztosra vehető. Amennyiben a legfrissebb változatot szeretnénk, a Netfilter honlapjáról letölthetjük (➲ http://netfilter.filewatcher.org). Töltsük le és telepítsük az ***INSTALL*** fájl utasításainak megfelelően. A következőkben feltételezzük, hogy a magforrások az ***/usr/src/linux*** könyvtárban találhatók. Ha mégsem, a következő utasításokat a könyvtárnak megfelelően módosítsuk. Amennyiben esetleg a

```
make pending-patches KERNEL_DIR=/usr/src/linux
```

vagy a

```
make patch-o-matic KERNEL_DIR=/usr/src/linux
```

parancsokat le kell futtatnunk, akkor folytatás előtt újra kell fordítanunk a rendszermagot. Egyébként a fenti két parancsot figyelmen kívül is hagyhatjuk. A `patch-o-matic` többnyire

különleges igényű felhasználóknak készült, és nemigen tartozik az átlagos felhasználók érdeklődési körébe.

Futtatás után

```
make KERNEL_DIR=/usr/src/linux
```

majd indítsuk a következő parancsot:

```
make install KERNEL_DIR=/usr/src/linux
```

Most már készen állunk az IP Tables használatára.

Az IP Tables parancssor

Az IP Tables parancssor hat részre bontható. Az első rész az `iptables` parancs, amit a későbbiekben részletezünk; a második a táblameghatározás, a harmadik pedig a láncnév. A negyedik rész a szabálymegadás, amely az IP- vagy ICMP-fejlécek között kereső parancs része, de néhány esetben szabályszám is lehet. Az ötödik rész a cél, végül a hatodik a célérték. Az általános parancssor tehát a következőképpen néz ki:

```
iptables [-t table] -ACDI CHAIN rule-spec
➥ -j TARGET [target option]
```

A fenti sor ugyan nem minden varázslatra igaz, viszont elég általános. A `-L` kapcsolót is igen hasznosnak találhatjuk majd (később még szó lesz róla a cikkben néhány más parancssor-változatról).

Táblák és láncok

A Netfilter három számunkra érdekes táblát tartalmaz. Ezek: a `filter` (szűrő), a `nat` (hálózaticím-átalakítás) és a `mangle` (ferdítő) táblák. Cikkünkben valahányszor csak valamilyen IP Tables-varázslatot mutatunk be, mindig megadjuk a hozzá tartozó táblát is. Egyébként – amennyiben nem adunk meg táblát – a `filter` tábla az alapértelmezett. Éppen ezért, ha nem adunk meg táblát és a szabály hibát jelez, adjunk meg táblameghatározást és próbáljuk meg még egyszer.

Minden táblához egy bizonyos lánc tartozik. A felhasználó által készített láncok mindig egy, és csakis egy táblához tartozhatnak. Látni fogjuk, hogy néhány beépített lánc több mint egy táblához tartozik, de ez csakis a beépített láncokra áll. Egy másik táblából származó láncot nem keverhetünk bele egy felhasználó által készített láncba.

Az alap csomagszűrő tábla a `filter`, amely az `INPUT`, `FORWARD` és `OUTPUT` beépített láncokkal rendelkezik. A `filter` táblába kerülő felhasználói láncokhoz adott szabályok csak olyan célokra vonatkozhatnak, amelyek érvényesek az `INPUT`, `FORWARD` vagy `OUTPUT` láncokban. A `filter` táblán áthaladó csomagok az `INPUT`, `FORWARD` vagy `OUTPUT` láncok valamelyikén (és csakis azon az egyen) haladnak keresztül. Az `INPUT` lánc akkor kerül a képbe, ha a csomag célja a helyi rendszer. A `FORWARD` lánchoz akkor jut el a csomag, ha a helyi rendszeren keresztül egy másik rendszerhez továbbítódik. Végül az `OUTPUT` lánchoz egyedül olyan csomagok kerülhetnek, amelyek a helyi rendszerről származnak és külső célra irányulnak. Minden csomag csak egyetlen lánchoz kerülhet – szemben az IP Chains-megoldással, amely az `input` és `output`, illetve a `forward` láncot is használta, ha egy csomag keresztülment a rendszeren.

Figyeljük meg, hogy az előző bekezdésben az IP Tables láncok nevei nagybetűvel szerepelnek, míg az IP Chains láncnevek kisbetűsek. Szándékosan van így, és az írásmód változását hivatottak jelezni.

A `nat` tábla a hálózati cím átalakításáért felelős. Beépített láncai a `PREROUTING`, a `POSTROUTING` és az `OUTPUT`. Minden lánc egyetlen adott célt engedélyez. A `PREROUTING` a `DNAT` célt fogadja, a maradék láncok pedig az `SNAT` célt. Nemsokára kicsit bővebben is megismerkedünk velük.

A `mangle` (ferdítő) tábla az IP-címen kívül a fejlécben található adat eltorzítására szolgál: megjelölhetjük vele a csomagokat, megváltoztathatjuk a szolgáltatás típusát (TOS), vagy akár az életciklusértéket (time-to-live avagy `ttl`) is.

Szabályok

A szabálymeghatározó rész minden `iptables` parancs szíve. A szabály helyes felépítésével pontosan megadhatjuk, hogy mely csomagokra vonatkozzon. Ez a kiválasztó ismérv annyira általános vagy egyéni lehet, amennyire csak szeretnénk. A legtöbb esetben nem árt, ha egyéni meghatározásaink az általánosabbak előtt következnek.

Ebben a cikkben nem fogom túl sokáig boncolgatni a szabályokat, éppen csak hozzájuk adom a szükséges többszörös lehetőségeket (melyek közül néhánynak saját külön lehetőségei is vannak). Mindenkinek magának kell figyelnie arra, hogy a szabálynak értelme is legyen, például ne adjunk meg kimenő csatolófelületet egy `input` láncban, mivel a szabályhoz itt soha nem fog semmilyen találat tartozni. A parancs szerkezete lehetővé teszi, hogy lehetetlen szabályokat írjunk le, ami azonban semmiképpen sem túl jó ötlet. Ha kétségeink támadnak egy adott szabállyal kapcsolatban, célként adjuk meg a naplócélt, majd keltsünk olyan forgalmat, amely összeillik az adott szabállyal, hogy lássuk, valóban végrehajtódik-e. Ha szabályellenőrző eszközre van szükségünk, vessünk egy pillantást a SendIP-re (➲ http://www.earth.li/projectpurple/progs/sendip.html).

Netfilter engedélyezése a rendszermagban

A Netfilter beállítási menüpont eléréséhez előbb engedélyeznünk kell a *Code maturity-level* menüpontban a fejlesztés alatt álló, illetve befejezetlen meghajtókat (*development and/or incomplete code/drivers*). Ezt a kapcsolót beállítva lépjünk a *Networking menüpontra*, ahol be kell állítanunk a csomagszűrő szolgáltatást (*Network packet filtering (replaces ipchains*). Hacsak nincs néhány gigabájt felesleges helyünk, jobb, ha a *Packet-filtering Debugging* (Network csomagszűrő nyomkövetés) nem kapcsoljuk be.

Most lépjünk az *IP: Netfilter Configuration* (Netfilter-beállításokra), hogy elérjük a Netfilter-modulok almenüit, amelyekben az összes lehetséges pontot kiválaszthatjuk modulként. A modulok a legtöbb esetben csak akkor töltődnek be, ha szükség van rájuk. E sorok írásakor négy kivétel létezik, ezekkel külön foglalkoztunk.

Ha Ipv6-ot szeretnénk használni, ki kell választanunk az IPv6 protokollt, majd be kell lépnünk az *IPv6: Netfilter Configuration* menüpontba. Itt is, akárcsak az előbb, modulként az összes pontot kiválaszthatjuk.

Célok

A Netfilter négy beépített céllal rendelkezik: `ACCEPT` (elfogad), `DROP` (dob), `QUEUE` (sor) és `RETURN` (visszatérés). A `DROP` cél az `ipchains` `DENY` célját váltja fel. Az összes többi cél a célként betölthető modulokon alapul. Ide tartozik a `REJECT` (elutasít), a `LOG` (naplóz), a `MARK` (jelöl), a `MASQUERADE` (álcázás), a `MIRROR` (tükröz), a `REDIRECT` (átirányít) és a `TCPMSS`. A végcélok (terminal), mint például az `ACCEPT`, a `DROP`, a `REJECT`, a `MASQUERADE`, a `MIRROR` és a `REDIRECT`, mindig befejezik a láncot, a `LOG` viszont nem vet neki véget.

A `LOG` ugyanakkor nem is fogadja, utasítja vagy veti el a csomagot, így a lánc értelmezése folytatódhat. Ezért az

`ipchains -l` kapcsoló tulajdonképpen szintén egy cél-, de nem végtípusú. A lánc maradék része is végigfut, mígnem egy házirendszabályt talál.
A házirendszabály a teljes láncra vonatkozó, teljes körű szabály. Ha `FORWARD` láncunk egy `DROP` szabályt tartalmaz, és semmilyen más korábbi szabály nem jelzett találatot a láncban, a csomag ennél a szabálynál fog végezni. A házirendszabály csak valamelyik beépített cél lehet, a `REJECT` szabályt azonban nem adhatjuk meg házirendszabálynak.

Példák

Mielőtt belekezdenénk a példákba, állítsunk be néhány dolgot. A parancsfájlok hasznos dolgok, különösen amelyek az ***rc.local*** könyvtárban futnak (bárhol legyen is a rendszerünkön). Írjunk hát példánk részeként egy ***rc.iptables*** parancsfájlt, hogy már rendszerindításkor kapcsoljuk a Netfiltert (lásd az ***1. listát*** – megjegyzés: minden `iptables` szabály $IPT-vel kezdődik és sortörés nélkül kell folytatódnia a sor végéig, azaz a következő parancsig. Semmilyen szabályt sem szabad a sor közepén megtörni!)
Beállítottunk néhány indításhoz szükséges változót, leállítottuk a forgalom továbbküldését a rendszeren, majd beillesztettünk néhány modult. Az `ip_tables` modul teszi lehetővé a számunkra, hogy szabályokat kezdjünk el írni. Az `ip_nat_ftp` modul akkor szükséges, ha a `NAT` táblát használjuk (a később ismertetésre kerülő) `SNAT` vagy `MASQUERADE` szolgáltatáshoz, ugyanakkor működő FTP-t is szeretnénk. Az `ip_conntrack_ftp` az FTP-kapcsolat követését teszi lehetővé. Ez a modul önműködően betölti az `ip_conntrack` modult, mivel tőle függ. Ha nincs szükségünk az `ip_conntrack_ftp` modulra vagy nem akarjuk betölteni, de azt szeretnénk, hogy a tűzfal IP-töredezettségmentesítést végezzen (ami jó dolog), az `ip_conntrack_ftp`-t az `ip_conntrack`-kal helyettesíthetjük. Nézzük tovább a parancsfájlunkat:

```
for i in filter nat mangle
do
$IPT -t $i -F
$IPT -t $i -X
done
```

A fenti sorok minden szabályt kiürítenek a `-F` értékeként megkapott láncban. Mivel a `-F` értékként semmilyen láncnevet nem kapott meg, az összes láncot ki fogja üríteni. Figyeljük meg, hogy ez a ciklus miatt minden táblán végrehajtódik. Ha nem használunk egy adott táblát, kitörölhetjük a listából. Ahogy az egyes ciklusok végrehajtódnak, megfigyelhetjük, hogy új modulok töltődnek be: először az `iptables_filter` modul, majd az `iptables_nat` és végül az `iptables_mangle`. Ha a `mangle` kulcsszót eltávolítjuk a ciklusból, az `iptables_mangle` modul nem fog betöltődni. A nem használt modulokat tetszés szerint el lehet távolítani. Ip Chains használata esetén ugyanehhez a feladathoz valami ilyesmit kellett volna beírni: `ipchains -F -X`.
Mielőtt folytatnánk, tételezzük fel a következő felállást: van egy otthoni felhasználó és három rendszer, amely az Internetet a mi tűzfalként működő PC-nken keresztül éri el. Az elérés betárcsázás-alapú (ppp0). Ha tényleges beállításunk ettől eltérő, helyettesítsük a megfelelő külső csatolóeszközt a ppp0 helyére. A rendszerek saját hálózatukon kívül semmilyen az eth0-n található szolgáltatást nem ajánlanak fel, ugyanakkor azt szeretnénk, ha az összes belső rendszer képes lenne szörfözni a Világhálón. Ehhez az első példához tételezzük fel, hogy az ISP-től kapott dinamikus IP-címmel rendelkezünk (lásd a 2. listát a 20. CD-n).

Mivel tűzfal-számítógépünk egyben munkaállomás is, saját forgalmát is maga szabályozza. Bár tűzfal esetében nem túl jó ötlet (az ilyen feladatokat többnyire külön rendszerek végzik), egy otthoni hálózatban azért nem igazán van szükségünk külön? tűzfalra. Ezt figyelembe véve emlékezzünk rá, hogy az IP Tables és IP Chains közti egyik különbség az `INPUT`, `FORWARD` és `OUTPUT` láncok eltérő kezelése. Az IP Chains használatakor a `FORWARD`-ra kerülő csomagok az `INPUT`-ról érkeznek, és miután keresztülhaladtak a `FORWARD`-on, az `OUTPUT`-ra utaznak, ezért a szabályainkat az `INPUT` láncban elhelyezve a `FORWARD` lánc csomagjait is máris biztonságban tudhatjuk. Az IP Tables megvalósítás az `INPUT`-ot ellenben csak a helyi rendszerhez használja, a `FORWARD`-ot pedig a többi rendszerhez. A mi esetünkben mind a `FORWARD`, mind az `INPUT` láncokban azonos szabályokra lesz szükségünk. Hogy a szabályokat ne kétszer kelljen leírnunk, készítsünk egy `tcprules` nevű felhasználói láncot és az `INPUT` és a `FORWARD` láncból egyaránt hívjuk meg. Parancsfájlunkat így folytassuk:

```
$IPT -t filter -N tcprules
```

A szabály IP Chains-megfelelője ugyanez lenne, kivéve a `-t` szűrőkapcsolót:

```
ipchains -N tcprules
```

Most lássunk egy kis Netfilter-varázslást. Meg szeretnénk akadályozni, hogy a rendszerünkhöz mások kívülről hozzákapcsolódjanak, de engedélyezni szeretnénk saját felhasználóink kifelé irányuló kapcsolatait. A következő szabályok kihasználják a Netfilter kitűnő képességeit:

```
$IPT -t filter -A tcprules -i ppp+ -m state
➥--state ESTABLISHED,RELATED -j ACCEPT
```

```
$IPT -t filter -A tcprules -i ! ppp+ -m state
➥--state NEW -j ACCEPT
$IPT -t filter -A tcprules -i ppp+ -m state
➥--state NEW,INVALID -j DROP
```

Ha IP Chains alatt szeretnénk valami hasonlót elérni, ahhoz leginkább a syn csomagok elutasításával juthatunk el, a ppp+ csatolófelületen:

```
ipchains -A input -i ppp+ ! -y -j DENY
```

Ezen a ponton érdemes néhány gyors megjegyzést közbeszúrni: a "!" megfordítja (negálja), ami utána következik, így a `! ppp+` ugyanaz, mintha minden más csatolót meghatároztunk volna (a mi otthoni felhasználónk esetében ez a `lo` és az `eth0`). A `"+"` a ppp végén azt jelzi a Netfilternek, hogy az adott szabály az összes ppp-csatolófelületre érvényes.

Az `ESTABLISHED`, `RELATED`, `NEW` és `INVALID` értékek közül az `ESTABLISHED` engedélyezi a forgalom folytatását, ha korábban már mindkét irányban volt forgalom. Az `ESTABLISHED` nyilvánvalóan minden TCP-kapcsolatra értelmezhető, de UDP-kapcsolatra is vonatkozhat, például DNS-lekérdezésekre és traceroutes kérelmekre, vagy akár az ICMP pingre. Az első lépés annak kiderítésére, hogy a kapcsolat létezik-e már a kapcsolatkövetési táblában (/proc/net/ip_conntrack), a csomagok ellenőrzése. Ha igen, a láncok nem futnak le, az eredeti szabály lesz érvényes, így a csomag áthaladhat. Néhány esetben a Netfilter akár gyorsabb is lehet, mint az elődei, hála ennek az ellenőrzésnek. A `RELATED` érték számos dolgot takar: működő FTP-re lehet alkalmazni, amely a megfelelő kapcsolatokat a húszas kapun hozza létre, de a TCP-kapcsolathoz tartozó ICMP-forgalomra is alkalmazható. A `NEW` érték azokra a csomagokra vonatkozik, melyeknek csak a SYN bitjük van beállítva (és ACK bitjük nem működik). Az `INVALID` azokra a csomagokra vonatkozik, amelyek az XMAS fa-keresés (XMAS tree scan) szerint hibás beállításokkal rendelkeznek.

A fenti szabályok lehetővé teszik, hogy a belső rendszerek belsőleg és külsőleg is átjuttathassák az új kapcsolatokat, ugyanakkor ne engedélyezzék a kívülről jövő vagy hibás csomagokat.

Mivel a hálózatunkat álcázás (masquerading) segítségével szeretnénk a tűzfal mögé tenni, be kell állítanunk pár álcázási szabályt. Ez sokban hasonlít az IP Cainsben használt folyamathoz. Feltételezve, hogy belső hálózatunk címe 192.168.0.0/24, a következő szabályt kell használnunk:

```
$IPT -t nat -A POSTROUTING -o ppp+
➥-s 192.168.0.0/24 -d 0/0 -j MASQUERADE
```

Az egyetlen különbség, amit az IP Chainst használóknak feltűnhet, hogy `-o ppp+` utasítást használtunk `-i ppp+` helyett. Ez azért van, mert míg IP Chains alatt a csatolófelületekre a `-i` segítségével hivatkozhatunk, IP Tables alatt a `-i` a bemeneti (input) csatolófelületet jelenti, a `-o` pedig a kimeneti csatolófelület jele. Ha a fenti sorba véletlenül `"-i ppp+"`-t írunk, az álcázás nem fog elindulni, hiszen a szabály soha nem fog semmivel sem egyezni.

Fejezzük be a parancsfájlunkat, és a fenti TCP-szabályokat szükség szerint illesszük be, hogy beindíthassuk a szűrőtábla-rendszabályokat és újraindítsuk az `ip_forwarding` szolgáltatást:

```
$IPT -t filter -A INPUT -j tcprules
$IPT -t filter -A FORWARD -j tcprules
$IPT -t filter -P INPUT DROP
$IPT -t filter -P FORWARD DROP
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Alapértelmezés szerint a szűrőtábla szabályok `ACCEPT`-értékűek. A Netfilter elegáns megoldásai miatt azonban – az IP Chainsszel ellentétben – így is megfelelő biztonságot nyújt. Ha visszagondolunk az eddig alkalmazott szabályokra, láthatjuk, hogy az alapértelmezett rendszabály semmiféle ártalmat nem okoz; sőt, eszerint tulajdonképpen semmi sem érkezhet meg. Néhányan azonban úgy vélekednek, hogy jobb mindinkább a biztonságra törekedni, így például dobhatunk mindent, ami nincs már korábban megcímezve. Figyeljük meg, hogy a rendszabályoknak nincsen céljuk, tehát nem is használhatjuk a `-j` kapcsolót. Ez az, amiért rendszabályok csak beépített célok lehetnek. A rendszabályok nem igazi célok. Ha tényleg `REJECT`-típusú rendszabályt szeretnénk, valami ilyesmit kell beírnunk:

1. lista Az „rc.iptables” parancsfájl

```
#!/bin/sh
# elsıkØnt adjuk meg a megb zhat   svØnyeket
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/
usr/bin:/usr/local/bin

export PATH

# m dos tsuk a k vetkezıket, hogy az
# IP Tables futtathat  ÆllomÆnyra mutassanak:
IPT=/usr/local/sbin/iptables

echo 0 > /proc/sys/net/ipv4/ip_forward

# nØhÆny modul, amit be akarunk t lteni
modprobe ip_tables
modprobe ip_nat_ftp
modprobe ip_conntrack_ftp
```

```
$IPT -t filter -A tcprules -i ppp+  -j REJECT
➥--reject-with icmp-host-unreachable
```

A fenti szabály a ppp csatolófelületen bejövő összes csomagra érvényes lesz. Ügyeljünk rá, hogy utolsóként szerepeljen, különben semmi sem fog átjutni ezen a szabályon.

Bár kiadhattunk akár egy

```
$IPT -f filter -A tcprules -j REJECT
➥--reject-with icmp-host-unreachable
```

szabályt is, azt tapasztaljuk, hogy a saját belső gépeink is blokkolva lesznek, mivel ez a szabály az összes csatolófelületre vonatkozik (ezt a rendszabályok használatakor is figyelembe kell venni).

Ezen a ponton megemlíteném, hogy az IP Chainsszel ellentétben a megfelelő cél megtalálása nem feltétlenül jelenti a lánc végét is. Ha például a `tcprules`-ban a következő szabályt használjuk:

```
$IPT -t filter -I INPUT -p ICMP -m icmp
➥--icmp-type echo-request -m limit
➥--limit 1/minute -j LOG --log-prefix
➥"ICMP-packet "
```

akkor a következő szabály (amely a csomaggal vagy egyezik, vagy nem) szintén végrehajtódik. Amennyiben a szabály nem ejti, utasítja, fogadja el vagy sorolja be (QUEUE) a csomagot és nem visszatérés (RETURN), akkor a lánc folytatódni fog.

Az IP Chains-használók figyelmét felhívnám rá, hogy IP Tables alatt a `LOG` önmaga is cél, és nem egyszerűen csak egy `-l` kapcsoló a cél mögött, mint az IP Chainsben. Ez bizonyos rugalmasságot nyújt, amint az a fenti szabályból is látható. Az egyezés korlátozása megakadályozza, hogy valaki rossz szándékkal

4. lista DNS-bejegyzés a levelezőkiszolgálóhoz

```
$IPT -t filter -N tcprules
$IPT -A tcprules  -i eth1 -m state
➥--state ESTABLISHED,RELATED -j ACCEPT
$IPT -A tcprules -i ! eth1 -m state
➥--state NEW -j ACCEPT
$IPT -A tcprules -i eth1 -p tcp
➥--dport 25 -m state --state NEW -j ACCEPT
$IPT -A tcprules -i eth1 -m state
➥--state NEW,INVALID -j DROP
$IPT -t nat -A PREROUTING -d 209.127.112.17
➥-p tcp --dport 25 -j DNAT
➥--to-destination
➥192.168.0.2:25
$IPT -t nat -A POSTROUTING
➥-o eth1 -s 192.168.0.0/24 -j SNAT
➥--to-source 209.127.112.17
$IPT -A INPUT -j tcprules
$IPT -A FORWARD -j tcprules
echo 1 > /proc/sys/net/ipv4/ip_forward
```

túlterhelje a naplónkat. A `LOG` cél az üzeneteihez előtagot is fűzhet, így a `grep` paranccsal a naplókból könnyen kikereshetők. Minden IP Chains-felhasználó jól tudja, hogy a láncokat a következőképpen lehet megtekinteni:

```
ipchains -L -n
```

Netfilter használatakor azonban a láncokat tábláról táblára kell megnéznünk:

```
iptables -t filter -L -n
iptables -t nat -L -n
iptables -t mangle -L -n
```

Használhatjuk a `-v` kapcsolót is, hogy az egyes szabályokról részletesebb ismeretekhez jussunk:

```
iptables -t filter -L -nv
```

Következzék egy példa, amely az új `SNAT`- és `DNAT`-képességeket mutatja be. Ha valaki érti a `mangle` táblát és az is világos számára, hogy mire való, annak valószínűleg nemigen van szüksége cikkünk elolvasására, és hamarosan elküldi nekem azokat a hibákat, amelyeket észrevett.

Ebben a példában azt fogjuk feltételezni, hogy otthoni felhasználónk széles sávú eléréssel rendelkezik, és az `eth0`-t használja a belső hálózathoz, illetve az `eth1`-et a külsőhöz. A parancsfájl az előzőekhez hasonlóan indul, de amikor a `tcprules` szabályokhoz érünk, egy kicsit mást fogunk alkalmazni. Jelen esetben tételezzük fel, hogy a felhasználó a 209.127.112.17 számú állandó IP-vel rendelkezik, ami egy kicsit megváltoztatja a dolgokat (lásd a 3. listát a 20. CD-n). Azt is elképzelhetjük, hogy a felhasználó saját tartománynévvel rendelkezik, és saját levelezőkiszolgálót futtat a 25-ös kapun, a belső rendszeren található a 192.168.0.2 IP-számon (a tűzfal száma 192.168.0.1). A DNS bejegyzése a 209.127.112.17 címet jelöli meg levelezőkiszolgálóként, ahogyan azt a ***4. listában*** láthatjuk.

Az első két `tcprules` szabály ugyanaz, mint a fenti példában. Mielőtt azonban minden más kapcsolatot elvetnénk, elfogadjuk a 25-ös kapun keresztül érkező csomagokat. Ezután a `nat` táblában elfogjuk a 25-ös kapcsolatot és átirányítjuk egy másik belső gép ugyanilyen számú kapujára. Ezt egyébként az összes kapcsolatunkkal megtehetjük. Ne feledjük viszont, hogyha a DNS-t is ilyen módon irányítjuk át, akkor UDP- és TCP-átirányításra egyaránt szükségünk lesz. Általában – hacsak valaki odakintről nem akar zónaátirányítást a TCP- részeket nyugodtan elvethetjük, és csak az UDP-részt engedjük át.

Azt is figyeljük meg, hogy a `MASQUERADE` használata helyett a kimenő kapcsolatokhoz a `SNAT`-ot használjuk célként. Ha valaki csodálkozna, az `S` a forrást (source) jelenti, ezt fogjuk megváltoztatni. A `DNAT` esetében a csomag célját változtatjuk meg. A `--to-source` érték IP-címtartományok fogadására is képes, így kívülről a tűzfal több gépnek is látszódhat. Ha például az ISP-től öt felhasználható IP-számot kaptunk, a kimenő kapcsolatokhoz mind az ötöt felhasználhatjuk. A különféle szolgáltatásokat külön IP-számokhoz rendelhetjük, így akár öt rendszer is válaszolhat DNS-lekérdezésekre, tárolhat weblapokat, fogadhat leveleket stb. A Netfilter segítségével kezdetleges terheléskiegyenlítést is készíthetünk. Ha céltartományt használunk a `DNAT`-hoz, a legkevesebb kapcsolattal rendelkező rendszer (ami nem feltétlenül jelenti a legkevésbé terhelt rendszert) kapja az új kapcsolatot.

Amit még érdemes tudnunk, hogy elindulhassunk, az, hogy miképpen növelhetjük meg az egyszerre követett kapcsolatok számát. Ez a szám alapértelmezés szerint a rendszeren rendelkezésre álló memória függvénye. A szám azonban megnövelhető. A következőképpen találhatjuk meg:

```
cat /proc/sys/net/ipv4/ip_conntrack_max
```

Az én 256 MB memóriájú rendszeremen ez a szám 16376.

Befejező megjegyzések

A Netfilter használatával minden erőlködés nélkül növelhetjük otthoni rendszerünk biztonságát: a kapcsolatkövetés józan használatával. Számos más lehetőség is rendelkezésünkre áll – ez a cikk épp csak a felszínt kapargatta meg. Mindenkinek ajánlom Rusty (korábban már említett) „Unreliable Guides” című írását, mely a Netfilter honlapon érhető el.

Egyszerű igényekkel rendelkező otthoni felhasználók jobb, ha egyszerű tűzfalat építenek, ezért nem javaslok sok elérhető tűzfaleszközt és parancsfájlt, hiszen mindössze felesleges bonyolultságot visznek a tűzfalunkba. Ha nem értünk meg egy szabályt, ne építsük be. Az első három meghatározó szabály (a `-m state` szabály alkalmazásával) már meglehetősen jó alapot ad. Amennyiben azonban a támadó már korábban bejutott a rendszerbe, a szabályok nem sokat érnek. Nem védenek meg továbbá a levélalapú trójaiaktól sem, megóvnak viszont a közvetlen támadásoktól. Azt javaslom, ha nem használunk IRC-t, naplózzuk és vessük el az IRC-kapcsolatokat:

```
$IPT -j filter -I tcprules -p tcp
➥--destination-port 6667 -j LOG
➥--log-prefix "IRC attempt "
$IPT -j filter -I tcprules 2 -p tcp
➥--destination-port 6667 -j DROP
```

Továbbá ha nincs rá szükségünk, hogy bárki is belépjen a rendszerünkbe, semmilyen kaput se nyissunk meg (ahogyan azt a második példában láthattuk). Cikkünk nem arról szólt, hogyan állítsuk be helyesen a hálózatot az Internet által elérhető rendszerek és a megbízható belső rendszerek elkülönítésére. Ha ilyen összetett rendszerre van szükséged, ideje, hogy egy hozzáértő segítségét kérd!

David A. Bandel
(dbandel@pananix.com) jelenleg Panamában él, Linux- és Unix-tanácsadással foglalkozik. Társszerzője a „Que Special Edition: Using Caldera OpenLinux” című könyvnek.

Az ujjlenyomat általi azonosítás meggátolása

„Ha nem akarunk harcolni, meggátolhatjuk, hogy ellenfelünk harcba szálljon velünk, mégha alig láthatók is a földön táborhelyünk körvonalai. Csak annyit kell tennünk, hogy valami furcsát és érthetetlent vetünk elébe.”

(Szun Cu: A háború művészete)

Az operációs rendszer ujjlenyomat alapján történő azonosítása az a folyamat, melynek során a távoli géptől visszakapott adatok jellemzői alapján meghatározzuk, milyen operációs rendszert futtat a távoli számítógép. Egyes esetekben ez csak annyit jelent, hogy kapcsolódunk a géphez és elolvassuk a szolgáltatás bejelentkező szövegét, máskor viszont bonyolult, mint például a TCP-kezdőadatok és zászlók (flagek) statisztikai elemzése. A kívülállók képesek általános adatokat megállapítani, mint például a gépen futó operációs rendszer típusát, azáltal, hogy megkeresik a TCP-verem operációs rendszerekre egyedileg jellemző megvalósításának a különbségeit. Egyes esetekben ezek a különbségek nagyon részletes adatokat is elárulhatnak, például az operációs rendszer változatszámát és a processzor típusát.

A gép operációs rendszerének pontos azonosítása révén a támadó jól tájékozott és pontos támadást tud intézni a célgép ellen. Ebben az átmeneti tár túlcsordulásaival teli világban a támadónak talán csak arra az alkalomra van szüksége, hogy pontosan megtudja az operációs rendszer és a processzor típusát. Az olyan programok használatával, mint például a linuxos Netfilter, a rendszergazdák ki tudják kerülni az operációs rendszer ujjlenyomata általi pontos azonosítását, néhány esetben pedig még meg is tudják hamisítani a külső erők által gyűjtött adatokat. Bár ezeket a szokásokat semmiképpen sem szabad megbízható biztonsági megoldásnak tekinteni, néha elbizonytalaníthatja, sőt össze is zavarhatja a betörni szándékozót az, ha célja titokzatos hálózati egységnek látszik.

Bár az ujjlenyomat általi azonosítás elkerülése remekül eltitkolja, milyen operációs rendszert futtat valójában a gép, semmiképpen sem teszi biztonságossá különféle sebezhető pontjain. A biztonsági eljárások és irányelvek célja az, hogy magasabbra emeljék a rendszer tönkretételéhez szükséges szakértelem szintjét –, az eltitkolás csak a valódi célpont elrejtését kísérli meg. Mégha a külvilág úgy is látja, hogy rendszereden a Microsoft IIS5 fut, nem véd meg téged, ha mondjuk a Sendmail egyik sebezhető változatát használod, és egy betörőpalánta (script kiddie) önműködő pásztázója kísérel meg betörni hozzád. Az ujjlenyomat általi felismerés elkerülésének célja a támadások elterelése, nem pedig a megakadályozása.

A felfedezés módszerei

Mielőtt egy lehetséges támadót azzal próbálnánk meg eltántorítani, hogy becsapjuk az operációs rendszerünket illetően, meg kell ismerkednünk az operációs rendszerek ujjlenyomat általi azonosítására használt eszközökkel és eljárásokkal.

A „támadó” kifejezést itt bő értelemben használjuk, és magában foglalja mindazokat, akik megpróbálják azonosítani egy gép ujjlenyomatát, és azokat is, akiknek szándékában állhat, hogy kárt okozzanak egy rendszerben. A biztonság a lépések és az azt követő ellenlépések egymást követő sorozata volt, és az író felfogása szerint mindig is az lesz. Ha megismerkedünk az ilyen támadásokra használt eszközökkel és eljárásokkal, akkor nem csupán a jelenlegi fogásokra tudunk felkészülni és terveket készíteni, de bepillantást nyerünk abba is, mit hozhat a jövő.

Számos nyilvánosan elérhető eszköz van, melynek célja az operációs rendszerek ujjlenyomat általi azonosítása. Úgy tűnik, ezek közül az eszközök közül a legnépszerűbb az nmap (➲ http://www.insecure.org/nmap/index.html), melyet ***Fjodor***, az Insecure.org webhely fenntartója készített. Az Nmap számos módszert használ, hogy megpróbálja felismerni a gép operációs rendszerét hálózati szintről, ezek között vannak kezdetleges megközelítésűek és bonyolultabbak is, melyekhez a TCP/IP-protokoll és a protokollok általános jellemzőinek alapos ismeretére van szükség. Íme, néhány figyelemre méltó módszer az Nmap használta módszerek közül:

- ***FIN-próba*** – ha a támadó gép egy nyitott kapujára olyan csomagot küld, amelyben csupán a FIN-zászló van beállítva, akkor bizonyos, a kérésre válaszoló gépekről adatokat szerezhet be. Ez a viselkedés nincs összhangban az RFC-kkel, ezért nagyon jól jelzi az operációs rendszer fajtáját.
- ***TCP ISN mintavétel*** – A TCP-csomagok ISN- (kezdősorszám) mintavételezése értékes módszer lehet a távoli gépek azonosítására és besorolására. Azáltal, hogy a támadó bizonyos mintákat keres az ISN-ekben, a támadó megalapozott találgatásba bocsátkozhat a gép operációs rendszerét illetően.
- ***ICMP-hibaüzenetek küldése*** – az ICMP- (internetvezérlő-üzenetprotokoll) hibaüzeneteket használva a támadó hasznos adatokra tehet szert a gép válaszai alapján. Különös érdeklődésre számot tartó területek az ellenőrző összegek, a hibaüzenet visszhangjának épsége, és a válaszüzenetekben található TOS- (szolgáltatástípus) mezők.
- ***TCP beállítások*** – minden TCP-veremnek talán a legárulkodóbb jellegzetessége, hogyan kezeli a választható TCP-zászlókat és adatokat. A géphez intézett valóságos kérések és a változó ablak- és szakasz- (segment) méret segítségével meghatározható, milyen operációs rendszert futtat a számítógép, annak alapján, hogy hajlandó-e elfogadni ezeket a választható változókat, vagy válaszol-e rájuk.

Bár az operációs rendszer ujjlenyomat alapján való azonosításának mindezen módszerei csomagszinten működnek, nagy gondot kell fordítani rá, hogy megértsük gépünket a szolgáltatások szintjén. Lehet, hogy a támadó rendszerezi és összehasonlítja a csomagok felépítését, de gyakran csak annyit tesz, hogy lekéri a webkiszolgálótól a HTTP-válaszfejléc ***Server*** (kiszolgáló) mezőjében szereplő értéket. Ha tudjuk, mely szolgáltatások azonosítják magukat készségesen, és ami még fontosabb, ha ismerjük az operációs rendszer felépítését, az további lehetőségeket is megmutat nekünk, amelyek révén távolról adatokat lehet szerezni.

Az ügyfélprogramok csendessége (vagy ennek hiánya) is

remek módszer, hogyan szerezhetünk adatokat egy számítógépről. A többi lehetőségtől eltérően ez a folyamat teljesen passzív is lehet. Annak megfigyelésével, hogyan mutatkozik be egy ügyfélalkalmazás valamely szolgáltatásnak, ésszerű találgatásra vállalkozhatunk az operációs rendszerre és a felépítésre vonatkozóan egyaránt. Az ügyfelek közül többnyire a webböngészők, a levelezőprogramok és az IRC-ügyfelek a leginkább vétkesek. Ha IRC-zünk, és lekérdezzük az egyik felhasználó CTCP-változatát, válaszként pedig a ***mIRC32 v5.81 K.Mardam-Bey***, szöveget kapjuk, akkor alapos okkal gondolhatjuk azt, hogy a számítógépen a Windows operációs rendszer valamelyik változata fut.


1.

Végezetül ott van a gyenge pontok kipróbálása. Bár nem túl tapintatos, azért hasznosnak bizonyulhat a számítógép operációs rendszerének megállapítására. Az operációs rendszerre egyedileg jellemző szolgáltatásmegtagadási (denial-of-service) támadások sorozatának a megindítása által a kívülálló próbálkozhat, és megállapíthatja, sebezhető-e a számítógép. Ezzel meghatározható, milyen besorolású rendszert futtat a gép, általában még a foltok szintjén is. A Windows-közösség hálás lehet Fjodornak és az ujjlenyomat általi azonosítást végző eszközök többi fejlesztőjének, amiért úgy döntöttek, hogy ezt a módszert nem foglalják bele szokásos pásztázási eljárásaik választékába.

Mire jó az ujjlenyomat alapján történő felismerés meggátolása?

Ha eddig elolvastad a cikket, egész biztosan érdekel, miért vállalná magára valaki a bonyodalmakat azért, hogy megakadályozza az operációs rendszer ujjlenyomata általi azonosítást. Jó kérdés! Úgy gondolom, az indítékok személyenként eltérőek. Mindenkinek megvan az oka, amiért el akarja rejteni, vagy éppenséggel nem akarja elrejteni az operációs rendszerét. Vannak, akiknél a fokozott titokzatosság a homályban való meghúzódás és a belső kényelem érzését váltja ki. Hasonlóan azokhoz, akik Telnet bejelentkező képernyőjükről eltávolítják a változatszámot tartalmazó üdvözlőszöveget, de a távoli helyek biztonságos elérésére a Telnetet használják az SSH helyett titokzatos, de szakmai szempontból kevésbé biztonságos. Mások esetében az operációs rendszer eltitkolása lehetővé teszi, hogy minden részletre kiterjedően beállítsák behatolásjelző-rendszerüket (IDS), ugyanis nemcsak azt tudják viszonylag jól, mi jöhet be a hálózatukba, hanem azt is, milyen adatok mehetnek ki belőle (titokzatos, óvatos és remélhetőleg biztonságos). Néhányaknak talán még szükségük is van a biztonságosságra, ugyanis minden hálózat, melyre rácsatlakoznak, ellenséges hálózatnak bizonyulhat; és minél jobban elrejtik az operációs rendszerüket, annál szélesebb lehetőségük nyílik rá, hogy elvégezzék mindenkori feladatukat, anélkül, hogy észrevennék őket (titokzatos, óvatos, biztonságos, és valószínűleg épp a leveleidet olvassa). Végül ott vagyunk mi, akik egyszerűen szórakozásból tesszük ezt, azért, mert képesek vagyunk rá; és mert nem kis örömet találunk abban, hogy be tudjuk csapni a körülöttünk levő ismeretleneket, akik állandóan pásztázókat irányítanak ránk (igen, bűnösek vagyunk a vád tárgyában).

A meggátolás módja

Itt az idő, hogy szerencsét próbáljunk az ujjlenyomat általi azonosítás elkerülésében. Ismerve a számítógépek operációs rendszerének megállapítására használt népszerű eljárásokat, ezeket az elképzeléseket vissza tudjuk fejteni, ami segíteni fog, hogy eltitkoljuk valós egyéniségünket.

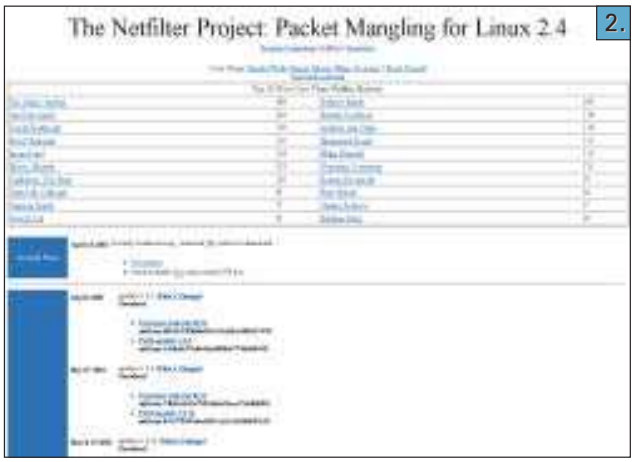
Először is meg kell bizonyosodnunk róla, hogy minden biztonsági folt a helyén van és a rendszer biztonságos. Mint korábban kijelentettem, a titokzatosságra csak azután törekedhetünk, miután gondoskodtunk a biztonságról. Biztosan akadnak, akik nem értenek ezzel egyet, és csupán a titokzatosságra támaszkodnak ahhoz, hogy biztonságossá tegyék rendszerüket, de vajon mire jó a titokzatosság, ha a 33. számú betörőpalánta önműködő programja rendszergazdai jogosultságot szerez a gépeden még ma este? Fogadni merek, hogyha egyszer sikerült rendszergazdai hozzáférést szereznie, akkor nem azzal fog foglalkozni, hogy kitalálja, a Linux melyik változatát futtatod.

Másodszor, meg kell figyelnünk a szolgáltatásainkat. Összhangban vannak azzal az operációs rendszerrel, melynek a használóiként szeretnénk feltüntetni önmagunkat? A legtöbb esetben ez nem okoz nagy gondot, hiszen a Unix-környezetek többsége hasonló vagy ugyanolyan szolgáltatásokat alkalmaz. De ha Windows-gépként vagy éppenséggel Cisco-útválasztóként szeretnéd feltüntetni magad, akkor nem biztos, hogy előnyös, ha látszik, hogy fut az IRC-d. Tegyünk erőfeszítést azért, hogy a szolgáltatásaink összhangban legyenek az álcaként használt számítógéppel.

Ha már a szolgáltatásoknál tartunk, az is jó ötlet, ha a szolgáltatások forráskódját átfésüljük, és megkeressük üdvözlőszövegeit és általános azonosítóit. Néhány árulkodó azonosító jel lehet az ASP-oldalak támogatása vagy az olyan webtartalom, melyet tömörített gzip-formátumban szolgáltatunk. Ismétlem, neked kell megszabnod, milyen szintű rejtőzködést és az álcaként használt géppel való egyezést próbálsz meg elérni.

Ezután azt kell megvizsgálnunk, hogyan jelenik meg gépünk a hálózaton, szembeállítva azzal, ahogyan az álcaként használt gép megjelenik a hálózaton. Ennek megkönnyítésére azt javaslom, tanulmányozzuk a már leírt anyagokat, vagyis azokat a friss ujjlenyomat-állományokat, amelyeket maguk a programok használnak. Időt kell szakítanunk nemcsak annak megállapítására, hogyan válaszol az álcaként használt gép a szokásos kérésekre, hanem arra is, mely egyedi zászlókat támogatja a TCP-ben. A TCP-zászlók hasznos adatokat szolgáltatnak a kívülállók számára, hogy megállapíthassák, milyen operációs rendszert futtatunk. Az ujjlenyomat-állományok nem tartal-

mazzák a gép által adható összes lehetséges választ, hanem csak olyan egyszerű módszereket, melyek megismételhetően működnek. Attól függően, milyen fokú rejtőzködést szeretnénk elérni, érdemes lehet megvizsgálni azokat az ujjlenyomatadatokat is, melyeket az Nmap nem használ (OSPF, OOB, IPv6 stb.). De az is előfordulhat, hogy az alaposság öröménél nagyobb súllyal esnek latba az ezeknek az adatoknak az összegyűjtéséhez szükséges munkával töltött (alvás nélküli, nem álmatlan) éjszakák.



2.

Végül hoznunk kell egy döntést. Cselesek vagy pedig megszállottak leszünk? Ha az utóbbi a választunk, akkor valószínűleg az ügyfélalkalmazásaink meghamisításával folytatjuk. Mint korábban említettem, a gép ügyfélprogramjai hajlamosak mindenféle adatokat kiszolgáltatni (közvetlenül vagy közvetve) a rendszerről. Előző példánkban egy IRC-ügyfél elárulja magáról, hogy Win32-höz készült, de vannak a számítógépek azonosításának kifinomultabb módszerei is, így például, amikor a kimenő levelek fejlécét olvassák el. Megismétlem, minden attól függ, hány alvás nélküli éjszakát vagy hajlandó ráfordítani, hogy rendszered megfeleljen az általad kívánt jellemzőknek.

A lehetséges gondok

Az operációs rendszer ujjlenyomat általi azonosításának elkerülése ugyanolyan, mint a biztonság bármely más nézőpontja: tervezésre, megfelelő megvalósításra, és mindenekfelett megértésre van szükség. Ha a biztonsági irányelveket nem megfelelően ültetik át a gyakorlatba, a rendszer még sebezhetőbbé válhat, mintha egyáltalán nem alkalmaznák őket.
A népszerűség elismeréshez vezet. A számítógépes programok legtöbbjénél az ismertség nagyszerű dolog; a figyelmet ráirányítja kemény munkádra és eltökéltségedre. Az operációs rendszerek ujjlenyomat alapján történő felismerése esetén munkád elismerése ellened dolgozik. Ha népszerű eszközt vagy csomagot használsz, végül fel fogják fedezni sebezhető pontjait és különlegességeit – ez elkerülhetetlen. Ugyanezek a programra jellemző azonosítók képessé tesznek majd másokat, hogy az ujjlenyomat alapján pontosabban azonosítsák ellenlépéseidet, mint magát az operációs rendszert. Csaknem minden operációs rendszer igyekszik véletlenszerűvé tenni TCP ISN-adatsorozatát, így próbálva meggátolni a TCP-eltérítést és a rendszer ellen intézett bonyolultabb támadásokat. Ha az általad választott eltérítési eljárás megpróbálja módosítani a TCP-kezdőadat számait, nagy gondot kell fordítani arra, nehogy alulbecsüljük, illetve leértékeljük ezt a szolgáltatást, és kiszolgáltassuk gépünket az ilyen jellegű támadásoknak. Mint minden a rendszerbe bekerülő számítógépes programcsomag esetén itt is elsődleges fontosságúnak tekintendő az alkalmazás biztonságossága. Az elrejtőzési folyamat része a meglevő szolgáltatások álcázása; a másik része az olyan kód, melynek a forgalom szűrése és a hálózaton folyó forgalmunk álcázása a feladat. Nagy gondot kell fordítani arra, hogy az erre a feladatra használt alkalmazás a jó programozási eljárásoknak és a szigorú ellenőrzésnek köszönhetően megfelelően biztonságos legyen. Csak egyetlen rosszul átgondolt `strcpy()` hívásra van szükség, hogy ebből az előnyből veszélyforrás legyen.
Az egyik korábban említett rejtőzködési mesterkedés az, hogy megváltoztatjuk a program önmagát azonosító üdvözlőszövegét. Elővigyázatosságra van szükség, mert néhány kiegészítő programcsomag ezeket az üdvözlőszövegeket használja, hogy megállapítsa, együttműködik-e a jelenlegi rendszerprogrammal.

Kockázat és előnyök

Most, hogy leszögeztük, a rejtőzködés nem jelent biztonságot, meg kell vizsgálnunk egy másik szempontból is ezt a folyamatot, mégpedig a hatásfokéból. Mivel minden jó rejtőzködési beállítás nagy mennyiségben szűri a forgalmat, igen valószínű, hogy a rendszer teljesítménye megsínyli ezt. Ha a kiszolgálód 10 000 ügyfél weblapjának ad otthont, nagyobb jelentősége van a teljesítménynek, mintha csak fel van állítva egy linuxos géped, melyen a barátaiddal nézegeted a leveleidet és IRC-zel. Rendszergazdaként el kell döntened, melyik a nagyobb előny neked (és felhasználóidnak): a teljesítmény vagy a titoktartás.

Az elmélet bizonyítéka

Azért, hogy szemléltessem, mennyire használható és viszonylag egyszerű az ujjlenyomat általi azonosítás, mellékeltem egy Linuxon, kicsi felhasználói területen futó mintaalkalmazást (OSFPE), mely a Netfilter rendszermagmodult használja (lásd az 1. listát a 20. CD-n). Az operációs rendszer ujjlenyomat általi azonosításának elkerülése egyre hatékonyabb lesz az olyan programok használatával, mint például a Linux alatti Netfilter (lásd még előző írásunkat). Egymás után bukkannak fel a hasonló módosítások és alkalmazások; BSD-ben ezt a feladatot az IP Filterrel és viszonylag kis mennyiségű kóddal lehet megoldani. A Windowst használók (akik messze a legnagyobb hátrányban vannak ezen a területen) fokozatosan fedezik fel a módszereket, melyekkel beburkolhatják TCP/IP-rendszerű adatforgalmukat, és a win32 alatti Libpcap megszületésével elfoghatják és meghamisíthatják a csomagokban küldött saját válaszukat.

A Netfilterről röviden

A Netfilter, ahogy szerzője mondja, „keretprogram a csomagok keveréséhez”. Érdekesen hangzik, nemde? A Netfilter összekapcsolódik a Linux rendszermaggal (pontosabban a 2.4.x és afölötti változatokkal), és mindegyik protokollhoz kezelőket jegyezhet be. Ha a megfelelő szabályok helyükön vannak, ezek a kezelők elfogják a meghatározott szabályoknak megfelelő bejövő és kimenő hálózati forgalmat. A csomagok ezután feldolgozásra kerülnek és NF_DROP jelzést kapnak, ha el kell ejteni, NF_ACCEPT jelzést pedig, ha a csomagot rendesen kell kezelni a veremben, végül NF_QUEUE jelzést kapnak, ha a csomagot sorba kell állítani, hogy fel lehessen dolgozni a felhasználói területen. Ha a csomagot a felhasználói területen való feldolgozásra sorba állítják, az `ip_queue` beállítja a sorba; ezután bármely, a felhasználói területen futó program

aszinkron módon feldolgozhatja a csomagokat, amely bejegyezte magát az ilyen fajta csomagokra. Amikor ezek az alkalmazások kivesznek egy csomagot a sorból, képesek módosítani, elfogadni vagy elutasítani azt. Ha a csomagot elfogadják, továbbadják a következő olyan alkalmazásnak, amely bejegyezte magát az ilyen csomagokra. Ha a csomag NF_DROP jelzést kap, akkor elejtésre kerül, és a szóban forgó csomag feldolgozása abbamarad. A Netfiltert használva a felhasználói területen futó alkalmazások gyakorlatilag magszinten ellenőrizhetik a hálózati forgalmat.

IP Tables

Az IP Tables olyan alkalmazás, mely a Netfilterhez kapcsolódik, és általa beállíthatók, megtekinthetők és eltávolíthatók a rendszer szűrésére használt szabályok. Azért említem meg itt az IP Tablest, mert az elmélet bizonyítására szolgáló kódunk fejlesztésekor úgy véltük, jobb, ha a felhasználókat a szabályok beállítására az IP Tables program használatával ismertetjük meg, mintha magával az alkalmazással dolgoztatnánk fel a szabályokat. Így az embereknek módjuk nyílik jobban megismerni, mi történik a sorba állított csomaggal.

A mi módszerünk

A Netfilter modulok és az IP Tables szabályfeldolgozó program használatával be tudtuk állítani a szabályokat, melyek elfogták a bejövő UDP-, TCP- és ICMP-csomagokat. A bejövő csomagtól és a küldő géptől függően vagy megengedjük, hogy szokványos módon férjenek hozzá rendszerünkhöz, vagy pedig Windows-gép látszatát keltő válaszokat ötlünk ki az Nmap operációs rendszer ujjlenyomat általi azonosításra használt bejegyzései alapján. Íme ez az ujjlenyomat, amelyet megpróbálunk lemásolni, és egy rövid leírás arról, hogyan értük el a célunkat:

```
TSeq(Class=TD|RI%gcd=1|2|3|4|5|8|A|14|1E|28|
➥5A%SI=<1F4)
T1(DF=Y%W=2017|16D0|860|869F%ACK=S++%Flags
➥=AS%Ops=M|MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=0%ACK=O%Flags=AR%Ops=)
T4(DF=N%W=0%ACK=S++|O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=S++|O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E
➥%UCK=E%ULEN=134%DAT=E)
```

Az első sorból kiderül, hogy az időponttól függő (TD) TCP-műveletsorozatot kell felépítenünk, vagy pedig egy olyat, mely véletlenszerűen növekszik (RI), és az egyes lépések egyenlők, de nem nagyobbak, mint 0x1F4 (500). Ezt tulajdonképpen elég egyszerű volt elérni, vagy jobban mondva lemásolni. Először is elfogtuk a bejövő csomagot, vettük a TCP-sorozat számát, és létrehoztunk egy véletlenszerűnek tűnő számot 1 és 500 között. Ez megfelelt az ujjlenyomattal szemben támasztott mindkét követelménynek, a véletlenszerű növekedésre és a legnagyobb közös osztóra vonatkozónak is.
Ezután felbontottuk a csomagokra vonatkozó próbákat (T1-T7) és mindegyik esetre létrehoztunk nekik megfelelő eseteket a TCP-kezelőnkben. Ezek mind nagyon egyértelműek voltak, és egyszerűen szükségessé teszik, hogy a gép adott módon válaszoljon a különféle nyitott és zárt kapukra érkező kérésekre. A részletes tesztek és a beállítások alaposabb feldolgozása megtalálható Fjodornak az operációs rendszerek távoli azonosításáról írt tanulmányában.
Ezután megalkottuk az UDP „nem elérhető kapu”-típusú UDP-kérésre adandó válaszunkat. Az Nmap ekkor annyit tesz, hogy a gép egyik zárt kapujára küld egy UDP-csomagot és várja az ICMP „nem elérhető kapu”-csomag formájában érkező választ. A „nem elérhető kapu” ICMP-csomagok egyszerűen annyit mondanak a küldő gépnek, hogy az UDP-csomagot nem sikerült eljuttatni arra a kapura, amelyre szerették volna, mert azon nincs működő UDP-szolgáltatás. Bizonyos hálózatokon ezeket az üzeneteket sohasem küldik vissza, mert az útválasztó eldobja őket. Annak érdekében, hogy alkalmazkodjunk az ujjlenyomathoz, igyekeztünk visszaküldeni azt, amire számítottak.
Végül, gépünk bizonyos kapuiról kis ráadásként Syn-Ack csomagokat küldtünk vissza arra az esetre, ha véletlenül pásztázással megvizsgálnák, nyitva vannak-e ezek a TCP-kapuk. Ezenkívül nem küldtünk vissza semmilyen választ bizonyos UDP-kapuknál, melyeknél azt a látszatot szerettük volna kelteni, hogy nyitva vannak a gépünkön (mint korábban megállapítottuk, csak a zárt UDP-kapuk küldenek vissza „nem elérhető kapu” üzenetet). Amikor gépünk pásztázása véget ér, úgy kell látszódnia, mintha a 135. és 139. TCP-kapu, valamint a 135., 137., és 138. UDP-kapu nyitva lenne. Ha gépünket az ujjlenyomata alapján megpróbálják azonosítani, bizonyosan megfelelünk a fenti ujjlenyomatnak és „Windows NT4, Windows 95, Windows 98” gépnek látszunk.
Végül az elmélet bizonyítására használt kód pontosan ezt jelenti: rövid programrészlet, melynek célja, egy elképzelés bizonyítása. Kíméljük meg magunkat a gondoktól, és ne futtassuk létfontosságú eszközökön. Próbáljuk ki, tanuljunk belőle, módosítsuk, hasznosítsuk, de ne támaszkodjunk rá! Igyekeztem biztonságosan és jól olvashatóan (ez vitatható) megalkotni a kódot, de semmit nem ígérhetek.

Köszönetnyilvánítás

Hálás köszönetem ***Rex Warren***-nek a kemény munkájáért, és amiért ennek a tanulmánynak és a mintakódnak az elkészítésében segített, Fjodornak azért, hogy megengedte nehéz munkája gyümölcsének felhasználását, és hogy ilyen nagyszerű biztonsági eszközt készített. Köszönet ***Dan Kurc***-nak kódom átolvasásáért és azért, hogy szüleményemet csúnyának nevezte (hé, ez az első C-ben készült programom), és ***Sir Dystic***-nek a C nyelv használatához nyújtott támogatását, valamint hála ***Courtnee***-nek.

Rob Beck jelenleg biztonsági rendszerek tervezőjeként dolgozik az @stake Security Consulting szolgálatnál, szakterülete a Windows operációs rendszerekbe, illetve alkalmazásokba való behatolás felmérése, és a biztonságos felépítés megtervezése.

Kapcsolódó címek

Fjodor tanulmánya az operációs rendszerek távoli felismeréséről ➲ http://www.insecure.org/nmap/nmap-fingerprinting-article.html (*1. kép*) Rendkívül tanulságosnak találtam ezt a tanulmányt az operációs rendszerek ujjlenyomat általi azonosítása témakörében. Netfilter magmodulok ➲ http://netfilter.samba.org (*2. kép*)

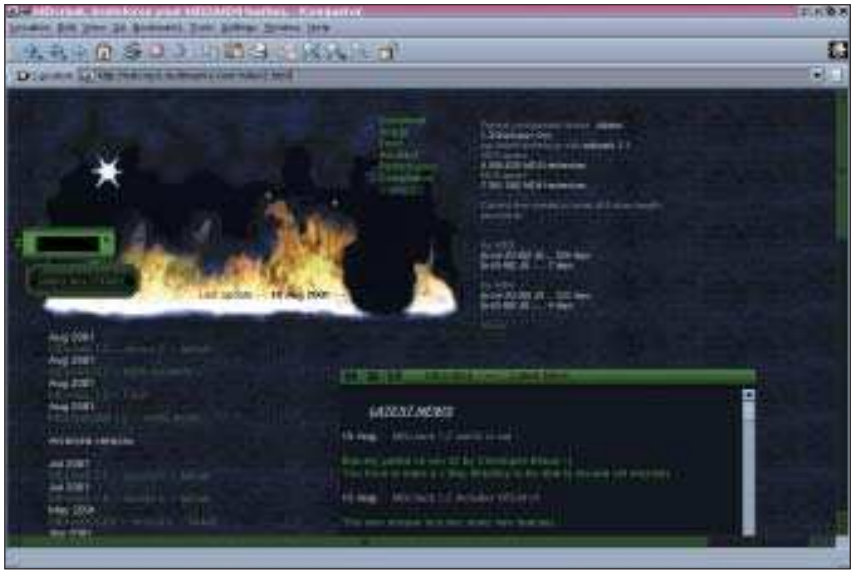
A jelszavas védelem fejlődése

Túlságosan kiszolgáltatott lenne a rendszerünk? Bemutatjuk hogyan hozhatjuk ki a legtöbbet az olyan Unix-alapú jelszavas védelmi rendszerekből, mint az MD5 és a PAM.

Ha látni szeretnéd, hogyan működik az egyedfejlődés, csak vess egy pillantást a GNU/Linux jelszavas rendszereinek fejlődésére! Bár az alapállapotban minden Unix-rendszerben meglévő jelszavas védelem már csak csökevényes szerve a rendszernek, a számítógépes betörők alakjában fellépő természetes kiválasztódás nyomására olyan fejlettebb védelmi rendszereket indítottak el a fejlődés útján, mint az árnyékjelszavak (shadow password), az MD5- és a PAM-módszer. Azóta a jelszavas védelem olyan új területeken is meghonosodott, mint például az indításkezelők, távoli bejelentkezések és egyéb fejlett biztonsági rendszerek. Nap mint nap használjuk ezeket a védelmi eszközöket, de ha a rendszerünket biztonságosabbá szeretnénk tenni, alaposabban meg kell ismerkedni velük. A Unix mára már szabványossá vált jelszavas rendszere a védelmi rendszerek fejlődésének szempontjából jura korszaknak számító 70-es években indult el. Eltekintve attól, hogy manapság a legtöbben grafikus felület alól, például a `gdm` program segítségével használják, az idők folyamán szerkezete nem sokat változott.
A folyamat maradt a régi: a rendszerbe való bejelentkezéskor a felhasználó egy legfeljebb nyolc (korábban hat vagy hét) karakterből álló jelszót ad meg, amelyet azután a DES (adattitkosítási szabvány) algoritmus egy titkos kulcsba kódol. Ez a kulcs a továbbiakban a */etc/passwd* állomány második oszlopába kerül, ahol tulajdonképpen bárki hozzáférhet.
Az eltelt idő során azonban a jelszavas védelmi rendszerek mégis megváltoztak és fokozódott a számítógépes betörők közti versengés is. A DES-algoritmus ma már másodpercek alatt feltörhető. Még elkeserítőbbé teszi a helyzetet, hogy a hagyományos jelszavas rendszer a kulcsokat mindenki által hozzáférhető, nyilvános helyeken tárolja, ahol a behatolók könnyűszerrel rátalálhatnak.
Az egyetlen megoldás az lehetne, ha szigorú korlátozásokkal sújtanánk a felhasználókat, és még az olyan gyakori parancsok kiadását is letiltanánk, mint az `ls -l`. Bár néhány biztonsággal foglalkozó szakember akkor lenne igazán boldog, ha a megvédendő gépet kikapcsolva, több kilométer mélységben lévő ólomfalú pincében helyezhetnék el, be kell látnunk, hogy az ilyen kemény megszorítások nem vezetnek megoldásra.
A '90-es évek közepén a már széles körben használt Internet egyre több lehetőséget kínált a betörőknek még megfeszítettebbé téve a versengést. Válaszlépésként különböző védelmi rendszerek indultak fejlődésnek. (Ehhez kapcsolódik a 46. oldalon lévő Könnyű álmok című cikkünk.) Kezdetben ezeket az eszközöket csak utólagos kiegészítőként lehetett a rendszerbe illeszteni, a kilencvenes évek második felében beköszöntő „Új kor hajnalán” viszont már egymást segítve fejlődtek, és a Linux-terjesztések alapcsomagjaiba is bekerültek.

Az árnyék válasza

Az árnyékjelszavak a nevüket onnan kapták, hogy ők a hagyományos jelszavak rejtett megfelelői. A különbség az, hogy az árnyékjelszavak lakhelye nem a mindenki számára hozzáférhető /etc/passwd, hanem a */etc/shadow* állomány második oszlopa, amelyet csak a rendszergazda képes „birtokolni”. Ebben az oszlopban Linux-változattól függően egy csillag vagy egy felkiáltójel mutatja, ha az adott felhasználóhoz nincs jelszó beállítva. A /etc/passwd jelszó oszlopában mindössze egy *x* utal az árnyékjelszavak jelenlétére.



Az árnyékjelszavak általános elterjedésével a kézi beállítások szinte teljesen eltűntek. Ennek ellenére léteznek eszközök, amelyek segítik az árnyékjelszavak kézi beállítását. A `pwconv` és `grpconv` parancsokkal a felhasználókra és a csoportokra vonatkozó bejegyzéseket a /etc/shadow és a /etc/passwd állományokban mindig szinkronban tarthatjuk, de a használatuk általában szükségtelen, mivel ez a művelet az új jelszavak megadásakor önműködően zajlik. Hasonlóképpen a `pwunconv` és `grpunconv` parancsokkal ugyan hagyományos jelszavakat is létrehozhatunk, de a mai korszerű rendszerek esetében ez a visszalépés ritkán szükséges. Talán az egyetlen valóban hasznos eszköz a Debian Linux `shadowconfig` programja, amelynek `off`-on beállításával rendszerünkben gyorsan és könnyen engedélyezhetjük az árnyékjelszavak használatát.
A /etc/shadow állomány többi oszlopa is evolúciós zsákutcának bizonyul. Első pillantásra hasznosnak tűnnek, hiszen

a segítségükkel beállíthatjuk, hogy mennyi idő után kell megváltoztatni egy jelszót, a felhasználó mikor kapjon erre vonatkozó figyelmeztetést, és mennyi idő után tiltsa ki a rendszer, ha mégsem változtatta meg a jelszavát. A fejlődés tekintetében ezek az oszlopok a túlélést elősegítő vonások, azonban ezeket a beállításokat minden egyes felhasználó esetén sajnos egyenként kell elvégeznünk. Még nagyobb hátrány, hogy az időpontokra vonatkozó bejegyzéseket az 1970. január 1-je óta eltelt napok számában kell megadni. Ha pontosak akarunk lenni, ez annyira macerásnak bizonyul, hogy a rendszergazdák többsége az oszlopok nagy részét üresen hagyja, a többibe pedig olyan nagy számokat ír, amelyekkel biztosan nem kell többet foglalkoznia.

Az MD5 kora

Az hagyományos jelszavas védelmi rendszer egy másik továbbfejlesztett utóda az MD5 névre hallgató titkosító eljárás. Az MD5 a legfrissebb termék azon algoritmusok fejlődésében, amelyet ***Ronald Rivest***, az MIT professzora és a titkosító eljárások kifejlesztésében több mint egy évtizede élenjáró cég, az RSA Security alapítója dolgozott ki. Az MD5 a 8-bites gépekre finomhangolt MD2 utóda, és közvetlenül a 32-bites rendszerekre kifejlesztett MD4 algoritmus módosításából jött létre. Rivest és munkatársai szerint az MD4 annak idején túlságosan hamar került a piacra. Az MD5-öst 1991-ben szabadon felhasználható (public domain) eszközként adták ki, majd 1994-ben további módosítására került sor.

Az MD5 algoritmus az azonosítást igénylő védelmi rendszerek szabványává vált, igaz, Rivest kitart amellett, hogy eredetileg nem ezzel a céllal fejlesztették ki. Bár az elmúlt években olyan kifinomultabb titkosító eljárások jelentek meg, mint például az IDEA, a Skipjack vagy a Bowfish, eddig még egyikük sem bizonyította, hogy annyival jobb teljesítménnyel bír, amennyivel az MD5-öt kiüthetné a nyeregből.

A kilencvenes évek közepén az MD5 csak utólagos kiegészítőként kerülhetett a Linux-rendszerekbe, mára azonban a legtöbb terjesztés tartalmazza. Biztonság tekintetében az MD5 előnye a hagyományos jelszavas rendszerekben használt DES-eljárással szemben az, hogy megengedi a hosszabb jelszavak használatát, és kifinomultabb titkosítást tesz lehetővé. Az MD5 engedélyezésével rendszerünk védelmére akár 256 karakter hosszú jelszavakat is használhatunk. A jelszó hosszától függetlenül az MD5 négy lépésben végzi a kódolást, melynek eredménye egy 256 karakterből álló kulcs. Mivel ez a művelet visszafordíthatatlan (legalábbis nem sok rá az esély), az MD5 algoritmust egyirányú hashnek is nevezik. Az ismertebb Linux-változatok telepítéskor felajánlják az MD5 telepítését is.

Bár az MD5 számos mai munkaállomás és hálózat adatrendszereiben gondokat okozhat, ez mégsem ok arra, hogy ne használjuk. Ha nem vagyunk biztosak benne, hogy az MD5 engedélyezve van a rendszerünkben, két módon meggyőződhetünk erről. Nézzük meg, hogy a /etc/shadow állomány jelszóoszlopa a `$1$` karakterekkel kezdődik-e, vagy nézzük bele a ***/etc/pam.d*** könyvtár állományaiba „md5”-re végződő sorok után kutatva. Ha az MD5 nincsen engedélyezve, az ehhez szükséges állományok előkeresése és a rendszer újbóli beállítása általában annyira időigényes folyamat, hogy a legtöbb kezdő felhasználó inkább a teljes rendszer újratelepítése vagy frissítése mellett dönt.

A csontváz tagolása: PAM

Az árnyékjelszavak és az MD5-eljárás elterjedése a bőség zavarát okozhatja, mivel a működésükhöz szükséges utólagos kiegészítők sokféleképpen kombinálhatók egymással, és mindegyikük saját parancsokkal rendelkezik, mint például a `passwd` vagy `login`. Erre a dilemmára nyújt megoldást a Pluggable Authentication Method (körülbelül: „csatlakoztatható hitelesítő eljárás”), röviden a PAM. A PAM tulajdonképpen a védelmi folyamatokban és a biztonsági szintek megváltoztatásában szereplő parancsok és munkafolyamatok közti közvetítő. A PAM-módszer párhuzamosan fejlődött az árnyékjelszavakkal és az MD5-tel, és mintegy 1997 óta része a különböző Linux-változatoknak.

A PAM-ot eredetileg a ***/etc/pam.conf*** állományban lehetett beállítani, a legtöbb terjesztésben ez az állomány azonban már annyira haszontalanná vált, mint testünkben a vakbél. Helyét a /etc/pam.d könyvtár vette át.

A könyvtár egyes állományaiban beállíthatjuk, hogy egy adott parancshoz mely felhasználók vagy csoportok férhetnek hozzá, például a ***/etc/pam.d/su*** állományba a `su` parancsot szabályozó bejegyzéseket tehetünk. A ***/etc/security/limits.conf*** állományban további korlátozásokat állíthatunk be, de a /etc/pam.d legtöbb állománya a ***/lib/security*** függvénykönyvtáraira mutatva közvetítő szerepet lát el a jelszavas biztonsági eszközök és egyéb parancsok között. Ezek közé tartoznak a `chfn`, `chsh`, `cron`, `gdm` és a `login`. Ezáltal nemcsak az árnyékjelszavak és az MD5 használata válik lehetővé, hanem az olyan fejlettebb biztonsági megoldások, mint például a Kerberos bevezetése is leegyszerűsödik.

A /etc/pam.d könyvtárban található állományok összes beállítási lehetőségének részletezése nem áll módunkban, de az állományok bőségesen el vannak látva megjegyzésekkel, a szerkezetük pedig könnyen átlátható. Különös figyelmet érdemelnek a `passwd`, `gdm`, `login` és a `su` állományok, segítségükkel végezhetjük el ugyanis a jelszavas védelem legalapvetőbb beállításait. A `login` állomány szerkesztésével például szabályozhatjuk a rendszergazda bejelentkezéseit, időkorlátot rendelhetünk hozzájuk,

illetve azt is beállíthatjuk, hogy miként történjen a bejelentkezések naplózása. Ha a `su` parancsot használjuk az egyébként több beállítási lehetőséget nyújtó `sudo` helyett, akkor a /etc/pam.d/su állomány lesz a segítségünkre. Bár nem szerencsés megváltoztatni a ***security*** függvénykönyvtárakra való hivatkozásokat, de a rendelkezésünkre álló finomhangolási lehetőségeket érdemes szem előtt tartani – akik számára rendszerük biztonsága kiemelten fontos, valószínűleg nem élnek a `null`-okkal, mivel lehetővé tenné, hogy a felhasználók megváltoztassák az üres jelszavakat. További példaként említhetjük a `chsh` állományt, amelyben a felhasználók által hozzáférhető parancsfájlokat korlátozhatjuk – listájuk a ***/etc/shells*** állományban található. Röviden összefoglalva: előfordulhat, hogy a ***pam.d*** könyvtár rengeteg lehetőségét látva megfájdul a fejünk, az eligazodásra tett erőfeszítésünk azonban mégsem hiábavaló, mert segítséget nyújt, hogy mi módon tehetjük a rendszerünket biztonságosabbá. A PAM témakörét részletesebben is körbejárjuk a 46. oldalon kezdődő cikben.

Fejlődés és új felhasználási területek

Az árnyékjelszavak, az MD5- és PAM-eljárások használata mind biztonságosabbá teheti a rendszerünket. Azt azonban tartsuk figyelmünk gyújtópontjában, hogy a biztonság mindig viszonylagos. Elegendő számítási teljesítőképesség és rendelkezésre álló idő esetén nyílt támadással bármilyen rendszer feltörhető. Ráadásul egyre könnyebbé válik, mivel mind az alkatrészek, mind a betörők által használt eszközök folyamatosan fejlődnek. A jobb átláthatóság kedvéért felidézzük az eseményeket: 1994-ben az RSA Security szakértői úgy becsülték, hogy egy átlagos biztonságú gép elleni nyers számítási erőn alapuló támadás 24 napon belül sikerrel jár. Ezzel szemben az `mdcrack` program fejlesztői (az MD5-eljárással védett rendszerek biztonságát ellenőrző eszköz) állítják, hogy egy 2.2-es változatú Linux-rendszermagot futtató átlagos gépen egy 56 karakterből álló jelszó húsz másodpercen belül feltörhető. Bár ez az idő még mindig kétszer olyan hosszú, mint egy Windows-alapú rendszer feltöréséhez szükséges átlagos időtartam, de már nyilvánvaló, hogy a Linux-felhasználóknak sincs okuk az önelégültségre. A helyzet ráadásul csak romlani fog. A növekvő fenyegetés elleni védekezés egyik legjobb módja, hogy fokozottabban kihasználjuk a jelszavas védelem adta lehetőségeket. Számos, főként otthoni felhasználó a telepítés után rögtön el is feledkezik a jelszavas védelemi rendszerről, és a későbbiekben is csupán a legalapvetőbb szolgáltatásait használja, pedig egy kis odafigyelés a részletekre már elég lenne ahhoz, hogy az őrületbe kergessük a botcsinálta betörőket. Nézzünk néhány gyakorlatból ellesett példát!

- A /etc/shadow állományban adjuk meg, hogy mennyi ideig engedélyezzük egy jelszó használatát. Bár ez elég kényelmetlen megoldás, a rendszeresen váltogatott jelszavakkal mégis visszaverhetjük az olyan támadásokat, amelyek sikerességéhez inkább sok idő kell, semmint számítási teljesítőképesség.
- A /etc/pam.d/passwd állományban mind a legkisebb, mind a legnagyobb jelszóhosszat növeljük meg. Ha minden más feltétel azonos, akkor igaz az állítás, hogy minél hosszabb egy jelszó, annál tovább tart feltörni.
- A /etc/pam.d/gdm állományban csökkentsük a bejelentkezési kísérletek számát. Azok a törvényes felhasználók, akik gyakran vétenek gépelési hibákat, ugyan panaszkodni fognak, de a behatolással próbálkozókat ez valószínűleg annyira megzavarja, hogy inkább odébbállnak.
- Személyesen hagyjuk jóvá a felhasználók által választott jelszavakat, de legalábbis ragaszkodjunk ahhoz, hogy olyan programot használjanak, amely biztonságos jelszavakat hoz létre, mint például a `pwgen`. Elmondani is fájdalmas, milyen sokan vannak, akik jelszavuknak a „jelszó" szót, legkisebbik lányuk vagy aranyhaluk nevét választják.
- Telepítsünk rendszerünkbe olyan programot, mint például a `cracklib2`, ami megakadályozza a könnyen kitalálható jelszavak használatát. Egyénileg összeállíthatunk egy szójegyzéket, amelyben megadhatjuk a nemkívánatos szavak listáját. Ilyenek lehetnek az ismertebb cégek, termékek nevei vagy a gyakori felhasználói nevek. Bár a `cracklib2` helyes beállításához szükségeltetik egy kis türelem, és a /etc/pam.d állomány néhány sora elől is ki kell vennünk a megjegyzéseket, mégis nagyon valószínű, hogy a `cracklib2` vagy a hozzá hasonló programok az elkövetkező években bekerülnek az elterjedtebb Linux-változatokba.
- Ne engedélyezzük, hogy a rendszergazda távolról is bejelentkezhessen.
- Kísérjük figyelemmel, hogy mikor járnak le a jelszavak, és szűrjük ki a jelszó nélküli felhasználókat. Mindkét eset rossz szándékú belépésre ad lehetőséget.
- Kapcsoljuk ki a gépünket vagy állítsuk le a hálózati kapcsolatot, ha már nincs rá szükségünk. Ne játszd a számítógépes nehézfiút, aki folyton azzal kérkedik, hogy milyen hosszú ideje nem kellett újraindítani a rendszerét. Hálózati kapcsolat nélkül a távoli betörések lehetetlenek.

A mai rendszerek kockázatos helyzetére adott válaszként megnövekedett a jelszavas azonosítást igénylő pontok száma rendszeren belül. Ha még nem tettük meg, a következő helyeken alkalmazhatunk jelszavas védelmet:

- BIOS: használjuk a jelszavas védelmet, és győződjünk meg arról, hogy nem kerülhető meg akkor sem, ha a rendszert hajlékonylemezről vagy CD-ről indítjuk.
- Indításkezelők: alkalmazzuk a LILO `password` vagy a GRUB `lock` parancsát.
- Távolról elérhető szolgáltatások: az SSH-protokoll titkosítva küldi el a jelszavakat a hálózaton – a Telnet- és FTP-protokollok viszont nem. Ugye nem kell elmondanunk, hogy melyiket használjuk…

Természetesen csak jelszavas védelemre támaszkodva mindössze a zöldfülű betörők ellen védekezhetünk. Ráadásul minél nehezebb feltörni egy jelszót, annál nehezebb fejben tartani – ez ahhoz vezethet, hogy a felhasználók a bonyolult jelszavaikat egy öntapadós cetlire írják le, amit azután felragasztanak a billentyűzetük aljára.
Még sincs semmi okunk, hogy ne használjuk a jelszavas védelem kínálta biztonsági előnyöket. Azt pedig végképp kerüljük el, hogy kiiktassuk vagy gyengítsük a jelszavas védelmet – sajnos ez a jelenség kezdi felütni a fejét a mai Linux-változatokban abból a célból, hogy a Linux jobban hasonlítson az átlagfelhasználó által használt operációs rendszerekhez. Az eszközök a kezünkben vannak, rajtunk áll, hogy használjuk-e őket!

Bruce Byfield (bbyfield@axionet.com) szabadúszó szakíró, termékmenedzser, újságíró. Amikor nem a számítógépe előtt ül, egzotikus madarak társaságában mozog, punk-folk zenét hallgat, és kedvtelésből fájdalmasan hosszú távokat fut.

Betölthető magmodulok felhasználási módjai

Védekezzünk a kalózok eszközeivel és tegyük biztonságosabbá a rendszerünket a segítségükkel.

Sereghyi számítógépes biztonsági eszköznek közös a forrása: gépkalózok világa. Az olyan eszközöket, mint a kapupásztázó vagy a jelszófeltörő programok, eredetileg rossz-szándékú emberek tervezték, hogy velük fenyegessék a számítógépes rendszerek biztonságát. Mostanra azonban már a rendszergazdák is felhasználják ezeket kiszolgáló számítógépeik és felhasználói nevük biztonságának átvizsgálására. E cikk egy ilyen gépkalózötletet mutat be – a rendszermagmodul kiaknázását –, és példát ad arra, hogyan fokozható a rendszer biztonsága néhány ugyanilyen módszer és ötlet felhasználásával. Legelőször a saját ötletem történetét és működési elvét fogom felvázolni, ezt követően néhány példa segítségével megkísérlem eloszlatni a rendszermag programozását körüllengő mítoszt. Végül egy ugyancsak hasznos, de terjedelmesebb példát fogok megvizsgálni, amely a rendszert érő támadások egész sorozatát segít elhárítani.
Mielőtt munkához fognánk, szót kell ejtenem még a szabványos lemondási nyilatkozatról. Tudatában kell lenned, hogy a rendszermag memóriaterületén levő programhiba képes tönkretenni a számítógépedet, a rendszermag területén előforduló végtelen ciklus pedig lefagyaszthatja. A fentieknek megfelelően tehát ne fejlessz vagy ne próbálgass programmodulokat a termelésben közvetlenül résztvevő számítógépen, a fejlesztett programmodulokat alaposan vizsgáld át, hogy biztosítsd rendszered üzembiztonságát és adataid épségét. Annak érdekében, hogy a programfejlesztés során a rendszerösszeomlásból eredő adatvesztést a lehető legkisebbre csökkentsük, a program kipróbálásra virtuális gépet vagy más működésmódot utánzó programot használjunk, úgymint Bochsot, plex86-ot, felhasználói üzemmódú Linux-kaput, avagy VMware-t, illetve a munkaállomásra naplózó állományrendszert telepíthetünk, mint például az SGI XFS-rendszere. A fent mondottakon kívül fontos tudni, hogy írásunkban szereplő példák egyike sem lett SMP-gépen kipróbálva, és nagyon is valószínű, hogy többprocesszoros gépeken nem működőképesek. Nos, minthogy a bennünket érintő kérdéseket megtárgyaltuk, térjünk át a modulokra.
Néhány hónapja linuxos belső nyomkövető programon dolgoztam. Minden folyamatra vonatkozóan nyomon szerettem volna követni a rendszerhívásokat az általuk átvett adatokkal együtt. A kísérletezgetés során számos megközelítést kipróbáltam, de egyik sem volt annyira sikeres, mint amennyire szerettem volna. Itt van például mindjárt a `libc` függvény a `write()`-nál, amely lehetővé tette számomra, hogy a C-programokból kezdeményezett `write()` hívásokat naplózzam, de a dinamikus bináris eszközkezelés már csak azokra a végrehajtható állományokra korlátozódott, amelyeket az eszközkezelő könyvtár elemi utasításokra tudott bontani (C, C++ és Fortran). Az, hogy az ellenőrzés csak néhány programnyelv által előállított állomány átvizsgálására korlátozódik, kisebbfajta megszorítás, hiszen a GNU/Linux-rendszeren úgyszólván minden programot C-ben, C++-ban, vagy olyan programnyelven írtak, amely rendelkezik C, illetve C++-alapú futásidejű programkönyvtárral, ilyen a Perl vagy a Python programnyelv. Csakhogy a szóban forgó megoldás eme hiányossága engem elméleti szinten valóban zavart. Tudtam, milyen egyszerűen félre lehet vezetni a rendszert egy kevésbé ismert, nem C-re vagy C++-ra támaszkodó programnyelvből indított rendszerhívással, vagy egy gépi nyelven (assembly) készített. Világos volt, hogy lehetetlen a felhasználói területet átvizsgáló, megkerülhetetlen eszközt készíteni, valamint az is egyértelművé vált, hogy elég nehéz hasznos eszközt készíteni úgy, hogy az ember ne ásná be magát a rendszermag rejtelmeibe. Minthogy nem kívántam rendszerfoltozással foglalkozni, és a fordítás-újraindítás-próbálgatás ciklusa sem volt vonzó a számomra, nem hittem benne, hogy ezt a kérdést a rendszermag területén meg lehet oldani.
Alighogy aggodalmaimat várakozólistára tettem, és dolgozni kezdtem ezen a mostani feladaton, a helyi LUG- (Linux User Group) levelezőlistán üzenetet kaptam, amely végül ötletet adott nekem. Az üzenet, tulajdonképpen egy továbbított javaslat, a rendszermagmodul kiaknázását taglalta. Az emlegetett programmodul különösen visszataszító volt, mivel bizonyos rendszerhívások viselkedését módosította, hogy elrejtse magát az `lsmod` parancs elől, valamint láthatatlanná tegye a pásztázókat, jelszófeltörőket és szaglászó (sniffer), valamint a hasonló programokat. Kis híján felkiáltottam az irodámban: „Megtaláltam!” Nem kell rendszermagfoltozással bajlódnom, vagy újrafordítással és rendszerbetöltéssel bíbelődnöm; eszközkészítés gyanánt egyszerűen csak betölthető modult kell fejlesztenem. Felismertem, hogy a modulkihasználás mögött rejlő általános módszerbe a rendszerhívások mellé sokféle hasznos tevékenység beállítása belefér, ideértve a különböző biztonsági házirendeket, a finomabb hangolású biztonságtechnikát (mint amilyet a Unix lehetővé tesz), és természetesen a belső nyomkövető programomat.

Szia, rendszermag!

A rendszerhívások megváltoztatásának és kibővítésének érdekességeit cikkünk későbbi részében fogjuk megvizsgálni, elsőként azonban végezzünk bemelegítő ujjgyakorlatot egy rendszermag-példamodulon. Maga a mintaprogram igen egyszerű és a mindenki számára kedves első program rokona, ugyanakkor a betölthető rendszermodul két alapvető szolgáltatását képes szemléltetni: az `init_module` és a `cleanup_module` függvényeket:

```
#include <linux/kernel.h>
#include <linux/module.h>

int init_module() {
   printk("<1> Hello, rendszermag!\n");
   return 0;
}

void cleanup_module() {
   printk("<1> Nem sØrtıd m meg amiatt,"
          "hogy kit r ltØl a tÆrb l engem."
           A viszontlÆtÆsra!\n);
}
```

A fordításnál a `MODVERSION` elnevezésre a `#define`, ugyanakkor a ***linux/modversions.h*** állományra vonatkozóan az `#include` irányelvet kell használni a rendszer telepítésének megfelelően. Nevezzük parányi programunkat ***hello.c***-nek, és fordítsuk le a következő utasítással:

```
gcc -c -DMODULE -D __KERNEL__ hello.c
```

A fentiek után a jelenlegi könyvtárban létrejön a ***hello.o*** névvel rendelkező állomány. Amennyiben az X éppen működik, indítsuk el a parancsmódú konzolt, és rendszergazdaként gépeljük be az `insmod hello.o` parancsot. Ekkor a „Hello, rendszermag!" üzenetnek kell a képernyőn megjelennie. Annak ellenőrzésére, vajon a modul betöltődött-e a memóriába, használjuk az `lsmod` parancsot: ez megmutatja, hogy a modul betöltése megtörtént és a modul mekkora tárterületet foglal el. Most az `rmmod`-dal eltávolíthatjuk a szóban forgó modult: ekkor tájékoztatót kapunk arról, hogy a modul memóriából való törlése megtörtént. A ***linux/kernel.h*** és a ***linux/module.h*** függvényprototípus-állományok bármilyen programfejlesztés két legfontosabb állománya, így valószínű, hogy erre neked is szükséged lesz, bármilyen modult is fejlesztesz. A legelőnyösebb – a ***modversions.h***-tól eltérően –, ha ezek a függvényprototípus-állományok a ***/usr/include/linux*** könyvtárban találhatók, és nem a linuxos forráskód könyvtárfájában. A baj az, hogy a gyakorlat hajlamos nagyszabású leállásokhoz és fejfájáshoz vezetni. Bármilyen jelentősebb modulhoz számos további rendszermagprototípus-állományt kell használni, és a programmodulok fejlesztése közben a `grep -l /usr/include/linux` parancsot jó barátnak fogod tartani.

A hasznos „task_struct" elnevezései

Name	*Type*	*Description*
uid	uid_t	User ID
euid	uid_t	Effective user ID
gid	gid_t	Group ID
egid	gid_t	Effective group ID
pid	pid_t	Process ID
pgrp	pid_t	Process group ID
p_opptr	task_struct*	Original parent task
fs	fs_struct*	Filesystem information
blocked	sigset_t	Set of blocked signals

Gondolj a programodban az `init_module`-ra az elem létrehozójaként: ez a függvény lefoglalja a szükséges tárterületet, kezdőértéket ad a változóknak és úgy változtatja meg a rendszer állapotát, hogy a modulod is el tudja látni a feladatát. Ebben az `init_module` nem tesz egyebet, mint egyszerűen jelzi a jelenlétét, majd 0 (nulla) visszatérési értékkel mutatja működésének sikerességét, több C-függvényhez hasonlóan. Ezért a hello modul kezdeti értékadásában kizárólag a `printk` függvény hívása szerepel, amely különösen jól használható, rendelkezésünkre álló függvény. Lényegében a szabványos C-nyelvi függvénynek, a `printf`-nek megfelelően működik, két eltéréssel. Elsőként a `printk` a legszembetűnőbb módon lehetővé teszi, hogy az üzenet számára előnyben részesítési vagy rangsorolási számot adjunk meg: a relációs jelek közé zárt egyes számjegyet. Másodszor a `printk` a kimenetét folyamatosan a használt átmeneti területnek küldi, amelyet viszont a rendszermagnaplózó használ fel, s ezt ezen kívül a `syslogd` is megkapja. Minthogy a `syslog` kimenete gyakran ürítésre kerül, a gondosan kiválasztott helyre beszúrt, magas rangsorolási számú üzenettel a hibakeresést nagymértékben elősegíthetjük – különösen amiatt, hogy a rendszermag tárterületén levő hiba hajlamos a gép összeomlásához vagy a rendszermag kisebb rendellenességeihez vezetni.

Nos, miért is ne használnánk éppen a `printf`-et? A válasz egyszerű: nem használjuk, mert nem tudjuk használni. A Linux-rendszermag nem kapcsolódik a C-könyvtárhoz, emiatt az olyan jó öreg bútordarabok, amilyen a `printf` is, nem érhető el a rendszermag területén lévő kódban. Viszont magában a rendszermagban rengeteg könyvtári eljáráshoz hasonló utasítássorozat létezik, beleértve a C-programkönyvtár karakterláncot kezelő `str`-függvényeinek legtöbbjét.

A fent leírtaknak saját modulban történő használatához csak be kell fűzni a ***linux/string.h*** állományt – kérlek, vigyázz, nehogy a C-könyvtárbeli változatot fűzd be! Ha az `init_module`-t létrehozó modulnak tekintjük, akkor a `remove_module`-t eltávolító modulnak kell tekintenünk. Amennyire csak lehetséges, gondoskodj a program által hátrahagyott felesleges adatok eltávolításáról. Abban az esetben, ha nem tudsz tárterületet felszabadítani vagy adatszerkezetet helyreállítani, a normál üzemmódhoz való visszatérés végett a gépet újra kell indítanod.

Egy másik érdekes modul

Most rátérünk a haladó szintű nehézségekre. Az 1. lista (➲ http://www.linuxvilag.hu/Magazin.html/) olyan programmodult mutat be, amely naplóz minden `uid 0`-tól (rendszergazda) és `uid 500`-tól (jómagam a saját munkaállomásomon) eltérő felhasználótól érkező írási rendszerhívást, amennyiben valahol az ideiglenes tárterületen előfordul a Linux szó. Megeshet, egy kicsit erőlködni kell, hogy ennek a modulnak önmagában való hasznát belássuk, de kezeskedem, több hasznos elgondolást is bemutat. Mindezt úgy tehetjük meg, ha az írási rendszerhívást a saját függvényünkkel helyettesítjük, amely elvégzi az ellenőrzést, majd a naplózást és végül kiadja az írási kérelmet. Nos, rajta, haladjunk végig a példán lépésről lépésre!

Fordíts figyelmet valamennyi befűzendő (***include***) állományra. Természetesen seregnyi van belőlük, de azért mégsem kell kétségbe esnünk, bennünket most csak a ***linux/sched.h*** és az ***asm/uaccess.h*** állományok érdekelnek. A ***sched.h*** állomány lehetővé teszi a hozzáférést a pillanatnyi `task_struct` szerkezethez a `current` makróutasításon keresztül, valamint a pillanatnyi folyamat számos hasznos jellemzőjéhez (a `task_struct` által biztosított adatokat a ***táblázatunkban*** láthatod). Az ***uaccess.h*** állomány viszont a felhasználói adatterületről szolgáltat adatokat (részletes tájékoztatás alább található).

Még ez a néhány mező a `task_struct` adatszerkezetben is elegendő ahhoz, hogy néhány érdekes modult működőképessé tegyen. Legyen-e szabad tetszőleges felhasználónak a `su` paranccsal rendszergazdaként tevékenykednie? Ha szükséges, meggátolhatod ebben, becsomagolva a SETUID-ot, és ellenőrizve az előre beállított UID-okat, mielőtt a valódi azonosítók (SETUID) felbukkannak. Ennek segítségével a rendszermag szintjén alakíthat ki „kormánycsapatot" (`wheel group`), azaz olyan csoportot, amelynek tagjai a `su` parancson keresztül rendszergazdai jogokat gyakorolhatnak. A Szabad Program Alapítvány (Free Software Foundation – FSF) már régóta azon a véleményen van, hogy a `wheel group` a hatalomelvű rendszergazdák eszköze (bővebben a GNU `su` leírásában olvashatunk a témáról).

Az a képesség, hogy a rendszerhívások viselkedését csupán az azokat kezdeményező felhasználói azonosító (UID) alapján megvizsgáljuk vagy megváltoztassuk, nyilvánvalóan nagy teljesítmény. Megteremti a jó házirend alapjait, amelyben meg lehet szabni a „nobody" felhasználónak és társainak az uucp-, levelező- és Postgres adatbázis-felhasználók tevékenységi körét. Az említetteken kívül még hatékonyabb módszer is

létezik az átadott érték alapján a viselkedés megváltoztatására. Egyelőre figyelmen kívül hagyjuk a `sys_call_table` (rendszerhívások táblázata) és az `origwrite` rendszerváltozókat, és közvetlenül a `wrapped_write` függvényre térünk, amely nemcsak a hívó folyamat felhasználói azonosítóját (`uid`), de az átmeneti terület jellemzőjét is megvizsgálja. A legelső dolog, amelyet észre kell vennünk, az, hogy a `wrapped_write` pontosan egy `kmalloc` függvényhívással indul. Jó, de miért nem a `malloc`-ot használjuk? Gondoljunk vissza a fentebb mondottakra: még mindig a rendszermag területén vagyunk és nem férünk hozzá a `malloc` és egyéb szabványos könyvtárbeli függvényhez. De ha még hozzá is férnénk, minthogy a `malloc` hívása felhasználói területre utaló mutatót ad vissza, ez haszontalan lenne. A rendszermag területén kell helyet foglalni, ahová a `buf` értékből bemásolhatjuk az adatokat. Fontos ez a mozzanat: a tárban ugyanaz a rendszermag és a felhasználói adatterület közötti láthatósági korlát, amely megakadályozza a felhasználói programokat, hogy a rendszermag tartalmát módosítsák, valami hozzátesz valamennyit rendszermag-programozói gyakorlatunkhoz is. Amikor egy C-programból a `write` utasítást meghívjuk, egy felhasználói tárterületre utaló mutatót adunk át, amely a rendszermagból nem érhető el. Ezért amennyiben felhasználói adatterületen levő mutató által kijelölt adaton szeretnénk műveletet végezni, először is a rendszermag területére kell másolni. Ezt megteszi a `copy_from_user` makróutasítás, amely három értéket használ: a „hová” mutatót, a „honnan” mutatót és a számlálót.
A `wrapped_write` hátralevő része meglehetősen egyszerű, feltéve, hogy ismerjük a jelenlegi állapotot és a `task_struct` tartalmát. Talán egy ugyanilyen szórakoztató modul az `strstr` függvényt használná a „Pocsék Linux” karakterlánc keresésére, s ha megtalálta, a `write_buf` átváltoztatná. „A Linux a nyerő”-re, majd visszaírná a felhasználói adatterületre a `copy_to_user` makróutasítással, még az eredeti írási utasítás kiadása előtt. Ekkor, ha a gyanútlan felhasználó a „Pocsék Linux” kifejezést használta, most „A Linux a nyerő” karakterlánccal lesz helyettesítve. E helyen a `kfree` fontosságára kell felhívnom a figyelmet. A „szivárgó” memóriaterületek nemkívánatosak a rendszermagban, ezért fordítsunk nagy gondot arra, hogy `kfree`-vel mindent felszabadítsunk, amit a `kmalloc`-kal lefoglaltunk. Valójában az `init_module`-ban van elhelyezve az a kapcsoló, mely alapján a vezérlést az eredeti függvény helyett az általunk írott függvény kapja meg. Emlékezzünk vissza, hogy a `sys_call_table` tulajdonképpen a függvényekre utaló mutatók tömbje. A `SYS_write` indexértékének – vagyis az írás iránti rendszerhívásnak megfelelő állandó – megváltoztatásával az eredeti írási tevékenység újjal helyettesíthető. Minden esetben gondoskodj az eredeti függvény mentéséről, hogy a saját függvénytárból való törlésed után az eredetit vissza lehessen tölteni a tárba. Most bemutatott modulunkat ki lehet próbálni, ha fordítás után és telepítését elvégzed az `insmod`-dal, majd 0-tól és 500-tól eltérő felhasználói azonosítóval kiadod a `su` parancsot, és beírod a konzolon, hogy

```
%echo " n kedvelem a Linuxot."
```

Ekkor a rendszermag üzenetet fog küldeni, hogy már megint a Linuxot hoztad szóba. Gratulálunk!

Az utolsó példa

A 2. lista – amely egyébként az ➲ http://www.linuxvilag.hu/Magazin.html/ webcímen található meg – , egy olyan hasznos modult mutat be, amely segít megvédeni a gépünket, hogy ne essen veremromboló támadások áldozatául. Az ilyen támadások alapvetően rögzített hosszúságú ideiglenes tárterület végén túli területre kívánnak írni, ily módon felülírva a pillanatnyilag futó függvény visszatérési címét, amely rendszerint valamilyen végrehajtható állományra (***/bin/sh...***) való ugrás. Mivelhogy a `httpd`, `fingerd`, vagy `wu-ftpd` programoknál nincs értelme héjprogramokat futtatni, a következőkben ismertetünk egy módszert, amellyel ez a lehetőség kizárható. Mostanra már talán akkora gyakorlatra tettél szert, hogy a példában szereplő programkód java teljesen érthető lesz a számodra, mindössze talán egy kivétellel, a `strncpy_from_user` függvényt leszámítva. Ahogyan bizonyára már kitaláltad, ez a függvény szinte pontosan úgy működik, mint C-könyvtárbeli rokona, és okos eszköz arra, hogy a felhasználói tárterületről nullavégű karakterláncot hozzunk át. Mivel a programkód egyszerű, röviden át fogjuk tekinteni a megközelítést, és utána rád bízom, hogy újabb gondolataidat megvalósítva miképpen fokozod számítógépes rendszered biztonságát.
A 2. listában bemutatott programot könnyű üzembe állítani. Nem annyira hatékony vagy nagyteljesítményű, mint amekkorát talán elvárnánk. Mivel a programkód írásánál az egyértelműség volt a fő irányelv, a `wrapped_execve` modulban könnyű a soros keresést valami hatékonyabbra cserélni. Tulajdonképpen amit ez a modul tesz, nem más, minthogy újra meghatározza a `kill` rendszerhívásokat, téve ezt olyan módon, hogyha 42-es jelet küldök egy folyamatnak, akkor felkerül a „nem biztonságos” folyamatok listájára, vagyis egy olyan listára, amelyben a folyamatok nem engedik futni az `sh` betűket tartalmazó parancsokat. A 42 az egyik valós idejű jel, talán éppen használaton kívül van. Amennyiben mégis használod ezt a jelet, bátran helyettesítsd bármilyen 32 és 64 közötti számmal. Az `execve` ellenőrzi, vajon a folyamat biztonságos-e, és ha nem az, megpróbál-e héjprogramot indítani. Ha erre a kérdésre is igenlő válasz adható, akkor a program sikeresen befejeződik anélkül, hogy bármi történt volna. A kiszolgáló számítógép minden folyamatánál könnyű használni ezt a modult, csupán csak a `kill -42 ...` sort kell beszúrni az kezdeti értékeket beállító héjprogramokba.
A 2. lista fejlődésbeli lépést képvisel az 1. listához képest: megmutatja, hogy a rendszerhívási útvonalhoz nemcsak viselkedésmintát lehet adni, de a rendszerhívások lefutása is módosítható. Röviden: hasznos munkát végez nekünk. Remélem, a szerény példák mindenkit legalább annyira lázba hoztak, mint amennyire a rendszermag kiaknázásának lehetőségei engem felvillanyoztak, hogy növeljem a rendszer biztonságát. E cikk megadja az elinduláshoz szükséges segítséget, és a kapcsolódó címeken – részben Linux-programozók számára – a leírások bőséges tárháza található, amely segít az összetettebb és szolgáltatás-központúbb modulok megalkotásában.

William C. Benton
(willb@acm.org) egyetemi tanulmányait a University of Wisconsinon folytatta. Párhuzamos programok teljesítményfigyelése, valamint biztonsági nehézségek programnyelvi és fordítási megközelítése iránt érdeklődik.

Kapcsolódó címek

- ➲ http://www.linuxdoc.org/guides.html
- ➲ http://www.plex86.org
- ➲ http://user-mode-linux.sourceforge.net

GPG, a legjobb szabad titkosító program (1. rész)

Bevezető az alulértékelt, százszázalékosan szabad segédeszköz használatába, amelyről nem tudtad, hogy szükséges (pedig az) – a GnuPG.

Tíz év telt el azóta, hogy ***Phil Zimmermann*** megjelentette a PGP v1.0-t (Pretty Good Privacy). Az ekkor még üldözési mániások titkos eszközének tartott PGP mára a levelezés titkosításának legelfogadottabb szabványává – mi több, internetes szabvánnyá – vált. A GnuPG (GNU Privacy Guard) a kereskedelmi forgalomban kapható PGP szabadon felhasználható változata, és a legtöbb Linux-terjesztés tartalmazza. Ennek ellenére mégsem használják annyian, amennyien megtehetnék.
Te is azok közé tartozol, akik idegenkednek a GnuPG használatától? Ez a kétrészes cikk remélhetőleg téged is meggyőz alkalmazásának fontosságáról. Miután legyártottad a személyes kulcsaidat és elküldted az első titkosított leveledet, továbbá ellenőrizted annak a szuper jó programnak a biztonsági aláírását, amelyet az imént töltöttél le, örülni fogsz, hogy elfogadtad a kihívást és megismerkedtél ezzel a többszolgáltatású csodával, a GnuPG-vel.
Az első részben a PGP/GnuPG hátterével, a hozzá kapcsolódó fogalmakkal foglakozunk, majd rátérünk a gyakorlati alapokra. A második részben részletesebben tárgyaljuk a fájl- és levéltitkosítást, a kulcskezelést és a grafikus felhasználói felületeket.

Néhány szó a PGP történetéről, és arról, hogy miért nem a PGP-ről írok

1991-ben, amikor az amerikai kongresszus arra készült, hogy törvénytelennek minősítse a titkosító programok magáncélú használatát, Phil Zimmermann megjelentette a PGP v1.0-t. Ez az eredetileg szabadon felhasználható eszköz tette lehetővé a számítógép-felhasználók számára, hogy titkosítsák a személyes adataikat és a levelezésüket. Ez annyira hatékonynak bizonyult, hogy a legelszántabb és biztos anyagi háttérrel rendelkező kíváncsiskodók számára is fejtörést okozott (ide sorolandó például az Egyesült Államok kormánya).
Phil Zimmermann története fontos és magával ragadó, elolvashatod ***Simon Garfinkel*** könyvében, amely Phil weblapján (lásd ***Kapcsolódó címek***) is megtalálható. Most legyen elegendő annyi, hogy a kormány nyomozása, a szabadalmi bonyodalmak és a céges felvásárlások ellenére a PGP tovább fejlődött –, Zimmermann elképzeléseinek megfelelően szolgálva a magánszemélyek és céges felhasználók igényeit világszerte.
Amikor ezt leírom, a tágabb értelemben vett PGP-re gondolok, amely a PGP-n kívül az OpenPGP-t, valamint a GnuPG-t is jelenti. Ez utóbbi megjelenése eredményezte a PGP kulcs- és üzenetformátumának internetes szabványként való elfogadását (az RFC 2440 rögzíti). Ezáltal a PGP teljesen szabad megvalósítása minden felhasználó számára elérhetővé vált a világon.
Annak ellenére, hogy Zimmermann kétségkívül a nyílt forráskód egyik igazi úttörője, a Network Associates Inc. (NAI) által terjesztett PGP általában gondot okozhat a nyílt forrás híveinek, a linuxosok számára pedig különösképpen. A legszembetűnőbb nehézség, hogy a kereskedelmi forgalomban kapható PGP csak Windowson és Mac OS-en fut. Másodsorban a PGP Freeware is csupán a nem üzleti felhasználók számára ingyenes, azaz az oktatásban dolgozók és a nem haszonelvű szervezetek számára. Harmadsorban a NAI nagymértékben csökkentette a PGP forrásának azon részét, amelyet kódátvizsgálás céljából nyíltan elérhetővé tesz.
Ez utóbbi fejlemény eredményeképpen Phil Zimmermann lemondott a PGP Securitynél betöltött tisztségéről. Felvetődik a kérdést, hogy vajon megbízhatunk-e a PGP NAI által terjesztett változatában? Ismervén az amerikai kormány titkosítással kapcsolatos kedvezőtlen álláspontját és a próbálkozásaikat, hogy a titkosítással foglalkozó programokba hátsó kapukat építtessenek, túlságosan könnyű elképzelni, hogy a NAI enged a nyomásnak, és valóban létrehoz ilyen hátsó bejáratokat.
Mivel a nyilvánosság számára a PGP forrása nem érhető el teljes egészében, nem áll módunkban ellenőrizni a NAI állításait, melyek szerint a program biztonságos.
A GnuPG ezzel szemben teljesen nyílt forrású és ingyenes programcsomag, nagyjából ugyanazt nyújtja, mint a PGP (hiányzik belőle a VPN-támogatás és a lemezkötet-titkosítás – ezek azonban megtalálhatók a PGP Desktopban). Rövid idő alatt a GnuPG lett a Linux-felhasználók legkedveltebb levél- és adattitkosító programja, hiszen minden Linux-terjesztésben fellelhető. A GnuPG fejlesztés alatt áll, melynek vezető fejlesztője ***Werner Koch***.

Mi az a GnuPG és miért van rá szükséged?

A GNU Privacy Guard egyetlen futtatható állományból áll, a *gpg*-ből. Valójában a csomag még egy ***gpgv*** nevű programot is tartalmaz, de ez a program a gpg szolgáltatásainak kényelmi szempontok szerint kiválogatott elemeit öleli fel, így bátran állíthatjuk: a futtatható gpg mindent tartalmaz. Így lehetővé válik, hogy felváltva alkalmazzam a gpg és GnuPG kifejezéseket – ezt ki is használom a cikk hátralevő részében. A PGP kifejezést is tágabb értelmében fogom használni: nemcsak a NAI által forgalmazott termékre, hanem a PGP, az OpenPGP és a GnuPG közös protokolljára, eljárásaira és a bizalomhálóra (Web of Trust) is.
A GnuPG több kiegészítő szerep mellett négy főbb feladatot lát el: az adattitkosítást, a titkosított adatok megfejtését, a digitális aláírás kezelését és ellenőrzését. A program segítségével létrehozhatjuk és kezelhetjük a titkosításhoz szükséges kulcsokat – ezek olyan tevékenységek, amelyek nem játszanak közvetlen szerepet a titkosításban, mégis elengedhetetlenek.
Egyszerűbben fogalmazva a GnuPG-t fájlok – kifejezetten

Igény az igazi bizalomra

Széljegyzet

Képzeljük el a következőt: valaki létrehoz egy hamis kulcspárt egy programfejlesztő nevében, feladja a nyilvános kulcsot pgp.mit.edu-ra (egy népszerű kulcskiszolgáló), majd a programcsomagot egy vírusokat tartalmazó csomagra cseréli ki, és létrehozza a megfelelő aláírást. Mi akadályozza meg ebben? Semmi. A Tripwire-höz hasonlóan a PGP nem gátolja meg a hamis fájlok létrehozását, csak figyelmeztet.

Ha a csomagot és a hozzá tartozó aláírást is meghamisították, jó esetben a következők egyike történik:

- Amikor ellenőrizni próbálod az aláírást, a GnuPG figyelmeztetni fog, hogy az a kulcs, amivel az aláírást létrehozták, eltér attól, amit a legutóbb használtak (erre a figyelmezetésre akkor nyílik lehetőség, ha már rendelkezel az eredeti nyilvános kulccsal).
- Ha még nincs a nyilvános kulcsokat tartalmazó kulcscsomón a fejlesztő nyilvános kulcsa, észre fogod venni, hogy a kulcs azonosítója nem egyezik meg azzal az azonosítóval, amit az igazi fejlesztő használ a fórumokra írt leveleiben vagy a webkiszolgálóján. (Igen, lehet, hogy ehhez egy kicsit keresgélni kell a Weben).
- Valaki más már észrevette a kulcsok körüli furcsaságokat, és a gondot elhárítják, még mielőtt megpróbálnád ellenőrizni az aláírást.

Nagyon könnyen előfordulhat, hogy a fent felsoroltak közül sajnos egyik sem történik meg. A fentiekben ecsetelt csalás után nagy valószínűséggel a következő események várhatók: néhány felhasználó letölti a trójai falovat, és a legtöbbjük meg sem próbálja ellenőrizni az aláírást. Ők nekifognak és azonnal telepítik a hamis programot. Rajtuk kívül még sok felhasználó letölti a programot, majd a hamis aláírás által megkövetelt hamis nyilvános kulcsot, ellenőrzik az aláírást, majd boldogan telepítik a hamis programot.

Szerencsére az is elegendő, ha csupán egyetlen felhasználó (aki rendelkezik a fejlesztő eredeti kulcsával) észreveszi a csalást és megkongatja a vészharangot. Remélhetőleg elég korán teszi meg ahhoz, hogy a kevésbé figyelmes felhasználók hamisítványtelepítése megakadályozható legyen. A történetből két tanulságot is levonhatunk: kiemelkedően fontos, hogy a lehető legtöbb felhasználó alkalmazza is a GnuPG-t az aláírások ellenőrzésére, valamint azt is meg tudják állapítani, hogy az aláírást érvényes kulccsal hozták-e létre.

Véleményem szerint ez megkérdőjelezi azt a gyakorlatot, hogy a programcsomagot, az aláírását és az aláíró nyilvános kulcsát ugyanazon a kiszolgálón tartják. Az egész kiszolgálót sem nehezebb meghamisítani, mint a részeit.

Utolsóként álljon itt ismét egy példa a központi szervezetekbe vetett bizalom buktatóiról. Emlékezzünk, hogy márciusban a Microsoft bejelentette: két microsoftos digitális aláírás hamisítványnak bizonyult, amelyet a VeriSign által hitelesnek minősített. (A programcsomagok és frissítések hitelesítéshez a Microsoft a saját megvalósítását – a GnuPG/PGP-vel azonos elven működő digitális aláírásokat – alkalmazza).

Ha valaki a hamis tanúsítványokkal aláírt volna egy módosított Internet Explorert, és elérte volna, hogy ez a hamisított csomag a Microsoft weblapjáról letölthető legyen, az álprogramot telepítő felhasználónál megjelent volna a felbukkanó ablak, amely arról tájékoztat, hogy a telepítésre kerülő program a VeriSign szerint biztonságos forrásból származik.

Ez történik, amikor az emberek, különösen az olyan szervezetek, mint a VeriSign, nem fordítanak kellő figyelmet arra, hogy milyen kulcsokban bíznak meg vagy akár írnak alá. A Microsoft a hamis kulcsok létezését csak három hónappal a felfedezésük után jelentette be. Ellenőrizd az aláírásokat – ne bízz vakon minden kulcsban!

levelek – titkosítására és a titkosított üzenetek, valamint mellékletek megfejtésére, továbbá dokumentumok, programforrások és más elektronikus adatok digitális aláírására és az ilyen aláírások ellenőrzésére használják. A digitális aláírás ellenőrzésével megállapíthatjuk, hogy az adott fájlt ténylegesen az írta alá, aki állítja magáról, valamint azt is, hogy a fájl tartalma nem változott-e meg az átvitel során (nem módosították-e rosszhiszeműen). A program segítségével a két legfontosabb kulcscsomót is kezelhetjük: a személyes kulcsokat tartalmazó „titkos kulcscsomót" (secret keyring), és az ismerőseink, munkatársaink, üzletfeleink nyilvános kulcsait tartalmazó „nyilvános kulcscsomót" (public keyring).

A GnuPG nyilvánvalóan akkor szükséges, ha más GnuPG-felhasználókkal (vagy OpenPGP-megfelelő programokat használókkal) szeretnél titkosított üzeneteket váltani. A program lehetőséget nyújt a helyi merevlemezen található fájlok titkosítására is. Ennek akkor lehet értelme, ha például hordozható számítógép merevlemezéről van szó és fennáll az eltulajdonítás veszélye. Ha egyetlen ismerősöd sem használ GnuPG-t vagy PGP-t, és úgy érzed, hogy nincsenek titkosításra szoruló adataid, még mindig létezik egy ok, ami miatt érdemes megismerkedni a GnuPG-vel: ez a programcsomag-aláírás. Néhány forgalmas nyilvános FTP-kiszolgáló feltörésének következménye – amikor a letölthető programcsomagokat hamis „trójai falovakkal" helyettesítették –, hogy a biztonságra valamit is adó programfejlesztők elkezdtek digitálisan aláírt programcsomagokat terjeszteni.

Így működik a nyilvános kulcsú titkosítás

A nyilvános kulcsú titkosítás (PK crypto) alapjairól már esett szó „Az OpenSSH száz meg egy előnye" című írásunkban (Linuxvilág, 2001. február–március, 62. oldal). A témakör azonban annyira lényegbevágó, hogy érdemes vele újra foglalkozni, annál is inkább, mert a GnuPG-vel is kapcsolatos.

Emlékezzünk, a PK crypto alapjai egyszerűek: mindenkinek két kulcsa van, egy nyilvános kulcs és egy személyes titkos kulcs. Egy adott nyilvános kulccsal titkosíthatjuk a nyilvános kulcs tulajdonosának szánt adatokat, valamint ellenőrizhetjük a kulcs tulajdonosa által aláírt fájlokat. Titkos kulcsunkkal fejthetjük meg a hozzánk érkező titkosított üzeneteket, és hozhatunk létre digitális aláírást.

A nyilvános kulcsokat terjesztésre szánják. Akinek a nyilvános kulcsodat odaadod, kódolt üzenetet küldhet neked, és ellenőrizheti digitális aláírásod hitelességét.

Ennek megfelelően a személyes kulcsnak szigorúan titkosnak kell lennie, a jogtalan másolástól és felhasználástól gondosan meg kell védeni. Fontos, hogy a neked szánt adatokat lehetőleg csak te fejthesd vissza, és csak te állíthass ki magadnak hiteles aláírást.

A nyilvános kulcsokat sokféleképpen lehet terjeszteni, ezeknél ugyanis a sértetlenség a fontos, nem a titkosság. Titkos kulcsaidat biztos helyen kell tartanod, például egy olyan könyvtárban, amire csak te rendelkezel olvasási joggal, továbbá hosszú és nehezen megfejthető jelmondatot kell használnod. Ezt a jelmondatot kell megadnod, amikor a titkos kulcsodat használod (nemsokára elárulom, hogyan teheted meg).

```
       PØlda az alÆ rÆs-ellenırzØsre

mick@kolach:~ > gpgv gpa-0.4.1.tar.gz.sig

gpgv: Signature made Thu 05 Apr 2001
09:21:26 AM CDT
      using DSA key ID 621CC013
gpgv: Can't check signature: public key
      not found

mick@kolach:~ > gpg --keyserver pgp.mit.edu
--recv-keys 621CC013

gpg: requesting key 621CC013 from
pgp.mit.edu ...
gpg: key 621CC013: public key imported
gpg: Total number processed: 1
gpg:               imported: 1

mick@kolach:~ > gpgv --keyring pubring.gpg
gpa-0.4.1.tar.gz.sig

gpgv: Signature made Thu 05 Apr 2001
09:21:26 AM CDT
      using DSA key ID 621CC013
gpgv: Good signature from "Werner Koch
<wk@gnupg.org>"
gpgv: Notation: remark=I have not checked
the source.
      It is just the current CVS version
```

A fent leírtak minden PK crypto rendszer esetében megegyeznek. A rendszerek – az SSH, a PGP, illetve a GnuPG és egyéb PK-alkalmazások – közötti különbség a nyilvános kulcsok terjesztésében és hitelességük megállapításában rejlik. Ez nagyon fontos, mert bármilyen gondosan őrzöd a titkos kulcsodat, komoly gondban leszel, ha nem lehet különbséget tenni az igazi és az ellenségeid által terjesztett hamis nyilvános kulcsok között (lásd az ***Igény az igazi bizalomra*** című széljegyzetet).
Az SSH esetében ez a kérdés nem igazán megoldott. Ha Béla barátod szeretne belépni a kiszolgálódra, és ehhez a DSA kulcspárját óhajtja használni, akkor a nyilvános kulcsot elküldi neked levélben. Ezután rajtatok áll, hogyan döntitek el, hogy a kulcs valóban hiteles-e (a legtöbb esetben felhívod Bélát telefonon és beolvasod neki a kulcs egy részét).
A GnuPG és a PGP esetében a kulcsok valódiságának ellenőrzésére léteznek eljárások, de rajtad áll, hogy ezeket a lehetőségeket kihasználod-e. Ezeknek az eljárásoknak az összességét bizalomhálónak nevezzük.

A bizalomháló

A bizalomhálót nem nehéz megérteni. Ha Béla ismeri Tamást és aláírja Tamás kulcsát; bárki, aki ismeri Bélát, megbízhat Tamásban is, tehát ha Béla aláírja Tamás nyilvános kulcsát a saját (Béla) titkos kulcsával, akkor Tamás terjesztheti a Béla által aláírt nyilvános kulcsot. Ha valaki birtokolja Béla nyilvános kulcsát és megkapja Tamás Béla által aláírt nyilvános kulcsát, ellenőrizheti Béla aláírásának valódiságát, ami igazolja Tamás kulcsának valódiságát.
Minél többen írják alá Tamás nyilvános kulcsát, annál nagyobb a valószínűsége, hogy ha valaki megkapja Tamás nyilvános kulcsát, annak valódi kulcsát egy ismert kulcs alapján ellenőrizni tudja. Ugyanez érvényesül azokra is, akik Tamás kulcsát aláírták. Ha Tamás elküldi nekem a nyilvános kulcsát, de nem ismerem se Tamást, se Bélát, akkor Béla aláírása nem hat meg egy csöppet sem. De ha Béla kulcsát aláírta Aladár, aki az én jó barátom, megbízom a kulcsában és rendelkezem vele, megbízhatok Béla kulcsában (hiszen Aladár aláírta) és így Tamás kulcsában is. Innen származik a „háló" elnevezés.
Ez az eljárás lényegesen eltér a ***VeriSign*** és az ***Entrust*** mintájától, ahol egy központi, mindenki által elismert szervezet (PKI – public key infrastructure) hitelesíti az összes nyilvános kulcsot. A bizalomháló – még ha messze is áll a tökéletességtől – hibatűrő rendszer: a bizalom megoszlik. (A széljegyzetben láthatsz olyan esetet, amikor a VeriSign modellje nem működik).
A bizalomháló legfőbb gyengesége abban lappang, hogy kevés felhasználó veszi a fáradtságot arra, hogy aláírja mások kulcsait, és ellenőrizze az aláírások valódiságát, azaz nem jön létre igazi háló. Az árva (aláírások nélküli) kulcsok jelensége sajnos teljesen szokványos. Természetesen a te kulcsaid nem maradnak aláírás nélkül, igaz? (Ugye, nem? Hiszen már írásunk puszta olvasásával is jó úton haladsz a PGP, illetve GnuPG bizalomháló erősítése felé.)
Ha már megemlítettem a PKI-kat, el kell mondanom, hogy léteznek PGP-kulcskiszolgálók is. Ezek hatalmas nyilvános kulcstárak, de csak a kényelmet szolgálják. A PKI-től eltérően, ahol a hitelesítő szervezettől letöltött kulcsban a meghatározás alapján megbízhatsz, a bizalomháló esetében nincs biztosíték az eredetre. A PKI-t az anyakönyvi bejegyzéshez hasonlíthatjuk, a PGP kiszolgálókat pedig egy klub havonta megjelenő hírleveléhez. Mindkét forrásból megtudhatod valakinek a születésnapját, de a két forrás megbízhatóságában nagy a különbség.

A GnuPG beszerzése, fordítása és telepítése

A GnuPG, mint már említettem, a legtöbb Linux-terjesztés része. Minden biztonsággal kapcsolatos programra, tehát a GnuPG-re is igaz, hogy nem árt, ha a legfrissebb változatot használod. Időnként érdemes ellátogatni a ➲ http://www.gnupg.org weblapra és tájékozódni az új kiadásokról.
Természetesen a GnuPG forráskódja is innen tölthető le, amire akkor van szükséged, ha a programot forrásból szeretnéd fordítani, esetleg nincs futtatható csomag a rendszeredhez. Egyszerűen töltsd le a csomagot egy tetszőleges könyvtárba (a /usr/src jó választás), és add ki a következő parancsokat:

```
cd /usr/src
tar -xzvf gnupg-1.0.6.tar.gz
cd gnupg-1.0.6
./configure
make
make check
make install
```

Megjegyezném, hogy a csomag neve eltérhet. Amikor ezt a cikket írom, a GnuPG legfrissebb változata az 1.0.6-os. Mikorra azonban olvasni fogod az írásomat, előfordulhat, hogy már kiadták egy frissebb változatát. Ugyanez vonatkozik a `tar` fájl kicsomagolása után létrejövő könyvtár nevére is.
Bár a `make check` parancsot nem szükséges kiadni – régebbi gépeken időigényes lehet –, én mégis hasznosnak tartom: részleteket ír ki a támogatott titkosítási algoritmusokról és próbaprogramok egész hadát futtatja le. Így azonnal kiderül, ha valami nem sikerült a GnuPG fordítása közben. Ha a próbaprogramok valamelyike hibával tér vissza, nincs értelme telepíteni a programot, előbb meg kell szüntetni a hiba okát.
Ha gondod támad a próbaprogramok futtatásakor, nézd meg az INSTALL és README fájlokat, esetleg megoldásokra bukkansz.

GPG rendszergazdai jogosultsággal?

Valószínűleg tudod, hogy a SUID (set user ID) bit használata általában kerülendő. A SUID bit – amelyet a `chmod` paranccsal állíthatsz be – hatására az adott program annak a felhasználónak a jogosultságaival fog futni, akinek a tulajdonában van, függetlenül attól, hogy ki indította el.
Ha például egy program nevének listázásakor (az `ls -l` parancs segítségével) a felhasználó jogosultságainál az `x` helyett `s` áll, és a program tulajdonosa a rendszergazda, ez azt jelenti, hogyha bármikor elindítjuk a programot, az rendszergazdai jogosultságokkal fog rendelkezni, vagyis ugyanazt megteheti, amit a rendszergazda.
Bizonyos esetekben egyes programok ok nélkül kapják meg a SETUID bitet, miközben tulajdonosuk a rendszergazda-felhasználó. A `gpg` nem ezek közé tartozik, érdemes SETUID bittel futtatni (SETUID=root, mert a `gpg` program a rendszergazda tulajdona), ami csökkenti annak az esélyét, hogy egy felhasználó titkos kulcsot vagy jelmondatot tudjon kiolvasni a memóriából. Ajánlom, hogy a `make install` után a következő parancsot add ki: `chmod u+s /usr/bin/gpg`.

GnuPG-gyorstalpaló: a digitális aláírás ellenőrzése

Miután telepítetted a `gpg`-t (forrásból a fent leírtak alapján, vagy Linux-terjesztésed CD-lemezéről), készen állsz arra, hogy létrehozd a saját kulcspárodat, és elkezdd felépíteni a bizalomháló rád eső részét. Mivel már kevés hely áll rendelkezésemre, egy olyan műveletet ismertetek, amihez nem kell saját kulcspár: kövessük végig valaki más aláírásának az ellenőrzését.
A digitális aláírás elkészítése elterjedt módszer (immár hazánkban is elfogadott – a szerk.), segítségével megbizonyosodhatunk róla, hogy a felhasználó által letöltött programcsomag megegyezik a fejlesztő által feltöltöttel.
A különálló aláírást (a PGP-aláírás lehet külön fájlban vagy csatolt fájlként ahhoz kapcsolva, amit aláírtak) a `gpgv` programmal ellenőrizzük. Ha a programnak megadunk egy aláírást, de nem rendelkezünk az aláíró nyilvános kulcsával, a `gpgv` hibát jelez. ***Listánkon*** egy ilyen próbálkozást láthatunk, nézzük meg közelebbről! Három parancsot adtunk ki:

```
gpgv gpa-0.4.1.tar.gz.sig
gpg --keyserver pgp.mit.edu \
       --recv-keys 621CC013
gpgv --keyring pubring.gpg \
       gpa-0.4.1.tar.gz.sig
```

A `gpgv` (amely az aláírások ellenőrzésére szolgáló egyszerűsített `gpg`) első futtatásánál csupán az ellenőrzendő aláírást adtam meg. Ha meglett volna a megfelelő nyilvános kulcs a `gpgv` nyilvános kulcscsomóján (***$HOME/.gnupg/trustedkeys.gpg***), ez a parancs lefutott volna, de mivel nem birtokoltam a szükséges kulcsot, hibával tért vissza.
A második alkalommal a `gpg`-t a `--recv-keys` kapcsolónak megadott kulcsazonosítóval futtattam. Ezt az azonosítót az előzőleg futott `gpgv` kimenetéből emeltem át. Meg kellett még adnom, hogy a `gpg` a pgp.mit.edu kulcskiszolgálón keresse a meghatározott kulcsot. A program megtalálta a kulcsot és sikeresen visszatért.
A harmadik parancsnál a `--keyring` kapcsolóval megadtam a `gpgv`-nek, hogy a nyilvános kulcsot az alapértelmezett kulcscsomón keresse a ***$HOME/.gnupg/pubring.gpg*** fájlban.
A parancs ezúttal megfelelő volt.
Egy valami maradt ki a fenti példából: természetesen a letöltött kulcs valódiságának ellenőrzése, azaz meg kell győződni róla, hogy a kulcs tényleg Werner Koch saját kulcsa. A feladat nem nehéz – körülbelül húsz másodpercembe került. Rákerestem a ➲ http://www.google.com-on a `621CC013 werner koch` karakterláncra.
A találatok között számos levelezési listára elküldött levél volt, amelyekben Werner aláírása – és benne a kulcs azonosítója – szerepelt. Ha valakinek sikerülne is arra a webkiszolgálóra betörni, ahonnan a programcsomagot letöltöttem, és elhelyezne ott egy trójai falovat vagy vírust tartalmazó fájlt, továbbá azt egy hamis kulccsal aláírná, a hamis nyilvános kulcsot pedig közzétenné a pgp.mit.edu kiszolgálón, nem érne el vele sokat, hiszen egy gyors webes kereséssel könnyen fény derülhet a csalásra. Kétlem, hogy akár a legelszántabb csaló is képes legyen arra, hogy a nyilvános kulcs minden közzétett példányát (weben, levelezőlista-archívumokban) felkutassa és megváltoztassa. Láthatjuk, hogy a bizalomháló működhet, feltéve, hogy óvatosak vagyunk, és alkalmanként szánunk egy kis időt az ellenőrzésre is.
E hónapra nincs már több helyem, de a következő részben a GnuPG számos további tulajdonságára is kitérek még. Remélem, sikerült felkeltenem az érdeklődést, a cikk folytatásáig pedig ajánlom, olvassák el az 58. oldalon található írásunkat ***A konzolos levelezés alapjai és a titkosítás*** címmel.

Mick Bauer (mick@visi.com)
hálózati biztonsággal foglalkozó szaktanácsadó. 1995 óta a Linux elkötelezett híve, 1997 óta pedig OpenBSD prófétaként tevékenykedik. Mick minden kérdést és megjegyzést szívesen fogad.

Kapcsolódó címek

The GNU Privacy Handbook
➲ http://www.gnupg.org/gph/en/manual.html nem annyira célirányos, mint a gpg(1) súgóoldala, írásunknál viszont részletesebb.

A GnuPG weblap ➲ http://www.gnupg.org a GnuPG legfrissebb változataira tartalmaz hivatkozásokat.

GPA és sok hasznos leírás
Az OpenPGP Alliance weblapja ➲ http://www.openpgp.org

PGP: Pretty Good Privacy *Simson Garfinkel* tollából az O'Reilly & Associates kiadásában, 1994. Annyiban ma már idejétmúlt, hogy még a GnuPG és az OpenPGP előtti időkből származik. Leírja, hogy mi a PGP, honnan származik és mire jó.

PGP Security ➲ http://www.pgp.com Phil Zimmermann által alapított cég, amit a NAI (Network Associates) felvásárolt. Phil már nem dolgozik a cégnél. A mai napig ez mind a kereskedelmi, mind a szabadon használható PGP program „hivatalos" forrása.

Phil Zimmermann weblapja
➲ http://www.philzimmermann.com

Az RSA-hoz intézett gyakran ismételt kérdések a titkosításról napjainkban, 4.1-es változat (más néven az RSA Crypto GYK) ➲ http://www.rsalabs.com/faq/index.html
Az egyik legjobb hálózaton megtalálható anyag a titkosításról, főleg mert nagy része a nyilvános kulcsú titkosításról szól.

Könnyű álmok (9. rész)

Felhasználóazonosítás és a PAM

A Linux többfelhasználós rendszer, ami azt jelenti, hogy egy rendszert több ember használhat – akár egyszerre is, úgy, hogy adataik egymástól megbízhatóan el vannak választva. Ez azonban több kérdést is felvet: hogyan különböztetjük meg egymástól a felhasználókat? Miként tudja egy felhasználó meggyőzően bebizonyítani, hogy ő valóban az, akinek mondja magát? Hogyan választjuk el egymástól a különböző felhasználókat? Ha a felhasználókat már megnyugtatóan azonosítani tudjuk, milyen módon érjük el, hogy több különböző program azonosítási eljárásai hasonlóan vagy egyformán hangolhatók legyenek? Miként érhető el, hogy a felhasználó azonosítása ne csak egy alrendszerre legyen megfelelő? Most a felvetett kérdéseknek csupán az azonosítással kapcsolatos oldalát vizsgáljuk meg.

A felhasználók azonosítása

A többfelhasználós rendszereknek lehetővé kell tenniük a felhasználók megkülönböztetését, hogy képesek legyenek adataik elválasztására. Ha a felhasználóknak nevet adunk, akkor nincs más dolgunk, mint meggyőződni arról, hogy valóban a megfelelő felhasználó kapcsolódott-e a számítógéphez. Ennek ellenőrzésére a leggyakrabban a következő három módszert vagy keveréküket használják:

- a felhasználó valamilyen tudás birtokában van (jelszó, személyi kód (pin), titkos kérdés-válasz párok);
- a felhasználó fizikai azonosítóeszközt birtokol (hagyományos kulcs, CryptoCard, chipkártya);
- a számítógép a felhasználót valamilyen fizikai tulajdonsága alapján azonosítja, ezek a biometrikus azonosítási eljárások (ujjlenyomat, recehártya (retina), hang, szag stb.).

A rendszert nem lehet tökéletesen megvédeni, mivel a jelszó visszafejthető vagy lehallgatható, a felhasználó azonosítóeszközét pedig el lehet rabolni, a hangját rögzíthetik, az ujját levághatják… A megkövetelt módszer vagy módszerek erősségét annak arányában kell eldönteni, hogy milyen típusú a védendő rendszer, mekkora az adat értéke, és milyen erősségű támadásokra lehet számítani. Egy otthoni rendszernél nyilván felesleges az ujjlenyomattal történő azonosítás, míg egy nagy cég pénzügyi adatait tartalmazó gépnél több védelmi szint is megfontolandó (például CryptoCard, valamint erős jelszavak).
Az esetek többségében a módszer ára a döntő. Ha minden monitorra alapkiépítésben ujjlenyomat-olvasót fognak szerelni, akkor az otthoni felhasználók is ezt az eljárást fogják használni.

Kis jelszótörténelem

A jelszavas azonosítás az egyik legegyszerűbb és legolcsóbb biztonsági megoldás. Alkalmazásához nincs szükség semmilyen kiegészítőre, nem igényel hosszú betanulást és magunkkal cipelni sem kell. A Linux-rendszerek alapbeállításban jelszóra épülő azonosítást alkalmaznak, ezért ezt részletesebben kifejtjük.
A korai Unix-rendszereken kutatócsoportok dolgoztak. Az azonosítás szükségtelen volt, hiszen nélküle a munka kényelmesebben folyt, ráadásul akkoriban a rendszert és az adatokat nem kellett még senkitől sem félteniük. Amikor azonban a rendszerek kezdtek elterjedni és hálózatba kapcsolták őket, a fejlesztők hamar rájöttek, hogy valamilyen módon azonosítani kell a felhasználókat és a jogosultságaikat. A jelszavakat először nem titkosítva tárolták, csupán bizonyos feltételekhez kötötték a jelszóállomány hozzáférési jogosultságait. Néha azonban programhibából adódóan a felhasználók hozzáférhetnek egymás jelszavaihoz, ebből következően adataihoz és erőforrásaihoz is. Ezért a rendszer tervezői úgy döntöttek, hogy biztonságosabb lesz, ha a jelszavakat a felhasználók más adataival együtt tárolják ugyan, de titkosítva.

A /etc/passwd állomány

Kezdetben volt a /etc/passwd állomány, ami a felhasználók adatait tartalmazta, beleértve kódolt jelszavukat is. Az állomány a rendszergazda tulajdonában állt, közvetlenül csak ő tudta írni, mivel azonban a benne tárolt adatok nem csak az azonosításhoz voltak szükségesek (a legegyszerűbb `ls` parancs is innen veszi a felhasználói azonosítóhoz tartozó felhasználói nevet), mindenki által olvashatóvá vált, vagyis mindenki hozzáférhetett a felhasználók kódolt jelszavához. Meg kell közelebbről ismernünk az azonosítás folyamatát, hogy a felmerült nehézségek mibenlétére rávilágíthassunk.
A jelszavak kódolására olyan algoritmus használatos, amely egyirányú (úgynevezett hash-függvény), továbbá a jelszó a kódolt alakból nem fejthető vissza (jelenleg nincs rá módszer). Hogyan ellenőrzi a jelszót a rendszer? Nagyon egyszerűen: a felhasználó beírja a jelszavát, a rendszer kódolja, és ha a kódolt alak megegyezik a passwd-állományban tárolttal, akkor a helyes jelszót adták meg. Ebből következik, hogy a jelszavak közvetlenül nem szerezhetők meg. Ha azonban ismerjük egy felhasználó kódolt jelszavát, próbálgatással mi is meghatározhatjuk azt a jelszót, ami az adott kódolt mintát adja vissza. Ez azonban nem feltétlenül egyezik meg az eredeti jelszóval – a jelenséget hash-ütközésnek hívják (***1. ábra***). Amennyiben az eredeti jelszó valamely szótári szó képzett alakja, netán a felhasználó személyéből valamilyen módon kikövetkeztethető (személyi szám, barátnő neve, miegymás), akkor a támadás viszonylag gyorsan kivitelezhető. Elég nagy számítási teljesítménnyel tetszőleges jelszó megtalálható, mivel a hash-függvény értékkészlete véges (1. ábra). A könnyebb megjegyezhetőséghez a felhasználók szívesebben választanak olyan jelszavakat, amelyek valamely szótári szóból képezhetők.
A fent vázolt okokból a korai Unix-kalózok minden elérhető felhasználónak megszerezhették a kódolt jelszavát és célprogramokkal [1.] igyekeztek megtalálni az eredetiket. Az emberek rossz szokásai miatt olyan szótárak álltak össze, amelyek segítségével egy komolyabb felhasználószámú rendszeren másodpercek alatt felhasználók tucatjainak a jelszavát vissza lehetett fejteni. Így a kalózok mások nevében nyomtattak, a processzoridőt és az egyéb erőforrásokat egyaránt használták.

Jelenleg a /etc/passwd állományban a legtöbb rendszeren már nem tárolnak titkosított jelszavakat, de a Linux-telepítőkészletek egy része mind a mai napig rákérdez, akarjuk-e a shadow-támogatást a rendszerünkre telepíteni. Javaslom, mindig válasszuk ki, és most nézzük meg, mit is jelent mindez.

A /etc/shadow állomány

A fent vázolt gondok egy része megoldható lenne, ha az azonosításhoz használt adatokat egy olyan állományban tárolnánk, amelyet kizárólag a rendszergazda tud írni és csupán egy külön csoport tudná olvasni. Ez az állomány lett a /etc/shadow, ami az olyan érzékeny adatokat tartalmazza, melyek az egyéb felhasználást nem érintik (titkos jelszó, mikor kell a jelszót kötelezően lecserélni, mikor jár le a hozzáférés érvényessége és így tovább [2.]). Ezzel a helyi felhasználókat védjük meg jelszavaik ellopásától. Bár egy felhasználó jelszava más úton is megszerezhető (szociális módszerek, login brute force stb.), ezekkel a módszerekkel terjedelmi okokból nem foglalkozunk. Gyakori gond, hogy a jelszóállományt bizonyos SETUID-programok felolvassák, és ha rá tudjuk venni őket egy „jóízű” core dumpra (a program szabálytalan leállása után a memóriában maradt szemetet a rendszer egy core nevű fájlba írja ki) abból kiolvashatóvá válik a kódolt jelszó. Amennyiben odafigyelünk arra, hogy a rendszeren milyen rendszergazdai jogokkal futó programok találhatók, valamint Linux-változatunk figyeli e programok a hibamentességét, ez a hiba nem fordul elő.

1. táblázat **A jelszavak betűszáma**

osztály	leírás	karakterszám
1	digit	10
2	angol kisbetűk	23
3	angol kisbetűk és számok	33
4	angol kis- és nagybetűk	46
5	angol kis- és nagybetűk, számjegyek	56
6	nyomtatható karakterek	95
7	ASCI-karakterek	128

Jelszókódoló algoritmusok

A jelszavak kódolására olyan algoritmusokat használtak, amelyek jelenlegi tudásunk szerint egyirányúak. Ez azt jelenti, hogy a kódolt jelszóból a próbálgatáson kívül semmilyen módon nem állítható vissza az eredeti. Vajon milyen algoritmusokat használnak egy korszerű Linux-rendszerben?

A crypt() és a bigcrypt()

Az első és még ma is sok helyen alkalmazott jelszókódoló algoritmus a crypt(), amely ***Robert Morris*** és ***Ken Thompson*** munkáját dicséri, és a DES-algoritmuson alapul. Működése röviden: az alap DES-algoritmus a nyílt szöveg 64 bites részeinek a titkosítására 56 bites kulcsot használ. A crypt() algoritmus a felhasználó jelszavát titkosító kulcsként használva titkosít 64 nulla bitet, majd annak eredményét és így tovább, mindezt huszonöt alkalommal. A fenti adatokból látszik, hogy a jelszó hossza behatárolt, a használható jelszóhossz pedig nyolc karakter (***1. táblázat***). A végeredményt 11 nyomtatható karakter formájában adja vissza. Ez kerül a ***passwd***- vagy a ***shadow***-állományba. Mivel a DES-algoritmus huszonötszöri végrehajtása a korabeli gépeken akár egy másodpercig is eltartott, az algoritmus ebben a formában tökéletesen megfelelt a kor igényeinek.

A számítógépek rohamos fejlődésével azonban hamarosan túl egyszerűvé vált a jelszavak megkeresése, valamint újabb gond is akadt: ugyanazt a jelszót használva egy másik rendszeren is ugyanaz a titkosított forma állt elő (1. ábra). Morris és Thompson a crypt algoritmusát az úgynevezett ***salt*** hozzáadásával továbbfjlesztették. A ***salt*** egy 12 bites szám, amely a DES-algoritmus végeredményét módosítja. Ezt a kódoláskor a rendszer véletlenszerűen választja ki, így a jelszó visszafejtésének ideje lényegesen növekszik. További előnye, hogy ugyanazt a jelszót két különböző rendszeren beállítva a titkosított alakok eltérőek, így nem nyilvánvaló az azonosság.

Ez természetesen a rendszergazda által átmásolt azonosítókra nem igaz, ilyen esetben akár ugyanazt a jelszót is érdemes újra beállítani. A jelszó hosszának behatároltsága miatt bizonyos rendszereken (HP-UX, Ultrix, BSD) olyan algoritmusokat (crypt16(), bigcrypt()) vezettek be, amelyek 16 vagy akár több karakteres jelszavak használatára is alkalmasak.

Az MD5 és más lehetőségek

A később bevezetett algoritmusok jóval hosszabbak, ezáltal nehezebben megtalálható jelszavak megadását teszi lehetővé. Ezek közül a Linux szempontjából a legnagyobb jelentőséggel a MD5 algoritmus bír. A jelszó hosszát az algoritmus nem korlátozza, csak a ***hash*** értéke mutatja, hogy nagyjából mekkora a lehetséges különböző eredményt adó jelszavak száma (1. táblázat). Az MD5 algoritmust [3.] alkalmazó jelszókódolás is ***salt***-ot használ, ami alapesetben 64 bit, így jóval számításigényesebbé teszi a jelszókeresést. Az algoritmus műveletigénye azonban sajnos nem hangolható, ezért más rendszereken további algoritmusokat fejlesztettek. Az OpenBSD már támogatja a ***bcrypt*** nevű algoritmust, amelynek az erőforrásigénye a `cost` nevű változó segítségével hangolható. Ezzel elérhető, hogy a rendszergazdák jelszavainak megkeresésére nagyságrendekkel nagyobb erőforrást kelljen a támadóknak fordítaniuk.

A jó jelszó titka

Milyen a jó jelszó? Az erős jelszavak előállítása során figyelembe kell venni néhány fontos alapelvet [4.] Ezek nyomán az alábbi jelszavak használatát célszerű elkerülni:

- ha a jelszó valamilyen módon kapcsolatba hozható a felhasználóval (saját, ismerősök vagy házi kedvencek neve, jellemző időpontok, bármely adat, amely a `GECOS` mezőben megtalálható – telefonszámok, személyi azonosító és személyi igazolvány száma, cím, iskolák);
- ha magyar vagy nagyobb világnyelvekben ismert szótári szó bármely formában (akár visszafelé is);
- ha bármely helyi felhasználó neve akármilyen formában (akár többszörözve is);
- ha helységnév;
- ha a billentyűzeten jól ismert minták (qwerty, q1w2e3, qweasd, asdfjkl stb.);
- ha bármely korábban felsorolt forma akár egyetlen számjeggyel kiegészítve (akár visszafelé is).

A jelszó előállításánál a következőket ésszerű betartani:

- legkevesebb nyolc karakter hosszú legyen;

2. táblázat **A választható jelszavak lehetséges száma**

jelszóhossz\osztály	1	2	3	4	5	6	7
4 karakter	1e4	2.8e5	1.2e6	4.5e6	9.9e6	8.1e7	2.7e8
6 karakter	1e6	1.4e8	1.3e9	9.5e9	3e10	7.4e11	4.4e12
8 karakter	1e8	7.8e10	1.4e12	2e13	9.7e13	6.6e15	7.2e16
10 karakter	1e10	4.1e13	1.5e15	4.2e16	3e17	6e19	1.2e21
12 karakter	1e12	2.2e16	1.7e18	9e19	9.5e20	5.4e23	1.9e25
14 karakter	1e14	1.2e19	1.8e21	1.9e23	3e24	4.9e27	3.2e29
16 karakter	1e16	6.1e21	2e24	4e26	9.4e27	4.4e31	5.2e33

- tartalmazzon angol kis- és nagybetűket;
- tartalmazzon számokat;
- lehetőség szerint egyéb jeleket is magában foglaljon (.,:;'"@&#><+!%/=());
- a legjobb esetben vezérlőkaraktereket is tartalmazzon (ezekkel azonban vigyázni kell, mert nem biztos, hogy minden terminál ugyanúgy értelmezi őket);
- legyen könnyen megjegyezhető;
- legyen könnyen begépelhető (gyakoroljuk be!).

Hogyan állíthatunk elő erős, könnyen megjegyezhető jelszavakat? Felhasználható a számos jelszógyártó programok valamelyike (makepassword, pwgen), de az előállított jelszavak ritkán jegyezhetők meg könnyedén. Példa egy előállított erős jelszóra:

```
atya@tensor[18:20][1]$ pwgen -s 14 1
_Rv9T,w[%QB_eU
```

Jóval használhatóbb jelszavak hozhatók létre az alábbi egyszerű módszerrel. Egy olyan mondatot kell kiválasztani, amely könnyen megjegyezhető, majd annak szavaiból bizonyos karakterek kiválasztásával és némi „tupírozással” nagyszerű jelszó nyerhető. Nézzünk rá példát!
A következő mondatot választottam:
A gonosz farkas hosszan üldözte Piroskát az erdőben.
Az ebből nyert jelszó ***+a!ZsNet$Zn.*** lesz.
Egy másik, szintén egyszerű megoldás több rövid szó összeállítása valamilyen egyéb karakterrel. Lássunk egy példát!
fantázia, panasz, lakat
A kapott jelszó: ***fantzia%pansz+lakt!***
Mindkét megoldással az a gond, hogy az előállított jelszavak általában nehezen gépelhetők, ezért gépelés közben könnyebben leleshetők. Az egyre finomabb megoldások felé vezető úton haladva sok olyan jelszó-előállítóval találkozhattunk [5.], amely a fenti kitételek lehető legtöbbjét figyelembe veszi. Megfelelő jelszó választása esetén a kalózok a jelszó kinyeréséhez kénytelenek az összes lehetőség végigböngészésével próbálkozni (brute force).
Megjegyzés: a cikkben közölt jelszavak természetesen semmilyen rendszeren nem használatosak, mivel nyomtatásban is megjelentek.

Az azonosítási rendszerek gyenge pontja

Az egyes Unix-rendszerek fejlesztői hiába dolgoztak ki közösen, kényelmesen használható megoldást, egy gondot mégsem tudtak orvosolni. Minden program csak az előre tervezett és beépített azonosítási eljárásokat támogatta. Bizonyos programok fejlesztői arra vetemedtek, hogy az általánosan használt azonosítási eljárásokat sem támogatták. Mondanom sem kell, egy új azonosítási eljárás beépítése minden azonosítást igénylő programban nagyon bonyolult és kínos feladat.

Általános megoldás: a PAM rendszer

A fenti feladat megoldására hozták létre a PAM-rendszert (Pluggable Authentication Modules). A PAM ötlete először a Sun fejlesztőinek a fejében született meg. Elsőként részlegesen a Solaris 2.5-ös sorozatában vezették be, a teljes támogatás pedig a Solaris 2.6-ban alakult ki. A PAM finoman hangolható azonosítási rendszer, mely a korábban vázolt gondokra oly módon ad egységes megoldást, hogy az azonosítással kapcsolatos feladatokat elválasztja a programtól, így annak tudomást sem kell vennie róla, hogy éppen milyen azonosítási eljárás lett

3. táblázat **Jelszótörés sebessége John The Ripper segítségével**

Algoritmus: Standard DES				
C	több salt	49740 c/s	egy salt	46566 c/s
P	több salt	224345 c/s	egy salt	189811 c/s
Algoritmus: BSDI DES (x725)				
C	több salt	608 c/s	egy salt	417 c/s
P	több salt	513 c/s	egy salt	321 c/s
Algoritmus: FreeBSD MD5				
C	több salt	1092 c/s	egy salt	1092 c/s
P	több salt	1669 c/s	egy salt	669 c/s
Algoritmus: OpenBSD Blowfish				
C	több salt	65 c/s	egy salt	65 c/s
P	több salt	116 c/s	egy salt	116 c/s

C – Kis asztali gép, 450 MHz Pentium II processzorral
P – Kiszolgáló osztályú gép két Coppermine 700 MHz-es processzorral

beállítva. A Linux-rendszer saját PAM-ot használ. A Linux-PAM fejlesztését 1996-ban kezdték meg, legfontosabb fejlesztője a mai napig ***Andrew Morgan***. Az idők folyamán a rendszer rengeteg fejlesztésen ment keresztül. Mi a Linux-PAM 0.72-es vátozatának főbb jellemzőivel ismerkedünk meg.

A PAM lehetőségei

A PAM olyan rugalmas azonosítási rendszert ad a rendszergazda kezébe, mellyel a felhasználókat a rendszeren futtatható bármely PAM-megfelelő program segítségével azonosíthatja [7. 8.]. Az azonosítási módszer a legegyszerűbb bugyuta ***titkos kérdés-titkos válasz*** azonosítási módtól a legbonyolultabb recehártya- vagy ujjlenyomat-felismerésig terjedhet. Morgan a rendszer lehetőségeinek bemutatására az alábbi példát hozza: ha a rendszergazda (mama) a felhasználói (gyerekek) matematikai tudását szeretné tökéletesíteni, beállíthatja, hogy a kedvenc lövöldözős játékuk (természetesen csak ha PAM-megfelelő) azonosításképpen a 12 alatti számok szorzatát kérdezze. Ha a felhasználó tudja a választ, játszhat. Ha nem… Amennyiben a játék érdekes, a gyerekek egykettőre megtanulják a szorzótáblát. A PAM négy különböző szolgáltatást nyújt: azonosítás, felhasználói név (account), munkamenet (session) és jelszókezelés. Ezeket a beállítási állományokban használjuk: `auth`, `account`, `session` és `password`. Az azonosítási szolgáltatás két dolgot jelent: egyrészt a PAM valamilyen eljárással meggyőződik a felhasználó személyéről, azaz azonosítja (authenticate), másrészt a PAM a felhasználót jogosultságokkal ruházhatja fel (authorize) (például valamilyen csoportba sorolhatja be a */etc/group* alapján vagy más módszerrel). A kapcsolatkezelés a rendszeren azonosításfüggetlen szolgáltatások beállítását teszi lehetővé. Jellegzetesen ilyen a kiszolgálók felhasználószámának korlátozása, a rendszergazda bizonyos helyekről (például hálózatról) való belépésének megtiltása. A munkamenet-kezelés szolgáltatásai olyan feladatok megvalósítását teszik lehetővé, melyeknek a felhasználó hozzáférése előtt vagy után kell végrehajtódniuk (naplózás, chroot stb.). A jelszókezelési szolgáltatás a felhasználó azonosítási zsetonjának módosítását teszi lehetővé. A PAM modulokból áll, amelyek akár több szolgáltatást is nyújthatnak.

A PAM beállítása

A PAM a */etc/pam.d* könyvtárban lévő állományokon keresztül hangolható. A beállításokat korábban a */etc/pam.conf* nevű állomány tartalmazta, de elavulttá vált, ezért használatát a Linux-rendszeren nem ajánlom (akad olyan operációs rendszer, amelyben csak ezt alkalmazzák). A */etc/pam.d* könyvtárban lévő állományok neve és a beállítandó szolgáltatás neve kisbetűkkel (su, ssh, ftp stb.) írandó. Az állományokban minden sor (a megjegyzésektől eltekintve) egy PAM-modul valamilyen szolgáltatását hívja meg. Az egyes sorok formátuma a következő:

<modul típusa> <ellenőrző zászló (control flag)>
<modulelérési út> <változó (argument)>

ahol a ***modul típusa*** egy a később megadott modul szolgáltatásai közül. A megadható lehetőségek: `auth`, `account`, `session`, `password`. Jelentésüket korábban már részletesen ismertettük. Az ***ellenőrző zászló*** lehetővé teszi, hogy az adott sorban meghatározott feltétel szükségességét szabályozzuk. Értékei a következők lehetnek:

- required (kötelező): a megadott modulnak sikeresen kell visszatérnie, különben a modultípusban megadott szolgáltatás nem térhet vissza hibátlanul. A modul hibája addig nem jut el a felhasználóhoz, amíg az ugyanilyen modultípussal rendelkező sorok mindegyike ki nem értékelődik.
- requisite (szükséges): az előzőtől csak annyiban különbözik, hogy a hibát a modul azonnal visszaadja az alkalmazásnak.
- sufficient (elégséges): ha a modul sikerrel tér vissza, és a korábban feldolgozott modulok között nincs sikertelenül visszatért ***required*** típusú, akkor erre a szolgáltatásra nem hívódik meg több a később felsoroltakból.
- optional (tetszőleges): miként a neve is mutatja, ez a bejegyzés nem életbevágóan fontos – amennyiben nem önmagában áll, nem okoz sem elutasítást, sem biztos elfogadást. Ilyen esetben csak a beállításainak köszönhetően teszik be a szolgáltatás végrehajtásába (naplóz, valamit kiír, egyéb).

A modul elérési útja megadja a helyét az állományrendszeren, a változók és azok beállítása pedig modulonként és szolgáltatásonként egyediek. A külön nem beállított alrendszerek az ***other*** állományban megadottak szerint vannak azonosítva. Egy kis példa a máshol nem meghatározott PAM-kérések kitiltására:

```
#
# /etc/pam.d/other - default; deny access
#

auth      required /usr/lib/security/pam_deny.so
accoun    required /usr/lib/security/pam_deny.so
password  required /usr/lib/security/pam_deny.so
session   required /usr/lib/security/pam_deny.so
```

A legfontosabb PAM-egységeket és helyes használatukat következő lapszámunkban mutatjuk be.

Hivatkozás jegyzék

[1.] Ismertebb jelszótörő programok: *Openwall Project John The Ripper* ➲ http://www.openwall.com/john/ *Alec Muffett Crack 5.0* ➲ http://www.users.dircon.co.uk/~crypto/ *LC3 (L0phtCrack 3)* ➲ http://www.atstake.com/research/lc3/

[2.] shadow(5) (elérhető a `man 5 shadow` paranccsal)

[3.] MD5: RFC 1321

[4.] GeodSoft: Good and Bad Passwords HOWTO ➲ http://geodsoft.com/howto/password/

[5.] A ***pwgen*** jelszólétrehozó honlapja ➲ http://sourceforge.net/projects/pwgen/

[6.] A Sun PAM honlapja ➲ http://www.sun.com/software/solaris/pam/

[7.] ***Andrew G. Morgan*** Linux-PAM System Administrator's Guide ➲ http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/

[8.] A Linux-PAM honlapja ➲ http://www.kernel.org/pub/linux/libs/pam/

Kötelező olvasmányként ***Simson Garfinkel***, ***Gene Spafford*** tollából a Practical Unix And Internet Security című könyvet ajánljuk. (ISBN: 1-56592-148-8)

Mátó Péter (atya@andrews.hu), informatikus mérnök és tanár. Biztonsági rendszerek ellenőrzésével és telepítésével, valamint oktatással foglalkozik. 1995-ben találkozott először linuxos rendszerrel. Ha teheti, kirándul vagy olvas.

Az XML (2. rész)

Folytatjuk gondolatfüzérünket a hordozható adatokról, vagyis a mindenki által könnyen használható dokumentumok készítéséről.

Sorozatunk második részében befejezzük az XML-adatfolyamok formai és nyelvtani érvényességének rövid ismertetését, majd áttekintjük azokat a fontosabb módszereket, amelyek az XML formát nagyon hatékony és széleskörűen használható eszközzé teszik.

A DTD tulajdonságlista (attributum list) megadása

Az eddigiekből kiderült, hogy az elemtulajdonságok az egyes XML-elemek jellemzésére szolgálnak, ezért a DTD-re alapuló nyelvtani érvényességvizsgálat során ezek elemzése is fontos követelmény. Természetesen nem mindegyik XML-elemnek van tulajdonságlistája, de ha létezik, akkor olyan DTD-beli nyelvtani szabályokat kell megfogalmaznunk, amelyek leírják, hogy az egyes elemek milyen jellemzőket tartalmazhatnak, illetve milyen értékeket vehetnek fel.

A VKONYV XML-adatfolyam <VENDEG>-elemének egyetlen tulajdonsága a sorszám. A tulajdonságlisták megadásának általános formája a következő:

```
<!ATTLIST egy_xml_elem_neve
          tulajdonsÆg_nØv1
                 tulajdonsÆg_jellege1 jelzı
           ...
          tulajdonsÆg_nØv_n
                 tulajdonsÆg_jellege_n jelzı
>
```

A <VENDEG>-elem példájánál maradva így adhatjuk meg a sorszam tulajdonság nyelvtanát:

```
<!ATTLIST VENDEG sorszam CDATA #REQUIRED>
```

A fenti sor mondja meg az elemző program számára, hogy a VENDEG elemnek létezik egy sorszam nevű tulajdonsága, amely CDATA-jellegű és az elem minden előfordulásánál kötelezően szerepelnie kell.

Az egyes tulajdonságok jellege a következők valamelyike lehet:

- CDATA – ezzel csak annyit állítunk, hogy a tulajdonság egy tetszőleges karaktersorozat
- ID – az ID-tulajdonság értékének egy névnek kell lennie,
- IDREF vagy IDREFS,
- ENTITY vagy ENTITIES,
- NMTOKEN vagy NMTOKENS,
- a névlista egyszerű felsorolástípus, például alma, körte, szilva.

Jelzőből a következő négy fajta ismeretes:

- #REQUIRED: az elem minden dokumentumbeli előfordulásánál egyértelműen meg kell adni a tulajdonság értékét.
- #IMPLIED: a tulajdonság megadása nem kötelező és nincs alapértelmezett értéke. Ha nincs érték megadva, az XML-feldolgozónak anélkül kell továbbhaladnia.
- Egyetlen érték: ekkor ez az egyetlen alapértelmezett értéke lesz a tulajdonságnak, például: "3,14"
- #FIXED „érték”: ilyenkor nem kötelező a tulajdonság megadása, de ha megtesszük, akkor ennek az értéknek kell lennie.

Egyedbevezetés (Entity declaration) a DTD-ben

Az XML-adatfolyam jól formált felépítését ismertető szakaszban már szó esett az egyedekről – most azt is leírjuk, hogyan és hol hozhatjuk létre őket.

A DTD-leírásban a következő három fajta egyed lehetséges: ***belső***, ***külső*** és ***paraméteregyedek***.

A ***belső*** egyedek makróhelyettesítő módszert valósítanak meg. Az

```
<!ENTITY MOL "Magyar Olaj Øs GÆzipari Rt.">
```

sor egy MOL nevű egyedet hoz létre, amelyet az XML-szövegben "&MOL;" alakban hívhatunk meg, és ennek hatására történik meg a szöveghelyettesítés.

A külső egyedek lehetővé teszik, hogy az XML-dokumentumból egy másik fájl tartalmára hivatkozzunk. Így szöveges és futtatható adatot is tartalmazhatnak. Ha a külső fájl szöveges adatot tartalmaz, az az eredeti dokumentumban a hivatkozási pont helyére illesztődik, és úgy kerül feldolgozásra, mintha a hivatkozó dokumentum része lenne. A futtatható adat értelemszerűen nem kerül elemzésre.

Paraméteregyedek csak a DTD-leírásban fordulhatnak elő.

1. lista

```
<!ELEMENT VKONYV (VENDEG)* >
<!ELEMENT VENDEG (NEV, EMAIL?, DATUM, SZOVEG)>
<!ATTLIST VENDEG sorszam CDATA #REQUIRED>
<!ELEMENT NEV (#PCDATA)>
<!ELEMENT EMAIL (#PCDATA)>
<!ELEMENT DATUM (#PCDATA)>
<!ELEMENT SZOVEG (#PCDATA)>
```

2. lista

```
<?xml version="1.0" encoding="WINDOWS-1250" ?>
<?xml-stylesheet type="text/css"
                 href="vkonyv.css" ?>
<VKONYV>
  <VENDEG sorszam="1">
...
</VKONYV>
```

3. lista

```
NEV { display: block; font-size: 20pt;
                 font-weight: bold; }
EMAIL { display: block; font-size: 15pt;
                 font-weight: bold; }
DATUM { display: block; font-size: 12pt;
                 font-weight: bold; }
SZOVEG { display: block; font-size: 10pt;
                 font-weight: bold; }
```

A paraméteregyedek megadása onnan ismerhető fel, hogy a nevük előtt a „%” (százalékjel) található. A paraméteregyed-hivatkozások a DTD-ben minden hivatkozás során feloldásra kerülnek, így a helyettesített szöveg is része a megadásnak. Nézzünk egy példát – eredetileg ezt írnánk:

```
<!ELEMENT KONYV (ELOLAP, LAPOK+, ZAROLAP)>
<!ELEMENT FUZET (ELOLAP, LAPOK+, ZAROLAP)>
```

A hasonlóság kihasználásával hozzunk létre egy *TARTALOM* nevű paraméteregyedet:

```
<!ENTITY %TARTALOM "ELOLAP, LAPOK+, ZAROLAP">
```

Az így kapott egyeddel a `KONYV` és `FUZET` elemek nyelvtani szabályai egységesebben fogalmazhatók meg:

```
<!ELEMENT KONYV (%TARTALOM;)>
<!ELEMENT FUZET (%TARTALOM;)>
```

A VKONYV XML-adatfolyam nyelvtani szabályai

A DTD-vel való ismerkedésünk befejezéseként alkossuk meg a VKONYV XML-adatfolyam nyelvtani szabályait (***1. lista***), amit a vkonyv.dtd fájlban tárolhatunk.

Az XML-adatfolyam megjelenítése

Az XML-fájlok semmilyen adatot nem tartalmaznak arra nézve, hogyan kell őket megjeleníteni. Azt láttuk, hogy az IE5 vagy a Netscape 6 alaphelyzetben faformában jeleníti meg az XML-leírásokat, kihasználva, hogy egy-egy fával lehet jellemezni őket. Az XML-dokumentumok megjelenítésére jelenleg két szabványos megoldás létezik:

- a CSS (Cascade Stylesheet) és
- az XSL (Extensible Stylesheet Language).

A CSS

A CSS részletes leírása a W3C webhelyről tölthető le (körülbelül 350 oldal), itt csak a használatát mutatjuk be. A CSS-leírások jellemzően ****.css*** fájlokban találhatók. Legyen adott egy ***vkonyv.css*** látványtervleírásunk. Ekkor az első változatú VKONYV XML-karakterfolyam megjelenítésére a CSS használatát a ***2. listán*** látható utasítással írhatjuk elő a megjelenítő alkalmazás számára. A szemléletesség kedvéért egy nagyon egyszerű CSS-t (lásd ***3. lista***) készítettem, amelyet a vkonyv.css fájlban tároltam.

A ***vkonyv.xml*** fájlba beszúrt vkonyv.css stíluslap-hivatkozás eredményeként az IE5 az XML-adatfolyamunkat az alapértelmezett fa helyett szintenként jeleníti meg. Érdekességképpen: ezt a formát a szabvány azért nevezi cascade-nak, mert a megjelenítő alkalmazás a megjelenítési beállítások különféle szintjeit egymás után vizsgálja meg. (Ezek a következő szintek: a böngésző alapértelmezései, egy külső stíluslap meghatározásai, a `<style>...</style>` tag által helyileg megadott lehetőség; majd a közvetlen stílusjegyek: `<B>`, `<FONT>`).

Az XSL

Az XSL több, mint egyszerű stíluslap-meghatározó nyelv, inkább olyan szövegfeldolgozónak tekinthető, mint a Perl vagy az AWK nyelvek. A feldolgozás eredmény-karaktersorozata természetesen egy HTML-dokumentum is lehet, amit a böngészők meg tudnak jeleníteni.

Az XSL két önálló nyelvi részből áll: az átalakító nyelvből (XSL Transformation) és a formázó nyelvből (XSL Formatting Objects, XSL-FO).

Az Apache Software Foundation több projektje is foglalkozik olyan ingyenes, a GNU/GPL felhasználási szerződésének hatálya alá eső és a Java nyelvre épülő rendszerek előállításával, amelyek XML-dokumentumok XSL-alapú megjelenítésével foglalkoznak. Ezen programok legfontosabb képviselői a következők:

- Cocoon – XML-alapú webes kiadványkészítő csomag,
- Xalan, mely az XSLT nyelvet képes értelmezni és futtatni,
- FOP (Format Objects Processor) – XSL formázóobjektumokat valósít meg (ezeket elsősorban a nem szövegalapú eredménydokumentumoknál használjuk).

5. lista A vkonyv1.xsl fájl tartalma

```
<?xml version="1.0" encoding="WINDOWS-1250" ?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
        <html><body> <H1>VEND GK NYV</H1>
      <xsl:apply-templates/>
      </body></html>
  </xsl:template>
  <xsl:template match="VKONYV"> <br/>
  <xsl:apply-templates/> <hr/>
  </xsl:template>
  <xsl:template match="VKONYV/VENDEG">
  <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="VKONYV/VENDEG/NEV">
     <hr/>Neve: <b>
     <xsl:value-of select="."/> </b>
  </xsl:template>
  <xsl:template match="VKONYV/VENDEG/EMAIL">
     Email c me: <b>
     <xsl:value-of select="."/> </b>
  </xsl:template>
</xsl:stylesheet>
```

Az XSL átalakító nyelv elemeivel olyan szabályokat határozhatunk meg, amelyek segítségével egy XML-dokumentumból egy másik dokumentumot, illetve általánosabban fogalmazva bármilyen bájtsorozatot (html-, pdf-, rtf-, ps-dokumentumokat) hozhatunk létre. Ennek következtében a létrehozott dokumentum nem szükségszerűen lesz XML-megfelelő. Az XSL Formatting Objects (XSL-FO) az XSL-ajánlás része. Ez a CSS-hez hasonlóan meghatározza az XML-dokumentumok kinézetének leírási lehetőségeit. A módszer a különböző tipográfiai, nyomdai kiadványoknál megszokott oldal- és szövegformázási elemek (fejléc, lábléc, tartalomjegyzék stb.) szabályainak leírásához is eszközöket szolgáltat. A szemléletesség érdekében elkészítettük a ***vkonyv1.xsl*** fájlleírását, amelyet a 4. (20. CD Magazin/XML) és ***5. listán*** látható módon – a CSS-hez hasonlóan – építhetünk be egy XML-dokumentumba.

Az ***ábra*** mutatja a VKONYV XML adatfolyam kinézetét akkor, amikor a vkonyv1.xsl stíluslapot használtuk. Észrevehető, hogy ez a módszer a megjelenítendő adatok körét is könnyen kezelni tudja (itt most csak a ***Név*** és az ***E-mail*** lett megjelenítve). Az XSL-lapok az egyes elemek megjelenítéséhez a mintaillesztés módszerét használják. A tagnevek `"xsl:"`-tal kezdődnek, utalva rá, hogy itt olyan XML-elemek találhatók, amelyeket az XSL-alkalmazás

VENDÉGKÖNYV

Neve: Nyíri Imre Email címe: irinyi@mol.hu

Neve: Koller József Email címe: jkoller@mailbox.hu

7. lista Egy SAX-elemző és -feldolgozó

```
//
import java.net.*;
import org.xml.sax.*;
import oracle.xml.parser.v2.*
//
//
// Az osztÆly leszÆrmazottja az
// alapØrtelmezett
// SAX-esemØnyeket kezelı osztÆlynak
//
public class TVendegListaSAX
             extends HandlerBase
{
    public static int vendeg_cnt = 0;
// Itt szÆmoljuk a vendØgeket

    //--------------------------------------
    // Ez egy esemØnykezelı eljÆrÆs, amely
    // az elemre lØpØskor mindig  nmßk dıen
    // megh v dik
    //--------------------------------------
    public void startElement( String name,
                AttribiteList attrlist )
    {
       if ( name.equals("VENDEG") )
       {
            vendeg_cnt++;
            szÆmoljuk a ➥vendØgeket
       }
    } // end startElement

   //-------
   // main
   //-------
   public static void main(String[] args)
   {

      if ( arg.length != 1 )
      {
        System.out.println
("HasznÆlat: java TVendegListaSAX xml_fÆjl");
      }

       // Az esemØnykezelı osztÆly
       // lØtrehozÆsa
       TVendegListaSAX esemenykezelo = new
                    TVendegListaSAX();

       // Egy SAX parser objektum
       // lØtrehozÆsa
       SAXParser parser = new SAXParser();

      // Az esemØnykezelı bejegyzØse a
      // most lØtrej tt parser objektumnÆl
      parser.setDocumentHandler
                       ( esemenykezelo );

      // Kezdj k az elemzØst, ek zben minden
      // œj VENDEG elemre lØpØskor  nmßk dıen
      // megh v dik a startElement tagf ggvØny,
      // azaz szÆmoljuk a vendØgeket
      try
      {
           parser.parse( UrlFromFÆjl
           ➥.createURL(args[0]));
      }
      catch (Exception)
      {
           System.out.println
            ("HIBA az elemzØs sorÆn!");
      }

        //--- BefejezØs l ki rjuk az
        // eredmØnyt a konzolra
        System.out.println("VendØgek szÆma:"
           + Integer(vendeg_cnt).toString() );

    } // end main

} // end TVendegListaSAX class
```

keretében akarunk használni. Ez a hivatkozási forma egy XML-névtér (namespace) szabványra épül, melynek célja, hogy az egyes XML-alkalmazások tagjai külön-külön névtérben legyenek (hasonlóan a C++ névtereihez). Ez esetben az `"xsl:"`-ben az `"xsl"` szó a névtér neve.

Röviden nézzük át, hogyan határozza meg a megjelenést a vkonyv1.xsl fájl tartalma. A fájl első sora arra utal, hogy XML-adatfolyam fog következni. A második sor az `"xsl"` névteret vezeti be. A 3–7 sor azt mondja, hogy az eredmény-karaktersorozatba az egyes elemek megjelenítése előtt írjuk ki a `"<html><body><H1>VEND GK NYV</H1>"` füzért, majd a mintaleírásokat (`apply-templates`) alkalmazva kezdjük el feldolgozni az XML-fa egyes elemeit. Az elemfeldolgozások után a befejező lépés, hogy a készítendő HTML-dokumentumot is befejezzük, azaz kiírjuk a `"</body></html>"` karakterfüzért. A következő sablonszakaszok arról rendelkeznek, hogy mi a teendője a fa bejárása során, illetve akkor, amikor adott XML-elemnél tart a feldolgozás. A `<xsl:template match="VKONYV/VENDEG/NEV">` sorban a `"VKONYV/VENDEG/NEV"` kifejezést XPathnak hívjuk. Az XSL-vezérlő működését jobban átgondolva világossá válik, hogy az `"apply-templates"` kulcsszóval egy rekurzív meghatározást (definíciót) tudunk a feldolgozásra megadni. Az `<xsl:value-of select="."/>` mindig a pillanatnyi faelem (node) értékét adja vissza.

Az IE5 azért tudja megjeleníteni XSL-stíluslappal ellátott XML-fájlunkat, mert ismeri az XSL átalakító nyelvet. Érdekességképpen megemlítjük, hogy az Apache Xalan használatával közvetlenül is előállíthatunk egy (XML, XSL) pár által meghatározott eredményfájlt. Nézzük meg a Xalan közvetlen használatát a fenti példára!

```
//A Xalan hasznÆlata (Java-alkalmazÆs, melyet
//a Process-osztÆly ind t)
java org.apache.xalan.xslt.Process -IN
➥vkonyv1.xml -XSL vkonyv1.xsl -OUT e.html
```

A parancs végrehajtása után egy ***e.html*** eredményfájl kelet-

kezik, amely a megadott (XML, XSL) pár alapján készült el. Ez a folyamat azért fontos, mert így a HTML-fájlok dinamikus előállítása kiszolgálóoldalon történhet.

Az XML-adatfolyam elemzése (Parse)

Az XML-dokumentumokhoz megadott nyelvtanok egyrészt egy új dokumentum létrehozásakor, másrészt egy létező dokumentum érvényességének ellenőrzéséhez szükségesek. Az XML-dokumentumok elemzésére három szabványos API terjedt el (a harmadik csak Javából használható):

- DOM-alapú (Document Object Modell alapú API) elemzők,
- SAX-alapú (Simple API for XML) elemzők,
- JAPX – Java API for XML Parsing.

A DOM felépíti az XML-adatfolyamnak megfelelő objektumokból álló fát, melynek elemeit ezután közvetlenül el lehet érni. Ez azt jelenti, hogy a memóriában létrejött objektumok tagfüggvényei és adattagjai elérhetők. A DOM fogalma ismerős lehet azok számára, akik a böngészőkben már írtak JavaScript, Java vagy Visual Basic Script programot, hiszen az ott használt Window, Document, Link, Form objektumok szintén egy DOM-fa alkotóelemei. Ez fa a dokumentum betöltődése során épül fel.
A SAX nem épít DOM-szerkezetet, ugyanis itt az elemzés során mindig csak a pillanatnyi elemet látjuk. A SAX emiatt az XML-elemek közvetlen elérését nem teszi lehetővé, csak sorosan olvassa fel őket. Ez a módszer viszont biztosít egy másik dokumentumelemzési és feldolgozási lehetőséget, ugyanis az XML-adatfolyam elemzése közben különféle események jönnek létre, amelyekre eseménykezelő függvényeket írhatunk.
A DOM és SAX API-k megadása a szabványos IDL nyelven van megfogalmazva, ezért azokat C++, Java, Pascal nyelveken kell megvalósítani. Az IBM, Oracle, Sun, Apache rendelkeznek Javában megvalósított változatokkal. Az Apache XML-elemzője az „Xerces”, amelyet mindenkinek a figyelmébe ajánlunk.
A példákat most az „Oracle XML Developer's Kit for Java” csomag használatával készítettük el, amely a
➲ http://www.otn.oracle.com címről tölthető le (itt C/C++ csomagok is találhatók).

Egy DOM-elemző működése

Egy DOM-elemző a VKONYV XML adatfolyamból hozzátartozó XML-fa felépítésének megfelelő szerkezetet épít fel a memóriában.
A Java oldaláról vizsgálva ez azt jelenti, hogy XMLElement, XMLAttr, XMLNode típusú osztályok objektumai vesznek részt a DOM-fa felépítésében. Ahelyett a felépített DOM-fa elemei XML-elemek, tulajdonságokat és adatokat megvalósító objektumok. Magát az XML-dokumentumot egy XMLDocument osztálybeli objektum képviseli.
A szemléletesség érdekében tekintsünk egy egyszerű Java programot (6. lista, lásd 20. CD Magazin/XML), ami elemzi a VKONYV XML-adatfolyamot, illetve feldolgozási tevékenység gyanánt a Java konzolon egymás alatt megjeleníti a vendégek neveit. A programban megjegyzések magyarázzák el a feldolgozó algoritmus működését.
A `java TVendegListaDOM vkonyv.xml` parancsra a program ezt jeleníti meg a konzolon:

```
Nyiri Imre
Koller J zsef
```

Egy SAX-elemző működése

A SAX-elemzők sorosan végigolvassák a bemenő XML-adatfolyamot, miközben különféle eseményeket hoznak létre.
Az elemző, feldolgozó alkalmazások írása során a fő feladat a megfelelő eseménykezelő eljárások megírása és bejegyzése.
A könnyebb érthetőség érdekében írjunk olyan Java SAX-elemzőt (*7. lista*), ami a konzolra kiírja, hogy hány vendég van leírva az XML-fájlban.
A `java TVendegListaSAX vkonyv.xml` parancsra a program a konzolon a következőt jeleníti:

```
A vendØgek szÆma: 2
```

A DOM és a SAX API az irodalomjegyzékben megadott összes webhelyről letölthető, tanulmányozásra ajánljuk őket.

További XML-módszerek röviden

Az XLL protokoll

A XLink és az XPointer feladata felváltani a HTML-dokumentumokból megismert egyszerű dokumentum-összekapcsoló és -hivatkozó (link) szolgáltatásokat. Az XLL-protokoll jóval összetettebb dolgokra képes, mint HTML-beli testvére:

- Közvetett hivatkozási lehetőség. Ez azt jelenti, hogy egy hivatkozás egy másik hivatkozásra hivatkozva érheti el a céldokumentumot. Ezzel a módszerrel valósítható meg a különféle telefonkönyvek könnyű létrehozása.
- Egy kapcsolat több dokumentumra is mutathat.
- Az XLink különböző források címzését teszi lehetővé (xml, html, pdf).

Az XLL lényeges része az XPointer, amely az XML-dokumentumok egy-egy részét címezi meg. Ennek során ismét felhasználja az XSL-nél már említett XPath nyelvet. Az XLL-protokoll azonban még fejlesztés alatt áll.

Az XSchema formai követelményeket leíró nyelv

A schema fogalom az adatbázis-kezelő rendszerek fogalomtárából érkezett az XML-módszerek világába, ahol az egyes adatelemek leírása sokkal szabatosabb, azaz az adattípus-fogalomnak megfelelő pontosságú. A cél a DTD nyelvnek egy olyan nyelvre történő cseréje, amelyben a nyelvtani szabályok sokkal többet árulnak el az egyes adatelemekről, mint az egyes adatokra kijelentett `(#PCDATA)` állítás. Más módszereknél már megszokhattuk az `integer`, `double`, `string`, `boolean` adattípusok használatát, ezt az XML-nél is megtartjuk.
A DTD felépítése nem XML-megfelelő, ellentétben az XSchema XML-alapú szerkezetével, ami további érv az XSchema használata mellett.
Mindenki saját XML-alapú schema nyelvet határozhat meg, és ha ellenőrzőt is készít hozzá, máris használatba lehet venni. Az XSchema nyelv szintén fejlesztési szakaszban van.

XML-RPC

Ez a módszer a távoli eljáráshívást az XML eszközeinek felhasználásával valósítja meg.

Nyíri Imre
(inyiri@mol.hu) jelenleg a MOL Rt.-nél dolgozik. Informatikai vállalkozásában az Internet, a Linux, valamint a Java programozás gyakorlati hasznosításával foglalkozik, örök szerelme a C++ maradt.

Kapcsolódó címek

➲ http://www.w3c.org
➲ http://www.xml.org
➲ http://www.xml.apache.org

A Samba mint PDC

A nyáron egy főként Windows NT-re támaszkodó cégnél vállaltam munkát. Szerencsére a rendszergazda hajlott az újdonságokra, így lehetőségem nyílt a Linuxot kiszolgálóként egy windowsos hálózatban kipróbálni.

Az új linuxos számítógépet két feladat ellátására terveztük: tartománykiszolgálóként kell üzemeljen, legyenek bejelentkezési parancsfájlok, valamint fájl- és CD-kiszolgálóként is működjön.
A rendszergazda kívánságára a SuSE Linux-változatot használtuk. Ez a rendszer mind a kezdő rendszergazdák, mind a haladóbb felhasználók igényeit megpróbálja kielégíteni. Természetesen a leírtak más rendszerekre is könnyen átültethetők, bizonyos esetekben csupán a fájlok helyei változnak. Mivel a későbbiekben az egészet Debianon is kipróbáltam, minden lépésnél a parancsok debianos változatát is leírom.

A Samba

A Samba fájl- és nyomtatókiszolgáló SMB protokollt használó rendszerekhez, vagyis olyan csomag, amelynek elsődleges célja fájlkiszolgálót varázsolni egy linuxos gépből windowsos hálózaton. A megosztott erőforrásokat más operációs rendszerek alól is el lehet érni. A Samba ügyfele csak a megosztott fájlokhoz férhet hozzá. Macintosh alá is létezik ügyfélprogram, a DAVE, melynek egyetlen hátránya, hogy kereskedelmi termék, tehát fizetni kell érte. Így már egy meglehetősen tarka hálózatot is elláthatunk!
Sőt, a Samba tudománya még itt sem áll meg: teljes értékű tartománykiszolgálóként üzemelhet, mely szolgáltatást azonban csak a windowsos gépek tudják kihasználni, azok közül sem mind. Ne rohanjunk azonban túlságosan előre: nézzük meg a PDC-t közelebbről.

Mi is az a PDC?

A PDC a Primary Domain Controller rövidítése. NT-s hálózatban azt a számítógépet nevezzük PDC-nek, amelyik a tartománybeléptetéseket végzi, valamint a bejelentkezési parancsfájlokat tárolja. A bejelentkezési parancsfájlok az ügyfeleken futnak, a kiszolgáló pedig valóban csak tárolja ezeket. PDC-ből csak egyetlen lehet egy hálózaton. A BDC (Backup Domain Controller) tükrözi a PDC-t, illetve szükség esetén egyfajta terheléselosztás lép életbe. BDC-ből több is előfordulhat egy hálózaton. Jelenleg a Samba hivatalos változata nem támogatja a BDC-t. Egy PDC felállításához a következők szükségesek: egy telepített, megfelelően beállított Samba, egy jelszóadatbázis, valamint egy olyan könyvtár, ahol majd a bejelentkezési parancsfájlokat tároljuk. Lássuk, hogyan oldhatjuk meg mindezt.

PDC-beállítások

A csomagot először telepítenünk kell. Ha a Samba segédeszközök szétszórtan, több csomagban helyezkednek el, mindet telepítsük. A SuSE-ben például külön ügyfélcsomag létezik, mely jól jön a hibakereséshez, és nem foglal el sok helyet. Debian alatt megkérdezi, hogy démonként fusson-e vagy az inetd-ből induljon. Ez örök bizonytalanság forrása, ha ugyanis egy folyamatosan üzemelő kiszolgálót akarunk felállítani, amelynek ez a fő feladata, mindenképpen démonként jó futtatnunk. Amennyiben viszont az inetd-ből indítjuk, megvan az az előnye, hogy a tcp-burkolón (wrapper) keresztül éri el a szolgáltatást, ami biztonságosabb. Ekkor azonban lassúbb lesz az elérés, hiszen nem figyeli folyamatosan a kaput, hanem csak szükség esetén indul el. Ez azt is jelenti, hogy kevesebb erőforrást fog felemészteni. Mindent összevetve én úgy határoztam, hogy démonként telepítem, de az ellátandó feladat alapján mindenki saját maga döntsön.
Ha telepítettük, jöhet a beállítás. A beállítóállomány a SuSE-nél a */etc/smb.conf*, Debian alatt pedig a */etc/samba/smb.conf*. Meglehetősen egyszerű felépítésű: különböző részekből áll, melyek kezdetét mindig a `'['` és `']'` jelek közé tett nevük jelzi, például: `[global]`. A következő rész eleje egyértelműen az előző végét jelenti, és a részek nem ágyazódnak egymásba. `NØv = rtØk` párok találhatók bennük, amelyek csak az adott részben fejtik ki hatásukat (kivéve a `[global]`-t). A megjegyzés jele a `'#'`, vagy a `';'`. Az alapértelmezett beállítások mind a SuSE-, mind a Debian-rendszernél igen ésszerűek, így nekünk csak a legfontosabbakat kell átírnunk.

[global]

A rendszerre általánosan érvényes beállításokat tartalmazza. Ezek közül a fontosabbak:

- `workgroup = SAMBA`
  A munkacsoport nevének SAMBA-t állít be. A kis- és nagybetűk nem különböznek. Mivel itt egy PDC-t állítunk fel, értelemszerűen nem munkacsoportról, hanem tartományról van szó. Itt kell megjegyeznem, hogy Windows 3.x-, 95- és 98-ügyfelek sohasem lehetnek teljes értékű tagjai egy tartománynak. Egy külön munkacsoportba kell őket szerveznünk, majd a tartományba történő belépést követően érhetik el a tartomány erőforrásait, a tartományban viszont nem fognak megjelenni.
- `interfaces = 192.168.1.110/255.255.255.0`
  A használandó hálózati csatolókat és a hozzájuk tartozó maszkokat adhatjuk meg. Nem kötelező kitölteni.
- `server string = %h server (Samba %v)`
  Windowsos gépeknél – ha a hálózaton tallózunk – a megjegyzés mezőben a gépünk neve mellett ez a szöveg fog megjelenni. Itt említeném meg a változóneveket: bárhol használhatjuk őket, mindegyik egy `'%'` jelből és egy betűből áll. A fenti például ezt jelent(het)i: `balazs server (Samba Version 2.0.7)`.
- `coding system = ISO8859-2`
  `client code page = 852`
  Használatukkal a magyar ékezetes betűket tartalmazó fájlnevek mind a linuxos gépen, mind az ügyfeleken helyesen fognak megjelenni.

- `wins support = yes`
  Segítségével a Sambát WINS-kiszolgálóként is üzemeltethetjük. Egy Windows-hálózaton mindenképpen érdemes felállítani egy WINS-kiszolgálót, mert sokkal gyorsabb lesz. Ha ezt a feladatot még egyik számítógépre sem bíztuk rá, mindenképpen „yes”-t írjunk.
- `os level = 65`
  `domain master = yes`
  `local master = yes`
  `preferred master = yes`
  `domain logons = yes`
  `logon script = logon.bat`
  Ezekkel állítjuk be ténylegesen a PDC-t. A `logon` parancsfájlnál megadott ***logon.bat***-ot a Samba a `[netlogons]` `megosztÆs`-ban fogja keresni (lásd később).
- `preserve case = yes`
  `short preserve case = yes`
  A kis- és nagybetűk kezelését állíthatjuk be.

`[homes]`

Valójában a `[global]`-on kívül minden rész egy-egy (share) megosztásként jelenik meg, amikor duplán kattintunk a Sambát futtató gépen. Léteznek különleges megosztások is: a `[homes]` a felhasználó saját könyvtárát helyettesíti, a `[printers]` a nyomtatókat. A többi mind egy-egy könyvtár, amelyre az elérési jogokat finoman szabályozhatjuk. Itt sem mutatok be minden beállítást, csupán egy alapbeállítást emelek ki közülük.

```
comment = Sajat konyvtar  ; ez lesz a home
path = %H/smbhome         ; nem t rli le
                          ; a fontosakat
browseable = no           ; tall zÆs k zben
                          ; nem lÆthat
read only = no            ;  rhat is bele
create mask = 0644        ; fÆjl
                          ; lØtrehozÆsakor
directory mask = 0755     ; k nyvtÆr
                          ; lØtrehozÆsakor
```

Egyetlen beállítás szorul magyarázatra, a `path`. Ha nem adjuk meg, a felhasználó home könyvtára lesz. Egy átlagos Windows-felhasználó könnyen letörölhet olyan `.`-tal (ponttal) kezdődő beállításfájlokat, amelyek nagyon fontosak. Éppen ezért tartottam lényegesnek a felhasználó saját könyvtárán `(%H)` belül egy `smbhome` könyvtár létrehozását, így az első bejelentkezéskor üres, és úgy viselkedik benne, ahogy a kedve tartja. Ezt az utat választva felhasználóink megszelídítésére természetesen ne felejtsük el az smbhome könyvtárat a /etc/skel könyvtárba felvenni, ezáltal minden új felhasználónknak önműködően létrejön.

`[netlogon]`

Smb.conf-unkba még számtalan egyéb megosztást is felvehetünk: lehetőségünk nyílik átmeneti könyvtárat, nyilvános területet és számtalan más hasznos dolgot is létrehozni. A PDC „életében” nagyon fontos szerepet játszik a logon parancsfájl. Fentebb említettem, hogy a `logon script = logon.bat` sorban megadott `logon.bat`-ot a Samba majd a `[netlogon]`-ban keresi. Itt kell megadnunk azt a könyvtárat, ahol bejelentkezési parancsfájljainkat fogjuk tárolni. Ne felejtsük el a parancsfájlt mindenki számára olvashatóvá tenni!

```
comment = Network logons
path = /home/smbnetlogon
public = no               ; bÆrki azØrt ne
                          ; olvashassa
write list = @adminkak
```

Az utolsó sorral meggyőződhetünk róla, hogy a könyvtárat csak az `adminkak` csoport tagjai írhatják. Biztonságunk érdekében erre a könyvtárra az `'x'` jogon kívül mindent vegyünk le az `others`-től. Fontos még a parancsállományok készítése során ügyelni rá, hogy a Windows másképp kezeli a sorvége jelet. Ne felejtsük el: a parancsfájlok az ügyfélen hajtódnak végre!

CD-kiszolgáló

Az eddig leírtak alapján nagyon egyszerűen készíthetünk CD-kiszolgálót: csak betesszük a CD-t a meghajtóba, majd a lemezlenyomatot egy egyszerű paranccsal elkészítjük:

```
dd if=/dev/cdrom of=/home/images/potato1.iso
```

Feltételeztem, hogy a /dev/cdrom helyes közvetett hivatkozás CD-meghajtónkra. Ezután fűzzük be a lenyomatot egy könyvtárba:

```
mount -t iso9660 -o ro,loop
➥/home/images/potato1.iso /mnt/isos/potato1
```

Ehhez rendszermagszinten hurokeszköz-támogatás (loopback) szükséges. Ezt követően osszuk meg Sambával a ***/mnt/isos*** könyvtárat. Megjegyzem, magának a lenyomatnak a megosztására is lehetőség nyílik, ekkor még a CD címkéjét is meghatározhatjuk. Legyünk azonban arra is tekintettel, hogy sok CD esetén átláthatatlanná teszi mind a beállításfájlt, mind a számítógépünk megosztásait.
Még egy ötlet, ha CD-kiszolgálót helyezel üzembe: a hurokeszközök legnagyobb száma alapértelmezés szerint nyolc, ez azonban igen kevés, ha Linux-rendszeredet ilyen célra szánod. Ha 2.2-es rendszermaggal rendelkezel és ragaszkodsz hozzá, akkor nehezebb dolgod lesz. A 2.2-es rendszermagba ugyanis bele van drótozva ez a szám, így át kell írnod egy fájlt a rendszermag forrásában, majd újrafordítanod az egészet. A ***/usr/src/linux/drivers/block/loop.c*** fájlban keresd meg a sort, és így írd át:

```
#define MAX_LOOP 256
```

Ha nem akarod túl gyakran újrafordítani a rendszermagot, érdemes a legtöbbet, vagyis 256-ot írni.
2.4-es rendszermag esetén a LILO-nak a következő rendszermagértéket kell átadni: `max_loop=256` (ehhez újra kell indítanod a számítógépet).
Remélem, tapasztalataim megosztásával a segítségedre lehettem, így a Linux használata más vállalatoknál is hasonlóan gördülékenyen mehet. Érdemes foglalkozni vele, mert megbízható, könnyen továbbfejleszthető és nem utolsósorban olcsó megoldás.

➲ http://www.samba.org.

Fülöp Balázs
(xut@freemail.hu)
17 éves gimnazista diák. Imádja a Túró Rudit, a Debian Linuxot és a teheneket. Az ELTE Radnóti Miklós Gyakorlóiskola tanulója immár ötödik éve. Kedvenc írója Slawomir Mrożek. Leginkább a számítógépes hálózatok biztonsága érdekli.

Itt a vége, fuss el véle...

Gimpről szóló cikksorozatunk a befejező részéhez érkezett, s reméljük, olvasóink kellő bepillantást nyerhettek a webes felületre szánt grafikák elkészítésének rejtelmeibe.

Ezárórészben áttekintjük az eddigi témákat, összefoglaljuk a tulajdonságokat és befejezzük a maszkolás témakörét, majd rövid bevezetést nyújtok a belső függvények készítésébe.

A kép hátterének megváltoztatása

Az előző cikkben szereplő elméleti bevezető után most a maszkolás gyakorlati részével foglalkozunk. Alkotásaink létrehozása során gyakran szükségünk lehet a háttér módosítására, ezért megnézzük, miként is változtathatjuk meg. Vessünk egy pillantást az ***1. képre***!
Az elkészült kép jól sikerültnek nevezhető, de a háttér bal szélén nyomasztóan hat a szikla, így hát lecseréljük. Első lépésként takarjuk ki a cserélendő területet, amelyet egy új csatornában ajánlott elkészíteni – ha így cselekszünk, később akár kijelölésként, akár maszkként is gyorsan alkalmazhatjuk. A maszkban mindenki kedve szerint, valamint megszokott eszközei segítségével készítheti el a kitakarást, ennek menetét sorozatunk előző részében mutattam be.
A cikkben szereplő kép szerkesztését időközben befejeztem, a munka végeredménye a ***2. képen*** látható.
Háttérként szándékosan választottam ezt a kissé meglepő képet, ezáltal ugyanis jobban elválik a korábbi és a mostani háttér. A „mű” még egyáltalán nincsen készen, például a szörfös vitorláján lévő ablakon át látszik a korábbi háttér. A hibák felsorolását folytathatnám, de gondolom, az olvasók még nálam is jobban látják ezeket. Szerencsére a Gimp kínálta maszkolásnak köszönhetően mindent kijavíthatunk. Hozzunk létre egy újabb maszkot, amely a szörfös vitorlájának ablakát tartalmazza, ezen tudjuk majd az átlátszóságot a kívánt mértékűre igazítani. A feladat megoldását olvasóinkra bízom, a munka elvégzéséhez szükséges fájlokat mind a CD-mellékleten, mind az Interneten megtalálhatjátok. Az elkészült képeket a Világhálón fogjuk „kiállítani”.

1. kép A kiindulási kép

2. kép A kép hátterének módosítása

A beépített függvények bővítése

A sorozatunk első részében számba vett szolgáltatások közül a Gimp egyik legelőnyösebb és legnagyobb témakörével eddig még nem foglalkoztunk. Többek kérésére most a Script-Fu függvények készítését tekintjük át röviden. E témakört több szempontból is sorozatunk végére hagytam:

1. Nem a programozással kezdtem, hogy a nem programozói beállítottságú olvasók se riadjanak vissza a Gimp használatától.
2. A Gimp programozásához jól kell ismerni a program működését.
3. A sorozat elsődleges célja nem a programozás elsajátítására, hanem a Linux grafikai lehetőségeinek kihasználására irányult.

Tehát figyelem, kevésbé „programozókedvű” olvasók is nyugodtan tovább olvashatják írásunkat.

Mi a Script-Fu és hogyan lehet használni?

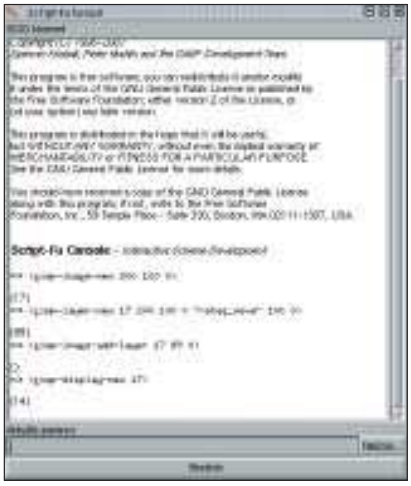
A Script-Fu elnevezés nem japán ételt vagy italt, hanem a Gimp Lisp-alapú, beépített programozói felületét takarja. A felépítés, valamint a programozási nyelvek fejlődésének részletes ismertetésétől terjedelmi okok miatt most tekintsünk el. A Script-Fu használatára két alkalommal nyílik lehetőségünk:

1. A Gimp belső parancsainak ismeretében kis programunkat szövegszerkesztő segítségével készítjük el (gyakorlott programozók a `cat` segítségével is írhatják).
2. A konzol (***3. kép***) és a DB-böngésző (***4. kép***) segítségével összeállítjuk a programot, majd a kódot kedvenc szövegszerkesztőnkkel készítjük el.

Krómfelirat készítése a Script-Fu konzolon

A következő részben a feliratot szerkesztjük meg a Script-Fu konzolon.
A rajzok elkészítése során már említettem, hogy alkotáskor az összetett feladatokat érdemes kisebb, a programban végrehajtható lépésekre bontani. Ez az állítás a függvények estében még fokozottabban érvényes. A konzol (3. kép) alsó sorába írhatjuk be a parancsokat. Ha valamelyik parancs felépítésével vagy működésével nem vagyunk teljesen tisztában, a Tallózás (Browse) gombra kattintva megnézhetjük a parancshoz tartozó leírást (4. kép).
Gépeljük be a következő parancsot:

```
(gimp-image-new 200 200 RGB)
```



3. kép A Script-Fu konzol

ekkor a konzolon az alábbi üzenetet látjuk:

```
=> (gimp-image-new 200 200 RGB)
(0)
```

Az első sorban láthatjuk a végrehajtott utasítást, a másodikban pedig az azáltal visszaadott értéket. Mit is csinált ez az utasítás? Létrehoztunk egy 200×200 méretű RGB-képet. De mi a (0)? A DB-böngészőben azt olvashatjuk, hogy ez a kép hivatkozási száma. Remek, sikerült elkészítenünk képet, viszont nem látjuk! Megszoktuk, hogy kép a létrehozása után azonnal látható lesz. Itt azonban a belső szűrők, kiegészítők miatt a képet

4. kép A beépített modulok

5. kép A Gimpben található krómfelületű függvény eredménye

láthatóvá kell tennünk. Az elején nehézséget okozhat nekünk, hogy azok a szolgáltatások, amelyeket a programban megszoktunk, itt külön működnek, tehát ahhoz, hogy a képet láthatóvá tehessük, egy réteget kell hozzáadnunk :

```
(gimp-layer-new 0 200 200
➥RGB-IMAGE "reteg"
➥100 NORMAL-MODE)

=> gimp-layer-new 0 200 200
➥RGB-IMAGE "reteg"
➥100 NORMAL-MODE
(2)
```

Sikerült! A parancs által visszaadott érték (2) a réteg hivatkozási száma. Tegyük láthatóvá a réteget:

```
(gimp-image-add-layer 0 2 0)
=> gimp-image-add-layer 0 2 0
()
```

Majd a képet is:

```
(gimp-display-new 0)
=> (gimp-display-new 0)
(1)
```

Senki ne ijedjen meg, nem rontott el semmit – a réteg csak a memóriában található „szeméttel” van feltöltve, ezért meg kell tisztítanunk:

```
(gimp-drawable-fill
➥2 GB-IMAGE-FILL)
=> (gimp-drawable-fill
➥2 BG-IMAGE-FILL)
()
```

Ennél a szolgáltatásnál ügyeljünk rá, hogy a réteg rajzolható legyen. Ha a fenti utasításokat a kódba szeretnénk írni, az alábbi módon tehetjük meg:

```
(define (my-make-new-image)
(let* ((image (car
➥(gimp-image-new 200 200 RGB)))
         (layer (car
➥(gimp-layer-new image
➥200 200 RGB-IMAGE "reteg"
➥100 NORMAL-MODE))))
        (gimp-drawable-fill
➥layer BG-IMAGE-FILL)
        (gimp-image-add-layer
➥image layer 0)
        (gimp-display-new
➥image)
        image))
```

Íme, első Script-Fu függvényünk – még nincs kész, de máris nagyon ígéretes. Írjuk bele a következő szöveget:

```
(gimp-text-fontname 0 3 0 0
➥"Linux" 0 1 25 PIXELS
➥"-abisource-courier-regular-
➥r-normal--*-*-75-75-p-90
➥-iso8859-1")
=> (gimp-text-fontname 0 3 0 0
➥"Linux" 0 1 25 PIXELS
➥"-abisource-courier-regular
➥-r-normal--*-*-75-75-p-90
➥-iso8859-1")
(3)
```

Már olvashatjuk is. (Figyelem, a fenti betűkészlet nem biztos, hogy más gépen is működik!) Készítsünk másolatot a rétegről, mivel azonban a DB-böngészőben nincs ilyen eljárás, most létre kell hoznunk egyet:

```
(define (my-duplicate-layer
➥image layer)
         (let* ((dup-layer
➥(car (gimp-layer-copy layer
➥1))))
                  (gimp-image
➥-add-layer image dup-layer 0)
➥dup-layer))
```

Miután az eljárást beírtuk a konzolba (egy sorba), hajtsuk is végre:

```
=> (my-duplicate-layer 0 3)
(8)
```

A következő lépésben méretezzük át a réteget:

```
(gimp-layer-resize 8 200 100 5 5)
```

Ezután az alfacsatorna szerint jelöljük ki, majd szövegünkhöz adjuk hozzá a vastag keretet is:

```
(gimp-selection-layer-alpha 8)
(gimp-selection-grow 0 5)*
```

A kép kiszínezése előtt rétegünket tegyük átlátszóvá:

```
(gimp-layer-set-preserve
➥-trans 8 0)
=> (gimp-blend 8 CUSTOM
➥NORMAL LINEAR 100 0 REPEAT
➥-NONE FALSE 0 0 0 0 30 50)
()
```

Parancsfájlunk számára minden apró kis építőkocka rendelkezésre áll, tehát nincs más hátra, mint leírnunk. Miután parancsfájlunk szövegszerkesztőbe vitelét sikeresen befejeztük, felmerül a kérdés: miként vehetjük rá a Gimpet, hogy használja is? A saját Script-Fu függvényeket a *~/.gimp-1.2/scripts/* könyvtárban tároljuk. A Gimp ezeket minden indulás után itt keresi, és ha a formátumuk megfelelő, azonnal használatba is vehetők. Mi szükséges ahhoz, hogy használható kódot kapjunk? A DB-böngészőben használt utasításokat adott formai követelményeknek megfelelően kell leírnunk. Kódunk végére az alábbi szöveget másoljuk be:

```
(script-fu-register
➥"kromfelirat keszitese"
   _"<Toolbox>/Xtns/Script
➥-Fu/Egyeb/Kromfelirat..."
   "Felirat keszitese
➥szinatmenettel"
   "Gabor Suveg"
   "Gabor Suveg"
   "Sept 2001"
   ""
   SF-STRING      "Text"
➥"Linux"
   SF-FONT        "Font" "-*-
➥bitwise alpha--*-r-*-*-*-*-
➥*-*-*-*-*-*"
   SF-VALUE       "Font Size"
➥"50" )
```

Az elkészült program

A fenti leírás csak betekintést kívánt nyújtani a Gimp programozási lehetőségeibe. A fenti programon még rengeteg dolgot kell javítani, illetve csiszolni. A továbbfejlesztett, kijavított szűrőt a ***Kit./Scipt-Fu/Logók/Króm…*** alatt találhatjuk meg (lásd az ***5. képen***).

Ezennel Gimppel foglalkozó sorozatunk a végéhez érkezett. További leírások és ismertetők elérhetők lesznek a ➲ http://www.gimp.hu oldalon, és a későbbiekben is szívesen válaszolok minden felmerülő kérdésre.

Köszönöm a figyelmet!

Süveg Gábor
(gsuveg@sgsytem.com)
Régóta használ Linuxot és BSD-t. Hobbija a búvárkodás, a vitorlázás és a számítógépes grafika.

A konzolos levelezés alapjai és a titkosítás

Ha utánagondolunk, egy levél elküldése során legalább két gépen halad keresztül, és felmerülhet bennünk a gyanú, hogy megőrzi-e hitelességét és épségét.

Nem új igény ez az emberiség számára, már Hérodotosz az ie. V. századi görög-perzsa háborúról szóló írásában is megemlíti. Demaratusz a perzsa Szúzában élt, így hamar értesült Xerxész Spárta elleni terveiről. Ezért – noha számkivetett volt – egy fatáblára véste figyelmeztető üzenetét, melyet aztán viasszal vont be. Ezt hívják szteganográfiának, amely a görög ***steganos*** (takar) és ***graphein*** (írni) szavak összetétele. Az elektronikus világban a szöveget gyakran képek, hangok bitjei közé rejtik, ám meg kell említeni, hogy természetesen megfelelő módszerek léteznek a felismerésükre is. A háború nyilván szélsőséges példa, ám a gyakorlat azt mutatja, hogy levelét azért mindenki borítékba teszi – noha gyenge védelem, mégis megóv a kíváncsi szemektől.

Ezzel párhuzamosan fejlődött a kriptográfia, melynek nevét a görög ***kryptos*** (rejtett) szóra vezethetjük vissza. Itt már nem magának az írásnak az elrejtése volt a cél, hanem az írás értelmének, üzenetének a kódolása. Az írást valamilyen visszafordítható eljárással érthetetlenné tették, majd ezt küldték el.

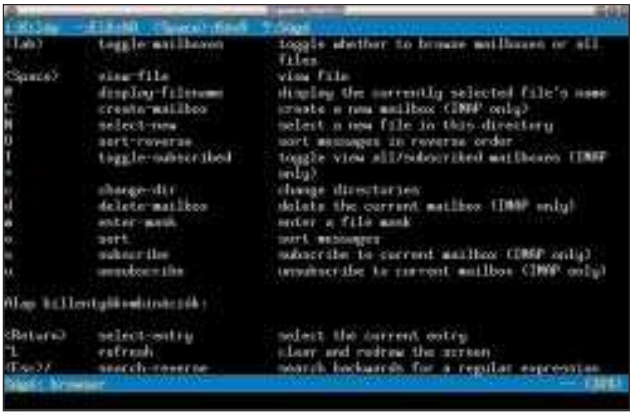
Az eljárás magában hordozta a kulcsot is, és az volt az előnye, hogyha az elküldött szöveget „elcsípték”, akkor is érthetetlen maradt. A kriptográfia alapvetően két módszert használ: a helyettesítést és az átrendezést, pontosabban egy adott kulcs segítségével egyértelműen helyettesíti az üzenet betűit, illetve egy előre rögzített szabály alapján cseréli fel azokat egymás között. Ismét következzék egy érdekes történelmi példa: az ie. VI. században a Kámaszutra szerint a nőknek 45. megszerzendő tudásként a ***mlecchita-vikalpa***-t, azaz a titkosírást kellett elsajátítaniuk. Egy javasolt módszer szerint az új ábécét az ábécé betűinek véletlenszerű társítása adta meg.

A fentiek alapján már mindenki tudna titkosítani, ám maradt még egy gond: a kulcs eljuttatása a fogadó félhez, hogy el is tudja olvasni az üzenetet. Itt lép be a nyilvános kulcsú kriptográfia, melynek lényege, hogy a kulcsot, pontosabban annak egy részét megosztjuk másokkal. A titkosítókulcsot két csoportra bontjuk: titkosra és nyilvánosra. Hogy ennek mi az értelme, egy példán keresztül szemléltetem.

Tegyük fel, hogy titkosított üzenetemet úgy küldöm el, hogy beteszem egy ládába, amit lelakatolok. A kulcsot megtartom magamnak. A címzett, amikor megkapta, ráteszi a saját lakatját is, és visszaküldi nekem. Ekkor én leszedem a saját lakatomat és visszaküldöm a ládát. Végül a címzett mindössze leszedi a saját lakatját és hozzáfér a levélhez. A „kulcsot” ezekután úgy érdemes elképzelni, hogy egy lakatból és annak kulcsából áll: a lakat legyen nyilvános, a kulcs pedig titkos. Azért kedvező, ha a lakat a nyilvános, mert előre szét lehet szórni a címzettek között. A feladó csak felteszi a címzett lakatját a ládára – a lakat ugyanis mindenki által könnyen zárható –, de (elvileg) csak a saját kulcsával nyílik, amit természetesen egyedül a címzett birtokol. Ha ilyen virtuális lakatunk lenne, könnyű lenne lemásolni és szétosztani a leendő feladók között. Nos, léteznek ilyen matematikai virtuális lakatok, amelyek az egyik irányban könnyen zárnak, ám nehéz kitalálni, milyen volt a hozzá tartozó kulcs. Ezek a módszerek leggyakrabban a véletlen (illetve a véletlenszerű) számokra és a faktorizációra épülnek.

Fontos még beszélnünk a hitelességről és az aláírásról. Amennyiben a fenti módszer segít abban, hogy a két fél megőrizze a titkát, ugyanúgy felhasználható a hitelesség garantálására is. Ehhez szavatolni kell, hogy a lakatok tulajdonosai valóban azok legyenek, akikre mi gondolunk. Ekkor az úgynevezett

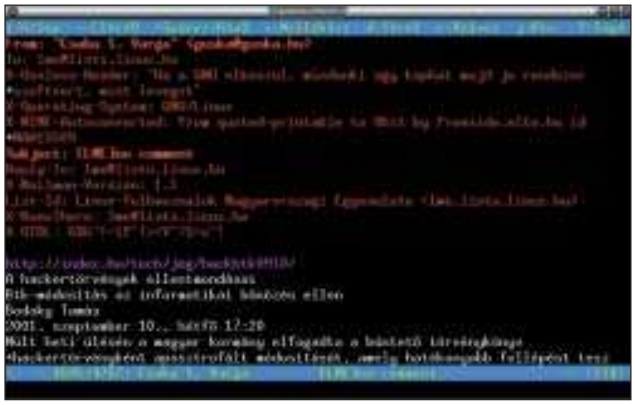
Mutt-rendezés, -rendszerezés



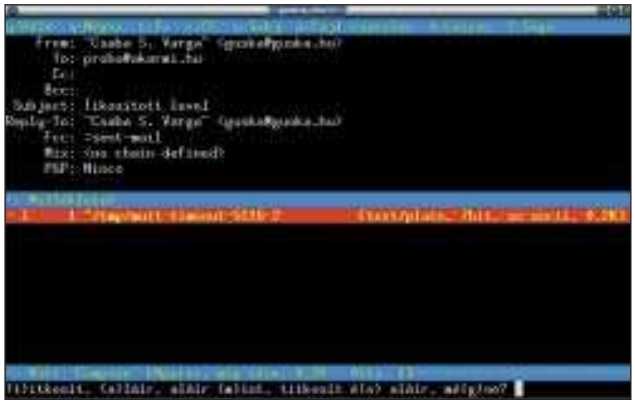
Mutt súgó

bizalmi háló használatos: a felek aláírják egymás nyilvános kulcsait, ha meggyőződtek a valódiságáról. Ezekután gondoljuk meg: ha valaki aláírta az én kulcsomat (mert ismer), akkor egy esetleges feladó, aki megbízik az ismerősömben, bízhat benne, hogy az én nyilvános kulcsomat használja. Természetesen a fenti viszonyokat igen összetetté lehet tenni, innen ered a háló elnevezés.

A fentiek összessége bonyolult rendszert képez, melynek alkalmazása különleges odafigyelést igényel, ez pedig ellentétes a könnyű használhatóság igényével. Néha olyan körülményes, hogy nem éri meg vele bajlódni, ám többnyire ilyenkor keletkeznek a biztonsági rések, hiszen lusták vagyunk és kódolást sem használunk. Ebben segít a Mutt levelezőprogram



Levél nézet



Mutt levelező és a GP menü

és a GnuPG programpáros: összekapcsolásukkal könnyen használható, olcsó levelezőrendszert hozhatunk létre, amely egy-két gombnyomásnyi közelségbe hozza a biztonságos levelezést. Nézzük, miképpen!

A Mutt levelezőprogramról mindenki úgy hiszi, hogy pilótavizsgás rendszer, de az első lépések megtétele után rá kell jönnünk, hogy bár a kezdet itt is nehéz, a továbbfejlődés könnyű és szinte határtalan. Elsőként határozzuk meg, hogy levelezőprogramunkban milyen szolgáltatásokra tartunk igényt:

- legyen könnyen kezelhető,
- öt különböző levélcímemet (család, munka stb.) kezelje,
- a beérkező leveleket válogassa – bizonyos szűrési feltételek szerint – külön könyvtárba,
- legyen belőle könnyen elérhető a GNUPGP,
- használja kedvenc szerkesztőmet és fájlnézegetőmet.

Általában majd minden Linux-terjesztésnek része a Mutt, a GPG, a Fetchmail és a Procmail csomagok, vagy forrásban hozzáférhetők az Interneten.

A Mutt első nekifutásra valóban nagyon keveset tudó alkalmazás, de ez csak azért van így, mert az alapbeállítás az úgynevezett „fapados” beállításokat tartalmazza. A telepítés után első lépésként saját könyvtárunkban a .muttrc-t kell létrehoznunk, mivel ez az állomány fog tartalmazni minden olyan apró beállítást, amelyet a program indulásakor mindig be fog olvasni, és ezek szerint fogjuk használni. Fenti igényeink megvalósításához a következő .muttrc szükséges:

```
#
# System-wide configuration file for Mutt
```

A színbeállítások

```
color hdrdefault red default
color quoted brightblue default
color signature red default
color indicator brightyellow red
color error brightred default
color status yellow blue
color tree magenta default
# the thread tree in the index menu
color tilde magenta default
color message brightcyan default
color markers brightcyan default
color attachment brightmagenta default
color search default green
# how to hilite search patterns in the pager

color header brightred default
➥^(From|Subject):
color body magenta default
➥"(ftp|http)://[^ ]+" # point out URLs
color body magenta default [-a-z_0-9.]+
➥@[-a-z_0-9.]+     # e-mail addresses
color underline brightgreen default
```

```
#
set folder=~/mail
```

Értelemszerűen itt határozzuk meg, hol legyen a kiindulási könyvtár.

```
set editor="vi"
```

Ez a kedvenc szerkesztőnk, amit a levelek megírásához szeretnénk használni.

```
set charset="iso-8859-2"
```

A levelek kódolása írásnál-olvasásánál egyaránt használatos.

```
set hostname="levelezo.munkallomas.hu"
```

A gépünk neve.

```
set sort=date-received
```

A beérkezés dátuma szerint rendezzük a leveleket, ami menet közben dinamikusan változtatható.

```
set spoolfile='~/mail/Inbox'
```

A bejövő levelek helye, illetve annak megadása, hogy az alkalmazás induláskor mely levélcímünkhöz tartozó leveleket töltse be.

```
set tmpdir=/home/guska/tmp
```

Ez adja meg, hogy a szerkesztőnk hova rakja az úgynevezett piszkozatot. Ennek akkor van nagy szerepe, ha egy esetleg bekövetkező áramszünet miatt az éppen megkezdett levelet szeretnénk megkeresni, továbbá azért is célszerű beállítani ezt a szolgáltatást, mert ellenkező esetben az alapértelmezés szerinti */tmp*-t használja, amelybe esetleg más felhasználók is beleláthatnak, és leveleink egy részét el is olvashatják.

```
my_hdr X-Operating-System: GNU/Linux
```

Megadhatjuk, hogy a levelező a levél fejlécében található operációsrendszer-meghatározásba mit is írjon. Ide érdemes nem túl sok adatot eláruló megjegyzést elhelyezni, ugyanis sokan rosszul értelmezett büszkeségből szinte a teljes rendszer jellemzését megadják, például: GNU/Linux Debian Woody, 2.4.9 rendszermag, Mutt 1.2 stb. Ezek az adatok gyengítik védelmi vonalainkat, hiszen ha például kijön egy ismert biztonsági hiba a 2.4.9-es rendszermagra, mi máris elárultuk, hogy azt használjuk.

Színbeállítások

Listánkon jól látható, hogy szinte minden mezőhöz színeket rendelhetünk, de csak a lényeges elemeket érdemes külön színnel ellátni, mert a túl színes levelező egy idő után átláthatatlan.

```
my_hdr From: Csaba S. Varga <guska@guska.hu>
my_hdr Reply-To: Csaba S. Varga <guska@guska.hu>
set signature="~/.signature"
folder-hook . my_hdr From: Csaba S. Varga
➥<guska@guska.hu>
folder-hook . my_hdr Reply-To: Csaba S. Varga
➥<guska@guska.hu>
folder-hook . my_hdr Cc: <>
folder-hook . my_hdr Bcc: <>
folder-hook . set signature="~/.signature"
```

Ezek a sorok mondják meg a levelezőprogramnak, hogyha egyszerűen csak a Postafiókból (Inbox), vagy valamilyen más, külön meg nem határozott postafiókból kezdeményezünk levélírást, ki legyen a feladó, és az aláírást melyik fájlból vegye stb.

```
folder-hook =lme my_hdr From: Csaba S. Varga
➥<guska@gnu.hu>
folder-hook =lme my_hdr Reply-To: Csaba S. Varga
➥<guska@gnu.hu>
folder-hook =lme set signature="~/.signature-lme"
```

Ha a levélírást az LME levelezőfiókból kezdeményezem, a levél a gnu.hu-s címmel és más aláírófájl használatával megy ki. Így lehet például a céges levélcímet és a magánlevélcímet ugyanabban a levelezőben használni.

```
fcc-hook . +fcc-hook .
```

Minden Muttból küldött levelet tárolás céljából a „sent-mail” levelesládába fűzzön le. Ez a szolgáltatás nagyon hasznos lehet vitás levelek utólagos előkeresésekor.

```
set pgp_verify_sig=yes
set pgp_replyencrypt
set pgp_replysign
set pgp_timeout=600
```

Ezek a sorok tartalmazzák a PGP- és GPG-programokra vonatkozó meghatározást, amelyben jelszavunk időbeni érvényességét rögzítjük, továbbá ha valaki eleve titkosított levelet ír nekünk, akkor véletlenül se fordulhasson elő, hogy általános levélként továbbítjuk, illetve segítségével a válasz is önműködően kódolt lesz.

```
send-hook kisvakond 'set pgp_autosign=yes; set
pgp_autoencrypt=yes'
```

Amennyiben valakivel olyan levelezési kapcsolatban állunk, hogy minden egyes levelünk titkosított küldését követeli meg, érdemes használni, hiszen így a Kisvakond becenevű illetőnek úgy írhatunk levelet, hogy a rendszer elküldés előtt rákérdez GPG-jelszavunkra.

Az új 1.1-es Mutt-rendszerek GPG-kezeléséhez további sorok szükségesek, mivel azonban ezek igen hosszú bejegyzések és igazából csak a GPG számára értelmezhetők, érdemes őket a GPG-leírásból bemásolni (körülbelül 10–15 sor).

A Mutt innen olvassa be a címjegyzékünket, amelyet az adott formátumban kell elhelyezni. Ha a levelezőprogramban egy levélen állva megnyomjuk az A billentyűt, a címzett nevét és címét a .muttrc-hez fűzi.

A Debian Woodyban található 1.3.20-1-es Mutt már a magyar nyelvet is támogatja.

Procmail

A Mutt összes szolgáltatásának kényelmes eléréséhez érdemes a Procmail segédprogramot használni. Ez olyan alkalmazás, amely a levelezőkiszolgálótól átveszi a leveleket, amelyeket a megadott feltételek alapján szűrni is tud, majd az általunk megnevezett levelesládába rakja. A Muttnál itt is szükséges egy beállítóállomány, amelyet .procmailrc néven a könyvtárunkba kell helyezni. A .procmailrc csak akkor használható, ha a gépünkre mind levelezőkiszolgáló program, mind a Procmail segédprogram telepítve van.

A .procmailrc

```
SHELL=/bin/bash
```

Ez adja meg, milyen héjprogramot használunk.

```
MAILDIR=~/mail
```

A levelek könyvtárának a helye.

```
LOGFILE=$HOME/procmailrc.log
```

A könnyebb kezelhetőség érdekében érdemes naplózni, hiszen ez alapján megtudhatjuk, hogy egy-egy letöltéskor mely levelesládánkba hány levél érkezett.

```
:0
* ^(From|Cc|To|Reply\-To):.*haragosom@domainje.hu
/dev/null
```

Egy Procmail-bejegyzés általában három sorból áll, a :0 határozza meg, hogy egy szűrési feltétel fog következni, a * után pedig felsoroljuk, hogy a levél milyen mezőjében mit szeretnénk szűrni. Jelen esetben, ha a haragosom@domainje.hu-tól kapunk levelet vagy neki küldték, önműködően a nagy „Unix-iktatóba” kerül.

```
:0
* ^Reply-To: lme@lists.linux.hu
$MAILDIR/lme
```

A formai követelmények ugyanazok, csak jelen esetben a levelet nem eldobjuk, hanem levelesládába rakjuk.

```
:0 wc
* .*
$MAILDIR/mentes
```

Ha ez az utolsó sor a .procmailrc-ben, akkor minden bejövő levélből az irattárba is kerül, és az előtte meghatározott szűrésekből értelemszerűen nem kerül mentésre, csak ami túlhaladt azokon a szűrési feltételeken. Ez hasznos lehet, mert így a különböző levelezési listákat külön tárolhatjuk, valamint ***mentés*** nevű levelesládánkba csak a fontos levelek kerülnek. Mivel másolatról van szó, a levél a Postafiókba is eljut.

```
:0
* .*
$MAILDIR/Inbox
```

Ezzel a Postafiókot adjuk meg.

További hasznos segédprogram még a Procmailhez tartozó Mailstat , amely a procmail.log-ot a 2. listán (20. CD Magazin/Mutt) látható módon képes kiértékelni.

Természetesen mind a Procmail, mind a Mutt beállításai a fentieken kívül még számos kedvező lehetőséget nyújtanak. A bemutatott beállítások segítségével jól használható levelezést alakíthatunk ki, amelyet szinte a végtelenségig tovább lehet fejleszteni.

Varga S. Csaba
(guska@guska.hu) A 1.1-es Slackware óta linuxozik. Kedvtelései közé tartozik a fotózás és Linux telepítése a PDA-kra. Legszívesebben a Gerecsében túrázik. Barátjával, *Somogyi (Jerry) Péterrel* együtt írta e cikket.

A Courier-IMAP

Írásunk vizsgálódásának a tárgya az egyik leghatékonyabb IMAP-kiszolgáló, melynek bemutatjuk a telepítését és a beállításait is.

Olvasóink választása a múlt havi szavazáson a Cyrus ellenében a Courier-IMAP-ra esett. Az okokat most nem vizsgálom, de megígérem, a jövő hónapban a Cyrus is sorra kerül.

Vizsgáljuk meg a Courier tulajdonságait, sajátosságait. A szokványos IMAP-kiszolgálóktól annyiban tér el, hogy csak és kizárólag a ***Dan Bernstein*** által bevezetett `Maildir` levéltárolási formát, illetve a saját formátumát, a `Maildir++`-t támogatja. A `Maildir` formátum feltételezi, hogy a beérkezett levelek külön állományban (fájlban) kerültek tárolásra a felhasználó saját könyvtára alatt található ***Maildir*** könyvtárban. Az új levelek a *~/Maildir/new/*, az olvasottak pedig a *~/Maildir/cur/* könyvtár alatt vannak. A *~/Maildir/tmp/* könyvtár egy köztes könyvtár, amelyben a levél beérkezésekor addig marad egy biztonsági másolat, ameddig nem biztos, hogy sikerült mentenünk a *new/* könyvtár alá. A könyvtárat a kiszolgáló akkor is használja, ha az olvasott levelet a *new/* könyvtár alól mozgatja át a *cur/* alá. A `Maildir++` annyiban különbözik az eredeti formától, hogy támogatja a tárkorlátokat (quota). A tárkorlátok segítségével szabályozhatjuk, hogy egy felhasználó mekkora mennyiségű adatot tárolhat a rendszeren. Az általános tárkorlát, a felhasználók által birtokolt összes állományra vonatkozik. A `Maildir++` beállatása csak a levelekre vonatkozik, azonban ne felejtsük el: ha az általános tárkorlát kisebb, mint a levelezéshez használt, természetesen az fog érvényesülni. A `Maildir++` formátum tárkorlátja nemcsak a tárolható levelek legnagyobb méretére vonatkozhat, hanem beállítható legnagyobb mennyiségük is, tehát megadhatjuk, hogy egy felhasználó tíz megabájttal avagy kétszáz levéllel rendelkezhet.

Másik kellemes tulajdonsága, hogy szinte bármilyen formában képes azonosítani a felhasználót. A teljes támogatottsággal bíró azonosítási formák között szerepel az `authpwd` a felhasználói adatbázis (*/etc/passwd*), valamint az `authshadow` a */etc/shadow* alapján. Az `authpam` az összes felületen tud azonosítani, mely a PAM-on keresztül lehetséges, például MySQL, LDAP, shadow és egyszerű jelszóállomány, Samba jelszóállomány. Egyéb támogatott formák (ezek közül is csak az érdekesebbek):

- `authldap` – az LDAP-kiszolgálóval való kapcsolattartáshoz,
- `authmysql` – azok számára ajánlom, akik szeretik a kihívásokat, működésében akad ugyanis egy-két „meglepetés”.

Ezenkívül több azonosítási formát is támogat, közülük az `authvchkpw` említésre méltó - ez szükséges ahhoz, hogy a `vpopmail` programot használhassuk (erről a cikk folyamán még szót ejtünk). Bevezetett egy saját formát is, a `userdb`-t, melyet csak a külön csomagként telepíthető `maildrop` program támogatja. A `maildrop` program is a Courier-kiszolgáló része és hasonlít a `procmail`-hez, de szűrőnyelvéhez képest a `procmail` egyszerűnek nevezhető.

Az azonosítási formák közül többet is használhatunk majd. Mikor hasznos ez a számunkra? Ha például felhasználók találhatók a rendszeren, de virtuális felhasználókkal is rendelkezünk, akiket például LDAP-adatbázisban tárolunk. Ekkor először lekérdezzük a rendszeren található „normál” felhasználókat, majd ha nem találunk a címzettnek megfelelő felhasználót, áttérünk a következő modulra. Erre a beállításnál példát is láthatunk majd.

Egyéb előnyei:

- SSL-protokoll támogatása, amennyiben az OpenSSL csomag telepítve van,
- IPv6-támogatás,
- megosztható levelező könyvtárak (erről lásd később),
- POP3-kiszolgáló (ez igen erősen korlátozott, lásd még lentebb).

Az azonosítási eljárások programjainak jellemzői

MySQL

Az SQL-alapú programok mindig is hatékonynak és méretezhetőnek számítottak. Az SQL könnyen elsajátítható, nagyon hatékony lekérdező nyelv. Az adatbázist úgynevezett táblák alkotják, melyekből lekérdezésekkel nyerjük ki az adatokat. A Courier IMAP-modulja is hasonló lekérdezést hajt végre a megadott táblákon. A MySQL az egyik legnépszerűbb nyílt forráskódú adatbázis-kezelő, mely főleg a gyorsaságáról híres, legnagyobb ellenfele pedig a PostgreSQL (lásd még augusztusi számunk 40. oldalát).

LDAP

Az LDAP – azaz Lightweight Direcory Access Protocol, a kifejezést körülbelül „Könnyűsúlyú Könyvtárelérési Protokoll”-ként fordíthatnánk. Nagy mennyiségű adat kinyerésére találták ki, a hangsúly tehát az adatok olvasásán, nem pedig az írásán van. Az adatok elrendezése fára hasonlít, alkalmazásával ugyanis a jellegzetes faszerkezetet láthatjuk viszont. Nyílt forrású megoldást az OpenLDAP biztosít a számunkra. Az eljárás nagyon hatékony, de a kezelését nehéz elsajátítani, és sajnos nem létezik könnyen kezelhető LDAP-szerkesztő, illetve kezelőfelület. Az iPlanet kereskedelmi termék, és ez a leghatékonyabb megoldás, mert jól használható kezelőfelülettel rendelkezik. A Novell NDS-e, valamint a Microsoft Active Directoryja is az LDAP alapjául szolgáló X.500-as szabvány leszármazottja.

PAM

Ha egy szolgáltatás megírásához PAM-alapú azonosítást végrehajtó modult használnak, a fejlesztő jókora terhet vesz le a felhasználók válláról. Ami ugyanis a Pluggable Authentication Module-on keresztül tud azonosítási adatokat közölni a rendszerrel, az mindent fel tud használni, amihez csak modult írtak. A legfontosabb modulok: `mysql`, `postgresql`, `ldap`, `shadow`, `passwd`.
A PAM lényege, hogy a lekérdezést végző programnak nem kell rendszergazdai jogosultság – például a shadow fájl lekérdezéséhez, ezt ugyanis megteszi helyette a modul.

Telepítés csomagkezelővel

A csomagkezelővel történő telepítést a Debian GNU/Linux Woody változatán követjük végig. Figyelem, ez a változat még kipróbálás alatt áll! (Bár szerintem már megfelelő üzembiztonsággal bír – a szerző.) A szolgáltatások teljes körű használatához a következő csomagok szükségesek:

`courier-base` – ez az alapcsomag, okvetlenül szükséges telepíteni;
`courier-imap` – az IMAP-démon;
`courier-ldap` – csak akkor telepítsük, ha LDAP-alapú azonosítást szeretnénk;
`courier-authmysql` – MySQL-alapú azonosítást tesz lehetővé;
`courier-pop` – a Courier POP3-démona;
`courier-ssl` – ha a kiszolgáló és az ügyfélgépek között titkosított adatátvitelt szeretnénk, ezt a csomagot is telepítsük;
`courier-imap-ssl` – az IMAP-démon titkosított adatcsatornán közvetítő változata;
`courier-pop-ssl` – a POP3-démon SSL-en keresztül kommunikáló változata.

A telepítés legegyszerűbb módja, ha kiadjuk az alábbi parancsot:

```
apt-get install courier-base courier-imap
➥courier-ldap courier-authmysql courier-ssl
➥courier-imap-ssl courier-pop courier-pop-ssl
```

Ez minden Courier-csomagot feltesz számunkra, és amennyiben nem lennének fent, az SSL-hez szükséges OpenSSL csomagokat is.

Telepítés forráskódból

A legújabb forráskódot a ➲ http://www.courier-mta.org/download.php#imap címről tölthetjük le. A jelenlegi csomag a `courier-imap-1.3.11.tar.gz`. Figyelem, hiába szoktuk meg, a tömörített állományt most nem a */usr/local/src* alá kell kibontani, egy egyszerű felhasználó könyvtárába csomagoljuk ki. A Courier ugyanis megköveteli, hogy a `configure` parancsfájlt és a `make` parancsot ne rendszergazdai jogokkal hajtsuk végre:

```
ago@bp:~[7]$ tar xvfz courier-imap-1.3.11.tar.gz
courier-imap-1.3.11/
courier-imap-1.3.11/Makefile.in
courier-imap-1.3.11/README
courier-imap-1.3.11/AUTHORS
...
courier-imap-1.3.11/imap/FAQ.html
courier-imap-1.3.11/rpm.release
courier-imap-1.3.11/courier-imap.spec
```

Majd ellenőrizzük a következőket:

- Rendelkezünk-e C++-fordítóval, ugyanis a program néhány részét ezen a nyelven írták, eltérően a kiszolgáló programoknál szokásos C nyelvtől.
- Ne használjuk a `gcc` 3-as változatát, mert nem tudjuk vele lefordítani a forráskódot.
- A GDBM- vagy a DB-csomagot rendszereden elérhetővé kell tenni. Ezek általában telepítve vannak a rendszeren, de a fordításukhoz `header` és `include` fájlok sokaságára is szükségünk lesz. Ha tehát a DB-csomagot csomagkezelővel telepítettük a rendszerre, tegyük fel a `dev` vagy `devel` végződésű csomagokat is. Debian-rendszeren ezek `libgdbmg1-dev` és `libdbx-dev` néven találhatók meg. Az x a változatszámot jelöli, a legutolsó a hármas. Mivel a kód C++-részeket is tartalmaz, a `libdb3++` csomag telepítésére is szükségünk lesz.
- Telepítsük az OpenSSL csomagjait. A libssl0.9.6 a binárisokhoz, a libssl-dev pedig az `include` és `header` fájlokhoz szükséges, ha a kiszolgáló SSL-képességeit szeretnénk használni. Amennyiben a titkosított kapcsolaton történő levelezésre nem tartunk igényt, szükségtelen a telepítésük.
- Az LDAP- és a MySQL-azonosítás alkalmazásához szintén kellenek majd a hozzájuk tartozó fejlesztői csomagok. Figyelem, a Courier-IMAP az OpenLDAP 2-es változatával nem tud együttműködni! Erre a célra használjuk a kiszolgáló 1.2.11-es változatát.

Ha minden készen áll, lépjünk át a forrás könyvtárába, és adjuk ki a következő parancsot:

```
ago@bp:~[9]$ ./configure -help
```

Ez a parancs felvilágosítást ad a lehetséges kapcsolók egy részéről. A forrás könyvtárában található ***INSTALL*** fájl bővebb tájékoztatást nyújt, amit mindig nézzünk át, ha új változatot telepítünk a rendszerre, mert lehet, hogy további lehetőségekkel bővültek. A fontosabb kapcsolók az alábbiak:

`--enable-unicode`
Hatására a Courier nemcsak a nyugat-európai országok betűkészletével íródott üzenetek között tud keresni, hanem az összes általa ismert karakterkészletet felhasználja. Ha nem kívánjuk az összes nyelvet támogatni, külön is megadhatók azok a készletek, amelyeket használni szeretnénk, például `--enable-unicode=iso-8859-1,iso-8859-2`.
Ezzel a nyugat- és a közép-európai karakterkészlet-támogatást fogja a kiszolgálóba fordítani.

`--with-db=db`
Ha ezt a kapcsolót használjuk, a Courier az alapértelmezett GDBM-könyvtárak helyett a BSD-s DB-könyvtárakat fogja használni.

`--without-`*`modulneve`*
A *`modulneve`* helyére azt az azonosítási modult helyettesítsük be, amelyet nem szeretnénk, ha lefordítana – ezzel is csökkenthetjük rendszerünk méretét.

`--enable-workarounds-for-imap-client-bugs`
Ha felhasználóink a Netscape Messengert vagy a StarOffice levelezőjét használják, hasznos lehet ez a kapcsoló, mert hibáik kiküszöbölésére tartalmaz megoldásokat. A Netscape 6-os változatában található levelező már jól működik, a hibái a 4-es sorozatban levőket sem érintik olyan súlyosan, hogy indokolt lenne élni e kapcsoló használatával. Felhívnám a figyelmet, hogy használata számos jól működő rendszernek csak árt.
Adjuk ki tehát az alábbi parancsot (ez csak példa):

```
./configure --enable-unicode
```

Így már az összes karakterkészletet támogatjuk. Ezek nem foglalnak olyan sok helyet, érdemes a támogatást a programba fordítani. Ne ijedjünk meg, ha a `configure` parancsállományt többször egymás után lefutni látjuk! Mivel a Courier-IMAP teljesen moduláris, a parancsfájlt az összes modulra végrehajtja.
Ha minden rendben lefutott, a `make` parancsot még mindig nem rendszergazdaként adjuk ki. Ugyanúgy, mint az előbb, a modulok itt is külön fordulnak le bináris állománnyá.
Amennyiben a következő parancsot kiadjuk, ellenőrizhetjük a fordítás sikerességét:

```
make check
```

Ha lefuttattuk és a rendszer nem jelzett hibát, következhet a telepítés. Váltsunk át rendszergazdai jogosultságra:

```
ago@bp.~[12]$ su -
Password:
root#
```

Megjegyzés: amennyiben nem szeretnénk a rendszergazdai jel-

szót kiadni, de bizonyos felhasználóknak jogosultságokat vagy akár teljes jogosultságokat akarunk juttatni, használjuk a `sudo` csomagot. Segítségével a jogosultságok finomhangolhatók.
Adjuk ki az alábbi parancsot:

```
root# make install
```

Ekkor az összes segédprogram és súgóoldal a */usr/lib/courier-imap* könyvtár alá fog kerülni. Ha a */etc/pam.d/* könyvtár létezik, a telepítő létrehozza benne a */etc/pam.d/pop3* és */etc/pam.d/imap* fájlokat. Figyelem, ez felülírja az ott található már létező fájlokat!
A következő parancs, amit ki kell adnunk (még mindig rendszergazdaként):

```
root# make install-configure
```

Ez a beállítófájlokat a */usr/lib/courier-imap/etc* könyvtár alá telepíti. Most a beállítások következnek: miután rengeteg létezik, és az egészen különleges beállítások akár egy egész újságot is kitölthetnének, csak a legalapvetőbb dolgokkal foglalkozunk, mert a Courier így is sok szolgáltatást nyújt.
A program elindításához egy háttérben futó alkalmazást, az `authdaemond`-t kell elindítanunk, ami úgy működik, mint egy proxy. Feladata az azonosítást végző programokkal a kapcsolat fenntartása, így a következő lekérdezésnél kevesebb erőforrást használ, ezáltal az gyorsabb lesz. Főleg az olyan azonosítási típusoknál érdekes ez, mint az SQL, illetve az LDAP. A következőképpen tudjuk elindítani:

```
/usr/lib/courier-imap/libexec/authlib/
➥authdaemond start
```

Érdemes egy indítófájlba beletenni, abba, amelyet a Courier-IMAP indítására használunk. Szemléltetésül azt használjuk fel, amelyet a Debian-csomagban találhatunk (ez el is indítja az `authdaemond`-t), természetesen az útvonalak módosításával (a parancsfájl a CD-mellékleten is megtalálható). Mi is írhatunk nagyon egyszerű parancsfájlt, amely leállítja a kiszolgálót, de érdemes ragaszkodni a már jól bevált módszerhez, hogy ne találjuk fel még egyszer a spanyolviaszt.
Az azonosítások közül – rugalmasságának köszönhetően – a PAM-ot fogjuk használni. Nyissuk meg a */usr/lib/courier-imap/etc/authdaemonrc* fájlt és ellenőrizzük, hogy a következők szerepelnek-e benne:

```
authmodulelist="authpam authcustom authcram
➥authuserdb authshadow authpwd"
authmodulelistorig="authcustom authcram
➥authuserdb authshadow authpwd"
daemons=5
authdaemonvar=/usr/lib/courier-imap/var/
➥authdaemon
```

Ebből az első a legfontosabb, mert ez adja meg, hogy a kiszolgáló az azonosítási módszereket milyen sorrendben próbálja ki. Ha egy modult kiveszünk a listából, az érvénytelen. Fontos tehát, hogy legelöl a `authpam` álljon. A `daemons` az egyszerre futó démonok számát adja meg. Amennyiben a kiszolgálás sebessége lelassulna, növeljük a démonok számát. A többi sort lehetőleg ne módosítsuk.
Ezt követően a */etc/pam.d/imap* fájlt szerkesztjük, melynek a következőképpen kell kinéznie:

```
auth        required      pam_unix_auth.so
account     required      pam_unix_acct.so
password    required      pam_unix_passwd.so
session     required      pam_unix_session.so
```

A POP3-kiszolgáló használatához a */etc/pam.d/pop3* szükséges, tartalma teljesen megegyezik az előzővel. Vigyázzunk: a PAM-modulok nevei terjesztésenként változók lehetnek, azonban elég könnyű rájönni, melyik is pontosan.
A kiszolgálót a következő paranccsal tudjuk elindítani:

```
/usr/lib/courier-imap/libexec/imapd.rc start
```

Leállítására az alábbi utasítás szolgál:

```
/usr/lib/courier-imap/libexec/imapd.rc stop
```

Könnyedén készíthetünk tehát saját parancsfájlt (bár még mindig a módosított útvonalakkal elkészítettet ajánlom).
A parancsfájl neve *courier.start* legyen, lelőhelye a */etc/init.d* könyvtár, a tartalma pedig:

```
#!/bin/sh
echo -n Courier-IMAP inditasa:
/usr/lib/courier-imap/libexec/imapd.rc start
echo
echo -n Courier-POP3 inditasa:
/usr/lib/courier-imap/libexec/pop3d.rc start
echo
```

Ez semmilyen ellenőrzést nem végez, de elindítja az IMAP- és POP3-kiszolgálókat. Figyelem, nagyon fontos tudnunk, hogy a POP3-kiszolgáló a két protokoll különbözősége miatt csak az Inboxot, vagyis az alapértelmezett tárolóhelyet látja. Sokszor nehéz a felhasználóknak elmagyarázni, miért is van így, de tapasztalatom szerint, ha nem is szeretik, de elfogadják a tényt.
A következő szolgáltatások leállításához pedig az alábbit használjuk:

```
#!/bin/sh
echo -n Courier-IMAP leallitasa:
/usr/lib/courier-imap/libexec/imapd.rc stop
echo
echo -n Courier-POP3 leallitasa:
/usr/lib/courier-imap/libexec/pop3d.rc stop
echo
```

Adjuk neki a *courier.stop* nevet és szintén a */etc/init.d* alá mentsük, majd készítsünk egy hivatkozást az alapértelmezett futási szintnek megfelelő könyvtárba. Debian-rendszer használatát feltételezve a következő parancsokat adjuk ki:

```
ln -s /etc/init.d/courier.start
/etc/rc2.d/S20courier
ln -s /etc/init.d/courier.stop /etc/rc1.d/K20courier
ln -s /etc/init.d/courier.stop /etc/rc6.d/K20courier
ln -s /etc/init.d/courier.stop /etc/rc0.d/K20courier
```

Az első sor minden induláskor elindítja majd a szolgáltatást, a többi pedig leállítja azokon a futási szinteken, amelyek sorrendben a következők: hálózat nélküli futás, újraindulás, leállás.
Egyetlen dolog maradt még hátra, annak beállítása, hogy milyen címekről engedje a kapcsolatokat, hány kapcsolatot engedjen stb. A fájl lelőhelye a */usr/lib/courier-imap/etc/imapd*. Tartalma itt már az eddigiekkel kitöltve szerepel.
Ha elkészültünk, használatba is vehetjük új IMAP-kiszolgálónkat. A Courier-IMAP kitűnően együttműködik a Maildirt támogató kiszolgálókkal, tehát a Postfix, az Exim vagy a Qmail hármassal, illetve a saját kiszolgálójával.

Deim Ágoston (ago@lsc.hu)
Kedveli a sört, szereti a futást és imádja Szabó Lőrinc verseit. Nem hisz vakon egyik rendszerben sem. Vonzódik a BSD-hez is. Tagja az LME-nek és a MBE-nek. Mottója: a gép nem lehet fontosabb az embernél.

Kapcsolódó címek

- http://www.courier-mta.org/
- http://www.inter7.com/courierimap/
- http://www.courier-mta.org/download.php#imap

PoV-Ray ismeretek (4. rész)

Elérkezett az ideje, hogy elsajátítsuk a különféle anyagok létrehozásának az alapjait, majd ismereteink birtokában bonyolult felületi mintázatokat hozzunk létre.

A PoV-Rayben erre számos lehetőségünk nyílik: az előre meghatározott anyagok körétől kezdve a saját magunk által készített anyagokig, amelyeket többféle mintázattal is elláthatunk, sőt, ezeket még különféle átlátszósággal is használhatjuk.

A korábbi példák alapján ismeretes, hogy egy tárgy anyagát a `texture` kulcsszóval határozhatjuk meg, melynek általános formája az ***1. listán*** látható.

Ezáltal az anyag három legfontosabb tulajdonságát adjuk meg. Elsőként a `pigment` kulcsszó beírásával az anyag színét közöljük a PoV-Rayjel, és az esetleges a színváltozásról, vagyis a mintázatról is tájékoztatjuk. A második fő tulajdonság a `normal` kulcsszóval meghatározott felületi érdesség, amely a tárgy felületén kiemelkedések és horpadások formájában jelentkezik. A harmadik jellemző, a `finish` kulcsszó szolgál a fényesség és fényvisszaverő képesség meghatározására.

Foglalkozzunk részletekbe menően a mintázat színének meghatározását segítő lehetőségekkel! Az anyag színét mindenképpen érdemes megadni, amit a következő módon tehetünk meg:

```
pigment {
   color rgb <V R S, Z LD, K K>
}
vagy
pigment {
   color rgbt <V R S, Z LD, K K, `TL`TSZ S`G>
}
vagy
pigment {
   color rgbf <V R S, Z LD, K K, FILTER_ RT K>
}
```

Ezzel egyszerű színeket határoztunk meg, amelyeket az `rgb` helyett megadott `rgbt` kulcsszóval és egy negyedik vektorérték begépelésével akár átlátszóvá is varázsolhatunk. Ilyenkor a fenti listán látható PoV-forrásunkban az utasítás második formáját kell használnunk. Ha a listában szereplő harmadik formát alkalmazzuk, a tárgy a `V R S, Z LD, K K` színekkel leírt színszűrőként fog viselkedni.

Természetesen a színezetek nem csak egyszerű színt tartalmazhatnak: számos matematikai kifejezéssel létrehozható minta található a PoV-Rayben, melyek közül tekintsünk át néhányat. A legegyszerűbb mintázat a `color_map` nevet kapta és jellemzően színátmenetek meghatározására alkalmas. A következő listán formailag helyes meghatározása szerepel:

```
pigment {
       gradient_x
       color_map {
              [ MAGASS`G_0 color SZ"N_0 ]
              [ MAGASS`G_1 color SZ"N_1 ]
              ...
              ...
              [ MAGASS`G_N color SZ"N_N ]
       }
}
```

1. lista A „texture” kulcsszó általános formája

```
texture {
       pigment {
              .....
              .....
              }
       normal {
              .....
              .....
              }
       finish {
              .....
              .....
              }
}
```

2. lista A checker kulcsszó alkalmazása

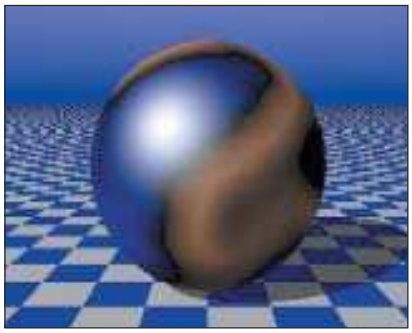
```
#include "colors.inc"
#include "woods.inc"
#include "stones.inc"

pigment {
       checker Red, Blue
}

pigment {
       checker {
              pigment { Jade } ,
              pigment { Black_Marble }
       }
}

texture {
       checker {
              texture { T_Wood_3A } ,
              texture { Stone2 }
       }
}
```

A színátmenet típusa függőleges irányú is lehet, ekkor a `gradient_x` helyett `gradient_y` kulcsszót kell használnunk. A listában szereplő `MAGASS`G_0...MAGASS`G_N` értékek adják meg, hogy az egyes színek miként oszlanak meg egy egységnyi magasságú kockára vetítve. Ha például az átmeneten három színt szeretnénk elhelyezni, de úgy, hogy egyenlő arányban foglaljanak helyet, a három `MAGASS`G-ØrtØk` a következőképpen alakul:

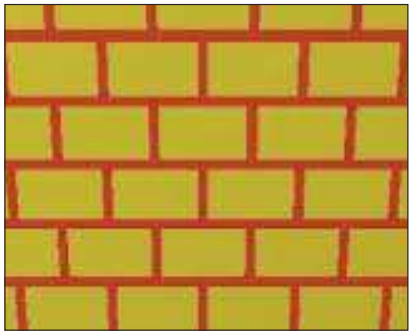
`MAGASS`G_0=0,33, MAGASS`G_1=0,66, MAGASS`G_2=1`

A létrehozott színátmenetet a későbbiek folyamán a már megismert átalakulást előidéző utasítások segítségével elforgathatjuk és átméretezhetjük, melyeket még a `texture` kulcsszót lezáró `'}'` karakter elé kell elhelyeznünk. Az átalakítási lehetőségeknek köszönhetően azt is megvalósíthatjuk, hogy egy vízszintes színátmenetet határozzunk meg és 90 fokkal elfordítsuk. Ez kitűnően helyettesítheti a függőleges színátmenetet, ezáltal jól szemlélteti a sokrétű PoV-Ray szolgáltatások és az alkotó emberi elme együttműködését.

A következő kiemelt mintázat a `bozo` nevet kapta. A tárgyon finom zajszerű mintát hoz létre, amelyet füst és felhők létrehozására használhatunk. A PoV-Ray előre elkészített felhőmintái közül a legtöbbet szintén e mintázat segítségével alkottak meg, ezeket a skies.inc fájl tartalmazza. A szóban forgó mintázatokat leginkább úgy képzelhetjük el, mintha egy szobában különleges szemüvegen át szemlélnénk a hőmérséklet-eloszlást. A szoba egymáshoz közel eső részeiben a hőmérséklet nem változik jelentősen, de az egymástól távolabb eső részeken lassú, véletlenszerű módosulást tapasztalhatunk. Az ***1. képen*** `bozo` mintázat által létrehozott alkotást láthatunk.



1. kép Kis munkával felhő is lehet belőle...



2. kép Közeli a téglafalról

Lássuk a téglafalak létrehozására alkalmas `brick`-mintázat meghatározását és az elkészült téglafalat!

```
pigment {
        brick COLOR_1, COLOR_2
        brick_size <VEKTOR>
        mortar VAL S
}
```

A fenti értékeknél meg kell adnunk a téglák és a közöttük lévő kötőanyag színét, majd a téglák méretét. A téglák alapértelmezett mérete 8×3×4,5 egység, a kötőanyag vastagsága – amit a `mortar` kulcsszót követő valós szám ad meg – pedig 0,5 egység.

A `checker` kulcsszóval a ***1. kép*** alján láthatóhoz hasonló mintázatot hozhatunk létre. Kifejezetten sakktáblamintázat létrehozására alkalmas: kétszínű pepitamintát alakíthatunk ki a segítségével, mely természetesen egyes négyzeteiben nemcsak színeket tartalmazhat, hanem további mintákat és anyagokat is (***2. lista***). Kísérletező kedvű olvasóink számára listáink kínálnak útmutatót. Ne feledjük, a példákban olyan előre meghatározott anyagokat és színeket használtunk, amelyek megfelelő beilleszthető fájlokat igényelnek.

Érdekes padló- és falmintákat varázsolhatunk a `crackle` használatával. Ez a mintázat leginkább egymás mellé helyezett, véletlenszerűen kialakított sokszögek csoportosulására hasonlít. A `turbulence` kulcsszót követő valós számmal a mintázatoknak örvényszerű hatást adhatunk, vagy ha helyette a `scale`-átalakítással növeljük az alapminta méretét, kőhöz hasonló mintázathoz jutunk, amit falak és padlók készítésére is felhasználhatunk. A méretet kicsire állítva kerámiára emlékeztető mintázat hozható létre, amit a `color_map`, a `pigment_map` (amelyet ugyanúgy határozhatunk meg, mint a `color_map`-et, azzal a különbséggel, hogy a `color` helyett a `pigment` kulcsszót, a `color_map` helyett pedig a `pigment_map` kifejezést használjuk) és a `texture_map` kulcsszavakkal együtt használhatunk. Ez szinte végtelen számú lehetőséget teremt számunkra a falak és padlók mintázatának meghatározásakor, továbbá az összes olyan tárgy anyagának életre hívása során, amelyeken apró, változatos mintákat szeretnénk látni. Fémek kialakításakor érhetünk el érdekes hatásokat a `dents` minta segítségével, amellyel a felületen fizikai hatások, például kalapácsütés nyomait jeleníthetjük meg. Nem használhatjuk azonban a felület normálvektorainak módosítására (az anyagmeghatározás `normal` részében), mert a felület normálvektorainak módosítására különleges eljárást használ. Ha azonban a `pigment`-résznél alkalmazzuk, csaknem ugyanolyan hatást érhetünk el vele, mintha a `normal`-nál adtuk volna meg, vagyis az előbbi hátrány nem jelent valódi veszteséget. A `color_map`, `pigment_map` és `texture_map` módosítókkal együtt alkalmazhatjuk. Utóbbit szintén a `color_map`-hez hasonlóan adhatjuk meg, a különbség csupán annyi, hogy nem színeket ad meg a különböző magasságokhoz hozzárendelve, hanem kész anyagokat.

A normálvektorok módosításánál a `granite`-mintázat alkalmazásával képezhetünk az utakon látható aszfalthoz vagy kőhöz hasonló anyagot, mely sziklák vagy öreg kőépítmények anyagául is szolgálhat. Ez a minta szintén együtt használható `color_map`-pel, `pigment_map`-pel és `texture_map`-pel.

A `leopard`-mintázattal kedvenc tárgyainknak megközelítőleg kör alakú foltos mintát kölcsönözhetünk (ugyancsak használható a `color_map`-pel, a `pigment_map`-pel és a `texture_map`-pel módosított formában). Érdekes hatásokat érhetünk el ezen a ponton a `turbulence` alkalmazásával, mellyel valódi leopárdbundát is modellezhetünk.

A `ripples` mintával vízhullámokat készíthetünk. A PoV-Ray egy-egy egység oldalú kockában tíz hullámforrást hoz létre, ahonnan a `frequency`-módosítóval meghatározott rezgésszámú hullámok indulnak ki. A PoV-Raynek a hullámforrások legnagyobb számát a következő sorokkal adhatjuk meg:

```
global_setting { number_of_waves FORR`SOK_SZ`MA }
```

A `phase`-módosító segítségével a későbbiekben bemutatott módon hozhatjuk mozgásba a hullámokat, animációt készítve pedig további látványos hatásokat teremthetünk. Létezik a PoV-Raynek olyan kiegészítése is, amellyel a hullámzó vízfelületen megcsillanó fénysugarakat is megjeleníthetjük. A kiegészítő csomagot az animáció készítéssel foglalkozó részben mutatom be. A hullámzó víz elkészítésénél használt vehetjük a méretezhetőségnek és az eltolásnak, mellyel a hullámforrások egymástól és az objektum határától való távolságát befolyásolhatjuk. A hullámok létrehozásához használatos kulcsszavunkat szintén érdemes `color_map`, `pigment_map` és `texture_map` alkalmazásával is kipróbálni.

Lassan a végére érünk a fontosabb mintázatok taglalásának, most a `spiral1` és a `spiral2` tanulmányozásához jutottunk el. A spirál első változata szabályosabb, mint a második, de mindkettőt a következő formában adhatjuk meg:

```
pigment {
        spiral1 | spiral2 SPIR`LKAROK_SZ`MA
}
```

A spirálokat szintén használhatjuk `color_map`-pel, `pigment_map`-pel és `texture_map`-pel együtt.
Az utolsó mintázat, amit még be szeretnék mutatni, a `wrinkle` nevet kapta és leginkább a gránithoz hasonlítható, azzal a különbséggel, hogy az átmenetei nem annyira finomak, inkább élesebbek. A mintázat gyűrött celofán vagy fólia megjelenítésére használható, viszont nem alkalmazhatjuk normálvektorok módosítására. A `color_map`, `pigment_map` és a `texture_map` a `wrinkle`-lel is kényelmesen együtt használható.

3. kép Szoba szőnyeg nélkül

A fentebb leírtakban találkozhattunk már a `frequency` és a `phase` kulcsszavakkal, most ezek működését szeretném jobban megvilágítani.
A `frequency` kulcsszóval határozhatjuk meg a mintázat ismétlődésének sűrűségét: ha például ez az érték 2, a mintát alkotó színek (mind a festékszemcsék, mind az anyagok, ha a `pigment_map`-pel vagy a `texture_map`-pel használjuk) kétszer fognak megjelenni ugyanakkora távolságon belül, mint amit az 1-es rezgésszámértéknél láthatunk. A `phase` kulcsszó használatakor a színek eltolását határozzuk meg, ami szintén felhasználható festékszemcsék és anyagok eltolására. Hasznunkra válik, ha nagyon szép átmenetet készítettünk, viszont használatba vételekor azt látjuk, hogy a tárgyon rossz helyen kezdődik a választott szín. Ilyenkor a színátmenetet alkotó összes színt átrendezhetjük, de elegánsabb megoldás, ha a kezdő állapotot változtatjuk meg a `phase` kulcsszó és a megfelelő eltolási érték megválasztásával. Természetesen, ahogyan a fentiekben már utaltam rá, e parancs segítségével animált anyagok és mintázatok is készíthetők. Ez a két kulcsszó a következő mintázatokkal együtt nem használható: `checker`, `brick`, `hexagon`, `image_map`, `bump_map`, `material_map`, valamint a következő normálvektor-módosítókkal sem: `bumps`, `dents`, `quilted`, `wrinkles`, hiszen ez esetben értelmetlen rezgésszámról és fáziseltérésről beszélni.
Végezetül szeretném még bemutatni a már oly sokat emlegetett `turbulence` utasítás értelmét és használatát. Segítségével örvényszerű hatást adhatunk a mintázathoz, vagyis egy kis „szabálytalanságot” vihetünk a művünkbe. A következő formában használhatjuk, de nem minden értékét kötelező meghatározni:

```
pigment {
        ...
        turbulence <VEKTOR>
        octave VAL S
        omega VAL S
        lambda VAL S
}
```

A `VEKTOR`-ral határozzuk meg, hogy az egyes tengelyek mentén milyen erős hatást szeretnénk elérni, míg az `octave` (értéke 1–10-ig terjedhet) adja meg, hogy a PoV-Ray a rendezetlenséget hány lépésből számítsa ki. A kisebb értékek gyorsabb számolást és ezzel együtt kevésbé részletes mintát eredményeznek, míg a nagyobb értékek használatával lassabban készül el a képünk, viszont a mintázat változása finomabb lesz. Ennek az értéknek az alapértelmezése 6, amely kellőképpen nagy érték. A `lambda` értékkel adjuk meg, hogy a véletlenszerű elmozdulás mekkora legyen az egyes lépések között. Nagyobb értékeknél a kapott mintázatban nagyobb összefüggő örvények keletkeznek, míg kisebb értéket használva a mintázat rendezetlenebbnek tűnik. Az `omega` kulcsszóval határozzuk meg, hogy a változás a pillanatnyi számítási lépésnél az előző oktávhoz képest milyen mértékű legyen. Alapértelmezésben ennek értéke 0,5, vagyis az előző lépésnél számítottakhoz képest az örvényszerű mintaváltozás a pillanatnyi lépésben a felére zsugorodik. Minél nagyobb ez az érték, annál gyorsabban csökken a változás mértéke. Végül meg kell említenem az `image_map` mintázatot, amellyel tetszőleges képet feszíthetünk a tárgyak felületére. Megadásának módja a következő:

```
image_map {
            F`JLT"PUS "fÆjlnØv.kit"
            M DOS"T K
}
```

A `F`JLT"PUS` gif, tga, iff, ppm, pgm, png lehet. A módosítók közül pedig az egyik legfontosabb a `map_type`, amellyel a feszítés módját határozzuk meg. Ez a következő értékeket veheti fel: 0 (síkszerű), 1 (gömbszerű), 2 (hengeres vetítés), 5 (vetítés tóruszfelületre). A 3-as és a 4-es érték még fenntartott, a jelenlegi PoV-Ray változatban nincs hatásuk.
A másik kiemelendő módosító az `once` – segítségével megakadályozhatjuk, hogy a kép egymás mellett többször jelenjen meg.
A mintázatok bemutatásának végére érve el kell mondanom, hogy azok a mintázatok, amelyeknél külön nem jeleztem, hogy alkalmazhatatlanok normálvektorok módosítására, a következő formában is használhatók:

```
texture {
        normal {
                VALAMILYEN_MINTA
                ...
                ...
        }
}
```

Ezzel elérhetjük, hogy a tárgy felszínén nemcsak a színek és anyagok változnak, hanem a felület érdessége is.
A fentiek alkalmazására jó példát láthatunk a ***3. képen***, melynek forrását a CD-melléklet is tartalmazza.
További jó ötleteket és könnyű alkotást kívánok!

Fábián Zoltán
(dzooli@freemail.hu, dzooli@yahoo.com)
23 éves, jelenleg programozóként dolgozik. Szabadidejében szívesen kirándul, túrázik. Emellett szeret rajzolni, érdekli a 3D grafika és a Linuxszal kapcsolatban minden olyan program és programnyelv, amit még nem ismer vagy nem próbált ki.

XMLC

Reuven az XMLC-vel, az Enhydra alkalmazáskiszolgáló egyik elemével ismertet meg bennünket.

Az utóbbi pár hónapban több módszert is megvizsgáltunk azzal kapcsolatban, hogyan lehetséges kiszolgálóoldali Java segítségével webalkalmazásokat készíteni. Egyszerű servletekkel kezdtünk, végül eljutottunk a JavaServer Pages-hez (JSP). Hogy JSP-nkből eltávolíthassuk a Java-kódot, nekiálltunk JavaBeanst használni, azokat a különleges objektumokat, melyek metódusai lapunkról automatikusan elérhetők lesznek.
Csak addig foglalkoztunk a JavaBeansszel, míg a testreszabott események elő nem kerültek (lásd Linuxvilág, 2001. augusztus, 67–70. oldal). Ezek – JSP-nkben XML-tagoknak és értékeknek álcázott események – egy-egy Java-osztály eljárásaihoz vannak kötve. Más szavakkal tagokat helyezve a JSP-be hatékonyan hívhatunk meg egy vagy több eljárást. A testreszabott tagok és a babok együttes használatával jókora adag Java-kódot távolíthatunk el JSP-programjainkból.
Mire jutottunk? Mint augusztusban láthattuk, a testreszabott eljárások okos használatával saját kis mininyelvet hozhatunk létre ciklusokkal, feltételekkel és változókkal. Saját tagok írásával mentesíthetjük a grafikus tervezőket a Java-használat alól, és jobban elkülöníthetjük a formát és a tartalmat. Gondjaink megoldásához azonban még ez sem elegendő.
Jó megoldás lehet az Enhydra alkalmazáskiszolgáló, amelyről az előző hónapban írtam (szeptemberi Linuxvilág 64. oldal). Az XMLC (vagy XML-fordító) az XML-fájlokat (ideértve a HTML- és XHTML-állományokat is) Java-objektumokká alakítja. Ha az ezekhez az objektumokhoz tartozó eljárásokat hívjuk meg, az éppen készülő HTML-t meg tudjuk változtatni.

XML és XHTML

Az XML – mint azt valószínűleg mindannyian tudjuk – egy bővíthető jelölőnyelv (markup language). Ami évekkel ezelőtt még egyszerű kis szabványnak indult, mára már hatalmas szabvány- és ajánlott szabványgyűjteménnyé nőtte ki magát. Az XML magja azonban mindvégig azonos maradt, így az emberek megegyező formai követelmények alapján készíthetik el a saját jelölőnyelvüket. Az XML nem közvetlen használatra készült; célja, hogy segítségével ki-ki a saját nyelvét készíthesse el. Mivel ezek a nyelvek mind XML-alapúak, jól érthető felépítésük van, amit minden XML-értelmező elfogad. Ráadásul, ha nyelvünkhöz adattípus-meghatározást is készítünk (DTD), az értelmező ellenőrizheti, hogy az elemek és értékek megfelelő formátumban vannak-e.
A HTML és az XML egyaránt a World Wide Web Konzorcium (W3C) szabványa, igen hasonló szerkezettel rendelkeznek, és gyakran veszik őket egy kalap alá. A valóság azonban az, hogy míg a HTML csak egy jelölőnyelv a sok közül, az XML saját jelölőnyelv megalkotására nyújt lehetőséget. Még jellemzőbb, hogy a HTML sokkal lazább felépítéssel rendelkezik, mint az XML, amit nem kis mértékben történelmi tényezőknek köszönhet. A következő kifejezés például helyes HTML-parancs: `<img src="foo.png">`.
Ezzel szemben az XML-alapú nyelvekben minden tagot szigorúan le kell zárnunk, ezért egy XML-dokumentumban az említett példa nem lenne szabályos. Helyesen a következőt kellene írnunk: `<img src="foo.png"/>`.
Hogy a HTML és az XML közti szakadékot áthidalja, a W3C egy XHTML-nek nevezett ajánlást adott ki, mely tulajdonképpen a HTML XML-átirata. Az XHTML használatának meglehetősen sok előnye létezik, melyek közül a legnagyobb talán, hogy HTML-dokumentumainkkal az összes XML-eszköz dolgozni tud.
Természetesen ez egyben azt is jelenti, hogy XHTML-dokumentumaink kicsit formálisabban fognak kinézni, mint a korábban megszokott HTML. Míg a HTML megengedi, hogy hanyagok legyünk, és mindössze egy `<P>`-vel válasszuk el a bekezdéseket, az XHTML sokkal szigorúbb: a bekezdéseket kötelező `<P>`-vel kezdeni és `</P>`-vel befejezni. Az értékeknek kötelezően macskakörmök közé kell kerülniük, amit sokan figyelmen kívül hagynak, amikor egyszerű HTML-ben dolgoznak. Bár a XHTML néha sok gondot okozhat az embernek, a programok terhelését nagymértékben csökkenti – felépítésüket ugyanis szabályosabbá teszi, így könnyebb lesz írni és olvasni őket.
A legnagyobb előny azonban mégis az a tény, hogy az XHTML-dokumentumokat XML-dokumentumnak minősíthetjük.

A DOM

Az XML-dokumentum egy faszerkezet, mely tény ismerősen csenghet azok számára, akik főiskolán tanultak számítástechnikát. A fákkal való munka elméletben figyelemreméltóan könnyű, a gyakorlatban viszont az alkalmazott csatolófelülettől függően időnként trükkösnek bizonyulhat.
Két népszerű felületfüggetlen API létezik az XML-hez: a SAX (Simple API for XML) programot úgy tervezték, hogy XML-adatfolyamokkal tudjon dolgozni, ezáltal kicsi és hatékony maradt. A DOM (Document Object Model) vele ellentétben a felhasználót egyszerre engedi hozzáférni a teljes dokumentumfához, miáltal lehetővé válik, hogy lerövidítsünk vagy megváltoztassunk csomópontokat, beleértve az új csomópontok felvételét és törlését is. Emellett ez azt is jelenti, hogy a teljes dokumentumot a memóriába kell tölteni, mielőtt a DOM segítségével dolgozni kezdhetnénk rajta. Ezek a képességek a SAX-nál hatékonyabbá, ám egyszersmind lassabbá és erőforrás-igényesebbé is teszik.
Az XMLC XML-fájlokat, rendszerint HTML-ben vagy XHTML-ben írt állományokat alakít át olyan Java-osztályokká, amelyek a DOM-fát létrehozzák, illetve módosítják. A megszokott DOM-eljárásokat használhatjuk csomópontok hozzáadására, törlésére és módosítására, megváltoztatva az éppen kimenetre kerülő dokumentumot.
Az XMLC igazán nagy ötlete a HTML `"id"`-értékek használata. Amikor az XMLC fordító egy `id`-értéket talál, készít egy eljárást, melynek segítségével lekérdezhetjük, illetve módosíthatjuk az adott értékben található szöveget. A laptervezők

tehát HTML-ben dolgozhatnak, és az egyes dinamikus szövegmezőket egyedi címkékkel jelölik meg. Amikor a tervezők elkészítették az eredeti HTML-lap vázlatát, egy Java-osztállyá fordítják le (`xmlc`-t használva). A fejlesztők ezután olyan servleteket készíthetnek, amelyek létrehozzák ezt az osztályt, az eljárások segítségével dinamikusan létrehozott tartalommal helyettesíthetik a vázlatszöveg mezőit, és a dokumentumot a felhasználó böngészőjére küldhetik ki.
Az alapötlet az, hogy a tervezőknek nem kell vegyesen szöveges és HTML formátumon dolgozniuk, hanem egyszerűen csak a végleges kimenet vázlatát készítik el. Amíg az `id`-értékek nem változnak meg, a HTML-fájl és a servlet fejlesztése párhuzamosan is folyhat, és sem a tervezőknek, sem a fejlesztőknek nem kell a társaikra várniuk.

Az Enhydra telepítése

Amint azt fentebb említettem, az `xmlc` az Enhydra-alkalmazás kiszolgálórésze. A 3.x-változatú Enhydra már termelésre alkalmasnak mondható, és tartalmaz egy `xmlc`-t, amelyet a legtöbb felhasználó több mint megfelelőnek fog találni. Mivel engem az Enhydrában most leginkább az érdekelt, miképpen tudnám az Enterprise JavaBeanshez (EJB) felhasználni, a 4.x próbaváltozattal dolgoztam, más néven az Enhydra Enterprise-változattal. Időközben valószínűleg az Enhydra Enterprise végső kiadása is elérhető, melyben a webfejlesztők egy nyílt forrású, termelési minőségű, J2EE-megfelelő alkalmazáskiszolgálóhoz juthatnak (még mindig nem jelent meg – a szerk.).
Az `xmlc`-vel való munkához letöltöttem az Enhydra Enterprise próbaváltozatot egy 15,7 MB méretű ***enhydra4.0.tar.gz*** nevű fájl képében. Utóbbit megnyitva gazdag könyvtárkészletet, alkalmazásokat, és leírást találunk az Enhydra alkalmazáskiszolgálóhoz, melyről előző számunkban már írtunk. Ezek nagy részét most figyelmen kívül fogjuk hagyni, és kizárólag az XMLC-re összpontosítunk.
Csaknem a teljes Enhydra héjprogramból hívott Java-osztályokból áll. Annak érdekében, hogy a héjprogramok megtalálják a Java-osztályokat, be kell őket állítani az adott telepítéshez, amit könnyen megtehetünk, ha belépünk az Enhydra könyvtárba (az én rendszeremen ***enhydra4.0***) majd lefuttatjuk a `configure` parancsfájlt:

```
./configure /usr/java/jdk1.3
```

A `configure` alapesetben egyetlen értéket vár: JDK 1.3 telepítésünk könyvtárának a teljes nevét. Bár az Enhydra korábbi változatai (különösen a korábbi Enhydra Enterprise-ok) nem működnek JDK 1.3 alatt, a jelenlegi változatok csak és kizárólag ezzel dolgoznak. Mivel a JDK 1.3 számos előnyös tulajdonsággal rendelkezik, és a Sun támogatja a linuxos változatot, nem rossz ötlet feltelepíteni.
Ha az Enhydrát a */usr/local/enhydra* könyvtáron kívüli helyre telepítettük, előfordulhat, hogy az `ENHYDRA` környezeti változót be kell állítanunk a telepítés könyvtárára.
Az XMLC teljes kihasználásához három különféle .JAR-fájlt kell `CLASSPATH`-unkba helyezni. Mivel a cikk további részeiben az `xmlc` használatán lesz a hangsúly, nem árt, ha most rögtön hozzáadjuk őket a `bash` formai követelményeit használva:

```
export CLASSPATH=$ENHYDRA/lib/xmlc.jar:\
$ENHYDRA/lib/enhydra.jar:\
$ENHYDRA/lib/xmlc-support.jar
```

A hozzám hasonlók bizonyára az Enhydrához kapcsolódókon kívül még számos elemet is a `CLASSPATH` változóban szeretnének tárolni. Nézzük meg, hogyan állítottam be én a `CLASSPATH` változómat!

```
export CLASSPATH=$ENHYDRA/lib/xmlc.jar:\
$ENHYDRA/lib/enhydra.jar:\
$ENHYDRA/lib/xmlc-support.jar:\
$TOMCAT_HOME/classes:\
$TOMCAT_HOME/lib/servlet.jar:\
/usr/share/pgsql/jdbc7.1-1.2.jar
```

Figyeljük meg, hogy rendszeremen az Enhydra .JAR-fájljai a többiek elé helyeztem, így elkerülhettem az ütközés miatt felbukkanó gondokat. Mivel néhány osztályból az Enhydra már a legújabb változattal rendelkezik, például amelyek a DOM-mal dolgoznak, ezeknek elöl kell lenniük.
Megjegyzem, az XMLC-vel való munka nem minden szakaszában szükséges az Enhydra mindhárom .JAR-állománya. Én azonban úgy gondoltam, kényelmesebb az összes lépésben mindet hozzáadni, így a későbbiekben elkerülhetem a kellemetlen meglepetéseket.

Egyszerű HTML-állomány

Most, hogy már minden, az `xmlc`-hez szükséges dolgot feltelepítettünk, próbáljuk is ki egy egyszerű HTML-fájlon:

```
<html>
    <head><title>This is a title</title></head>

    <body>
        <h1>This is a headline.</h1>
        <p id="firstpara">This is a
                          paragraph.</p>
        <img src="foo.gif"/>
        <p>This is a second paragraph.</p>
    </body>
</html>
```

Bár az `xmlc` egyszerű HTML-fájlokkal is éppen olyan jól dolgozik, az XHTML használata jobb ötlet, ugyanis megakadályozza, hogy olyan fájlokat készítsünk, amiket a DOM nem tud megjeleníteni. Az XML például tiltja az átfedő tagokat:

```
<i><p>Wow</i>, he thought.</p>
```

A fenti sor HTML-ben még éppen elfogadható, de tilos az XML-ben és az XHTML-ben. Így – bár a böngésző ezzel a HTML-sorral még valahogy elboldogul és értelmezni is képes – az `xmlc` figyelmeztetést fog küldeni arról, hogy eldobott egy használhatatlannak minősített bezáró tagot. Az `xmlc` gyakran fog figyelmeztetni, ha a HTML nem helyesen van megformázva, ezzel segítve a lehetséges hibák felderítését. Bár egyszerű HTML-ek írásakor nemigen kell a dokumentumunk szerkezetével foglalkoznunk, a módosítások, amelyeket az `xmlc` segítségével végre szeretnénk hajtani, a dokumentumok kiírási szabályainak világos ismeretét követelik meg.
Az előző példa első bekezdését a `"firstpara" id`-érték azonosította. Hamarosan meglátjuk, miképpen tudjuk ezt a szöveget Java-programból megváltoztatni, hivatkozási pontként használva az `id`-értéket.
Hogy a dokumentumot Java-osztállyá változtathassuk, meg kell hívnunk az `xmlc` programot. Feltételezve, hogy a fenti HTML-fájlt ***foo.html***-nek neveztük el, írhatjuk be a következőt:

1. lista. A PrintFoo.java – rövid parancssoros program, amely módosítja a „firstpara” id-vel jelölt szöveget, majd a dokumentumtartalmat a kimenetre írja

```
**// Az EnhydrÆval Ørkezı DOM-osztÆlyok
**// bet ltØse
import org.w3c.dom.html.*;

public class PrintFoo {

    // Csak egy eljÆrÆst hatÆrozunk meg,
    // amely a parancssorb l lesz megh vva.
    public static void main (String[] args)
{

        // a "foo" lap objektumunk
        // pØldÆnyÆnak elkØsz tØse.
        foo myfoo = new foo();

        // a "firstpara" bekezdØs nk
        // sz vegØnek megvÆltoztatÆsa.
        myfoo.setTextFirstpara
        ("This has been changed");

        // Az eredmØny megjelen tØse
        System.out.print(myfoo.toDocument());
    }
}*
```

```
$ENHYDRA/bin/xmlc -parseinfo -verbose -keep
➥foo.html
```

Ezáltal a foo.html ***foo.java*** Java-forrásfájllá alakul, amelyből fordítás után létrejön a ***foo.class***. A `-keep` érték megtartja a foo.java fájlt ahelyett, hogy a foo.class elkészítése fordítás után letörölné. Bár szükségtelenek, én a `-parseinfo` és `-verbose` értékeket is szeretem használni, amikor az `xmlc`-vel dolgozom, mivel így némi visszajelzést kapok a fordítás menetéről.
A Java `xmlc` által készített forráskód meglehetősen hosszú és unalmas, bár megjegyzésekkel bőven el van látva. Azok számára, akik módosítani szeretnék a foo.html-t, a foo.java legfontosabb része a `getElementFirstpara()` és a `setTextFirstpara()` eljárás. Az első visszaadja a `"firstpara"` által azonosított szöveget, az utóbbi pedig lehetővé teszi, hogy valamilyen tetszőleges karaktersorozatra cseréljük le.
Az ***1. lista*** egy rövid parancssoros Java-osztály (***PrintFoo.java***) forráskódját tartalmazza, amely kiírja a „javásított” foo.html-változat tartalmát. Mielőtt azonban kiírná, a `setTextFirstpara()` segítségével megváltoztatja a kimenetet:

```
myfoo.setTextFirstpara("This has been
changed");
```

Ezután a változtatás után már megjeleníthetjük a szöveget:

```
System.out.print(myfoo.toDocument());
```

A DOM-fát magunk is átalakíthatjuk, ha kikeressük az adott `id`-vel rendelkező csomópontokat, majd kézzel megváltoztatjuk őket. Az `xmlc` kényelmes eljárásai azonban az ilyen szövegek módosítását különlegesen egyszerűvé és magától értetődővé teszik.
Ha most lefuttatjuk a `PrintFoo`-t, megfigyelhetjük, hogy HTML-kimenet az eredeti szóközök nélkül jelenik meg. Az eredménydokumentum ugyan nehezebben olvasható az emberi szem számára, a böngészőknek azonosnak számít. A magam részéről mindig igyekeztem, hogy HTML-dokumentumaim a könnyebb hibakeresés érdekében szépen formázottak legyenek, ezért jó néven venném az `xmlc`-től, ha egy `-preserve-whitespace` (őrizd meg a szóközöket) kapcsolót is támogatna.
Abból, amit eddig láttunk, kiderült, hogy az `xmlc`-vel egyszerű egy teljes bekezdést megváltoztatni, de nehéz módosítani egyetlen szót. Az `xmlc` azonban kihasználja a HTML `"span"`-tag előnyeit, amely elfogadja az `id`-értéket, és lehetővé teszi, hogy azonosítóval lássunk el módosítani kívánt egyedi szavakat, karaktereket vagy képeket, például:

```
<P id="para">This is a paragraph,
  <span id="phrase">and this is a phrase</span>.
</P>
```

Miután az `xmlc`-vel ezt a HTML-t lefordítottuk, a `SetTextPara()` eljárással az egész bekezdés tartalmát megváltoztathatjuk, a `SetTextPhrase()`-zel pedig a `span`-tagok közötti részt módosíthatjuk.

Servletek

Most már látjuk, miképpen lehet az `xmlc`-vel parancssorból dolgozni, nézzünk meg hát egy servletet is, amely ugyanezt a feladatot látja el. A kezdők kedvéért: az egyszerű PrintFooServlet egy HTTP-kérelmet fog kapni, eredményül pedig a HTML-dokumentumot fogja visszaadni.
A ***2. lista*** tartalmazza a foo.html-t megjelenítő servlet kódját. Akárcsak parancssoros testvére, `"foo"` osztályunkból létrehoz egy példányt, megváltoztat benne néhány szöveget, végül az XML-fa szöveges megjelenését a kimeneti folyamra írja. Ebben az esetben a kimeneti folyam éppen a felhasználó böngészőjéhez kerül. A felhasználó így a módosított vázlatot látja, anélkül, hogy tudná, két Java-osztály (és az eredeti HTML-dokumentum) is részese volt a folyamatnak.
A servlet működéséhez a foo.class egy másolatát kellett behelyeznem a Jakarta-Tomcat servletmotor `CLASSPATH` környezeti változója által meghatározott egyik könyvtárba. Úgy döntöttem, hogy a legfelső szintű ***$TOMCAT/classes*** könyvtárba teszem. Ha egy termelési osztály lenne, nyilván értelmesebb helyre raktam volna, kihasználva a Java egymásra épülő névterét (namespace). Igaz, már az `xmlc`-t is a csomagok meghatározása nélkül futtattam le, ami azt jelenti, hogy a foo.class-t mindenképpen a legfelső szintű névtérben kell elhelyezni, amihez az ***il.co.lerner*** névtérben a `-class` lehetőséget kellett volna használnom:

```
$ENHYDRA/bin/xmlc -class il.co.lerner.foo\
-parseinfo -verbose -keep foo.html
```

A ***$TOMCAT/classes***-ben található foo.class segítségével a PrintFooServlet.java fájlt sikeresen le tudtam fordítani. Immár az egyetlen kihívás a servlet végrehajtása, továbbá a megváltozott HTML-lap megjelenítése maradt. Még egyszer hangsúlyozom: meg kellett ugyan változtatnom a `CLASSPATH`-t, de ebben az esetben a megváltoztatandó `CLASSPATH` a Tomcat servletmotoré volt, amely számunkra a servleteket végrehajtja.

2. lista. PrintFooServlet.java, a PrintFoo.java servletváltozata

```
**// PrintFooServlet servletvÆltozat
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

// Az EnhydrÆval Ørkezı DOM-osztÆlyok
// bet ltØse
import org.w3c.dom.html.*;

// a "foo" osztÆly bet ltØse
import il.co.lerner.*;

public class PrintFooServlet extends
        HttpServlet {

   public void doGet(HttpServletRequest
   request,
        HttpServletResponse response)
        throws IOException, ServletException
   {
      // a MIME tartalomt pus beÆll tÆsa
      // a vÆlaszhoz
      response.setContentType("text/html");

      // a kimeneti folyam beÆll tÆsa
      // STDOUT-ra
      PrintWriter out = response.getWriter();

      // a "foo" lapobjektumunk pØldÆnyÆnak
      // elkØsz tØse.
      foo myfoo = new foo();

      // a "firstpara" bekezdØs nk
      // sz vegØnek megvÆltoztatÆsa.
      myfoo.setTextFirstpara
      ("This has been changed");

      // Az eredmØny megjelen tØse
      out.println(myfoo.toDocument());
   }
}*s
```

Megváltoztattam a *$TOMCAT/bin/tomcat.sh* fájlt, mely mielőtt éppen exportálta volna a `CLASSPATH`-t, hozzá kellett adnom a három Enhydra .JAR állományt, majd újraindítanom a Tomcatet. Pár pillanattal később böngészőmmel a servletre mutatva elégedetten figyelhettem meg képernyőmön az eredeti HTML-fájl megváltoztatott formáját.

Adatbázisok és az XMLC

Könnyű belátni, hogyan lehetne a lapot relációs adatbázisból származó adatokkal benépesíteni: például itt van egy kicsi PostgreSQL-tábla, amelyben az egyes naptári napokhoz tartozó mondásokat (`saying`) tárolhatjuk:

```
CREATE TABLE DailySayings (
    date     TIMESTAMP    NOT NULL,
    saying   TEXT         NOT NULL,
    UNIQUE(date)
)
```

Most szúrjunk be pár mondást a táblánkba:

```
INSERT INTO DailySayings(date, saying)
VALUES (CURRENT_DATE,
·A bird in the hand is worth two in the
bush.·);
INSERT INTO DailySayings(date, saying)
VALUES (CURRENT_DATE+1,
·A penny saved is a penny earned.·);
INSERT INTO DailySayings(date, saying)
VALUES (CURRENT_DATE+2,
·The rain in Spain falls mainly in the
plain.·);
```

A mai mondás lekérdezéséhez mindössze a következőre van szükségünk:

```
SELECT saying
FROM DailySayings
WHERE date = CURRENT_DATE
```

Ha olyan servletet szeretnénk írni, amely megjeleníti a mai mondásokat, két osztályra lesz szükségünk: egy mintára, amit az `xmlc`-vel készítünk el, (illetve a ***saying.html***-re, amiből a ***saying.class***-t fordítjuk), és egy másikra, amely betölti és módosítja a mintát (***DailySaying.java***). XMLC-dokumentumunk és a módosítóosztály fejlesztése közben megegyezünk abban, hogy a `"saying" id` fogja a kettőt összekapcsolni.

XMLC-dokumentumunk eléggé magától értetődő:

```
<html>
   <head><title>Today·s saying</title></head>

   <body>
      <h1>Today·s saying</h1>

      <p>And now, as you requested, today·s
      saying:
        <span id="saying">Saying Goes
        Here</span>.</p>
   </body>
</html>
```

Ezt a HTML-dokumentumot az ***il.co.lerner.saying*** Java-osztállyá fordítottam le, a móka kedvéért a .java fájlt is megtartva:

```
$ENHYDRA/bin/xmlc -class il.co.lerner.saying\
-parseinfo -verbose -keep saying.html
```

Ezután az eredményül kapott saying.class fájlt bemásoltam a ***$TOMCAT_HOME/classes/il/co/lerner*** könyvtárba, ahol egyébként a servletekkel kapcsolatos osztályaimat tartom. Miután telepítettem a dokumentumot, meg kellett írnom a módosítóosztályt, ami végrehajtja a fent bemutatott SQL-lekérdezést, letölti az eredményeket és beilleszti a lefordított XMLC-dokumentumba. A 3. lista (20. CD-mellékletünkön találhatják meg) tartalmazza a servletünkhöz tartozó forráskódot, amit lefordítás után a Tomcat kiszolgáló működő

Kapcsolódó címek

Amint már említést nyert, az XMLC a Lutris Enhydra alkalmazáskiszolgáló része, amely teljes mértékben együttműködő próbál maradni a J2EE-vel. Az Enhydra 4, amely nem sokkal e cikk megírása előtt lépett a próbaváltozat állapotába, az Enterprise JavaBeansnek és más magas szintű programrendszernek az Enhydrába történő bevezetését célozta meg. Többet is megtudhatunk az Enhydráról, és a 3.x vagy 4.x sorozat legújabb változatait tölthetjük le a ➲ http://www.enhydra.org/ címről.

Az Enhydráról általánosságban szóló, néhány XMLC-példával megtűzdelt írás *Roger Metcalf*-nak az ArsDigita Systems Journalban található cikke, lásd: ➲ http://www.arsdigita.com/asj/enhydra/.

Ugyancsak kitűnő XMLC-segédanyag található a Weben, a ➲ http://www.plugged.net.au/publications/xmlc-tutorial/urls.html címen.

A W3C (➲ http://www.w3.org/TR/2000/REC-xhtml1-20000126/) honlapján érhető el az XHTML-t bemutató dokumentumuk, amely figyelemre méltóan jól olvasható ismertetés az XHTML-ről. Csak azt írja le, ami szabályos (és ami nem). Bár meglehet, hogy XMLC-alkalmazásainkban nem akarunk szigorú XHTML-t használni, azért nem árt, ha megismerjük.

Az Apache Jakarta-Tomcat JSP és a servletmotorról szóló tájékoztató a ➲ http://jakarta.apache.org/ címen érhető el.

Végül két egymást kiegészítő *O'Reilly*-könyv elég ismerettel szolgálhat, hogy az XMLC-vel dolgozni kezdjünk. A *Brett McLaughlin* (most a Lutrisnál dolgozik) által írt „Java and XML" részletesen tárgyalja a DOM-ot, összehasonlítja a SAX-szal és más XML-értelmező lehetőségekkel. A „Java Servlet Programming" második kiadása *Jason Hunter*-től és *William Crawford*-tól egy teljes fejezetet szentel az XMLC-nek, rögtön a hasonló, de versenytársként szereplő kiszolgálóoldali Javát alkalmazó dinamikus lapelőállító rendszerek fejezete után.

servlettárolójába (active servlet context) helyeztem. A Tomcat és az Apache újraindítása után böngészőmön keresztül már hozzá tudtam férni a mai szóláshoz a HTML-dokumentumba helyezett SQL-eredmények segítségével.

Jó megoldás az XMLC?

Amikor először kezdtem foglalkozni az XMLC-vel, komoly kétségeim voltak a használhatóságával kapcsolatban. Több év vegyes sablonon alapuló munka után egyszerűen túl vadnak látszott a HTML-fájlok Java-osztállyá változtatása csak azért, hogy a DOM segítségével kezelhessük őket. És hát egy DOM-értelmezőt elindítani valóban sokkal több erőforrást igényel, mint egyszerűen megjeleníteni a fájlt.
Ahogyan azonban egyre többet dolgoztam már az XMLC-vel, fokozatosan felfigyeltem a sablonokkal szembeni előnyeire. Az XMLC lényegében rákényszeríti a tervezőket és a fejlesztőket, hogy megállapodást kössenek, vagy API-leírást alkossanak dokumentumaik és a program között. Ha egyszer ez az API elkészült, nem lehet egykönnyen megváltoztatni, ami nem feltétlenül rossz dolog. A legfontosabb, hogy a fejlesztők és a tervezők közötti API üzembiztossága lehetővé teszi számukra az egymás tevékenységét szinte egyáltalán nem zavaró párhuzamos munkát.
Mivel a Java-módosító osztály a lefordított dokumentum HTML-tartalmát bármilyen módon meg tudja változtatni, könnyen elképzelhetünk egy olyan helyzetet, amelyben egyszerre három osztállyal dolgozik: egy fejlécfájllal, a dokumentum testével és a láblécfájllal. Az osztályunk ezek után a fejlécet a DOM-eljárások segítségével hozzácsatolhatja a dokumentum elejéhez, a láblécet pedig a végéhez. Ezzel a módszerrel általános formátumot tudunk adni a honlapnak anélkül, hogy ugyanazt a szöveget kellene minden fájl elejére másolnunk. Természetesen számos bosszantó részlet is akad a XMLC-vel való munka során. Hamar unalmassá válik például, hogy minden HTML-fájlhoz servletet kell írni. Igaz, egyetlen servletet is írhatnánk, amely a fájlnevet egy lekérdezésből kapja meg, és tulajdonképpen majdnem úgy működik, mint egy XMLC által készített különböző osztályokhoz tartozó dokumentumsablon. Meglehet, hogy egyszerűen csak nem néztem át elég tüzetesen az Enhydrát ahhoz, hogy megtaláljam e kérdésre a választ, vagy az is lehet, hogy az Enhydra-fejlesztők gyorsan hozzászoknak ahhoz, hogy minden megjeleníteni kívánt laphoz két Java-osztályt kell készíteniük. Mindenesetre ezáltal igen rövid idő alatt óriási számú osztály jöhet létre, még kicsi vagy közepes honlap esetén is.
A legnagyobb gond az XMLC-vel kapcsolatban szerintem a magas szintű HTML (és XML – a pontosság kedvéért) módosító API hiánya. Az XMLC egyik GYK-ja a „Hogyan adhatok egy sort egy HTML-táblához?" Egy ilyen, a HTML-ben magától értetődő feladat hamar terhessé válhat az XMLC-vel. Először is meg kell találni annak a táblának az alját, ahová a sort be szeretnénk szúrni, majd különleges csomópontokat (és értékeket) kell hozzáadni a csomóponthoz. Ez igencsak nem-HTML „szagú" és rákényszeríti a fejlesztőt, hogy csomópontokban gondolkodjon, amikor ő HTML-ben szeretne gondolkodni. Annak alapján, hogy az Enhydra egy Java-eljárások segítségével történő SQL-lekérdezés szerkesztési lehetőséget tartalmaz, úgy gondolom, egy ehhez hasonló HTML-módosító API nem lenne túl bonyolult.

Következtetés

Az XMLC igen fondorlatos, az Enhydra alkalmazáskiszolgáló szívében helyezkedik el. Az XMLC rákényszeríti a fejlesztőket (és a tervezőket), hogy mielőtt dolgozni kezdenének, megtervezzék, hogyan fognak együttműködni, majd lehetővé teszi, hogy egymástól függetlenül dolgozzanak. Bár az ilyesfajta művelet kibillentheti egyensúlyukból a tapasztalt sablonhasználókat, hamarabb válik természetessé, mint az ember valaha is gondolná. Az igazság az, hogy a Zope ZPT-je is hasonló módszert alkalmaz az űrlaptartalom szétválasztására, ami talán a webfejlesztő közösség jelenlegi irányvonalát mutatja. A közeljövőben várhatóan további XMLC-szerű rendszerekkel fogunk találkozni. Ha szerencsénk van, lehet, hogy valamiféle szabványosítás is lezajlik a minták közt, így a tervezők könnyen mozoghatnának a különféle rendszerek között a köztük lévő apró különbségek elsajátítása nélkül.

Reuven M. Lerner (reuven@lerner.co.il) egy kis webes-internetes tanácsadó cég tulajdonosa. Feleségével, Shirával és lányával, Atara Margalittal a „jövő városában", az izraeli Modi'inben él.
➲ http://www.lerner.co.il/atf/

A Mandrake Linux 8.1 telepítése

Ebben a rövidke leírásban megpróbálom összefoglalni a Mandrake Linux 8.1-es telepítésénél tapasztalt újdonságokat, változásokat és azokat a beállításokat, amelyekre figyelni érdemes.

A leírás elsősorban azoknak szól, akik kezdőként próbálkoznak a Linuxszal, vagy a parancssoros beállítások mellőzésével szeretnének gyorsan, jól működő rendszert létrehozni.
A telepítést windowsos hálózatban hajtottam végre, ahol ez volt az egyetlen Unix-rendszeren alapuló eszköz (néhány nyomtatón kívül). A telepítési környezet ismeretéhez az is hozzátartozik, hogy a hálózat többi tagja általában Win9x, ritkán NT 4.0-s munkaállomás. A hálózat TCP/IP-protokollt használ, az ügyfelek DHCP útján kapnak IP-címet. A kiszolgálókon természetesen szintén Windows NT 4.0 változat futott, illetve – ha jól tudom – most már néhány Windows 2000 is megtalálható rajtuk. A levelezést az Exchange kiszolgálóval oldották meg.
A telepítés alapjául szolgáló számítógép egy Dell GX110-es modell, amely az alábbi eszközöket tartalmazta (csak a jellemzőbbek):

Intel(r) 82801AALPC alaplap,
- SoundMAX alaplapra épített hangkártya,
- Pentium III 800 MHz-es processzor 256 MB memóriával, 20 GB-os Quantum merevlemezzel,
- Sony CD-újraíró.

Elöljáróban elárulnom, hogy a leírás a Mandrake 8.0 beállításaihoz íródott, ám írásunk közvetlen megjelenése előtt megérkezett a Mandrake 8.1-es változata. Ez azonban annyi új eszközt tartalmaz, hogy ilyen gyorsan nem tudtuk kitapasztalni, mi mennyire állja meg a helyét az új változatban. Elődeivel ellentétben a Mandrake 8.1 a Hálózatról három CD-t enged letölteni. Az első lemez az alaprendszert, illetve a beállításhoz szükséges eszközöket tartalmazza; a második korong a szabad, bárki által felhasználható programokat foglalja magába; a harmadik CD pedig a saját fejlesztéseket, illetve a GPL-től eltérő felhasználási szerződésű programokat (ezek közül számos ingyenesen használható fel) rejti. A három lemez alapján azt gondolhatnánk, hogy a telepített rendszer mérete is ennek megfelelően fog növekedni – szerencsére ez nem következett be.

A hálózat felderítése

Miután a három Mandrake 8.1-es alap CD-t beszereztük, legelső lépésként gyűjtsünk adatokat a számítógépünkről (ellenőrizzük, hogy Linux-változatunk támogatja-e a benne lévő eszközöket), illetve a hálózati környezetről, hiszen az új rendszerben szükségünk lesz a kiépítés ismeretére. Gépünk felderítésére alkalmas lehet a windowsos ***Sajátgép*** eszközkezelője is, itt számítógépünk alkatrészeiről nagyjából pontos adatokat kaphatunk. Szerencsés helyzetben voltam, hiszen e gép elődjére már telepítettem a Mandrake 7.2-es és 8.0-s változatát, és akkor nem jelentett gondot beállításuk.

7.


8.


9.

10.

Általában elmondható, hogy a Linuxot is mindenki olyan gépekre telepíti előszeretettel, amelyeknek eszközeit már gyárilag összehangolták, és ennek köszönhetően kicsi az esélye, hogy a felhasznált eszközök hibája vagy összeférhetetlensége (például ugyanazt a megszakítást szeretnék használni) miatt a telepítés meghiúsul.
A második igen fontos lépés a hálózat felderítése. Mindenki maga döntse el, hogy sajnálatos-e vagy sem, de a Mandrake 8.1 már rendelkezik hálózatfelderítő eszközökkel, amelyek lehetővé teszik a hálózat elemeinek felismerését, és még a kezdő felhasználók számára is gyorssá és könnyűvé teszik a hálózatbeállítást. Ennek ellenére célszerű minél több adatot összegyűjtenünk a hálózatról. Az egyik lehetőség a hálózat rendszergazdájának megkérdezése, ami azonban számos esetben megoldhatatlan, hiszen nálunk például a hálózat mérete miatt meglehetősen elfoglaltak, illetve korántsem biztos, hogy repesnének az örömtől, ha azt hallanák, hogy egy másik operációs rendszert is telepíteni szeretnénk a hálózatukban. Ez részben érthető, hiszen a háta mögött folyó „partizánakciókért” senki sem rajong, számomra viszont érthetetlen, hogy sok cégnél miért félnek az olyan újdonságtól, amely bizonyítottan minőségi javulást eredményez majd a munkájukban.
A hálózat felderítéséhez a Windows 98-ban is megtalálható `netstat` és `net` programok szolgáltatásait használtam.
A `net` parancs segítségével gépünk hálózati beállításainak nézhetünk utána, ehhez használjuk a `net config` parancsot.

```
Computer name                    \\sajat_gep
User name                        TOTHB
Workgroup                        HUN
Workstation root directory       C:\WINDOWS
Software version                 4.10.1998
Redirector version               4.00
The command was completed successfully.
```

A kiadott parancs eredményeként tulajdonképpen csak a gép windowsos nevét, valamint a munkacsoport nevét tudjuk felhasználni.
A másik utasítás a `netstat`, amely a hálózatunkról szolgáltat adatokat. A parancs eredménye az alábbi:

```
Active Connections
Proto  Local Address  Foreign Address
                      ➥State
TCP    bela:1138      kzsa.l.ge.com:nbsession
                      ➥ESTABLISHED
TCP    bela:1156      hun.l.ge.com:smtp
                      ➥TIME_WAIT
```

Ha az utasítást a `-a` kapcsolóval alkalmazzuk, a hálózat többi gépéről is adatokat nyerhetünk. Náluk a hálózat viszonylag nagy mérete miatt ez a parancs sajnos nem működött. Ha részletesebb adatokat szeretnénk kapni, például a `netstat -r` parancssort használhatjuk.

```
Active Routes:
Network Destination   Netmask          Gateway
                      ➥Interface      Metric
127.0.0.0             255.0.0.0        127.0.0.1
                      ➥127.0.0.1      1
3.30.235.173          255.255.248.0 ...
```

A hálózat beállításait még a ***My Network ...*** segítségével is célszerű megvizsgálni, hátha további adatokhoz juthatunk, például a DHCP-, illetve DNS-kiszolgálóról.

A Linux telepítése

A számítógép újraindítása után az első telepítő CD indításával belevághatunk a telepítésbe. A telepítőablak felépítése a bejelentkező képernyőtől eltekintve a 8.0-s változatéhoz képest szinte alig változott. A rendszer két telepítési módot kínál: a ***Recommended***-et a kezdő felhasználók számára, és az ***Expert***-et, mely a haladó felhasználók számára ajánlott. A kezdő módot választottam, hiszen arra voltam igazán kíváncsi, milyen új segítséget kapnak a kezdők. A telepítő nagyon kevés beállítást kért a telepítés során, például nyelvi beállításokat, a telepítendő eszközök nevét, monitortípust (nem túl szerencsés, hogy a telepítő a monitorbeállításnál nem próbálta ki a beállításokat).
A telepítési folyamat közben igyekeztem a hálózatot is beállítani, amihez a DHCP-kiszolgáló használata következtében csak a

1. kép KCron az időzített alkalmazásindítás beállításához
2. kép A kwuftpd az FTP-szolgáltatás grafikus beállítóeszköze
3. kép A rendszer- és felhasználói menük testreszabását együttesen a MenuDrake-ben lehet elvégezni
4. kép A Linuxconf hálózatbeállítási része
5. kép A 8.1-es Mandrake egyik újdonsága a hálózati eszközök beállítását teszi elvégezhetővé
6. kép A Mandrake 7.2 hagyományainak megfelelően, az RpmDrake biztosítja a könnyű csomagkezelést
7. kép A Swat minden Linux-változatban elérhető grafikus beállítóeszköz a Sambához
8. kép Minden befűzött adattároló eszközt egy helyen állíthatunk be
9. kép A felhasználói jogok egyik grafikus eszköze a Mandrake-ben a UserDrake
10. kép A WizDrake a Mandrake 8.1 egyik újdonsága: a hálózatbeállító eszközök grafikus felületen, egy helyen érhetők el

gépnév helyes kiválasztása szükséges. Figyelembe kell vennünk, hogy a teljes gépnév szükséges, például: ***tesztgep.linux.net.*** Ezekután a telepítő újraindította a gépet, és körülbelül húsz perccel a telepítés után már elérhető volt a hálózat!

A telepítő a hálózat mellett hangkártyámat és CD-írómat is helyesen állította be.

Érdekesség, hogy a régebbi változatokkal ellentétben a CD-írót eleve SCSI-eszközként kezelte. A telepítés során megadtam egy felhasználót, és azt is, hogy a rendszer elindulása után próbálja meg önműködően beléptetni. Így a gép újraindulását követően a program magától beléptetett, ráadásul kellemes meglepetés ért, ugyanis a rendszer felajánlotta a grafikus felületnek és témájának kiválasztási lehetőségét, majd a KDE eszközei számára kérte a felhasználó adatait.

Ezt követte a levelezőprogram beállítása – a Netscape-et választottam. Az előzőleg elvégzett X-beállításnak megfelelően már grafikus felületen jelentkeztem be.

A telepített rendszer mérete az összes játékkal, ablakkezelővel, szövegszerkesztővel, multimédiás eszközzel együtt megközelítőleg mindössze 1300 MB lett.

Linux-listákon többen olvashattuk, hogy a 8.0-s változatban a betűkészletekkel valami nagyon nincs rendben. A telepítés után magam is hasonlót tapasztaltam. A hibát tökéletesen kijavították a 8.1-es változatban, ugyanis a betűk többféle méretben is jól olvashatók maradtak, és az sem okozott gondot, ha a nekem leginkább tetsző betűkészletet választottam ki. Ahhoz, hogy tökéletesen honosított KDE (KDE 2.2.1) felületet kapjunk, elegendő a nyelvi környezetet magyarra állítani, és a betűkészleteknél az ISO-8859-2-es kódolását kiválasztani. A telepítés során csupán elvétve találkoztam angol nyelvű kérdésekkel (talán csak az elején, a nyelvi környezet esetében). A KDE is magyarul beszélt és az alapbeállításoknál is csak néhány angol kérdés fordult elő.

Előfordulhat, hogy valamiért nem sikerül azonnal beállítani a Sambát, e feladat megoldásához szeretnék most segítséget nyújtani. A DHCP ügyféloldali része nagyon rosszul dokumentált, ezért eléggé sokáig kellett próbálkoznom a 8.0-s hálózat beállításával (nem vagyok hálózati szakember).

Ha valami gond lenne a grafikus beállítással, a */etc/dhcpd/dhcp.conf* fájlt kell megkeresnünk és a következők szerint módosítanunk: a Samba működőképessé tételéhez gépünket állandó IP-címmel kell ellátni, illetve a gépeket tartalmazó beállítófájban is rögzítenünk kell. Szemléltetésül álljon itt egy példa:

```
subnet 3.30.232.0 netmask 255.255.248.0 {
# --- alapØrtelmezett Ætjær =default gateway
option subnet-mask 255.255.248.0;
option domain-name "l.ge.com";
option domain-name-servers 3.30.232.11;
# a gØp nk beÆll tÆsai, illetve a szÆmunkra
# fontos gØpek adatai: a gØp neve: host lin
{
# a gØp hÆl zati kÆrtyÆjÆnak IP-c me:
hardware ethernet 00:B0:D0:9A:83:B7;
# a hozzÆrendelt Ælland  IP-c m:
fixed-address 192.168.1.2;
}
}
```

Célszerű megemlíteni, hogy a *dhcpd.conf* fájl első négy hálózati címe megegyezik a Windows `netstat -r` parancsának eredményeként kapott lista második sorával.

A kártya címét az `ifconfig eth0` parancs második sora tartalmazza. Az itt megadott állandó IP-címet és a gépnevet fel kell tüntetni a host.conf fájlban is, valahogy így:

```
192.168.1.2 lin.l.ge.com lin
```

Ezekután a Samba elindult. A tapasztalataim alapján leszűrhető tanulság: a hagyományos parancssoros beállítás szinte minden esetben működőképes.

A Linux beállítása

A hálózat beállítására és a felhasználók felvételére a *Linuxconf* eszközt szoktam használni. Most, hogy az új változat eszközeit akartam kipróbálni, mindig a Mandrake saját eszközkészletét alkalmaztam. A felhasználói jogok módosítására a ***Userdrake*** programot használtam.

A hozzáértő felhasználók számára célszerű megfelelő jogosultságot adni a `linuxconf` használatára és a változások jóváhagyására, valamint a `samba` karbantartására is. E beállításokat grafikus felületen a `linuxconf`-ból végezhetjük el.

Alaphálózat és Internet

Az alaphálózat és az Internet beállítása zajlott a legegyszerűbben, hiszen minden szükséges eszköz rendelkezésre állt hozzá, valamint az összes beállítás ismert volt.

A Mandrake három olyan eszközt is tartalmaz, amelyek e beállítások elvégzését lehetővé teszik: a ***Linuxconf***-ot, a ***WizDrake***-et és a ***Mandrake Vezérlőpult***-ot (Control Center).

Talán kicsit furcsának tűnhet a beállítóeszközök ilyen mértékű részletezése, de korántsem biztos, hogy minden olyan gördülékenyen zajlik, mint esetünkben tette.

A `linuxconf` elindítása után az alap hálózati beállításoknál a ***Host name and IP devices*** ablakban kell megadni a gép nevét (`lin.l.ge.com`), és beállítani az első hálózati eszközt (***4. kép***).

A beállítások mentéséhez az ***Enable*** gombbal engedélyezni kell az eszközt. Mivel tudjuk, hogy az IP-cím megadása DHCP útján történik, a ***Config mode***-ban ezt választottam ki. A ***Netmask*** értékét a `netstat -r` parancs eredményeként kapott listában találjuk meg, az ***Active Routes*** rész második sora tartalmazza. Ha gépünk felismerte a hálózati kártyát, a következő sorokkal nem kell foglalkoznunk. E sorok közül a ***Net device*** (általában ez eth0) és a ***Kernel Module*** (esetünkben 3c509) mezőknek kell kitöltve lenniük. A DNS-beállításokkal ne foglalkozzunk, mivel a rendszer a DHCP-kiszolgálóról önműködően beállítja őket.

Ha mindent jól csináltunk, az `ifconfig eth0` parancs eredményeként ilyesmit fogunk látni:

```
[root@lin tothb1]# ifconfig eth0
eth0 Link encap:Ethernet
HWaddr 00:B0:D0:9A:83:B7
inet addr:3.30.235.173 Bcast:3.30.239.255
Mask:255.255.248.0
```

– itt még folytatódik a parancs kimenete, de számunkra a fenti két sor a lényeges. Ha a sorok tartalmazzák az `inet addr` és a `Bcast` értékeit, jó munkát végeztünk.

A WizDrake hálózatbeállítóját nem próbáltam ki, csak a Vezérlőközpont hálózatbeállítását, kíváncsiságból. Használatáról most nem írok, mert a varázsló minden beállítással kapcsolatos kérdést eléggé egyértelműen és magyar nyelven tesz fel.

A következő lépés az internetelérés beállítása. Az átjáró megtalálásában a `netstat` parancs, illetve a Windows alatt

található böngészők proxybeállításai segíthetnek, én az utóbbiakat használtam.
Az Internet Explorer esetében nyissuk meg a ***Tools*** menü ***Internet Options…*** menüpontját, és a megjelenő ablakban a ***Connections*** fül ***LAN settings*** menüpontját válasszuk ki. Az itt felbukkanó ablakban lévő ***Use proxy server*** beállítását kell feljegyeznünk. Ha a rész üres, a proxykiszolgálót a `netstat` szolgáltatásának vagy a rendszergazda segítségével találhatjuk meg. Közvetlen kapcsolódás esetén a linuxos böngésző számára semmit nem kell beállítanunk. Jobban kedvelem, ha például a Netscape 6.1 a Windows böngésző eszköze. Tapasztalataim szerint Windows alatt a 6.1-es Netscape nagyrészt javította a 6.0-s változat hibáit. Tökéletesen kezeli az Explorerek számára készített weboldalak beállításait. Weblapszerkesztője is meglehetősen egyszerű, segítségével gyorsan és megbízható módon készíthetünk weboldalakat. Az ily módon létrehozott weblap az esetek nagy részében ugyanúgy jelenik meg az MS Explorerben is. A levelezés az előző változatéhoz képest sokat javult. A 6.0-s változat levelezőjében olyan eset is előfordult, hogy a csatolt fájlokat nem tudta megnyitni, illetve a hozzájuk rendelt programokat nem kezelte. Esetenként a csatolt fájlok eltűntek vagy a saját könyvtárakban tárolt levelek a könyvtár indexhibája miatt elvesztek. Tapasztalataim szerint ezeket a hibákat kijavították és ismét üzembiztosan, valamint jóval kisebb gépigénnyel működik, mint az Outlook változata.
A beállításokhoz visszatérve: mi Exchange kiszolgálót használunk. A levelek letöltéséhez és kezeléséhez IMAP-os beállítást használok, melyeket Linux alá is át kell vinnünk. Általában az Exchange kiszolgálót futtató gépek is állandó IP-címmel rendelkeznek, így Linux alatt nem kell a gép „megtalálásával” foglalkozni. A kiszolgáló pontos nevére azonban szükségünk van. Ha elosztott Exchange kiszolgálóval rendelkezünk, vagyis több gép szolgálja ki a hálózatot, akkor ki kell próbálnunk, hogy Linux alól melyiket tudjuk igazán jól használni. A választás általában nem okoz gondot, hiszen ha az Outlook ***Eszközök*** menüjében a ***Karbantartás***-t választjuk ki, az Exchange kiszolgálóhoz tartozó részben meg fogjuk találni a kiszolgáló nevét. Gondot az okozhat, ha nem csak az itt látott kiszolgáló létezik, és emiatt a levelezéssel (főleg a levélküldésnél) valamilyen gondunk akad, ezért másik kiszolgálót szeretnénk beállítani. Ebben az esetben a következő megoldást javaslom: egy „kintről érkezett” levelet szöveges fájlként mentsünk, és a fájl fejlécét a szövegszerkesztőben nézzük meg. Ha minden igaz, az Outlook-beállításban már látott kiszolgálót itt is megtaláljuk, illetve a levél útját végigkövetve a belső levelezőkiszolgálók neveire is rábukkanhatunk.
Levelező- és böngészőprogramként én a Netscape-et vagy Mozillát ajánlanám (ezeket alaposabban kipróbáltam). A KDE és a StarOffice (az utóbbi Windows alatt jól működött) böngésző- és levelezőeszközeit sajnos nem állt módomban próbára tenni. FTP-programként a gFTP-t telepítettem, amit az átjáró címének és fajtájának ismeretében (HTTP-proxy) könnyen be tudtam állítani. Eközben arra kellett igazán figyelnem, hogy az ***Option*** menüben a HTTP-proxyt, illetve a levelezőprogramból már ismert proxynevet állítsam be (***2. kép***).
A DHCP-vel nyert IP-cím egyedi beállításokat igényel (hiszen szükségünk van a hálózati kártya saját címére). Ezt a feladatot a ***WizDrake***, a Samba beállítóeszköze látja el. Ezekután a Samba-szolgáltatást a ***Swat*** segítségével be tudjuk állítani. Fontos az engedélyezett gépek beállításnál szereplő értékeire figyelni, ha ugyanis szűk hálózatot jelölünk ki egy több IP-tartományban dolgozó belső hálózat esetében, ezzel akaratlanul is meglehetősen korlátozhatjuk az eszközeinket használni kívánók körét.

A ***Mandrake Vezérlőult/Hardver/Csatlakoztatási pont*** ablakán keresztül ugyancsak grafikusan kapcsolódhatunk más megosztásokhoz.
A csatlakoztatási ponthoz azonban nem tudtam jelszót és felhasználónevet megadni, továbbá a csatlakoztatott megosztások leválasztása sem működött tökéletesen.
A hálózat beállítása a Mandrake 8.1 szerves részét képező ***Webadmin*** segítségével szintén pofonegyszerű feladat.
Továbbra is gondot jelentett, ha nem tudtam fejből a célgép pontos címét, ilyenkor a hálózat átvizsgálása 10–15 percig is eltartott.
A magyar karakterek helyesen jelentek meg, egyedül az ***ő*** betűvel küszködtem. Az angol Windows 9x-es ügyfélhez hasonló módon a magyar karaktereket tartalmazó fájlokat itt sem sikerült átmásolnom (ez inkább a Windows hibája, ráadásul még a 2000-es változatban sem javították ki).
A Mandrake ***Vezérlőpult***-jának használatával próbálkoztam a nyomtatókiszolgálóhoz kapcsolódni, de ez az eszköz sajnos szintén állandó IP-című nyomtatókiszolgálóhoz (amelyet egyébként villámgyorsan be tudtam állítani) készült, és nincs grafikus lehetőség arra, hogy egy ilyen típusú hálózatban a Win95-ös ügyfélhez csatolt nyomtatót távolról gyorsan működésre bírjuk.

Összegzés

Kellemes tapasztalatokat szereztem a Mandrake 8.1-es változattal, hiszen ebben a környezetben hozzá nem értő módjára is gyorsan be tudtam állítani a linuxos ügyfelet. Az előző változathoz képest számos hibát kijavítottak, és a szükséges engedményekkel együtt is megmaradt a linuxos testreszabhatóság. Remélhetően a Linux eddigi fejlődését ismerve hamarosan további olyan eszközöket sikerül létrehozni, amelyek biztosítják az ilyen és ehhez hasonló környezetben történő teljes körű Linux-használatot.

➲ http://www.linux-mandrake.com/en/

Tóth Béla (tothb1@freemail.hu)
Nős, két gyermek büszke atyja. Dolgozott földmérőként, majd térinformatikus szakmérnöki képesítést szerzett. Egyaránt otthonosan mozog a CAD és a térinformatikai programokban, valamint a DOS- és Windows-alkalmazásokban. Legkedveltebb elfoglaltsága már két és fél éve a Linux.

Brochettes de Sécurité

Marcel bemutatja, hogyan állíthatunk vissza törölt állományokat, miként rejthetjük el titkos adatainkat, illetve hogyan készíthetünk róluk biztonsági másolatokat.

Á, François! Gyere, ezt nézd meg! Eszmefuttatásom tárgya a biztonság, és sikerült is összeállítanom a témáról egy étlapot. Quoi? Természetesen nem a hálózatokról van szó. Tudod, mon ami, egy étterem sikeres vezetéséhez mi a legfontosabb? Bien sur, egy gazdagon felszerelt borospince nagyon fontos, ebben egyetértünk, de most az ételre gondoltam. Tudod, a vendégeink sok mindenre éhesek és a mi kötelességünk, hogy olyan ízekkel kápráztassuk el folyamatosan az ínyüket, amilyenekre nem számítottak.

Qu'est-ce que tu dis? François, a biztonság a Linuxszal való főzőcskézés igen sokoldalú kérdése. A hálózati biztonság is egy ilyen kérdés, s nyilván mindenkinek ez jut először az eszébe. Ahhoz azonban, hogy az ízlésünk friss és éber maradjon, érdemes egy kicsit válogatni a hozzávalók között. Nézd meg például az első fogást! Nem tagadhatod le, hogy köze van a biztonsághoz, továbbá – François, miért nem figyelsz? Hogyan? Á, mais pardon, megérkeztek a vendégeink. Kérek mindenkit, foglaljon helyet! François azonnal hoz egy kis bort nekünk! Akad még a pincében egy 1998-as Clos de Vougeot, mely azt hiszem, ez mindenki ízlésének megfelel, és különleges légkört teremt. Egy finom Burgundy, mes amis. Vite, François, vite! François-val éppen a biztonság kérdésének különböző nézőpontokból történő megközelítéséről beszélgettünk. Amikor Linuxszal főzünk, a „biztonság” fogalmát emlegetve mindnyájan a hálózati biztonságra gondolunk, pedig a szó monsieur Roget híres fogalomtára szerint számos egyéb dolgot is jelenthet. Olyan szavakat lelhetünk fel benne, mint kötvény, letét, adóslevél, óvadék, kezes, hitelesítés, nyugta, hogy csak néhányat említsünk közülük. És mi a helyzet azzal a biztonsággal, amit rendszeresen végzett biztonsági mentésként ismerünk? Önök minden bizonnyal gyakran készítenek biztonsági másolatokat, ily módon épségben tudhatják a munkájukat, és ezt a fajta biztonságot a rendszeresen végzett adatmentés nyújtja. De még ebben az esetben is előfordulhat, hogy elvész egy állomány. Mi a helyzet akkor, ha az utolsó biztonsági mentés tegnap este vagy akár két órával ezelőtt történt? Jó szokásunk és a biztonságos munkához való egészséges vonzódásunk ellenére mégis elvesztettük az állományunkat. Mit tehetünk ebben a helyzetben? Már hallom a hétköznapi bölcsességet: ha egyszer letöröltünk egy állományt Linux-rendszerünkben, annak vége, örökre elveszett. Ez azonban nem teljesen van így, lehet még esélyünk a megmentésére.

Tesztrendszeremen egy kis meghajtó van csatlakoztatva a /mnt/tinydrive útvonalon. A `df /mnt/tinydrive` parancsra az alábbiak jelennek meg:

```
/dev/hdd6   42913   9063   31634   22%
/mnt/tinydrive
```

Mondtam, hogy kicsi, igaz? Létrehoztam néhány könyvtárat és állományt ezen a meghajtón. Az egyik állomány neve ***cabernet*** és a családom Cabernet-titkait tartalmazza Az állományt kilistázva ezt kapom eredményül:

```
$ ls -l
-rw-r--r--   1 marcel   wines   77 Jun 21 04:38
cabernet
```

77 bájt mint látjuk a titkok mindig elegánsak. Ha most az `rm cabernet` paranccsal az állományt, eltűnik, és nem tudjuk olvasni. Ennek ellenére egy kis kitartással és szerencsével visszaállíthatjuk, ahogy mindjárt be is mutatom. Fontos, hogy a kérdéses meghajtót azonnal leválasszuk, vagy kapcsoljuk ki a gépet, ha a gyökérkönyvtárról van szó:

```
unmount /mnt/tinydrive
```

Minél több idő telik el ugyanis az állomány törlésétől kezdve (következésképp minél több adat íródik a lemezre), annál kisebb az esélyünk arra, hogy az állományt maradéktalanul sikerül visszaállítani. A mentési kísérlethez a ***Debugfs*** nevű programot használom. Csak semmi fejetlenség, mes amis, minden rendszeren megtalálható ez a parancs az ***e2fsprogs*** csomag részeként:

```
# debugfs /dev/hdd6
debugfs 1.14, 9-Jan-1999 for EXT2 FS 0.5b,
95/08/09
debugfs:
```

Ha a `debugfs` parancssorába beírom, hogy `lsdel`, egy listát kapok, amely valahogy így néz ki:

```
7665    0 100644   5248    6/   6 Thu Apr 19
18:05:30 2001
  32 500 100600 12288   12/ 12 Thu Jun 21
04:38:39 2001
 158 500 100644     78    1/   1 Thu Jun 21
04:38:39 2001
 157 500 100644     77    1/   1 Thu Jun 21
04:40:25 2001
```

Az első oszlop a fájlleíró (inode). A második oszlopban az állomány tulajdonosának azonosítója található (UID). Mivel az én azonosítóm 500 és a véletlen törlés 21-én, szerdán következett be, úgy tűnik, hogy a keresett állomány a felsoroltak valamelyike. Továbbá arra is emlékszem, hogy a fájl 77 bájt hosszúságú volt. Mivel fájlnevek nincsenek a listán, minél több adat áll rendelkezésre az eredeti állományról, annál könnyebb a dolgunk. Még mindig a `debugfs` parancssoránál maradva, beírom a következő sort:

```
debugfs: dump <157> /home/marcel/cabs
```

A 157-es szám a fájlleíró, a /home/marcel/cabs pedig egy állomány egy másik, már csatlakoztatott fájlrendszerben. A `dump` parancsot használva a `debugfs`-ben a 157-es fájlleírójú állomány tartalmát átirányíthatom egy új, ***cabs*** nevű állományba. Nézzük a művelet eredményét:

```
$ cat /home/marcel/cabs
MiØrt is pr bÆlkoznÆnk j  Cabernet
eliÆll tÆsÆval, amikor meg is vÆsÆrolhatjuk
Henri's Fine Wines  zletØben?
```

Mais non! A titok felfedve!

Egy másik eszköz, ami hasznosnak bizonyulhat törölt állományok visszaállításánál, *Tom Pycke* ***Recover*** nevű programocskája. A ***Recover*** a törölt állományok akár hihetetlenül

nagyra duzzadó listájával való munkát könnyíti meg, sőt, szinte fájdalommentessé teszi. Töltsd le a legfrissebb forrásállományt, a (➲ http://recover.sourceforge.net/linux/recover) csomagold ki és fordítsd le:

```
tar -xzvf recover-1.3.tar.gz
cd recover-1.3
make
make install
```

A program a `recover` parancs begépelésével indul, ezután a rendszeren elérhető különböző fájlrendszereket keresi meg, illetve a megfelelő fájlrendszer parancssorban történő megadásával is indítható. Ezt egy kérdés-felelet rész követi, amellyel a törölt állományok hatalmas listáját szűkíthetjük kezelhető méretűre. Egy példa:

```
# recover /dev/hdd6
Recover v1.3 by Tom Pycke
<Tom.Pycke@advalvas.be>

Getting inodes (this can take some time)...
debugfs 1.14, 9-Jan-1999 for EXT2 FS 0.5b,
95/08/09

In what year did you delete the file? (eg.
1999): 2001
```

Ezután a program megkérdezi a hónapot, amelyet a keltezés- és időtartam, valamint az állomány lehetséges méretének alsó és felső határa követ. A következő kimenetet kaptam:

```
=> 158 78 JUN THU 21 4:38:39 2001
=> 157 77 JUN THU 21 4:40:25 2001
2 inodes found. Where shall i dump them?
(directory):  /tmp
```

A kapott állományok a `dump157` és `dump158` névre hallgatnak. Ezután nincs más teendőnk, mind a `cat`-tel megvizsgálni a tartalmukat. Gondolva rá, hogy esetleg futtatható állományokról van szó, érdemes lehet a `file` paranccsal ezt előtte megvizsgálni. Mint jól tudjuk (és ahogyan monsieur Roget is rájött), a biztonságnak más nézőpontja is létezik. Megnyugtat a tudat, hogy bármikor vissza tudunk állítani egy véletlenül törölt állományt az `ext2fs` fájlrendszerünkben, de mi a helyzet az ellenkező oldalról nézve a dolgot, ha például az állomány tartalma szigorúan titkos, és nem akarjuk, hogy az életben még egyszer valaki belepillanthasson? Ha az állományt inkább valami nagyon sötét, a szőlőművelésre teljesen alkalmatlan vidékre szeretném száműzni? Amennyiben rendszereden megtalálható a ***fileutils*** csomag legújabb kiadása (olyan programokkal, mint a `cp`, `mv` stb.), máris kéznél van a megoldás. Ebben található a `shred` nevű program, amely olvashatatlanná szabdalja az állományt:

```
$ shred secret_instructions
```

Ebben a formában kiadva a parancsot a `secret_instructions` állomány tartalma valóban titokban marad, maga az állomány azonban nem törlődik, csak összekeveredik, mint a jól felvert tojáshab. Igazi biztonságérzet azonban inkább a parancs alábbi formájával érhető el:

```
$ shred -z -u secret_instructions
```

Ennek hatására a program nullákkal írja felül az állományt (a `-z` kapcsoló hatása), majd törli (`-u`). Titkainkat ezennel biztonságba helyeztük.

Konyhai felfedező utunkat ma néhány könnyű, a biztonsági mentéssel kapcsolatos javaslattal kezdtük. Tételezzük fel, hogy adatainkat rendszeres időközönként szalagra mentjük amelyeket azután más, biztonságos helyre szállítunk. Így azokat a titkos állományokat is tároljuk, amelyeket oly elszántan igyekszünk a rendszerünkről eltávolítani. Tegyük fel, hogy a `tar` paranccsal elmentettem az /etc/profile állományt:

```
tar -cvf /tmp/profile.tar /etc/profile
```

Egy ily módon előálló állomány meglehetősen nyílt:

```
$ more /tmp/profile.tar
etc/profile0100644000000000000000000000001163072
353262310117170
ustar  rootroot
# /etc/profile
```

Amint látható, a tárállományban lévő fájlok tartalma könnyedén olvasható. Természetesen tudom, hogy az állományokat egy pofonegyszerű

```
tar -xvf filename.tar
```

paranccsal egyszerűen ki is csomagolhatnám, de most mást szeretnék megmutatni. Mi lenne, ha kódolnánk az tárállományt? Éppen ez a gondolat hívta életre *Brian Wagener* és *Katrina Illari* ***Sectar*** nevű programját, amely a GNU `tar` biztonsági kiterjesztése. Könnyen hozzájuthatunk a ➲ sourceforge.net/projects/star címről.

Két csomagra lesz szükségünk: az első a GNU `tar` javított változata, a második maga a `sectar`. A csomag kibontásával és fordításával kell kezdenünk:

```
tar -xzvf tar_sectar.tar.gz
cd tar-1.13
make
make install
```

Ez majdnem megegyezik az eredeti `tar`-csomaggal, amely a rendszerünkön található, kivéve egy nagyon fontos beállítási lehetőséget. Ha kiadjuk a

```
/usr/local/bin/tar -help
```

parancsot, az alábbi sort is láthatjuk:

```
-e, --encrypt    filter the archive through
                 encryptor
```

E tulajdonság használatához azonban a `sectar` csomag lefordítása is szükséges. A legújabb forrást használva az alábbi lépéseket követtem:

```
tar -xzvf sectar-1.02.tar.gz
cd sectar-1.02
make
make install
```

Ezt a most felfedezett titkosítási lehetőséget kihasználva újra előállítom szigorúan titkos tárállományomat:

```
$ tar -ecvf /tmp/profile.tar /etc/profile
The 256 bit key created is keyfile.993515072
```

Most ezt a kulcsállományt (amely a munkakönyvtáramban található) lemezre vagy egyéb biztonságos helyre másolom addig az időpontig, amikor majd vissza kell állítanom az állományt. Ha a kulcsállomány rendelkezésre áll, az adatok `tar`-állományból való kicsomagolása gyerekjáték, hiányában viszont az adatok titkosak maradnak. Ha a kulcsállomány nélkül nézzük át az állományt, zavarosnak, olvashatatlannak és nyomtathatatlannak találjuk.

Örömömmel láttunk vendégül benneteket, a következő alkalommal se felejtsetek el csatlakozni hozzánk! A votre santé! Bon apétit!

Marcel Gagné (mggagne@salmar.com) Mississaugaban (Ontario, Kanada) él, a Salmar Consulting Inc. rendszerépítéssel és hálózati tanácsadással foglalkozó cég elnöke. A Világhálón elérhető honlapján sok hasznos dolog ➲ http://www.salmar.com/marcel/ lelhető.

Roget fogalomtára ➲ http://www.thesaurus.com

Microlite BackupEDGE 01.01.08-as kiadás

A Microlite BackupEDGE nevű terméke kiváló, biztonsági másolatot készítő program, amely könnyen kezelhető parancssoros, illetve menüs felülettel rendelkezik. Hálózaton keresztül is készíthetünk vele biztonsági másolatot, a rendszer leállása esetén pedig a segítségével visszaállíthatjuk az eredeti állapotot. Leírása teljes és részletes. A BackupEDGE versenyképes árú szabadalmaztatott termék, mely a Linux rendelkezésre álló segédprogramjait használja, például az RSH-t, az SSH-t és a crontabot, ezáltal jobban együttműködik más alkalmazásokkal.

Telepítés

A BackupEDGE CD-n jelenik meg, de ha a számítógépünk nem rendelkezik CD-olvasóval, egy másik gépen a CD-ről telepítőlemezeket készíthetünk, akár Windows alatt is. FTP segítségével hatvannapos próbaváltozatot tölthetünk le, és ha úgy döntünk, hogy tovább szeretnénk használni, ezért megvásároljuk a BackupEDGE-t, a Microlite elküldi a megfelelő kulcsot – és a program teljes értékűvé válik.

Előnyök

- Könnyen kezelhető felület
- Ingyen kipróbálható
- Remek telepítőprogram
- Egyszerű helyreállítás
- Részletes leírás

Hátrányok

- Színvilágát néha az asztalon hagyja
- Nincs más felhasználók címét tartalmazó lista

A telepítő könnyen használható, a programhoz hasonlóan parancssoros. A kurzort nyilakkal mozgathatjuk, amire azonban a tabulátor nem mindig „hajlandó”, és ez kissé zavaró. Szimpatikus viszont, hogy szöveges üzemmódban a számítógép INSERT billentyűjével váltogathatunk beillesztés és felülírás között – ezt használtam, amikor a meghajtó fajtáját a neve elé írtam. A súgó már a telepítésnél is rendelkezésre áll.
A telepítő felismeri, hogy milyen meghajtók vannak a gépben; az én két SCSI-szalagos meghajtómat valószínűleg a */proc* állományrendszer segítségével azonosította be. A telepítő azt is képes eldönteni, hogy a szalagos meghajtó tud-e gyorskeresést végezni, és ezt milyen megbízhatóan teszi.
A gyorskeresés többek között kis számú fájl egyidejű visszaállítására használatos. Az általam Linuxon látott telepítők közül ez ismeri fel a legjobban a meghajtókat. Telepítéskor egy háttéralkalmazás fut, amely a „ritka” fájlokat ellenőrzi (egy „ritka” fájl lyukakat tartalmaz: nullabájtok sorozatát, amelyek nem foglalnak fizikai lemezblokkokat – a Microlite ezeket „látszólagos fájloknak” nevezi). A „ritka” fájlok helyes kezelésével visszaállításkor temérdek tárhelyet szabadíthatunk fel, viszont az ilyen fájlokat tartalmazó rendszerek rossz kezelése következtében a biztonsági másolatot készítő alkalmazások használhatatlanok lehetnek. Mindkét számítógépen, amelyet használtam, a kereső-program sajnos megállt, és csak a hibaüzenet számát volt hajlandó közölni.
A BackupEDGE támogatja a nyers tárterületeket is, ami az adatbázis-kiszolgálók számára igen hasznos. A telepítő egy ikont helyezett el KDE-asztalomon, az ezáltala hívott parancssor apró módosításaival beállíthatjuk a betűméretet, ami fontos lehet a régi linuxosok számára.

A program

Telepítés után parancssorból indítottam az edgemenu nevű programot. A színvilág: a kék háttér szürke karaktereivel leginkább a régi DOS-os időkre emlékeztet. Kétféle színösszeállítás közül választhatunk: világoskék-szürke vagy fekete-fehér. Az edgemenu néha kilépés után is otthagyja színeit mind a KDE-ben, mind az xtermben. Nem nagy ügy, elviselem, ha a program üzembiztos.
A program csak parancssorosan is működtethető, a `man edge` parancs minden lehetőséget listáz. Mivel a konzolos menüprogram felületet ad a parancssoros programnak, megfigyelhetjük, hogyan működnek a különböző parancsok. Egyszerűen tudunk önműködő mentéseket időzíteni. Az alkalmazás a rendszergazda crontab könyvtárába menti a biztonsági másolatokat, így könnyen változtathatunk a másolatokon, megkönnyítve más időzített alkalmazások használatát.

Adatok

Gyártó: Microlite Corporation
E-mail: sales@microlite.com
URL: ➲ http://www.microlite.com
Ár: 90–450 dollár

Biztonsági másolat készítése

Először egy kisebb, 9 MB-os adatmennyiségről készítettem biztonsági másolatot: a */etc* könyvtárról. Az első próbálkozás nem volt sikeres, gyanítom, a „ritka” fájlok háttérben folyó keresése akadályozta meg az SCSI adapteren. Újraindítás után mindenesetre már mindkét szalagos meghajtóra képes voltam másolatot készíteni.
A gépet újra kellett indítanom, ugyanis két gépet használtam, hogy kipróbálhassam a hálózaton keresztüli másolat készítését (a Microlite „távoli másolatkészítésnek” nevezi). Mivel az SSH már működött a két gép között, a BackupEDGE-et beállítottam, hogy ezt használja – ezáltal jelszó nélkül is biztonságos kapcsolat létesíthető. Beállítottam az ügyféloldalt, azonban a kiszolgáló leállt. Hoppá! Újraindítottam a kiszolgálót, majd másolatot készíthettem rajta, ekkor viszont az ügyfél viselkedett az előbbi módon. Másodjára legalább a kiszolgáló nem fagyott le, ami már haladás.
Ez a fajta másolatkészítés bármely két gép között működik. Menet közben sokat kell várni, úgy tűnik, az RSH- vagy az SSH-kapcsolat nem folyamatos.

Ellenőrzés

A másolatot összevethetjük az eredetivel, vagy ha már módosult, használhatjuk a CRC-ellenőrzőösszeget. Ez arra is alkalmas, hogy az olcsóbb szalagos meghajtók, mind például néhány QIC-termék fejállását is ellenőrizzük. Nagyon jó, hogy az ellenőrzés a biztonsági mentés folyamatának részeként is végrehajtható.

Az eredeti állapot visszaállítása

A visszaállítási folyamat az edgemenu programból könnyen végrehajtható, és ezt mindhárom felületen megtehetjük. Grafikus felületen az edgemenuból

indítható ***edge.emx***-et használhatjuk, ami a fájlok egyenkénti visszaállítására szolgál. A könyvtár kiválasztásával a benne található állományokat és jegyzékeket is kijelöljük. Egyszerűen válasszunk ki egy adatbázist és kattintsunk rá. A könyvtár faszerkezetén lefelé haladva kattintsunk minden állományra, amelyet ki szeretnénk választani. Kattintsunk a ***Transfer*** menüpontra, így a kijelölt adatok a restore ablakba kerülnek. A ***Restore*** menüpont kiválasztásával az adatokat eredeti állapotukba állítjuk vissza.

Leírás

A kézikönyv vastag, több mint 270 oldalas, szerencsére nagyméretű betűkkel szedett, így könnyen olvasható. Hetven oldal a súgó másolata, míg a Linuxtól eltérő operációs rendszereknek néhány oldalt szenteltek. A következő 130 oldalas rész a rendszerleállás utáni visszaállítást végző programmal foglalkozik. Minden fejezethez részletes tartalomjegyzék tartozik, bár a tárgymutató kissé hiányos: nem szerepel benne például a „ritka” fájl, tehát a felhasználónak tudnia „kell”, hogy a Microlite „látszólagos fájlnak” nevezi őket. Mindent egybevetve a leírás teljes, igen részletes és olvasmányos – a Microlite ezért ötös osztályzatot érdemel.
A leírás arról is tájékoztat, hogyan lehet a BackupEDGE egyes lehetőségeit személyre szabni. A távoli másolatkészítés például RSH-t használ, de azt is részletesen leírják, mit kell tennünk, hogy SSH-t alkalmazza.

Terméktámogatás

A felhasználót levélben, telefonon, faxon és a weboldalukon keresztül is segítik. Nem található sajnos az összes felhasználó levélcímét tartalmazó lista, pedig ezáltal könnyen megoszthatnák egymással a tapasztalataikat. A kipróbálás során ezt a szolgáltatást mindössze egyszer kellett igénybe vennem, amikor az RSH helyett SSH-t szerettem volna használni, de nem sikerült működésre bírnom. Valószínűleg csak azért alakult így, mert belefáradtam a hibakeresésbe. A kapott tanács udvarias volt, de kissé felületesnek tűnt. Sűrű levélváltás után talán működött volna az SSH, de időközben elment a kedvem az egésztől.

Rendszerleállás utáni visszaállítás

Rendszerünket vészleállás után valószínűleg könnyen vissza lehet állítani a kapott RecoverEDGE programmal, ha sikerült beállítanunk. Azért írtam, hogy valószínűleg, mert egyszer sem próbáltam ki ezt a lehetőséget. A HP OBDR névre hallgató, rendszerindításra is képes szalagos meghajtójával vagy akár hajlékony lemezekkel rendszermásolatot tartalmazó szalagot készíthetünk. Ha a gépünk leáll, indítsunk a lemezről vagy a szalagos meghajtóról, és újból működni fog.
Ha rendszerünket nagyobb merevlemezre állítanánk vissza, a RecoverEDGE megváltoztatja a lemezrések méretét, hogy ne maradjon kihasználatlan lemezterület. A RecoverEDGE használatával a rendszer-visszaállítás hálózaton keresztül is lehetséges.

Charles Curley
szabadúszó programtervezőként és újságíróként tevékenykedik Wyomingban, ahol időnként tehénpásztori feladatokat is ellát (➲ http://w3.trib.com/~ccurley).