Egyéves a Linuxvilág!

Ünnepelj, Világ! Megértük az első születésnapunkat! A születésnapokon az embert a nagy ünneplés mellett mindig utoléri valamiféle borongós hangulat is. Egy év. Sok vagy kevés idő egy lap életében? Honnan indultunk és meddig jutottunk? Most nem csak a Linuxvilágra gondolok, hanem a magyarországi GNU/Linux-közösségre. Hogy a nemrég megtartott Internet Konferencia mottóját idézzem: „Előreszaladtunk vagy lemaradtunk?" Gondoljuk csakmeg, egy évvel ezelőtt a világ is más volt. Ma már természetes, hogy egy nagyobb hálózatban GNU/Linux van. Sőt, manapság már az is természetes, hogy asztali gépnek is használható, igaz, csak ott ajánlott, ahol legalább egy Linux-hívő akad a közelben. Akkoriban még a csodájára jártak, ha valaki azt mondta: „Ne haragudj, nem használok Windowst". Igaz, abból, hogy valaki GNU/Linuxot használt, egyértelműen következett, hogy programozó és jól ért angolul. Ma már a legtöbben nem is akarnak irodai munkára olyan Linuxot használni, amelyiknek a felülete nem magyar.

És az üzleti háttér? Ha jól emlékszem, négy évvel ezelőtt az ELTE levelezési listáján azon vitatkozgattunk, vajon elég erős-e a Linux a „nagy ellenfél" legyőzésére. A listán azt írtam, hogy a Linuxnak csak akkor van esélye, ha komoly „marketing" épül fel mögé. Az ellenoldal véleménye tömör volt: „Soha! Az már nem Linux lenne!"

Ma már mindenki számára világos és egyértelmű: nem az a cél, hogy valamiféle magasztos harcot vívjunk, a jelenlegi állapotot a piac követelte ki, saját maga hozta létre. Már elindultunk valamerre, és nem tudunk (már nem is akarunk) visszafordulni. Cégek épültek a Nyílt Forrás Közössége köré, szolgáltatások tömkelege érhető el kereskedelmi szinten, és lássuk be, a piac igényli is, hogy ilyen cégekkel köthessen szerződéseket. Nem véletlen, hogy az IBM, a HP és a többi előrelátó „Nagy" is komolyan támogatja a Linuxot.

Sarkalatos pont az oktatás is. Négy éve csak egy nagyon szűk réteg került közel a Unixokhoz, nem volt nagy igény rá, hogy tömegek ismerjék és használják komoly szinten ezeket a rendszereket. Ma már, ha valaki azt mondja, hogy rendszergazda, akkor legalább egy GNU-rendszert ismernie kell. Az új helyzet lassan, de biztosan az oktatást is rákényszerítette a változásra. Szerencsére a SuliNet elindulásával egyre több helyen van Internet, így egyre több területen szűnt meg a legnagyobb nehézség, a folyamatos kapcsolat hiánya. És hadd emeljem ki, hogy külön köszönettel tartozunk azoknak a lelkes és vállalkozó szellemű tanároknak, akik neki mertek vágni az újnak. Hogy miért őket emelem ki? Az indok egyszerű. Azok közül, akikkel e témáról beszéltem, a legtöbben azt vallották, hogy a tanárok „nagyon elfoglaltak, nem törődnek vele, hogy a diák, ha kikerül az iskolából, használható tudással legyen felvértezve és amúgy sem mernek olyat oktatni, amihez nem értenek". Sajnos rossz példa is akad, de szerencsére azt tapasztaljuk, hogy egyre több tanár kedveli meg és oktatja kedvencünk használatát.

Szóval, komolyan haladunk. Kezdünk kikupálódni, olyannyira, hogy már olyan nagy cégekről is hallani lehet, akik döntő többségben GNU/Linux-rendszert használnak. Röviden összefoglalva: ma már nem tagadhatják le a Pingvin létjogosultságát.

Gyorsan vissza is térek a lapra. Ha az első számot összehasonlítjuk a mostanival, azt hiszem, mindenki számára szembetűnő a különbség. Nem csak a küllemre gondolok. Természetesen nagyon sok gyermekbetegség ütötte fel a fejét, igyekeztünk kikúrálni magunkat, több-kevesebb sikerrel. Az első pár szám után például kaptunk olyan véleményeket, amelyek szerint igazán több hazai cikket is rakhatnánk a lapba. Örömmel írhatom, hogy az újságunkban szereplő cikkek jelentős hányada már hazai szerzők tollából származik, és nyugodtan hozzátehetem, elérik a nemzetközi lapok cikkeinek a színvonalát, sőt, gyakran túl is szárnyalják azokat. Úgyhogy ezúton ragadom meg a lehetőséget, és minden hazai szerzőnek is szívből köszönöm az együttműködést!

A közösség

Amikor a lapot elindítottuk, azt reméltem, hogy egyedi terméket tudok létrehozni. Egy olyan magazint, ami ténylegesen használható és élő. Élő, abból a szempontból, hogy a szerkesztőség nem úgy készíti, hogy bevonul egy toronyba, elzárkózik az olvasók elől, majd a lőrésen át kidobja a készterméket, hanem ténylegesen segíti a felhasználókat, amennyire tud, fórumot biztosít nekik és részt vesz egy közösség kialakításában. Sokszor, megtörtént hogy a dolgok nem egészen úgy mentek, mint ahogyan szerettük volna, és nagyon jólesett, amikor támogató, fejlesztő szándékú leveleket kaptunk. Az egész csapat nevében szívből köszönöm a véleményeket! És nemcsak a dicséretekre gondolok, hanem arra a sok-sok levélre is, melyekben az olvasók a hibákra mutattak rá. Ezek nélkül a sötétben tapogatóznánk és mi magunk sem hinnénk el, hogy tényleg értelme volt a munkánknak. Számomra még egy érdekes tapasztalata van ennek az egy évnek. A legtöbb felhasználó a linuxos közösséget egyfajta baráti társaságnak tekinti. És ez nagyon jó! Igaz, vannak hátulütői, például többször tapasztaltam az LME-n belül, hogy személyes ellentétek miatt nagyon jó ötletek és kezdeményezések kerültek a küszöbre, de ezek az apró összezörrenések, remélem, idővel elcsitulnak.

A tervek

Fontos tudni, hogy merre tartunk. Többen kérdezték, hogy így, az első év után mi lesz velünk? Megszűnünk és eltűnik az újság, megcélozzuk a kezdő felhasználókat, és igyekszünk a zavarosban halászni, vagy csak rendkívül magasröptű cikkekkel töltjük fel a lapot? Nos, amíg tehetek ellene, egyik sem következik be. A tervek szerint igyekszünk a lapot megerősíteni, még több érdekes és hazai cikkel feltölteni, és próbálunk időben megjelenni. Igen, sajnos ez nagy hiányossága a lapnak, de remélem, ezt is kinőjük, mint annyi más hibát. Köszönöm még egyszer, hogy oly sokáig velünk tartottak, jó olvasást és linuxozást kívánok!

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

Programvadászat

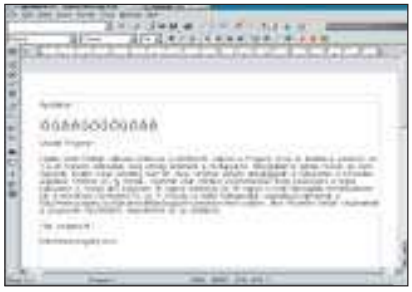
Debian/GNU Woody 3.0 próbaváltozat

Egy éve, amikor először tárult olvasóink elé a Linuxvilág magazin, a Debian akkori friss változatát, a Potato 2.2r0-t adtuk közre. A Debiant használók sajnos már hozzászoktak a lassú változáshoz, így az sem meglepő, hogy egy év alatt sem készült el a következő végleges változat. Mindezek ellenére a mi mostani CD-nkre a Woody-próbaváltozat első két CD-jét tesszük. Bár számos olvasói levelet kaptunk, hogy ne tegyük, mert a Debian csak akkor Debian, ha már a végleges változatot köszönthetjük benne. Ezzel egyfelől egyetértek, másfelől viszont nem, mivel én már több mint fél éve a Woodyt használom; a csomagok sokkal frissebbek, mint a Potatóban, és sok újdonsággal melengeti a felhasználók szívét, és ha az embernek nem az életét kell rábíznia, akár kiszolgálóként is nyugodtan használható. A Woodyban több ezer program közül válogathatunk, melyek között bármilyen feladatra megtalálhatjuk a megfelelőt (természetesen, azért vannak kivételek!). A legegyszerűbb megoldása annak, hogy a rendszerünket frissen tartsuk, és programokat telepíthessünk az `apt` program használata. Segítségével akár CD-ről, akár hálózatról és az Internetről is telepíthetünk csomagokat azok függőségeivel együtt. Az *1. listán* látható pár sor a saját */etc/apt/sources.list* fájlomból való – látható, hogy a mellékletekként megjelenő három CD a fájl elején szerepel, ezután következik a Debian-tükör a Linuxvilág kiszolgálóján, itt mind a ***testing*** mind pedig a ***debian-non-US testing*** könyvtárakat érdemes megadni, mivel néhány program csak így érhető el.
Listánkhoz CD-lemezt az `apt-cdrom add` paranccsal adhatunk.
Miután az apt programnak megadtuk a csomagok helyét, az adatbázist, frissíteni kell, amit az `apt-get update` paranccsal tehetünk meg. Ennek a 2. listán láthatóhoz hasonló kimenete lesz.

Ha egész rendszerünket frissíteni szeretnénk, az `apt-get upgrade` paranccsal tehetjük meg. Ekkor a rendszerünkben lévő összes olyan csomag frissítésre kerül, amihez valamilyen hibajavítás vagy frissítés jelent meg.
A programok telepítését az `apt-get install csomagnØv` paranccsal végezhetjük el.
Ha valakinek Potato rendszere van, miután a CD-ket hozzáadta a forráslistához, régi Debianját Woodyra frissítheti az `apt-get dist-upgrade` parancs segítségével.
A 68. oldalon összefoglalót olvashatnak a Woodyról rövid telepítési útmutatóval.
Remélem, olvasóink örömmel fogadják ezt a születésnapi ajándékot.

A harmadik CD

A harmadik korongon a legfrissebb rendszermagok szerepelnek, ezek jelen pillanatban: a 2.4.14-es és a kissé elhanyagolt 2.2.20-as. A 2.2.x sorozatú rendszermagot annyira mostohán kezelik, hogy a ➲ http://www.kernel.org honlapján a kiadásról szóló hír meg sem jelent.
Az OpenOffice 638c is felkerült, ez az alapja a Sun StarOffice 6 próbaváltozatának, a jelek szerint már a Sun is elég meg-



1. lista A sources.list fájl egy éles rendszeren

```
deb cdrom:[Debian GNU/Linux 3.0 _Woody_ - fsn.hu's i386 Binary-3 (20010907)]/ unstable
➥contrib main non-US/contrib non-US/main non-US/non-free non-free

deb cdrom:[Debian GNU/Linux 3.0 _Woody_ - fsn.hu's i386 Binary-2 (20010907)]/ unstable
➥contrib main non-US/contrib non-US/main non-US/non-free non-free

deb cdrom:[Debian GNU/Linux 3.0 _Woody_ - fsn.hu's i386 Binary-1 (20010907)]/ unstable
➥contrib main non-US/contrib non-US/main non-US/non-free non-free

deb ftp://ftp.linuxvilag.hu/ testing main contrib non-free
deb ftp://ftp.linuxvilag.hu/debian-non-US testing/non-US main contrib non-free
```

2. lista Új források megadása után frissítenünk kell az adatbázist

```
sharon:/etc/apt# apt-get update
Get:1 ftp://ftp.linuxvilag.hu testing/main Packages [1461kB]
Get:2 ftp://ftp.linuxvilag.hu testing/main Release [81B]
Get:3 ftp://ftp.linuxvilag.hu testing/contrib Packages [41.0kB]
Get:4 ftp://ftp.linuxvilag.hu testing/contrib Release [84B]
Get:5 ftp://ftp.linuxvilag.hu testing/non-free Packages [68.4kB]
Get:6 ftp://ftp.linuxvilag.hu testing/non-free Release [85B]
Get:7 ftp://ftp.linuxvilag.hu testing/non-US/main Packages [59.8kB]
Get:8 ftp://ftp.linuxvilag.hu testing/non-US/main Release [88B]
Get:9 ftp://ftp.linuxvilag.hu testing/non-US/contrib Packages [927B]
Get:10 ftp://ftp.linuxvilag.hu testing/non-US/contrib Release [91B]
Get:11 ftp://ftp.linuxvilag.hu testing/non-US/non-free Packages [3651B]
Get:12 ftp://ftp.linuxvilag.hu testing/non-US/non-free Release [92B]
Fetched 1635kB in 3m48s (7144B/s)

Reading Package Lists... Done
Building Dependency Tree... Done

sharon:/etc/apt#
```

bízhatónak tartja a próbára. A StarOffice 6 beta a Sun honlapjáról tölthető le.
➲ http://www.sun.com/software/star/staroffice/6.0beta/get.html
CD-nkre a Mandrake 8.1-es 3. CD-jéről származó KDE-magyarítás és más egyéb szabadon terjeszthető programok is felkerültek. Mivel ezen a CD-n nemcsak szabadon hozzáférhető programok vannak, sajnos nem áll módunkban teljes egészében közreadnunk, sok hasznos és szükséges program azonban felkerült. A legtöbben bizonyára a magyarításoknak fognak örülni. A ***kde-i18n-hu-2.2.1-2mdk.noarch.rpm*** csomag ahhoz szükséges, hogy a KDE magyarul szóljon hozzánk, ezenkívül az összes ***kde-i18n*** csomag is szerepel, amely a KDE felületét rendkívül sok nyelven teszi elérhetővé. A korongon magyar nyelvű HOGYAN-ok is helyet kaptak html, ps, és txt formátumban is. Megtalálhatjuk, hogyan készítsünk indítólemezt, valamint a CD-írás HOGYAN-jait, a DOS- Windows-ról Linuxra történő áttérés HOGYAN-jait, a DOSEMU HOGYAN-okat, továbbá a Framebuffer, a Glibc2, a rendszermag, az MP3 és a NIS HOGYAN-okat. Ezeken kívül a súgóoldalak (man-oldalak) egy részének magyarítása is szerepel a lemezen. Ezek sok segítséget tudnak nyújtani a kezdő felhasználók számára, bár a leírások és segédletek fordítása sajnos még nem teljes, így sok dolgot csak angol nyelven tudunk elérni. Mindenesetre indulásnak nem rossz! A ***XFree86-ISO8859-2-Type1-fonts-1.0-15mdk.noarch.rpm*** csomagban a közép-európai, így a magyar Type1-es betűkészleteket találhatjuk, ezzel is bővítve magyar nyelvű lehetőségeinket a grafikus felületen. A betűkészletek telepítésére és beállítására a ***kfontinst*** programot használhatjuk, segítségével TrueType és Type1 készletet tehetünk elérhetővé rendszerünk és programjaink számára.
A Nessus nagyon kifinomult felderítőeszköz, olvasóink már olvashattak róla a ***Linuxvilág*** június-júliusi számának 52. oldalán kezdődő „Ellenőrzés pásztázókkal (2. rész) – Nessus” című cikkben. Ha kíváncsiak vagyunk saját biztonságunkra vagy a mások által hagyott lyukakra, továbbá arra, hogy milyen kapuk várják a támadókat és a kellemetlenkedőket, akkor futtassuk le a Nessus programot, melyet kényelmes grafikus felülete könnyedén beállíthatóvá tesz.
A ***cdrdao*** csomag segítségével hang-CD-inkről készíthetünk „nyomatot”, amit egy másik CD-re írhatunk rá. Ez különösen akkor előnyös, ha olyan korongokat akarunk másolni, amelyeken nincsenek szünetek a számok között, mint például koncertfelvételek esetében.
A ***ksetiwatch*** programot „UFO-k figyelésére” használhatjuk. Kinézete szinte teljesen megegyezik a windowsos változatéval.

Netscape 6.2

A Netscape böngésző is fejlődik, sebességét tekintve nekem határozottan az volt az érzésem, mintha gyorsult volna. A különféle kiegészítők telepítése hibátlanul zajlott le, így egy Java és Flash kezelésére is alkalmas böngészőt kaptam. Aki megfelelő erőforrással rendelkezik, a régi Netscape-sorozatról nyugodtan áttérhet erre a változatra.

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

1 éves a Linuxvilág

A magyar Linux-barátok magazinja

Köszönjük minden olvasónknak a bizalmat, a türelmet és a lelkes biztatást!

Linuxvilág-ajándék
Évfordulónk alkalmából hűséges olvasóinkat egy kis ajándékkal szeretnénk meglepni: előfizetőinknek, akik lejárt előfizetésüket 2001. november 31-ig meghosszabítják (vagy már korábban meghosszabították), Linuxvilág-pólót küldünk.

Születésnapi sorsolás
Azon új előfizetőink között, akik a novemberi számtól kezdve fizetnek elő a Linuxvilág magazinra, plüsspingvineket, pólókat, valamint egy 10 000 forint értékű könyvutalványt sorsolunk ki. A sorsolásban résztvevő új előfizetők megrendelésének határideje: 2001. november 31. A sorsolás eredménye a decemberi számban, valamint a Linuxvilág honlapján kerül kihirdetésre.

Mandrake Linuxhoz is „apt”

Biztosan sokan ismerik az `apt`-ot, a Debian-rendszerekben megjelent csomagkezelő programot. Most már Mandrake alá is elérhető, így aki ezt a kényelmes és hatékony megoldást szeretné használni, ezentúl megteheti. Az RPM-csomag az alábbi címről tölthető le, a csomagokat telepítés és beállítás után a Debianban megszokott módon frissíthetjük, tehát `apt-get update`, `apt-get upgrade`. Egyelőre még csak a fejlesztői változathoz érhető el.

➲ ftp://ftp.rpmfind.net/linux/Mandrake-devel/contrib/RPMS/apt-0.3.19cnc51-1mdk.i586.rpm

Microsoft az Opera ellen?

Az Opera készítői szerint a Microsoft az Opera böngészőt használó internetezőket kizárta az MSN oldalairól. A fejlesztők úgy fogalmaztak, hogy nyilván jó úton haladnak, az Opera a jelek szerint már veszélyt jelent az Internet Explorerre nézve. A kizárt felhasználók között volt *Jon S. von Tetzchner*, az Opera Software ASA ügyvezetője is. A Microsoft szerint azonban nem kizárásról volt szó, hanem az Opera egyik hiányossága miatt nem lehetett megjeleníteni az oldalakat. Ez a hiányosság pedig az XHTML-szabvány kezelése. Az ügyvezető szerint ez viszont teljességgel lehetetlen, mivel az Opera a lehető legjobb szabványtámogatással rendelkezik.

➲ http://www.opera.com/pressreleases/

VMware 3.0

Úgy látszik, mostanában köszöntött be a 3.0-s változatszámú termékek megjelenési időszaka. Közéjük tartozik a VMware is.

Támogatja a Microsoft Windows XP operációs rendszert, az USB-s eszközöket, a DVD-olvasókat, beépített NAT-támogatást (Network Address Translation) tartalmaz a könnyű hálózatba kötéshez, sokkal egyszerűbb és testreszabhatóbb

beállításokkal rendelkezik, valamint az erőforrás-kihasználása is jobb.

A támogatott gazda operációs rendszerek: Linux, Microsoft Windows NT 4.0, Windows 2000, Windows XP.

Az alábbiak pedig futtathatók vele (teljes támogatás):

Linux, Microsoft Windows XP, Windows 2000, Windows NT 4.0, Windows Me, Windows 98, Windows 95, Windows 3.1 és MS-DOS 6.

➲ http://www.vmware.com

OpenBSD 3.0

December elsején jelenik meg az OpenBSD 3.0. Sok újdonságot tartalmaz, amelyek közül talán a PF nevű csomagszűrőkód a legérdekesebb. A régebbi változatokat használóknak az átállás előtt mindenképpen érdemes tanulmányozniuk a leírásokat. A rendszer része az XFree86 4.1.0 és 3.3.6, így a legújabb grafikus kártyákat is használni tudjuk.

A kiadás tíz különféle számítógép-felületre lesz elérhető: i386, Alpha, MacPC, Amiga, HP300, mvme68k, Mac86k, VAX, Sparc és Sparc64. A rendszer jellemzői közé tartozik még a kiterjedtebb programkínálat, amely a fejlesztők szerint a 2.9 óta sokat fejlődött.

➲ http://www.openbsd.org/

➲ http://www.openbsd.org/orders.html

Linuxos tudáspróba

Aki kíváncsi, mennyire is van otthon a Linuxban, annak érdemes az alábbi címre ellátogatnia, és ott egy 25 kérdéses vizsgalapot kitöltenie. Akad egy-két egészen elgondolkodtató kérdés is!

➲ http://www.unixreview.com/documents/s=1780/urm0111c/0111c.htm

Még egy születésnap

November 3-án 30 éves lett a Unix. Pontosan harminc évvel ezelőtt adták ki a Unix Time-Sharing System V1-et. Az Open Group's UNIX történeti leírásában olvashatjuk, hogy a Version 1 „a PDP-11/20-akhoz assemblerrel, valamint fájlrendszerrel rendelkezett, ismerte a `fork()`-ot, tartalmazta a ***roff***-ot és az ***ed***-et. Szükség szerint akár szöveges dokumentumok feldolgozására is használhattuk”.

A weboldalon pedig egy teljes Unix-családfát is találhatunk, érdemes körülnézni rajta, hogy meglássuk, milyen családból származik a mi kis pingvinünk!

➲ http://perso.wanadoo.fr/levenez/unix/

➲ http://portal.fsn.hu/index.php

A Kék Óriás tervei a kis pingvinnel

Jelenleg is nagyon sokat hallani linuxos berkekben a Kék Óriásról. Az már tavaly nyilvánvalóvá vált, hogy a cég komoly terveket dédelget a GNU/Linuxszal kapcsolatban, Amerikában egész városokat kiplakátoztak a „Peace, Love, Linux" reklámhadjárat során. Nálunk azért nem minden működik úgy, mint odakint, ezért is örültem, amikor lehetőségem nyílt az egyik legilletékesebb személlyel, *Bartfai-Walcott Katalin*-nal, az IBM Linux-stratégia nemzetközi képviselőjével beszélni.

Szy György: *Kérem, mondja el, milyen viszony alakult ki az IBM és a GNU/Linux között és merre terveznek továbbmenni?*

Bartfai-Walcott Katalin: Amit az IBM a világ felé mutat, nem egyszerű marketingfogás. Ténylegesen örömmel fogadtuk a GNU/Linuxot, és mind a rövid, mind a hosszú távú terveink között szerepel. És hogy hosszútávon az IBM milyen irányt célzott meg? Nos, terveink között az egyik legfontosabb a Nyílt Szabvány (Open Standard) támogatása.

Sz. Gy.: *Hallhatnánk néhány szót az Open Standard hátteréről?*

B. K.: Ahogy a szakma egyre több szabványt kezdett el használni, egyre bonyolultabbá vált olyan eszközöket, programokat fejleszteni, amelyek különböző környezetekben is képesek futni. Az Open Standard lényege, hogy egy olyan gerincet alkosson, ami minden szinten megadja az illeszkedés feltételeit. Itt egy szélesebb látókörű szabványra kell gondolni, ami az alkatrészek és a gép szintjétől az ügyfél által használt eszközökig megadja az egész rendszer alapszerkezetét. A szabvány „nyitottsága" annyit jelent, hogy minden egyes szinten bárki fejlesztheti új elemet, csupán figyelembe kell vennie a szabvány ajánlását. Így ki lehet alakítani egy olyan teljes ügymenetet, ahol nem okoz fennakadást, ha valamilyen alkatrészt szeretnénk cserélni, át kívánunk térni egy erősebb kiszolgálóra vagy esetleg le kívánjuk cserélni mondjuk a levélkiszolgáló programot vagy a személyes adatokat kezelő programot.

Sz.Gy.: *Visszatérve a GNU/Linuxhoz, hol és mire ajánlják ezt a rendszer ügyfeleiknek?*

B. K.: Számos helyen, például olyan esetekben, amikor egy nagygép az „ágyúval verébre" esete volna. Egy kisvállalat fájlkiszolgálójának teljesen felesleges beállítani egy S390-eset. A másik nagyon jó példa az olyan területek, ahol sok-sok operációs rendszerre van szükség. Fejlesztői telepekre, ASP-kiszolgálókra vagy éppen oktatási rendszerekre gondolok. Az előbb említett S390-esen például akár 40 000 Linux futtatható egyszerre.

Sz. Gy.: *Nagyon fontos ilyen helyeken a költség. Mennyi megtakarítást jelenthet ez az ügyfél számára?*

B. K.: Rengeteget! Gondoljunk csak bele, ha például száz operációs rendszert kell üzemeltetni, és egy gépen egy-két rendszer futhat egyszerre, az mondjuk ötven gépet jelent. Itt sokkal nagyobb gépparkot kell figyelni, és több szakemberre is szükség van. Ha ezt hasonlítjuk össze egyetlen S390-essel, amely a sarokban elzümmög, és két rendszergazda képes ellátni, érezhető a különbség.

Sz. Gy.: *A cégen belül mennyien foglalkoznak kifejezetten a GNU/Linuxszal?*

B. K.: Oregonban 250 ember foglalkozik ezzel a vonallal, ez önmagában is komoly szám, valamint ehhez jön még az a körülbelül 1500 ember, akik a föld legkülönbözőbb pontjain találhatók.

Sz. Gy.: *Ennyi ember képes ellátni az összes ilyen típusú feladatot?*

B. K.: Természetesen nem. Itt főleg a belső fejlesztések folynak, illetve a külsősök támogatása. Nagyon sok tevékenységet ugyanis kisebb helyi cégekre bízunk, ugyanis óriási igények merültek fel ezen a területen. Fontos továbbá, hogy megfelelő támogatást tudjunk biztosítani azoknak a fejlesztőknek, akik Linux alá kívánják átültetni programjaikat. Ez az egyik legfontosabb kérdés, az ügyfél ugyanis csak akkor kezd el ilyen típusú rendszereket használni, ha a neki megfelelő programok futnak rajta.

Sz. Gy.: *Hogyan segíti az IBM a vásárlókat? Mennyire nézi, hogy a vevőnek mi mennyibe kerül?*

B. K.: Ez nagyon fontos pont. Az ügyfél akkor elégedett, ha jó terméket kap jó áron. Elsősorban fontos, hogy a TCO-modellt kövessük, hiszen így hosszútávon biztosítani tudjuk a vásárlók elégedettségét. Ez a modell (Total Cost of Ownership – teljes birtoklás költsége) nem azt nézi, hogy az ügyfél mennyit fizet egy alkalommal, hanem hogy az adott rendszer beszerzése, üzemeltetése mennyibe kerül neki összesen. Itt figyelembe kell venni az alkalmazandó szakemberek bérét, a szervizköltségeket stb. Ha egy cégnek komoly költségeket kell fizetnie egy elavult rendszer üzemeltetésénél, az IBM igyekszik olyan megoldást kínálni, amely hosszú távon jóval kedvezőbb az ügyfél számára. Mint már említettem, ezen ajánlatok ma már szervesen építenek a GNU/Linuxra is. Fontos emellett az ügyfeleknek, hogy a nemzeti nyelven legyenek elérhetők a rendszerek. Ezért a honosítás kérdése is folyamatosan napirenden van.

Sz. Gy.: *Az ilyen tervek mellett az oktatás is rendkívül fontos. Az IBM mennyire támogatja a GNU/Linuxot az oktatásban?*

B. K.: Igyekszünk támogatni minden fontos vonalat, oktatóközpontokat üzemeltetünk, vizsgarendszereket alakítottunk ki, és virtuális tanfolyamaink is szép számmal vannak. A tanfolyamok között szerepelnek ingyenesek és fizetősek is. Mivel a GNU/Linux ilyen szervesen jelenlévő nálunk, ezért az oktatási keretünkbe is igyekszünk minél jobban beépíteni.

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

Internettanácskozás Tihanyban

Az idei Internet Konferenciát október legvégén tartották Tihanyban, több mint harminc előadással. A témák szépen felölelték az Internettel kapcsolatos legtöbb izgalmas kérdést – illetve a gazdasági és jogi szempontból izgalmasakat. A látogatók is széles körből kerültek ki: megjelentek az informatika területének (szolgáltatók, webfejlesztők, programozók), a marketingnek és a pénzügyi szférának az emberei, valamint az államvezetés képviselői is.

Az előadások csoportokra voltak osztva és két helyszínen párhuzamosan folytak. Bár számos előadásba igyekeztem belehallgatni, természetesen nem tudtam mindegyiken jelen lenni. Az izgalmasabbakról viszont hadd írjak kicsit részletesebben.

Az első, amit előrebocsátok, hogy szinte mindegyik előadáson előkerült a következő téma: „a .dotcom a csődhullám után", illetve „az Internet-lufi kipukkadása után" stb. Tehát mindenki általánosan siratott egy olyan „paradicsomi" időszakot (az elmúlt évet), amikor minden pénzintézet és kockázati befektető mint az őrült, nyomta a pénzt a szakmába – ellentétben a jelennel, amikor mindenki nagyon is komolyan megnézi, hogy hova teszi a pénzkötegeit.

A második napon találkoztam csak olyan gazdasági elemzőkkel és pénzügyi szakemberekkel, akik azt feszegették, hogy nem a világvége jött el. Az egyik előadó grafikonokat mutatott be, mellyel igazolta, hogy az „Internet-piac" ugyan tényleg komoly visszaesést élt át, viszont ez nem több, mint az „újdonság" varázsának elmúlása. Hallottam egy érdekes gondolatmenetet is: az előadó vette a fáradtságot és végigsétált reggel a parkolóban. Furcsamód csupán egyetlen keleti autót talált.

Nos, vissza az előadásokhoz. Az első előadások az üzlettel foglalkoztak. Természetesen jelen voltak a Nagyok, és külön hangsúlyt kaptak a cégek közti B2B-alkalmazások. Igazából semmi izgalmasat vagy újat nem hallottam azon kívül, hogy az egyik előadáson *Veréb Elemér* szép lassan vadvízi evezéshez öltözött be, a végén még a sisakot is felvette.

Ezután két izgalmas előadás következett, mindkettő a politika és az Internet kapcsolatát firtatta. Érdekes kérdés: milyen szerepe lesz a Hálónak a választásokon? Eljutunk, eljuthatunk-e oda, hogy a gép mellett választjuk meg kedvenc politikusainkat? Melyik párt mire használja a Hálót és mire nem? Kell-e, hogy egy párt honlapja érdekes, szórakoztató legyen, vagy elsősorban az adatok és a hírek továbbítására célszerű szorítkoznia? A második vitán az MTE, az MSZP, az SZDSZ és a Fidesz képviseltette magát. Kiderült, hogy mindegyik párt nagyon komoly erőt érez az Internetben, de másként értékelik a hasznosságát, használhatóságát. Az MSZP elsősorban csalinak kívánja használni (a magyarázat kicsit hosszabb volt, de számomra ezt jelentette) játékokkal, propagandával stb.; az SZDSZ kiemelte az általa üzemeltetett NetPárt fontosságát (a honlap alapján mintegy kilencszáz tagot számlálnak); a Fidesz pedig elsősorban azt hangsúlyozta, hogy a Hálón lehet a leggyorsabban és legrészletesebben tájékoztatni a választókat. Ígértek egy úgy honlapot is bővített tartalommal, új kinézettel (érdemes megnézni!).

A következő (számomra) érdekes előadáson befektető cégeket szólaltattak meg. A fő kérdés természetesen így hangzott: mibe fektetnek és mennyit. A befektetők elmondták, hogy elsősorban a megtérülést kell tekintetbe venniük, és csak olyan cégekbe hajlandók pénzt ölni, amelyek biztos, hogy behozzák a várt nyereséget. Mivel ezt nagyon kevés cégnél látják biztosítva, csak olyan feltételek mellett adnak anyagi juttatásokat, amelyekről ők maguk is tudják, hogy a partnereknek nem felelhetnek meg.

Ezután *Gáspár Mátyás* tartott előadást a Teleházakról. Szerintem rendkívül jó kezdeményezésről van szó. Vállalkozó kedvű emberek a Szövetség támogatásával teleházat nyithatnak – internet-hozzáférést és „irodát" biztosítva a rászorulóknak. A Szövetség közel tíz éve indult, ma már több mint 250 házat számlál, és a társintézetek külföldön is rendkívül jó eredményeket érnek el (➲ http://www.telehaz.hu)

Ezután egy fontos bejelentés következett! A Széchenyi-terv két új pályázattal bővült, az első a „Magyar nyelvű internetes tartalmak fejlesztésének támogatása" (SZT-IS-7), a másik pedig „Az informatikai módszertanok honosításának, elterjesztésének támogatása az EU-harmonizáción belül" (SZT-IS-14). Mindkettő megtalálható a ➲ http://www.ikb.hu címen.

A második napon három érdekes témára bukkantam. Elsőként a jogi kérdések kerültek elő az MTE, a MAHASZ és az Artisjus képviselőinek részvételével. Érdekes kérdés volt számomra: vajon mit terveznek „odafönn" az olyan ügyekkel kapcsolatban, mint a GNUtella és testvérei? Kérdésemre kitérő választ

kaptam. Igazából nincs kiforrott terv, azt látják, hogy értelmetlen szélmalomharc volna küzdeni, de őszintén remélik, hogy ez a terület is letisztul.
Többször előkerült a szabályozás, az önszabályozás kérdése. Furcsálltam, hogy oly sokak számára bizonyult rendkívül fontos kérdésnek, hogy „miért nem tesznek valamit a pornóoldalak ellen?" Volt, aki felvetette, hogy állami szinten vagy a Sulinet rendszerén kellene felállítani ilyen jellegű szűrőket. Egyik kedvenc volt tanárom, *Győző Miklós* jutott az eszembe, aki egyszer egy érdekes történetet mesélt el. Történt, hogy egy évfolyamnál az egyik munkatársa nagyon komoly megszorításokat hozott, igyekezett ugyanis rákényszeríteni a diákokat a tanulásra. A félév végére több diák kiképezte magát a biztonsági kérdésekből és az egész gépparkban úgy jártak keresztül-kasul, mint az árnyak. A tanulmányi átlag viszont romlott egy kicsit…

Egy röpke órára előkerült a média és az Internet kapcsolata is. Érdekes bejelentés volt, hogy az új rádiótelefonok a tervek szerint akár két megabites átvitelre is képesek lesznek, és a sávszélesség folyamatosan nő.
A rendezvényt két vita zárta. Kár, hogy ezek az előadások a legvégére kerültek. Az első vita az internetelérések, a kapcsolódási és egyéb szolgáltatások piacát feszegette, a második pedig egy „beszélgetés" keretén belül hasonlította össze a Microsoft operációs rendszerét és a GNU/Linuxot.
A Szabad oldalt *Zelena Endre* és *Varga S. Csaba* képviselte az LME színeiben, a cégóriás pedig két rendszermérnököt és egy marketingvezetőt küldött.
Az óriási tömegek előtt zajló (mivel az utolsó előadás volt, a nagyteremben alig fért el az a maréknyi ember) beszélgetés szakmailag teljesen értékelhetetlenre sikeredett, már a felvezetőből egyértelmű volt, hogy sokan még mindig nincsenek tisztában a GNU, a Linux, a szabad program és egyéb fogalmakkal.
Az összehasonlítás összegzése pedig így szólt: mivel a CEU-ba különböző nemzetiségű, a számítástechnikához kevéssé értő diákok járnak, és az egyetemnek ilyen szinten nem kérdés a pénz, egyértelmű, hogy ügyfélgépként a Windowst választották. A beszélgetés ennek ellenére érdekes volt, remélem, jövőre komolyabb helyet kap ez a téma (ha még lesz miről vitatkozni), és a beszélgetés szereplői jobban felkészülnek.
Az előadásoknak sajnos csak töredékére tudtam ellátogatni, az érdeklődőknek mindenképpen javaslom a Média Hungária honlapjának meglátogatását (➲ http://www.mediahungaria.hu), ahol a szervezők sok anyagot közkinccsé tettek.

Szy György

Hárommilliárd forint nyomában

Előző számunkban már írtam a Microsoft és a Miniszterelnöki Hivatal között létrejött szerződésről. Ígértem, hogy utánajárunk, ténylegesen mire is fordítják az összeget, ki fizet és mennyit, valamint mi is a célja ennek a szerződésnek. Nos, egy-két dolog tisztázódott, sok minden azonban továbbra is kérdéses maradt.
Először is a szerződés értelmében a kedvezményes csomagokat meg kell igényelni egy nem túl egyszerű dokumentum kitöltésével és az igénylés beadásával. Ha jól értettem, egy Microsoft-partnernek kell intéznie ezt a kérelmet, mely után kiderül a fizetendő összeg. Ezt nem a vásárló (diák vagy oktató), hanem a MeH fizeti. A vásárló ezek után juthat hozzá (egyelőre még nem tisztázott, hogy milyen úton) a termékhez. Itt jön az első számú bökkenő. Ez a termék elméletileg frissítés, tehát törvényeen csak akkor használhatjuk, ha előtte már rendelkeztünk a termék egyik elődjével. Például a MeH vásárol nekünk egy Windows ME-t, de csak akkor, ha hivatalosan birtokolunk egy Windows 98-at. Ami szintén nem tízfillér. Ha ez tényleg így zajlik, kinek jó?
Ezzel a kérdéssel igyekeztem felkeresni a MeH illetékeseit. Gondoltam egyszerű halandóként, írok egy levelet a MeH sajtóosztályára, ahonnan rövid időn belül kaptam egy választ, az illetékes nevével, telefonszámával. Az illetékes titkárnője viszont átirányított egy negyedik emberhez és így tovább… Röviden összefoglalva eleddig nem találtam senkit, aki hitelt érdemlően el tudta volna magyarázni, ki milyen szerepet játszik e szerződés kapcsán. Továbbra is csak annyit tudok mondani: ígérem, a nyomozást folytatom, ha megtudok valamit, megírom.

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

Hátamon a házam?

Az igényes vásárló gondosan állítja össze a számítógépét: márkás alaplapot vásárol, körbekérdezi az ismerősöket, hogy éppen melyik merevlemez a legmegbízhatóbb és a legnagyobb teljesítményű, átböngészi az Interneten található teszteket, hogy vajon melyik VGA-kártya a leggyorsabb, melyiknek a legszebb a képe és így tovább. Amikor a megfelelő gépet összeállította, az egészhez „valamilyen" házat is vásárol. A választás vagy arra esik, amelyiknek a doboza a polcon legközelebb volt, vagy arra, amelyiknek az előlapja a leginkább tetszett.

Codegen-nagytorony

Az igényes vásárló két hét elteltével nem érti, miért fagy állandóan a gépe. A fagyások folytatódnak, de nem sokáig, mert a gép tápjában hamarosan felrobban az egyik kondenzátor. Ekkor két esélyünk marad: vagy tesznek bele rendes tápot, vagy nem.

Az olcsóbb számítógépházakat nagyjából három tulajdonsággal lehet jellemezni: először is a gyártók a beléjük szerelt tápegységekből néhány filléres alkatrész kispórolásával azt érik el, hogy az egység nem tud állandó és kellő teljesítményt adni, ami a számítógép szeszélyes viselkedését eredményezheti. Természetesen nem viseli el kellőképpen a hálózati feszültség ingadozásait sem, hamarabb elöregszik, és gyakorta egyszerűen felrobban benne valamelyik kondenzátor. Másodsorban a belül található lemezek élét sokszor nem sorjázzák, ezért a házba nyúló személy bizton számíthat rá, hogy az éles szélek megsebzik a kezét. A takarékos belső kialakítás miatt az olcsó és kisméretű házakat nehéz szerelni: egy-egy merevlemez eltávolítása vagy beszerelése sokszor más alkatrészek ideiglenes kiszedését is igényli. Harmadrészt a gép belsejének hűtése sem lesz mindig kielégítő, pedig az újabb merevlemezek és processzorok sokszor jelentős hőt adnak le.

Papolni könnyű, de mit tehet a vásárló, ha a legtöbb üzlet kínálatában csak az olcsóbb házak szerepelnek? Nem marad más, mint nyakunkba venni a várost vagy az Internetet, és körülnézni.

Jómagam hét házat gyűjtöttem be Budapest különböző üzleteiből. A zsákmány sokszínűre sikeredett, a választék mind árban, mind célterület szerint bármely igényt kielégít. Közös vonásuk, hogy valamilyen szempontból mindegyik valami pluszt tudott nyújtani a másikhoz és az olcsóbb házakhoz képest. Akár asztali munkaállomáshoz, akár komolyabb kiszolgálóhoz is találni megfelelő házat; ilyenkor fizetéskor egy kicsit mélyebben kell a zsebünkbe nyúlnunk. Az itt szereplő házak mindegyike megfelel az ATX-szabvány előírásainak. Tápjuk teljesítménye változó, és nem mindegyik használható Pentium IV-es processzorral – ezek tápellátásához ugyanis a megszokott ATX-es tápcsatlakozón kívül további vezetékek is szükségesek.

Codegen-nagytorony

Szerzeményeim közül a Codegen-nagytorony volt a legolcsóbb és legegyszerűbb ház. Nem luxusdarab, de a típus kedvező ára és megbízható tápja következtében jelentős népszerűségre tett szert az utóbbi időben. Mint a fényképen is látható, helyet bőven találunk benne, ha a nagy meghajtókat szellősen szeretnénk elhelyezni, ám a merevlemezekre ez nem áll. A hajlékonylemezes meghajtót a felső takarólemez eltávolítása után felülre szerelhetjük be, az alsó 3,5"-os helyekre három merevlemez szorítható be, valamint elhelyezhető egy a táp fölé is. Ha megfelelő hűtésről szeretnénk gondoskodni, egy 80 mm-es ventilátort helyezhetünk el a bővítőkártyák végénél, továbbá a táp fölött is akad neki hely. Az alsó hely hatékonysága erősen kétséges, hiszen a ventilátor mögött nincs megfelelő légtér a levegő mozgatásához; a fenti hely pedig egymagában árválkodik, legalább még egyet kialakíthattak volna mellette, illetve fölötte. A processzor környékére sem tudunk további hűtést szerelni, kárpótlásul viszont a hátoldalra, a bővítőkártyák fölé különféle méretű kapukat és kivezetéseket rögzíthetünk. Érdemes még megemlíteni, hogy a bővítőkártyák elhelyezésekor a ház hátsó takarólemezeit ki kell törni, és később nem is tudjuk visszaszerelni őket. Kivétel ez alól a legalsó lemez, és enyhít a gondon, hogy a házhoz mellékelt csomagban további két csavarozható lemez található. A házhoz tartozik két nagyméretű műanyagtalp is, ízlés dolga, hogy felszereljük-e őket. Maga a ház szegecseléssel, néhol csavarozással készült, szétszedésére nem sok esélyünk van – igaz, erre általában nincs is szükség.

AOpen közepes torony

Az AOpen közepes méretű tornya meglepően egyszerű, ám módfelett rokonszenves. Valójában semmi különlegeset nem nyújt, viszont munkaállomáshoz kiváló választás. Három-három kis- és nagy méretű meghajtónak jut benne hely, az előbbiek közül egy hajlékonylemezes meghajtó lehet. A ház masszívnak tűnk, a biztos tartást azzal is elősegíthetjük, hogy a meghajtók beszerelésekor a ház fémlapjait úgy távolítjuk el, hogy egyik felüket kitörjük, a másikat pedig egyfajta támasztékként hátrahajtjuk. A ház elülső lapja formatervezett, kicsit íves. Hűtésre itt sincs sok lehetőségünk: a bővítőkártyák végéhez rakható be egy bőséges légtérrel rendelkező ventilátor. Kárpótlásul két kicsi és egy közepes méretű kivezetést helyezhetünk el a processzor mellett, ráadásul a hátulsó takaró fémlapokat sem kell kitörnünk, ezek egyszerűen megszorulnak a helyükön, és könnyedén eltávolíthatók vagy visszahelyezhetők. A házba gyárilag hét alaplaptámasztó csavart is beszereltek.

Az AOpen tornya volt az egyetlen a nálam járt példányok közül, amelyiken egy „AMD & ATX 2.03 Certified" matrica díszelgett, büszkén hirdetve, hogy a termék a legújabb elvárásoknak is megfelel.

AOpen-nagytorony

Az AOpen nagytornya lényegesen izmosabb darab. Alsó felének kialakításáról nem sokat lehet mondani, nagyban

hasonlít kistestvéréhez. A hajlékonylemez viszont felülre került, és a táptól felfelé eső rész is jókorát nőtt. A tápegység felett nem kevesebb, mint négy merevlemeznek jut hely, tehát akár hét merevlemezt is bezsúfolhatunk a házba. Nagy meghajtóból öt fér el, a tápellátásról pedig a tápegységből bőséggel kibuggyanó kábelek sokasága gondoskodik.

Ekkora erőműhöz hűtés is kell, a fenti rész hátoldalán két 80 mm-es ventilátornak alakítottak ki egymás mellett helyet. Az AOpen nagytornyának borítása is meglehetősen erőteljes, az alul és felül elhelyezett karmoknak köszönhetően oldalait csak úgy tudjuk eltávolítani, ha hátrafelé teljesen lehúzzuk őket. A felső részt külön kell leszerelni, ha a hajlékonylemezes meghajtó helyéhez is hozzá akarunk férni.

Az alaplapot tartó csavarokat ezúttal magunknak kell a helyükre csavaroznunk, a ház méretére való tekintettel viszont talpakat is kaptunk hozzá. A ház jól szerelhető, ugyanakkor nem mehetek el egy apróság mellett: a felül elhelyezett tápkapcsoló vezetékét képtelenek voltak az elülső fedőlap mögé bevezetni, így a ház közepén lógva vezet az alaplapig – szerelés közben lehetőleg ne akadjunk bele.

DTK közepes torony

A DTK közepes tornyának zöld műanyag sínjei első ránézésre kicsit meglepőek, viszont ügyes segítséget nyújtanak a gyors szereléshez. A meghajtókat csak be kell illesztenünk a helyükre, majd a sínt előrehúzva az egyik oldalon a csavarok helyére apró fémpöckök ugranak be. Egy reteszt lenyomva rögzíthetjük a meghajtót, és már végeztünk is a szereléssel. A sínes megoldás az összes nagy- és a hajlékonylemezes meghajtó esetében is használható, a merevlemezeket viszont csavarozni kell.

A gyors szerelés lehetőségéért természetesen fizetni is kell: mivel a sín csak az egyik oldalon tartja a meghajtókat, a másik oldalon mindössze egy erős, hajlított fémlap gondoskodik beszorításukról, a meghajtók kismértékben mozoghatnak, ami például szállításkor nem biztos, hogy jót tesz a gépnek.

Ha kereszthuzatot szeretnénk kelteni a gépben, nincs más dolgunk, mint a bővítőkártyák végéhez és a processzor mellé egy-egy 80 mm-es ventilátort elhelyezni. Érdemes megfigyelni, hogy a merevlemezeknek helyt adó fémlemez oldalain két-két apró fül található. Ezekre egy

	Codegen-nagytorony	AOpen közepes torony	DTK közepes torony	Chieftec közepes torony	Orca mini fájlkiszolgáló	AOpen-nagytorony	Chieftec-nagytorony
Típus	ATX-7002-8	HQ45	WT-PT074W	TX-10W	D112	HX-08	TA-10W
Táp teljesítménye (W)	300	250	300	300	250	300	340
Kikapcsolható táp	nem	igen	nem	igen	nem	igen	igen
Fordulatszámmérős táp	nem	nem	nem	igen	nem	nem	igen
Pentium IV processzorhoz megfelelő-e a táp	nem	nem	igen	nem	nem	igen	igen
5,25"-os bővítőhelyek	5	3	4	3	6	5	6
3,5"-os bővítőhelyek (hajlékonylemezes meghajtóval együtt)	4	3	3	7	2	8	7
Nagy tápcsatlakozók száma (merevlemezhez, CD-ROM/DVD-meghajtóhoz)	4	5	6	6	4	6	6
Kis tápcsatlakozók száma (hajlékonylemezes meghajtóhoz)	1	2	2	2	1	2	2
Méret (mélység×(magasság×szélesség)	435×622×190	420×414×198	–	470×522×205	440×449×233	420×590×198	470×670×205
Nettó kiskereskedelmi ár	kb. 11 000 Ft	20 600 Ft	16 240 Ft	kb. 31 000 Ft	23 500 Ft	29 750 Ft	kb. 38 000 Ft
Üzlet	CAM-EL-COM	Macroda	Mikland	Kelly-Tech	Herta	Macroda	Kelly-Tech
Üzlethonlap	*1a	*2a	*3a	*4a	*5a	*6a	*7a
Termékhonlap	*1b	*2b	*3b	*4b	*5b	*6b	*7b

*1a http://www.camelcom.hu *1b http://www.codegengroup.com/product-7010-7001.html; *5a http://www.herta.hu/ *5b http://www.denco.com.hk/
*2a http://www.macroda.hu *2b http://www.aopen.com/products/housing/hq45.htm *6a http://www.macroda.hu *6b http://www.aopen.com/products/housing/hx08.htm
*3a http://www.mikland.hu *3b – *7a http://www.kellytech.hu *7b http://www.chieftec.com/products/scorpio/ta10w.htm
*4a http://www.kellytech.hu *4b http://www.chieftec.com/products/scorpio/tx10w.htm

további merevlemezt csavarozhatunk fel, vagy ha megfelelő fémkeretet tudunk szerezni, illetve készíteni, azt is felszerelhetjük rá.
Az alaplapot tartó csavarokkal nekünk kell megküzdenünk, plusz a mellékelt zacskóban nagyon igényes fémkiegészítőket találunk. A kártyahelyek takarólemezeit is külön kapjuk, a vékony fémlapokat szükség szerint illeszthetjük be a helyükre. Ugyancsak a mellékelt zacskóban bukkantam két állítható műanyag távtartóra is, amelyek rendeltetése homályban maradt előttem, valamint egy apró figyelmességre, egy kábelkötegelőre is – filléres apróság, mégis hasznos.


Chieftec-nagytorony

Chieftec közepes torony

A Chieftec-házak teljesen új jelentéssel ruházzák fel a számítógépház szót. Velük már nemcsak egy dobozt vásárlunk, hanem valódi munkaeszközt. Lelkesedésem oka már ránézésre nyilvánvaló: a Chieftec-házat a legkönnyebb szerelni, ez nyújtja a legtöbb extrát. A merevlemezek két „háromszemélyes" keretben kapnak helyet. A keretek külön-külön eltávolíthatók, ehhez csak a rögzítő kart kell elfordítani, majd a keretet hátrafelé kihúzni. A nagy meghajtóknak először mindkét oldalára egy-egy műanyag sínt kell szerelni, majd becsúsztatni őket a helyükre. A sínek kifelé eső végén található fémlemezek beakadnak a ház elülső részén kialakított résbe, és biztosan rögzítik az egységet. Természetesen a sínek elvesztésével sem lehet gondunk, a ház alján elhelyezhető öntapadó műanyag tartólemezekbe csúsztathatjuk őket. A nagy meghajtók szerelésekor egyébként a ház elülső műanyag fedőlapjának felső a részét két oldalsó gombot megnyomva el kell távolítani.
A hűtéssel nem lehet gond, a keretekbe egy-egy 80 mm-es ventilátor helyezhető el.
Ügyelni kell arra is, hogy ne zsúfoljunk túl sok hosszú bővítőkártyát a gépbe, hiszen az alsó merevlemezkeret benyúlik a felső bővítőhelyekre. Más rosszat viszont nem lehet mondani a házról, a gyárilag felszerelt és kihajtható tappancsoknak köszönhetően a nehéz, erőteljes fémdobozt még felrúgni is csak nehezen lehet, záras-kulcsos oldallemeze pedig biztonságosan védi a gép belsejét.

Orca mini fájlkiszolgáló

A Herta árlistájában az „Orca mini fájlszerver" néven szereplő ház jócskán kirí a sorból. Magasságát tekintve egy közepes toronyra emlékeztet, ám jóval szélesebb társainál, és a belső kialakítása is rendhagyó. Mindössze egyetlen hajlékonylemezt és merevlemezt tudunk beszerelni, a többi bővítőhely nagyméretű, tehát akár hat 5,25"-ös meghajtót is használhatunk a gépben. Valójában sokkal rugalmasabb kialakítást tesz lehetővé, hiszen kiegészítő sínekkel a nagy helyekre is bármit beszerelhetünk, ezzel a kialakítással viszont egyszerre akár hat CD-meghajtó is rendelkezésünkre állhat.
Ennek ellenére a ház hűtési lehetőségeit szegényesnek is nevezhetjük: a gyárilag beszerelt ventilátor hasznos ugyan, de továbbiak elhelyezésére nincs mód.
Mivel a tömzsi kialakítás miatt az alaplap a meghajtókeret mögé kerül, ügyelni kell a bővítőkártyák behelyezésekor.
A kábelek elvezetése is gondot jelenthet, hiszen az alaplapi IDE és SCSI vezérlők csatlakozói általában az alaplap jobb szélére kerülnek, ezért először a keret mögül kell kibújtatni.
Érdemes megemlíteni, hogy a ház elejére egy átlátszatlan műanyag ajtó került, amely kulccsal is zárható. Sajnálatos módon csak a hat nagymeghajtót rejti el, a hajlékonylemezest nem – már ha kerül ilyen a gépbe. Ha nem, a gép szinte bombabiztosnak nevezhető, hiszen a lezárt ajtó mögött található meghajtókhoz nem lehet hozzáférni, a hajlékonylemezes meghajtó keretének elülső részét meglepő módon gyári állapotban egy csavarozott fémlemez torlaszolja el.
A ház hátsó részével nem lesz gondunk, a takarólemezek csavarozhatók, és be is szorulnak a helyükre, a pótlólagos, különféle hosszúságú kivezetéseknek, kapuknak pedig bőven jut hely a processzor mellett.
Az Orca-ház bizonyos helyzetekben helytakarékos, célszerű megoldásnak tűnik. Az előlapi IDE LED-ről viszont le kell mondanunk, ezt ugyanis nem sikerült elhelyezni a házon.

Chieftec-nagytorony

A Chieftec nagytornya tűnik a legsokoldalúbb megoldásnak, ez egyben a legdrágább is. Sok mindenben hasonlít a közepes toronyhoz, ugyanazok a pillanatok alatt kiemelhető merevlemez- és ventilátorkeretek köszönnek vissza, a nagyméretű meghajtókat ugyanolyan módon helyezhetjük be. Fontos különbség azonban, hogy a merevlemezkeretekbe egy-egy ventilátort tudunk rögzíteni, és a közepes toronnyal ellentétben merevlemezeket is szerelni tudunk eléjük. Ily módon a lemezek között légmozgást lehet kelteni, hűtésük tehát hatékonyan oldható meg. A gép egyéb részeinek hűtéséről további három ventilátorral gondoskodhatunk, továbbá a táp fölé is csavarozható egy nagyobb méretű ventilátor.
A Chieftec nagytornyába került a legnagyobb teljesítményű, 340 wattos tápegység. A tápkábeleket teljesen kinyújtva akár a ház aljától 20 cm-re található készülékeket is üzemeltethetnénk, így a kábelek elvezetésével sem lehet gond. Mint a fényképen is látható, a Chieftec-nagytorony masszív, merevítései erősek – ez súlyában is érezhető, az üres ház egymagában nehezebb az egyszerűbb számítógépeknél.

Medgyesi Zoltán
(mzx@axelero.hu)
A BMGE 23 éves informatika szakos hallgatója. Szabadidejét legszívesebben barátnőjével tölti. Szeret autózni, és bográcsban főzni. A Linuxot öt éve ismeri, de még nem volt lelkiereje, hogy áttérjen rá.

Palm- és Handspring-gépek összekapcsolása Linuxszal

Természetesnek mondható, hogy az Amerika-szerte méltán népszerű kézi eszköz Linux operációs rendszer alatt is támogatást élvez, az azonban, hogy Linux alól az adatainkat sokkal kényelmesebben és gyorsabban tudjuk átvinni a PDA és a számítógép között, sokak számára újdonságként hathat. Nézzük, hogyan és milyen programokkal teszik lehetővé, hogy adatokat küldjünk vagy fogadjunk.
Ha GNU/Debian Woodyt veszünk alapul, a következő alapcsomagokra lesz szükségünk: *jpilot*, *pilot-link*, *pyrite-publisher*.
Miután letöltöttük, vagy a GNU/Debian-rendszerből a csomagkezelő segítségével telepítettük, még be is kell állítanunk őket.

JPilot

Grafikus kezelőfelület, amely nagyon hasonlít arra a programra, amit a gyári windowsos CD-ROM-on kapunk. Ezzel a programmal lehetőségünk nyílik a jegyzetfüzet, a naptár, a tennivalólista és a címjegyzék szerkesztésére. A segítségével adatainkat össze tudjuk hangolni a gépünk és a Palm között, valamint biztonsági mentést (Backup) tudunk készíteni. E program használatát akkor javasolhatjuk, ha a grafikus felületen történő munkához szoktunk hozzá, ám amennyiben a parancssori vezérlést kedveljük jobban, arra is lehetőségünk nyílik. A program beállítási lehetőségeit mindenképpen nézzük végig, mert könnyen előfordulhat, hogy bár a kapcsolat működik, az alapbeállítás 9600-as kapusebességet feltételez, és emiatt nagyon lassú lesz az átvitel. Mielőtt a grafikus felületet elkezdhetnénk használni, kénytelenek vagyunk parancssorból felhasználó azonosítót „gyártani", amelyet Palmunk az összehangolás kapcsán használ, tehát az `install-user` programot kell futtatnunk a megfelelő értékekkel.

```
install-user soros_kapu felhasznÆl
             ➥felhasznÆl _ID
```

például:

```
install_user /dev/ttys0 Guska 1000
```

Ezután már zökkenőmentesen tudjuk használni a *jpilot* nyújtotta lehetőségeket.
Manapság a legtöbb gép USB-bölcsővel kerül forgalomba, és egy-egy soros bölcsős változat szinte különlegességnek számít. Az USB-szabvány bizonyos szempontból jobb a sorosnál, hiszen gyorsabb adatkapcsolatot tesz lehetővé, akad azonban egy nagy hátránya: a soros eszköznél – például éjszakára –a mentést önműködővé tehetjük (ekkor a gépünk úgyis „ráér"), hiszen a soros kapun „WAKE UP" jelet tudunk küldeni a PDA-ra, tehát a kapcsolat emberi beavatkozás nélkül, a *Sync* gomb megnyomása nélkül is létrejöhet. Az USB-szabvány esetében viszont mind a két félnek kezdeményeznie kell a kapcsolatot, és csak akkor jöhet létre az eszközfájl is (*/dev/pilot* vagy a */dev/ttyUSB1*), amikor az első „kézfogás" (handshaking) sikeres mivoltáról tudomást szereznek. Ebben az esetben mindenképpen szükség van egy fizikai eszközre (például emberi kézre), amely az adott pillanatban megnyomja a Sync gombot.

Pilot-Link

Ez a program ugyanazokra a feladatokra is használható, mint a *jpilot*, csak mindent parancssorból kell megadnunk neki. Ez előny és hátrány is lehet. Hátrány, mivel mindent be kell gépelnünk; nagy előny azonban, ha önműködő mentési rendszert szeretnénk felállítani, amely például hajnali három órakor a gép teljes tartalmát menti, hiszen remekül jön egy héjprogramgyűjtemény.
Például ha egy 500–800 rekordot számláló telefonkönyvvel rendelkezünk, nehéz lenne az adatokat egyenként felvinnünk, azonban CSV formátumban (a StarOffice is fogadni tudja), a `pilot-address` program segítségével egyszerűen fel tudjuk tölteni. Viszonyításképpen: egy Psion *Contacts*-ból CSV-be mentett 800 bejegyzést tartalmazó állományt a `pilot-adress` program segítségével megközelítőleg három perc alatt fel lehet dolgozni.

Palm-DocToollit

Miként a neve is mutatja, arra szolgál, hogy dokumentumokat alakítsunk át *Palm Reader* által olvasható formátumba. A program a `pyrpub` névre hallgat. Az átalakítandó szövegfájlt értékként kell megadni, ekkor egy hasonló nevű .PDB-állományt készít belőle. Ennek olvasásához azonban valamilyen olvasóprogram szükséges, én a CSpotRunt javaslom, mert nagyon jól méretezhető és rengeteg beállítási lehetőséggel rendelkezik.
A CSpotRun program a ➲ http://pda.tucows.com címről tölthető le. Mivel a Palm kijelzője igen kicsi és a különböző szövegek általában 76–80 karakter szélesek, mielőtt átalakítjuk, érdemes a `fmt` (simple optimal text formatter) segédalkalmazás segítségével betördelni.

Töltögetnivalók

Elektronikus dokumentumok olvasgatására a ➲ http://mekiif.hu-t javaslom.

Kapcsolódó címek

jpilot ➲ http://www.jpilot.org
pilot-link ➲ http://www.pilot-link.org
palm-doctoolkit ➲ http://www.pyrite.org

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.

Jönnek a robotok, jönnek a robotok!

A Linux lépésről lépésre hódítja meg a világot – apró lábnyomokat hagy, de egyre biztosabb lábakon áll. *Isamu* 135 cm magas, 55 kg súlyú, és több mint egy mérföldet tud megtenni óránként. Isamu tud még lépcsőkön fel- és lelépdelni, kézszerű fogóival közel kétkilós súlyok cipelésére képes, és kétkamerás sztereo képrendszerével az emberi arcokat is felismeri. Isamunak agya is van, a két Pentium processzorral működő beágyazott számítógép RTLinux rendszert futtat.
Isamu a Tokiói Egyetem Jouhou System Kougaku Laboratóriuma (JSK Lab) és a Kawada Industries, Inc. (Tokió, Japán) Légiközlekedési és Mechanikus Rendszerek részlege összefogásának eredményeképpen

született. A tervezet célja olyan berendezések készítése, amelyek az interaktív emberi mozgásvezérlési módszerek fejlesztése során tehetnek jó szolgálatot.
A Kawada Industries például számos kereskedelmi alkalmazást tervez, többek között az építési rendszerek, a kármentesítések, a mozgássérültek segítése, a rehabilitációs és oktatási eszközök, illetve a szórakoztatás területén.
Ahhoz, hogy emberi mozgásra legyen képes, Isamu harmincöt „szabadsági fokkal" rendelkezik: minden lábára hat, lábfejeire egy-egy (a lábujjait együtt mozgatja), karjaira hét, fogóira egy-egy, nyakára kettő, szemeire pedig három jut. A beépített számítógép két 750 MHz órajelű Pentium III processzort tartalmaz, és RTLinux operációs rendszert futtat, amely a valós idejű szervo- és egyensúlyvezérlésért felelős, irányítja a robot háromdimenziós látását, valamint a mozgástervező programrészeket.
Az erőteljes akkumulátoroknak, a vezeték nélküli ethernetcsatolónak és a nagyteljesítményű beépített számítógépnek köszönhetően Isamu kábelek és folyamatos emberi beavatkozás nélkül is működhet. Ha emberi irányításra van szüksége, mozgása botkormánnyal vezérelhető. Isamu két lábon való járását és rendszervezérlő programját a JSK Lab, a robottestet magát, valamint a szervoalapú szintvezérlő rendszert pedig a Kawada Industries fejlesztette. A Kawada a repülőgépiparból átvett elemek segítségével készítette el a testet, így erős, mégis könnyű szerkezetet sikerült alkotniuk. Látogass el a JSK Lab honlapjára (➲ http://www.jsk.t.u-tokyo.ac.jp), ahol lélegzetelállító mozgóképeken nézheted meg, Isamu mi mindenre képes.

Linux-frissítés Palm III gépekhez

Az Empower Technologies bejelentette „a világ első jelentősebb operációsrendszer-frissítését a Palm IIIx és IIIxe kézi számítógépekhez". A Linux DA előzetes bemutatópéldánya a cég honlapjáról tölthető le.
A bemutatóváltozat ugyan még csak a kézi számítógépek legalapvetőbb lehetőségeit támogatja (címjegyzék, határidőnapló, számológép, jegyzettömb és néhány játék), a kereskedelmi változat több lehetőséget tartalmaz majd, például böngésző, levélküldési lehetőség és sok egyéb is megtalálható lesz benne. A cég állítása szerint a Linux DA alapját általános Linux-rendszermag képezi, a grafikus megjelenítést, az adatbázis-kezelést, az energiakezelést, a rendszertöltést, a flashmemóriák kezelését, az adatösszehangolást, az infravörös kapu vezérlését és a személyes adatkezelő alkalmazások futtatását viszont saját fejlesztésű programjuk végzi. Részletek a ➲ http://www.empower-technologies.com címen olvashatók.

Linux 27×27 milliméteren

Az Axis Communications, az ETRAX nevű lapkagép (system-on-chip – rendszer egyetlen lapkán) készítője több mint ötven összetevőt épített egyetlen többlapkamodulra (multichip modul – MCM), szabványos 27×27 mm-es PBGA IC-tokozással. Az Axis szerint mindössze két külső kiegészítő szükséges, hogy működőképes Linux-rendszer álljon össze: egy 20 MHz-es órajel-létrehozóra és egy 3,3 voltos áramforrásra.
Az új MCM tartalmazza az Axis ETRAX 100LX nevű, RISC-alapú lapkagép processzorát, valamint az összes olyan rendszerösszetevőt, ami az erre a lapkára épülő eszközökben megtalálható, például DRAM-ot, flashmemóriát, egy ethernetcsatolót és az összetevők közötti összeköttetést biztosító alkatrészeket. Az Axis állítása szerint a Linux, valamint egy kisebb alkalmazás sikeresen futott az eszközön, teljes egészében az MCM beépített 2 MB flash- és 8 MB SDRAM memóriáját használva. Az eszköz fejlesztői környezete és néhány jellemző példaalkalmazás már elérhető. Az első termékminták hozzáférhetősége a közeljövőben várható. A végleges árak még nem ismeretesek, de valószínűleg 50 (10 000 darabos vásárlásnál) és 75 dollár (egyetlen darab) között fognak alakulni. Bővebb tájékoztatás a ➲ http://www.developer.axis.com honlapon található.

Rick Lehrbaum (rick@linuxdevices.com) hozta létre a LinuxDevices.com „beágyazott Linuxok portálját", amely nemrég tagja lett a ZDNet Linux Resource Centernek. Rick 1979 óta foglakozik beágyazott rendszerek fejlesztésével. Társalapítója az Ampro Computersnek, alapító tagja a PC/104 Consortiumnak, és fontos szerepet játszott abban, hogy elindulhatott az Embedded Linux Consortium.

Nevet a nyílt forráskódú programoknak!

A védjegyek értéke és fontossága

Tegyük fel, hogy kedvenc ebédlőhelyünkön édesített, szénsavas, karamellszínű, természetes és mesterséges aromákkal ízesített vizet kínálnak. Vajon meginnánk? Érdekelne-e, hogy ki gyártotta? Szeretnénk-e előre tudni, hogy italunk ugyanolyan ízű lesz-e, mint az az ital, amely korábban többször is olyan nagyon ízlett?

Most tegyük fel, hogy kedvenc ebédlőhelyünkön Pepsit kínálnak. Nem foszlanának-e szét hirtelen a termék (ilyen vagy amolyan) minőségével kapcsolatos és a felismerhetőségére vonatkozó kérdéseink? Melyiket választanánk a Coca-Cola és a Melvyn's Cola közül? Nagyobb valószínűséggel döntünk-e egy termék mellett pusztán azért, mert ismerjük hírét-nevét?

Mindannyiunkat behálóznak a reklámok: a tévé és más hirdetési csatornák folyamatosan terméknevekkel árasztanak el minket, amelyeket majdnem olyan természetesen építünk be a szókincsünkbe, mintha anyanyelvünk szavai volnának. Ezek a terméknevek azonban nem a hagyományos értelemben vett szavak, és nem is ugyanolyan használatra szánták őket. Az olyan kifejezések, mint Windows, Apple, Java, Apache, Linux, Jabber és Jboss marketingnevek – védjegyek. Ha a védjegyeket gondosan választják ki, a terméket megóvják és hatékonyan népszerűsítik, a minőség, a hozzáértés és a hírnév jelképeivé válhatnak.

A védjegyek a szabad és nyílt forrású programokat terjesztő cégek legértékesebb vagyontárgyai közé tartozhatnak. Bizonyos fejlesztések esetében a védjegy sokkal fontosabb lehet, mint a program terjesztési szerződése.

A védjegyek szellemi tulajdont képeznek, de nagymértékben különböznek a szabadalmaktól, amelyek találmányokat védenek; a szerzői jogoktól, amelyek megnyilatkozásokat védenek, és az üzleti titkoktól, amelyek cégek bizalmas adatait védik. A védjegyek termékek vagy szolgáltatások márkanevei, egy termék hírnevét kell védeniük. Közelebbről meghatározva az áruvédjegy olyan szó, név, jelkép vagy eszköz, amely egy cég termékeit azonosítja és megkülönbözteti azokat a versenytársak termékeitől. A szolgáltatás védjegye ehhez hasonló, de szolgáltatásokat különböztet meg. A bejegyzett cégnév egy céget azonosít, nem pedig egy cég egyedi árucikkeit vagy szolgáltatásait. A következőkben a védjegy kifejezést általánosan fogom használni, és figyelmen kívül hagyom a szolgáltatás- és a cégnevek aprólékos megkülönböztetését.

Egy cég pusztán egy védjegy választásával nem szerez védjegyhez kapcsolódó jogokat, sőt azzal sem, hogy bejelenti igényét a használatára. Az Egyesült Államokban a védjegyeket meghatározott árucikkekkel (vagy szolgáltatásokkal) kapcsolatos használatuk alapozza meg. Vannak előnyei annak, ha a védjegyet bejegyzik az illetékes hivatalnál (US Patent and Trademark Office), de a bejegyzés önmagában nem tesz egy védjegyet érvényessé vagy értékessé. Érvényes és értékkel bíró védjegy csak úgy jöhet létre, ha a védjegyet az árucikkeken ténylegesen használják, illetve marketingtevékenységet végeznek annak érdekében, hogy ismertségét az ügyfelek körében növeljék.

A szabad és nyílt forráskódú termékek bonyolult marketingfeladatot jelentenek. Azok a szerződések, amelyek az ilyen termékekre érvényesek, megkövetelik a forráskód terjesztését, és lehetővé teszik az ebből származtatott termékek létrehozását és terjesztését. Az ilyen termékek terjesztői számára nehéz egyedül az ár terén versenyezni, mert bármely cég, amely rendelkezik a szükséges tudással, letörheti az árakat pusztán az eredeti program lemásolásával.

Az ilyenfajta környezetben különösen hasznosak lehetnek a védjegyek. A cégnek először meg kell mutatnia, hogy a terméke magas színvonalú, megbízható, hatékony, sok szolgáltatást nyújt és felhasználóbarát. Ígérhet folyamatos javításokat, terméktámogatást, felhasználói csoportokat, és más tevékenységeket is végezhet az ügyfélkör kiépítése érdekében. Később az erőfeszítések eredményeképpen a cég ügyfelei a termékek vagy szolgáltatások védjegyeit társítani kezdik ezekkel a termékekkel vagy szolgáltatásokkal. Az új vagy visszatérő üzletfelek olyan árukért és szolgáltatásokért fognak fizetni, amelyek vélhetően megérik a rájuk fordított pénzt, és olyan termékeket fognak előnyben részesíteni, amelyek márkajelzését fölismerik.

A védjegyek korlátozzák a szabad és nyílt forráskód bizonyosfajta leágazásait is. A felhasználói szerződés ugyan lehetővé teszi, hogy mások a mi kódunkból származó munkákat hozzanak létre, de nem engedi meg (és valójában megvan a jogi ok, amiért nem is engedheti meg), hogy a leszármazott termékek a mi védjegyünket viseljék. Minden olyan iparágban, ahol technikai értelemben nem nehéz új terméket létrehozni, legyen az a kólaipar vagy a szabad és nyílt forráskódú programok ipara, a termékek és szolgáltatások védjegyei nyújthatják a legjobb védelmet egy cég termékeinek és szolgáltatásainak. Mindennap tapasztaljuk ezt, amikor a vásárlók Coca-Colát vagy Pepsit rendelnek Melvyn's Cola helyett. Ugyanezen okból vásárolnak RedHat Linuxot és nem „Jó Ö'csó" Linuxot. Ezért uralják a Hálót az Apache-kiszolgálók, nem pedig az Apache eredeti kódjára épülő kiszolgálók. És ezért lehetséges okosan megválasztott, tökéletesített és használt védjegyekkel megvédeni a szabad és nyílt forráskódú programjaink piacát azoktól a versenytársaktól, akik csupán le akarják másolni a munkánkat, és hasznot húzni a jól megérdemelt hírnevünkből.

A jogi tanácsadás megfelelő kerete egy jogász-ügyfél kapcsolat, amely egy adott helyzet minden tényállását figyelembe veszi, és a helyileg érvényes jognak felel meg. Bár ezt a cikket egy jogász írta, a benne foglalt adatok nem helyettesíthetik az esetre szabott, bejegyzett jogásztól származó tanácsadást.

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).

Cégcsokor (3. rész)

Sorozatunkban olyan cégeket gyűjtünk csokorba, amelyek huzamosabb ideje számos területen Linuxot alkalmaznak. Lássuk, mi a helyzet az oktatási és az ahhoz kapcsolódó intézményekben.

A miskolci Andrássy Gyula Műszaki Középiskolában három Linux-kiszolgálót is működtetnek. Az első lassan az iskola leggyengébb gépévé válik (166 MHz-es Pentium MMX-es processzor, 128 MB RAM és három merevlemez), most ezt nézzük meg közelebbről. Maga a rendszer egy 1,6 GB-os merevlemezen helyezkedik el, egy tízgigásat használnak a tanulók saját könyvtárainak tárolásához, valamint egy 1 GB-ost a Squid gyorstáraként (cache). A gépet Mandrake 7.2 „hajtja”, és a következő feladatokkal bír: webkiszolgáló – Apache; FTP-kiszolgáló – proftpd; levelezőkiszolgáló – Postfix; proxykiszolgáló – Squid; IP-címek kiosztása – bootp; SSH-kiszolgáló – OpenSSH; fájlkiszolgáló és tartományvezérlő – Samba; csomagszűrés – IP Chains; kapufigyelés – portsentry; fájlváltozások ellenőrzése – Tripwire; naplók feldolgozása – logcheck, webalizer, sqmgrlog, http-analyze, valamint saját fejlesztésű parancsfájlok.

Az iskola összes tanulója rendelkezik levélcímmel és csak ezt a kiszolgálót érik el, de a gépen egyes tanulók héjeléréssel is bírnak. Olyan diákok is akadnak, akik a Pine segítségével leveleznek. A gép hat éve üzemel folyamatosan, és az „Operációs rendszerek” elnevezésű tantárgy óráin is kitűnő segédeszközként működik.

A második kiszolgáló az iskola ügyvitelének zavartalan kezeléséért felelős. 400 MHz-es Celeron processzorral és 128 MB RAM-mal van felszerelve, és egy 5,1 GB-os, egy 6,3 GB-os, valamint két 10 GB-os merevlemez gondoskodik az operációs rendszer és az adatok tárolásáról. RedHat 6.1 felel (Samba segítségével) a 14 windowsos munkaállomás kiszolgálásáért és a felhasználók beléptetéséért (tartományvezérlőként működik). Az iskola dolgozóit érintő szolgáltatások ezen a kiszolgálón kaptak helyet, teljesen elkülönítve a tanulókéitól: levelezőkiszolgáló – Sendmail; proxykiszolgáló – Squid; csomagszűrés – IP Chains; SSH – OpenSSH; kapufigyelés – portsentry; fájlváltozások ellenőrzése – Tripwire; naplók feldolgozása – logcheck, webalizer, sqmgrlog, http-analyze, továbbá saját fejlesztésű parancsfájlok. Érdekesség, hogy az SSH-t nemcsak távoli felügyeletre használják, hanem a tanárok közül néhányan ezen keresztül intézik a levelezésüket is. A gép két éve üzemel zavartalanul. Naponta készül mentés a fontos adatokról saját parancsállománnyal. A `find` segítségével megkeresik az új fájlokat, amelyeket a `tar` egy tárolófájlba rakja be, ezek a későbbiekben CD-ROM-ra kerülnek.

A harmadik gép egyelőre fejlesztési szakaszban van, adatbázis-kiszolgáló és X-terminál kiszolgáló lesz belőle. Hálózati kártyákból a következő típusokat használják: ne2000, rtl8029, rtl8139.

A gépek üzemeltetése sem időben, sem a munkában nem okoz gondot. Természetesen az új dolgok keresése és kipróbálása elég tennivalót ad az igazán érdeklődő rendszergazdáknak. Eddig csak előre tervezett módon történt gépleállás: magfrissítés vagy változatátállás miatt. Az üzemeltetés ideje alatt egyetlen merevlemez hibásodott meg, de a mag üzeneteiből ez is előre látható volt, így fel lehetett készülni a cseréjére.

A tervek között tűzfal készítése, valamint egy olyan terem felszerelése szerepel, ahol linuxos munkaállomások lesznek. A Debian-változatra való áttérést is fontolgatják. Arra kérdésre, hogy miért döntöttek a Linux mellett, a rendszergazda a következő választ adta: „A Linuxszal körülbelül 1995-ben kezdtem ismerkedni, és azóta is elkötelezett híve vagyok. Itt érzem azt, hogy ha valamit megteszek, az úgy működik, ahogy akarom. Megadja az alkotás örömét, amelyet a Windows az egérkattintgatás *nem tudom, miért, de ide kell kattintani* hozzáállásával elvett. Az informatikát karakteres felületen tanultam, ahol gondolkodni kellett, mert nem volt fölösleges hely és memória sem. A programnak kellett jónak lennie. A Linux az alkotás örömét adta vissza: tudod, mit akarsz csinálni, és meg is csinálod. Azt már meg sem említem, hogy távolról is mindent meg lehet tenni rajta, és ez fontos szempont.”

Fővárosi Oktatástechnológiai Központ

A használatban lévő gép egy HP NetServer LH3 Pentium II-es, 350 MHz-es processzorral, 256 MB RAM-mal felszerelve. Az operációs rendszer számára egy 4 GB-os SCSI-UW merevlemez biztosítja a helyet, az adatokat pedig öt 9 GB-os SCSI-UW merevlemezre bízták, amelyek RAID-5-be vannak kötve. A hálózati kapcsolatot egy 3Com 3c509B-TX hálózati kártya biztosítja. Slackware-terjesztésen a következő feladatokat oldották meg:

- belső fájlkiszolgáló – Samba,
- webkiszolgáló – Roxen Challenger,
- levelezőkiszolgáló – Sendmail,
- proxykiszolgáló – Squid.

Ezek mellett a gépnek még apróbb munkái is akadnak, DNS-kiszolgáló, nyomtatókiszolgáló, tűzfal, betárcsázó-kiszolgáló. Az általa kiszolgált felhasználók száma harmincra tehető.

A Pécsi Tudományegyetem Linux Konzultációs Központja

A központ helyileg a Természettudományi Kar Informatika és Általános Technika Tanszékén található. Három laborban látnak el Linuxok kiszolgálói feladatokat. Minden laborban 12 Sun Sparcstation X-terminál helyezkedik el, ezeket szolgálja ki a három gép, amelyek két 600 MHz-es Pentium III processzorral, 512 MB RAM-mal és 20 GB merevlemezzel vannak felszerelve. Mindhárom gép alkalmazáskiszolgálói feladatokat lát el, tehát rajtuk futnak a munkaállomásokon használt programok.
Fejlesztési terveik a következők: a közeljövőben három Sun gépet szeretnének beszerezni 400 MHz-es processzorral és 512 MB RAM-mal. Az elképzelések szerint ezeken is Linux fut majd fájlkiszolgálói és webkiszolgálói feladatokat ellátva. A tűzfal mögött e gépek egyikére várna a felhasználók nyilvántartása, a jelenlegi álláspont szerint NIS segítségével. Elképzelésük nyomán a levelezés lebonyolítását ugyancsak ezekkel a gépekkel oldanák meg, egyrészt webes felületen, másrészt a Pine segítségével. Távlati terveik között a 2001-es évben egy ötvenfős labor létrehozása szerepel, valamint ECDL-tanfolyamok tartása Linux alatt. A tananyag kidolgozása hátravan, az akkreditációs folyamatot még nem indították el.
A munkaállomásokat jelenleg alapfokú Linux- és (február elejétől) rendszergazdai tanfolyamokon használják, például C nyelv oktatására. A jövőben szakprogramok (elektronika, térképészet, fizika, matematika) is futni fognak rajtuk. Irodai programcsomagként a StarOffice-t (vagy talán az OpenOffice-t) választják, és természetesen ezeken a gépeken informatikusoktatás is zajlik majd.
Az alkalmazott operációs rendszer neve Renegát II, amely a RedHat Linux általuk készített magyarítása.

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

A HP meg(v)eszi a Compaqot

A hír mindenkit meglepett. A Hewlett-Packard megvásárolja a Compaqot, illetve annak 25 milliárd dollár értékű részvényét. A bejelentést követő napon a tőzsde visszahatása következtében már csak 20 milliárd dolláros üzletről beszélhettünk annak ellenére, hogy hasonló bejelentéseknek lehettünk tanúi: *„Az új HP* lesz a világon az első számú a kiszolgálók, a PC-k és a kézi eszközök, a képalkotás és a nyomtatás, az IT-szolgáltatások, a tárolórendszerek és a kezelőprogramok területén."
A számadatok megdöbbentőek: a házasulandó vállalatok összegzett bevétele az elmúlt négy negyedév során elérte a 87 milliárd dollárt, és ezzel az IBM mögött (90 milliárd dollár) a második a világon. Az új behemót a létszámcsökkentések előtt – melyek valószínűleg inkább a Compaqot fogják érinteni – 160 országban 145 000 alkalmazottal dicsekedhetett. Ha az üzlet létrejön, a Compaq már csak „márkanévként" él tovább, mint ahogyan a Netscape is az AOL általi felvásárlása után. A HP és a Compaq együttese a boltokban eladott PC-k hetven százalékát tudhatja a magáénak, de összességében még a társulás után is a Dell vezet, ami valószínűleg nyomós indok lehetett a felvásárlás mellett.
A szakmai sajtó visszhangja a legjobb esetben is vegyesnek volt mondható. *Dan Gillmor* a San Jose Mercury Newsban a következőt írta: „Nem mondható nagyon izgalmasnak egy olyan piacot vezetni, amely unalmas, innovációtól mentes és alig nyereséges".
Nagy-Britanniában, a The Registerben ezt olvashattuk: „Merészebb választás lett volna a HP részéről, ha új piacok meghódítására indul, megveszi például a Nokiát vagy a Nortelt, esetleg az ARM-et. De ahelyett, hogy előre nézne, a HP visszatekint és egy olyan céget vásárolt meg, amely nagyon hasonlít saját magára, csak anyagilag gyengébb lábakon áll."
A linuxos közösségben ennél pörgősebb volt a hangulat. *Dave Sifry*, a Linuxcare társalapítója nyilatkozta: „Mindenki mondta, hogy a linuxos piac megszilárdul, de ez azért egy kicsit több, mint amire számítottam!"
Dan Kusnetsky, a VP System Software for International Data Corp.-tól az vezető linuxos elemzők egyikeként a következőket mondotta:
„A közelmúltban mindkét cég megoldást keresett a programfejlesztéssel kapcsolatos kiadások csökkentésére, ezért nagymértékben egy harmadik fél gyártotta programokra tértek át. Ez érthető lépés. Elég drága mulatság világméretű operációs rendszerek szállítójának lenni."
Hozzáteszi, hogy az egyesült cégnek nem kevesebb, mint nyolc operációs rendszert kell támogatnia, ezért a következőképp vélekedik:

1. Mivel a Tru64 UNIX a legkevésbé elterjedt operációs rendszer a nyolc közül, és szorosan kapcsolódik az Alpha-rendszerekhez, jó tulajdonságait átemelik a HP-UX-ba és a Linuxba, majd a terméket megszüntetik.
2. Az OpenVMS és az MPE/ix jó tulajdonságait beolvasztják a Windowsba, és a két operációs rendszer közül az egyiket vagy mindkettőt megszüntetik.
3. Az új cég a nagyvállalati megoldások keretében HP-UX-ot fog szállítani, Windowst a munkacsoportok és egyéni felhasználók igényeinek kielégítése végett, és Linuxot a webalapú szolgáltatásokhoz.

Ez nem lenne rossz hír a Linux számára.

Doc Searls

A 2001. évi olvasói szavazás eredményhirdetése

Lássuk, mennyire értünk egyet a *Linux Journal* olvasóival!
A Hálózaton hat héten át tartó 2001. évi olvasói szavazás eredményeit böngészve megállapíthatjuk, hogy a Linux Journal olvasóinak mindenről van véleményük, mégpedig igen sokszínűen. Az is tisztán látszik, hogy több lehetőség, segédeszköz és módszer közül választhattunk, mint valaha – ezt jó tudni ilyen zűrzavaros időkben. Idén több mint 6500 olvasó szavazott 24 tárgykörben, a legkedveltebb Linux-kiadványtól az irodai programcsomagon és beszélgető fórumon át a mentési segédprogramokig. Mindenkinek köszönjük a részvételt, és most lássuk az eredményeket!

A legkedveltebb Linux-változat

1. RedHat
2. Debian
3. Mandrake

A RedHat kapta a szavazatok a harminc százalékát, és ezzel megismételte tavalyi győzelmét. A Debian a negyedikről a második helyre lépett elő, a Mandrake pedig maradt harmadik. A legtöbb jelölést a Linux from Scratch és a lengyel Polish Linux Distribution (PLD) kapta.

A legkedveltebb grafikus program

1. Gimp
2. xv/xview
3. CorelDraw

A várakozásoknak megfelelően a Gimp győzött a szavazatok 77 százalékával. A legtöbb jelölést az xFig és a Photoshop kapta, és nagyon sokan kérték a Photoshop átültetését Linux alá.

A legkedveltebb szövegszerkesztő

1. StarOffice
2. Abiword
3. Kword

A StarOffice megőrizte tavalyi első helyét. Legesélyesebb vetélytársa, az Abiword csak feleannyi voksot kapott, és mindössze 19 szavazat választja el a Kwordtől és az Emacstól. Bár nem nevezhető éppen szövegszerkesztőnek, a legtöbb jelölést mégis a LaTeX kapta.

A legkedveltebb karakteres szerkesztő

1. vim
2. vi (és klónjai)
3. GNU Emacs

Megfogadtuk a tavalyi szavazás során kapott tanácsotokat, így idén külön indítottuk a vi-t és a vim-et. Most a vim kapta a díjat, miután kétszer annyi szavazatot gyűjtött, mint a vi. A legtöbb jelölés az mceditre érkezett.

A legkedveltebb asztali környezet

1. KDE
2. Gnome
3. Window Maker

Ez volt az egyik legnépszerűbb tárgykör, és egyértelműen a KDE lett a nyerő a szavazatok negyven százalékával. A Gnome végzett a második helyen 24,5 százalékkal, a legtöbb jelölést pedig az xfer szerezte. Ne feledkezzünk meg azonban a parancssorról sem!

A legkedveltebb irodai programcsomag

1. StarOffice
2. KOffice
3. WordPerfect

Az idén, csakúgy, mint 2000-ben, a StarOffice lett a legkedveltebb szövegszerkesztő és irodai csomag. A tavalyi gyenge szereplés után a KOffice most erős második helyezést ért el. Az utólagos jelölések pedig az OpenOffice-t ajánlják mindenki számára melegen.

A legkedveltebb programozási nyelv

1. C
2. Perl
3. C++

Még egy tárgykör, ahol megfogadtuk a tanácsotokat és külön indult a C és a C++, hiszen „haver, a kettő nem egy és ugyanaz”! A Java és a PHP még belefért az első ötbe, és a Python is csak 15 szavazattal maradt le tőlük. Számos jelölés érkezett a Kylix/Object Pascal javára: több mint 200 szavazatot kapott.

A legkedveltebb fejlesztőeszköz

1. GCC
2. Emacs
3. KDevelop

A GCC megismételte tavalyi győzelmét, de lényegesen kisebb százalékkal nyert. Az Emacs rugalmassága itt is megmutatkozik. A múlt évi jelölők kedvence, a KDevelop most a harmadik helyet szerezte meg, míg a Borland Kylixa ebben az osztályban is gyakorta feltűnt a jelöltek között.

A legkedveltebb munkakörnyezet

1. bash
2. tcsh
3. ksh

A szavazók 81 százaléka választotta a bash-t legkedveltebb munkakörnyezetül, míg a tcsh csak messziről követi a második helyen. A ksh harmadikként végzett, mindössze öt szavazattal előzve meg a negyedik helyen álló zsh-t.

A legkedveltebb processzor

1. AMD Athlon
2. Intel Pentium
3. PowerPC

Olvasóink választott processzora, az AMD Athlon a szavazatok 42 százalékát söpörte be. Sok jelölés érkezett az AMD Duron, K6-II és a Celeron javára is. A Pentiumra szavazók közül jó néhányan megjegyezték, hogy inkább szükségből választották, semmint a teljesítménye miatt.

A legkedveltebb kapcsolatfelület

1. Cyclades
2. Equinox
3. Digi International

Ebben a tárgykörben szavaztak a legkevesebben, mégpedig – a megjegyzésekből ítélve – azért, mert sokan nem tudták, mire is gondolunk. Nos, nem az ókori Róma főterére (Forum Romanum) és nem is valamelyik kávéházra. Azok közül, akik megértették szándékunkat, a Cyclades mellett voksoltak a legtöbben.

A legkedveltebb adatbázis

1. MySQL
2. PostgreSQL
3. Oracle

Sorozatban másodszor verte meg a MySQL a PostgreSQL-t 2 az 1-hez arányban. Együttesesen a szavazatok csaknem nyolcvan százalékát kapták. Sokan jelölték a (PostgreSQL-re támaszkodó) RedHat Database-t és a GemStone/S-t is.

A legkedveltebb mentési segédprogram

1. tar
2. Amanda
3. Arkeia

A tar fölényesen nyerte a legkedvel-

tebb mentési eszköz címet. Az Amanda és az Arkeia áll a második, illetve a harmadik helyen, csupán egyetlen szavazatnyi különbséggel. Sokan használják a BRU-t, a dump-ot és egyéb saját készítésű mentési programokat.

A legkedveltebb Linux Journal-rovat

1. Cooking with Linux
2. Kernel Korner
3. At the Forge

Marcel Gagné, a Cooking with Linux (Fogadó a Linuxhoz) rovat szerzőjének rajongói alighanem teljes létszámban részt vettek az idei szavazáson, és az első helyre juttatták kedvencüket. Második lett a változó szerzőjű Kernel Korner (Szaktekintély). Nagyon sok szavazó legkedveltebb rovata – és ezt nem csak kitaláltuk – „mindegyik".

A programozók legkedveltebb itala

1. kávé
2. víz
3. tea

Ki gondolta volna, hogy ez a tárgykör váltja ki a legnagyobb vitát? Akadt olyan ital, amit a tavalyi közfelháborodás hatására idén felvettünk a listára, erre jól leszidtatok minket, amiért más üdítőkről elfeledkeztünk. Mint kiderült, mindannyian a kávé rabjai vagyunk, és száznál is többen bátran vállalták, hogy az újmódi édes-habos kávéfélékkel élnek.

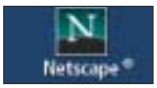
A legkedveltebb linuxos játék

1. Quake III
2. xBill
3. Tux Racer

Egyes vélemények szerint a játékok nem a komoly Linux-felhasználók számára készültek, de kétségkívül jól jövedelmeznek és ösztönzik a fejlesztéseket. És ismerjük el, néha mindannyiunknak jár egy kis szórakozás is, nem? Idén ismét a Quake III vitte el a pálmát, a jelölések listáját pedig a klasszikusok: a Mahjongg és a Shisen-sho vezetik.

A legkedveltebb webböngésző

1. Netscape
2. Mozilla
3. Konqueror



A Netscape a szavazatok harminc százalékával büszkélkedhet, mögötte kullog a Mozilla (minden hibájával együtt) alig több mint 300 szavazattal. A legtöbb jelölést az Internet Explorer mondhatja a magáénak, de sokak egybehangzó véleménye szerint „mind szivacs, csak másképpen".

A legkedveltebb linuxos webhely

1. Slashdot
2. Freshmeat
3. LinuxToday

A jelöltek listájáról hosszasan sorolhatnánk a Linux-termékeket és szolgáltatásokat ajánló vállalati webhelyeket, nemzetközi és szakmai közösségi weboldalakat. Természetesen a szavazók nagy része, csaknem harminc százalékuk elsőként a Slashdot oldalain próbálkozik. A legtöbb jelöléssel pedig egy lengyel linuxos webhely, a ➲ http://www.linuxnews.pl dicsekedhet.

A legkedveltebb levelezőügyfél

1. Netscape
2. KMail
3. Pine

Bár ennek a tárgykörnek is megvannak a győztesei, egyik sem bír elsöprő fölénnyel. Alig 70 szavazat választja el az első helyen álló Netscape-et a harmadik helyezettől. Az elm, a hajdani kedvenc, a lista aljára zuhant vissza.

A legkedveltebb csevegőügyfél

1. Xchat
2. Jabber
3. BitchX

A hálózati csevegésben résztvevők közül négy százalékkal többen választották az Xchatet, mint a Jabbert. A legtöbb jelölés a Licq és a GnomeICU javára érkezett. Sokan megvetően nyilatkoztak a csevegőprogramok és a gyorsüzenők minden fajtájáról. Úgy tűnik, még többen folyamodnak azonban az AOL, a Yahoo vagy valamelyik MS-változat használatához, mondván: „minden barátom ezt használja".

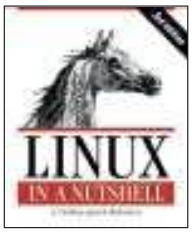
A legkedveltebb osztott fájlkezelő rendszer

1. Gnutella
2. Freenet
3. OpenNAP

Egy éve Amerika-szerte még minden kávézóban és bíróságon erről folyt a vita. Nos, a Metallica szerencsésen elkerülte a szegényházat, és a Gnutella és a Freenet maradt a legnépszerűbb fájlmegosztási megoldás. A jelölések között az audiogalaxy és a MORPHEUS fordult elő leggyakrabban.

A nélkülözhetetlen Linux-könyv

1. *Ellen Siever*: „Linux in a Nutshell"
2. *Matt Welsh* et al.: „Running Linux"
3. *Vicki Stanfield* et al.: „Linux System Administration"



Az örökös kedvencek, név szerint a „Linux in a Nutshell" és a „Running Linux" ismét a dobogó két felső fokára állhattak, bár a tavalyihoz képest fordított sorrendben, ugyanis most az előbbi lett a győztes. A hálózatról elérhető különböző kézikönyvek és leírások vezetik a jelöltek listáját, s úgy tűnik, az összes valaha megjelent Linux-könyv legalább egy szavazatot kapott.

A legkedveltebb reklámszűrő eszköz

1. Junkbuster
2. Homemade
3. Squidguard

Valaki a következőt írta nekünk: „A reklámok a weboldalak szerkesztésének fontos elemei, és igenis arra valók, hogy megnézzék őket." A jelek szerint egyedül maradt a véleményével, mivel mindenki más valamilyen szűrőt használ, legtöbben a Junkbustert említették. Sokan saját szűrőt készítenek.

A legkedveltebb hanglejátszó (audio) eszköz

1. xmms
2. pmg123
3. RealAudio

A szavazatok 58,4 százalékával újra az xmms lett olvasóink legkedveltebb hanglejátszó eszköze. Ehhez képest egyetlen másik eszköz sem tudhat magáénak a szavazatok kilenc százalékánál többet. A jelöltek között a leggyakrabban a Grip bukkan fel, és még mindig eléggé sokan esküsznek az ogg-eszközökre is.

Heather Mead a Linux Journal társszerkesztője. Szabadidejében fotózik, filmeket néz, zenét hallgat, valamint novellákat ír.

Új termékek

CapeStudio RAD-eszköz

A CapeStudio RAD webes szolgáltatások kifejlesztésére és telepítésére használható. A CapeStudio Java- vagy Visual Basic-kódot gyárt a webes szolgáltatást leíró nyelven (WSDL) írott fájlokból. Ezek a fájlok írják le a webes szolgáltatás felületét, segítségükkel mérséklődik a szolgáltatás megvalósításához szükséges idő. A CapeStudiónak egy grafikus környezet is a része, amellyel XML- és SOAP-üzenetek között kétirányú átalakításokat adhatunk meg. A fejlesztőkörnyezet minden szükséges összetevőt magában foglal, és bármely webes szolgáltatófelületen működőképes, amely támogatja az XML, SOAP, WSDL és UDDI ipari szabványokat.

Adatok: Cape Clear Software, Inc., 900 East Hamilton Avenue, Suite 100, Campbell, California 95008, telefon: 866-227-3226 (ingyenes), e-mail: info@capeclear.com, ➲ http://www.capeclear.com

Beowulf Professional Edition

A Scyld Computing kiadta Beowulf Professional Editiont, a géptelepek operációs rendszerét, amely kicsomagolás után azonnal használatba vehető. A Professional Edition egyszerűbbé teszi a géptelep üzembe helyezését és felügyeletét. Az eredeti Beowulf fejlesztőcsapata által kiadott leírás és az általuk nyújtott terméktámogatás szintén a csomag része. A mostani változat újdonsága a teljes Alpha-támogatás, a Myrinet és a Gigabit ethernettámogatása, a kötegelt sorrendszer (BBQ), a webalapú rendszerfelügyelet és feladatmegfigyelés, valamint a *PVFS*, az *NFSv3* és a *ROMIO* fájlrendszerek használata.

Adatok: Scyld Computing Corporation, 410 Severn Avenue, Suite 210, Annapolis, Maryland 21403, telefon: 410-990-9993, e-mail: sales@scyld.com, ➲ http://www.scyld.com

TurboLinux az IBM eServer kiszolgálóhoz

A TurboLinux 6.5 támogatott terjesztés az IBM *eServer iSeries* és *pSeries* rendszereire, amelyet kicsi és közepes méretű vállalkozások számára terveztek. A TurboLinux 6.5 képes tűzfal-, webkiszolgáló, fájl- és nyomtatókiszolgáló, valamint levelezési feladatok ellátására. Az egységes alapkód, amely támogatja a Li18nux és az LSB szabványokat, egyszerű telepítést biztosít. Egyetlen iSeries kiszolgáló 31 különálló Linux-kiszolgálót képes futtatni. Minden Linux-kiszolgáló a saját tartományában fut, és képes a különböző erőforrások megosztására (processzor, lemez, szalag, CD-ROM, DVD és helyi hálózat) az iServeren futó más alkalmazásokkal.

Adatok: TurboLinux, 8000 Marina Boulevard, Suite 300, Brisbane, California 94005, telefon: 650-228-5000, ➲ http://www.turbolinux.com

Black Adder 1.0beta3

Megjelent a Black Adder 1.0 beta3 kiadása, amely Linux, illetve Windows alatt teszi lehetővé Qt-alapú Python- és Ruby-alkalmazások fejlesztését. A Black Adder ötvözi a látványalapú tervezést, a hibakeresést, a nyelvi elemek kiemelését, az ODBC felületet és a mindenre kiterjedő HTML-leírást – ezek együttesen minden igényt kielégítő felületet nyújtanak a Python- és Ruby-alkalmazások fejlesztéséhez. Újdonságok többek között: a Qt 2.3.1, valamint a Python 2.0.x és a Python 2.1.x támogatása.

Adatok: theKompany.com, PO Box 80265, Rancho Santa Margarita, California 92688, telefon: 949-713-3276, e-mail: sales@thekompany.com, ➲ http://www.thekompany.com

Cleanscape SourceMill

A Cleanscape Software bejelentette a Cleanscape SourceMill megjelenését, amely egy forráskód-létrehozó program. Segítségével az alkalmazások fejlesztése és módosítása felgyorsítható, mert önműködővé teszi az ismétlődő programozási feladatokat. Kereskedelmi színvonalú forráskódot hoz létre objektummodellekből és kódmintákból. A SourceMill használatával a fejlesztők olyan alkalmazás-keretrendszert hozhatnak létre több felületre, amely az egyes alkalmazásokat sablonokból alkotja meg. A SourceMill az egységes és szabványos programozás elősegítésére is felhasználható.

Adatok: Cleanscape Software International, 2231 Mora Drive, Suite E, Mountain View, California 94040, telefon: 650-864-9600, e-mail: sales@cleanscape.net, ➲ http:// www.cleanscape.net

Recital Linux Developer

A Recital Linux Developer rendszer egy teljes többfelhasználós adatbázis-kezelőt foglal magába, 4GL-t, valamint karakteres felületen működő linuxos alkalmazások fejlesztését és telepítését lehetővé tevő eszközkészletet. A csomag az adatok és a nyelv szintjén FoxPro-, FoxBASE- és Clipper-megfelelő, így a meglévő alkalmazások átvehetők. A RAD-környezet része a beépített adatbázis, képes Javát használni, támogatja a POP3- és az SMTP-protokollokat, valamint XML kezelésére is alkalmas. A Recital, FoxPro, dBase, Informix és DB2-adatok elérése szabványos xBase-parancsokon és a felhasználó által megadott űrlapokon keresztül valósul meg.

Adatok: Recital Corporation, Inc., 85 Constitution Lane, Danvers, Massachusetts 01923, telefon: 800-873-7443 (ingyenes), e-mail: info@recital.com, ➲ http://www.recital.com

FlexeLint C/C++-hoz

Megjelent a FlexeLint for C/C++ 8.0 változata. A FlexeLint forráskód-elemző segítségével C és C++ programokat elemezhetünk, feltárhatjuk a hibákat és következetlenségeket, ezáltal programjaink karbantarthatók és hordozhatók lesznek. A 8.0 változat újdonságai között található a változók értékének függvények közötti követése, a továbbfejlesztett kivételkezelés, a MISRA szabályrendszer követésének ellenőrzése, valamint további húsz új lehetőség. A további ellenőrzések közé tartozik a felhasználó által megadott függvények értelem szerinti ellenőrzése, a futás folyamatát figyelembe vevő mutatók követése és a változók értékadásának elemzése.

Adatok: Gimpel Software, 3207 Hogarth Lane, Collegeville, Pennsylvania 19426, telefon: 610-584-4261, e-mail: sales@gimpel.com,
➲ http://www.gimpel.com

Zend Accelerator

A Zend Technologies bejelentette az Accelerator 2.0 megjelenését.

Ezt a sokak által már jól ismert kiszolgálóoldali gyorstárazó programot a dinamikus PHP-parancsfájlok válaszidejének csökkentésére tervezték. Az Accelerator 2.0 egy olyan modult is tartalmaz, amellyel a rendszergazdák lemérhetik a kiszolgáló valós idejű sebességnövekedését. A PHP4 új CGI-támogatásának segítségével több menetben is hatékonyabbá tehető. A csomag telepítése rendkívül egyszerű. A kódok párhuzamosításának köszönhetően az Acceleratorral akár háromszoros sebességnövekedést is elérhetünk, a tervezők a késleltetési időt szinte nullára csökkentették. További kényelmi szempont, hogy a nagyon ritkán használt, de memóriaigényes parancsfájlokat kihagyhatjuk.

Adatok: Zend Technologies, Inc., 11 Penn Plaza, 5th floor, Suite 5013, New York, New York 10001, telefon: 877-936-3872, e-mail: info@zend.com,
➲ http://www.zend.com

Tarantella Enterprise 3 Starter

A Tarantella Inc. megjelentette a Tarantella Enterprise 3 Starter csomagot. Ezt a hálózatkezelő programot kisebb vállalatok, üzletek

számára tervezték. Sokféle feladatra használhatjuk, például a kiszolgálók távoli karbantartására, alkalmazások felületfüggetlen elérésére. A további hasznos szolgáltatások között találjuk a webes alkalmazások egyesített kezelésével kialakítható hálózatokat és a vezeték nélküli átjárót. A Tarantella Enterprise 3 Starter több Linux-terjesztés számára hozzáférhető, a próbaváltozatot letölthetjük a cég honlapjáról.

Adatok: Tarantella, Inc., 425 Encinal Street, Santa Cruz, California 95061, telefon: 888-831-9700,
➲ http://www.tarantella.com

HP x2000- és x4000-munkaállomások

A HP bejelentette, hogy linuxos változatban is piacra dobja x2000 és x4000 típusú munkaállomásait. Az x2000 alapját az Intel 850-es lapkakészlete és egy 1,7 GHz-ig működtethető Pentium IV-es processzor képezi, míg az x4000 az Intel 860 köré épül, és akár két Xeon processzort is elhelyezhetünk benne. Mindkét rendszer ISV tanúsítvánnyal rendelkező 2D-s és 3D-s alkalmazások használatát teszi lehetővé. Az x2000 és x4000 típusokat leginkább grafikai és nagy memóriaigényű műszaki alkalmazások számára tervezték, ilyen például a digitális tartalomkészítés vagy az animáció. Mindkét rendszert RedHat 7.1-gyel szállítják és teljes körű segítségnyújtást vállalnak hozzájuk. A gépek összes jellemzőjét és az optikai részegységeket a ➲ http://www.hp.com/workstations/products/linux címen tekinthetjük meg.

Adatok: Hewlett-Packard, 3000 Hanover Street, Palo Alto, California 94304, telefon: 800-752-0900,
➲ http://www.hp.com

StuffIt Engine SDK

Az Aladdin Systems terméke a StuffIt Engine Software Developer Kit, melynek segítségével a fejlesztők tömörített állományok kezelését adhatják projektjeikhez. Az SDK használatával több fájl röptében tömöríthető egyetlen önkicsomagoló tárfájlba, illetve csökkenthető a fájlok küldéséhez és fogadásához szükséges idő. A támogatott tömörítő és kódoló eljárások a következők: *StuffIt*, *Zip*, *Gzip*, *Tar*, *Rar*, *Bzip2*, *Uuencode*, *UNIX Compress*, *BinHex*, *MacBinary*, valamint néhány további eljárás is. Az összes formátum egységes programozói felületen keresztül érhető el.

Adatok: Aladdin Systems, Inc., 245 Westridge Drive, Watsonville, California 95076, telefon: 831-761-6200, e-mail: info@aladdinsys.com,
➲ http://www.aladdinsys.com

Network Security Services

A Network Security Services (NSS) egy olyan könyvtárakból és segédeszközökből álló csomag, mely a titkosítást és a biztonságos hálózati forgalmat érintő alkalmazások felületfüggetlen fejlesztésében nyújthat jelentős segítséget. Az NSS-könyvtárak C nyelvű API-t hoznak létre, melyet C/C++ alkalmazásokból hívhatunk meg – segítségével titkosítási eljárásokat végezhetünk, tanúsítványokat kezelhetünk, S/MIME-üzeneteket küldhetünk és fogadhatunk, illetve biztonságos hálózati forgalmat bonyolíthatunk le az SSL, valamint a TLS használatával. Az NSS a v.2 és v.3-változatú SSL-t, a TLS-t, az 5-ös, 7-es, 11-es és 12-es számú PKCS-t, az x.509 v.3 tanúsítványokat támogatja, de gépi (alkatrészes) titkosítóeszközöket és intelligens kártyákat is kezelhetünk a segítségével. A forráskód és a futtatható változat egyaránt letölthető.

Adatok: Network Security Services, The Mozilla Organization,
➲ http://www.mozilla.org/projects/security/pki/nss

A hónap szakmai tanácsai

Óriási „malloc()”

Szeretném megérteni, hogy milyen tényezők határozzák meg egy Linux alatt futó folyamat által használható tárterület nagyságát. 1,2 GHz-es Athlon processzort használok 1,5 GB RAM-mal és 2 GB lapozóterülettel. Operációs rendszerem RedHat 7.1 2.4.3-12 rendszermaggal. A rendszer induláskor mind az 1,5 GB-ot látja, és ennyit jelez a `top`, valamint más segédprogramok is. Egy folyamat számára azonban nem tudok nagyjából 940 MB-nál többet lefoglalni. A folyamat által lefoglalható tárterület nagyságát egy olyan egyszerű C-programmal vizsgálom, amely egyetlen nagyméretű karaktertömb számára foglal helyet.
Ned Piburn, npiburn@oti.gd-ots.com

A Linux-rendszermagban beállítható, hogyan ossza fel a tárat a rendszermag és a felhasználói programok között. Lehetséges, hogy az általad használt rendszermag úgy lett építve, hogy 3 GB-ot ad a rendszermagnak és 1 GB-ot a felhasználói programoknak. Amikor a rendszermag beállításait módosítod (`cd /usr/src/linux; make menuconfig`), nézd meg a *Processor type and features/Maximum Virtual Memory* beállítást, és itt adj meg 3 GB-ot (ugyanis bizonyos foltozott rendszermagok 2 GB-tal hibásan működnek).
Marc Merlin, marc_bts@valinux.com

A GNU libc a `brk()`-t használja a kis helyfoglalásokhoz és az `mmap()`-ot a nagy helyfoglalásokhoz. A `brk()` használatával nagyjából 900 MB foglalható le. Kevesebb nagyméretű helyfoglalás sikeres lehet ott, ahol sok kis helyfoglalás sikertelen volt.
Ha ez a gondod, megoldást jelenthet egy saját `malloc()` írása. Ez vagy az `mmap()`-ot használná mindig, vagy először `mmap()`-okkal nagy darabokat foglalna le, és ezekbe helyezné el a kisebb részeket.
Scott Maxwell, maxwell@ScottMaxwell.org

A `mallopt` függvény leírása megtalálható a libc GNU leírásában a *Malloc Tunable Parameters* fejezetben. Állítsd be az `M_MMAP_THRESHOLD`-ot, hogy a `malloc()` mindig `mmap()`-ot használjon `brk()` helyett.
Don Marti, dmarti@ssc.com

Az RPM nem tudja frissíteni az RPM-et

Jelenleg RedHat Linux 6.2-t használok RPM-3.0.3-mal. Az RPM-3.0.3-ról RPM-4.0.2-re történő frissítés céljából a db3-3.1.17 csomagot az előírásoknak megfelelően próbáltam telepíteni, de az alábbi hibaüzenetet kaptam:

```
rpm can only install packages with
major version number <= 3
```

Atul, atul_info@yahoo.com

Telepítsd a 3-as sorozatú RPM legfrissebb kiadását, az az RPM3 és az RPM4 csomagokkal egyaránt boldogul. Letöltheted az ➲ ftp.rpm.org/pub címről.
Keith Trollope, keith@wishing-well.demon.co.uk

Adaptec SCSI-kártya RedHat 6.2 alatt

A RedHat 6.2 önműködően felismeri az *Adaptec 29160* kártyát, és az *AIC-7xxx* dinamikus modult hozzáadja a */etc/conf.modules* fájlhoz. Amikor azonban egy SCSI-merevlemezt csatlakoztatok a kártyához, a */dev/sda* az `fdisk` számára nem érhető el. Az eszközfájl létezik, az `fdsik` azonban nem fér hozzá. Ha RedHat 7.1-et indítok, az felismeri az SCSI-lemezt és képes használni. Ezt a SCSI-kártyát RedHat 6.2 alatt kell használnom – hogyan lehetne működésre bírni?
Joshua, cschen@asiaa.sinica.edu.tw

Megoldást jelenthet, ha a RedHat 7.1 rendszermagját RedHat 6.2 alá telepíted. Néhány további csomagot is frissítened kell, például a *modutils*-t.
Marc Merlin, marc_bts@valinux.com

Ugyanez volt a gondom, amikor az *Adaptec 29160* kártyámat megkaptam (mellesleg nagyszerű kártya). A Linuxot ideiglenesen egy IDE-meghajtóra telepítettem, letöltöttem a legfrissebb 2.2 sorozatú rendszermagot a kernel.org egyik tükörkiszolgálójáról, és az *AIC-7xxx* meghajtóprogramot a rendszermagba építettem, azonban nem modulként fordítottam le. Az új rendszermaggal indítottam újra a gépet, majd mindent átmásoltam a SCSI-meghajtóra.
Don Marti, dmarti@ssc.com

Ne lehessen kiadni „cd ..”-t!

Mit tehetnék, hogy a felhasználó könyvtára gyökérkönyvtárként viselkedjen, azaz a felhasználó csak azt és annak alkönyvtárait érhesse el?
Rafael, rafaelss@ig.com.br

A `chroot`-ra van szükséged. Számos FTP-démon alapértelmezetten a `chroot`-ot használja. Ha `telnet`-tel (vagy ami még jobb, SSH-val) szeretnéd használni a `chroot`-környezetet, akkor `chroot`-héjat kell készítened. További adatokért lásd a ➲ http://www.freshmeat.net/projects/jail_cp címet.
Ben Ford, ben@kalifornia.com

Ha a felhasználót bezártad mondjuk a */home/user* könyvtárba, a */lib* és a */bin* könyvtárak bizonyos részeit elérhetővé kell tenned a */home/user* alatt, különben a felhasználó semmilyen parancsot sem fog tudni futtatni.
Marc Merlin, marc_bts@valinux.com

A tűzfal indítása órákig tart

RedHat 6.0 rendszerem magját 2.2.16-3-ra frissítettem annak érdekében, hogy az IP Chainst használni tudjam. A ➲ http://www.redhat.com címen található útmutatásokat követtem. Most nagyszerűen működik tűzfalként. Ha elmegy az áram, újraindítás után a ➲ http://www.linuxdoc.org címről letöltött */etc/rc.d/rc.firewall* parancsfájlt futtatom, ez azonban legalább egy óráig fut. Létezik valamilyen módszer a folyamat felgyorsítására?
Paul Lamping, palamping@yahoo.com

Nagy merevlemezt használsz ext2-es fájlrendszerrel? Ebben az esetben a hosszú folyamat nem mást takar, mint a hibás rendszerleállás utáni fájlrendszer-ellenőrzést. Ennek kiküszöbölésére többféle módszer is létezik. Próbálkozhatsz naplózó fájlrendszerrel vagy használhatsz szünetmentes áramforrást (UPS-t), így áramszünet esetén elég időd marad a biztonságos rendszerleállításra. Megoldást jelenthet az is, ha a fájlrendszert úgy szervezed, hogy minél kevesebb könyvtárra adsz írási jogot, a legnagyobb részt pedig csak olvashatóvá teszed.
Ben Ford, ben@kalifornia.com

A tűzfal felállásának pillanatok alatt meg kell történnie, semmiképpen nincs szükség órákra, de még percekre sem. A fő gondok egyikét az okozhatja, ha a különböző gépek elérésére neveket és nem IP-címeket használsz. Ha az indítófájl a tűzfalparancsfájl lefutása után indítja el a névkiszolgálót, akkor minden sorban időtúllépés jön létre, melyben a DNS használatára van szükség. Akkor is ez lehet a baj, ha olyan névkiszolgálóra hivatkozunk, amelyet a tűzfalparancsfájl még azelőtt kizár, mielőtt a megfelelő engedélyező bejegyzések létrejönnének.
Chad Robinson, crobinson@rfgonline.com

Hogyan változtassam meg a Samba-jelszavakat?

Beállítottam a Sambát és a kiszolgálóm meg is jelenik a Win98 munkaállomások „Teljes hálózat" listájában. A kiszolgálón lévő könyvtárak is láthatók, de az eléréshez jelszót kér. Próbálkoztam azzal, hogy a Sambában az adott megosztott könyvtárakhoz kikapcsolom a jelszót, de nem tudok rájönni, pontosan hol is kell ezt megtennem.
Dan Schmeh, dschmeh@turningpnt.org

Az alapértelmezett telepítés a */etc/passwd* fájlt használja. Ha az alapértelmezett telepítést választottad, a Windows98-as gépeken a regisztrációs adatbázist is módosítani kell, hogy titkosítás nélküli jelszavakat használhass. Hozd létre az alábbi kulcsot:

```
HKEY_LOCAL_MACHINE\System\CurrentCont
➥rolSet\Services\VxD\Vnetsup\
➥EnablePlainTextPassword = 1
```

Ezt követően indítsd újra a rendszert.
A titkosított és nem titkosított jelszavakról kitűnő leírás olvasható a Samba-csomag leírásában. RedHat 7.0 esetén a */usr/share/doc/samba-2.0.7/docs/textdocs/ENCRYPTION.txt* fájlt keresd.
Christopher Wingert, cwingert@qualcomm.com

Nincsen telefonom, de kapcsolódnom kell!

Az Egyesült Államok egyik legnagyobb és legszegényebb megyéjében élek, a közelben sehol egy elektromos vagy telefonvezeték. A hozzám hasonlóan a kanadai határ közelében, magas hegyek között élő emberek nemrégiben jutottak számítógéphez, de az internet-hozzáférés távoli álomnak tűnik.
Bár a DirecPC képes lenne a műholdvevő tányért a lefelé irányuló adatforgalomhoz használni, ehhez a módszerhez azonban még mindig szükség van egy, a felfelé menő adatokat továbbító telefonvonalra. A legtöbbünknek még vezetékes telefonja sincs, bár egy, a Ham rádiónál olcsóbb rádiós megoldás igen nagy változást jelenthetne az életünkben. Bennünket inkább a letöltés érdekel, feltölteni jóval kevesebb adatot szeretnénk. Hogyan oldhatnánk meg a gondjainkat?
Robert Thomas, homeschoolu@hotmail.com

A DirecPC-t Linuxról futtathatod a ➲ http://www.helius.com-ról letöltött útvonalválasztóval együtt. A tervek szerint 2001 harmadik negyedévében dobják piacra a kétirányú DirecPC-hez való kétirányú útvonalválasztót. Javaslatom, hogy a szomszédaiddal közösen állíts fel egy helyi hálózatot, s egy gyorstárazó kiszolgálóval növeld meg a teljesítményt.
Ben Ford, ben@kalifornia.com

Az elzárt közösségek internetelérését általában WiFi-kártyákkal és módosított műholdvevő tányérokkal oldják meg. Ezek a nagy sebességű kapcsolatok több mérföldet képesek átfogni. A módszerrel a műholdas internetelérést egyszerűen megoszthatod a szomszédaiddal. Látogass el a ➲ http://www.toaster.net/wireless/community.html honlapra.
Don Marti, dmarti@ssc.com

A nyomtatás nem működik tovább

Eddig tökéletesen sikerült nyomtatnom egy Jetdirectet használó, helyi hálózatba kötött HP LaserJet 4Plus nyomtatóval úgy, hogy csak az IP-címét kellett tudnom. Nemrégiben azonban leállt. Az összes `lp` parancs (`lpq`, `lpr` stb.) az alábbi hibaüzenetet eredményezi:

```
Printer 'lp@localhost' - cannot open
➥connection
Connection refused
Make sure LPD server is running on
➥the server
```

Murray Zangen, murray@nj.com

Próbáltad a nyomtatót újraindítani? Ez a hiba a nemrég felbukkant CodeRed IIS féregvírus jelenlétét jelzi. Számos nyomtatót és Cisco-eszközt padlóra küldött már.
Ben Ford, ben@kalifornia.com

Úgy tűnik, hogy a gond magával a géppel van, nem pedig a HP nyomtatóval. Az `lp` azt akarja mondani, hogy az LPD nem fut a helyi kiszolgálón, amelyhez mindenképpen kapcsolódnia kell, hogy a távoli nyomtatóhoz intézett kérelmeket nyomtatási sorba rendezhesse. Ellenőrizd a `ps ax` paranccsal, hogy az LPD megfelelően működik-e. Ha igen, akkor a nyomtatóval való kapcsolatot is ellenőrizd: Telnettel lépj be a */etc/printcap* által meghatározott IP-címre. Végül győződj meg arról, hogy a megfelelő nyomtatási sort választottad-e ki és végezz próbanyomtatást az `lp` paranccsal (tehát ne valamilyen alkalmazásból).
Chad Robinson, crobinson@rfgonline.com

A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tükröoldalait, 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.

Fogadjunk a bazárra!

A Linux népszerűbb Jézusnál? A mérvadónak számító Google kereső szerint igen.

Amennyiben a Google még nem lenne az első számú keresőgép, akkor jó eséllyel azzá válhat. Mit mondhatunk róla? Néhány hónappal ezelőtt még a Yahoo is beadta a derekát és elkezdte támogatni a Google-t. Tízezernél is több linuxos gép küld ki keresőrobotokat, amelyek több mint 1, 3 milliárd oldalt járnak be, és olyan mindent kimerítő eredményekkel térnek vissza, amely lehetővé teszi, hogy valamennyi oldalt kereshető formában tárolják.
Nagyon sok kép mutat a Google-ra, és a Usenet hírcsoportok témáiban is előkelő helyet foglalnak el, annak ellenére, hogy nem népszerűsítik magukat, sőt (ezen írás idején) még mindig zajlik a bétaállapotú rendszer nyilvános fejlesztése.
A Google oldalain történő reklámozás inkább az újságok tárgykörökre osztott hirdetéseihez hasonlít, mint a hirdetőtáblák, magazinok vagy a televízió hirdetéseire. A megállapítást elismerésnek szántam. Jelenleg talán csak ők képviselik azt a fajta reklámozási formát, amelyre komoly igény mutatkozik, így a Google-nak megvan rá az esélye, hogy az első csapat legyen, amely olyan webalapú reklámfelületet kínál, amelyet még a felhasználók is örömmel fogadnak – kétségtelenül látványos tett lenne, ha sikerülne elérniük.
Keresési eredményeik rangsorolását nem annyira ravasz metatagok vagy az oldal készítője által bevetett HTML-trükkök befolyásolják, sokkal inkább az adott oldalra mutató hivatkozások száma. Ha valamit biztosan elmondhatunk a Google keresési eredményeiről, akkor az az, hogy az eredmények az oldal elismertségét tükrözik. Mivel a weboldalak közötti összekapcsolást a legtöbb szövegfeldolgozó program nem önműködően végzi, az oldal készítője részéről a kapcsolatok létrehozása külön odafigyelést igényel. A google-i gondolatmenet szerint tehát úgy tekinthetünk az egy weboldalra mutató minden egyes hivatkozásra, mintha egy szavazat lenne, amellyel az érdeklődők elismerik az oldal hasznosságát.
A Google-nál a saját rendszerüket úgy emlegetik, mint „gép és program különleges ötvözetét". Egész pontosan tízezer Linux-alapú gépről van szó. A „miért pont Linux?" kérdésre valamennyien tudjuk a szakmai választ. Létezik azonban egy nyilvánvaló gazdasági válasz is: névtelen kereskedelmi gépekre olcsón lehet telepíteni. És ami még ennél is fontosabb: Linuxot használva csökkenthetők a fizikai rendszerrel szemben

támasztott követelmények. A Google szakemberei számára a rendszert könnyebb volt Linux-alapokon elképzelni, mint bármi máson. Végezetül egy demokratikus válasz is létezik a kérdésre, és ezt a keresési eredmények adják meg.
A Google azon túl, hogy a keresési eredményeket rangsorolja, megjeleníti, hogy hány olyan oldalt talált, amely az adott kulcsszót, vagy kulcsszavakat tartalmazza. A néhány perce lefuttatott keresésem eredményeképpen (amikor az Linux-indexeket egyszerre használtam) a Google a „linux" szót 31,6 millió oldalon találta meg. Mit is jelent ez valójában?
A keresés két legelső eredménye (231 millióból) a LinuxGames.com és ***Richard M. Stallman*** „Miért nem szabadna, hogy a programoknak tulajdonosa legyen" című oldala volt.
Ha a piacot a szóbeszédek éltetik, ennek fényében mit mondhatunk a Linux-piacról? Valóban „nagyobb" lenne a mi Linux-bazárunk, mint a szex vagy a Microsoft? Nyilvánvalóan.
A Linuxszal kapcsolatban álló piac ugyanis folyamatosan változik. Nem számít, hogy a Microsoft milyen fontosnak próbálja feltünteti magát (ügyeskedve növeli a számára hivatalból kijáró fontosság látszatát), hiszen az a zárkózottság, amivel a cég saját szellemi termékeit körülveszi, behatárolja azokat a témákat, amelyekről a Microsoft kapcsán egyáltalán beszélni lehet.
Határa van ugyanis annak, hogy valaki mennyire élvezheti a tudatot, miszerint ő a Microsoft programjainak szakértője. Míg a Microsoft szellemi tulajdonának nagy részéhez senkinek semmi köze nem lehet, addig a Linuxhoz bárkinek. Ez azt jelenti, hogy a Linux – mint eredeti szakterület – jobb üzlet, mint a Microsoft vagy bármely más olyan programfejlesztő cég, amely pont azon programjaival kapcsolatos párbeszédet korlátozza, amelyektől mindenkit függővé akar tenni.
Ez az, amit észben kell tartanunk, amikor a linuxos cégek életrevalóságát fontolgatjuk. Igaz, sokuk a padlóra került egyéb cégekkel együtt, amikor a .com-vállalkozások kudarcba fulladtak. De ha bizalmunkat a piaci erőkbe próbáljuk helyezni, akkor nem engedhetjük meg magunknak, hogy figyelmen kívül hagyjuk a világ legizgalmasabb párbeszédeit.

- Active X: 2 350 000 találat
- Python: 2 080 000 találat
- Gates: 3 020 000 találat
- KDE: 3 560 000 találat
- Gnome: 3 720 000 találat
- Perl: 7 650 000 találat
- Jesus (Jézus): 8 800 000 találat
- solution (megoldás): 13 300 000 találat
- market (piac): 31,200,000 találat
- Microsoft: 20 200 000 találat
- god (Isten): 24 300 000 találat
- sun: 25 500 000 találat
- sex (szex): 28 400 000 találat
- linux: 31 600 000 találat
- business (üzlet): 86 900 000 találat
- naked (meztelen): 8 080 000 találat.

Doc Searls (doc@ssc.com) a LinuxJournal szerkesztője és a Cluetrain Manifesto társszerzője. Nemsokára megjelenő könyvének címe „A valódi piacok: melyek azok és hogyan működnek".

A biztonságos kísérletezés kulcsa: freeVSD

A freeVSD használatával önellátó rendszerek készíthetők, továbbá sok fejfájástól kímélhetjük meg magunkat a programokkal kapcsolatban.

Környezetemben számos különleges képességgel felvértezett programfejlesztő és rendszergazda használ naponta Linuxot. Gyakran találkozunk új alkalmazásokkal, és néha sokat habozunk, telepítsük-e a programot akkor, ha nem teljesen bízunk meg benne. A tapasztalatunk ugyanis az, hogy némely gaz program könnyen tehet tönkre létfontosságú szolgáltatásokat egy élesben használt kiszolgálón. Szélsőséges esetben az is előfordulhat, hogy a rosszul felépített telepítő a munkaállomás operációs rendszerét is károsítja.
Többnyire a nem túl összetett webkiszolgálással kapcsolatos alkalmazások felélesztéséhez is szükség van a telepítésére: egy új felhasználó létrehozására (a SUID működéshez), a *httpd.conf* megváltoztatására, a webkiszolgáló újraindítására és a rendszergazda által birtokolt könyvtárak (például */etc* vagy */usr/local/*) fájljainak módosítására. Mindezeket – ha később úgy döntünk, hogy a rendszert mégsem akarjuk élesben használni – vissza kell tudnunk vonni. Bár az eltávolító (uninstall) parancsfájlok ebben a segítségünkre lehetnek, könnyen kudarcot is vallhatnak, és ilyenkor a rendszer meghatározhatatlan állapotba kerül.
A freeVSD olyan GPL-termék, amelyet arra találtak ki, hogy ISP-n keresztül virtuális kiszolgálókat nyújtson. Képes rá, hogy bármelyik RedHat-változatot hatékony, olcsó próbakörnyezetté változtassa. A freeVSD akár 250 teljes értékű egyedi kiszolgálót képes egy időben a rendelkezésünkre bocsátani. A rendszerfájlokra mutató közvetlen hivatkozások (Hard links) által minden virtuális kiszolgáló számára tömör, mégis egységes környezetet nyújt. A virtuális kiszolgálókra való beléptetést továbbra is a hagyományos chroot eszköz végzi, hatékonyan teremtve biztonságos „játszóteret".
Immár akár nemtörődöm módon is kísérletezgethetünk, tapasztalatlan kezdőknek adhatjuk át a kormánykereket, vagy rendszergazdai jogosultságot juttathatunk vadidegeneknek, miközben nem kell a kellemetlen következményektől tartanunk.
A rendszergazda szemszögéből nézve a freeVSD lehetővé teszi, hogy több önmagában működő rendszert készítsünk, amelyek saját karbantartói azonosítóval rendelkeznek; lehetőség nyílik továbbá a felhasználói azonosítók meghatározására, a webszolgáltatások, a levelezés és az adatbázis-kiszolgáló egyéni beállítására – akár Linux „Lite"-változatának megalkotására is, ha úgy akarjuk.
A freeVSD-t eredetileg három év alatt fejlesztették egy brit ISP számára. A levelezőlisták tanúsága szerint a freeVSD igencsak népszerű és jól támogatott: a kérdéseket vagy maguk a fejlesztők, vagy más felhasználók gyorsan megválaszolják.
Az összes virtuális kiszolgáló legtöbb lényeges szolgáltatását egy rendszergazdai szintű azonosítófélével, az Admin segítségével lehet állítani: ez felhasználókat vehet fel, megváltoztathatja jogosultságaikat, módosíthatja a *httpd.conf* fájlt, a kiszolgálót bizonyos szempontok szerint újraindíthatja és így tovább.

A freeVSD telepítése

A freeVSD telepítése egy kicsit trükkös. Különösen óvatosnak kell lennünk, ha a későbbiekben vissza szeretnénk állítani a rendszer eredeti állapotát. Mint mindig, nagyon fontos, hogy minden olyasféléről biztonsági másolatot készítsünk, ami kellemetlen lenne, ha valamilyen szerencsétlenség során elveszne. A weblap szerint a freeVSD nemsokára a Debian-, a Mandrake- és a Slackware-rendszereket is támogatni fogja, egyelőre azonban be kell érnünk a RedHat 6.x és 7.x. hivatalosan támogatott rendszereivel. Az 1.4.6-változattól kezdve a RedHat 7.0-t is támogatja, de szerintem a RedHat 6.2-hez kicsit jobban ki van dolgozva.
A freeVSD-t lehetőleg szinte teljesen „szűz" rendszerre telepítsük. Kezdjük egy frissen telepített RedHat 6.2-vel. Ezt követően el kell döntenünk, milyen különleges kiszolgálóprogramokat szeretnénk használni, például MySQL-t, PostgreSQL-t vagy PHP-t akarunk-e alkalmazni. Telepítsük a foltokat. Eszményi esetben minden alkalmazást a freeVSD beállítása előtt teszünk fel. Megjegyzendő, hogy a freeVSD a VMware alatt meglehetősen jól működik, ami az első telepítés során megtakaríthat nekünk némi fejfájást. A fájlrendszerváz elhelyezéséhez körülbelül 800 MB szabad merevlemez-területre lesz szükségünk.
Tételezzük fel, hogy az első virtuális gépünkhöz létezik (vagy szerezni tudunk) teljes értékű tartománynév vagy nyilvántartott IP-cím (a freeVSD IP-álneveket használ). Természetesen nem árt, ha a hálózatért felelős személytől engedélyt kérünk, mielőtt olyasféle tevékenységbe kezdünk, amelyet esetleg támadásként is értelmezhet.
Ezután az első virtuális gépnek válasszunk valamilyen nevet. Jó ötlet a gépnév (hostname) választása (például „myhost", ha a myhost.mydomain.com) vagy a tartománynév választása (mydomain), amennyiben több tartománynak is otthont adunk.
A freeVSD telepítésének lépései (amint a */usr/doc/freevsd-x.y.z/user-guide.txt* fájlban részletesen megtalálható) a következők:

1. A fő RPM telepítése (például *freevsd-1.4.6-2.i386.rpm*).
2. Az RPM pkgs telepítése (például *freevsd-pkgs-1.4.6-1.i386.rpm*).
3. A `/usr/sbin/vsd-install.pl` futtatása.
4. A `/usr/sbin/vsd-genskel.pl` futtatása (legyünk türelmesek, ezalatt ugyanis néhány száz megabájtnyi adat másolódik át). Ezt a folyamatot viszonylag egyszerű testreszabni. A */etc/freevsd.conf* fájl néhány testreszabási lehetőséget nyújt, ezekkel meghatározhatjuk, hogy milyen fájlok kerüljenek, illetve ne kerüljenek a készítés során a vázba. A RedHat 7.x-felhasználók esetében előfordulhat, hogy ehelyütt a */etc/xinetd.conf* fájlt, illetve az `xinetd` újraindítását is be kell állítaniuk.
5. Az első virtuális gép készítése a következő paranccsal történik: `/usr/sbin/vsdadm vs_create localhost ➥virtuÆlis-kiszolgÆl -neve virtuÆlis-➥kiszolgÆl -IP virtuÆlis-kiszolÆl -FQDN 200 0`.
6. A parancsfájl végrehajtása a `/usr/sbin/vsd-vsbatch.pl` futtatásával.
7. Virtuális kiszolgáló(k) indítása a `vsboot --start` paranccsal.

8. Próbáljuk ki a virtuális héjprogramot a `/usr/bin/bevs -r ➥[virtuÆlis nØv]` paranccsal (így egy virtuális héjprogramot kapunk).
9. A `passwd -u admin` utasítással állítsuk be a rendszergazda jelszavát.
10. Az `exit` paranccsal lépjünk ki a virtuális héjprogramból.

Ezen a ponton – feltételezve, hogy minden rendben zajlott – működő virtuális kiszolgálóval rendelkezünk, amelyre Telnettel vagy FTP-vel csatlakozhatunk.
Az eltávolításhoz a `/usr/sbin/vsboot -stop` utasítással az összes virtuális kiszolgálót állítsuk le. Majd ha szeretnénk, a létező virtuális gépeket a `/usr/sbin/vsdadm vs_delete ➥localhost myhost` paranccsal töröljük le.
Ezután futtassuk le a `/usr/sbin/vsd-uninstall.pl` parancsfájlt, hogy visszaállítsuk az eredeti beállításokat, és ha szükséges, töröljük a fájlokat. Ügyeljünk rá, hogy helyesen válaszoljunk a feltett kérdésekre, mivel nincs második lehetőségünk, és a beállításokat kézzel kell helyreállítani. Végül távolítsuk el a pkgs és a main RPM-eket.

Mi rejlik a köpeny alatt?

A `/usr/sbin/vsd-genskel.pl` telepítő parancsfájl a gép rendszerfájljait a vázként megadott könyvtárba másolja át. Ezt a folyamatot a */etc/freevsd.conf* bejegyzései szabályozzák, itt dől el, mely bejegyzések törlődnek vagy másolódnak át. A másolatok alapértelmezetten a */home/vsd/skel/* könyvtárba kerülnek:

```
$ ls /home/vsd/skel
bin  dev etc  home  lib  proc sbin  tmp  usr
```

Minden virtuális gép saját fájlrendszere az e vázra hivatkozó közvetlen hivatkozásokon alapul. A közvetlen hivatkozás egy adott fájlra vonatkozó második (vagy további) könyvtárbejegyzés. Tudnunk kell, hogy a közvetlen hivatkozások a közvetett hivatkozásoktól abban különböznek, hogy az összes közvetlen hivatkozást törölni kell, mielőtt a fájl a fájlrendszerből valóban törlődne; ellenben ha a közvetett hivatkozás célfájlját eltávolítjuk, a közvetetten hivatkozott fájl eredetije megmarad. Mivel minden kiszolgáló a fájlok egyazon másolatán osztozik, az alapértelmezett fájlrendszeren a közvetlen hivatkozások alkalmazása óriási lemezterület-megtakarítást jelenthet.
Ha `ps`-sel vagy `top`-pal nézzük, úgy tűnik, hogy a freeVSD felhasználói folyamatok rendszergazdai jogosultsággal futnak; noha valójában nem ezzel a jogosultsággal kezdenek. Minden virtuális gép esetén alapértelmezés szerint a UIDS-ek 1000-rel kezdődnek és kétszázasával növekednek (például az első vm 1000-től kezdődik, a következő 1200-tól stb.). Ezt a beállítást a */etc/vsd.conf* fájlban lehet megváltoztatni.
Mint korábban említést nyert, az otthont adó gép IP-álnév segítségével feltételezhetően több IP-címet használ. A démon a `/usr/sbin/virtuald` és a `inetd` (avagy `xinetd`, amely az `inetd`-t helyettesíti RedHat 7.0 alatt) segítségével fogja el a szolgáltatásokat (például Telnetet vagy FTP-t) célzó ügyfélkapcsolatokat. A bejövő kapcsolat a virtuális környezetben a megfelelő démonhoz továbbítódik – a gazdagép chroot eszközét és alapértelmezetten a */home/vsd/vs/* könyvtárat használva.
A lehetséges biztonsági hibák miatt a virtuális webkiszolgáló nem közvetlenül a 80-as kapun fut. Ehelyett a `vsredirect` nevű gazdakiszolgáló folyamat a 80-as kapu forgalmát a 8080-as kapura irányítja át, és a 443-as HTTPS-kapu forgalmát átmozgatja a 8443-as kapura. A *security.txt* ismerteti, miként szerezhet a rossz szándékú felhasználó e biztonsági intézkedés nélkül rendszergazdai jogosultságot. Az átirányítás minden 1000 alatti

kiváltságos kapu esetében ajánlott.
A freeVSD-fájlrendszerben néhány megszokott parancsot, például az `rm`-et, az `ls`-t és a `passwd`-t kissé megváltoztatták, hogy az Admin-azonosítónak a virtuális kiszolgálón található felhasználói azonosítók felügyeletéhez szükséges jogosultságai meglegyenek. Ide értendő az új felhasználók felvétele, illetve a fájljaik kezelése is.
Külön démonfolyamatok (HTTPD, Pro-ftpd, Sendmail és így tovább) készülnek minden egyes virtuális kiszolgálóhoz.
A SUID parancsfájl a virtuális gép Adminja számára lehetővé teszi, hogy a `/usr/sbin/rebootvs` paranccsal démonokat indítson, illetve állítson le.

Admin-tapasztalatok

Az Admin-azonosító színlelt rendszergazdai jogosultságokkal rendelkezik, ezzel lehetővé teszi az Admin-felhasználó számára, hogy különféle felügyeleti feladatokat hajtson végre anélkül, hogy veszélyeztetné a rendszert. Az Admin azonban nem gyengített rendszergazdai azonosító, hanem sokkal inkább megerősített felhasználói azonosító, korlátozott fájl- és azono-

sítókezelési képességeket biztosítva az adott virtuális gépen. Bejelentkezés után a `whoami` jelentése szerint az admin és egyben a saját könyvtárunk a */root*. A / könyvtár vizsgálatával kideríthetjük, hogy az Admin birtokolja a */root*, */home* és */tmp* könyvtárakat. A többi Admin által birtokolt fájlt a `find / -name admin -print` segítségével kérdezhetjük le.
Most valóban Linuxot futtatunk és `bash`-héjprogramot használunk, nem pedig valami „felvizezett” dologgal állunk szemben. Futtathatunk Pythont vagy Perlt, `gcc`-vel pedig akár új alkalmazásokat is fordíthatunk. Hagyományos Linux-héjkörnyezetbe kerültünk, ami olyan, mintha a saját rendszerünk lenne.
A */home/httpd/docs* könyvtárban találhatjuk a webkiszolgálóhoz tartozó *DocumentRoot* könyvtárat, a naplófájlokat pedig a */home/web/log* tartalmazza. A httpd.conf fájl a */etc/httpd/conf/* alatt található és az Admin módosíthatja. A `/usr/sbin/rebootvs` tulajdonképpen a virtuális kiszolgálót indítja újra azáltal, hogy a hozzá tartozó folyamatokat indítja.
A */etc/passwd* fájlban a szokásos rendszerazonosítókat és az admin-azonosítót találjuk; ez a fájl azonban csak olvasható. Ugyanakkor a `/usr/sbin/useradd <œj_felhasznÆl >` az elvárásoknak megfelelően működik, beleértve az új felhasználónak az /etc/passwd fájlba történő felvételét.
A */etc/vsd/priv* fájl formátuma hasonló a */etc/groups* fájléhoz. Meghatározza, hogy mely felhasználóknak van joga a login, Telnet és FTP eléréséhez, illetve futtathatnak-e Perlt vagy éppen gcc-t. A `/usr/bin/listrights` parancs megmutatja az adott felhasználó jogait. A `/usr/sbin/setrights` parancs pedig eme fájl kezelésére szolgál; bár a forráskódot átböngészve a *setrights.c* nem igazán tűnik olyan eszköznek, amellyel a bejelentkezési jogosultságot meg lehetne változtatni. A */etc/vsd/priv* fájlt kézzel átszerkesztve természetesen ezt a jogosultságot is meg lehet adni; s mivel az Adminnak a */etc/*-ben nincs írási joga, a bejelentkezési jogosultságot csak a rendszergazda adhatja meg.

freeVSD-felügyelet a gazdáról

A freeVSD démon a Sysvinit indítási előírásokat követi és a következő parancsokkal szabályozható:

```
/etc/rc.d/init.d/vsd start
/etc/rc.d/init.d/vsd stop
/etc/rc.d/init.d/vsd restart
/etc/rc.d/init.d/vsd status
```

(bár ez nem mindig hoz pontos eredményt).
Néhány beállítást a */etc/vsd.conf* fájlon keresztül végezhetünk el.

Egy adott virtuális kiszolgálót `/usr/sbin/vsboot` paranccsal indíthatunk el vagy éleszthetünk fel.
A `/usr/sbin/vsdadm` parancs, amelyet még a telepítési folyamatnál ismertettünk, virtuális gépek készítésére vagy törlésére használható. Ez az utasítás úgy működik, hogy a 1725-ös kapun üldögélő `vsd` démonnak ad parancsokat. Az elrendezés lehetővé teszi, hogy a freeVSD-t különféle felületeken keresztül kezeljük, ideértve a felügyeleti készletet is, amely az Idaya Ltd. által kifejlesztett eszközöket tartalmazza – ezek némelyike webböngészőn vagy Microsoft Windows alatt fut.
A `/usr/bin/bevs` program („become a virtual server”, azaz változz virtuális kiszolgálóvá) a gazdagép felhasználója számára lehetővé teszi, hogy a virtuális gépen a rendszergazda szerepébe kerüljön anélkül, hogy telnetezne vagy más módon kapcsolódna hozzá a virtuális géphez. Leginkább figyelemre méltó képessége azonban az Admin jelszavának alapbeállítása, vagy a nem az Admin vagy egy virtuális felhasználó által birtokolt fájlok kezelése.
A következő parancsfájlok, bár telepítéskor hasznosak, a későbbiekben veszélyt jelenthetnek. Esetleg kiadhatunk rájuk egy `chmod a-x` parancsot, nehogy véletlenül végrehajtsuk őket:

```
/usr/sbin/vsd-genskel.pl
/usr/sbin/vsd-install.pl
```

A `/usr/sbin/vsd-refreshskel.pl` program frissíti a freeVSD-vázat és a főfájlrendszer fájljait minden virtuális gépnél újrafűzi. Ezáltal válik lehetővé, hogy töröljük vagy frissítsük a virtuális kiszolgálók számára is elérhető alkalmazásokat.

Mindenképpen olvassuk el a leírást vagy nézzük meg a forráskódot, mielőtt ezt a parancsot kipróbálnánk. Azokat a csomagokat, amelyeket a virtuális gépeken elérhetővé szeretnénk tenni, de nem akarjuk őket feltenni a gazdagépre, a következő paranccsal telepíthetjük:

```
rpm -ivh --force --root=/home/vsd/skel/ fÆjl.rpm
```

Hasonlóképpen a

```
rpm -ivh --force --root=/home/vsd/vs/some-vs
fÆjl.rpm
```

parancsot használhatjuk, ha csak egyetlen virtuális gépre telepítünk RPM-csomagot. Mivel a váz nagy része sajnos nem az RPM telepítéséből, hanem fájlok másolásával keletkezett, az RPM-adatbázis nem pontosan fogja tükrözni a telepített csomagokat, ezért van szükség a force (kényszerítés) kapcsolóra. Mivel a `/usr/sbin/vsd-refreshskel.pl` frissíti a vázat, arra is használhatjuk, hogy minden gépen frissítsük a rendszerprogramokat. Megjegyezzük, hogy úgy tűnik, ez a szolgáltatás RedHat 7.x alatt a freeVSD 1.4.6 változatával hibásan működik.
A leírást a */usr/share/doc/freevsd-1.4.6/* könyvtár tartalmazza, amely alapértelmezés szerint minden virtuális gép számára elérhető. Végül a */usr/share/freevsd/* a testreszabáshoz tartalmaz hasznos parancsfájlokat.

Félhivatalos biztonsági elemzés és vita

Veszélyes dolog a felhasználókat rendszergazdai jogosultságokkal felruházni. Gyakran láthatjuk a rendszergazdai jogosultságok héjprogramokon keresztüli rossz szándékú kihasználását, ezért meglepő lenne, ha ezekkel a gondokkal a freeVSD által nyújtott chroot-környezetben nem találkoznánk.
Másfelől egyetlen rendszerfolt egy időben akár 250 webhely biztonságát növeli meg. Ráadásul a HTTPD-folyamatok használata számos olyan gondot megelőzhet, amelyek egy jellemzően héjprogramon alapuló környezetben gyakoriak, ahol az egyik felhasználó műveletei a másik szolgáltatásait akár véletlenül is megszakíthatják.
A gazdagép felhasználója az összes virtuális gép összes felhasználójának futó folyamatait láthatja, ráadásul a `bevs --r` parancs segítségével alapértelmezés szerint bármely virtuális kiszolgálón rendszergazdai jogosultságot szerezhetnek! Mivel a `bevs` parancs SUID-jogosultságú, úgy tűnik, ez szándékosan van így. Javaslom, változtassuk meg a jogosultságot, és kapcsoljuk ki ezt a lehetőséget (például a `chmod o-xr /usr/bin/bevs` megfelel erre a célra), miközben a rendszergazdacsoport tagjainak továbbra is joguk van futtatni a `bevs` programot.
A virtuális kiszolgáló felügyeleti démonja, a `/usr/sbin/vsd` a 1725-ös kapura figyel. Emiatt a virtuális kiszolgáló felhasználólistáját meglehetősen egyszerű dolog kívülről lekérdezni, a felhasználók jogait megváltoztatni, tehát a `vsd` démon tulajdonképpen minden azonosítás nélkül távolról is irányítható. Amint a *security.txt fájl* is javasolja, nagyon fontos, hogy ezt a gondot a gazdagépen kezeljük, a következőkhöz hasonló módon (valahová az *rc.local* fájlba helyezve):

```
ipchains -A input -p tcp -s W.X.Y.Z
➥--dport 1725 -j ACCEPT
ipchains -A input -p tcp -s 0/0
➥--dport 1725 -j REJECT
```

Az egyre szaporodó operációsrendszer-frissítések miatt fontos hogy üzembiztos terjesztést válasszunk.

Más lehetőségek: alkalmazáspróba freeVSD nélkül

Az új alkalmazások kipróbálásának talán legegyszerűbb módja, ha van egy tartalékgépünk, amelyen a merevlemezt nyugodtan letörölhetjük és a rendszert újratelepíthetjük.
A másik lehetőség lemezlenyomatok (disk images) készítése. Miután az operációs rendszert telepítettük és az ízlésünknek megfelelően be is állítottuk, a `dd if=/dev/hda1 ➥of=/tmp/image` parancs segítségével egy másolatfájlt készíthetünk, amit a későbbiek során az operációs rendszer visszaállítására használhatunk fel. Ha ezt a stratégiát választjuk, figyeljünk a 2 GB-os fájlméretmegkötésre, ebben az esetben jól jöhet a `split` eszköz.
A VMware nevű üzleti alkalmazás teljes egészében szimulálni képes egy x86-os PC-t – egészen a teljes értékű Phoenix BIOS-ig. A legnagyobb teljesítmény érdekében a VMware a legtöbb utasítás végrehajtására a CPU emulálása helyett közvetlenül a gazdagép processzorát használja. A VMware-alapú megközelítés azonban lemezkezelés szempontjából nem kifejezetten hatékony: minden egyes munkafolyamathoz a teljes operációs rendszert újra kell telepíteni. A VMware virtuális lemezeket képes készíteni fájlokból, így nem szükséges több meghajtó létrehozása. A nagy lemezhasználat mellett minden futó virtuális rendszerhez memóriát is foglalni kell. Igaz viszont, hogy a VMware esetében nem vagyunk egyetlen Linux-terjesztéshez kötve, sőt még a GNU/Linuxhoz sem. Ez a rendszer teljesebb virtuális kiszolgáló környezetet nyújt, de erőforrásigényesebb is, ideértve az összes operációs rendszer egyenkénti felügyeletét is. Ide kívánkozik, hogy jelenleg folyik a VMware ingyenes változatának, a Plex86-nak a fejlesztése (➲ http://www.plex86.org/).
Egy másik lehetőség lehet a User Mode Linux. Röviden a Linux-rendszermagot átültették egy másik géptípusra – a folyamatok egy hagyományos Linux-rendszermag belsejében futnak. A teljes leírás a (➲ http://user-mode-linux.sourceforge.net/ címen érhető el.

Összegzés

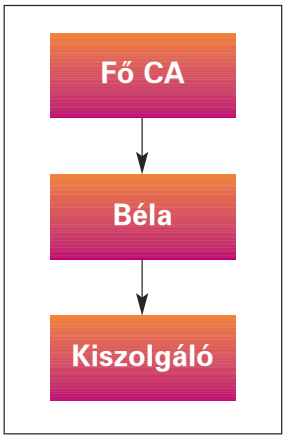
A freeVSD ígéretes projekt. A minden virtuális kiszolgálón több kiszolgáló folyamat futtatásának módszere igen nagyméretű és meglehetősen hatékony megoldást nyújt. A kiszolgáló látszata a virtuális gépeken igen meggyőző, de a virtuális fájlrendszer feletti teljes irányítás hiánya kiábrándító lehet. Ráadásul a lehetőségeket behatárolja, hogy az Admin-felhasználó nem képes RPM-csomagokat telepíteni. Egy átlagos fejlesztéshez, különösen, ahol legfontosabb eszközök már telepítve vannak és együttműködnek a freeVSD-vel, ez a leggyorsabb módja, hogy új Linux-rendszerhez jussunk. Annak esélye, hogy a gazdagépben a virtuális kiszolgálóból véletlenül kárt teszünk, elenyésző.

Randall Embry (randall@embry.com) a feleségével, négyéves kislányával és két macskával él együtt Bloomingtonban, Indiana államban. A programozáson és Linux élvezetén kívül Randall egy csapat programozót és hálózati mérnököt irányít a Dataworksnél, amely a Fine Light IT tanácsadó részlege. További érdekességekért lásd ➲ http://www.embry.com/randall/ címet.

Bevezetés az OpenSSL programozásába

Égető szükséged van egy egyszerű webkiszolgáló-ügyfél párra? Most megtudhatod, miért neked való az OpenSSL.

Egy TCP-n alapuló hálózati alkalmazást legegyszerűbben és leggyorsabban az SSL használatával tehetünk biztonságossá. Ha C nyelven dolgozol, talán a legjobb választás az OpenSSL (➲ http://www.openssl.org). Az OpenSSL ***Eric Young*** SSL/TLS-alapú SSLeay csomagjának szabad forrású változata, amely BSD stílusú felhasználói szerződés hatálya alá esik. Az OpenSSL-hez tartozó leírás és a példaprogramok sajnos sok kívánnivalót hagynak maguk után. A súgóoldalak – ahol vannak – meglehetősen jók, de a mélyebb összefüggéseket gyakran figyelmen kívül hagyják, mivel ezek elsősorban nem tankönyvnek, hanem vonatkozó kézikönyvnek íródtak. Az OpenSSL API hatalmas és bonyolult, ebben a cikkben nem is kíséreljük meg a maga teljességében bemutatni. A célunk inkább az, hogy a súgóoldalak alapján megtanítsunk hatékonyan dolgozni. Ebben a cikkben, amely egy kétrészes sorozat első fele, felépítünk egy egyszerű webes ügyfelet és kiszolgálót, amelyek az OpenSSL alapvető tulajdonságait használják ki. A második cikkben a fejlettebb sajátosságokat mutatjuk majd be, többek között a kapcsolat újrafelvételét és az ügyfél azonosítását.



Kiterjedt tanúsítványlánc

Felteszem, tisztában vagy az SSL és a HTTP főbb fogalmaival. Ha nem, az ismerkedést érdemes az RFC-kkel kezdeni (lásd a ***Kapcsolódó címeket***).

Terjedelmi okok miatt a forráskódokból csak szemelvényeket közlünk. A teljes forráskód a szerző weblapjáról tölthető le (➲ http://www.rtfm.com/openssl-examples).

Programok

Ügyfélprogramunk egyszerű HTTPS-ügyfél (lásd RFC 2818). SSL-kapcsolatot kezdeményez a kiszolgálóhoz, és a kapcsolaton keresztül HTTP-kérést küld át. Ezután várja a kiszolgáló válaszát, majd azt kiírja a képernyőre. Ez a fetch- és a cURL-programok tevékenységének nagymértékben leegyszerűsített változata.

A kiszolgálóprogram egyszerű HTTPS-kiszolgáló, amely az ügyfelek által kezdeményezett TCP-kapcsolatot várja. Amikor egy kérés beérkezik, SSL-kapcsolatot épít fel. Felépülte után beolvassa az ügyfél HTTP-kérését, majd HTTP-választ küld neki, ezt követően pedig lebontja a kapcsolatot.

Első feladatunk a környezeti objektum beállítása (`SSL_CTX`). Ez a környezeti objektum használható a későbbiekben új kapcsolati objektumok létrehozására minden egyes SSL-kapcsolathoz. A kapcsolati objektumok pedig az SSL kézfogási, olvasási és írási műveleteihez nyújtanak segítséget.

Ennek a megközelítésnek két előnye van. Először is a környezeti objektum használatával sok szerkezetnek csak egyszer kell kezdőértéket adnunk, ami teljesítménynövelő hatású. A legtöbb alkalmazásban minden SSL-kapcsolat ugyanazokat a kulcsadatokat, tanúsítványhitelesítő (CA) listákat stb. fogja használni. Ahelyett, hogy ezeket az adatokat minden kapcsolatra újra betöltenénk, a program indulásakor egyszerűen a környezeti objektumba töltjük be őket. Amikor új kapcsolatot szeretnénk létrehozni, csak a környezeti objektumra kell hivatkoznunk.

A környezeti objektum használatának második előnye, hogy több SSL-kapcsolat között teszi lehetővé az adatok megosztását, úgy, mint az SSL-kapcsolatgyorstár a kapcsolat folytatásához. A környezet előkészítése négy elsődleges feladatból áll, mindegyiket az `initialize_ctx()` függvény végzi el az *1. listán* (32. oldal) látható módon.

Mielőtt az OpenSSL-t bármire is használhatnánk, a programkönyvtárat elő kell készíteni. Ezt a feladatot végzi el az SSL_library_init(), amely betölti az OpenSSL által használt algoritmusokat. Ha részletes hibajelentéseket szeretnénk, az `SSL_load_error_strings()` függvénnyel a hibaüzeneteket is be kell töltenünk, máskülönben az OpenSSL hibakódjait nem tudnánk a hibaüzenetekre leképezni.

A hibakiíráshoz is létrehozunk egy objektumot. Az OpenSSL a bemenet és kimenet kezeléséhez egy elvonatkoztatott BIO nevű objektumot használ, amivel a programozó számára lehetővé válik, hogy különféle I/O-csatornákat (foglalatok, terminál, memória stb.) ugyanazokkal a függvényekkel kezeljen, mindössze a megfelelő BIO objektumot kell választania. Ebben az esetben egy `stderr`-hez csatolt BIO objektumot hozunk létre, amelyet a hibák kiírására használunk.

Ha olyan ügyfelet vagy kiszolgálót írsz, amely képes az ügyfél hitelesítésére, be kell töltened a saját nyilvános, illetve titkos kulcspárodat és a hozzá kapcsolódó tanúsítványt. A tanúsítvány titkosítatlanul tárolódik és az `SSL_CTX_use_certificate_chain_file()` függvény tölti be a CA tanúsítványokkal együtt – létrehozva a tanúsítványláncot. A titkos kulcs az `SSL_CTX_use_PrivateKey_file()` függvénnyel tölthető be. Biztonsági okokból a titkos kulcsot sokszor jelszóval védik. Ha ez a helyzet, a vezérlés a jelszóbekérő visszahívó függvényre adódik (az `SSL_CTX_set_default_passwd_cb()`-t kell beállítani).

Amennyiben a gépet, amelyhez kapcsolódsz, hitelesíteni fogod, az OpenSSL-nek tudnia kell, mely tanúsítványhitelesítőkben bízol meg. A CA-k az `SSL_CTX_load_verify_locations()` hívással tölthetők be.

Az erős biztonság érdekében az SSL-nek jó minőségű véletlen számokra van szüksége. Általában az alkalmazás feladata a véletlenszám-előállító kezdőértékét megadni. Az OpenSSL azonban – amennyiben lehetséges – a ***/dev/urandom*** eszközt használja erre a célra. A /dev/urandom a szabványos Linux-rendszer része, ezért ez ügyben semmit nem kell tennünk, ami nagyon kényelmes, mert a véletlen számok gyűjtése bonyolult és könnyen elrontható feladat. Ne feledd, ha nem Linuxot használsz, egy bizonyos ponton hibaüzenetet kaphatsz, mert a véletlenszám-előállító nem kapott kezdőértéket. Az OpenSSL rand(3) súgóoldala többet mond erről.

1. lista initialize_ctx()

```
SSL_CTX *initialize_ctx(keyfile,password)
  char *keyfile;
  char *password;
  {
    SSL_METHOD *meth;
    SSL_CTX *ctx;

    if(!bio_err){
      /* A rendszer elıkØsz tØse */
      SSL_library_init();
      SSL_load_error_strings();

      /* A hibaki r  k rnyezet */

    bio_err=BIO_new_fp(stderr,BIO_NOCLOSE);
    }

    /* A SIGPIPE kezelı beÆll tÆsa */
    signal(SIGPIPE,sigpipe_handle);

    /* A k rnyezet nk lØtrehozÆsa*/
    meth=SSLv23_method();
    ctx=SSL_CTX_new(meth);

    /* A kulcsaink Øs tanœs tvÆnyaink
    bet ltØse */

if(!(SSL_CTX_use_certificate_chain_file(ctx,
      ➥keyfile)))
      berr_exit ("A tanœs tvÆnyfÆjl nem
                  ➥olvashat ");

    pass=password;
    SSL_CTX_set_default_passwd_cb(ctx,
      ➥password_cb);
    if(!(SSL_CTX_use_PrivateKey_file(ctx,
      keyfile,SSL_FILETYPE_PEM)))
      berr_exit("A kulcsfÆjl nem
                ➥olvashat ");

    /* A megb zhat  CA-k bet ltØse*/
    if(!(SSL_CTX_load_verify_locations(ctx,
       ➥CA_LIST,0)))
       berr_exit("A CA-k listÆja nem
                  ➥olvashat ");
#if (OPENSSL_VERSION_NUMBER < 0x0090600fL)
    ➥SSL_CTX_set_verify_depth(ctx,1);
#endif

    return ctx;
  }
```

Az ügyfél

Miután az ügyfél előkészítette az SSL-környezetet, készen áll a kiszolgálóhoz történő kapcsolódáshoz. Az OpenSSL megköveteli tőlünk, hogy a TCP-kapcsolatot magunk hozzuk létre az ügyfél és a kiszolgáló között, majd a TCP-foglalatot használjuk SSL-foglalat létrehozására. A kényelem kedvéért a TCP-kapcsolat létrehozását az elkülönített `tcp_connect()` függvény végzi (amely itt nem látható, de a letölthető forrásban benne van).

Miután a TCP-kapcsolat létrejött, létrehozzuk a kapcsolatot kezelő SSL-objektumot, amelyet a foglalathoz kell csatolni. Nem közvetlenül csatoljuk hozzá, hanem létrehozunk egy BIO objektumot, amely a foglalatot használja, tehát az SSL-objektumot a BIO-hoz csatoljuk.

Ez az elvonatkoztatási réteg lehetővé teszi, hogy a foglalatokon kívül más csatornákon keresztül is használhasd az OpenSSL-t, feltéve, hogy akad megfelelő BIO-d. Például az OpenSSL próbaprogramjainak egyike az SSL-ügyfelet és -kiszolgálót a memórián keresztül kapcsolja össze. Sokkal gyakorlatiasabb alkalmazás lehet olyan protokollok támogatása, amelyek nem érhetők el foglalatokon keresztül, így például SSL-t a soros vonalon keresztül futtathatsz.

Az SSL-kapcsolat első lépése az SSL-kézfogás végrehajtása. A kézfogás hitelesíti a kiszolgálót (esetleg az ügyfelet is), és meghatározza a későbbi adatforgalom védelmét biztosító kulcsokat. Az `SSL_connect()` hívás hajtja végre az SSL-kézfogást. Mivel tömbös foglalatokat használunk, az `SSL_connect()` nem tér vissza, amíg a kézfogás nem fejeződik be vagy hiba nem történik. Az `SSL_connect()` 1-et ad vissza siker, és 0-t vagy negatív számot hiba esetén. A hívás így néz ki:

```
/* Kapcsol dÆs a TCP- foglalathoz */
sock=tcp_connect(host,port);

/* Kapcsol dÆs az SSL- foglalathoz */
ssl=SSL_new(ctx);
sbio=BIO_new_socket(sock,BIO_NOCLOSE);
SSL_set_bio(ssl,sbio,sbio);
if(SSL_connect(ssl)<=0)
  berr_exit("SSL kapcsol dÆsi hiba");
if(require_server_auth)
  ➥check_cert(ssl,host);
```

Amikor a kiszolgálóhoz SSL-kapcsolatot kezdeményezünk, ellenőriznünk kell a kiszolgáló tanúsítványláncát. Az OpenSSL elvégzi az ellenőrzések egy részét, de a másik rész sajnos alkalmazásfüggő, ezért azokat nekünk magunknak kell elvégeznünk. Példaprogramunk fő tevékenysége a kiszolgáló „személyazonosságának” ellenőrzése – ezt a *2. listán* olvasható `check_cert` függvény végzi.

Miután meggyőződtél róla, hogy a kiszolgáló tanúsítványlánca érvényes, ellenőrizned kell, hogy az éppen vizsgált tanúsítvány arra a kiszolgálóra vonatkozik-e, amelyik felmutatta. A legtöbb esetben ez azt jelenti, hogy a kiszolgáló DNS neve megjelenik a tanúsítványban, vagy a ***Jogosult neve*** (Subject Name) mező ***Általános név*** (Common Name) részében, vagy a tanúsítvány kiterjesztésében. Bár minden egyes protokoll a kiszolgáló azonosságának ellenőrzését kissé eltérő módon oldja meg, az RFC 2818 tartalmazza az SSL/TLS-en keresztül megvalósított HTTP szabályait. Ha nincs okod más megoldás alkalmazására, kövesd az RFC 2818-at.

Mivel a legtöbb tanúsítvány a tartománynevet még mindig a ***Közönséges*** név mezőben tartja, és nem a kiterjesztésben, csak a Közönséges név ellenőrzését mutatjuk be. Az `SSL_get_peer_certificate()` segítségével egyszerűen megszerezzük a kiszolgáló tanúsítványát és összehasonlítjuk a közönséges nevet azzal a gépnévvel, amelyhez csatlakozunk. Ha a két név nem egyezik, akkor valami nincs rendben, és kilépünk.

A 0.9.5-változat előtt az OpenSSL ki volt téve a tanúsítvány-kiterjesztéses támadásnak. Ennek megértéséhez vegyük az

2. lista check_cert() függvény

```
void check_cert(ssl,host)
  SSL *ssl;
  char *host;
  {
    X509 *peer;
    char peer_CN[256];


if(SSL_get_verify_result(ssl)!=X509_V_OK)
      ➥berr_exit("A tanœs tvÆny nem
                  ➥ellenırizhetı");

    /* Ellenırizz k a tanœs tvÆnylÆncot!
       A lÆnc
       hosszœsÆgÆt  nmßk dıen ellenırzi az
       OpenSSL, ha beÆll tjuk az ellenırzØsi
       mØlysØget a k rnyezetben. */

    /*Az Æltalænos nØv ellenırzØse */
    peer=SSL_get_peer_certificate(ssl);
    X509_NAME_get_text_by_NID
      ➥(X509_get_subject_name(peer),
      ➥NID_commonName, peer_CN, 256);
    if(strcasecmp(peer_CN,host))
    err_exit
      ("Az Æltalænos nØv nem egyezik a
        ➥gØpnØvvel. ");
  }
```

ábrán látható esetet, ahol a kiszolgáló hitelesítését Béla írta alá. Béla nincs a CA-id között, de az ő tanúsítványát aláírta egy olyan CA, amiben megbízol.

Ha elfogadod ezt a tanúsítványt, nagy bajba kerülhetsz. Az a tény, hogy a CA aláírta Béla tanúsítványát, azt jelenti, hogy a CA elhiszi Béláról, hogy ő az, akinek állítja magát, de nem jelenti azt, hogy Béla megbízható. Ha Bélával akarsz üzletelni, ez rendben van, viszont ez nem sokat ér, ha Aladárral szeretnéd ugyanezt megtenni, és Béla (akiről sosem hallottál) áll jót érte. Eredetileg ez ellen a támadás ellen az egyetlen védekezési mód az volt, hogy megkötötték a tanúsítványláncok hosszát, így tudható volt, hogy a vizsgált tanúsítványt a CA írta alá. Az X.509 3. változata lehetővé teszi, hogy a CA bizonyos tanúsítványokat úgy címkézzen meg, mint más CA-k. Így egy CA-nak egyetlen gyökere lehet, amely azután egy csomó al-CA-nak nyújthat tanúsítványt. Az OpenSSL újabb változatai (a 0.9.5-nél frissebbek) figyelik ezeket a kiterjesztéseket, ezért akár ellenőrzöd a lánc hosszát, akár nem, védve vagy e támadás ellen. A 0.9.5 előtti változatok egyáltalán nem ellenőrzik a kiterjesztéseket, ezért – amennyiben ilyen régi változatot használsz – neked kell a lánc hosszának az ellenőrzését elvégezned. A 0.9.5-nek gondjai akadhatnak az ellenőrzéssel, ha tehát ezt a változatot használod, fontold meg a frissítést. Az `#if` utáni sor régebbi változat használata esetén az `initialize_ctx()` kódjában ellenőrzi a lánc hosszát. Az `SSL_CTX_set_verify_depth()` függvényt használjuk a lánchossz ellenőrzésének kikényszerítésére. Mindent összevetve nagyon ajánlatos a 0.9.6 változatra frissíteni, különösen azért, mert a hosszú (de szabályosan felépített) láncok egyre népszerűbbek. A legújabb és legjobb OpenSSL-változat jelenleg a 0.9.66.

A 3. listán (23. CD Magazin/OpenSSL könyvtár) látható kód segítségével írunk HTTP-kérést. A bemutató kedvéért egy többé-kevésbé bedrótozott HTTP-kérést használunk, amely a `REQUEST_TEMPLATE` változóban van. Mivel a gép, amelyhez csatlakozunk, változhat, ki kell töltenünk a ***Gép*** (Host) fejlécet, amit az `snprintf()` végez el. Ezután az `SSL_write()` elküldi az adatokat a kiszolgálónak. Az `SSL_write` API-ja nagyjából ugyanaz, mint a `write()`, kivéve hogy SSL-objektumot adunk át és nem fájlleírót.

A tapasztalt TCP-programozók észrevehették, hogy ha a visszatérési érték nem egyenlő a kiírni próbált értékkel, a kiírás körbejárása helyett hibát jelzünk. Tömbös módban az `SSL_write()` minden vagy semmi elven működik, a hívás nem tér vissza, amíg az adatot ki nem írta vagy hiba nem történt, míg a `write()` esetleg csak az adatok egy részét írja ki. Az `SSL_MODE_ENABLE_PARTIAL_WRITE` kapcsoló (itt nem használjuk) engedélyezi a részleges írást, ebben az esetben szükség van a ciklusra.

A régi stílusú HTTP/1.0 használatánál a kiszolgáló átküldi a választ és lezárja a kapcsolatot. A későbbi változatoknál bevezették az állandó kapcsolatot, amelynél több, sorban egymást követő tranzakció egy kapcsolatot használ. Az egyszerűség és a kényelem kedvéért mi nem fogunk ilyet használni. Elhagyjuk az ezeket engedélyező fejlécet, ennek hatására a kiszolgáló a válasz végén lezárja a kapcsolatot, ami gyakorlatilag azt jelenti, hogy a fájl végéig folyamatosan kell olvasni, és ez nagyban leegyszerűsíti a dolgokat.

Az `OpenSSL` az adatok beolvasására az `SSL_read()` API-hívást használja, ahogyan a 4. lista (23. CD Magazin/OpenSSL könyvtár) mutatja. Akárcsak a `read()`-nél, egyszerűen választunk egy megfelelő méretű tárolót és átadjuk az `SSL_read()`-nek. A tároló mérete nem túl fontos. Az `SSL_read()` a `read()`-hez hasonlóan az összes elérhető adatot visszaadja, akkor is, ha az kevesebb, mint a kért mennyiség. Egyébként, ha nincs elérhető adat, az olvasóhívás útja elzáródik.

A BUFSIZZ választása befolyásolja a hatékonyságot, de nem olyan módon, mint amikor egyszerűen egy foglalatból olvasunk. Abban az esetben minden `read()` híváskor át kell váltani a rendszermagba. Az átváltás drága művelet, ezért a programozók nagy tárolókat szoktak választani, hogy csökkentsék a szükséges átváltások számát. Amikor azonban az SSL-t használjuk, a `read()`-hívások száma és így az átváltásoké is inkább a kiírt adatrekordok számától függ, semmint az `SSL_read()`-hívások számától.

Ha például az ügyfél kiír egy 1000 bájtos rekordot, és mi 1 bájtonként hívjuk meg az `SSL_read()` függvényt, akkor az első `SSL_read()`-hívás alkalmával az egész rekord beolvasódik, és a többi hívás csak kiolvassa az értékeket az SSL-tárolóból. Ezért a tároló mérete, ellentétben a szokásos foglalatokkal, SSL használata esetén kevésbé lényeges. Ha az adatok sok kis rekord sorozataként íródnak ki, egyszerre egyetlen `read()`-hívással beolvashatod őket. Ilyenkor az OpenSSL `SSL_CTRL_SET_READ_AHEAD` kapcsolója használható.

Ne feledd el kiértékelni az `SSL_get_error()` visszatérési értékét! A szokásos foglalatoknál minden negatív szám (általában -1) hibát jelez, és ekkor az `errno` értékét megvizsgálva megtudhatjuk, mi történt. Természetesen az `errno` itt nem használható, mert csak a rendszerhibákat mutatja, és minket az SSL-hibák érdekelnek. Az `errno` alkalmazása nagy körültekintést is igényel ahhoz, hogy többszálú környezetben is biztonságos legyen.

Az `errno` helyett az OpenSSL az `SSL_get_error()` hívást biztosítja. Ez a hívás lehetővé teszi a visszatérési érték vizsgá-

latát és az esetleges hiba mibenlétének megismerését. Ha a visszatérési érték pozitív, akkor adatokat olvastunk be, amelyeket egyszerűen kiírunk a képernyőre. Egy valódi ügyfél természetesen értelmezné a HTTP-választ, és vagy megjelenítené az adatokat (például egy weboldalt), vagy lemezre mentené. Az OpenSSL szempontjából azonban teljesen mindegy, hogy mi lesz az adatok sorsa, ezért nem foglalkozunk vele. Ha a visszatérési érték nulla, nem jelenti azt, hogy nincs elérhető adat – ez esetben az utunk el lett volna zárva, ahogyan fent említettük. Ez inkább azt tükrözi, hogy a foglalat zárva van és olvasásra soha semmilyen adat nem lesz elérhető. Így kilépünk a ciklusból.

Ha a visszatérési érték negatív, valamilyen hiba történt. Kétféle hibatípussal kell számolnunk: közönséges hibákkal és idő előtti lezárásokkal. A hiba típusát az `SSL_get_error()` hívással állapítjuk meg. A hibakezelés ügyfelünkben nagyon kezdetleges, a legtöbb hibát a `berr_exit()` meghívásával csak kiírjuk, amely ezután kilép a programból. Az idő előtti lezárásokat külön kell kezelnünk.

A TCP a FIN-csomagrészt használja annak jelzésére, hogy a küldő minden adatot elküldött. Az SSL 2. változata az SSL-kapcsolat lezárására mindkét félnek megengedte a TCP FIN küldését, ami úgynevezett csonkolásos támadásra adott lehetőséget. A támadó elhitethette, hogy az üzenet rövidebb a valóságosnál, egyszerűen a TCP FIN-t meghamisítva. Ha az áldozat más módon nem tudhatta meg az üzenet várható hosszát, könnyen azt hihette, hogy a hossz rendben volt.

Ennek a biztonsági kockázatnak a kivédésére vezették be az SSLv3-ban a ***close_notify*** figyelmeztetést, ami egy SSL-üzenet (emiatt biztonságos), de nem része magának az adatfolyamnak, ezért az alkalmazás nem látja. A ***close_notify*** elküldése után semmilyen adatot nem szabad átvinni. Ezért amikor az `SSL_read()` nullával tér vissza, amely azt jelzi, hogy a foglalatot lezárták, ez valójában azt jelenti, hogy a ***close_notify*** figyelmeztetés megérkezett. Ha az ügyfél FIN-t kap a ***close_notify*** előtt, az `SSL_read()` hibával tér vissza. Ezt a jelenséget nevezik idő előtti lezárásnak. Az ügyfél minden idő előtti lezárásnál dönthet úgy, hogy kiírja a hibát és kilép. Ezt a viselkedést vonja maga után az SSLv3-szabvány. Az idő előtti lezárás sajnos elég gyakori hiba, különösen az ügyfeleknél. Ezért hacsak nem akarsz állandóan hibákat jelenteni, gyakran figyelmen kívül kell hagynod őket – kódunk külön figyel erre a hibára: jelenti az idő előtti lezárást az `stderr`-en, de nem lép ki.

Ha hiba nélkül olvastuk el a választ, a kiszolgálónak egy ***close_notify*** figyelmeztetést kell küldenünk. Ezt végzi el az `SSL_shutdown()` API hívás. A kiszolgálónál részletesebben fogjuk tárgyalni az `SSL_shutdown()`-t, de az alapötlet egyszerű: a visszatérési érték 1, ha a leállítás teljes, 0, ha a leállítás nem teljes, és -1, ha hiba történt. Mivel már megkaptuk a kiszolgáló ***close_notify*** üzenetét, az egyetlen dolog, amivel gond lehet, ha nem tudjuk elküldeni a saját ***close_notify*** üzenetünket. Egyébként az `SSL_shutdown()` sikeres lesz (1-et ad vissza).

Végül meg kell semmisítenünk a különféle objektumokat, amelyeknek helyet foglaltunk. Mivel a program kilépésre készül, az objektumok felszabadítása nem lenne feltétlenül szükséges, de egy általánosabb programban szükséges volna.

A kiszolgáló

Webkiszolgálónk néhány különbségtől eltekintve az ügyfél tükörképe. Először a `fork()` használatával elágaztatjuk a programot, hogy a kiszolgáló több ügyfelet is kiszolgálhasson. Másodszor: az OpenSSL BIO API-ját használjuk az ügyfél kérésének sorról sorra történő beolvasásához és a válasz kiírásához. Végül a kiszolgáló lezárási folyamata bonyolultabb.

A több ügyfelet kezelni képes kiszolgáló írásának Linux alatt az a legegyszerűbb módja, hogy minden kapcsolódó ügyfélhez egy új kiszolgáló folyamatot indítunk. Ezt a `fork()` hívás segítségével tesszük meg, miután az `accept()` visszatért. Minden egyes új folyamat függetlenül fut, és egyszerűen kilép, ha végzett az ügyfél kiszolgálásával. Bár nagy forgalmú webkiszolgálóknál a megközelítés nagyon lassú lehet, esetünkben teljesen elfogadható.

A kiszolgáló fő fogadóciklusát az 5. listán (23. CD Magazin/OpenSSL könyvtár) láthatjuk.

Az elágazás és az SSL-objektum létrehozása után a kiszolgáló meghívja az `SSL_accept()` függvényt, amely a kézfogás kiszolgálóoldali részéért felelős. Akárcsak az `SSL_connect()`, a tömbös foglalatok használata miatt az `SSL_accept()` is lezárja a csatornát, amíg a kézfogás be nem fejeződik. Ezért az `SSL_accept()` csak akkor tér vissza, ha a kézfogás befejeződött vagy hiba lépett fel. Az `SSL_accept()` 1-et ad vissza siker esetén, és 0-t vagy negatív számot, ha hiba történt.

Az OpenSSL BIO objektumai bizonyos mértékig egymásra halmozhatók. Ezért az SSL-objektumot becsomagolhatjuk egy BIO-ba (az `ssl_bio` objektumba), és ezt a BIO-t csomagoljuk tovább egy tárazott BIO objektumba a következő módon:

```
io=BIO_new(BIO_f_buffer());
ssl_bio=BIO_new(BIO_f_ssl());
BIO_set_ssl(ssl_bio,ssl,BIO_CLOSE);
BIO_push(io,ssl_bio);
```

Ez lehetővé teszi a számunkra, hogy tárazott olvasási és írási műveleteket hajtsunk végre az SSL-kapcsolaton keresztül a BIO_* függvények és az új `io` objektum segítségével. E ponton felvetődhet a kérdés, hogy mire jó mindez.

Elsősorban a programozás lesz kényelmesebb: a programozó számára lehetővé válik, hogy az SSL-rekordok helyett természetes egységekkel (sorokkal és karakterekkel) dolgozzon.

A kérés

A HTTP-kérés egy kéréssorból, az ezt követő fejlécsorokból és az esetleges törzsből áll. A fejlécsorok végét üres sor jelzi (azaz egy CRLF pár, bár a hibás ügyfelek ehelyett néha egy LF karaktert küldenek). A kéréssort és a fejlécet a legkényelmesebb soronként beolvasni, amíg üres sorral nem találkozunk. Az `OpenSSL_BIO_gets()` hívás segítségével ez megtehető, lásd a 6. listát (23. CD Magazin/OpenSSL könyvtár). A `BIO_gets()` hívás hasonlóan működik, mint az `sdtio fgets()` hívása. Fog egy tetszőleges nagyságú tárolót és egy hosszúságot, és az SSL-ről beolvas egy sort a tárolóba. Az eredmény mindig `NULL` karakterre végződik (de benne van a végső `LF`). Ezért az adatokat egyszerűen addig olvassuk be soronként, amíg olyan sorral nem találkozunk, amely csak `LF`-et vagy `CRLF`-et tartalmaz.

Mivel állandó méretű tárolót használunk, előfordulhat, bár nem valószínű, hogy túl hosszú sort kapunk. Ebben az esetben a hosszú sor kettébomlik. Abban a nagyon valószínűtlen esetben, ha a szétbontás közvetlenül a `CRLF` előtt történik, a következő beolvasott sor az előző sorból származó `CRLF`-et fogja tartalmazni. Ez esetben azt fogjuk hinni, hogy a fejléc idő előtt véget ért. Az igazi webkiszolgálók erre az esetre is figyelnek, de itt nem éri meg nekünk a fáradságot. Jegyezzük meg, hogy bármilyen nagy a bejövő sor hossza, nem léphet fel tártúlcsordulás. A legrosszabb esetben félreértelmezzük a fejlécet. Igazából semmit nem kezdünk a HTTP-kéréssel, egyszerűen

beolvassuk, utána elfelejtjük. Az igazi alkalmazások elolvasnák a kéréssort és a fejlécet, megnéznék, hogy van-e törzs, és azt is elolvasnák.
A következő lépés a HTTP-válasz kiírása és a kapcsolat lezárása:

```
if((r=BIO_puts
  ➥(io,"HTTP/1.0 200 OK\\r\\n"))<0)
  err_exit("´rÆshiba");
if((r=BIO_puts
  ➥(io,"Server: EKRServer\\r\\n\\r\\n"))<0)
  err_exit("´rÆshiba");
if((r=BIO_puts
  ➥(io,"KiszolgÆl  tesztoldala\\r\\n"))<0)
  err_exit("´rÆshiba");

if((r=BIO_flush(io))<0)
  err_exit("Hiba a BIO ki r tØse k zben");
```

Vegyük észre, hogy `BIO_puts()` hívást használtunk az `SSL_write()` helyett, ami lehetővé teszi, hogy soronként írjuk ki a választ, de az egész válasz egyetlen SSL-rekordba kerüljön. Ez az SSL-rekord összeállításának költségei miatt fontos. A sértetlenség ellenőrzése és a titkosítás jelentős erőforrásokat köt le, emiatt jó ötletnek látszik olyan nagy rekorddal dolgozni, amilyennel csak lehet.
Érdemes megemlíteni a fent említett tárazott kiírás néhány finomságát. Először is lezárás előtt ki kell ürítened a tárat. Az SSL-objektumnak fogalma sincs arról, hogy egy BIO-t rétegeztél fölé, így ha az SSL-kapcsolatot megsemmisíted, az adatfolyam utolsó darabja ott marad a tárban. A `BIO_flush()` megoldja a gondot. Alapértelmezés szerint az OpenSSL a BIO-khoz 1024 bájtos átmeneti tárakat használ. Mivel az SSL-rekordok akár 16 KB hosszúak is lehetnek, az 1024 bájtos átmeneti tárak használata nagymértékű töredezettséghez (és emiatt teljesítménycsökkenéshez) vezethet. A `BIO_ctrl()` API-hívást használhatod a tár méretének növelésére.
A válasz átvitele után el kell küldenünk a ***close_notify*** üzenetet. Akárcsak az előbb, most is az `SSL_shutdown()` segítségével végezzük el a feladatot. A helyzet sajnos egy kissé bonyolultabb, amikor a kiszolgáló zárja le előbb a kapcsolatot. Az első `SSL_shutdown()` hívás elküldi a ***close_notify***-t, de azt nem keresi a másik oldalon, ezért azonnal visszatér, de 0 értékkel, ami azt jelenti, hogy a lezárási folyamat nem ért még véget. Az alkalmazásnak kell ismét az `SSL-shutdown()`-t meghívnia. Kétféleképpen járhatunk el: mondhatjuk azt, hogy megkaptuk a teljes HTTP-kérést, amely bennünket érdekel. Semmi mással nem törődünk. Emiatt az sem érdekel minket, hogy az ügyfél küld-e ***close_notify***-t vagy sem. A másik lehetőség, hogy a protokoll előírásait szigorúan vesszük, és mástól is ezt várjuk el, ezért megköveteljük a ***close_notify***-t.
Az első hozzáállás könnyű életet biztosít. Meghívjuk az `SSL_shutdown()`-t, elküldjük a ***close_notify***-t és azonnal kilépünk, függetlenül az ügyfél viselkedésétől. Ha a második utat követjük (ahogy példánkban a kiszolgáló), az élet sokkal bonyolultabb, mert az ügyfelek gyakran helytelenül viselkednek. Az első gond, amellyel szembe kell néznünk, hogy az ügyfelek sokszor egyáltalán nem küldenek ***close_notify***-t; egyes ügyfelek (például az IE bizonyos változatai) a HTTP-válasz kézhezvétele után azonnal lezárják a kapcsolatot. Amikor elküldjük a ***close_notify***-t, a túloldal egy TCP RST-csomagrészt küldhet. Ebben az esetben a program egy SIGPIPE jelzést kap. Védekezésképpen az `initialize_ctx()` függvényben beállítunk egy egyszerű `SIGPIPE`-ot kezelő programrészt.

A második gond, hogy az ügyfél a ***close_notify***-t válaszul a mi ***close_notify***-unkra esetleg nem azonnal küldi. A Netscape bizonyos változatai azt várják, hogy először te küldj egy TCP FIN-t. Ezért hívjuk meg a `shutdown(s,1)`-et a második `SSL_shutdown()`-hívás előtt. Ha a `shutdown()`-t az 1 módértékkel hívjuk meg, elküldi a FIN-t, de a foglalatot nyitva hagyja olvasásra. A kiszolgáló leállításának kódja a 7. listán (23. CD Magazin/OpenSSL könyvtár) olvasható.

Ami kimaradt

Ebben a cikkben csak az OpenSSL felszínét karcolhattuk. Következzék a további lehetséges témák (nem teljes) felsorolása. A kiszolgáló tanúsítványainak és gépnevének összevetésére sokkal kifinomultabb módszer az X.509 subjectAltName kiterjesztés használata. Az összehasonlításhoz ezt a kiterjesztést ki kell szedni a tanúsítványból, és össze kell vetni a gépnévvel. Továbbá jó lenne a gépneveket a tanúsítványokban szereplő helyettesítő karakterekkel megadott nevekhez hasonlítani. Figyeljük meg, hogy ezek az alkalmazások hiba esetén egyszerűen kilépnek. Az igazi alkalmazások természetesen képesek felismerni a hibát és kilépés helyett jelezni a felhasználónak, vagy a naplóba írni.
A következő cikkben számos fejlett OpenSSL-sajátosságot tárgyalunk, többek között a kapcsolat folytatását, a többutas és nem tömbös I/O- és ügyfélhitelesítést.

Köszönetnyilvánítás

Köszönettel tartozom a következő személyeknek, amiért segítettek az OpenSSL-ben és lektorálták a cikkemet: ***Lisa Dusseault***, ***Steve Henson***, ***Lutz Jaenicke*** és ***Ben Laurie***.

Eric Rescorla
1993 óta foglalkozik az Internet biztonságával. Ő a szerzője az SSL and TLS: Designing and Building Secure Systems (Addison-Wesley, 2001) című könyvnek.

Kapcsolódó címek

A cikk egyes részeit a könyvemből vettem át (SSL and TLS: Designing and Building Secure Systems, copyright Addison-Wesley, 2001, ISBN 0-201-615980-3). A könyv részletesen tárgyalja az SSL protokollt és alkalmazásait. További adatokért lásd a ➲ http://www.rtfm.com/sslbook weblapot.

Az itt bemutatott programok forráskódja a szerző weblapjáról letölthető: ➲ http://www.rtfm.com/openssl-examples. A forráskódra BSD stílusú felhasználói szerződés vonatkozik.

Az OpenSSL a ➲ http://www.openssl.org-ról tölthető, ahol a leírás is elérhető.

Az SSLv2 és az SSLv3 leírása megtekinthető a ➲ http://www.netscape.com/eng/security/SSL_2.html és a home.netscape.com/eng/ssl3/index.html weblapokon. A TLS (RFC 2246) és a HTTPS (RFC 2818) leírása a ➲ http://www.ietf.org/rfc/rfc2246.txt és a ➲ http://www.ietf.org/rfc/rfc2818.txt címeken lelhető fel.

Programból megvalósított RAID

A Linux adta lehetőségekkel
és olcsó alkatrészekkel is nagyfokú adatbiztonságot érhetünk el.

A RAID a Redundant Array of Inexpensive Disks kifejezés rövidítése, amelyet magyarul nagyjából ***olcsó lemezek hibatűrő tömbje***-ként adhatnánk vissza. Az eljárás lényege: az adatokat több lemezen, szétszórtan helyezzük el, ezáltal próbálva meg teljesítménynövekedésre vagy nagyobb biztonságra szert tenni. Ezt több gyártó SCSI-vezérlőkártyája is biztosítja, de ugyanezt a rendszermag támogatásával programból is megtehetjük – így az adattöbbszörözést akár két IDE-merevlemezzel megvalósíthatjuk. Külön öröm, hogy a Linux programszintű RAID-vezérlőjének készítője, ***Ingo Molnar*** magyar származású (mingo@chiara.csoma.elte.hu). A RAID a fájlrendszerréteg alatt helyezkedik el, semmi köze sincs hozzá, ezért bármilyen fájlrendszerrel használni tudjuk.

IDE vagy SCSI?

Többen is vannak, akik rendíthetetlenül állítják, hogy Linuxot csakis SCSI-s géppel lehet futtatni, IDE-merevlemezt tenni linuxos kiszolgálóba főbenjáró bűn. Ez természetesen nem igaz. Az IDE kevés lemez esetén gyakran gyorsabb, mint a SCSI, viszont mondjuk egy nyolc lemezes kiszolgáló IDE-sínnel nem éppen bizalomgerjesztő. A túl hosszú IDE-kábelek majdnem biztos, hogy hibás adatokat eredményeznek. Az IDE másik gondja, hogy a szolgalemez elérése nagyon lassúvá válhat, ami pedig RAID esetén, ahol gyakori ugyanazon adat több lemezre írása, nagy teljesítménycsökkenést idézhet elő. Ezért ha úgy döntesz, hogy IDE-s merevlemezekkel oldod meg a RAID-et, feltétlenül csak elsődleges merevlemezeket használj.

Mire van szükség a RAID-hez?

Legalább két merevlemezre. A programból megvalósított RAID egy lemezzel is létrehozható, amennyiben több lemezterületet használsz. Ez azonban ismét teljesítménycsökkenéshez vezet, hiszen ugyanazt az adatot nem két lemez írja párhuzamosan, hanem egy lemez, csak két lemezterületre.
2.4-es rendszermagra. A 2.0-s és 2.2-es sorozatban is létezik RAID-támogatás, de kevesebb lehetőséget biztosít, és a fejlesztője szerint nem annyira megbízható, mint a 2.4-esben szereplő. Ha valamiért mégis ragaszkodsz régebbi rendszermagodhoz (én nem tenném, mert az új leírhatatlanul gyorsabb), és az újabb RAID előnyeit is élvezni szeretnéd, kénytelen leszel letölteni egy foltot, és újrafordítani a rendszermagot. A legfrissebb mag jelenleg a 2.4.14-es. Továbbá szükséged lesz a raidtools csomagra. Ezt a legtöbb terjesztés tartalmazza, ha mégsem találnád, töltsd le az Internetről. A rendszermagot az ➲ http://ftp.kernel.org/pub/linux címről szerezheted be, a raidtools ugyanitt, a ***daemons/raid/alpha*** alkönyvtárban található.

Néhány magyarázatra szoruló fogalom

Egy tömbben raid- és tartaléklemezeket (spare disk) különböztetünk meg. A tartaléklemezeket a rendes működés során valójában sosem használjuk, csak akkor lépnek működésbe, ha az egyik raidlemez kiesik. Ekkor indul el az összehangolás, hogy a tömb az új lemez teljes értékű tagja lehessen. Ezalatt már lehet használni, viszont ha az összehangolás közben még egy raidlemez leáll, a RAID-megvalósítások egyike sem biztosítja az adatok sértetlenségét. A RAID-et természetesen tartaléklemez nélkül is meg lehet oldani.
A tömbbel kapcsolatos tudnivalók egy beállítási állományban találhatók. A tömb használatba vételéhez minden rendszerindítás után fel kell éleszteni, ezt pedig csak a beállítási állomány alapján lehet megtenni. Ez az állomány azonban nem mindig érhető el. Gondoljunk arra, ha a saját fájlrendszerünkből akarunk RAID-et készíteni, hogyan fog elindulni a rendszer? Erre való a ***persistent superblock***. Ha bekapcsoljuk, minden hibatűrő lemezterület elején egy szuperblokk található, amelyből a rendszermag megtudhatja a tömbbel kapcsolatos adatokat, ehhez tehát nincs szükség a beállítási állományra. Ha azonban tükrözött lemezről akarjuk a rendszert betölteni, mindenképpen szükségünk lesz rá! Mindenkinek ajánlom, hogy a persistent superblock mellett még egy beállítási fájlt is tartson meg, ugyanis bármikor szükség lehet rá.

Hogyan fogjak hozzá?

Először ellenőrizd, hogy a RAID-támogatást modulként fordítottad-e. Ha igen, mindenekelőtt a következő parancsot add ki:

```
modprobe md
```

Ezzel betöltötted a modult. Ha eddig mindent jól csináltál, proc fájlrendszeredben egy új állományt fogsz látni: ***/proc/mdstat***. Ez az a fájl, amely az egyetlen segítséged a RAID állapotának megfigyeléséhez. Minden lépés után nézd meg, hogyan változott.

```
cat /proc/mdstat
```

Ha most kiadod, ezt kell látnod:

```
Personalities:
read_ahead not set
unused devices: <none>
```

Ezután el kell döntened, hogy milyen RAID-megvalósítást akarsz kivitelezni. Ennek megfelelően kell szerkesztened a ***/etc/raidtab*** állományt, ami nem más, mint a fentebb már említett beállítási fájl. Az alábbi egy egyszerű tükrözést állít be:

```
# RAID-0 (mirroring) beÆll tÆs
# ezen az eszk znØven Ørj k el majd a t mb t
raiddev /dev/md0
# raid-megval s tÆs: linear, 0, 1, 4 vagy 5
raid-level                1
# raidlemezek szÆma
nr-raid-disks             2
# tartalØklemezek szÆma
nr-spare-disks            1
# (kB); minden megval s tÆsnÆl mÆst jelent,
# lÆsd kØsıbb
chunk-size                4
# persistent superblock: 0 a nem, 1 az igen
persistent-superblock     1
# a t mb elemei; itt kell megadni a raid- Øs
# a tartalØk-lemezter leteket
device                    /dev/hda1
```

```
raid-disk                       0
device                          /dev/hdc1
raid-disk                       1
device                          /dev/hde1
spare-disk                      0
```

Ezekután lássuk az egyes RAID-megvalósításokat!

Linear mode

A segítségével több lemezt egyként kezelhetünk. Ha az első lemez betelt, az írás a következőn onnan folytatódik, ahol az előzőn abbamaradt. A lemezeknek nem kell azonos méretűeknek lenniük. A tömb mérete az elemek méretének az összege. Ebben az esetben nincs adattöbbszörözés, egy lemez kiesésével is elveszítheted minden adatodat. A chunk size-nak itt nincs sok értelme, viszont nem lehet elhagyni. Legalább két raidlemezt igényel, tartaléklemezek nincsenek.

RAID-0 (stripe)

A csíkozás hasonló a linear mode-hoz, csak annyi a különbség, hogy a lemezek párhuzamosan töltődnek fel. A lemezek nem feltétlenül azonos méretűek, a tömb mérete az összes lemez együttes mérete. Nincs adattöbbszörözés. A chunk-size határozza meg, hogy egy adatsorból mekkora „falat" kerülhet egy lemezre. Abban az esetben, ha két lemezből álló RAID-0-s tömböd van, amelynél a chunk size 4 KB, egy 8 KB-os állomány mentésekor annak egyik fele az első lemezre, a másik pedig a másodikra íródik. Az írási folyamat ilyen módon közel feleannyi időt vesz igénybe. Az olvasás hasonlóan zajlik. Láthatod, mennyire fontos, hogy egy RAID-tömb eleme ne egy szolgameghajtó legyen! Legalább két raidlemez szükséges a kiépítéshez, tartaléklemezek nincsenek.

RAID-1 (mirroring)

A tükrözés az első megvalósítás, amely valós hibatűrést jelent. A lemezek között egyszerű tükrözés történik és minden raidlemez tartalma megegyezik. Így ha egy kivételével az összes lemez kiesett, az adatok még mindig sértetlenek maradnak. A tömb mérete megegyezik a legkisebb lemezrész méretével. A chunk-size íráskor semmilyen szerepet nem játszik, hiszen az adatot az összes lemezre fel kell írni, olvasáskor viszont azt határozza meg, hogy egy lemezről mennyi adatot kell sorban beolvasni – így az olvasás párhuzamosan történhet. Legalább két lemezrész szükséges, a tartaléklemezek pedig elhagyhatók.

RAID-4

Nagyon ritka megvalósítás. Az adatok a raidlemezeken csíkozva helyezkednek el, mint a RAID-0-nál, és egy kiemelt lemez van, amely a párosságadatot (parity) tárolja. A tömb méretét úgy számolhatjuk, ha a párosságadatot tároló lemezt nem számítva az összes raidlemez számát a legkisebb méretével szorozzuk meg. Lemez elvesztésekor a párosságadatok alapján a helyreállítás egy tartaléklemezre indul meg. A chunk size a párosságadatot tároló lemezen a párosságblokkok mérete. A kiépítéshez legalább három lemezrészre van szükség, és legalább egy tartaléklemez rendszerbe állítása ajánlott.

RAID-5

A RAID egyik legnépszerűbb fajtája. Hasonló a RAID-4-hez, csak a párosságadatok nem egy lemezen tárolódnak, hanem az összes lemezen szétszórva. A tömb méretét ugyanúgy számoljuk, mint a RAID-4-nél. A chunk size ugyanazt jelenti, mint a „kistestvérénél". Használatához legkevesebb három lemezrész szükséges, valamint legalább egy tartaléklemez ajánlott.

RAID-10 (mirrored stripe)

A tükrözött csíkozánem beépített megoldás, hanem két RAID-0-s tömb RAID-1-es tömbje. Ritkán használt kiépítés.

Ha a fentiek mérlegelése után elkészültél a beállítási állománnyal, először ellenőrizd, hogy a fájlban szereplő eszköz létezik-e. Az én SuSE 7.2-esem például 4 MD-eszközt tartalmaz (***/dev/md0***, ***/dev/md1***, ***/dev/md2*** és ***/dev/md3***). Szinte soha nincs többre szükség, amennyiben mégis, az új RAID-eszközt az alábbi paranccsal hozhatod létre:

```
mknod /dev/md4 b 9 4
# a 4-es szám változhat; lásd még
man mknod
```

Ezután létrehozzuk vagy felélesztjük a tömböt:

```
mkraid /dev/md0
```

RAID-1-nél ilyenkor rögtön megindul az összehangolódás. Ha kiíratod a ***/proc/mdstat*** fájlt, a folyamatot megfigyelheted:

```
Personalities : [raid1]
read_ahead 1024 sectors
md0 : active raid1 hdc1[1] hda1[0]
      292224 blocks [2/2] [UU]
      [=========>...........]  resync = 45.8%
➥(134272/292224) finish=2.6min speed=990K/sec
unused devices: <none>
```

Bármi is történjék, ha az `mkraid` nem szakad meg, a tömb már működik is. A fenti példában az eszközt már formázhatod, a folyamat végét sem kell megvárni:

```
mke2fs /dev/md0
```

Ezekután erre az eszközre is úgy hivatkozhatsz, mint bármi másra. Ha a RAID-et ki szeretnéd kapcsolni, add ki a következő parancsot:

```
raidstop /dev/md0
```

Ha RAID-ről szeretnéd indítani, a tömböt persistent superblockkal kell létrehoznod. A RAID elemeinek lemezterület-típusa 0xFD legyen, így a rendszermag a tömböt már induláskor önműködően felismeri; arra viszont ügyelj, hogy a LILO hivatalos változata nem támogatja a RAID-ről történő rendszerbetöltést. Vannak terjesztések (például a RedHat), amelyek különböző foltokkal próbálnak segíteni ezen, én viszont azt ajánlom, inkább készíts egy külön ***/boot*** lemezterületet. Szükségtelen nagyra „növeszteni", és az sem fontos, hogy „biztonságban" legyen.

Csak a kezemet figyeljétek…

Felmerülhet az igény, hogy a csereterületet (swap) csíkozd. Az adattöbbszörözésnek ebben az esetben nincs túl sok értelme, viszont előnyös lenne a virtuális memória gyorsabb elérése. A csereterület nagyon egyszerűen RAID-0-zható, még a ***raidtools*** csomag sem kell hozzá, a rendszermagban ugyanis hosszú ideje létezik hozzá támogatás. Ha a ***/etc/fstab*** valahogy így néz ki:

```
# fontos: a csereter letek azonos
/dev/hda2   swap   swap   defaults,pri=1   0   0
# elsıbbsøgßek
/dev/hdc2   swap   swap   defaults,pri=1   0   0
```

akkor a rendszermag önműködően elvégzi a csíkozást, anélkül, hogy ezt külön mondani kellene neki. Ezzel jelentős teljesítménynövekedésre tehetünk szert.

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 Mro□ek. Leginkább a számítógépes hálózatok biztonsága érdekli.

Pehelysúlyú könyvtárelérési protokoll

Azonosítsuk Linux-rendszerünk felhasználóit a könnyen alakítható LDAP segítségével.

Az Interneten könnyűszerrel bukkanhatunk olyan nagy adatmennyiségre, amely gyakorta igen különböző könyvtárszerkezetbe van rendezve. A könyvtárak szabványosítására tett első próbálkozás az X.500 volt, amely a címlistát az X.400 levelezőrendszerhez tette elérhetővé. Ebből a könyvtárból az adatokat a DAP (Directory Access Protocol) protokoll segítségével lehetett kinyerni, ez azonban meglehetősen nehézsúlyú volt.

A későbbiekben számos más könyvtárat is kifejlesztettek, és ezek – bár néha meglehetősen különbözőek voltak – többnyire az X.500 szabványban gyökereztek. Kijött azonban egy új adatelérési protokoll is: a pehelysúlyú könyvtárelérési protokoll (Lightweight Directory Access Protocol – LDAP), amelyet csaknem az összes jelenlegi számítógépfajtán fel lehet használni szinte bármely X.500-zal együttműködő könyvtár adatainak az eléréséhez.

Eme pehelysúlyú könyvtárelérési protokoll használatát mutatjuk be ebben a cikkben. Elsőként azt vizsgáljuk, miképpen lehet olyan könyvtárszerkezetet összeállítani, amelyből az adatokat LDAP segítségével lehet elérni, majd rátérünk a protokoll Linux alatti használatának ismertetésére. Végezetül azzal foglalkozom, hogy miként alkalmazhatjuk az LDAP-könyvtárat akár rendszerünk felhasználóinak azonosítására is, ez ugyanis a NIS (Network Information Service) mellett érdekes választásnak ígérkezik.

A szakszókincs használata

Az LDAP elsődleges célja az X.500-megfelelő könyvtárszerkezetek adatainak az elérése. Ez a könyvtár tulajdonképpen rendszerbe szervezett adatbázis, amelyben különböző típusú adatok érhetők el. Gyakran használják nevek és címek tárolására, de ugyanilyen kényelmesen lehet benne a hálózati erőforrások adatait vagy bármi mást, például a linuxos gépek felhasználóinak azonosítására szolgáló adatokat is tárolni. A könyvtárrendszert a tárolóobjektumok alkotják, amelyeket gyakran könyvtárelemeknek is neveznek (Directory Components – DC). Ezeket a DC-ket akár a DNS-rendszerek tartományaival is össze lehet kapcsolni. Az LDAP-tároló akár DNS-tartományokhoz is láncolható; és ha a cégünk már rendelkezik bejegyzett DNS-tartománynévvel – például ***azlan.com*** –, az LDAP szervezési egységeket akár nevekhez is rendelhetjük, valahogy így:

```
ou=training, dc=azlan, dc=com.
```

A tárolóobjektumok tartalmazzák a levélobjektumokat, amelyeket gyakran egyszerűen csak bejegyzésnek hívnak, mert valóban csak azok az LDAP-adatbázisban. Egy ilyen levélre példa a felhasználó és a hozzárendelt levélcím, illetve minden más adat, amelynek segítségével az adott felhasználót azonosíthatjuk a gépünkön. Minden egyes ilyen bejegyzésnek egyedi neve van, amelyet megkülönböztetett névnek (Distinguished Name – DN) nevezünk. Például a bejegyzett tartománynévvel rendelkező ***Azlan*** részlegnél dolgozó ***Paul*** nevű felhasználó a következő megkülönböztetett nevet kapja `cn=Paul`, `ou=training`, `dc=azlan`, `dc=com`. Emellett a bejegyzésnek egy általános neve (Common Name – CN) is létezik, amely az objektum tárolójában egyedi azonosító – ez például a vezetéknév lehet.

Ezen objektumok tulajdonságai adják meg az objektumhoz rendelt adathalmazt; a felhasználói objektum tulajdonsága lehet például a cím és a jelszó. Ha a Linux-felhasználókat az LDAP segítségével szeretnénk azonosítani, nagyon fontos, hogy ezeknek a tulajdonságoknak pontos értékeket adjunk, például a */etc/passwd* fájl ***UID*** mezőjének az értékeit, amennyiben a Linux-erőforrásokat el szeretnénk érni. A bejegyzések pontos meghatározásai, vagyis az, hogy hol helyezkedjenek el a könyvtárban, és milyen tulajdonságok csatlakozzanak hozzájuk, a sémában jelenik meg. Linux alatt a bejegyzések a ***slapd.oc.conf***, a tulajdonságok pedig a ***slapd.at.conf*** fájlban találhatók.

Az LDAP-adatbázisok általános adatlekérdező nyelve az LDAP adatcsere-formátum (LDIF). Amennyiben ezt szeretnénk használni, nagyon fontos, hogy az egyes bejegyzések kötelezően kitöltendő mezőit ne hagyjuk üresen; ellenkező esetben objektummeghatározáskor kellemetlen hibaüzeneteket kaphatunk. A kötelező tulajdonságokat a ***spad.oc.conf*** tartalmazza.

OpenLDAP

A Linux alatt talán a legtöbbet használt LDAP-változat az ***OpenLDAP*** (➲ http://www.openldap.org). LDAP-megfelelő üzleti könyvtárak is beszerezhetők, például a ***Novell eDirectory*** vagy a ***Netscape's directory***. Linuxon az OpenLDAP telepítése után (amely gyakran az alapértelmezett kiszolgálótelepítés része) néhány fájl másolódik a rendszerre. Mielőtt azonban belemerülnénk a tényleges beállítások taglalásába, vessünk egy gyors pillantást a felmásolt fájlokra.

Az LDAP-telepítés legfontosabb állománya egy önálló LDAP démon: a `slapd`. Ezt kell elindítanunk az LDAP-rendszer használatba vételéhez. Ha a hálózaton egynél több LDAP-kiszolgálót használunk, és az adatokat mindkét rendszeren meg szeretnénk ismételni, a `slurpd`-re is szükségünk lesz, ugyanis ez másolja le az adatokat a LDAP-főkiszolgálóról egy vagy több LDAP-alkiszolgálóra.

Az LDAP-kiszolgáló beállításához természetesen néhány beállításfájlt át kell szerkesztenünk. A legtöbb ilyen fájl a */etc/openldap* könyvtárban található, azonban nem árt, ha odafigyelünk, mert ugyanazok a fájlok néha más könyvtárakban is jelen lehetnek, például a */etc*-ben, és ez megnehezíti a helyes beállítást. Ha a

/etc/openldap fájljai egynél több helyen vannak jelen, én többnyire csak a /etc/openldap könyvtár fájljait hagyom meg, az összes többi helyre csak hivatkozásokat teszek.
A legfontosabb beállításfájl a ***slapd.conf***. A `slapd` szinte összes tulajdonságát itt állíthatjuk be, ezt fájlt azonban még a `slapd` futtatása előtt át kell szerkesztenünk. A slapd.conf mellett található még két másik fájl is, amelyek a sémát tartalmazzák: a slapd.oc.conf és a slapd.at.conf. A legtöbb esetben ezekkel nem kell foglalkoznunk, hagyjuk érintetlenül őket úgy, ahogy vannak. Előfordulhat viszont, hogy a hálózat működtetéséhez mégis át kell szerkesztenünk őket. A legutolsó beállításfájl az ***ldap.conf*** – ez egy kicsi, de annál fontosabb állomány, amelyet az LDAP-ügyfél használ annak a kiszolgálónak az azonosítására, amelyikről az adatokat le kell töltenie.
A beállításfájlokon kívül néhány parancs is található itt, amelyekkel adatokat gyűjthetünk a könyvtárunkba, illetve meggyőződhetünk róla, hogy van-e benne egyáltalán valami. Az `ldapadd` segítségével adatokat adhatunk a könyvtárhoz, az `ldapmodify` a már létező bejegyzések tulajdonságait módosítja, végül az `ldapsearch` használatával egy adott bejegyzést kereshetünk meg. Az LDIF-fájlok révén a fenti néhány és pár további parancs segítségével kezelni tudjuk könyvtárunk adatait.
Végezetül azokról a fájlokról beszélünk, amelyekre akkor lesz szükségünk, ha a rendszerünkkel LDAP-kiszolgálóhoz szeretnénk csatlakozni. Ezek más programcsomagok részei, így nem biztos, hogy a rendszeren telepítve vannak. A névkiszolgálás-kapcsoló által használt modul neve ***nss_ldap***. A ***/etc/nsswitch.conf*** fájlban kell megadnunk, hogy az adatokat ezentúl a LDAP-könyvtárból vegye és ne a ***/etc/passwd***-fájlból. A másik fontos csomag a ***pam_ldap***. Ez az a Linux alatti modul, amelynek segítségével a felhasználók az LDAP-adatbázisban beilleszthető azonosító modulokkal (Pluggable Authentication Modules vagy PAM) azonosíthatók. Az LDA-könyvtárat könnyű beállítani a rendszerhez, mindössze négy lépés szükséges hozzá.

1. A program telepítése

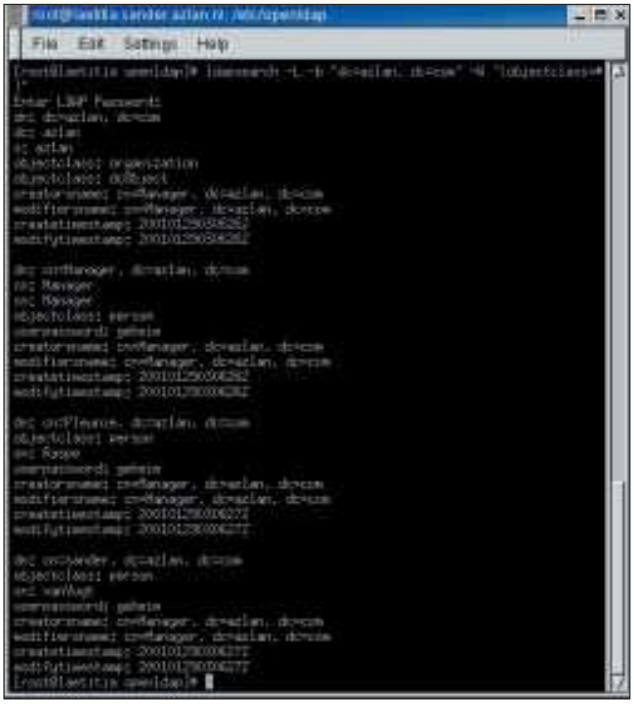
Ha az LDAP telepítése az alapértelmezett kiszolgálótelepítés folyamán még nem történt meg , a ➲ http://www.openldap.org címről vagy valamelyik tükörkiszolgálóról letölthetjük és telepíthetjük. A telepítéshez először tömörítsük ki:

```
tar -zxvf openldap-stable-xxxxx.tgz
```

ahol az xxxxx a letöltött fájl változatszáma. Lépjünk be az előzőleg létrejött könyvtárba, majd futtassuk le a `configure` parancsfájlt. Ez a parancsfájl ellenőrzi, hogy az LDAP telepítéséhez az összes feltétel rendelkezésre áll-e a rendszeren. Futtassuk a `make`-et; először el kell készítenünk a függőségeket a `make depend` segítségével, majd a programot egy egyszerű `make` utasítással le kell fordítani. Ezután futtassuk le a `make`-et a ***/test*** könyvtárban, ezzel ellenőrizve, hogy mindent helyesen fordítottunk-e. Végül a programot ténylegesen feltelepíthetjük a rendszerre: gépeljük be a `make install` parancsot abban a könyvtárban, amely a kicsomagoláskor keletkezett.

2. A slapd.conf beállításfájl átszerkesztése

A telepítést követően a /etc/openldap könyvtárban egy példát találhatunk a slapd.conf beállításfájlra. Ezt a saját rendszerünknek megfelelően át kell szerkesztenünk. Kezdetben nem szükséges túlbonyolítanunk fájlt; szerkesszük az *1. listához* (40. oldal) hasonlóra.
Vessünk egy pillantást a legfontosabb sorokra! Az első két sor feladata, hogy két másik beállításfájlt fűzzön be. Jelen esetben ezek a sémafájlok, amelyeket ugyan nem módosítottunk, de ettől még a `slapd`-nek meg kell mondani, hogy hol is találhatja meg őket. A `schemacheck = off` sor szintén nem túl érdekes – ez közli a `slapd`-vel, hogy szükségtelen ellenőriznie a sémát. A következő két sor ugyancsak külső fájlokra mutat: a ***slapd.pid***-re, amely a `slapd` által használt PID-számot tartalmazza, és a ***slapd.args***-ra, amely a `slapd` elindításakor megadott értékeket foglalja magában. Ezután egy olyan sort találunk, amely a felhasznált adatbázis típusát határozza meg. A megadható értékek: `ldbm`, `shell` és `passwd`, a legáltalánosabb az `ldbm`.



ldapsearch

Ezt három fontos bejegyzés követi. Az első a `"suffix dc=azlan, dc=com"` kezdetű sor, ez a bejegyzés határozza meg azt az általános tárolót, amelyben a `slapd` működni fog. Esetemben ez azonos annak a cégnek a DNS-nevével, ahol dolgozom. Ezt követi annak az azonosítója, akinek lehetősége lesz karbantartást vagy módosítást végrehajtani a teljes megkülönböztető névvel megadott adatbázison. A harmadik sor adja meg a karbantartó jelszavát; ez mint láthatjuk, egyszerű szövegként szerepel, ami azonban nem biztonságos, de erről még a későbbiek során szót ejtünk.
A `directory /usr/local/var/openldap-ldbm` sor azt a könyvtárat adja meg, ahol a LDAP-adatbázis telepítve van. Ellenőrizzük, hogy `700`-as módban van-e, valamint a `slapd`-folyamat tulajdonosa képes-e olvasni és írni.
E sorok után jó néhány olyan beállítás következik, amelyek ugyan nem feltétlenül szükségesek, de igen hasznosak lehetnek. Az első a `lastmod on`, amely nyomon követi azokat a felhasználókat, akik változtatásokat végeztek az objektumon. Erre a célra a `modifiersName`, a `modifyTimestamp`, a `creatorsName` és a `createTimestamp` tulajdonságokat használja. A következő beállítások segítségével indexelést végezhetünk. Az OpenLDAP sajnálatos módon nem éppen a leggyorsabb a fellelhető LDAP-könyvtárak között, ezért nem árt, ha néhány indexfájl segítségével kicsit felpörgetjük.
A `loglevel 64` beállítás, amely kiterjedt naplózást eredmé-

1. lista slapd.conf példa

```
include /etc/openldap/slapd.at.conf

include /etc/openldap/slapd.oc.conf
schemacheck off

pidfile /var/slapd.pid
argsfile /var/slapd.args

database ldbm
suffix "dc=azlan, dc=com"
rootdn "cn=manager, dc=azlan, dc=com"
rootpw secret

directory       /var/openldap-ldbm

lastmod on
index cn,uid

loglevel 64

defaultaccess   read

access to attr=userpassword
       by self write
       by dn="cn=manager, dc=azlan, dc=com"
             write
       by * compare
```

nyez, nagyon hasznos szolgáltatás lehet, ha gyorsabb működésre szeretnénk ösztökélni. Ennek legkisebb értéke `1`, a legnagyobb pedig 256. E két érték közt használhatjuk még a `2`, `4`, `8`, `16`, `32`, `64` és a `128` beállítást.
Végül néhány, a könyvtár elérési jogosultságára vonatkozó bejegyzést találunk. Az alapértelmezett `read` azt jelenti, hogy mindenki mindent olvashat, beleértve a jelszavakat is. A négy `access to attr=userpassword` kezdetű sor tartalmazza azokat a meghatározásokat, amelyek eldöntik, ki és mit tehet a jelszókkalaz adott könyvtárban. Az első sor engedélyezi, hogy mindenki módosíthassa a saját jelszavát. A rendszergazdának bármely jelszó átírására lehetősége nyílik, a felhasználók azonban csak olvasni tudják őket (természetesen, hiszen ez mindenképp szükséges ahhoz, hogy be tudjanak jelentkezni a rendszerbe).

3. A „slapd" indítása

Ha az ***slapd.conf*** fájlt kedvünkre átszerkesztettük, a következő lépés az `slapd` LDAP-démon indítása lesz. Ehhez az is elég, ha egyszerűen begépeljük, a `dn` kapcsolóval (ahol az `n` a kívánt hibakeresési szintet jelenti) azonban akár arra is utasíthatjuk, hogy minden nyomkövető üzenetet mutasson meg.

4. Adjunk adatokat a könyvtárhoz!

Most már áttérhetünk a következő lépésre, az adatok könyvtárba töltésére. Ebben a példában néhány egyszerű adatot fogunk beilleszteni, ehhez viszont előbb egy LDIF-fájlt kell szerkesztenünk a 2. listában (23. CD Magazin/OpenLDAP könyvtárban) láthatóhoz hasonló tartalommal.
Ha elkészítettünk egy a 2. listához hasonló fájlt, amelyet, tegyük fel, *~/users.ldif*-nek neveztünk, a következő paranccsal adhatjuk a könyvtárhoz:

```
ldapadd -D "cn=manager, dc=azlan, dc=com"
➥-W < ~/users.ldit
```

Meg kell adnunk a jelszót, amely azonos azzal, amit az slapd.conf-ban a rendszergazda-bejegyzéshez megadtunk. Ha minden jól ment, már képesek vagyunk adatot adni a könyvtárhoz.
Számos hibát azáltal is egyszerűen kiküszöbölhetünk, hogy ellenőrizzük, fut-e egyáltalán az `slapd` (igen, már kellene futnia), illetve nincs-e valahol a beállítás- vagy LDIF-fájlokban felesleges szóköz.

5. Nézzük meg, működnek-e a dolgok

Ha az adatot már a könyvtárhoz adtuk, a következő parancsokkal győződhetünk meg arról, hogy a rendszer működik-e:

```
ldapsearch -L -b "dc=azlan, dc=com"
➥-W "(objectclass=*)"
```

Eredményképp a könyvtárba helyezett összes adatot vissza kell kapnunk (lásd a *képen*).
Ha egyszer eljutottunk idáig, már elég sok mindent megtehetünk a könyvtárban. Többek között foghatjuk a böngészőnket és megtekinthetjük az LDAP-könyvtár adatait. És ez még csak nem is a legérdekesebb rész! További lehetőségként Linux-ügyfelünket úgy is beállíthatjuk , hogy az azonosítást többé ne a helyi jelszó- és árnyékfájlokon, hanem az LDAP-kiszolgálón keresztül végezze, így a felhasználók felügyeletét egyetlen pontba gyűjthetjük, s nem kell a saját jelszófájllal rendelkező számítógépek százaival foglalkoznunk. Ezt a következőképpen valósíthatjuk meg.

1. Telepítsük a programot

Mielőtt az ügyfelünket LDAP-kiszolgálón keresztüli azonosításhoz állítanánk be, bizonyosodjunk meg afelől, hogy minden szükséges program fel van-e telepítve. Ha RPM-alapú rendszert használunk, az ***openldap***, ***auth_ldap*** és ***nss_ldap*** csomagoknak kell fent lenniük. Ezt az `rpm -q` *`csomagnØv`* utasítással könnyen ellenőrizhetjük. Amennyiben nincsenek meg, a ➲ http://rpmfind.com címen megtalálhatjuk őket.

2. Az „ldap.conf" szerkesztése

A rendszereken gyakorta két ***ldif.conf*** nevű fájl is található. Az első a */etc* könyvtárban lelhető fel, és az `nss_ldap`, illetve a `pam_ldap` használja a szükséges adatok meghatározásához. A másik a */etc/openldap*-ban helyezkedik el, és az `ldapadd`, valamint az `ldapsearch` eszközök használják annak meghatározására, hogy melyik tárolón dolgozzanak. Amint már korábban is tettük: töröljük az egyiket, majd a munkánk leegyszerűsítése végett a helyére készítsünk közvetett hivatkozást a másik példányra. Ha elkészültünk, a szükséges adatokat beilleszthetjük. Egy egyszerű beállításhoz mindössze két sor szükséges:

```
BASE    dc=azlan, dc=com
HOST    laetitia.azlan.com
```

Az első sor határozza meg az alapértelmezett tárolót, ahol az ügyfél az adatokat keresni fogja, a második sor ad nevet az LDAP-kiszolgálónknak. Természetesen rendszerünknek képesnek kell lennie kiértékelni (resolve) ezt a nevet valamilyen DNS vagy hasonló eszközön keresztül, vagy az IP-címet kell használnunk.

3. Az „nsswitch.conf” szerkesztése

Következő lépésként meg kell adnunk a névszolgáltatás-kapcsolónak, hogy hol keresse az adatokat. Ezt a */etc/nsswitch.conf* fájl szerkesztésével tehetjük meg. A fájl a következő sorokat tartalmazhatja:

```
passwd: files ldap
shadow: files ldap
group: files ldap
```

Rendszerünk elsőként a fenti sorok segítségével kísérli meg helyi jelszófájlokon keresztül a felhasználók azonosítását, majd ha így nem jár sikerrel, az LDAP-adatbázison keresztüli azonosítással próbálkozik. Ha tehát a felhasználó a */etc/passwd* fájlban már létezik és a */etc/shadow*-ban jelszóval is rendelkezik, akkor az LDAP-rendszer használatára nem kerül sor.

4. PAM-beállítások szerkesztése

Ezután vessünk egy pillantást a PAM-fájlra. A legtöbb Linux-terjesztésben minden program ezt a korszerű módszert használja, ha csak egy csepp köze is van a felhasználóazonosításhoz. Ezt használják többek közt a `login`, az `ftp`, az `su`, az `ssh`, a `passwd` és további programok is. A PAM jelenlegi változataiban minden egyes ilyen fájlnak létezik beállításfájlja, amelyek alapesetben a */etc/pam.d* könyvtárban találhatók. Ezekben a beállításfájlokban adhatjuk meg, hogy az illető modul mely PAM-modulokat használhatja.
Amennyiben azt szeretnénk, hogy a ***login*** folyamat az azonosítást az LDAP-n keresztül végezze, a megfelelő beállításfájlnak valahogy úgy kell kinéznie, ahogyan azt a 3. listában (23. CD Magazin/OpenLDAP könyvtárban) láthatjuk.
Nézzük meg egy kicsit részletesebben! A felhasználó- és a jelszóadatokat négy folyamatban használjuk. Az első az azonosítás, ezt a PAM-fájlban az `auth` jelképezi. Ez a folyamat enged be minket a rendszerbe, illetve a feladatai közé tartozik a jelszó ellenőrzése is. A következő az `account` (felhasználói név), amely ellenőrzi, hogy a felhasználónak van-e valamilyen korlátozása, ami miatt nem jelentkezhet be a rendszerbe. Ezután következik a `password` (jelszó), amelyet akkor használunk, ha a jelszavunkat meg szeretnénk változtatni. Végül a `session` (munkamenet) határozza meg azokat a feladatokat, amelyeket akkor kell végrehajtani, ha egy olyan rendszeren szeretnénk más erőforrásokat használni, ahol már korábban azonosítottuk magunkat.
Minden ilyen modulnak külön jól meghatározott feladata adódik. Ezeket a feladatokat a PAM-modulok tartalmazzák, amelyek közül az egyik legfontosabb a ***pam_unix.so***. Ez a modul felügyeli a hagyományos jelszó-, illetve árnyékjelszó-azonosítás folyamatát és többnyire elengedhetetlen a rendszer eléréséhez. Ha azonban LDAP-t használunk, az is megfelel, ha az LDAP enged be, ezért a `pam_unix` sor meghívása előtt valahol léteznie kell egy sornak, amely a `pam_ldap`-t hívja meg. Ez nem kötelező (valószínűleg akkor is el szeretnénk érni a rendszert, ha az LDAP-kiszolgáló leáll), de elégséges. Következésképp ha a `pam_ldap` segítségével azonosítjuk magunkat, már nem szükséges a `pam_unix`-hoz is fordulnunk. A két nagyobb modul mellett még néhány kisebb modul is létezik, amelyek tárgyalásától most terjedelmi okok miatt eltekintünk.

5. Felhasználók létrehozása az összes szükséges tulajdonsággal

Miután megtettük a négy bevezető lépést, számítógépünk készen áll az LDAP-könyvtáron keresztüli azonosításra. Vajon a könyvtárunk szintén készen áll? Könyvtárunk azonosításhoz történő felkészítéséhez az összes szükséges felhasználói tulajdonságot be kell illeszteni, mindazokat, amelyek egyébként a */etc/passwd* és */etc/shadow* fájlokban megtalálhatók. Helyesen előállított felhasználói azonosítók nélkül miként használhatnánk értékes erőforrásainkat és e fájlok egyéb hasznos képességeit? Az adat megszerzéséhez olyan Perl-parancsfájlokat használhatunk fel, amelyeket kifejezetten a gépünkön található fájlok adatainak kigyűjtésére és LDAP-adatbázisba vitelére fejlesztettek ki, esetleg saját LDIF-fájlt is készíthetünk, amellyel felvihetjük a kívánt felhasználókat.
Ha az önműködő megoldás mellett döntünk, több Perl-parancsfájlból is választhatunk a ➲ http://www.padl.org-on. Olyan parancsfájlok is találhatók itt, amelyek majdnem minden beállítást össze képesek gyűjteni, legyen az NIS-adatbázis jelszófájl, a hostfájlunk, a networkfájlunk stb. Mielőtt azonban használnánk őket, az általános beállításfájlt, a ***migrate_common.ph***-t át kell szerkesztenünk. Ennél néhány olyan értéket kell megváltoztatnunk, amely az adat létrehozásának helyét határozza meg. Különösen fontos a `DEFAULT_MAIL_DOMAIN` és a `DEFAULT_BASE`; ezek határozzák meg azt a DNS-tartományt, ahol a felhasználók címei találhatók, illetve azt az LDAP-tárolót, ahol a felhasználókat létre kell hozni. Ha ez megtörtént, elkezdhetjük a bevitelt. Minden egyes adathoz egy-egy külön parancsfájlt találunk; a legérdekesebb közülük talán a `migrate_all_online.sh`, amely az összes hálózati adatot, illetve a `migrate_passwd.pl`, amely a felhasználókat gyűjti össze a rendszeren.
A másik lehetőség egy saját LDIF-fájl készítése, ennek tartalmát az `ldapadd` segítségével az adatbázishoz kell adnunk. Ne feledjük, az összes jogosultságot be kell állítanunk! A 4. listában (23. CD Magazin/OpenLDAP könyvtárban) bemutatott példán láthatjuk, hogyan valósíthatjuk ezt meg a legegyszerűbben.
Ennek a módszernek két hátránya van: az egyik, amikor a felhasználót az LDIF-adatbázisban létrehozzuk, de a felhasználói-könyvtár nem készül el önműködően (igaz, létezik egy ***pam_mkhomedir.so*** nevű PAM-modul, amely ezt a nehézséget is megszünteti). A másik gond a felhasználók jelszavaival kapcsolatos, sajnos ugyanis nincs rá jó módszer, hogy az adatbázisba titkosítva rejthessük el őket. Nem túl elegáns megoldásként javasolhatjuk, hogy a felhasználót hozzuk létre a /etc/passwd és /etc/shadow fájlokban, adjunk neki jelszót, majd a titkosított karaktersorozatot másoljuk ki a /etc/shadow-ból, és rakjuk be az LDIF-fájlba.
Ezután nem maradt más hátra, próbáljuk ki a rendszert: töröljük a felhasználót a helyi fájlokból, nyissunk meg egy bejelentkezési ablakot, és kíséreljünk meg bejelentkezni – ha minden jól ment, sikerrel járunk.

Sander van Vugt
(sander.van.vugt@azlan.nl) Hollandiában él. Linux-, Novell- és Nortel-szakoktatóként az Azlan Trainingnél dolgozik, és már jó néhány könyvet és cikket írt a Linuxról.

Kapcsolódó címek

OpenLDAP ➲ http://www.openldap.org
RPM Find ➲ http://www.rpmfind.com
Perl Scripts ➲ http://www.padl.org

Szolgálatára! – munkaütemezés Linux alatt

Egyszerű parancssoros eszköz késleltetett programvégrehajtás tervezésére és kezelésére: az at.

A kapcsolattartás világából kölcsönözve a szolgáltatásokat két nagy csoportra bonthatjuk: azonnaliakra és késleltetettekre. Az azonnali kapcsolattartási szolgáltatásokhoz tartozik a telekonferencia, a hálózati csevegés stb., késleltetett szolgáltatás a levél (e-mail) és a fax. Ezek a sorok a Linux `at` eszközének használatáról adnak rövid áttekintést, amely a programok késleltetett (vagy időzített) végrehajtására képes.

Természetesen a legtöbb programot azonnal szeretnénk indítani. Az sem ritka azonban, hogy kívánatosabb (esetleg egyenesen elkerülhetetlen) lenne, ha a program (illetve munka – job) indítása valamilyen okból kifolyólag egy későbbi időpontban történne meg. Gondoljuk végig a következő eseteket:

1. Egy fejlesztőcsoport kifejlesztett egy parancsfájlt, amely újraépíti a forrásfát és frissíti az alkalmazást. A parancsfájl teljes mértékben gépesített, de igencsak idő- és processzorigényes. Gyakorlati okokból a csoport úgy döntött, hogy kipróbálásának feladatát egy késő éjszakai időpontra időzíti, amikor viszonylag kevés felhasználóval és folyamattal kell versenyeznie.
2. Egy nagyfelbontású színes nyomtató igen nagy terhelés alatt áll az üzleti órákban, ezért a projektvezető a gyorsan bővülő projektterv több mint 200 oldalát egy héten többször is újranyomtatja, elkerülve azonban a nyomtató csúcsidőszakbeli teljes kisajátítását, a feladatot a kora reggeli órákra időzíti (még a legtöbb felhasználó megérkezése előtt).
3. A projektvezető levélben emlékeztetőt, illetve napirendet szeretne szétküldeni minden résztvevőnek, négy órával a következő péntek délután három órára időzített találkozó előtt.
4. A rendszergazda jelentést kap, miszerint a munkanap elején a rendszerteljesítmény különlegesen gyenge. A gyengülő teljesítmény oka azonban nem teljesen nyilvánvaló. A rendszergazda a számos naplófájl végigböngészése helyett úgy dönt, időzít egy programot, amely pillanatképet készít a rendszerterhelésről. A feladatot következő nap kilenc órától tízig minden öt percben le kell futtatni.
5. A helyi raktár adatai frissítésére használt külső adatforrás minden péntek délután hat órakor válik elérhetővé. Ha a hosszadalmas frissítési folyamat megszakad vagy leáll, egy órával később próbálkozhat újra, máskülönben a következő péntekre halasztódik el.

A fenti példákban felvázolt esetek időzítési követelményei az `at` eszköz igénylését tükrözik. Sőt, ezek a példahelyzetek a leggyakoribb `at`-alkalmazások egészen széles kínálatát ölelik fel. Az `at` eszményi a nagy számításigényű programoknak kis felhasználói terhelésű időszakokban történő végrehajtására; a szűk erőforrásoknak általában a jobban elérhető időszakokban történő használatára; emlékeztetőknek az adott időpontban való elküldésére; folyamatok olyan időben történő végrehajtására, amikor a felhasználó nem szándékozik aktívan a rendszerhez csatlakozni; illetve olyan munkafolyamatok végrehajtására, amelyek csak valamilyen adott időpontban elérhető erőforrástól függnek.

Az at képességei

Az `at` tulajdonképpen az e feladatokhoz kapcsolódó programok gyűjteménye, amelyek segítségével a Linux-felhasználók az időzített munkák végrehajtását kezelhetik és ütemezhetik. Azt gondolhatnánk, hogy az `at` csak olyan folyamatokhoz használható, amelyek nem igényelnek interaktivitást – és ez így is van. Az `at` segítségével ütemezett programok csak olyanok lehetnek, amelyek háttérfolyamatként is képesek futni, az `at` ugyanis a végrehajtáshoz nem rendel felhasználói megjelenítő felületet. A felhasználók a parancssoros felületet általában elég rugalmasnak találják, emellett hatékony is, így minden elképzelhető időzítési követelménynek megfelel. Mielőtt belekezdenénk a parancssoros példák bemutatásába, foglaljuk össze az `at` által nyújtott alapvető szolgáltatásokat. Egyfelől lehetőséget nyújt, hogy a munka végrehajtását egy adott időpontra, illetve dátumra időzítsük, másfelől képes a jelenleg időzített munkákat megjeleníteni, egy adott időzített munkát leállítani, valamint az `at` használatára jogosult felhasználók listáját kezelni.

Egyéb „at”-tények

- Az `at` nem cron! Azok a felhasználók, akik az `at`-et egyáltalán nem vagy csak kevéssé ismerik, hajlamosak egy kalap alá venni a cron-alkalmazásokkal. Igaz mind az `at`, mind a cron ütemezett feladat-végrehajtást tesz lehetővé, a cron azonban kifejezetten szakaszosan végrehajtott feladatokhoz (például mindennap délután négykor futtatandó programok) lett kifejlesztve. Az `at`-et ezzel szemben olyankor használjuk, amikor a feladatot csak egyszer szeretnénk – valamilyen adott időben – végrehajtani, vagy amikor a végrehajtás dátumát programozottan kell meghatározni.
- Ha az `at` más megvalósításait is használtuk már, megfigyelhetjük, hogy a kimenet és a parancssori írásmód tekintetében különbségek vannak a linuxos `at` és más üzleti Unixok között.
- Mennyire lehet egy feladatot „a jövőbe” időzíteni? A példák többségének futtatásához használt rendszer hamar morcossá vált, amikor a feladatot 2037 utánra időzítettem. Ugyanakkor, nem hiszem, hogy ez bármely felhasználónak komoly hátrányt jelentene.
- Az `at` és `atq` programok által megjelenített időértékek olvashatóbbak, ha a POSIXLY_CORRECT változó be van állítva, például: `POSIXLY_CORRECT=1; export POSIXLY_CORRECT`
- Az `atq` parancs a `-l` kapcsoló, az `atrm` pedig a `-d` kapcsoló rokon értelmű kifejezése.

Az „at” használata

Az itt bemutatott példák feltételezik, hogy valamilyen szintű ismeretekkel már bírunk a héjprogramok végrehajtását és a Bourne-héj formai követelményeit illetően. Minden példát kipróbáltam a RedHat Linux 7 (2.3 rendszermag) alatt, és úgy

1. táblázat **Lehetséges idő- és dátumbejegyzések**

Jelölés	*Értelmezés*
10:15 pm 10/10/2001	2001. október 10. este negyed tizenegy
12:15 7/4/2001	2001. július 4. negyed egy
14:00 7/4/2001	2001. július 4. délután kettő
8 am Apr 15, 2001	a idén április 15. reggel nyolc
10:10 Sep 5, 2001	2001. szeptember 5. tíz óra tízkor
noon 10 pm tomorrow	ma (vagy holnap) délben holnap este tízkor
teaidő	adott nap (vagy azt követő nap) délután négykor
now + 5 minutes	pontosan öt perc múlva
now + 1 week	pontosan egy hét múlva
tomorrow + 1 hour	pontosan 25 óra múlva

gondolom, ezek minden általános terjesztés alatt változatlan formában működnek.
Az `at` legnagyobb erőssége, hogy segítségével a programokat későbbi időpontra időzíthetjük. Általában a */usr/bin* könyvtárban találjuk, és alapvető parancssoros írásmódja a következő:

```
at [kapcsol k] ID  [D`TUM] < bash-hØjprogram
```

A fenti írásmód követelményeinek megfelelően a következő hívási módok is ugyanilyen jól működnek:

```
cat bash-hØjprogram | at [kapcsol k] ID  [D`TUM]

at [kapcsol k] ID  [D`TUM]
## BÆrmi, amit rendes esetben a Bourne-hØj
## elfogad Øs Ørtelmezni tud. LezÆrva a
## CTRL-D karakterrel
utas tÆs 1
utas tÆs 2
utas tÆs 3
utas tÆs 4
CTRL-D
```

Az első lényeges részlet, amit érdemes megismernünk, az idő és a dátum helyes formázása. A dátum mindig elhagyható. Ha kihagyjuk a parancssorból, a program azt a dátumot feltételezi, amihez képest az adott időpont a legközelebb van: azaz vagy az adott, vagy az azt követő napot. Ha a megadott időpont délután fél kettő, de már fél öt van, a feladat másnap fél kettőre lesz időzítve. Az `at` a kiterjesztett POSIX.2 szabványnak megfelelő dátum- és időadatok beolvasására képes. A jelölésmód leírása valószínűleg megtalálható rendszerünk */usr/doc* könyvtárában, a ***timespec*** fájlban.
Az első táblában található idő- és dátummegadási példák elég jól mutatják, mit fogad el az `at` parancs – amint láthatjuk, egy igen gazdag parancshalmazt. A parancssorban kapott értékeket balról jobbra értelmezi. Ha az idő, illetve dátum szabályai sérülnek, a ***Garbled time*** hibaüzenet jelenik meg, és a program leáll. Általában valamilyen velős magyarázattal is szolgál a zavar okát illetően. Figyeljük meg például a következő híváspróbálkozást:

```
$ at 6am Mar 32
Error in day of month. Last token seen: 32
Garbled time
$
```

A következő hívás viszont sikeresen időzíti a feladatot. Emlékezzünk a korábban bemutatott harmadik példaalkalmazásra:

```
$ at 1pm friday
warning: commands will be executed
➥using /bin/sh
    STAFF="moe larry curly"
    cd mail
    mail -s "Meeting Reminder"
    ➥$STAFF < friday_agenda.txt
    CTRL-D
job 6 at Fri Apr  13 13:00:00 2001
$
```

Ha jobban belegondolunk, azért felmerül néhány kérdés. Tárolódnak-e valahol a szabványos kimenet- és a hibafolyamok kimenetei? Az `at` szolgáltatás (illetve a */usr/sbin/atd* program) minden szabványos kimenetet és hibaüzenetet elfog, és amikor az időzített feladat véget ér, levélben elküldi a parancsot kiadó felhasználónak. A levél a következő fejléccel fog megjelenni:

```
Subject: Output from your job 17
```

Vajon csak Bourne-héjprogramokat tudunk átadni? Amint a fenti példa visszajelzése is sejteti, az `at` a rendszer Bourne-héjprogramját használja (*/bin/sh*) a felhasználó által átadott utasítások értelmezésére. Így bármi, amit a Bourne-héjba gépelhetünk, azonnal érvénybe lép, ideértve a Bourne-héjprogramokat vagy bármilyen a környezetünkben található végrehajtható állomány indítását, tehát egy másik nyelv értelmezőjét is. Figyeljük meg a következő példát, ahol egy Perl-parancsfájlt időzítünk:

```
$ at 6pm tomorrow
warning: commands will be executed
➥using /bin/sh
    perl /home/moe/perls/script1.pl
    CTRL-D
job 28 at Wed Apr  18 18:00:00 2001
$
```

Milyen ***felhasználó/csoport*** azonosítók rendelhetők az időzített folyamathoz? A munkafolyamat környezetének milyen értékei őrződnek meg és jutnak tovább a végrehajtási környezetbe? Annak feltárása, hogy egy adott eszköz a háttérben miképpen dolgozik, a legtöbb felhasználó számára teljesen közömbös. Az `at` esetében az eszköz működését egy példán szemléltetve egyszerűen kaphatunk képet arról, hogy mit miért tehetünk meg, és mit miért nem. Minden egyes folyamatidőzítés eredményeképpen létrejövő parancsfájl a */var/spool/at/* könyvtárba kerül. Ezek önműködően létrehozott parancsfájlok és a következő részekre bonthatók:

- Programmegjegyzések, amelyek tájékoztatást adnak a program végrehajtásáról. Ugyanitt találhatók a feladathoz rendelt érvényes felhasználói és csoportazonosítók.
- Az `umask`-beállítások, ezek döntenek a fájlok és könyvtárak létrehozásának miként jéről.
- A környezeti változók teljes listája a folyamat elküldésének időpillanatában (kivéve azok, amelyek a megjelenítő eszközökkel kapcsolatosak, mint például a TERM és a DISPLAY).

2. táblázat **Parancssori lehetőségek**

Kapcsoló	*Viselkedés*
-f job_script	A job_script tartalma lesz a feladat tartalma. Ez a kapcsoló akkor lehet érdekes, ha az `at` parancsot parancsfájlból használjuk.
-m	A feladat elvégzéséről akkor is kap értesítést a feladó felhasználó, ha a program szabályosan lefutott vagy nem lépett fel a hiba.
-v	A felhasználó azonnali idő-, illetve dátumvisszajelzést kap. Azaz, megtudjuk, hogy az adott dátum- és időadatok miképpen értelmeződtek.

3. táblázat **Felhasználók szabályozása at-tel**

/etc/at.allow	*/etc/at.deny*	*Eredmény*
Nem létezik	Nem létezik	Csak a rendszergazdának van jogosultsága.
Létezik	-	Csak a */etc/at.allow* fájlban felsoroltak jogosultak.
Nem létezik	Létezik	Mindenki jogosult, kivéve a */etc/at.deny* fájlban felsoroltakat. Ez a jellemző alapértelmezett beállítás a Linux telepítésekor.

- A pillanatnyi könyvtár átváltása abba a könyvtárba, ahol a felhasználó a feladat ütemezésekor tartózkodott. Ha a folyamat végrehajtásakor a könyvtár már nem létezik, a folyamat megszakad, és figyelmeztető levelet kapunk.
- Végül az `at`-nek átadott programszöveg van hozzáfűzve.

Ezt mutatja be az 1. lista (23. CD Magazin/at_szolgalatara könyvtár) lerövidített példája, amely a *Moe* felhasználó által időzített folyamat eredményeképpen létrejött parancsfájlt tartalmazza. A program parancssoros felületének kiegészítéséhez vessünk egy pillantást a *2. táblázatra*, ahol néhány igen hasznos parancssori lehetőséget találunk.
A következő példa a feladat időzítésére mutat még egy módszert:

```
$ at -mf ~/shells/program1  6pm tomorrow
warning: commands will be executed
➥using /bin/sh
job 29 at Thu Apr  12 18:00:00 2001
$
```

Két másik program is tartozik az `at` alapprogramjához: az `atq` és az `atrm`. Az `atq` kilistázza a jelenleg időzített feladatokat, míg az `atrm` egy vagy több feladatidőzítést szakít meg. Míg azonban az összes munkafolyamat megtekintése és megszakítása a rendszergazda jogosultságai közé tartozik, addig a felhasználók csak a saját folyamataikat nézhetik meg, illetve állíthatják le.
A 2. lista (23. CD Magazin/at_szolgalatara könyvtár) az `atq` által megjelenített kimenetre mutat példát, ahogyan azt a rendszergazda láthatja (a mezőcímkéket az olvashatóság kedvéért én illesztettem be, az `atq` linuxos változata sajnos nem támogatja őket).
A ***rank*** (rang) egy egyedi sorozatszám, amely az időzített feladatok más programok általi azonosítására szolgál. Az időzítési időponton és a feladó felhasználón kívül a ***queue*** (sor) értéket találhatjuk itt. Hacsak a felhasználó másképpen nem rendelkezett, a feladatok az ***a*** sorba kerülnek. (A különböző ***queue***-értékek használatának mikéntjéről a megfelelő súgóoldalon olvashatunk – tömören megfogalmazva ez befolyásolja a feladat futásidejű besorolását). Az `atq` a jelenleg futó feladatokat a ***queue*** sorban az = értékkel jelöli, így a *3. táblázatbn* látható 17-es feladat éppen az adott pillanatban hajtódik végre.
Mi történik, ha a felhasználó elküldött egy feladatot, de ráébredt, hogy az időpont, illetve a program nem a megfelelő volt? Az `atrm` eszközt egy vagy több időzített feladat eltávolítására használhatjuk. Például a rendszergazda a *3. táblázatban* található első és második feladatot a következő paranccsal törölheti:

```
atrm 18 19
```

Az `atrm` nem ad visszajelzést. A parancsot követő `atq`-listázás pedig a 3. listában (23. CD Magazin/at_szolgalatara könyvtárban) látható eredményt adta.

Az at felügyelete

Ha valamelyik `at`-program a következő üzenetet adja: ***You do not have permission to use at***, lépjünk kapcsolatba a rendszergazdával.
Ahogy el is várjuk, a rendszergazda az eszköz használatát illetően teljes körű jogosultsággal rendelkezik, és ugyanezeket a jogokat a felhasználóknak is megadhatja. Két rendszerfájl, a */etc/at.allow* és a */etc/at.deny* irányítja az `at`-eszközelérést. A *3. táblázatban* mutatja be, hogy e fájlok megléte és tartalma miképpen befolyásolja az adott rendszeren a felhasználók jogosultságait.
Figyeljünk arra, hogy amennyiben a */etc/at.allow* fájl létezik, a rendszer a */etc/at.deny* fájlt figyelmen kívül hagyja. A felhasználók mindkét fájlban Linux-azonosítójuk révén vannak megadva, külön-külön, soronként egy névvel. Az `at` e fájlok szerkesztésére nem ad parancssoros eszközt. Ha szükséges, a rendszergazda ezeket a fájlokat a kedvenc szerkesztőjét használva kézzel szerkeszti.
Összefoglalásul: a felhasználók számára az `at` használatát közvetlen vagy közvetett módon lehet megtiltani vagy megengedni. A rendszergazda a feladatnak megfelelően kezelheti az elérést: kizárásos vagy engedélyezéses alapon. Válasszuk azt a megoldást, amelyik telepítésünknek a legjobban megfelel. Azoknál a gépeknél, amelyeknél a biztonság nagyon fontos, az `at`-ot engedélyezéses alapon érdemes működtetni (azaz a felhasználóknak nem lesz jogosultságuk, hacsak kifejezetten engedélyt nem kapnak – a */etc/at.allow* fájl létezik). Ugyanakkor a Linux alapértelmezett beállítása a legtöbb fejlesztő-, illetve próbakörnyezethez tökéletes lehet (a felhasználók tehát jogosultak, hacsak nincsenek kitiltva – a */etc/at.allow* nem létezik, és a */etc/at.deny* nulla vagy több bejegyzést tartalmaz).

Összegzés

Összességében az `at`-programok a késleltetett alkalmazás-végrehajtás feladatára kínálnak lehetőséget. Minden egyszerűsége és hasznossága dacára a Linux-rendszergazdák és -fejlesztők gyakran elfeledkeznek az `at` eszközről.

Louis J. Iacona
(ljj@omnie.com) 1982 óta Linux/Unix alatti alkalmazásokat tervez és fejleszt. Jelenleg az OmniE Labs, Inc vezető embere (➲ http://www.omnie.com).

Hogyan írjunk linuxos USB-eszközvezérlőt?

Greg USB-vezérlővázlatát osztja meg velünk és azt is megmutatja, miképpen formálhatjuk azt a saját igényeinknek megfelelően.

A Linux USB-alrendszer a 2.2.7 rendszermagban megjelent mindössze két különböző eszköz (az egér és a billentyűzet) támogatásából nőtte ki magát. Mára a 2.4-es rendszermag több mint húsz különféle eszköztípust támogat. A Linux jelenleg csaknem minden USB-osztályú eszközt támogat (például a billentyűzetet, az egeret, a modemet, a nyomtatót és a hangszórót), illetve folyamatosan növekszik a gyártófüggő eszközök (mint az USB – soros átalakítók, digitális kamerák, etherneteszközök és MP3-lejátszók) támogatottsága. A jelenleg támogatott különféle USB-eszközök teljes listáját a ***Kapcsolódó címek*** között találhatjuk meg.

A fennmaradó USB-s eszközök, amelyekhez nincs Linux-támogatás, csaknem mind gyártótól függő eszközök. Minden gyártó maga határozhatja meg saját eszköze protokollját, amelyen keresztül az eszközzel kapcsolatot lehet tartani – emiatt többnyire saját vezérlőt kell írnunk. Néhány gyártó USB-protokollját nyílttá tette és segíti a Linux-vezérlők készítését, mások nem teszik közzé, ezért a fejlesztők kénytelenek visszafejteni azt. Ez pedig komoly jogi kérdéseket vonhat maga után, de szerencsére egyre több gyártó látja be, hogy a Nyílt Forrás Közössége is értékes célréteg számára.

Mivel minden eltérő protokollhoz új vezérlőt kell készíteni, egy általános USB-vázat hoztam létre, amelynek alapjául a ***pci-skeleton.c*** fájl szolgált. Ez az a fájl a rendszermag-forrásfában, amelyen igen sok PCI hálózati kártyavezérlő alapul. Egy USB-váz található még a ***drivers/usb/usb-skeleton.c*** helyen a rendszermag-forrásfában. Ebben a cikkben végigsétálunk a vázvezérlő alapjainak lépésein, elmagyarázom az egyes részeket, illetve elmondom, mit kell tennünk, hogy egy adott eszközhöz igazítsuk.

Ha linuxos USB-vezérlőt akarunk írni, elsőként nem árt, ha megismerkedünk az USB protokollal. Ezt számos más hasznos leírással együtt az USB honlapon találjuk (lásd a ***Kapcsolódó címeket***). A linuxos USB-alrendszerrel kapcsolatos kitűnő bemutatkozó írást találhatunk az „USB Working Devices List” helyen. Ez bemutatja, miképpen épül fel az USB-alrendszer, és az olvasót megismerteti az USB ***urbs***-ok fogalmával, amelyek létfontosságúak az USB-eszközökhöz.

Az első dolog, amit a linuxos USB-vezérlőnek meg kell tennie, hogy bejegyzi magát a Linux USB-alrendszeren. Ilyenkor néhány adatot ad át arról, hogy a vezérlő mely eszközöket támogatja, és milyen eljárásokat kell meghívnia, amikor a vezérlő által támogatott eszközt csatlakoztatnak vagy választanak le a rendszerről. Mindezen adatok az USB-alrendszerhez kerülnek, és az `usb_driver` szerkezetben tárolódnak. A vezérlőváz az `usb_driver`-t a következőképpen határozza meg:

```
static struct usb_driver skel_driver = {
      name:   "skeleton",
      probe:  skel_probe,
      disconnect:    skel_disconnect,
      fops:   &skel_fops,
      minor:  USB_SKEL_MINOR_BASE,
      id_table:      skel_table,
};
```

A `name` változó egy karaktersorozat, amely a vezérlőt azonosítja. A rendszer ezt használja a rendszernaplókba írt üzenetekhez. A `probe` és `disconnect` függvények mutatóit akkor kell meghívnunk, amikor az `id_table` változóban megadott adatoknak megfelelő eszköz megjelenik vagy eltávolítják.

A `fops` és `minor` változók elhagyhatók. A legtöbb USB-vezérlő más rendszermag-alrendszerekre is csatlakozik, mint például SCSI-, hálózati vagy TTY-alrendszer. Ezek a típusú vezérlők más rendszermag-alrendszerekben is bejegyzik magukat, és minden felhasználói beavatkozás e csatolófelületeken keresztül megy végbe. Azon vezérlők esetében, amelyekhez nem tartozik rendszermag-alrendszer, mint például az MP3-lejátszóknál vagy a lapolvasóknál, valamilyen módszert kell találnunk a felhasználóval való kapcsolattartásra. Az USB-alrendszer lehetőséget ad kisebb eszközszám (minor device number) megadására és olyan `file_operations` függvénykészlet-mutatók megadására, amelyek megengedik ezt a felhasználói beavatkozást. A vázvezérlőnek ilyen csatolófelületre van szüksége, ezért a `file_operations` függvényekre egy kisebb kezdőszámot és egy mutatót szolgáltat.

Az USB-vezérlőt ezután az `usb_register` hívással bejegyezzük, általában a vezérlő `init` függvényében, ahogyan azt az *1. lista* is mutatja.

Amikor a vezérlő törlődik a rendszerből, az USB-alrendszerből ki kell jelentenie magát. Ezt az `usb_unregister` függvénnyel tehetjük meg:

```
static void __exit usb_skel_exit(void)
{
   /* a vezØrlıt kijelentj k az
   USB-alrendszerbıl*/
   usb_deregister(&skel_driver);
}
module_exit(usb_skel_exit);
```

Ha a linux-hotplug rendszert engedélyezni szeretnénk, hogy a megfelelő vezérlő önműködően betöltődjék, amikor az eszközt csatlakoztatják, egy `MODULE_DEVICE_TABLE` táblát kell készítenünk. A következő kód a `hotplug` parancsfájlnak azt mutatja meg, hogy ez a modul egyetlen eszközt támogat – egy adott gyártó- és termékazonosítóval:

```
/* a vezØrlıvel mßk dı eszk z k listÆja */
static struct usb_device_id skel_table [] = {
    { USB_DEVICE(USB_SKEL_VENDOR_ID,
      USB_SKEL_PRODUCT_ID) },
    { }            /* Megsz ntetı bejegyzØs */
};
MODULE_DEVICE_TABLE (usb, skel_table);
```

1. lista Az USB-vezérlő bejegyzése

```
static int __init usb_skel_init(void)
{
   int result;

   /* a vezØrlıt bejegyezz k az USB-
      alrendszerben */
   result = usb_register(&skel_driver);
   if (result < 0) {
      err("usb_register failed for the"
           ➥__FILE__
           ➥" driver. Error number %d",
           ➥result);
      return -1;
   }

   return 0;
}
module_init(usb_skel_init);
```

2. lista A skel_write függvény

```
/* mind ssze egyetlen egy urb-ot tudunk  rni */
bytes_written = (count > skel->bulk_out_size)
  ? skel->bulk_out_size : count;

/* az adatot a felhasznÆl t l mÆsoljuk az urb-ba */
copy_from_user(skel->write_urb->transfer_buffer,
  ➥buffer, bytes_written);

/* az urb beÆll tÆsa */
FILL_BULK_URB(skel->write_urb, skel->dev,
            usb_sndbulkpipe(skel->dev,
            skel->bulk_out_endpointAddr),
            skel->write_urb->transfer_buffer,
            bytes_written,
            skel_write_bulk_callback,
            skel);

/* az adatot Ætk ldj k bulk-kapun */
result = usb_submit_urb(skel->write_urb);
if (result) {
      err(__FUNCTION__ " - failed submitting write
            ➥urb, error %d", result);
}
```

Vannak más makrók is, amelyeket olyan `usb_device_id`-k leírásához használhatunk fel, amelyek egy egész USB-vezérlő-osztályt támogatnak. További adatok az ***usb.h*** fájlban találhatók. Amikor az USB-sínre olyan eszközt csatlakoztatunk, amelynek device ID-mintája megegyezik a USB core-ban bejegyzett vezérlőével, a `probe` függvény hívódik meg. Az `usb_device` szerkezet, a csatolószám és a csatolóazonosító értékként átadódik a függvénynek:

```
static void * skel_probe(struct usb_device
              *dev, unsigned int ifnum, const
              struct usb_device_id *id)
```

A vezérlőnek most ellenőriznie kell, hogy az eszköz valóban olyan, mint amilyet vár. Ha az az alaphelyzetbe állítás közben nem ilyennek bizonyul, vagy bármilyen hiba lép fel, a `probe` függvényből `NULL` értékkel tér vissza. Máskülönben a vezérlő visszatérési értéke egy olyan mutató, amely az ehhez az eszközhöz tartozó állapotot leíró belső adatszerkezetre mutat. Ez a mutató az `usb_device` szerkezetben tárolódik, és a vezérlő összes meghívása (`callback`) majd ezt az értéket adja át.
A vezérlővázban megadjuk, hogy milyen végpontok vannak adattömeg-bemenetként (bulk data input) és -kimenetként megjelölve. Készítünk egy átmeneti tárat, ami tárolja az eszközre kiküldésre szánt, illetve onnan beolvasott adatot, illetve egy USB urb-ot, ahová az adatot az eszköz alaphelyzetbe állításához írjuk. Ugyanakkor az eszközt a devfs alrendszeren bejegyezzük, hogy a devfs felhasználói elérhessék. Ez a bejegyzés valahogy így néz ki:

```
/* az eszk zh z alaphelyzetbe Æll tunk egy
   devfs csom pontot Øs bejegyezz k */
sprintf(name, "skel%d", skel->minor);

skel->devfs = devfs_register
      (usb_devfs_handle, name,
      DEVFS_FL_DEFAULT, USB_MAJOR,
      USB_SKEL_MINOR_BASE + skel->minor,
      S_IFCHR | S_IRUSR | S_IWUSR |
      S_IRGRP | S_IWGRP | S_IROTH,
      &skel_fops, NULL);
```

Ha a `devfs_register` függvény kudarcot vall, nem törődünk vele, hiszen a devfs alrendszer úgyis jelenteni fogja a felhasználónak.
Ugyanígy, amikor az eszközt eltávolítják az USB-sínről, a `disconnect` függvény hívódik meg az eszköz mutatójával. A vezérlőnek minden saját adatot, amit csak lefoglalt, ki kell ürítenie, és minden függőben lévő ***urb***-ot le kell zárnia az USB-rendszeren. A vezérlő a devfs alrendszerről a következő hívással egyúttal ki is jelenti magát:

```
/* eltÆvol tjuk a devfs csom pontot */
devfs_unregister(skel->devfs);
```

Most, amikor az eszközt már csatlakoztattuk a rendszerhez és a vezérlőt az eszközhöz rendeltük, az USB-alrendszernek átadott `file_operations` szerkezetben található bármely függvény meghívható egy felhasználói programból, amely megpróbál kapcsolatot tartani az eszközzel. Az első meghívott függvény az `open` lesz, hiszen a program az eszközt megpróbálja megnyitni ki- és bemenetre (I/O-ra). Ha a vezérlőváz `open` függvényében modulról van szó, a vezérlő használtság-számlálóját megnöveljük a `MODULE_INC_USE_COUNT` hívással. Ezzel a makróhívással, ha a vezérlőt modulként fordítottuk, a vezérlőt addig nem törölhető, amíg a megfelelő `MODULE_DEC_USE_COUNT` makró le nem fut. Növeljük saját felhasználás-számlálónkat és a mutatót mentsük a fájlrendszerben található belső szerkezetbe. Ezt azért tesszük, hogy a későbbi fájlműveletek engedélyezzék a vezérlőnek, hogy az meghatározhassa, melyik eszközt címzi meg a felhasználó. Mindezeket a következő kód hajtja végre:

```
/* n velj k meg a modulhasznÆlat-szÆmlÆl t*/
MOD_INC_USE_COUNT;
```

```
++skel->open_count;

/* objektumunkat ments k a fÆjl sajÆt
   szerkezetØbe */
file->private_data = skel;
```

Az `open` függvény meghívása után a `read` és `write` következik, hogy az eszköznek adatot fogadjanak, illetve küldjenek. A `skel_write` függvényben olyan adatra fogadunk mutatót, amit a felhasználó az eszközre szeretne küldeni. A függvény meghatározza, hogy az elkészített írási ***urb*** alapján mennyi adatot tud az eszközre küldeni (ez a méret függ a tömeges kimenetnek az eszközben meghatározott végpontjától). Ezután az adatot a felhasználói területről a rendszermagterületre másolja, az ***urb***-ot rávezeti az adatra, és kiküldi az az USB-alrendszernek (lásd a *2. listát*).
Amikor a kiíró ***urb*** a megfelelő adatokkal a `FILL_BULK_URB` függvény segítségével fel lett töltve, az ***urb*** befejezésjelző visszahívóját (`completion callback`) úgy állítjuk be, hogy a saját `skel_write_bulk_callback` függvényünket hívja meg. Ez a függvény fog ugyanis meghívódni, amikor az USB-alrendszer az ***urb***-ot befejezi. A visszahívási függvény a megszakításkor hívódik meg, ezért figyelnünk kell, hogy ilyenkor ne túl sok számítást használjunk. A mi `skel_write_bulk_callback` megvalósításunk egyszerűen csak jelzi, hogy az ***urb*** befejezése sikeres volt-e vagy sem, majd kilép.
Az olvasás kicsit másképpen működik, mint az írás, ugyanis az eszközről a vezérlőhöz történő adatátvitelhez nincs szükségünk ***urb***-okra. Ehelyett meghívjuk az `usb_bulk_msg` függvényt, amelyet arra használhatunk, hogy az eszközről ***urb***-ok készítése nélkül küldjünk vagy fogadjunk adatot, és az ***urb*** befejezésjelző visszahívásait kezeljük. Meghívjuk az `usb_bulk_msg` függvényt egy átmeneti tárat adva meg neki, ahová az eszközről érkező adatokat helyezheti, valamint egy időtúllépési határértéket. Ha az időtúllépési érték anélkül jár le, hogy az eszközről adatot kapnánk, a függvény sikertelen és hibaüzenetet ad vissza (lásd a 3. listát; 23. CD Magazin/USB könyvtár).
Az `usb_bulk_msg` függvény nagyon hasznos lehet, ha az eszközzel egyszeri olvasásokat vagy írásokat akarunk végezni; ugyanakkor ha az eszközről folyamatosan kell írni vagy olvasni, ajánlott, hogy saját ***urb***-jainkat állítsuk föl és küldjük el őket az USB-alrendszernek.
Amikor a felhasználói program elereszti a fájlkezelőt, amit eddig az eszközzel való kapcsolattartásra használt, a vezérlő `release` függvénye hívódik meg. Ebben a függvényben a modul használatszámlálót a `MOD_DEC_USE_COUNT` használatával (a korábbi `MOD_INC_USE_COUNT` hívásunk párjaként) csökkentjük. Azt is meghatározzuk, hogy jelenleg van-e valamilyen más program, amely éppen az eszközhöz beszél (az eszközt egy időben több program is megnyithatta). Ha a felhasználó az eszköz utolsó használója, az összes esetleg még futó írást leállítjuk. Ezt a következőképpen tehetjük meg:

```
/* cs kkents k az eszk zh z tartoz
   hasznÆlatszÆmlÆl t */
--skel->open_count;
if (skel->open_count <= 0) {
   /* `ll tsunk le minden  rÆst, ami esetleg
      mØg megy */
   usb_unlink_urb (skel->write_urb);
   skel->open_count = 0;
}
```

```
/* cs kkents k a modulhoz tartoz
   hasznÆlatszÆmlÆl t */
MOD_DEC_USE_COUNT;
```

Az egyik legnehezebb feladat, amit az USB-vezérlőnek könnyedén kezelnie kell, hogy az USB-eszközt a rendszerből bármikor eltávolíthatják, még akkor is, ha a program éppen kapcsolatban áll vele. Ehhez az kell, hogy a pillanatnyi olvasási és írási műveleteket le lehessen állítani, és a felhasználói programot értesíthessük, hogy az eszköz már nincs többé jelen (lásd a 4. listát; 23. CD Magazin/USB könyvtár).
Ha a program rendelkezik az eszközhöz nyitott kezelővel, az `usb_device` szerkezetet a helyi szerkezetben egyszerűen csak `null`-ra állítjuk, mivelhogy már nem létezik. Minden írás, olvasás, elereszt és más olyan szolgáltatás, amely az eszköz jelenlétét feltételezi, először ellenőrzi, hogy az `usb_device` szerkezet jelen van-e még. Amennyiben nincs, elereszti az eltűnt eszközt, és a felhasználói programnak egy `-ENODEV` hibaüzenetet ad vissza. Ha az elereszt függvénye hívódik meg, előbb meghatározza, hogy valóban megszűnt-e már az `usb_device` szerkezet, és ha nem, akkor szokásos módon törli a `skel_disconnect` függvényt, éppúgy, mintha az eszközön nem lennének nyitott fájlok (lásd az 5. listát; 23. CD Magazin/USB könyvtár).
Ez az USB-vezérlőváz a megszakításokra vagy az eszközre küldött, illetve onnan fogadott `isochronous` adatokra semmilyen példát nem tartalmaz. A megszakításadatokat csaknem pontosan úgy kell küldeni, mint a tömegadatokat, néhány kisebb eltérés van közöttük. Az `isochronous` adat másképpen működik: az eszközről állandó adatfolyamokat küld vagy fogad. Az audio- és videokamera-vezérlők nagyon jó példák az `isochronous` adatkezelésre, és hasznosak lehetnek, ha nekünk is ilyesmit kell tennünk.
Linuxos USB-eszközvezérlő írása nem nehéz feladat, amint azt az USB-vezérlőváz is jól példázza. Ez a vezérlő más jelenlegi USB-vezérlőkkel kombinálva már elég példát nyújt, hogy egy kezdő alkotó rövid idő alatt működő vezérlőt készíthessen.
A linux-usb-devel levelezőlista levéltára ugyancsak rengeteg hasznos adatot tartalmaz.

Greg Kroah-Hartman
a Linux-rendszermag USB-fejlesztőinek egyike. Ingyenes programját sokkal több ember használja, mint amennyit zárt forrású projektek fejlesztéséért valaha is fizettek neki.

Kapcsolódó címek

Linux Hotplug Project ➲ http://linux-hotplug.sourceforge.net
The Linux USB Project ➲ http://www.linux-usb.org
A Linux alatt használható USB-eszközök listája
➲ http://www.qbik.ch/usb/devices
A linux-usb-devel levelezőlista levéltára
➲ http://marc.theaimsgroup.com/?l=linux-usb-devel
Programming Guide for Linux USB Device Drivers
➲ http://usb.cs.tum.edu/usbdoc
USB honlap ➲ http://www.usb.org
Linux alatt működő USB-csatolós lapolvasók listája
➲ http://www.buzzard.org.uk/jonathan/scanners-usb.html
USBMan-Linux USB Guide
➲ http://www.usbman.com/linuxusb.htm

Szavakon innen, Wordön túl

Hogyan kezeljünk MS Word-állományokat Linux alatt, és miképpen állíthatunk elő rendszerfüggetlen dokumentumokat?

Azt állítják, hogy a Microsoft Word-állományok bármilyen szövegszerkesztőben megtekinthetők. Vélhetőleg ez lehet az oka annak a tévhitnek is, hogy számosan ragaszkodnak az egészen egyszerű szövegek csatolt állományként történő elküldéséhez. Az alábbi idézet a Microsoft weblapjáról származik: „Ezek a letölthető dokumentumok Microsoft Word 6.0 formátumban készültek. Kibontásukat követően tartalmuk bármilyen szövegszerkesztőben megtekinthető, beleértve a Microsoft Word, a Wordpad és a Microsoft Word Viewer programokat." A mindennapok folyamán vajon milyen gyakran kapunk Word-formátumú csatolt állományt tartalmazó elektronikus üzenetet, mert a küldő habozás nélkül feltételezi, hogy Microsoft Wordöt használunk? Vajon a felmerült-e benne egyáltalán a halvány kétely, hogy nem mindenki ezt a programcsaládot alkalmazza?

A fent idézett feltételezés nemcsak azok számára veszélyes, akik a csatolt állományok megnyitására éppen Wordöt használnak – e fájlok ugyanis esetenként vírus(ok)at tartalmaz(hat)nak –, hanem mostanra már a nem Microsoft-termékeket használók táborára nézve is kétségtelenül kellemetlenné váltak. A Microsoft Worddel rendelkező felhasználóknak célszerű a program legfrissebb változatát birtokolniuk (sőt, a programfrissítésért még fizetniük is kell), mert előfordulhat, hogy a programváltozatok közötti eltérések miatt a Word-programok egymás dokumentumait nem mindig képesek hibátlanul olvasni (például a 2000-es dokumentumait a 95-ös csak bővítményekkel tudja olvasni – a szerk.).

Írásunkban a Word-dokumentumok kezelési lehetőségeinek feltárásával az irodai munka megkönnyítését kíséreljük meg. Tekintettel arra, hogy a jelenlegi helyzetet a szabványostól eltérő formátumú dokumentumok visszautasítási lehetőségének hiánya jellemzi (erre a cikk későbbi részében még visszatérünk), tudomásom szerint a Linuxban nem létezik teljes értékű módszer az MS Word-dokumentumok kezelésére. Amint fentebb már említettük, a Word-programok néha még egymással sem „értenek szót". A legtöbb állomány megnyitására azonban számos módszer fellelhető akár a formátum megtartása mellett is: léteznek olyan teljes értékű szövegszerkesztő programok, amelyek nagyon hasonlítanak a Microsoft Wordre (ezekkel annak fájljai megnyithatók), valamint jó néhány állományátalakító is akad. Továbbá több olyan különleges eszköz is elérhető, amelyek segítségével a .DOC állományokból kinyerhető a tartalom. Mindig az adott helyzetnek megfelelően választhatunk a megoldások közül.

Nagyszabású Word-változatok

Ha munkánk során számos szövegfeldolgozással kapcsolatos feladatunk adódik, és gyakran kell dokumentumokat küldenünk vagy fogadnunk, célszerű teljes irodai csomagot telepítenünk. Az irodai programcsomagok többek között szövegszerkesztőt is tartalmaznak, amelynek segítségével többféle MS Word-formátumot olvashatunk (ilyen állományokba esetleg írhatunk is), emellett mindegyik saját állományformátummal rendelkezik. A Linux-rendszerben általánosan hozzáférhető irodai programcsomagok a következők: ApplixWare Office, Corel WordPerfect Office 2000, KOffice és StarOffice, illetve OpenOffice.

ApplixWare Office

A fentiek közül az ApplixWare az egyetlen olyan termék, amely egyáltalán nem használható ingyenesen. E cikk megírásához a programot az Applix bocsátotta rendelkezésemre, amelynek kereskedelmi ára jelenleg 99 dollár. Kaptam néhány színes dobozt, amelyek az ApplixWare Office-t, ApplixWare Wordsöt (önálló program) és az ApplixWare Spreadsheetset (önálló program) tartalmazták.

A vásárlók az irodai csomaggal együtt egy tetszetős kézikönyvhöz is hozzájutnak, amit igazán azonban csak a telepítés után tudnak használni.

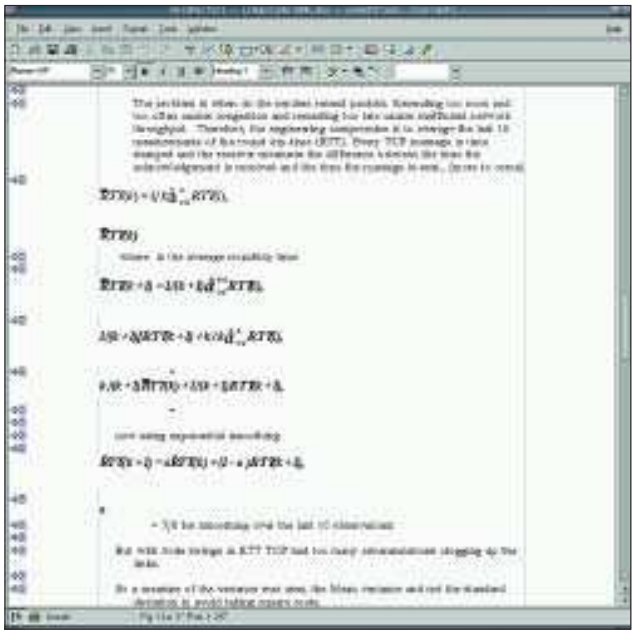
Égve a vágytól, hogy az új programot mihamarabb használatba vegyem, az ApplixWare Words telepítése során igyekeztem pontosan követni az útmutatóban leírtakat. A Debian-rendszer indításakor azonban figyelmen kívül hagytam, hogy az RPM-csomagok telepítését leíró utasítások közül egy lépést kihagytak, és gyanútlanul futtattam a telepítőállományt.

Eleinte úgy tűnt, minden rendben, de a vámnál meg kellett fizetnem, amit addig a réven nyertem. Úgy tűnik, hibásak a telepítő héjprogramok, mivel lehetetlenné tették számomra a program telepítését. A program által javasolt módon a programot a */opt/applix* könyvtárba szerettem volna telepíteni, de a hibanapló megmutatta, hogy a */optapplix* könyvtárba próbálta meg telepíteni, annak ellenére, hogy a /opt/applix könyvtárat hozta létre. Ez csupán apró kellemetlenség maradt volna, ha átszerkeszthettem volna a telepítő héjprogramot, de kereskedelmi termék lévén a program zárt kódú – tehát semmit sem tehettem. Megnéztem, mire jutok a /opt/applixszal, ezenkívül még néhány további trükkel is megpróbálkoztam, mindhiába. A kielégítetlen függőségek miatt – a Debian-rendszeren úgy látszott, mintha a legalapvetőbb csomagokat sem telepítettem volna – még az RPM-telepítés is csődöt mondott, vagyis annyira rosszul sikerült, hogy végső elkeseredésemben az RPM-ekből az `alien` segítségével .DEB-csomagokat készítettem :

```
mkdir /tmp/applix
cp cdrom/RPMS/*.rpm /tmp/applix
cd /tmp/applix
alien *.rpm

dpkg -i-force-overwrite *.deb
```

Utóbb úgy tűnt, ez telepíti ugyan a csomagokat, de az alkalmazás futtatása hibákhoz vezetett. Végül feladtam a kísérletezést, és levelet küldtem az Applixnak, hogy megoldási módról tudakozódjam. Miközben a közvélemény ezt a terméket megbízhatónak tartja – a programkészlet előnye, hogy saját



1. kép WordPerfect

állományformátuma a cég honlapjáról ingyenesen beszerezhető egyéb jellemzőkkel kiegészített közönséges ASCII-kódú szövegállomány, amely megkönnyíti a szűrők készítését –, a fent ismertetett hibák miatt bennem a fiaskó érzetét kelti.
A szép kivitelű kézikönyv és a jó minőségű program birtoklásából származó előnyt a program zárt forrásúsága megszünteti. Tehát nem vághatok neki magam a hibák kijavításának.

Corel WordPerfect Office 2000

A Corel cég, amely a saját Linux-változatáról és a CorelDRAW elnevezésű termékéről ismert, nagyon hatékony irodai programkészletet fejlesztett. A Corel WordPerfect Office 2000 programcsomag a közismert WordPerfect szövegszerkesztőt tartalmazza, ami minden olyan szolgáltatást nyújtani képes, amit az ember az efféle eszközöktől csak elvárhat.
Sokak szerint a Corel WordPerfect Office 2000 a rangsorban az MS Wordöt is megelőzi. Windows- és Linux-felületre egyaránt megvásárolható, de furcsamód nem létezik Machez, annak ellenére sem, hogy a Corel más termékeit erre a felületre is kínálja. Amennyiben irodánk teljes ügyvitelét a WordPerfect Office 2000-rel szeretnénk intézni, számlánkról szép „kis" összeget utalhatunk érte, ugyanakkor kizárólag személyes használatra bárki letöltheti a Világhálóról. A program telepítése és használata könnyed nyári fuvallatnak tűnt: minden Word-dokumentumot gond nélkül megnyitott, amit csak merevlemezemen találtam, mi több, még a matematikai képletek helyes ábrázolására is képes volt.

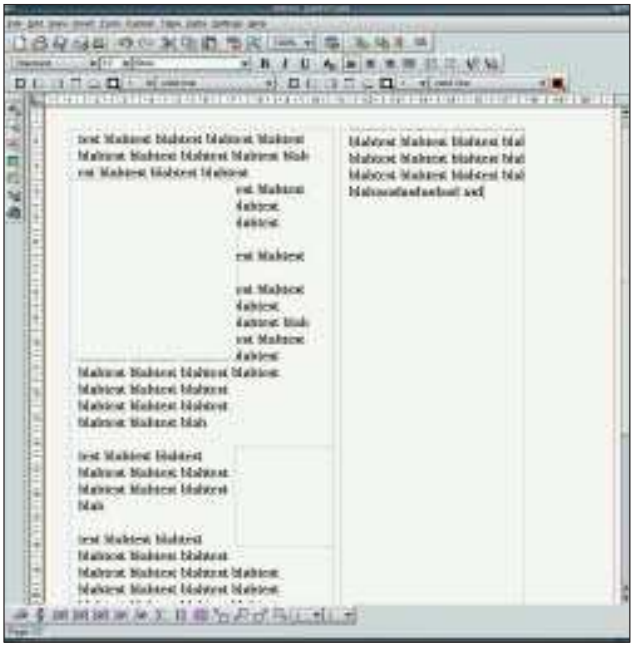
KOffice

A KOffice programcsomag tavaly októberben jelent meg a KDE 2.0-val együtt (akkor még csak bétaállapotban), és a KDE felhasználóbarát munkatársai fejlesztették. A KWordöt lenyűgöző programnak találtam: a többi KDE-alkalmazással együtt szerves egészet alkot. Microsoft Word-dokumentumaim többségét szépen be is hozta.
A bajok akkor kezdődtek, amikor matematikai képleteket tartalmazó dokumentumokat próbáltam vele megnyitni, mivel azonban ezekre ritkán van szükség, jó szívvel ajánlhatom ezt a programot.

Biztos vagyok benne, hogy a KOffice 1.1 megjelenésekor a KWord már minden igénynek képes lesz megfelelni. Ez az irodai programkészlet természetesen a GPL-szerződés hatálya alá tartozik, s ennek megfelelően kedvenc tükörkiszolgálónkról ingyenesen letölthető. A Debian `apt-get` telepítője gondoskodott a csomagfüggőségek ellenőrzéséről, azóta viszont a KOffice már a KDE 2.0-ra (2.2.1-re) és a Qt 2.2-re támaszkodik, így könnyen megeshet, hogy előzetesen a csomagok frissítésével kell foglalatoskodni, s csak ezt követően vehetjük használatba.

StarOffice, illetve OpenOffice

Már eltelt valamennyi idő azóta, hogy a Sun Microsystems felvásárolta a StarOffice-t, azt az irodai programcsaládot, amely több operációs rendszerben is képes dolgozni . A StarOffice az irodai programcsomagok közül az elsők egyike, amely képes felvenni a versenyt a Microsoft cég Office-termékével.
Annak ellenére, hogy a Sun a StarOffice-t ingyenes letöltésre mindig is hozzáférhetővé tette, a Nyílt Forráskód Közössége



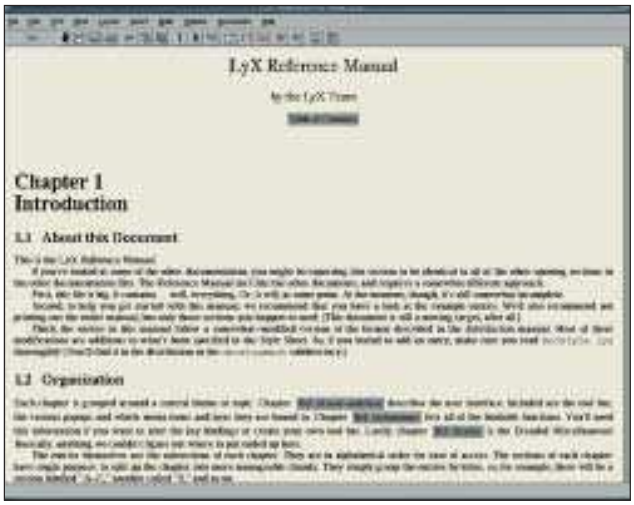
2. kép KWord

számára a programkészlet forráskódjának felszabadítását csak a közelmúltban jelentették be – íme a GPL-esített programok újabb példánya.
A StarOffice-ba, illetve az OpenOffice-ba roppant hatékony szövegszerkesztőt építettek, amely a különféle Word-dokumentumok legtöbbjének megnyitására, sőt még .DOC-típusú állományok írására is alkalmas. Hátulütője is akad: ez aztán a memóriafaló alkalmazás!
A teljes telepítéshez nemcsak komoly lemezméret szükséges, hanem az is sok időt vesz igénybe, amíg a rendszer az összes alkalmazást elindítja. Abban az esetben, ha lassú géppel rendelkezünk, nem valószínű, hogy ez lesz a legcélszerűbb választás. Másrészről ha gépünkben elegendő a tárkapacitás, biztos vagyok benne, hogy a StarOffice, illetve az OpenOffice az összes szövegfeldolgozással kapcsolatos igényünknek meg fog felelni.

Apró Word-változatok

Valamennyi fent említett termék teljes irodai programcsomag, mindegyikük nagyméretű és összetett alkalmazásokból áll,

amelyeket olyan felhasználók igényeihez szabtak, akik nagy mennyiségű szövegszerkesztést végeznek, emellett táblázatkezeléshez, bemutatókészítéshez megfelelő alkalmazásokra szintén szükségük van.
Azoknak, akik a szövegszerkesztőt csak a házmesternek címzett panaszlevelek írására használják, könnyedebb megoldások is léteznek. A legközkeletűbb könnyű fajsúlyú szövegszerkesztő program az Abiword. Az Abiword, amelyet „teljes körű szolgáltatást nyújtó, de a szikárságát megtartó program"-nak terveztek, úgy tűnik, nagyon is megfelel a kitűzött céloknak: gyors, több felület számára hozzáférhető, továbbá ingyenes és szabad felhasználású (az utóbbi szavakat abban az értelemben használva, ahogyan a sör és a beszéd, illetve a szólásszabadság kapcsán szoktuk érteni őket). Ezenkívül az Abiwordöt folyamatosan fejlesztik. Be kell vallanom, hogy némely dokumentummal fuldoklik, vagy az eredeti formátum megőrzése nélkül nyitja meg. Különösen az ejtheti zavarba az Abiwordöt, ahogyan az MS Word kezeli a táblázatokat. Komoly gondot jelentenek néha az Abiword számára a nyelvek egyedi karakterei (ő, ű, betűk stb).



3. kép LyX

A másik igazán parányi és pehelysúlyú szövegszerkesztő a pathetic writer (`pw`), amely a Siag Office Suite irodai programcsomag része. Azért itt említem a `pw`-t és nem a teljes körű irodai szolgáltatást nyújtó programkészletek között, mert meglehetősen karcsúnak tűnik. A `pw` ugyan nem hajlandó megnyitni a Microsoft .DOC-állományokat, de boldogan elvégzi a mindennapi szövegszerkesztési feladatokat, és a leggyakoribb adatformátumokkal is megbirkózik. A Siag Office az Abiwordhöz hasonlóan, a GPL-szerződésnek megfelelően jelent meg és ingyenesen letölthető.

A szokásostól eltérő Word-változatok

A fentebb említett alkalmazások mindegyikének eltérőek a futtatási környezettel szemben támasztott követelményei: némelyik már előzetesen telepített programkönyvtárak meglétét feltételezi (ilyen a KWord), másokra az erőforráséhség jellemző (StarOffice, illetve OpenOffice), egyéb programok pedig költségesek, illetve zárt forrásúak. Abban azonban egységesek, hogy mindegyik megpróbál valamilyen stílust vagy éppen bizonyos dokumentumformázási módot megőrizni.
Miközben mindez bizonyára alapvető és célravezető, úgy találtam, hogy a szövegszerkesztő programok nem túl hasznosak, és tulajdonképpen mindegy, éppen melyikről is van szó. Az esetek kilencven százalékában, amikor egy meggondolatlan személy .DOC formátumú állományt küld nekem, a benne szereplő tartalom az eredeti állomány méretének töredékét elfoglaló szövegállományban is közölhető lett volna.
Nos, beszéljünk az üzletről és nézzünk utána, hogyan is nyerhetjük ki a szükséges tartalmat az egyes állománytípusokból. Létezik néhány említésre méltó eszköz, amelyek szépsége abban rejlik, hogy használatukhoz még X-felületre sincs szükségünk, minthogy kivétel nélkül parancssori eszközök.

Az antiword

Az `antiword` bemenő adatként Word-dokumentumot használ, amelyből kivonatot készít, és azt közönséges ASCII-karakterekből álló vagy PostScript-szöveggé alakítja át. Az állomány tartalmának gyors megtekintéséhez csővezetékben például a `less` parancsot használhatjuk:

```
antiword ORIASI.DOC | less
```

Abban az esetben, ha az állomány tartalmát nyomtatva szeretnénk látni:

```
antiword -p letter ORIASI.DOC | lpr
```

Az `antiword`-öt annyira hasznosnak találtam, hogy a korábbi MS Word-állományok kezelését leíró ***mailcap***-bejegyzésemet (amellyel előtte az Abiwordöt indítottam el) az alábbi sorral cseréltem le:

```
application / msword ; antiword % s | vim -
```

A fenti sor lehetővé teszi a számomra, hogy a levelezőből (`mutt`) átküldjem a csatolt Word-állományokat, és minthogy a program kimenetét a kedvenc szövegszerkesztőmhöz irányítom, a szöveget akár módosíthatom és másik állományba menthetem. Meg kell jegyezni, hogy a ***mailcap***-be tett fent említett bejegyzés hatására a .DOC állományok megjelenítésére valamennyi, az állomány által meghívott alkalmazás az `antiword`-öt és a `vim`-et fogja használni. Amennyiben grafikus böngészőprogrammal dolgozunk – például a Netscape-pel –, elképzelhető, hogy más szerkesztőprogramot szeretnénk használni, vagy a `vim`-nél a grafikus megjelenítés bekapcsolására a `-g` kapcsolót kell alkalmaznunk.
Ha a legpuritánabb megoldásra szorítkozunk, megállapíthatjuk, hogy a .DOC-állományokból pusztán a szöveges tartalom kivonására gyakran elegendő a parancsok – amelyek a ***binutils*** csomagban találhatók – használata.
Az `antiword`-nek a sorparancsokkal szemben az az óriási előnye is megvan, hogy a szöveg mellett a képeket is elő tudja csalogatni a dokumentumból. A program által biztosított különböző lehetőségek, valamint a képeknek a Word-állományból való kiemelésének módja felől az `antiword` honlapján, a ➲ http://www.winfield.demon.nl/index.html címen tájékozódhatunk.

wv

A korábban mindenki által mswordview néven ismert alkalmazás már régóta mindenki számára hozzáférhető. Amikor néhány éve először telepítettem a RedHat 5.2 rendszert, a Netscape böngésző az mswordview-t használta a .DOC állományok kezelésére, vagyis megbízhatóan alakította át őket HTML-állományokká. Emlékezzünk rá, hogy jelen esetben

nem a Wordviewról, a Microsoft egyik termékéről van szó. A névegyezés a program szerzőjét a program nevének megváltoztatására sarkallta. Noha minden bizonnyal elismerésre méltónak tartjuk, hogy a böngészőprogram valamilyen másik program segítségével a Word-állományokat HTML-formátumúvá alakítja, mégsem minden esetben ez a legmegfelelőbb kimeneti állományformátum. Pontosan emiatt a WV programba a formátumok igen gazdag választékát építették be, így az ASCII-szöveg, HTML-, LaTeX-, PostScript- és PDF-állománytípusokat is. A WV a GPL-szerződés feltételeinek megfelelően jelent meg és ingyenesen letölthető.

Word-programok nélküli szövegszerkesztés

Az eddigiek során áttekintettük, hogyan juthatunk a Word-dokumentumok tartalmához, valamint azokat a lehetőségeket, amelyekkel olyan dokumentumokat szerkeszthetünk, amelyeket a windowsos világban valószínűleg Worddel készítettünk volna el. Nincs azonban mindig szövegszerkesztőre szükség, megeshet, hogy az esetek döntő többségében éppenséggel haszontalan.

4. kép xpdf

Így például akkor is, ha a felhasználók többsége a főnökének készítendő jelentést az alábbi séma szerint készíti el: gépel valamit; az egérrel kijelöl egy szövegrészt; az egérrel rámutat valamire; kattint egyet itt, félkövérrel szed betűket ott; leüti néhányszor az ENTER billentyűt, majd a szóköz billentyűt; mérlegeli, hogy tetszik-e, amit addig írt; párszor leüti az ENTER billentyűt; ezután megint rámutat valamire az egérrel, dőlt betűvel szed ezt-azt, majd újra kattint az egérrel; végül újrakezdi az egészet. Meg vagyok győződve, hogy nem ez az a mód, ahogyan a nagy tudású szövegszerkesztő programokat használni kell, de nézzünk szembe a tényekkel: a felhasználók többsége mégis pontosan így használja – végtére is ők azok, akik számára ezeket a „felhasználóbarát” alkalmazásokat tervezték. A szükséges erőfeszítés, amely a tartalomjegyzék, az irodalomjegyzék vagy a kereszthivatkozások beviteléhez kell, mindenki számára elképzelhető.

Az eredmény a kész dokumentum, amelynek megalkotása hosszabb időt vesz igénybe, bár olyan a külleme, amilyennek lennie kell, általában adott rendszerben és a szövegszerkesztő program meghatározott változataiban. Az ehhez hasonló rossz gyakorlat elkerüléséhez néhány olyan újabb módszert fogunk megvizsgálni, amelyekkel felületfüggetlen dokumentumokat készíthetünk.

Az ASCII: az örök klasszikus

Amint már több alkalommal is említettem, a dokumentumokba foglalt tartalom általában szöveg. Sok esetben a fantáziadús formázás ugyan szép, mégsem kötelező erejű. A dokumentum írójának fő érdeke az információ közlése. Az egyszerű ASCII kódkészletű szöveg tökéletesen alkalmas, hogy információt továbbítson egyik embertől a másikig: ez az oka annak, amiért a levelezés során még mindig gyakran csupán szöveghordozó közegként használják. Az elektronikus levelezésben használatos HTML-formátum már semmit sem tesz hozzá a tartalomhoz. Az ASCII-kódkészletű szöveget bárhol, bármilyen szövegszerkesztővel el lehet olvasni, nem feltétlenül MS Word szövegszerkesztőt értve ezalatt. A szöveg egyértelmű tagolásával, a bekezdésekkel és a kötőjelekből álló elválasztó sorok kialakításával, valamint a Useneten már megszokottá vált **félkövér**, ***dőltbetű***, és egyéb formázások használatával könnyedén írhatunk jól olvasható, könnyen érthető, és ami még fontosabb: hordozható formátumú dokumentumokat.

A LyX és LaTeX

Igaz ugyan, hogy a legtöbb esetben az ASCII megfelelő választás, az is kétségbevonhatatlan, hogy igény merülhet fel több szövegformázás használatára. A megoldás: a LyX, a LaTeX grafikus felülete.

A LaTeX elképesztő képességű betűszedő program, amelyet a TeX alapján fejlesztettek. Bemeneteként .TEX állományt dolgoz fel és eredményül .DVI állományt készít. A felületek bőséges választéka számára hozzáférhető; a LaTeX-szel készített dokumentumok hihetetlenül jó benyomást keltenek. A fent említettek ellenére mégsem kell lemondani kedvenc szövegszerkesztő programunk használatáról, hiszen a LaTeX parancssori eszköz.

A LaTeX használata közben a szerző a küllem helyett nyugodtan a tartalomra összpontosíthat, a lap nyomtatási képéről ugyanis a betűszedő motor gondoskodik. A .TEX állományban szerepel néhány címke – a HTML-formátumra emlékeztetnek –, amelyek a szöveg megjelenítési módját befolyásolják.

A LaTeX a szokványos szövegszerkesztőkhöz képest a dokumentumkészítés gyökeresen eltérő módját kínálja: itt nincs egérrel való rámutatás, kattintgatás, kijelölés és hasonló műveletek. Az is előfordulhat, hogy olyasvalakinek, aki eddig grafikus felhasználói felülethez szokott, ijesztőnek hat.

Nos, ez az a pont, ahol a LyX-et készítő szakemberek a segítségünkre sietnek, ugyanis a LaTeX-hez grafikus felületet fejlesztettek – ez képessé teszi a tapasztalatlan felhasználót, hogy a TeX által biztosított előnyöket anélkül használhassa ki, hogy a programmal való ismerkedést a legelejétől kellene kezdenie. Első pillantásra úgy tűnhet, a LyX meglehetősen hasonlít megszokott szövegszerkesztőnkhöz, de ha figyelemmel kísérjük az oktatóanyagot, hamarosan feltűnik a különbség, és az is felismerhető, hogyan növelhető a hatékonyság a képi

Makefile ehhez a dokumentumhoz

```
TARGET          = words

LATEX           = latex
DVIPS           = dvips -o
PS2PDF          = ps2pdf
PDFTOTEXT       = pdftotext

.SUFFIXES: .tex .dvi .ps .pdf .txt

all:        $(TARGET).dvi

.tex.dvi:
        $(LATEX) $<
        $(LATEX) $<

ps:         $(TARGET).dvi $(TARGET).ps

.dvi.ps:
        $(DVIPS) $@ $<

pdf: $(TARGET).dvi $(TARGET).ps
$(TARGET).pdf

txt: $(TARGET).dvi $(TARGET).ps
$(TARGET).pdf
$(TARGET).txt

html:latex2html $(TARGET).tex
.ps.pdf:
        $(PS2PDF) $< $@

.pdf.txt:
        $(PDFTOTEXT) $< $@

clean:
        rm -f *.log *.aux *.dvi *.ps *.pdf
*.toc *.txt
        rm -fr $(TARGET)/
```

ábrázolás helyett a munkára és a dokumentum tartalmára való összpontosítással.
Ha teendőink elvégzése céljából gyakran kapcsolódunk távolról számítógépekhez, nem minden esetben lehet az X-felületet továbbítani. Ekkor érkezik el az a pillanat, amikor megtanuljuk értékelni a parancssort – rájöhetünk, hogy minden, amire csak valaha is szükségünk lehet, már a kezünk ügyében van. A kedvenc szövegszerkesztőnkkel – a `vim`-et használom – és a LaTeX-szel a gépünkhöz kapcsolódó egyetlen terminálról az összes szövegszerkesztési feladatot könnyűszerrel elvégezhetjük.
A LyX és a LaTeX használatának további előnye, hogy állományainkat olyan felületfüggetlen formátumúvá alakíthatjuk, mint amilyen a PostScript vagy a PDF. Ha pedig a `make` és a LaTeX erőit egyesítjük, mindezt néhány parancs segítségével megtehetjük. Vegyük például a képen látható dokumentumot: a bemeneti állományt gyönyörű PDF-fé alakítottam, egyszerűen a `make PDF` parancs segítségével (*4. kép*, 51. oldal).
Noha maga a `Makefile` állomány egyszerű, mégis lehetővé teszi, hogy a parancssorban a sokféle állományformátum közül kiválasszam a számomra megfelelőt: `ps2pdf` (PostScriptből PDF-állományt állít elő) vagy `latex2html`.
Végül ejtsünk szót arról, hogy a LaTeX kiterjeszthető, vagyis olyan saját stílusokat hozhatunk létre, amelyek különböző eredményeket adnak a bemeneti dokumentum fajtájától függően. Az a legjobb, hogy valaki ezt már meg is tette Comprehensive Tex Archive Network – CTAN („Átfogó TeX Archívumhálózat") néven, amely a Perl CPAN-jához hasonlót.

Összefoglalás

Bármelyik szövegszerkesztési mód mellett döntünk is, hangsúlyoznom kell a hordozható dokumentumformátumban történő információtovábbítás fontosságát. Tegyen kísérletet arra, hogy a most megszerzett ismereteket továbbadja azok számára, akik küldeményeik „megjavíttatását" követelik, amikor nem tudják megnyitni őket, vagy ha valamilyen formázási elem elvész.
Be kellett látnom, hogyha valaki barátságosan elmagyarázza, hogyan nyithatok meg PDF- vagy PS-állományokat szinte bármilyen felületen, az ilyen mester tanításait más is szívesen fogadja, kivéve a windowsos világ legkonokabb polgárait. Biztos vagyok benne, hogy még a hétköznapi feladatokra is nagyságrendekkel jobban használhatónak fogják találni a LaTeX-et, nem is szólva a profi küllemű dokumentumokról.
A LaTeX nyújtotta előnyök kihasználásához a Világhálón számos hasznos dokumentáció található. A vele most ismerkedő felhasználók számára a legfontosabb olvasmány valószínűleg a „The Not So Short Introduction to LaTeX2" („Nem is olyan rövid bevezetés a LaTeX2-be") című áttekintés lehet; amit a Comprehensive Tex Archive Network ➲ http://www.ctan.org honlapról lehet beszerezni.
Az oktatóanyag tanulmányozása és a példák kipróbálása után már könnyebb lesz áttérni a használatára.
A szavamat adom rá.

Jan Schaumann
(jschauma@netmeister.org) Iserlohnban született, majd a németországi Altenában nevelkedett. Egyetemi tanulmányait két évig a német irodalom és a média, valamint az amerikanisztika területén folytatta Marburgban. 1998-ban New York Citybe költözött.

Kapcsolódó címek

AbiSource ➲ http://www.abisource.com
antiword ➲ http://www.winfield.demon.nl/index.html
Applixware Office
➲ http://www.vistasource.com/products/axware
Comprehensive Tex Archive Network
➲ http://www.ctan.org
Corel ➲ http://www.corel.com
Corel Linux
➲ http://www.linux.corel.com/products/linux_os/index.htm
Corel WordPerfect ➲ http://www.linux.corel.com/products/wpo2000_linux/index.htm
KOffice ➲ http://www.koffice.org
LyX ➲ http://www.lyx.org
OpenOffice ➲ http://www.openoffice.org
Siag Office ➲ http://siag.nu
StarOffice ➲ http://www.sun.com/products/staroffice
wv ➲ http://www.wvWare.com

PoV-Ray ismeretek (5. rész)

Az anyagmegadás utolsó simításaként ismerkedjünk meg a felület fényességének meghatározásával, hogy a későbbiek folyamán bonyolult felületi mintázatokat hozhassunk létre.

A Pov-Ray-ben az anyagmeghatározás utolsó lépése a felület fényességének meghatározása. Ezt a program szóhasználatával élve `finsh`-nek nevezzük, amely a felület fényességét és egyéb tulajdonságait befolyásolja. Használatával fényes és matt felületeket hozhatunk létre; fénytörést utánozhatunk; meghatározhatjuk, mi történjen, amikor a fény áthalad az átlátszó részeken; valamint megszabhatjuk a fény szóródását a felületen. Amikor a fénysugár a vékony felületekről visszaverődik, a felületen látható fénytöréseket ugyancsak a `finsh` hozza létre.

Az anyag fent leírt viselkedését több különféle érték segítségével alakíthatjuk. Ezek közül a következők a fontosabbak: `ambient`, `diffuse`, `phong`, `phong_size`, `specular`, `metallic`, `reflection`, `refraction`, `fade_power`, `fade_distance`, `interior` (korábbi változatokban `ior`) és `irid`. Az alábbiakban ezeket az értékeket taglalom bővebben.

1. lista Az izzólámpa megvalósítása

```
sphere { <0,0,0>, 1
       pigment { White }
       finish {
              ambient 1
              diffuse 0
       }
       scale 2*y
}
```

2. lista Példa a „phong” és a „phong_size” használatára

```
spehere { <0,0,0>, 1
       pigment { Gray50 }
       finish {
              ambient 0.2
              diffuse 0.6
              phong 0.75
              phong_size 25
       }
}
```

Az `ambient` kulcsszó határozza meg azt a fénymennyiséget, amely a tárgyat nem közvetlenül a fényforrásokból éri, hanem a jelenet más részeiből verődik vissza. Ezt leginkább úgy érthetjük meg, ha egy olyan almát képzelünk el, amit fehér asztalon balról világítunk meg. Ekkor az alma jobb alsó területe szórt fényt kap, amit a PoV-Ray-ben az `ambient` értékkel határozhatnánk meg, amennyiben az almát modellezni szeretnénk. A tárgyat érő fény mennyiségének meghatározását szolgálja a `diffuse` kulcsszó is, ám ezzel a tárgyat érő közvetlen fénymennyiséget adjuk meg. Ha mindkét érték 0, tárgyunk csupán fekete foltként jelenik meg a képen, hiszen semmilyen fény nem éri. A PoV-Ray alapbeállításként a valóságban leggyakrabban előforduló értéket tartalmazza, amely egyben jó kiindulási alapot nyújthat a kísérletezéshez, viszont akadnak kivételes esetek, amikor ezeket meg kell változtatnunk. Ilyen lehet, ha tárgyunk teljesen átlátszó és kemény anyagból készült – például üveggömbből –, ráadásul a fénytörést is szemléltetni szeretnénk. Ilyenkor a legjobb hatást az `ambient` és a `diffuse` értékek kicsire állításával érhetjük el. Működését az alábbi példa segítségével szemléltethetjük:

3. lista A fénytörés szemléltetése

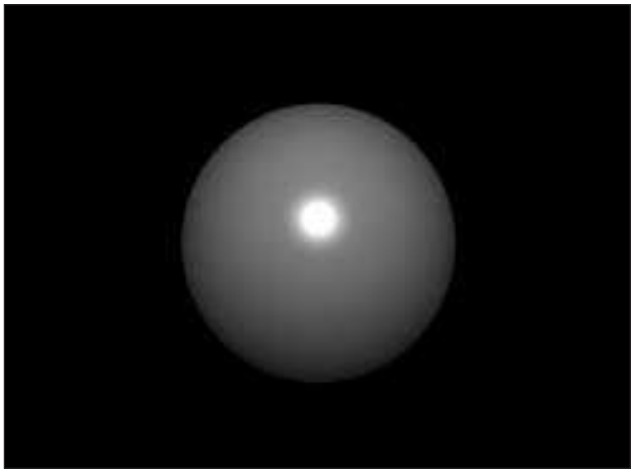
```
sphere { <0,0,0>, 1
       pigment { White filter 1 }
       finish {
              ambient 0
              diffuse 0
              reflection 0.25
              refraction 1
              ior 1.45
              specular 1
              roughness 0.001
       }
}
```

```
sphere { <0,0,0>, 1
       pigment { White filter 1 }
       finish {
              ambient 0
              diffuse 0
              reflection 0.25
              refraction 1
              ior 1.33
              specular 1
              roughness 0.001
       }
}
```

Természetesen a törésmutatóból (`ior`) látszik, hogy a fenti példa egy üveggömböt határoz meg a PoV-Raynek. Csak kismértékben látható, mert gyakorlatilag minden ráeső fényt visszaver vagy átenged, tehát csak a fénysugarak csillanása következtében látható. A különféle üvegfajtákra további példá-

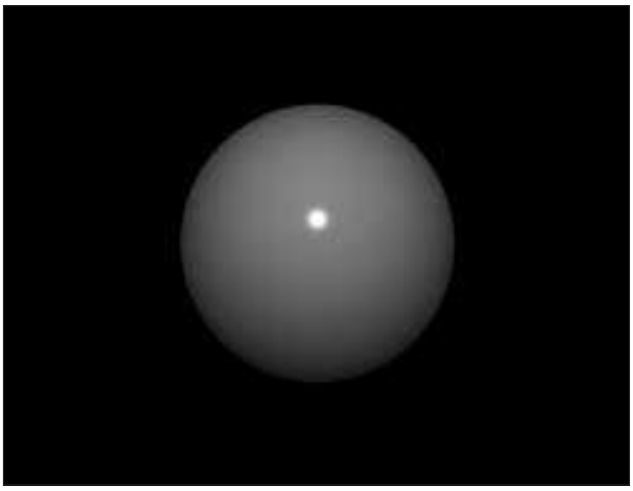
4. lista Példa a 12 érték használatára

```
sphere { <0,0,0>, 1
      pigment { White filter 1 }
      finish {
            ambient 0
            diffuse 0
            reflection 0.25
            refraction 1
            specular 1
            roughness 0.001
            caustic 1
            irid {
                  0.35
                  thickness 0.5
                  turbulence 0.5
            }
      }
      interior {
            ior 1.45
            fade_power 1
            fade_distance 5
      }
}
```

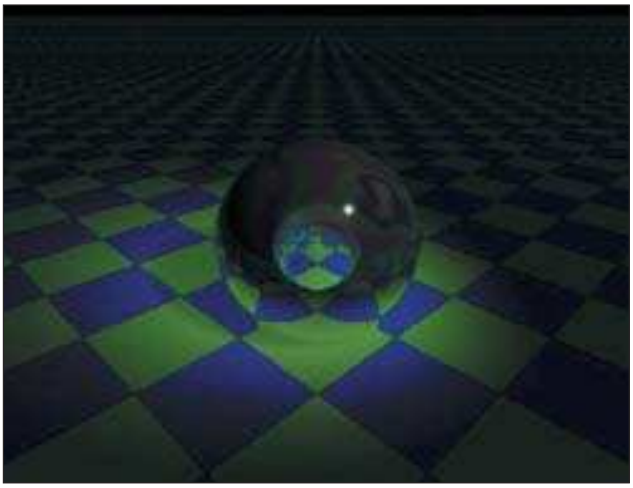


1. kép Gömb, a phong_size=25

kat a PoV-Ray-csomagban található ***GLASS.INC*** beilleszthető állományban láthatunk.
Ha egyszer olyan anyagra lesz szükségünk, ami a ráeső fénymennyiségtől függetlenül mindenképpen egyformán világosnak látszik, az `ambient` tulajdonság értékét állítsuk 1-re, míg a `diffuse` értékét 0-ra. Ennek hatására a PoV-Ray a közvetlen megvilágítást figyelmen kívül hagyja, csak a szórt fény hatását jeleníti meg. Példaképpen említeném az olyan tárgyakat, amelyek fényt bocsátanak ki: az izzólámpa vagy a fénycső. Kevésbé egyértelmű a fenti megoldás használata abban az esetben, amikor egységesen megvilágított égboltra van szükségünk.
Ilyen egyszerűen (lásd *1. listán*) hozhatunk létre szórt fénnyel világító izzót, de természetesen a PoV-Rayt arra is utasítanunk szükséges, hogy az izzónk valóban fényt bocsásson ki, amelyet egyrészt az árnyékvetés letiltásával (`no_shadow`), másrészt fényforrás elhelyezésével oldhatunk meg. Utóbbinak adjuk meg, hogy lámpaizzó formájú legyen. Ezt a fényforrásokkal és kamerákkal foglalkozó részben (lásd a ***Linuxvilág*** 6–7. számában a 92. oldalon) már taglalt `looks_like` kulcsszó használatával érhetjük el.
Az anyagok keménységét a fény által létrehozott csillanások méretével és szélei élességének vagy elmosódottságának mértékével érzékeltethetjük. Egy sima felületű üveggömbön nem olyan csillanásokat fogunk látni, mint egy sima felületű műanyag gömbön. A PoV-Rayben ezeket a tulajdonságokat a `phong` és a `phong_size` kulcsszavakkal határozhatjuk meg. A `phong` a csillanás fényességét adja meg, a `phong_size` pedig a méretét. Próbáljuk is ki a *2. listán* látható módon.



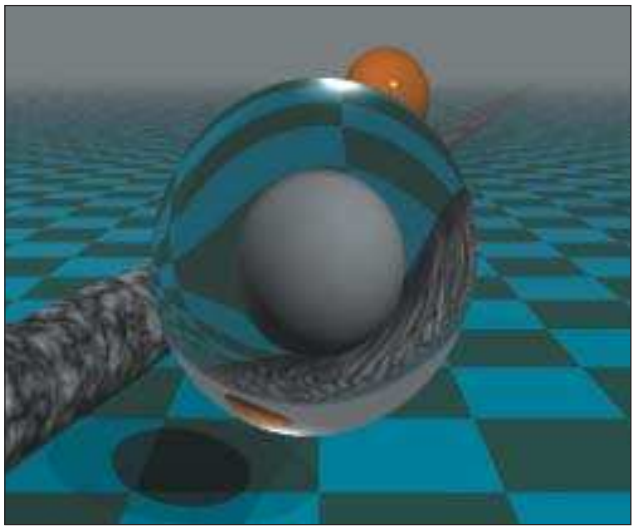
2. kép Ugyanaz a gömb 150-es phong_size értékkel



3. kép A végső gömbünk

A képet elkészítve észrevehetjük, hogy a csillanás mérete – ahhoz képest, amelyet a `phong_size` 150-es értéke mellett kapunk – elég nagy.
Az *1. kép* inkább műanyagjelleget tükröz, míg a *2. képen* a nagyobb érték kisebb méretű csillanást eredményezett, ezzel keltve a keményebb anyaghatást.
A felület keménységének meghatározására még a következő két kulcsszó is alkalmas, viszont a velük nyert felület általában valósághűbb és pontosabb képet eredményez. A két kulcsszó: a `specular` (fényességként fordíthatjuk), és a `roughness` (magyar megfelelője az érdesség lehetne). Itt érdemes megjegyezni, hogy a valóságnak e két érték meghatározási módja

felel meg jobban, hiszen itt nem fordított hatást eredményeznek, mint a `phong_size` esetében. Itt a kisebb `roughness` érték valóban élesebb, keményebb hatást ad, míg a másik meghatározási módnál a nagyobb érték jelentette a kisebb csillanó felületet.
A csillanáshoz szorosan kapcsolódik a tükröződés létrehozására alkalmas `reflection` kulcsszó, amelyet szintén a `finish` egyik értékeként használhatunk. Segítségével azt határozzuk meg, hogy egy adott anyag a ráeső fénymennyiség mekkora hányadát tükrözze vissza. Értéke 0–1-ig változtatható valós szám, amit százalékos értékként kell értelmeznünk. Itt újabb példát láthatunk arra az esetre, amikor a megfelelő tükröző hatás eléréséhez a `diffuse` értéket az alapértelmezettnél kisebbre kell állítanunk. Általában a magasabb tükröződési értékhez kisebb saját fénymennyiséget (`diffuse`) kell használnunk. Az így előállított anyag már csaknem teljesen fémesen tükrözi vissza a fényt és a környezetében lévő tárgyakat, de a tökéletesen fémes felülethez még a `metallic` kulcsszót is érdemes használnunk, amely – mint neve is mutatja – kifejezetten fémes felületek létrehozásához került a programba. Ennyi ismeret elegendő e két értékről, úgyis az alkalmazása során szerezhetünk jártasságot benne.



4. kép Fénytörés a nagyítóban

Akadnak azonban olyan tárgyak is, amelyeken áthaladva a fény iránya megváltozik. Ezek a tárgyak általában félig vagy teljesen átlátszók. Gondoljunk a lencsékre, a prizmákra, az üveggömbre, a vastag ablaküvegre, a vízre vagy a vízzel telt pohárra. Ezt a fénytani jelenséget nevezzük fénytörésnek, modellezésére a PoV-Ray is lehetőséget ad. Az itt használatos kulcsszó a `refraction` névre hallgat, amely azonban a PoV-Ray számára csak azt határozza meg, hogy figyelembe vegye-e az adott anyag fényáteresztő és -szűrő képességét, vagy sem. Emiatt a `refraction` értéke 0 vagy 1 lehet, vagy az ezeknek megfelelő `on` és `off` logikai értékek. A fényáteresztést és -szűrést a `pigment` utasításnál adjuk meg, miként azt az előző részben ismertettem.
Fénytörés esetén figyelembe kell venni az adott anyag törésmutatóját, amit az `ior` (Index Of Refraction = törésmutató) értékkel határozhatunk meg, ezt külön táblázat tartalmazza. Ezt a kulcsszót az újabb változatokban felváltotta `finish`-en belül használható `interior` rész, melynek használatára a *4. lista* mutat példát. Segítségképpen itt is megadok néhány adatot: a víz törésmutatója 1,33; az üvegé 1,45 körüli; a gyémánté pedig 1,75.
A fentiek szemléltetésére íme, egy könnyen megvalósítható példa, amely a *3. listán* látható.
Szót kell még ejtenünk arról az esetről, amikor az átlátszó tárgyban a fény erőssége csökken, vagyis a tárgy elnyeli a fényt. Ezt a PoV-Rayben szintén modellezhetjük, mégpedig a `fade_power` és a `fade_distance` értékek használatával. A `fade_distance` azt határozza meg, hogy a fény erőssége mekkora távolság megtétele után csökkenjen a felére, miután belépett a testbe, míg a `fade_power` azt, hogy a fényerősség milyen mértékben csökkenjen egységnyi idő alatt (a távolság függvényében). Itt szintén azokat az elveket kell alkalmaznunk, amelyeket a fényforrásoknál már ismertettem, nevezetesen az egyes érték jelenti a lineáris csökkenést, a kettes érték a négyzetes csökkenést, míg a hármas érték használatával a fényerősség csökkenésének mértéke a távolsággal köbös arányban áll majd.
A PoV-Ray legújabb változatában ennek létrehozására külön utasításblokkot kell alkalmaznunk, a pontos ismeretek elsajátításáért az utolsó példa áttanulmányozását javaslom.
Tekintettel arra, hogy a részecskerendszerek létrehozása a jelen cikk kereteibe nem fér bele, a következő alkalommal fogom ismertetni. Most azonban még lássuk az utolsó fontos értéket, amellyel az anyag-fény kölcsönhatást befolyásolhatjuk. Biztosan mindenki megfigyelte már azokat az érdekes színjátszó foltokat, amelyeket a fény játéka hoz létre egy olajfolton. Ezt a jelenséget irizálásnak vagy vékonyfilm-interferenciának nevezzük. A PoV-Rayben is lehetőségünk nyílik e jelenség képi megjelenítésére az `irid` kulcsszó használatával. Formája a következő:

```
irid {
       HAT`S_M RT KE
       thickness VAL S
       turbulence VAL S
}
```

Itt a `HAT`S_M RT KE` adja meg, hogy a tárgy színe mellett az irizálás mennyire érvényesüljön. Általában 0,1-től 0,5-ig az érték megfelelő. A `thickness` valós számmal határozzuk meg, hogy ezzel különleges hatással az anyag felülete menynyire legyen kitöltve, vagyis mennyi legyen látható az anyag eredeti színéből és mintázatából. A legszebb eredményt 0,25 és 1 közötti értékkel érhetjük el. Végül a jól ismert örvénylést befolyásoló `turbulence` kulcsszó következik, ami azonban kicsit különbözik a mintázatoknál használttól. Itt nem adhatunk meg `octave` és `lambda` értékeket, hanem csak egy százalékos értéket, amely az irizáló felület vastagságát más módon fogja befolyásolni, mint a `thickness` értéke.
Végül mindezeket összefoglalva lássunk egy utolsó példát, amelyben mind a tizenkét tárgyalt értéket alkalmazzuk (lásd 4. listán).
További kellemes 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.

Postfix-csemegék (4. rész).

Folytassuk problémamegoldással: hogyan használjunk két levelezőkiszolgálót felváltva?

Nemrégiben egy nem átlagos, de nem is túl kirívó nehézséggel fordultak hozzám. A levél írója két postafiókkal rendelkezett, ennek megfelelően két kiszolgálón keresztül kellett elküldenie a leveleket. Saját számítógépén Postfixet futtatott, hogy kétfelé menő leveleit ott tudja gyűjteni. Kérdése a következő volt: miként lehetne a legegyszerűbben megoldani, hogy hol az egyik, hol a másik kiszolgálót használja átjáróként a levelezéshez? A Postfix modularitása és futás közbeni módosíthatósága itt is a segítségünkre volt. Útmutatásaimat követve a levélíró a man-oldalak használatával a feladatot egyszerűen meg tudta oldani. A többi olvasó kedvéért azonban nézzük meg, milyen megoldást javasoltam – feltételezem, a levélíró is hasonló eredményre jutott –, és mely programokat használtam fel. Most azonban nem egy szokványos Postfix-leírás következik, mert parancsfájlírás és egy alkalmazás, a `sudo` beállítása is szerepel benne. Ezek a feladatmegoldás szerves részei, tehát kihagyhatatlanok, továbbá ismereteink bővítésére is alkalmasak, hiszen a `sudo` sokoldalú program, és a parancsfájlok írása is még hasznunkra válhat a későbbiek folyamán. Remélem, olvasóink örömmel fogadják ezt a kis „mixet”.

A hozzávalók

`postfix`

Szerintem a legjobb levelezőkiszolgáló, a feladat is ennek kapcsán merült fel, így a program megléte feltétele a megoldásnak.

`sudo`

Az alkalmazás segítségével a kiszemelt felhasználók rendszergazdai jogosultságot kaphatnak a kijelölt programok futtatására, vagy éppen jogosultságokat vonhatunk meg tőlük. Beállítási állománya a */etc/sudoers*, ezt azonban ne szerkesszük kézzel! Nem véletlenül mellékelik hozzá a `visudo` nevű segédprogramot, amelyben a `vi` felületét használva szerkeszthetjük a fájlt. A fájl lezárásakor és a kilépéskor ellenőrzi a formai követelmények betartását, ezzel könnyítve meg a munkánkat. Használjuk bátran!

Egy kevéske parancsfájlkészítési ismeret

A most megírásra kerülő parancsállomány nem teljesíti a komoly parancsállományok követelményeit: a hibaellenőrzést (ezt csupán nagyon csekély mértékben támogatja) és az interaktivitást. Egyszerűen adjuk meg az új kiszolgáló nevét, amelyet használni szeretnénk és beállítja. Cserébe öt perc alatt elkészíthetjük, és semmilyen parancsfájlírási gyakorlatot nem igényel.

Legelső lépésként állítsuk be a leggyakrabban használt levéltovábbító kiszolgálót. Ehhez a */etc/postfix/main.cf* fájlban keressük meg a `relayhost` értéket, és ha még nem szerepelne a beállítások között, írjuk bele.

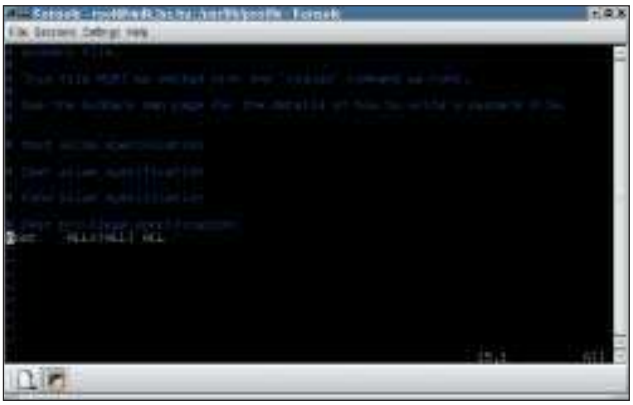
```
/etc/postfix/main.cf:
relayhost = mail.szolgaltato.hu
```

Csaknem készen is vagyunk, mindössze kedvenc kiszolgálónkkal kell újraolvastatnunk a beállítófájlt:

```
root@localhost# postfix reload
```

Következő lépésként írjuk meg a parancsfájlt, amely legyen a */usr/bin/relaychange*:

```
#!/bin/sh
# A programra a GNU GPL felhasznÆlÆsi
# szerzıdØs ØrvØnyes. A felhasznÆlÆsi
# szerzıdØs a tovÆbbi rØszletekØrt megtekintheti
# a www.geekfinder.hu/licenszek.html oldalon
PARANCS=/usr/sbin/postconf
echo "LevØltovÆbb t  ÆtÆll tÆsa a k vetkezıre: $1"
sudo $PARANCS -e relayhost=$1
echo "A kiszolgÆl  ÆtÆll tÆsa megt rtØnt"
echo "Az œj levelezıkiszolgÆl : $1"
```



Ez a parancsfájl azonban semmilyen hibaellenőrzést nem végez, egyszerűen csak átveszi a parancs utáni első értéket és megpróbálja végrehajtani. Ha nem jár sikerrel, azt a konzolon látni fogjuk, mert a Postfix programjai a hiba okát is visszaadják. Elemezzük a sorokat! Az első sorban adjuk a rendszer tudtára, hogy ez egy parancsfájl és a végrehajtáshoz a `bash` segítségét kéri majd.

A felhasználási szerződés feltételei egyértelműek. A `PARANCS` változóban adjuk meg azt az utasítást, amelyikkel a Postfix futási időben tudja változtatni a beállításait. Ennek használata után tehát már nem szükséges a beállításokat a Postfixszel újraolvastatni! Ennél a parancsfájlnál ez a lépés akár ki is maradhatott volna, hiszen csak egy változóval dolgoztunk, ha azonban a jövőben is szeretnénk parancsfájlokat írni, akkor jó, ha megszokjuk a változók használatát. A következő sor közli, hogy éppen mi történik majd. A `$1` változó a héjprogramok parancsállományaiban a parancs után álló első változót jelöli: az `$2` értelemszerűen a másodikat és így tovább. Most érkeztünk el a legfontosabb sorhoz: a `sudo`-t arra használjuk, hogy végrehajtsa számunkra a parancsot. Mint már említettem, ha valaki ezzel indít egy programot, akkor az úgy fog végrehajtódni, mintha a rendszergazda hajtotta volna végre. Ne keverjük azonban össze a `SETUID` bittel, ezáltal ugyanis minden felhasználónak lehetősége nyílna a programot rendszergazdai jogosultságokkal végrehajtani! Így csak a „kiválasztottak” férhetnek hozzá. A `sudo` kiadása után `$PARANCS` változóval helyettesítjük be a `postconf` elérési útját a parancsba, és a `-e` kapcsolóval

(szerkesztés (edit)) indítjuk. Az utána álló változót a Postfix-nek ismernie kell, tehát a beállítási fájlba illeszthetőnek kell lennie. A levéltovábbító beállítására a `realyhost` értéket használjuk. A változó értékét a parancs utáni első érték adja meg: `$1`. A `relaychange mail.szolgaltato.hu` parancs kiadása és a behelyettesítések elvégzése után a következő értéket kapjuk:

```
sudo /usr/sbin/postconf -e
relayhost=mail.szolgaltato.hu
```

Ezután már csak a visszajelzések maradtak hátra, amelyek a korábbi sorokat tanulmányozva egyértelműek. Fény derül arra is, hogy a parancsállományok túlságosan egyszerűek: ha nem lenne jogosultságunk a műveletek végrehajtására, akkor is azt írná ki, hogy a kiszolgáló átállítása megtörtént. Jelen esetben minket ez nem érint, mert mindent pontosan beállítunk, a parancsfájlt azonban érdemes továbbfejleszteni.

A második lépésben szerkesszük át a */etc/**sudoers*** fájlt. Adjuk ki a `visudo` parancsot, ekkor a *képen* (56. oldal) látható látvány fogad bennünket.

```
# sudoers file.
#
# This file MUST be edited with the 'visudo'
        ➥command as root.
#
# See the sudoers man page for the details
        ➥on how to write a sudoers file.
#
# Host alias specification

# User alias specification

# Cmnd alias specification
Cmnd_Alias      POST=/usr/sbin/postconf
# User privilege specification
root            ALL=(ALL) ALL
felhasznÆl nØv   ALL= NOPASSWD:POST
```

Szerkesszük át a fájlt, hogy a *listánkon* látható módon nézzen ki. Nézzük végig, mi mit is jelent a listánkban.

- a `Host alias` sor utáni mezőbe a számítógépek nevei tehetők, csoportosítva. Jelenleg érdektelen a számunkra, mert csak egyetlen számítógépünk van.
- A `User alias` sor már érdekesebb, mert ha több felhasználónk is létezik, akiknek jogosultságot szeretnénk adni arra, hogy megváltoztassák a levéltovábbító kiszolgálót, akkor csoportba tehetjük őket, és a csoport nevével hivatkozhatunk rájuk. Ez leegyszerűsíti a felügyeletet – a későbbiek folyamán csak egy nevet kell elvenni vagy hozzáadni a csoporthoz. Nézzünk egy példacsoportot, amely ezt valósítja meg:

```
User_Alias RELAY = felhasznalonev1,
                   felhasznalonev2,
                   felhasznalonev3
```

A továbbiakban elég csak RELAY-ként hivatkozni rájuk.

- A `Cmnd alias` nagyon fontos. Itt szintén hasonlóképpen csoportosíthatjuk a parancsokat, mint a felhasználókat. Egyelőre itt is csak egyetlen utasítás található, ha viszont a csoportosított felhasználók több parancsot is végrehajthatnak, érdemes a parancsokat is csoportosítani. Itt csupán egy `alias`-t használunk.

```
Cmnd_Alias POST=/usr/sbin/postconf
```

Több parancsot az `alias` után vesszővel elválasztva lehet írni, például:

```
Cmnd_Alias PARANCSOK=/usr/sbin/postconf,
/usr/sbin/postfix
```

- A következő osztályban adhatunk jogosultságokat az egyes személyeknek vagy csoportoknak. A példából a formátuma világosan látszik. Az `ALL` fenntartott szó, mindenre jogosultságot ad, és mint láthatjuk, a rendszergazdának mindenhez jogosultsága van. Jelenleg felhasználónknak – bár ez minden gépen érvényes – csak a POST-csoportban lévő parancsokhoz `postconf` van használata. Ezekhez azonban nem kell külön jelszót beírnia. Ha azonosítani szeretnénk a felhasználót, a `NOPASSWD:` lehetőséget hagyjuk el a fájlból – ekkor a rendszer az engedélyezett parancs végrehajtása előtt jelszót kér a felhasználótól. Mivel otthon vagyunk, nyugodtan megbízhatunk magunkban, élesben azonban a ***sudoers*** fájlba sose tegyünk a következő sorhoz hasonlót:

```
felhasznalonev      ALL= (ALL) NOPASSWD:ALL
```

Ezzel teljes rendszergazdai jogosultságot adtunk a felhasználónak, amellyel a jelszó használata nélkül is bármikor élhet! Ha megszerzik vagy megfejtik a jelszavát, a rendszer elesett.

A szabályok csoportokra is egyszerűen alkalmazhatók. Maradjunk előző példánknál, amelyiknél egy beállítási sor a következőképpen néz ki:

```
RELAY                ALL=NOPASSWD:PARANCSOK
```

Tehát a `RELAY`-csoportba tartozó felhasználók jelszó használata nélkül végrehajthatják a `PARANCSOK` csoportjába tartozó programokat - mintha ők lennének a rendszergazdák.

Lépjünk ki a szerkesztésből és mentsük a módosításokat. (az Esc, majd a : billentyű lenyomása után írjuk be: `wq` és nyomjunk Enter-t. Ha a fájlba szeretnénk írni, először az I betűt nyomjuk le, majd a Nyíl billentyűkkel lépegessünk). Most már megvan a parancsfájl, él a ***sudoers*** fájl módosítása, tehát próbáljuk is ki.

```
felhasznalo@localhost$ relaychange
mail.szolgaltato.hu
Level tovabbito atallitasa a kovetkezore:
mail.szolgaltato.hu
A kiszolgalo atallitasa megtortent
Az uj levelezo kiszolgalo: mail.szolgaltato.hu
```

Ellenőrizzük az eredményt, amihez ismét a `postconf` programra lesz szükségünk. Adjuk ki a `postconf relayhost` parancsot, ami visszaadja az eredményt. Figyeljük meg, hogy most hiányzik-e a `-e` kapcsoló.

Ha az eredmény egyezik a kívánttal, hátradőlhetünk és vállon veregethetjük magunkat.

Egy másik olvasónk kérésének eleget téve egyik kedvenc Szabó Lőrinc versemet kitettem a

➲ http://www.geekfinder.hu/vers.html címre.

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.

A levelezéshez kapcsolódó protokollok (1. rész)

Olvasóink között akadt, aki a legutóbbi cikkem után azzal a kéréssel fordult hozzám, hogy a levelezéshez használható, illetve kapcsolható protokollokról írjak.

SMTP – rfc821

A levelezés alapját képező protokollt 1982-ben mutatták be a nagyvilágnak. A levelek e protokollon keresztül jutnak el egyik kiszolgálóról a másikra. Jól bevált, üzembiztos protokoll, mégsem mentes a hiányosságoktól. A korszerű levelezőkiszolgálók (MTA-k) egyes részeit biztonsági kockázatok miatt alkalmazásaikban nem is valósítják meg, ilyen például a `TURN` parancs. Jelentős szerepet játszott az Internet elterjedésében, hiszen a kezdeteknél a számítási műveletek távoli elvégzése mellett a fő cél a levélváltás megvalósítása volt.

Az SMTP kódjai

220 – A szolgáltatás készen áll.
221 – A szolgáltatás megszakítja a kapcsolatot.
250 – Minden rendben.
251 – Távoli felhasználó a címzett, a levél továbbítva lesz.
354 – Adat fogadása, a fogadás a „.” jellel ér véget.
450 – A kért szolgáltatás elmarad, a ***postafiók*** nem érhető el. (Ez általában időleges hiba.)
451 – Hiba történt a feldolgozás alatt, a kért szolgáltatás nem lett végrehajtva.
452 – Nincs elég tárhely. (Tárkorlát – quota – használata miatt is előfordulhat, nem biztos, hogy a hely fogyott el.)
500 – Formai hiba, nincs ilyen parancs.
501 – Formai hiba az értékeknél.

Az SMTP leggyakoribb parancsai

HELO – Az ügyfélszámítógép így mutatkozik be a kiszolgálónak. Megeshet, hogy ha az ügyfél által átadott név nem egyezik meg azzal az IP-címmel, amelyről érkezett, s a kiszolgáló visszautasítja a kapcsolatot.
MAIL – A küldőt azonosítja. Általános formája a `MAIL FROM: <levØlc m>`
RCPT – A címzettet azonosítja. Általános formája a `RCPT TO: <levØlc m>`. Egy levél több címzettel is rendelkezhet.
DATA – Az ügyfél ezzel jelzi, hogy adatot fog küldeni a kiszolgálónak. Az adatküldés végét egy soremelés utáni, a sorban egyedül álló . (pont) karakter jelzi.
RSET – Elindítja a kapcsolatot, egészen a `HELO` parancsig. Az összes addigi ***MAIL***, ***DATA***, illetve ***RCPT*** mező értéke lenullázódik. Az ügyfél akkor használja, ha a fogadóoldalon hibát feltételez, de a kiszolgálók elleni támadásnál is használatos.
VRFY – Ellenőrzi a kiszolgálón lévő felhasználó meglétét. A levélszemétküldők egyik kedvenc módszere, hiszen csak próbálgatni kell a neveket, és egyszer csak visszajön egy jó cím. Ha megtalálta a felhasználót, akkor kiírja a címét és a 250-es kódot. Érdemes a kiszolgálókon kikapcsolni.
EXPN – Nemcsak a helyi felhasználók listáját ellenőrzi, hanem az `alias` fájlokat és levelezőlistákat is. Ezzel még több címhez juthat hozzá az, aki címeket akar szerezni. A Postfix már nem is valósítja meg, itt nem kell külön kikapcsolni.
TURN – Biztonsági okokból sehol sem használatos . Egyik kiszolgálóprogramba sem írták bele a szolgáltatást. Helyét az `ETRN` parancs vette át (lásd az ESMTP-nél)
HELP – A kiszolgáló közli az ügyféllel, hogy milyen szolgáltatásokat képes nyújtani.
QUIT – Az ügyfél közli a kapcsolat lezárását a kiszolgálóval.

ESMTP – rfc1869

Az ESMTP-t elődje gyengéinek kiküszöbölésére hívták életre. Ez nem azt jelenti, hogy az elődöt rosszul írták meg, csak a körülmények változtak. 1995-ben adták ki és az SMTP-hez képest számos újdonságot hozott. A kódok ugyanazok maradtak, de a parancsok köre bővült, illetve részben módosult – itt csak a különbségeket mutatom be.

EHLO – A `HELO`-t váltotta fel. Ha egy ügyfél így jelentkezik be, akkor a kiszolgáló tudja, hogy a másik oldal `ESMTP`-parancsokat képes küldeni és fogadni. Bejelentkezéskor a kiszolgáló felsorolja az általa biztosított szolgáltatásokat (ETRN, DSN stb).
DSN – Segítségével az ügyfél visszajelzést kaphat, hogy üzenetét sikerült-e elküldeni vagy sem.
ETRN – A `TURN` parancsot váltotta le. Ha kiadjuk a parancsot és utána a tartománynevünket, a kiszolgáló elküldi a nekünk szánt, de addig a kiszolgálón tárolt leveleket. A küldésnél nem a jelenlegi IP-címet veszi figyelembe, hanem a DNS-ben bejegyzett MX rekordot. A `TURN` parancs azért nem volt biztonságos, mert bárki le tudta tölteni a saját gépére a nekünk szánt leveleket, ugyanis a kiszolgálók a pillanatnyi IP-címet tekintették érvényesnek. Az `ETRN` szolgáltatást főleg internetszolgáltatók teszik elérhetővé az ügyfeleik számára. Ha megszakad a hálózati kapcsolatom és az egyik felhasználóm számára levelet küldenének, valamint a szolgáltató egy `backup` rendszert biztosít a levelezéshez, a levelek oda továbbítódnak majd. Ha megint élő kapcsolatom lesz, akkor bejelentkezhetek a szolgáltató gépére és az `ETRN` parancs segítségével átvehetem tőle a leveleket.
AUTH – Lehetővé teszi, hogy az ügyfelek egy felhasználónév-jelszó párossal vagy más megengedett módon azonosítsák magukat. Ha az azonosítás sikeres volt, az ügyfél elérheti azokat a lehetőségeket, amelyeket az azonosítatlan felhasználó nem. Ilyen lehet például a bárhonnan engedélyezett levélküldés a kiszolgálón keresztül.

A fentieken kívül még az IMAP és a POP3 protokollt emelném ki (bővebben lásd róluk a Linuxvilág 2001. szeptemberi számának 53. oldalát).

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.

GPG: a legjobb szabad titkosító program (2. rész)

Mick onnan folytatja a GnuPG-ről szóló cikkét, ahol abbahagyta – és ha lehet, még jobban erőt vesz rajta az üldözési rögeszme.

A múlt hónapban bemutattam a ***GNU Privacy Guard***-ot, az ***OpenPGP*** szabvány szabad, de kevéssé használt megvalósítását. A ***GnuPG***, ahogyan már biztosan tudod, nagyon hasznos lehet fájlok titkosításához a megfejtéséhez – ez főleg a levelezésnél előnyös –, valamint digitális aláírások létrehozásához és a hitelességük ellenőrzéséhez.
Múlt alkalommal helyszűke miatt sajnos épphogy csak el tudtam magyarázni a nyilvános kulcsú titkosítás alapjait, a bizalomhálót, az egymás kulcsai aláírásának és az ismeretlen kulcsok ellenőrzésének a fontosságát. Ezután már csak néhány telepítéssel kapcsolatos gyakorlati tanácsra és a digitálisan aláírt fájlok hitelességellenőrzési módjának rövid taglalására maradt lehetőség.
Ebben a hónapban azoknak is a kedvére teszek, akiket a téma mélyrehatóan érdekel. Folytassuk onnan, ahol abbahagytuk.

A kulcspár létrehozása

A fájlok titkosításához és a titkosított üzenetek megfejtéséhez, illetve a digitális aláírások készítéséhez saját kulcspár szükséges: egy nyilvános és egy titkos kulcs. Hozzunk létre GnuPG kulcspárt! A `gpg` ezt a feladatot párbeszédes formában oldja meg: a kulcsok sikeres legyártása érdekében a parancssoron az egyszerű `gpg --gen-key` parancsot kell begépelni, aminek hatására a program kérdéseket tesz fel, amelyeket meg kell válaszolnunk. Az 1. listán (23. CD Magazin/GPG könyvtár) láthatunk példát a kulcskészítéshez szükséges párbeszédre (a felhasználó által beírt részt dőlten szedtük). Észreveheted, hogy a folyamat során számos döntést kell meghoznod: meg kell adnod a kulcs típusát, a hosszát, az élettartamát és a levélcímet, amelyet a kulcshoz tervezel társítani.
Általános kulcspár létrehozásához válaszd a ***DSA/ElGamal***-lehetőséget (ez az első). Ez két kulcspárt készít: a DSA-kulcspárt, ez használható az aláírások készítésére és ellenőrzésére, valamint az ElGamal-kulcspárt, amit a `gpg` titkosításra és visszafejtésre használ. Igaz, így rögtön két kulcspárod lesz, de ne aggódj, nem fogja megnehezíteni az életedet. A DSA- és az ElGamal-kulcsok – a két titkos és két nyilvános kulcs egyaránt – egy-egy fájlban találhatók.
Ha csupán az aláíráshoz van szükséged a kulcsokra, választhatod a DSA-lehetőséget (ez a második), ha pedig csak titkosításra szeretnéd használni őket, az ElGamal-lehetőséggel teheted meg. Nem ajánlom, hogy az ElGamal-kulcspárt használd mind titkosításra, mind aláírásra (bár ez a negyedik lehetőség), ugyanis ***Bruce Schneider*** „Applied Cryptography" (Alkalmazott kriptográfia) című könyvében ismertetett egy egyszerű módszert az ilyen módon titkosított üzenetek megfejtésére. Ez az eljárás („chosen plaintext") csak akkor alkalmazható sikerrel, ha a titkosításhoz és az aláíráshoz ugyanazt a kulcsot használjuk, ezért ezt a módszert az elkerülendők iskolapéldájaként említhetjük.
A kulcsméret különös fontossággal bír. A GnuPG által támogatott legkisebb kulcs 768 bites, azonban az 1024 bites kulcs használata ajánlott, ugyanis ez az arany középút a biztonság és a használhatóság között. A hosszabb kulcs biztonságosabb, de a titkosítás tovább tart vele. A rövidebb kulcsot gyorsabban létrehozza a gép, és a mindennapi használatban kevesebbet várakoztatja az embert, viszont nem annyira biztonságos. Megjegyezném, ha DSA-, illetve ElGamal-kulcspárt hozol létre, a DSA-kulcs hossza mindenképpen 1024 bit lesz, a feltett kérdés csak az ElGamal-kulcs hosszára vonatkozik.
Fontos azt is eldönteni, hogy a kulcs mennyi ideig maradjon forgalomban. Ha úgy állítod be őket, hogy visszavonásig érvényesek legyenek, egyfelől megmenekülsz az új kulcsok létrehozásának kényelmetlenségétől, viszont kellemetlen helyzetbe kerülhetsz, ha esetleg elfelejted a jelmondatodat, és előzőleg nem készítettél visszavonási tanúsítványt (erről később bővebben írok) – ilyen esetben ugyanis elég körülményes a nyilvános kulcs kivonása a forgalomból, azaz a kulcskiszolgálókról való törlése.
Másrészről ha a kulcsod lejárati ideje adott, nem kell aggódnod, hogy a régen elfeledett és használaton kívüli nyilvános kulcsod az idők végezetéig tárolódik egy kulcskiszolgálón: ha a levélcímed megváltozik, vagy úgy döntesz, hogy a régi kulcsod már

Páncél (Armor) és más beállítások

A `gpg --export` és `--gen-revoke` kapcsolói valójában parancsok, ezek határozzák meg a `gpg` teendőit. Ha azt is meg akarod mondani a `gpg`-nek, hogy hogyan tegye, amit tesz, a parancsok előtt különböző beállításokat alkalmazhatsz. Ilyen beállítás lehet például a `--armor` (ASCII-páncél alkalmazása) és a `--output` (kimenet a megadott fájlba). A `--output` vár még egy argumentumot (meg kell adni a fájl nevét), a `--armor` kapcsolónak nincs argumentuma.

A páncélozott ASCII (kiterjesztése .ASC) hordozhatóbb adatformátum, mint a `gpg` alapértelmezett bináris formátuma (.GPG kiterjesztés). A bináris adattal ellentétben a páncélozott ASCII-t ki lehet vágni, és be lehet illeszteni mondjuk egy levelezőprogramba. Ha lemezre mented, a páncélozott ASCII-fájl teljesen szokványos szöveges állomány. Ebből kifolyólag a kulcsok terjesztésére, mentésére és továbbítására ASCII-páncélt érdemes használni. Azért említem ezt itt meg, mert a beállításokat más úton is meg lehet adni: az options fájlban, ami a *~/.gnupg* könyvtárban található. Ha a `gpg` alapértelmezett fájlformátumát páncélozott ASCII-ra akarod változtatni, az options fájlhoz add hozzá a következő sort:

```
armor
```

Általában mindent, amit a parancssorban meg lehet adni, az *options* fájlból is be lehet állítani, csak a kezdő `"--"`-t kell elhagyni. A parancssorban megadott beállítások felülbírálják az options fájlban lévőket. További részleteket a gpg(1) súgóoldalán találsz.

2. lista Egy nyilvános kulcs

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org

mQGiBDug+KURBACtQFiSy057YI5Q7I5EDQjWxn/laQ
KUiXsbpt3ar5bZ7ObjGxpcdWiwNqYqWMcSFwEDEKHd
Cr098CGK0B247I/Y9xg1WDOpDQolbgq4Z94uEWENXm
lpftNxQjbz9Dil3VtoRHh325hFb9DzGBx5JB8mb4kp
up9uc+c11UDXuf3hXwCguIezZMF4q5VCv3xIDDlTk7
W6WvMD/3gc3ob2gix68F7JvYduAGrXjg8ExTx7x8MP
0PnmcHBkLefzBYWrQGIdaqaTOSBGVpPxRJAfpdSieG
HKLCyFcV7h1FyIVYBqUmw81cq0Ap62nYVJxCucRWIv
UObW98q/8dxx8K5HVTJ8ItUMh47C/GkkNtlYKYcKKr
aBb/9oulzHA/kBGNM7F1zlVHbAUEgJ+vJ1rbnXnY4E
luAG/I28rE36aVfwiiGzXOR/KUtBl7ln9f7b6z4EfB
FqQuauY9T8f2kEEb9gRtEKlOVDr1///adSrCVv/C9w
Hk2xAyuAhh1NLFb37tTCrm687twHtcyTHgwffnnFSs
Pp79OrdvOzI/5iLbQdRW1lc2UgS292YWNzIDxlbWVz
ZUBnbm9tZS5odT6IVwQTEQIAFwUCO6D4pQULBwoDBA
MVAwIDFgIBAheAAAoJEGQlJf7FBw/7J6kAoI/D5puP
O9v0QOxPYeAs1+Dxf0jaAJ9BdBXHM6vJsnJrMLORKj
eaxPj3johGBBARAgAGBQI7sczTAAoJECXTh7xXlOhV
ZIMAoJnigXUHM6CVvYlF4avmUPV/3/RsAJ4zf6O1UT
D282uY4q8upywdtUyworkBDQQ7oPi3EAQA3mCnLcPC
Le0zbq8WQuAdGvd9UwYg7/sDZs3CqLQEXHvV8XbFKm
BuAuBKklu2XB8dUFyVVKz12Aa0Hlce93Ty1INsL7Ns
bHwouLK8Z176N4gFQqEpfurf1iQylkilifbrj1QvwX
gLthQd/vxDlVsebpEw9UqdDJgTLgtFeqxRlgsAAwYE
ANuB189hTgJBUb58NNdD9ZOzM8HQXV37gVyvghMbRE
r/V5I7G4Jq72dr3rCei7X/zGrtpJaCB8zJLXqM8GZZ
176IHI2TE8rQxGJ+gSmTbWmlXktazTpiajDochlKqP
1z9GkYq6Wrt+fWatNLrDGNAd+9YJJe51M0Z7NTxJY1
3/z+iEYEGBECAAYFAjug+LcACgkQZCUl/sUHD/v2LQ
CeIFJn6B5/UN4zcHwktPYOqgaVRgIAoKH6Xhimw7vo
nZgBdBJTD97/1nFw=Oy2y
-----END PGP PUBLIC KEY BLOCK-----
```

nem elég biztonságos, esetleg a tudomásodra jut, hogy valaki lemásolta a titkos kulcsodat és valamiért nem tudod visszavonni – semmi gond, a kulcs előbb-utóbb elöregszik. A rövid élettartam legfőbb hátránya, hogy adott időközönként új kulcsokat kell létrehozni, az új nyilvános kulcsot terjeszteni kell, és talán a legnehezebb feladatként rá kell bírnunk a többieket, hogy ezt használják.

Régebben csak határozatlan élettartamú kulcsokat használtam, de mára meggyőződésemmé vált, hogy a meghatározott lejárati idő előnyei felülmúlják a hátrányait. Ezért azt a tanácsolom, hogy a kulcs érvényességét 18 és 24 hónap közötti időtartamra állítsd be. Nekem az egy év túl rövid (tempis fugit!), de nem hiszem, hogy egy másfél évnél hosszabb életű kulcs ellenállhatna a számítástechnika fejlődése következtében elérhetővé váló számítási teljesítménynek és az új kulcstörési módszereknek.

Ezt követően meg kell adnod egy nevet, levélcímet és esetleg megjegyzést is. Felhívnám a figyelmedet, hogy a későbbiek folyamán a kulcshoz további levélcímeket adhatsz. Ezt a `gpg --edit-key` kapcsoló után megadott `adduid`, illetve `addkey` paranccsal teheted meg.

A kulcsok létrehozásához utoljára egy jó jelmondat szükséges. Amikor „jelmondatot” írok, arra is nagy gondot fordítok, hogy szóközöket is tartalmazzon, ugyanis minél hosszabb, annál biztonságosabb. A jelmondatban jó, ha van kis- és nagybetű, számok és írásjelek (például bOTTLE rockeT!). Az utóbbi időben a jelmondatok létrehozásához dobókockát és szólistát használok. Ha kedvet kaptál a kipróbálásához, az eljárás pontos leírását megtalálod a ➲ http://www.diceware.org oldalon. Semmiképp ne válassz rövid, könnyen kitalálható jelmondatot. Nem kell, hogy a jelmondat így nézzen ki: „B1&SSja-sd0c as-d$%@KFSAAs-,ssd w0a-00sdp23m”, de ez sem megfelelő: „Az én béna jelmondatom”. Teljesen rendjén való, ha egy-egy bonyolultabb jelmondatot felírsz egy kis kártyára, és a levéltárcádban magadnál tartod (csak arra vigyázz, hogy használat után mindig tedd el, ne maradjon szem előtt!).

Visszavonási tanúsítvány készítése

Miután létrehoztad a kulcspárodat, érdemes azonnal visszavonási tanúsítványt készítened. Abban az esetben, ha a kulcsodat vissza akarnád vonni, ezt a karakterláncot kell a kulcskiszolgálónak elküldeni.

Természetesen bármikor létrehozhatsz visszavonási tanúsítványt, ezt azonban érdemes rögtön a kulcsok létrehozása után megtenni, mert még a legkörültekintőbb felhasználókkal is előfordulhat, hogy elfelejtik a jelmondatukat. A jelmondatra szükség van a visszavonási tanúsítvány létrehozásakor, a későbbiekben viszont, a felhasználás folyamán már szükségtelen.

Ezért jó, ha azonnal elkészítjük a visszavonási tanúsítványt és eltesszük egy biztonságos helyre (akár ki is lehet nyomtatni és a „metatérben” tárolni, ugyanis a tanúsítványok nem túl hosszúak). Mindössze arról bizonyosodj meg, hogy a fájl jogosultsága megegyezik a titkos kulcsodéval (például csoport és a világ által nem írható vagy olvasható). Nem annyira rettenetes, ha valaki a visszavonási tanúsítványt megszerzi és érvényteleníti vele a kulcsodat, mintha a titkos kulcsodat használná, bár a lejárati ideje előtt forgalomból kivont kulcs kellemetlenségeket okozhat.

A visszavonási tanúsítvány létrehozásához add ki a következő parancsot:

```
gpg --output vissz_tan_fÆjlnØv.asc
➥--gen-revoke kulcsnØv
```

ahol ***vissz_tan_fájlnév.asc*** a fájl neve, amelyikbe a `gpg` elmenti a tanúsítványt (csak arra figyelj, hogy .asc legyen a kiterjesztése) és a kulcsnév a szóban forgó kulcs azonosítója (például 0586AF78) vagy a nevednek egy része („Smooth JoJo” pont elég lenne a példakulcsunkhoz).

Nyilvános kulcsod terjesztése

A GnuPG a fájljait a felhasználói könyvtáradban található ***.gnupg*** könyvtárban tárolja. Minden titkos kulcsot a ***secring.gpg*** fájlban tárol, minden nyilvános kulcsot a ***pubring.gpg***-ben. Alapértelmezetten a secring.gpg csak a tulajdonos által olvasható fájl, ezt hagyd is így: nagyon fontos, hogy ezt a fájlt megvédd. Mindenképpen mentsd hajlékony lemezre vagy CD-re és tárold biztonságos helyen. Ha valaki megszerzi a titkos kulcsod másolatát, kitalálhatják vagy megfejthetik a jelmondatodat és ellophatják a személyazonosságodat (de legalábbis megfejthetik a neked szánt titkos üzeneteket).

A ***pubring.gpg*** és a ***secring.gpg***} egyaránt bináris adatfájl. Ahhoz, hogy a kulcsokhoz további kulcsokat tudjál hozzáadni, módosítani vagy a készletből törölni, a `gpg` parancsot kell

3. lista Kulcs szerkesztése (hitelesítése)

```
jojo@linux:~ > gpg --edit-key dan
gpg (GnuPG) 1.0.4; Copyright (C) 2000 Free
Software Foundation, Inc.
This program comes with ABSOLUTELY NO
WARRANTY.
This is free software, and you are welcome
to redistribute it under certain conditions.
See the file COPYING for details.


pub  1024D/C9F34866  created: 2001-07-27
     expires: 2001-08-10      trust: -/q
sub  2048g/C5569A5B  created: 2001-07-27
     expires: 2001-08-10
(1)  Dan Sparty (Party on!)
<dan@boogiemeister.com>

Command> sign

pub  1024D/C9F34866  created: 2001-07-27
     expires: 2001-08-10      trust: -/q
             Fingerprint: FD084 F92C EC62
             8955 98E2 58FB 178A 2673 D1F3
             6866

     Dan Sparty (Party on!)
<dan@boogiemeister.com>

Are you really sure that you want to sign
this key with your key: "John J. Figplucker
(Smooth JoJo) <jojo@figpluckers-supreme.to>"

Really sign? y

You need a passphrase to unlock the secret
key for user:
       "John J. Figplucker (Smooth JoJo)
       <jojo@figpluckers-supreme.to>"
1024-bit DSA key, ID C1C34866, created 2001-
07-27

Command> save
jojo@linux:~ >
```

használnod a megfelelő kapcsolókkal.
Tegyük fel, hogy nyilvános kulcsodat terjeszteni szeretnéd a barátaid között. Ehhez a kulcsot ki kell emelnünk a nyilvános kulcscsomódból, és el kell helyeznünk egy szövegfájlban (lásd a „ASCII páncél vagy bináris GPG fájlok” széljegyzetet). A nyilvános kulcs képernyőre való kiíratásához csak ennyit kell megadnod:

```
gpg --armor --export
```

A kimenet a *2. listában* találhatóhoz hasonlít. Ezt a kulcsot igény szerint bárhova lehet másolni, illetve beilleszteni.
A fenti példa egy kicsit egyszerű volt: ha nem adsz meg felhasználói azonosítót, a `gpg` alapértelmezett kulcspárod nyilvános részét írja ki. Ha csak egy titkos kulcsod van, akkor az ahhoz tartozó kulcspár az alapértelmezett, így a hozzá tartozó nyilvános kulcsot kapjuk meg.
Ha más nyilvános kulcsot szeretnél terjeszteni, egy felhasználói azonosítót (levélcímet) kell megadni.
Példánkat folytatva: ha Mr. Figplucker kulcscsomójáról akarjuk JoJo nyilvános kulcsát terjeszteni, a következőt kell megadunk:

```
gpg --armor --export jojo
```

Ez azt is mutatja, hogy a `gpg` okosan megpróbál rájönni, hogy melyik kulccsal szeretnél dolgozni. Valójában a `grep` parancshoz hasonlóan működik: megadod a cím egy részletét vagy valami más jellegzetes, a kulcsot azonosító szövegrészt, és a `gpg` az első mintára illeszkedő kulcsot használja fel. A saját kulcscsomóim kezelése során a legegyszerűbbnek az bizonyult, ha a teljes levélcímet megadom, mert több titkos kulcsom is van, amelyeken a felhasználói név azonos.
Például: `gpg --armor --export mick@visi.com`
Ha már itt tartunk, a kulcsot nem kötelező a képernyőre kiíratni, a `--output` kapcsolóval azonnal fájlba is irányítható. JoJo nyilvános kulcsával ez a következőképpen nézne ki:

```
gpg --armor --output jojo_pub.asc --export jojo
```

Ekkor a *jojo_pub.asc* tartalmazza a kulcsot.
Lementetted már az új kulcspárodat? Nyilvános és titkos kulcscsomódat egyaránt terjesztheted, ezt azonban nem ajánlom. Sokkal egyszerűbb, ha a *pubring.gpg* és *secring.gpg* kulcscsomófájlokat egy az egyben biztonságos helyre mented a *~/.gnupg* könyvtárból. Ha valamilyen okból kifolyólag mégis ragaszkodsz a terjesztéshez, a fentiekhez hasonló módon teheted meg, a `--export` helyett a `--export-secret-keys` kapcsoló használatával.

Azt hittem, a 128 bites kulcs már hosszúnak számít

Lehet, hogy a Blowfish, a Triple DES (3DES) és a hasonló kulcsok hosszához vagy szokva, amelyeknél a 128 bites titkosítás már erősnek minősül. Ez a szimmetrikus titkosító algoritmusok esetén igaz, amikor ugyanazzal a kulccsal titkosítunk és fejtjük meg a titkosított adatfolyamot. A nyilvános kulcsú titkosítás világában a kulcsok durván tízszer hosszabbak, mert ez teljesen más matematikai alapokon nyugszik, mint a szimmetrikus eljárások. Olyan, mintha az almát hasonlítanád a körtéhez.

Más kulcsának beolvasása, ellenőrzése és aláírása

Barátod, ***Dan Sparty*** éppen most küldött neked egy levelet, amelyhez a *danskey.asc* fájlban az új nyilvános kulcsát mellékelte. Az új kulcsot nyilvános kulcscsomódra a következőképpen fűzheted fel:

```
gpg --import ./danskey.asc
```

Álljunk csak meg egy pillanatra! Az internetes levelezés köztudottan nem a biztonságos adatátviteli módok közé tartozik. Honnan tudhatod, hogy Dan kulcsát nem cserélték-e ki útközben?
Egyszerű: a kulcs ujjlenyomatát kell ellenőrizned. Minden gpg-kulcsnak van egy ellenőrző karakterlánca, ez az ujjlenyomat.

Ez minden kulcs(pár)ra nézve egyedi, de elég rövid ahhoz, hogy fel lehessen olvasni telefonon vagy fel lehessen írni egy képeslapra. Hívd fel Dant telefonon és kérd meg, hogy olvassa fel a kulcsa ujjlenyomatát; aminek meg kell egyeznie a következő parancs kimenetével (ezt a parancsot az új kulcs beolvasása után kell kiadni):

```
gpg --fingerprint dan
```

Megjegyezném, hogy itt is, miként a `--export` parancsnál, elegendő, ha a kulcs kiválasztásához az őt egyértelműen azonosító névrészletet adod meg. A kimenet ehhez fog hasonlítani:

```
pub  1024D/C9F34866 2001-07-27 Dan Sparty
     (Party Down!) <dan@boogiemeister.org>
      Key fingerprint = D084 F92C EC62 8955
      98E2  58FB 178A 2673 D1F3 6866
sub  1024g/C5569A5B 2001-07-27 [expires:
     2001-08-10]
```

Egy másik megoldás (tegyük fel, még csak délelőtt van, és nem akarod Dant felébreszteni), a nyilvános levelezőlistákon és Usenet-hírekben utánanézni annak, hogy Dan hogyan írta alá a leveleit. Ez kihangsúlyozza az ujjlenyomatok fontos tulajdonságát: minél több helyen jelenik meg a nyilvános kulcsod, illetve az ujjlenyomata, annál nehezebben tudják meghamisítani a személyazonosságodat.
Most, hogy már biztosan tudod, hogy a kulcs valóban Dané és nem hamisítvány, Dan kedvéért megteheted, hogy az aláírásoddal hitelesítheted a kulcsát, azaz aláírhatod a titkos kulcsoddal. Hogy ezt megtedd, a `gpg`-t a `--edit-key` paranccsal kell futtatnod. Ennek eredménye a `--gen-key` esetében már megfigyelt üzemmód lesz. A *3. listán* látható a programmal folytatott párbeszéd, amelynek során a felhasználó alapértelmezett kulcsával ír alá egy nyilvános kulcsot.
Észrevetted a `save` parancsot a végén? Ez menti a kulcsokon véghezvitt változásokat (ebben az esetben az aláírást) és kilép a párbeszédüzemmódból. Ha a kulcsot most a `gpg --list-sigs dan` paranccsal nézed meg, a következőt látod:

```
jojo@linux:~ > gpg --list-sigs dan
pub  1024D/B9E0868B 2001-07-27 Dan Sparty
(Party On!)
     <dan@boogiemeister.org>
sig       B9E0868B 2001-07-27  Dan Sparty
(Party On!)
        <dan@boogiemeister.org>
sig       C1C34866 2001-07-27  John J.
Figplucker
         (Smooth JoJo) <jojo@figpluckers-
supreme.to>
sub  1024g/A0B78448 2001-07-27 [expires: 2001-
08-26]
sig       B9E0868B 2001-07-27  Dan Sparty
(Party On!)
           <dan@boogiemeister.org>
```

Dan saját aláírása mellett (a gpg a kulcs létrehozásakor azt titkos párjával magától aláírja) most már JoJoé is ott díszeleg. Ezek után JoJonak terjesztenie kell Dan nyilvános kulcsának új, aláírt változatát:

```
gpg -- output dan_jojosig.asc --export dan
```

JoJonak ezt a fájlt el kell juttatnia Danhez, például levélben – hiszen nyilvános kulcsról van szó, így a biztonság nem létfontosságú kérdés. Dannek az aláírt kulcsot fel kell fűznie a kulcscsomójára, ahol az új változat helyettesíti a régit:

```
gpg --import ./dan_jojosig.asc
```

Lehet, hogy a „terjesztés” nem tűnik ésszerűnek, hiszen már létező kulcsról van szó, tulajdonképpen Dan frissíti a nyilvános kulcsát, és nem új kulcsot fűz fel. De bízz bennem, ezt kell tennie ahhoz, hogy csatlakozzon azon büszke `gpg`-felhasználókhoz, akik vették a fáradtságot és a kulcsukat aláíratták egy ismerősükkel.
Most, hogy Dannek megvan a csúcsszuper hitelesített kulcsa, készen áll, hogy elküldje egy kulcskiszolgálóra, és mások titkosított üzeneteket küldhessenek neki, és még többen aláírhassák a kulcsát. Ehhez a következő parancsot használja:

```
gpg --keyserver pgp.mit.edu --send-keys
➥dan@boogiemeister.org
```

A `--keyserver` kapcsolóval lehet megadni a PGP/GPG kulcskiszolgáló nevét vagy IP-címét. Másik megoldásként bele lehet írni a

```
keyserver pgp.mit.edu
```

sort a *~/.gnupg/options* fájlba. (Ugyanebben a fájlban érdemes megadni a ***charset*** `iso-8859-2` sort is, így a magyar ékezeteket is helyesen értelmezi a progam – a főszerk.)
Ha ez utóbbi eljárást alkalmazzuk, tudnunk kell, hogy ennek hatására a `gpg` elkezdi magától letölteni az aláírások ellenőrzéséhez szükséges nyilvános kulcsokat a kiszolgálóról. Emlékszel, amikor a múlt hónapban ellenőriztem a programcsomag aláírását? Az első próbálkozásnál még nem volt a kulcscsomómon az aláírást létrehozó kulcs nyilvános párja, ezért hibaüzenetet kaptam. Meg kellett keresnem és le kellett töltenem az ellenőrzéshez szükséges kulcsot, amit a `gpg` a fenti sor alkalmazása esetén saját magától megtesz. Neked kell eldöntened, hogy melyik viselkedés áll a legközelebb az elvárásaidhoz: van, aki szereti használni ezt a lehetőséget, van, akit idegesít. (A parancssorban megadott `--no-auto-key-retrieve` kapcsolóval felülbírálhatod az önműködő letöltést.)

Titkosítás és titkosított üzenetek megfejtése GnuPG-vel

Végre elérkezett a pillanat, amikor JoJo elkezdhet titkosítani mindent, amit csak ér. Tételezzük fel, hogy JoJo titkosított üzenetet szeretne küldeni Dannek. Ennek legegyszerűbb módja, ha a levelet megírja a kedvenc szövegszerkesztőjében és a fájlt menti. JoJo tehát ír egy levelet a `vi` segítségével, és ***dan0729.txt*** néven menti. A fájlt a következő paranccsal titkosítja:

```
gpg --output dan0729.txt.asc --encrypt
➥--recipient dan@boogiemeister.org dan0729.txt
```

JoJo ekkor elküldi a ***dan0729.txt.asc*** fájlt, akár mellékletként, akár a levél törzsében (JoJonak be van állítva az `armor` sor az ***options*** fájljában).
Megjegyezném, ha JoJo nem adja meg a `--armor` kapcsolót a parancssoron és az `armor` sor nincs benne az ***options*** fájlban, a kimeneti fájlnak a ***dan0729.txt.gpg*** nevet illik adni, mert ilyenkor a kódolt fájl a `gpg` bináris alakjában jön létre. Ilyen esetben

csak mellékletként lehet elküldeni. Ne felejtsd el: az ASCII páncéllal felvértezett fájl sokoldalúbb felhasználást tesz lehetővé, ezzel szemben az eredeti gpg-alak kisebb fájlméretet eredményez, ezért olyan esetben használandó, amikor a fájl mérete fontos tényező.
Amikor Dan megkapja a fájlt, mentenie kell és a következő paranccsal meg kell fejtenie a rejtjelezést:

```
gpg --output dan0729.txt
➥--decrypt dan0729.txt.asc
```

A titkosítással ellentétben ebben az esetben nem kell megadni a megfejtéshez használandó kulcsot, a gpg magától kitalálja, hogy melyiket kell alkalmaznia. Hasonlóképpen lényegtelen, hogy Dan milyen formátumban kapja meg a fájlt (szöveges vagy bináris), a gpg ezt is önműködően kezeli. Az üzenet megfejtéséhez Dannek meg kell adnia a titkos kulcsához tartozó jelmondatot. Ha nem tudja beírni a helyes jelmondatot, nem képes visszafejteni a fájlt.

Hitelesítés és ellenőrzés GnuPG-vel

A hitelesítés és az ellenőrzés sok tekintetben hasonlít a titkosításhoz és az üzenet megfejtéséhez. Tegyük fel, JoJo ír egy fontos, de nem bizalmas levelet Dannek. Ilyenkor szükséges, hogy Dan ellenőrizni tudja a levél hitelességét, de nem kell titkosítania. A ***beercontract.txt*** fájl aláírásához JoJonak a következő parancsot kell kiadnia:

```
gpg --output beercontract_signed.txt
➥--clearsign beercontract.txt
```

Ez a parancs fejlécet és láblécet ad a fájlhoz és ***beercontract_signed.txt*** néven menti. Fontos, hogy a kimeneti fájlnév és az eredeti fájl neve ne ugyanaz legyen, ebben az esetben ugyanis az eredeti fájl helyén egy a gpg-aláírás fej- és láblécével ellátott üres fájlt találnánk. A szöveges módot akkor érdemes használni, ha a levelet a levelezőprogramba be akarjuk illeszteni, vagy ki akarjuk onnan vágni.
A másik lehetőség, igen, kitaláltad, a bináris aláírás. Ebből kétféle létezik: ha tömörített bináris fájlt szeretnél, amely a levelet és az aláírást egyaránt tartalmazza, a `--clearsign` helyett használd a `--sign` parancsot. Egy sokkal kisebb bináris fájl létrehozásához, amely csak az aláírást tartalmazza, használd a `--detach-sig` parancsot. Ebben az esetben két fájlod lesz: a bináris aláírás és az eredeti levél. Mind a `--detach-sig`, mind a `--sign` előtt használhatod a `--output` kapcsolót.
Amikor Dan megkapja JoJo sörszerződését, ellenőrizni tudja az aláírást. Ehhez a fájlt a merevlemezre kell mentenie, mondjuk ***bcs.asc*** néven és a következő parancsot kell kiadnia:

```
gpg --verify bcs.asc
```

Ne felejtsük el, ha Dannek nincs meg JoJo nyilvános kulcsa, a gpg hibával tér vissza. Ha Dan kulcscsomóján megtalálható JoJo nyilvános kulcsa, és az aláírás hitelesnek bizonyul, a program valami ehhez hasonlót fog kiírni:

```
gpg: Signature made Fri 27 Jul 2001 04:46:46
PM CDT
      using DSA key ID C1C34866
gpg: Good signature from "John J. Figplucker
      (Smooth JoJo)
       <jojo@figpluckers-supreme.to>"
```

Dan ekkor és csakis ekkor lehet biztos abban, hogy azt a szerződést, amit az imént kapott, JoJo titkos kulcsával hitelesítették. Lehet, hogy JoJo tarkójához pisztolyt fogtak? Nem tudjuk. Lehet, hogy JoJo a jelmondatát felírta a billentyűzete aljára és most a munkatársai űznek gúnyt belőlünk? Megint azt kell mondanom: nem tudhatjuk biztosan. De ha hiszünk abban, hogy JoJo felelősségteljesen bánik a kulcsaival és gondosan őrzi őket, minden okunk megvan feltételezni, hogy az ő érvényes aláírását ellenőriztük.

A GnuPG barátságosabb felületei

Remélem, nem ijesztettelek meg ezzel a sok kapcsolóval és paranccsal (Isten hozott a Unix világában!). Ez inkább csak áttekintés volt, sok mindenről még nem eshetett szó e cikksorozat keretében. Meggyőződésem, hogy a gpg fontos és hasznos segédeszköz. Ez olyannyira igaz, hogy sokan dolgoznak azon, hogy barátságosabbá tegyék. A hivatalos GnuPG grafikus felületet neve GNU Privacy Assistant (GPA). A program még fejlesztés alatt áll, de nagyon ígéretes.
A Gimp Toolkitre épül, ezért csöppet sem meglepő módon kellemes látványt nyújt.
Természetesen más grafikus felületek is léteznek a gpg-hez: ilyen a Seahorse és a GnomePGP, a KDE Geheimnis nevű alkalmazása és a TkPGP (ezt Tk-val írták, ezért viszonylag ablakkezelő-független). Ezek mellett léteznek még modulok és bővítmények az elterjedtebb levelezőprogramokhoz is. További tájékoztatásért látogass el a GnuPG weblapjára és vess egy pillantást a ***Frontends*** részre.

Összefoglalás

Véget ért a kétrészes bevezetés a GnuPG alapvető használatába. A gpg olyan program, amelyet sokkal többeknek kellene használniuk, ezért kérlek, ne habozz: titkosíts. Pontosabban olyan kulcsokkal titkosíts, amelyeket ismerőseid aláírtak és ellenőriztek.

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

Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition, Bruce Schneier. John Wiley & Sons, 1995.
Dice-Based Password Generation Method (Jelszavak létrehozása dobókockával) ➲ http://www.diceware.com
The GNU Privacy Handbook
➲ http://www.gnupg.org/gph/en/manual.html
Kevésbé bőbeszédű, mint a gpg(1) súgóoldal, de részletesebb, mint ez a cikk.
A hivatalos GnuPG honlap ➲ http://www.gnupg.org. Erről az oldalról tölthető le a GnuPG legfrissebb változata és a GPA, valamint sok hasznos tudnivaló is található itt.
➲ http://www.gnupg.org/download.html/

Vírusellenőrzés a levelezőkiszolgálón

Az Amavis program segítségével elektronikus levelekhez csatolt vírusoktól szabadulhatunk meg.

A levelezőkiszolgálón futó Amavis program, ellenőrzi az átmenő leveleket, lecsatolja a mellékleteket, és egy külső vírusirtó programot futtat rajtuk. Amennyiben vírust észlel, értesíti a kiszolgáló postamesterét, a levél feladóját és a címzettet is. Ilyen esetben az eredeti levél kézbesítését a kiszolgáló megtagadja.
A program használatával sem kerülhető el azonban az ügyfeleken futó vírusirtó alkalmazása, hiszen vírusokat nem csak levelek útján kaphatunk. Ennek ellenére nagyobb biztonságban érezhetik magukat a rendszergazdák és a felhasználók, ha a felismert vírusos levelek az ügyfélgépekig el sem jutnak.
A program telepítése nem egyszerű, de ha összeszedtük a működéséhez szükséges összetevőket és egyéb alkalmazásokat, gyorsan működésre bírható, és karbantartást sem igényel. A telepítés menetét Debian 2.2-es rendszerre írom le. Mivel az Amavis programot Perlben írták, működése csaknem rendszerfüggetlen.

Az Amavis program lelőhelye ➲ http://www.amavis.org/

Előkészületek

Első lépésként szedjük össze az összes szükséges programot. Töltsük le az Amavis-Perl programot
`wget http://www.amavis.org/dist/perl/amavis-perl-11.tar.gz` paranccsal és csomagoljuk ki a */usr/src* könyvtárba:

```
cd /usr/src; tar -xzvvf
  ➥/home/peter/packages/amavis-perl-11.tar.gz
```

Az Amavis-Perl néhány olyan Perl-modulra támaszkodik, amely valószínűleg nincs a rendszerünkön. Kényelmes telepítésükhöz használjuk a Perl CPAN-modulját. Rendszergazdaként írjuk be:

```
perl -MCPAN -e shell
```

Ha hibaüzenetet látunk, töltsük le a CPAN-modult, csomagoljuk ki a */usr/src* könyvtárba és telepítsük:

```
make
make install UNINST=1
```

Az alábbi utasítás kiadása után

```
cd /usr/src; tar -xzvvf
  ➥/home/peter/packages/CPAN-1.59.tar.gz
cd CPAN-1.59
```

itt adjuk ki a

```
perl -MCPAN -e shell
```

parancsot. A megjelenő kérdésekre általában elegendő az ENTER megnyomásával válaszolnunk, kivéve azokra, amelyek a hozzánk közeli tükörkiszolgálót választják ki. Értelemszerűen válasszuk először Európát, majd Magyarországot. Ha sikerült a beállítás, megjelenik a `cpan>` parancssor. Ekkor kezdhetjük el az Amavis README! fájljában leírt Perl-modulok telepítését az `install` *`modulnév`* parancs beírásával. A következőket kell beírnunk:

```
install Unix::Syslog
install Convert::Uulib
install Convert::TNEF
install Compress::Zlib
install Archive::Tar
install Archive::Zip
install G/GB/GBARR/MailTools-1.15.tar.gz
install MIME::Tools
install Bundle::libnet
```

A vírusirtó telepítése

Az Amavis-csomag súgójában olvashatunk a használható vírusirtókról. Én a Sophos vírusirtóját választottam. A Sophos cég által készített csomag sajnos (mint a legtöbb víruskereső program) fizetős, ezért a cég webhelyéről egy csökkentett változatot tudunk letölteni. Ez a változat ugyanúgy megtalálja a vírusokat, de nem távolítja el azokat. Egy nagyobb cégnél ma már elengedhetetlen, hogy komoly vírusvédelem legyen. Töltsük le a programot a ➲ http://www.sophos.com/downloads/products/unix_506.html címről. Linuxhoz libc5-ös és libc6-os változattal működőt is találhatunk. Próbáljuk ki a libc5-öshöz tartozót, ami a ➲ http://downloads.us.shopos.com/products/full/linux.intel.libc5.tar.Z címen érhető el. Letöltés után az `uncompress` paranccsal csomagoljuk ki, majd a `tar` paranccsal irányítsuk az *src* könyvtárba:

```
cd /usr/src; tar -xvvf
  ➥/home/peter/packages/linux.intel.libc5.tar
```

Váltsunk át a létrejött ***sav-install*** könyvtárba, olvassuk el az ***install.txt*** és ***readunix.txt*** fájlokat.
A `./install.sh -v -ni` paranccsal lehet telepíteni. Ha a

Az OpenAntivirus projekt honlapja: ➲ http://openantivirus.prg

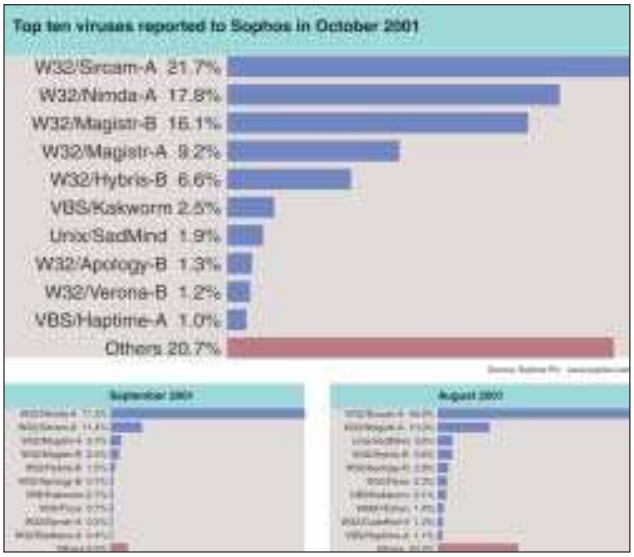
Az Amavis által küldött értesítés vírusos levélről

A Sophos Anti-Virus weboldala ➲ http://www.sophos.com/

`-ni` kapcsoló nélkül telepítjük, egy `sweep`-felhasználót kell létrehoznunk, de ne tegyük, mert fölösleges.
A telepítés a `sweep` programot a */usr/local/bin*-be teszi. A vírusmeghatározó fájlok a */usr/local/sav* könyvtárban vannak, havi frissítéssel jelennek meg, a mostani változat a *vdl-3.51.dat* nevet viseli. A program mindig a *vdl.dat* fájlt keresi, úgyhogy a pillanatnyi vírusadatbázisra egy *vdl.dat* nevű közvetett hivatkozás mutat. Az újabb *dat*-fájlok megjelenése előtt is kapunk frissítéseket *ide*-végződésű fájlok képében, ezeket szintén ebbe a könyvtárba kell helyeznünk. A ➲ http://www.sophos.com/downloads/ide/ könyvtárból az összeset egyenként vagy zipfájlba tömörítve tölthetjük le. Válasszuk a zippelt változatot és bontsuk ki a */usr/local/sav* könyvtárba:

```
cd /usr/local/sav; unzip /home/peter/351_ides.zip
```

Rögtön próbáljuk ki a vírusirtót a `sweep` paranccsal. Ha nem adunk meg fájlnevet, a program rögtön segítséget kínál a használatához. Mostanra letöltöttük az Amavis programot, a hozzávaló Perl-modulokat és egy vírusirtót. Még a levelekhez csatolt tömörített állományok kicsomagolásához szükséges programokat kell beszereznünk: ezek a `file`, `arc`, `bunzip2`, `lha`, `unarj`, `uncompress`, `unrar`, `zoo`.
Telepítsük ezeket is. A `file` parancs valószínűleg már elérhető.



A Sophos folyamatosan figyeli, hogy mely vírusok a legaktívabbak

A többi tömörítőprogramra akkor van szükség, ha a levélmelléklet éppen ilyen formátumú. Az `arc` programot nem sikerült Debian-csomag formájában fellelnem az Interneten, forrásból azonban telepíthető. Ha valaki ehhez nem érez kedvet, az Amavis *configure* parancsfájljában tegye megjegyzésbe az 1573-1575-ös sorokat:

```
#if test "x$arc" = "x" ; then
#{ echo "configure: error: Sorry, you need
#arc" 1>&2; exit 1; }
#fi
```

Ezután rátérhetünk az Amavis telepítésére. Lépjünk be a */usr/src/amavis-perl-11* könyvtárba és gépeljük be: `./configure`. Ha sikeresen lefutott, adjuk ki a `make` parancsot. Az eredményt a `make check` paranccsal lehet ellenőrizni. A folyamat sajnos hibával áll le, ha az `arc` programot nem telepítettük, mivel az ellenőrzéshez használt csatolt állomány *arc* formátumú. Ez azonban nem fog gondot okozni, hiszen manapság senki se használ arc-tömörítést. Hozzunk létre egy Amavis nevű felhasználót, ne adjunk neki parancssoros héjat, sőt – biztonsági okokból – a felhasználót tiltsuk le:

```
adduser amavis
chsh -s /bin/false amavis
passwd -l amavis
```

```
make install
```
Ezzel az Amavis telepítése sikeresen befejeződött.

Az Exim program beállítása

Ezt követően a levelezőkiszolgáló programját kell beállítanunk. Debianhoz az Exim települ fel, ezért ennek a beállítását részletezem, de az Amavis leírásában a többi ismert programhoz is találunk ismertetést. Módosítsuk az Exim beállítófájlját (*/etc/exim.conf*):

1. A `trusted_users = mail` részhez írjuk be az Amavis-felhasználót:
```
trusted_users = amavis:mail
```

2. A `TRANSPORT CONFIGURATION` rész elejéhez illesszük:
```
amavis:
driver = pipe
command = "/usr/sbin/amavis -f
  ➥${sender_address} -d ${pipe_addresses}"
prefix =
suffix =
check_string =
escape_string =
# for debugging change return_output to true
return_output = false
return_path_add = false
user = amavis
group = amavis
path = "/bin:/sbin:/usr/bin:/usr/sbin"
current_directory = "/var/amavis"*
```

3. A `DIRECTORS CONFIGURATION` rész elejére pedig az alábbi sorokat írjuk:
```
amavis_director:
condition = "${if eq
  ➥{$received_protocol}{scanned-ok} {0}{1}}"
driver = smartuser
transport = amavis
```

4. A ROUTERS CONFIGURATION rész elejére:
```
amavis_router:
condition = "${if eq
  ➥{$received_protocol}{scanned-ok} {0}{1}}"
driver = domainlist
route_list = "*"
transport = amavis
```

Mivel valószínűleg a nyomda ördöge sem alszik, jól tesszük, ha a fenti példákat az Amavis leírásában található ***README.exim*** fájlból másoljuk ki. Az ellenőrzéshez írjuk be:

```
tail -f /var/log/syslog
```

Indítsuk újra az Eximet (*/etc/init.d/exim restart*) és próbáljunk vírusos levelet küldeni magunknak. Rendes működésnél a naplófájlokban például ilyen üzenettel találkozhatunk:

```
amavis[3900]: Virus found - quarantined as
virus-20011020-084755-3900
```

Ha nincs kéznél vírusos levél, bármilyen vírusos fájl jó, amit csatolt állományként küldhetünk. Ha ilyen sincs, a ➲ http://www.eicar.org/download/eicar.com címről letölthetünk egy fájlt, amely ugyan nem tartalmaz igazi vírust, csupán egy olyan bájtsorozatot, amit a legtöbb vírusirtó vírusként ismer fel.

A vírusadatbázis frissítése

A Sophos frissen tartásához időnként (havi 1-2 alkalommal) egy teljes változatot le kell töltenünk, a legutolsó teljes változat kiadása óta megjelent új vírusokhoz tartozó ***ide*** állományokat tartalmazó zipfájlt pedig napi rendszerességgel .
Nézzünk erre egy bash-héjprogramot, amely a ***README.scanners*** fájlban található példa módosított változata (a sorszámok csak a magyarázat kedvéért kerültek a sorok elejére):

```
1. #!/bin/bash
2. VIRFILE="'sweep -v|/bin/grep 'Product
      ➥version'| /usr/bin/tr -d -c
      ➥[:digit:]'_ides.zip"
3. SOPHOS_URL='http://www.sophos.com/
      ➥downloads/ide/'
4. IDE_PATH='/usr/local/sav'
5. cd $IDE_PATH
6. /usr/bin/wget $SOPHOS_URL$VIRFILE
7. /usr/bin/unzip -q -n $VIRFILE
8. rm -f $VIRFILE
9. chmod 644
```

A 2. sorban a `VIRFILE` változóba az ***ide***-fájlokat tömörítve tartalmazó állomány neve kerül. Ez általában az ***_ides.zip***-re végződik, az eleje pedig a vírusirtó változatszáma. Ezt a változatszámot kapjuk meg, ha a `sweep` programot elindítjuk a `-v` értékkel:

```
peter@mercury:~$ sweep -v
SWEEP virus detection utility
Copyright (c) 1989,2001 Sophos Plc,
www.sophos.com
System time 10:31:40, System date 20 October
2001

Product version: 3.50
Engine version: 2.6
User interface version: 2.03.079
Platform: Linux/Intel
Released: 01 October 2001
```

A kimenet `Product version` szöveget tartalmazó sorából a `tr` paranccsal az összes nem számjegyet tartalmazó karaktert kiszűrjük, így kapjuk meg a ***350_ides.zip*** szöveget. A parancsfájl többi részében `wget`-tel letöltjük a fájlt, `unzip`-pel kicsomagoljuk, töröljük a letöltött fájlt és beállítjuk a fájljogosultságot. A fenti parancsfájlt naponta a crontabból futtathatjuk. Ha a Sophos új változatot ad ki, inkább azt kézzel töltsük le és telepítsük.

Az Amavis egyéb lehetőségei

Vírus észlelésekor levél érkezik a postamesterhez és a feladóhoz. Ha a címzetteket is értesíteni akarjuk, akkor a `configure` parancsfájlt a `--with-warnrecip=yes` értékkel hívjuk meg. Az értesítésekhez tartozó levélmintákat az Amavis könyvtárának ***amavis/notify*** könyvtárában találjuk. Az itt levő ***admin***, ***recip*** és ***sender*** fájlokat a megfelelő óvatossággal módosítva tetszőleges üzenetet készíthetünk.

Borkuti Péter
(borkuti@freemail.hu) matematika-informatika szakos tanár, rendszergazda, informatikus, rendszerépítő és programozó.

Woodyra várva

Szárnyakat bontott a Debian új pingvinje, amely már a 3.0-s változatszámot viseli és a fedőneve: Woody. E cikk írásának pillanatában még csak a próbaváltozat érhető el, így ezt vettük nagyító alá.

A Debian GNU/Linux sokban különbözik a RedHattől és társaitól. Sokak szerint például abban, hogy a Debian kevésbé felhasználóbarát változat, amelynek telepítése és beállítása bárki számára komoly feladatot jelenthet. Igaz, a RedHat, a SuSE, a Mandrake és a hozzájuk hasonló terjesztések esetében már tízpercnyi telepítés után kész, előre beállított rendszer fogad minket, amelyet azonnal munkára foghatunk. A Debiannál más a helyzet: a telepítés hosszadalmasabb, „macerásabb", de végeredményként az adott feladatra tökéletesen testreszabott és beállított rendszerhez jutunk.
A Debian nem üzleti vállalkozás, készítői csupán kedvtelésből fejlesztik. Az említett változatokkal szemben ez is fontos különbség, ugyanis a Debiant nem az üzleti érdek vezérli, hanem a szakemberek szemszögéből kínál megoldást a különböző feladatokra, ezért is nagyon népszerű a Linuxot már behatóbban ismerő felhasználók körében.
A kezdő felhasználóknak általában nem ajánlják, hogy a Debian legyen életük első Linuxa, mivel a telepítéshez elkelhet némi linuxos tapasztalat. Ezért mi is azt javasoljuk, hogy a Linuxszal való első ismerkedéshez először valamelyik, kifejezetten a kezdő felhasználókat megcélzó változattal próbálkozzunk, például Mandrake-kel vagy SuSE-vel. Akik azonban elég bátrak ahhoz, hogy kipróbálják a Debiant, azoknak e cikkben bemutatjuk a telepítés legfontosabb lépéseit.

A Debian gyökerei

A Debian története 1993 augusztusára nyúlik vissza, amikor „alapító atyja", ***Ian Murdock*** elhatározta, hogy új nyílt Linux-változatot hoz létre. Így született meg a Debian elnevezés, amely két – Ian és felesége, ***Debra Murdock*** – keresztnév ötvözéséből állt össze. Murdock a GNU-projekt szellemében kívánt valami újat létrehozni. Ez lett az 1984-ben elindított projekt neve, amely nem kisebb célt tűzött maga elé, mint egy vadonatúj, nyílt, Unix-típusú operációs rendszer létrehozását. A GNU legelterjedtebb változatai a Linux-rendszermagot használják, habár már készülőben van a GNU saját magja is, amely a Hurd nevet kapta. A Hurd még fejlesztés alatt áll, de ha egyszer elkészül, akkor elképzelhető, hogy a Linux komoly vetélytársa lesz. A Debianban található alapprogramok nagy része ebből a GNU-projektből származik, erre utal a GNU/Linux név is. Természetesen a Debian együttműködik a többi Linux-változattal.

Jelen

Jelen pillanatban a Debian üzembiztos változata a Potato fedőnevű 2.2-es. A Woody – amelyről cikkünk is szól – jelenleg még csak próbaszakaszban található, azaz jócskán lehetnek benne eddig még fel nem derített hibák. Ennek következtében a Debian semmiféle biztosítékot nem ad a rendszer helyes működésére, mindenki csak saját felelőségére használja! A Debiannak létezik már a Woodynál is „újabb" változata, amely SID néven került a köztudatba. (A SID a Stil In Development szavakból tevődik össze, jelentése „fejlesztés alatt", tehát a SID-ből sohasem lesz üzembiztos változat.)

Mit is tartalmaz a Woody?

Az operációs rendszer öt és fél CD-n „terpeszkedik el", habár a legfontosabb alkalmazások az első három korongon elférnek. Az alapmag a 2.2.19-es, de természetesen a 2.4-es sorozat is helyet kapott benne. A rendszer alapkönyvtárának szerepét a GLibC 2.2-es tölti be, a grafikus rendszer motorja pedig az Xfree 86 4.1-s változata. Ezenkívül megtaláljuk benne a KDE 2.1.1-et és a Gome 1.4-et is.

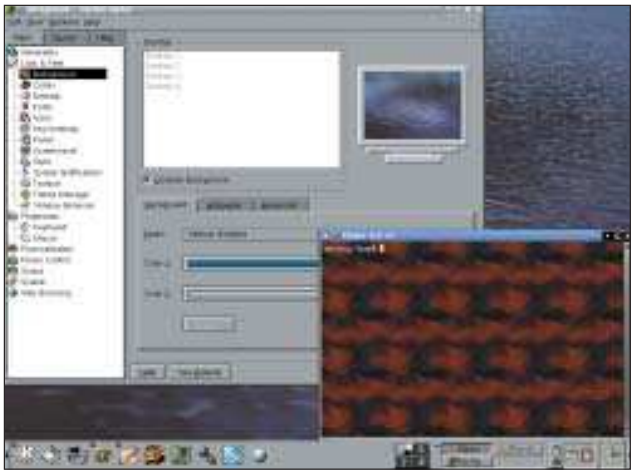
Felépítés

A Potatóhoz hasonlóan a csomagok feltelepítése itt is az APT (Advanced Package Tool) segítségével történik. Az APT ugyan még fejlesztés alatt áll, ámde nagyon hatékony csomagkezelő rendszer. Segítségével például akár az Internetről, akár CD-ről egy parancs kiadásával felpakolhatunk egy Debian-csomagot úgy, hogy az adott csomaghoz szükséges további csomagok is telepíthetők. Ezenkívül alkalmazásainkat, sőt akár az egész rendszert egy utasítás segítségével frissíthetjük.

A telepítés és a beállítás

A Debian Woody telepítése, hasonlóan a Debian GNU/Linux korábbi változataihoz, két részből épül fel. A telepítés első szakaszában csak egy alaprendszer kerül a merevlemezünkre, amely ugyan már működőképes, de a legfontosabb csomagokon kívül semmi egyebet sem tartalmaz. Ennek az alaprendszernek a telepítése után immár új Linuxunk alól kezdhetjük meg a telepítés második szakaszát, azaz a szükséges csomagok kiválasztását és telepítését.
A Debian telepítőjét a legegyszerűbb az első CD-ről elindítani, de a telepítést DOS alól is elkezdhetjük a CD ***INSTALL*** könyvtárában található ***boot.bat*** indításával. Telepítés közben szabadon választhatunk a különböző menüpontok között, a telepítő azonban mindig felajánlja a következő logikus lépést.
Első komoly feladatunk a merevlemez felosztása lesz, itt azonban ugyanazokat az elveket kell követnünk, mint a többi Linux-telepítés során: létre kell hoznunk egy csereterületet és legalább egy linuxos lemezterületet. A csereterület méretének meghatározására általában az a bevált módszer, hogy meglévő fizikai memóriánk méretének a kétszeresét vesszük, de a csereterület méretének nem érdemes meghaladnia a 128 MB-ot (minél több memóriánk van, annál kevesebb csereterületre lesz szükségünk). A rendszer lemezrészének mérete a rendszer feladatától függ. Egy átlagos rendszer mérete körülbelül 500 MB, de például egy levelezőkiszolgálóé gyakran még a 100 MB-ot sem éri el. Egy otthoni rendszer mérete viszont a feltelepített alkalmazásoktól függően akár az 1 GB-ot is meghaladhatja.
A többi Linuxhoz hasonlóan itt sem kötelező az egész rend-

szert egy lemezterületen elhelyezni. A Linux szempontjából mindegy, hogy a rendszert elsődleges vagy logikai lemezterületen helyezzük-e el. Ha régebbi (1998 előtti) BIOS-t birtoklunk, egy dologra vigyáznunk kell: ha az indításvezérlőt (LILO) a merevlemez rendszerindító területére (Master Boot Record – MBR) telepítjük, és a rendszer merevlemezünk felső területein (az első



Kde-felület

1023 cilinder feletti részében) található, akkor a LILO nehézségekbe ütközhet. Ezt a gondot úgy kerülhetjük el, hogy a */**boot*** könyvtárat, amelyben operációs rendszerünk rendszermagja is fellelhető, a merevlemezünk elején létrehozott 10–20 MB-os részen helyezzük el (ez biztonsági szempontól is előnyös megoldás). Természetesen a jelenlegi BIOS-oknál ezzel már nem kell foglalkoznunk, a LILO merevlemezünk bármelyik „zugából" el tudja indítani rendszerünket.

A merevlemez felosztása a `cfdisk` nevű menüvezérelt programmal zajlik. Használata könnyen elsajátítható, így most nem térünk ki rá. Két dologra vigyázzunk: ne felejtsük indíthatóvá tenni a ***Bootable*** gomb segítségével, és a ***Write*** gombbal menteni ténykedésünk eredményét.

Ezután következik a csereterület felélesztése és a lemezterületek formázása. Ha több linuxos lemezterületet készítettünk, akkor először azt formázzuk le, amelyiket fő (`/`) lemezterületnek szánunk.

Ha elkészültünk, következhet a rendszermag és az eszközfájlok telepítése. A megjelenő listából válasszuk ki a CD-ROM-ot, majd az ezután megjelenő párbeszédablaknál nyomjunk ENTER-t.

Ezt követően lehetőségünk nyílik a különböző magmodulok betöltésére. A Linux rendszermagja modularizált, azaz egyes eszközök támogatása nem magában a magban, hanem egy külön egységben (modulban) található. Ezeket a modulokat a felhasználó kedvére ki- és betöltheti. A modularizált mag kevesebb helyet foglal a memóriában, mivel az adott modul csak akkor töltődik be, amikor valóban szükség van rá. Itt most csak azokat a modulokat töltsük be, amelyeket állandóan használunk. Gyakran az értékeiket is meg kell adnunk, de ez alól a PCI-os eszközök általában kivételt képeznek. Hogy melyik modulnak milyen értéket adhatunk, arról a rendszermag leírásában olvashatunk részletesen. Ha elkészültünk, az ***Exit*** menüponttal léphetünk tovább.

A következő lépés gépünk nevének (***hostname***) beállítása, és ha gépünk közvetlenül csatlakozik valamilyen hálózathoz, akkor a hálózati adatokat is itt adhatjuk meg. Ilyen például az IP-cím, a hálózati maszk, az átjáró és a DNS-kiszolgáló címe. Ezeket az adatokat hálózatunk rendszergazdájától tudakolhatjuk meg.

Ha ezen is sikeresen túljutottunk, a telepítő elkezdi feltelepíteni a legszükségesebb csomagokat. Az alaprendszer a merevlemezen körülbelül 80 megányi helyet foglal el.

A telepítés első szakaszának lezárásaként a LILO telepítése kerül sorra. Ha csak Linux található a számítógépünkön, vagy az indítani kívánt operációs rendszert a LILO-val szeretnénk kiválasztani, telepítsük az MBR-be. Amennyiben Linuxunkat más indításvezérlővel szeretnénk indítani, a linuxos lemezterületünk indítórészébe helyezzük el. Itt nyílik lehetőségünk arra is, hogy a LILO-hoz további operációs rendszereket adjunk, amelyek közül majd a számítógép bekapcsolásakor választani lehet.

Ha a LILO használatát el szeretnénk kerülni, a telepítővel egy indítólemezt is készíttetnünk kell. Azt javaslom, hogy ezt a lépést akkor is végezzük el, ha a LILO-t már telepítettük, ugyanis ha a Linux indításvezérlője megsérül vagy törlődik, rendszerünket lemezről akkor is el lehet indítani.

WindowMaker-felület

A gép újraindítását követően már egy működőképes Debian GNU/Linux csücsül a gépünkön, csak épp az alkalmazások hiányoznak még róla. Linuxunk első indulásakor fogadjuk el a Debian fejlesztőinek köszöntését a sikeres telepítés örömére, de hátradőlni még nincs okunk, hiszen akad dolgunk bőven.

Első feladatunk rendszerünk néhány alapvető tulajdonságának meghatározása lesz. Első ízben döntenünk kell, hogy rendszeróránk a GMT (Greenwich Main Time – greenwichi idő) szerint járjon-e. Ezt csak akkor válasszuk, ha masinánkon a Linux az egyedüli operációs rendszer. Ha rendszeróránk a GMT szerint jár, akkor a nyári, illetve a téli időszámítás-váltás önműködően zajlik.

A következő, amiben választanunk kell, az MD5 titkosító algoritmus használata a felhasználói jelszavak tárolására. Az MD5 lehetővé teszi, hogy nyolc karakternél hosszabb jelszavakat is használhassunk. Érdemes erre a kérdésre igennel felelni. A árnyékjelszófájl (shadow) használatára is igennel válaszoljunk, mert így a titkosított felhasználói jelszavakhoz csak a rendszergazda férhet hozzá. Tévedés ne essék: a rendszergazda sem tudja visszafejteni a jelszavakat, mivel ma még nem létezik olyan algoritmus, amely elfogadható időn belül elvégezné a jelszavak feltörését, de az árnyékjelszófájl megakadályozza, hogy a gyenge jelszavakat (például ugyanaz a jelszó, mint az azonosító stb.) a próbálgatásos módszer segítségével megfejtsék. Ezek az óvintézkedések nagymértékben növelik rendszerünk biztonságát.

A következő lépés a rendszergazda jelszavának beállítása. Megint csak rendszerünk biztonságára nézve törekedjünk arra, hogy ne könnyen megfejthető jelszót válasszunk. Ne feledjük: a jelszavak visszakereshetetlenek. Ezután alkalmunk nyílik egy saját felhasználó létrehozására is – ne hagyjuk ki ezt a lépést. Alapszabály, hogy a rendszerben rendszergazdaként csak akkor dolgozzunk, amikor valami olyan műveletet hajtunk végre, amely rendszergazdai jogosultságot igényel (például csomagok telepítése, eltávolítása stb.).
Ezt követően a modemes internetelérés beállítására is lehetőségünk nyílik, ha az a telepítéshez szükséges. Mivel most a CD-ről való telepítést mutatjuk be, itt nemmel válaszoljunk. A későbbiek során a PPP-kapcsolatot a ***pppconfig*** nevű csomag segítségével könnyedén beállíthatjuk. Ilyenkor a beállításhoz a `pppconfig` nevű programot kell indítanunk, ahol majd meg kell adnunk a kapcsolatunk nevét, a szolgáltatónk telefonszámát, továbbá a felhasználói nevünket és a jelszavunkat. Modemünket az alkalmazás önműködően megkeresi. Csatlakozni a `pon` *`kapcsolatnØv`* utasítással fogunk, a kapcsolat bontására pedig a `poff` parancsot használhatjuk.
Ezután az APT beállítása következik. A Woodyban minden csomag telepítése az APT-vel zajlik, így a CD-ken található csomagokat a gépünkre szintén ennek a segítségével pakolhatjuk fel. Ezért itt minden CD-t végig kell nézetnünk és a rajtuk található összes csomag bekerül az APT adatbázisába. Ha készen vagyunk, egyéb APT-forrást is felvehetünk, például internetes Debian-tükröket.
Most kerülhet sor a telepítendő csomagok kiválasztására. Elsőként lehetőségünk nyílik a `tasksel` nevű alkalmazás használatára, amellyel előre elkészített beállítások közül választhatunk, megkímélve magunkat a csomagok egyenkénti válogatásától. Ezután a `dselect` program segítségével „finomíthatunk" ezen a beállításon. Az általunk kipróbált Woodyban a `tasksel` sajnos nem volt hajlandó elindulni, így a `dselect`-ben minden szükséges csomagot kézzel kellett kiválogatni. Reméljük, ezt a hibát hamarosan kijavítják.

A dselect használata

A `dselect` program használata először kicsit nehézkesnek tűnhet, pedig rendkívül hatékony alkalmazás. Mivel kezelésének elsajátítása nem könnyű feladat, szóljunk erről is pár szót, habár a ? megnyomásával is részletes útmutatást kaphatunk. Minden sor egy-egy csomagról ad tájékoztatást. Az első oszlopban a csomag pillanatnyi állapotát láthatjuk, például a már előzőleg feltelepített vagy telepítésre, eltávolításra, illetve megsemmisítésre (`purge`) kijelölt. A megsemmisítendő csomagoknál a különböző beállítóállományok is törlődnek. Ha ez a jelölésrendszer túl bonyolult, a V billentyű segítségével javíthatunk a helyzeten. Egy csomag állapotán az INS, – és _ billentyűkkel tudunk változtatni. Az első a csomagot telepítésre jelöli ki, az utolsó kettő pedig törlésre, illetve megsemmisítésre.
A második oszlop a csomag minősítését tartalmazza, például hogy fontos alapcsomag-e vagy csupán kiegészítő. Ezt a besorolás követi, amely tulajdonképpen azt mondja meg, hogy valójában mi is az a csomag (például leírás, hálózati alkalmazás, programozási könyvtár stb.). Ezt követően jön a csomag neve, majd a feltelepített és az elérhető változat száma. Végezetül pedig egy rövidebb leírást olvashatunk róla.
A `dselect` – mint minden rendes csomagkezelő – figyel a függőségekből adódó gondokra. Ha például grafikus alkalmazást szeretnénk telepíteni, azonban a grafikus rendszert még nem telepítettük, egy új ablakot kapunk, ahol listát láthatunk a további szükséges, illetve ajánlott csomagokról is. A `dselect` általában önműködően kijelöli a csomag működéséhez elengedhetetlen további csomagokat, ezért az ilyen ablak megjelenésekor csak az ENTERT kell buzgón nyomogatnunk. Előfordulhat azonban az is, hogy egy csomag puszta léte a feltelepítendő csomagot megzavarja a helyes működésben. A `dselect` az ilyen esetleges ütközésekre is figyel, bár ezeket nem mindig tudja orvosolni, emiatt gyakran nekünk kell végiggondolnunk, mi is lenne számunkra az eszményi megoldás.
Ha ezzel is készen vagyunk, nyomjunk ENTER-t, és a telepítés kezdetét veszi. A telepítő először az összes csomagot kicsomagolja, majd csak ezután áll neki a beállításuknak. Készüljünk fel lelkileg, hogy számos kérdést kell majd megválaszolnunk. Például itt kerül sor a grafikus környezet beállítására is, amit szintén menüvezérelt alkalmazás segítségével tehetünk meg.

Grafikus rendszer

Ha már a grafikus rendszernél tartunk, a Woodyban a 3.x-es és a 4.x-es változatú X-kiszolgálók is megtalálhatók. Az X-kiszolgáló a Linux grafikus rendszerének a lelke, ez tartja a kapcsolatot a grafikus programok (X-ügyfelek) és a gépünk között.
A 3-as változatnál minden videokártya-típusra külön X-kiszolgáló létezett, a 4-esnél pedig csak egyetlen X-kiszolgáló maradt, ugyanis a grafikus rendszer modulokból áll. Ha tehát a 3.x-es X-et szeretnénk használni, a megfelelő X-kiszolgálót a `dselect`-ben kell kiválasztanunk. Amennyiben a 4-esre esett a választásunk, az `xserver-xfree` nevű csomagot kell felraknunk.
Ha több X-kiszolgálót is telepítettünk, ki kell választanunk, hogy melyiket szeretnénk használni. Ha a 4-es mellett döntünk, a beállításhoz az `xf86cfg` grafikus programot is használhatjuk. Tapasztalataink szerint a 4-es X is elég megbízhatóan működött, bár a Woody fejlesztésének ebben a szakaszában néhány alkalmazás még nem dolgozott alatta tökéletesen, míg a 3-ason gond nélkül futott. Akadtak azonban egyéb kisebb hibák is, például az, hogy a soros kapura csatlakozó szabványos Microsoft-egér időnként teljesen lefagyott.
Elérkeztünk a Woody telepítésének végéhez, most már akár be is jelentkezhetünk a rendszerbe. Ha további csomagokat is telepíteni szeretnénk, azt vagy a `dselect` segítségével, vagy ha ismerjük a csomag pontos nevét, az `apt-get install` *`csomagnØv`* utasítással tehetjük meg.

Tapasztalatok

Ahogyan már említettük, a Woody jelenleg próba alatt áll. Ehhez képest még mindig elég sok zavaró hiba található benne: a `tasksel` nem hajlandó működni, s a grafikus rendszerben kisebb hibák rejtőznek, ez azonban csak egy része a gondoknak. Például elég sok zavarba ejtő hibaüzenetet kapunk telepítés közben, továbbá az állandóan jelentkező ***Neighbour table overflow*** rendszermag hibaüzenetet sem tudtam mire vélni – habár ez utóbbi jelenség a rendszermag újrafordításával megszüntethető.
Remélem, ezeket a hibákat hamarosan kiküszöbölik. Leszámítva e zavaró tényezőket, meg kell állapítanunk, hogy a Woody eléggé megbízhatóan működött. Sőt, úgy tűnt, hogy valamivel gyorsabb is, mint a nemrég megjelent Mandrake 8.1-es.
Tapasztalatainkat összegezve: a Woody tökéletes választás akár kiszolgálónak, akár otthoni operációs rendszernek. Lehet, hogy a telepítése kissé körülményes, de az APT-rendszernek köszönhetően a frissítése gyerekjáték. Reméljük, az üzembiztos Woody-kiadásra már nem sokáig kell várnunk, és reméljük azt is, hogy hamarosan kinövi gyermekbetegségeit.

Zaphod, zaphodmail@freemail.hu

Bemutatkozik a Tkinter

Grafikus felületek készítése Pythonból gyorsan és egyszerűen.

A Tkinter segítségével Pythonban írt alkalmazásainknak könnyedén varázsolhatunk egyszerűen kezelhető grafikus felületet, mely – a Pythonnak köszönhetően – nemcsak Linux alatt, de Windowson vagy bármely Python által támogatott felületen elérhető, és a már megírt kód változtatás nélkül bárhol lefut. Mindemellett az egyes felületeken programjaink is egységesen néznek ki, eltekintve a betűtípusok miatt kialakuló különbségektől.

A Tkinter a grafikus felületek elkészítéséhez a Tcl/Tk-t hívja segítségül. Mint a legtöbb Python-bővítmény, a Tcl/Tk-hoz kapcsolódó Tkinter is C-alapokra épül. Megvalósítási szinten a Tkinter két rétegre bontható: a C-ben írt _tkinter programkönyvtárra, valamint a már Pythonban írt ***Tkinter.py***-re és az ehhez kapcsolódó egyéb Python modulokra. A _tkinter kapcsolja össze a Python-modulokat a Tcl/Tk-val, de a programozó elől ezt a réteget teljesen elrejti a ***Python.py***, azaz az egyetlen olyan modul, melyet a tkinteres alkalmazásaink fejlesztéséhez valóban ismernünk kell.

A Mix-inekről

A Mix-inek, azaz a bekeveredő osztályok a Tkinter szerves részét képezik. Mint neve is utal rá, a Mix-in-osztályok az öröklődés során nem elsődleges ősként vannak jelen, hanem az alapos tulajdonságait bővítik ki bizonyos feladatfüggő szolgáltatásokkal, azaz saját tulajdonságaiknak az elsődleges ős tulajdonságaihoz történő keverésével. Az így létrejövő gyermekosztály mind az elsődleges ős, mind a hozzákevert Mix-in-ős tulajdonságait tartalmazza. Gyakran előfordul, hogy egy Mix-in-osztály egyetlen más osztálynak sem elsődleges őse. (A Mix-inekről bővebben a *Linuxvilág* áprilisi számában olvashattatok.)

A Tkinter felépítése

A Tkinter felépítése, ellentétben egyéb GUI-rendszerek osztályrendszereivel, nagyon egyszerű. A szerkezetben a GUI-elemek egy úgynevezett alapelemre (basewidget) épülnek, amelyhez az elemek elhelyezéséért felelős osztályok (Pack, Place és Grid), illetve a Tkinter működését befolyásoló egyéb osztályok Mix-inként kapcsolódnak (lásd alább).

A Tkinter használata

Először is készítsük el a már jól ismert „Szia Világ!” program tkinteres változatát (*1. lista*). A program legelső sorában a Tkinterből használt elemeket vezetjük be a `from Tkinter import Label, mainloop` paranccsal. Ha korábban már foglalkoztál Pythonnal, különösnek tűnhet ez a sor. Vajon miért használjuk a `from modulneve import` írásmódot a már jól bevált `import modulneve` helyett? Nos, képzeljük csak el, milyen fárasztó lenne, ha a programunkban minden Tkinter-elemre hivatkozáskor be kellene írogatnunk, hogy az adott elem mely modulból származik. Mivel azonban a programozók kényelmes emberek, egyszerűbb megoldást találtak ki helyette, amely az adott modul névteréből minden elemet saját programunk névterébe emel át. Bánjunk vele nagyon óvatosan, mert ezáltal minden modul minden tagfüggvénye (method) egy névtérbe kerül, és előfordulhat, hogy egy azonos nevű, bár más feladatú tagfüggvény több modulban is megtalálható, így semmiképpen sem ajánlott több modul hasonló elvű beillesztése.

Lépjünk tovább a második sorra! Ebben nincsen különösebb magyarázatra szoruló rész. A `Label`-lel címkét hozunk létre, benne a jól ismert szöveggel, majd a címke `pack()` tagfüggvényével az elemet alkalmazásunk ablakába helyezzük. Az utolsó sorban a vezérlést a felhasználónak adjuk át. A `mainloop()` a felhasználó és a rendszer által keltett eseményeket dolgozza fel, pontosabban ez a szolgáltatás az esetleges billentyűleütéseket „csípi el”, illetve az ő dolga az is, hogy elemeinket újrarajzolja, ha azok előzőleg valamilyen okból takarásba kerültek.

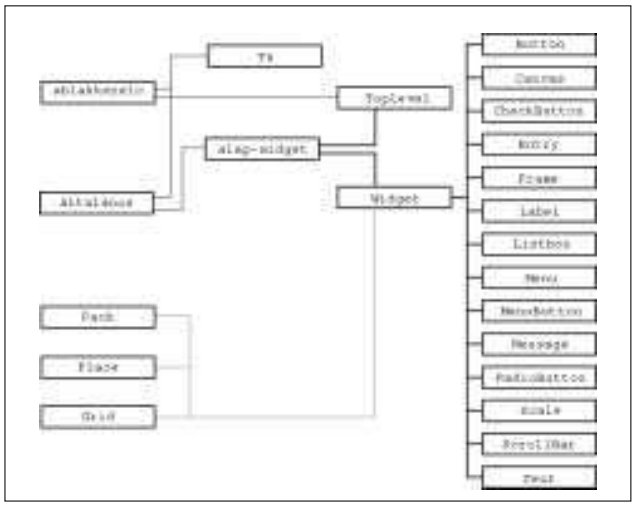
Egy fejlettebb program

Mint előző példánkban láttuk, pillanatok alatt létrehozhatunk olyan programot, mely a felhasználó számára adatot közöl. Igen ám, de mi történik olyankor, amikor a felhasználót nemcsak tájékoztatni szeretnénk valamiről, hanem a véleményére is kíváncsiak vagyunk? Példánkban programunkat egy gombbal bővítjük ki, amelyre kattintva megváltoztathatjuk az ablakban megjelenő szöveget (*2. lista*).

Ez a program már valamivel bonyolultabb. A legfontosabb különbség, hogy ezúttal egy osztályt hoztunk létre, melynek `__init__()` eljárása az objektum megalkotása után önműködően meghívódik, s az ablakunkat, illetve az abban megjelenő gombot ez az `__init__()` eljárás hozza létre. A gombon kezdetben a megszokott „Szia Világ!” szöveg található, majd ha rákattintunk, programunk ***Guidó***-nak, a Python megalkotójának köszön.

A program legelső sorában bevezetjük a Tkinter-modul összes elemét. Ezt követően a futás a 13. sorra ugrik, ahol azt vizsgáljuk, vajon mi is csak egy modul vagyunk-e vagy pedig önálló programként futunk. Ha önállóak vagyunk, osztályunkat létrehozva annak `mainloop()` eljárását nekünk kell meghívnunk. A `mainloop()`-ot ebben az esetben a Frame-osztálytól örököljük – erről nem elfeledkezve hívjuk meg szülőosztályunk `__init__()` eljárását is (még a `mainloop()` előtt).

A 7. sorban egy gombot hozunk létre, és az ezt követő két sorban ennek tulajdonságait módosítjuk, végül a `pack()` eljárással életre is keltjük. A 9. sorban azt állítjuk be, mi történjen, amikor valaki a gombunkra kattint. Esetünkben ilyenkor osztályunk `hello_callback()` eljárása hívódik meg. Ez az eljárás – talán már feltűnt – a programból közvetlenül sehol sem hívódik meg, e feladatot a `mainloop()`-ra bízták. Ha valaki a gombon kattint, egy esemény keletkezik, amelyet a `mainloop()` kap el, és az adott elemhez tartozó ***command*** tulajdonságból kiemeli, milyen eljárást kell ilyenkor hívnia. Fontos tehát, hogy lássuk: az `__init__()` eljárás még a `mainloop()` előtt hajtódik végre, a `hello_callback()` eljárást viszont már a `mainloop()` hívja meg. Miként a kife-



1. kép A Tkinter felépítése

2. kép Az IDLE működés közben

jezés maga is utal rá, az ilyen eljárásokat callback, azaz visszahívó eljárásoknak nevezzük. Bármilyen grafikus felület létrehozásakor programunk logikája mindig az ilyen visszahívó eljárásokra épül, általuk értesülünk arról, hogy a felhasználó éppen mit szeretne, hol kattint, milyen gombot nyomott le, vagy éppen merre halad el az egérmutató.

A Tkinter egyéb lehetőségei

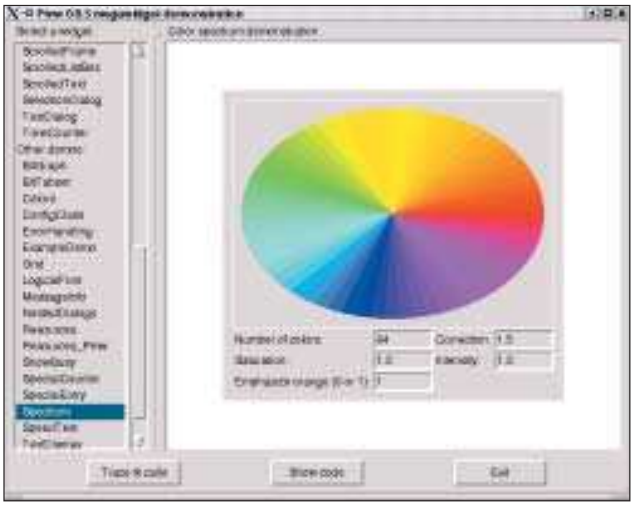
Az eddig felsoroltakkal még nem merítettük ki a Tkinter lehetőségeit. Valójában beállítások és szolgáltatások egész tárháza áll a rendelkezésünkre ahhoz, hogy szinte bármilyen elképzelhető alkalmazást létrehozzunk, az egészen egyszerűektől a legbonyolultabbakig. Az eszközök a felhasználói programoktól kezdve a játékok vagy a multimédiás programok elkészítéséig adottak.

A Tkinterhez létezik egy PMW nevű modul, amellyel előre elkészített, bonyolultabb GUI-elemeket kapunk kézhez.

A PMW-ben az összes olyan eszköz megtalálható, amelyre hétköznapi programjaink során szükségünk lehet, ideértve a menüsorokat, a görgetővel ellátott szerkesztőablakokat vagy az egérmutató mellett megjelenő „segítséglufikat”, ezért a beszerzése mindenképpen ajánlott, mert így rengeteg vesződségtől kímélhetjük meg magunkat.

A Tkinter nem elhanyagolható tulajdonsága az sem, hogy a segítségével pillanatok alatt készíthetünk olyan programokat, amelyek C-ben vagy C++-ban megírva hosszú-hosszú órákon vagy akár napokon keresztül tartanak. A Tkintert éppen ezért előszeretettel használják arra is, hogy úgynevezett „mintapéldány-alkalmazásokat” készítsenek benne. Ez azt jelenti, hogy mielőtt egy cég pénzt és energiát ölne hosszú hónapokig tartó fejlesztésekbe, a programot előtte elkészítik Pythonban, és így előre kialakítható és kipróbálható az alkalmazás felülete. Az esetleges módosítások néhány mozdulattal elvégezhetők, majd amikor kialakult a végleges programterv és megbizonyosodtak arról, hogy az alkalmazásra valóban igényt támasztanak, az egészet egy nem értelmezett, hanem gépi kódban futó és ezáltal gyorsabb programnyelvre ültetik át.

Sőt, a Tkinter úgynevezett „ragasztó”-alkalmazások készítésére is alkalmas: a sebességnövelés érdekében vagy más okból kifolyólag valamilyen egyéb nyelven megírt programot, esetleg programokat foghatunk össze vele egyetlen felület alatt.



3. kép Példa egy PMW-alkalmazásra

1. lista Egyszerű „Szia Világ!” program

```
1. from Tkinter import Label, mainloop
2. Label(text='Szia VilÆg!').pack()
3. mainloop()
```

2. lista „Szia Világ!” – kicsit bonyolultabban

```
1. from Tkinter import *

2. class hello(Frame):
3.     def __init__(self):
4.         Frame.__init__(self)
5.         self.pack()
6.         self.master.title('Szia')
7.         self.button = Button(self)
8.         self.button['text'] = 'Szia VilÆg!'
9.         self.button['command'] =
               ➥self.hello_callback
10.        self.button.pack()
11.    def hello_callback(self):
12.        self.button['text'] = 'Szia Guido!'
13.if __name__ == '__main__':
14.    hello().mainloop()
```

Ha a Python összes lehetőségét kiaknázó C/C++ programokat szeretnénk írni, rendelkezésre állnak a szükséges könyvtárak, amelyekkel hozzáférhetünk a Pythonban már megszokott típusokhoz és elemekhez is.

A Python IDE

Létezik egy szintén a Tkinter felhasználásával készített program, melynek jó hasznát vehetjük Python-programok írása során. Ez a felület – egyébként IDLE-nek hívják – ***Guido van Rossum***-nak, a Python atyjának a műve. Segítségével feltérképezhetjük a szerkesztett modul osztályrendszerét, az olvashatóság végett a kódot a logikai egységeknek megfelelően színezi, így az egyes részek áttekinthetőbbek és egyszerűbben módosíthatók, illetve még egy sor olyan hasznos szolgáltatással rendelkezik, amelyek jól jönnek, ha Python-kódot szerkesztünk. Segítségükkel például egyszerűen változtathatunk a behúzásokon, vagyis az identáción, amely azt határozza meg, hogy egy sor, illetve függvény hova, melyik függvénybe, ciklusba vagy osztályba tartozik. Aki szerkesztett már Python-kódot egyszerű szövegszerkesztőből, az tudja, mennyi vesződséggel jár, ha egy kódrészt egy más beosztású részbe akarunk átcsoportosítani, és soronként kell a tabulátorokat kitörölni vagy éppenséggel tabulátorok garmadáját kell minden sor elé beszúrni. Nos, az IDLE-ből ez sem gond.

A Tkinter telepítése

Az 1.5.2-es Python-változattól kezdődően a Tkinter az alapcsomag része. Ha a rendszerünk mégsem tartalmazná, a Python webhelyéről letölthetjük (lásd ***Kapcsolódó címek***).

A Debian-felhasználók helyzete a legkényelmesebb, ugyanis az `apt-get install python-tk` parancs kiadása után a csomag, illetve szükség esetén az egész Python-alaprendszer telepítődik, továbbá a ***python-pmw-*** és ***idle***-csomagok szintén ilyen módon telepíthetők a gépünkre.

Összegzés

A Tkinterrel nagyon gyors és egyszerűen használható eszközt kapunk a kezünkbe, amellyel pillanatok alatt elkészíthetjük a mindennapi munkánkat segítő programokat, vagy ha eléggé magabiztosak vagyunk, akár mindentudó szuperalkalmazások írásába is foghatunk – a lehetőségek adottak.

Gludovátz Gábor
(ggabor@sopron.hu)
1996 óta foglalkozik Linux-rendszerekkel. Egyik kedvenc időtöltése a programozás, jelenleg éppen egy C++-ban írt KDE-s játékon dolgozik, de szívesen kódol Pythonban és PHP-ben is. Honlapja ➲ http://www.sopron.hu/~ggabor/

Kapcsolódó címek

Python ➲ http://www.python.org
Tkinter on-line tanfolyam
➲ http://www.pythonware.com/library/tkinter/introduction/
PMW – Python megawidgets ➲ http://pmw.sourceforge.net/

Csomagfrissítések

David élvezi, amikor pörögnek a dolgok – ingyenes orvosi program, CD-írás.

Miközben ezt a cikket írom, olybá tűnik, hogy az összes játékos új csomagokkal száll ringbe. Végigtanulmányoztam a kedvenceimet, és noha honlapjaik továbbra is hiányosak, korábbi állapotukhoz képest mégis jelentős javulást mutatnak. Ez alkalommal mégis úgy tapasztaltam, hogy a felhasználói közösségben a megelőzőekhez képest sokkal több zúgolódás és elégedetlenség tapasztalható. Az új kiadások megjelenését mindig egyfajta örömünnepként éltem meg, hiszen a gondot mások vették a nyakukba, de számomra is mindent frissítettek. A legutóbbi alkalommal rengeteg zokszó volt hallható amiatt, hogy a csomagok telepítése után az égvilágon minden be van kapcsolva (Telnet, FTP stb.). A csomagok készítői ezeket a panaszokat is meghallgatták, és szűkítettek a szolgáltatásokon. Most ugyanazt a jajveszékelést hallom, ám amiatt, hogy semmi – vagy szinte semmi – nincs bekapcsolva alapállapotban. Bizonyos dolgok, azt hiszem, sosem változnak. A magam részéről élvezem a pillanatot, amikor felderíthetem, hogy a mérnökök ez alkalommal vajon hová rejtették el az összes parancs- és beállítófájlt. Lassan kiöregszem belőle, mégis úgy érzem magam. Rajta, linuxozni akkor is jó!

EthStatus

Ha a rendszerben található egy vagy több ethernethálózati csatoló állapotát szeretnéd megfigyelni, ez az apróság neked készült. Grafikusan és számértékekkel is megjeleníti, hogy a csatoló éppen mit csinál, valamint, hogy éppen működik-e vagy sem. A szerző egy kiegészítővel a program rendszertöltéskor történő indításában is segítséget nyújt. Nem gond, ha nem az ajánlott Linux-változatok valamelyikét használod, nyugodtan felteheted. A kimenetet kiküldheted használaton kívüli terminálra, így bárki bármikor felügyelheti a rendszert. Futtatásához szükséges: libncurses, glibc.
➲ http://ethstatus.calle69.net

multiCD

Előfordul, hogy egyszerre több CD-t kell írnunk. A legtöbb általam használt rendszer esetében több CD-re lett volna szükség, ha biztonsági mentést akartam volna készíteni. A ***multiCD***-t pontosan erre a célra hozták létre. Miközben az egyik CD írása folyamatban van, máris megkezdődhet a következő előkészítése, így az írás szinte megszakítás nélkül folyhat. Ez utóbbi lehetőség a lassúbb gépeken érthető okokból nem fog működni. A futtatáshoz Perl szükséges.
➲ http://www.multicd.rmdashrf.org

SimpleCDR

Semmi bajom nincs az `xcdroast` programmal, sem az X-felülettel, azonban TTY-felületről időnként mégiscsak könnyebb és gyorsabb dolgozni, ezért gyakran igénybe veszem. A ***SimpleCDR*** gyakorlatilag bármely környezetben működik, de terminálról különösen jól használható. A CD-lemezeket nem fogja gyorsabban megírni, de X-et sem kell indítani a használatához. A futtatáshoz szükséges: libncurses, libstdc++, libm, glibc.
➲ http://ogre.rocky-road.net/cdr.shtml

wavemon

Ha vezeték nélküli hálózatot használsz, valószínűleg már rájöttél, hogy ezekhez a kártyákhoz sajnálatosan kevés eszköz érhető el. Szerencsére, ha WaveLAN vagy hasonló hálózatot használsz, legalább egy eszköz a rendelkezésedre áll. A `wavemon` ugyan nem csodaszer, mégis a kezdete valaminek. Milliófféle kártyával működik, például a Lucent Orinocóval és sok egyébbel is. Most már csak az hiányzik, hogy ezt a kártyát valamilyen eszköz segítségével Linux alatt hozzáférési pontként is használni lehessen. A futtatáshoz szükséges: libm, libncurses, glibc.
➲ http://www.jm-music.de/projects.html

IC-RADIUS

Ha bármilyen okból kifolyólag vagy valamilyen célra ***RADIUS***-kiszolgálót kell felállítanod, vess legalább egy pillantást az ***IC-RADIUS*** kiegészítőre is. A telepítés után az összes rendszergazdai feladat a böngészőn keresztül végezhető el (nyilván biztonságos kiszolgálót fogsz használni a távoli felügyelethez). Csoportokat és felhasználókat hozhatsz létre, megtekintheted a kimutatásokat és sok mást is. A telefonos hálózati bejelentkezés nagyon gyors. A futtatáshoz szükséges: MySQL, libmysqlclient, libm, libcrypt, libnsl, libz, glibc.
➲ http://www.radius.innercite.com

FreeMED

A FreeMED teljes csomag orvosi gyakorlatokhoz való vagy olyan kórházi környezetbe, mely az ISO9000 előírásoknak próbál megfelelni. Még nem az igazi, számos fontos lehetőség hiányzik belőle, mindenesetre jól néz ki, megfelelően működik, és nem utolsósorban könnyű telepíteni. Ha biztonságra vágysz, csak dobd át a biztonságos webkiszolgáló saját könyvtárába és kész. A futtatáshoz szükséges: webkiszolgáló PHP4- és MySQL-támogatással, MySQL kiszolgáló, phpwebtools, böngésző.
➲ http://www.freemed.org

Sysmon

Ha nagyszámú rendszert kell felügyelned, a ***Sysmon*** segítségével könnyedén kézben tarthatod a munkát. A programot hálózatfelügyeleti központokhoz tervezték, ezért a `sysmon` démont egy felügyeleti állomásnak kinevezett gépre kell feltelepítened, lekérdezését pedig a `sysmon`-ügyfél segítségével a hálózat bármely pontjáról elvégezheted. A legnehezebb rész a beállítás, ehhez ugyanis nincs megfelelő leírás, emiatt a kezdők számára elrettentő feladat lehet. Futtatásához szükséges: libresolv, libnsl, libncurses, glibc.
➲ http://www.sysmon.org

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

Adatmodellezés Alzabóval

Reuven ebben a hónapban egy kis kitérőt tesz, hogy megmutassa, miképpen hidalhatjuk át az objektumrelációs adatbázis-szakadékot.

Az utóbbi néhány hónapban több szemszögből is megnéztük a kiszolgálóoldali Java-programozás művészetét. A servletektől kezdve a JSP-ken keresztül az Enhydra alkalmazáskiszolgálóig számos módszert ismertünk meg a nyílt forrású Javát használó dinamikus és adatbázis-vezérelt honlapok készítése terén.

Eredetileg e hónapban is ezt az irányvonalat szerettem volna követni, belepillantva az Enhydra érdekes, DODS objektum-relációs modellezőprogramjába. A DODS magas szintű Java-elvonatkoztatási (abstraction) réteget nyújt, amelyen keresztül elérhetők a relációs adatbázis kezelőtáblái.

A DODS-eljárások önműködően a megfelelő SQL-utasítássá alakítódnak, majd az adatbázis-kezelőhöz kerülnek. Az eredmény: Java-objektumokat és -eljárásokat látunk, az adatbázis(kezelő) pedig táblákat és SQL-lekérdezéseket, így mindenki boldog lehet.

A Lutris- (az Enhydra anyacége) dolgozók jó szándéka és segítőkészsége sajnos nem igazán talált partnerre az izraeli kiszállítócégekben és helyi FedEx-társaságban. A CD és a könyv – amely a DODS-ról szóló további leírásokat tartalmazza – e sorok írása közben is valamelyik raktárban pihen, ezért kénytelen vagyok eredeti tervemtől eltérni.

Az e havi cikk elkészítéséhez a DODS körül vizsgálódtam, ugyanis az érdeklődésemet felkeltette az objektumvonatkozású leképezés témaköre. Az egyik legérdekesebb és legkönnyebben használható eszköz, amit e témakörben láttam, az Alzabo, amely tulajdonképpen egy Perl-modulgyűjtemény. Ez teszi lehetővé a kiszolgálóoldali Perl-programozóknak, hogy relációs adatbázisaikat objektumok formájában képezzék le (a projekt a Gene Wolfe című sci-fi egyik teremtménye után kapta a nevét). Teljesen lenyűgözött, amit láttam, és biztos vagyok benne, hogy számos Perl-programozó ugyanilyen boldog, ha ilyen hatékony eszközre bukkan.

A nehézség

A programozók az objektumok használatával számos előnyre tehetnek szert: az újrafelhasználhatóságtól kezdve az öröklésen át az egymásba ágyazhatóságig. Csakhogy míg a programozók tömegesen állnak át objektumközpontú programozásra, az objektumalapú adatbázisok különféle okokból kifolyólag sokkal kevésbé népszerűek. Ehelyett az utóbbi pár évben a relációs adatbázisok váltak egyre népszerűbbé, hiszen időközben hihetetlen mennyiségű adatot halmoztak fel bennük. A feladat ezután természetesen az, miképpen modellezzük táblában tárolt adatainkat objektumként. Az egyik lehetőség, hogy minden táblát külön osztályként modellezünk, minden táblaoszlopot egy-egy példányváltozónak tekintünk, és minden sort az osztály példányaiként fogunk fel. Csakhogy bárki, aki már próbálta, tudhatja, hogy mennyivel könnyebb ezt mondani, mint megvalósítani, különösképpen webalkalmazások készítése során, például hogyan tudunk egyesíteni (`join`) két táblát? Mi történik, ha két program ugyanazt a sort fogja módosítani a memóriában, és csak később írják vissza (`commit`) ezeket a változásokat az adatbázisba? Hogyan tudjuk biztosítani, hogy az adatbázis tükrözze az osztálymeghatározások változtatásait is, és fordítva?

Másik lehetőségként az egész táblát egyetlen objektumpéldányként olvassuk be, módosítjuk az objektumot, majd visszaírjuk, amikor valamilyen eljárást meghívunk. Ez kis táblák esetében nagyon jól működik, de mi történik, ha a tábla néhány megabájt (vagy gigabájt, esetleg terabájt) méretű? Főnökünk esetleg vesz még egy kis memóriát a webkiszolgálóba, de nem azért, hogy teljes táblák memóriába olvasására pocsékoljuk! Emellett ha a táblákat az objektumokban modellezzük, megfelelő zárolási mechanizmust is be kell építenünk a `commit` és `rollback` szolgáltatásokkal, és ennek végrehajtásához a felkészült programozó még kevés.

Amíg kis alkalmazásokkal dolgozunk, könnyedén megbirkózhatunk az ilyen nehézségekkel, de amint az alkalmazás és az adatbázis növekszik, meg szeretnénk bizonyosodni róla, hogy az események a várakozásoknak megfelelően zajlanak. Ez különösképpen igaz egy olyan objektumrelációs leképezőrendszer esetében, mint az Alzabo. Tavaly az egyik alkalmazottammal egy egyszerű objektumrelációs leképező középréteget (middleware layer) készítettünk, és nagyon elégedettek voltunk azzal, amit csináltunk – egészen addig, amíg rá nem jöttünk, hogy közel sem minden lehetőséget vettünk számításba, így végül kivételek és alapértékek tömegével lettünk gazdagabbak.

Minden Perl-programozó szerencséjére, egyikünk, ***Dave Rolsky*** vette a fáradságot, leült és az összes ilyen gondot megoldotta, illetve számos egyéb feladattal is megbirkózott. Az Alzabo objektumközpontú középréteget képez, amely feleslegessé teszi, hogy közvetlenül az adatbázissal kelljen bajlódnunk. Az Alzabo azonban többet nyújt pusztán magas szintű adatbázis-kezelő felületnél. Lehetőséget ad az adatbázisséma-meghatározások programozott megváltoztatására, ideértve a böngészőalapú táblakészítést, valamint az olyan karbantartó eszközöket, amelyek az SQL-parancsokat önműködően állítják elő a számunkra. Ezenkívül az Alzabo képes egy már létező adatbázis visszafejtésére is, lehetővé téve, hogy ezeket az adatbázisainkat is ugyanúgy kezelhessük Alzabóval, mint az újonnan készítetteket.

Az Alzabo telepítése

Mint a legtöbb Perl-modul, az Alzabo is letölthető a CPAN-ról. A telepítése azonban valamivel bonyolultabb feladat, mint más moduloké, mert az Alzabó számos egyéb modultól függ. Nemcsak a DBI (az adatbázis-eléréshez), a ***DBD::mysql*** vagy a ***DBD::Pg*** (PostgreSQL-eléréshez) használatát követeli meg, hanem a böngészőalapú sémakészítéshez a ***HTML::Mason***-t is használja, ennek pedig szüksége van a mod_perl-re. Ha a

rendszer mindezeket már tartalmazza, az Alzabo telepítése sem lesz túl bonyolult.
Minden nehézség nélkül sikerült telepítenem az Alzabót azáltal, hogy a megfelelő CPAN-modulokat letöltöttem és önműködően telepítettem, majd magával az Alzabóval is így jártam el.
A program beállítása során csaknem minden esetben az alapértelmezett értéket fogadtam el, kivéve a ***.mhtml*** utótag-beállítást, amelyet az Alzabo alapértelmezés szerint a Mason-elemekhez feltételez. A Mason-elemeknek általában az egyszerű ***.html*** utótagot adom, ugyanis Apache rendszerem nem tud mit kezdeni a ***.mhtml*** kiterjesztéssel és ***Content-type text/plain*** tartalomtípus szerint küldi el, ezért a böngésző ablakában a Mason-alkalmazás forráskódja jelenik meg. Rendszeremen a telepített Mason-alkalmazások utótagjának ***.html***-re történő módosítása megoldotta ezt a gondot, de az is igaz, hogy ugyanilyen könnyűszerrel változtathattam volna meg a Mason- vagy az Apache beállításokat is.
Az Alzabo minden sémát a saját könyvtárában követ nyomon. Alapesetben ez a könyvtár a ***/usr/local/alzabo***. Ebben a könyvtárban található a sémakönyvtár – minden egyes Alzabo által modellezett adatbázissémához egy-egy alkönyvtárral. Például az ***appointments*** sémák a ***/usr/local/alzabo/schemas/appointments*** könyvtárba kerülnének.
Mindössze két buktatóval találkoztam az Alzabo telepítése során, amelyeket meg kellett szüntetnem. Elsőként meg kellett változtatnom a ***/usr/local/alzabo*** könyvtár jogosultságát, hogy a webfelhasználóm képes legyen olvasni és írni. Másodikként pedig a PostgreSQL-indítófájlt kellett módosítanom, hogy a `-i` kapcsolót tartalmazza, és az ügyfél TCP/IP-hálózaton keresztül is csatlakozhasson. Alapértelmezés szerint a legtöbb PostgreSQL-telepítés (ideértve az RPM-változatokat is) nem állítja be a `-i` kapcsolót, ami azt jelenti, hogy még a legegyszerűbb beállítások sem fognak működni a ***pg_hba.conf***-ban (a PostgreSQL-elérést szabályozó fájlja). Bár egyébként Unix-socketek segítségével hálózat nélkül is hozzá tudunk kapcsolódni a PostgreSQL-hez, az Alzabo mindig megadja a gazdagép nevét, ami azt eredményezi, hogy még a helyi hálózaton elhelyezkedő gép használata esetén is hálózati kapcsolat épül fel.
A webalapú sémakészítő telepítéséhez Apache-kiszolgálónk legalább egy könyvtárát a ***HTML::Mason***-nak kell irányítania. Az Alzabo telepítőfájl egy új ***new/alzabo*** alkönyvtárat fog készíteni azokkal a Mason-elemekkel együtt, amelyek az általunk készítendő sémameghatározásokat fogják felépíteni és módosítani. Munkaállomásomon például az összes ilyen Mason-elem a ***/usr/local/apache/mason*** könyvtárban található, amely a ***/mason*** kezdetű URL-ekhez van rendelve. Alzabo-telepítésem tehát a ***/usr/local/apache/mason/alzabo*** könyvtárban található, és a ***/mason/alzabo*** URL-en keresztül érhető el. Ha eddig még nem tettük volna meg, az Apache-nak a `DirectoryIndex` segítségével megmondhatjuk, hogy az ***index.mhtml*** elfogadható a könyvtárhoz rendelt kezdőlapként.

Sémák szerkesztése

Az Alzabo telepítése után a böngészőalapú tervezőeszközzel készítsünk egy egyszerű adatbázissémát. Kétségtelenül nem olyan tiptop, mint az üzleti vagy az ügyféloldali eszközök, viszont jól megfelel a célnak.
Kezdjük egy új séma készítésével (PostgreSQL vagy MySQL értelemben adatbázissal), amelynek természetesen nevet kell adnunk. A sémának a PostgreSQL és az MySQL szerint is érvényes adatbázisnévnek kell lennie. A PostgreSQL-t választottam a beépített hivatkozásépség (referential integrity), az idegen kulcsok (foreign keys), a nézetek (views) és a kioldók (triggers), illetve az általánosabb SQL-nyelvjárás azon képessége végett, mert így számos nyelven nyílik lehetőségünk belső eljárások írására.
Készítsünk egy egyszerű telefonkönyvet és találkozónaptárat az Alzabo segítségével! Nyomon követhetjük az általunk ismert embereket, a címüket és telefonszámukat, illetve a velük megbeszélt találkozók időpontjait. Ezen adatbázis segítségével megtudhatjuk, milyen emberekkel kell találkoznunk egy adott napon, illetve lekérdezhetjük egy adott emberrel megbeszélt összes találkozásunkat.

➲ sourcefog.net/projects/alzabo

A séma elkészítéséhez a böngészőt az ***alzabo/schema*** URL-re kell irányítanunk az imént említett Mason-könyvtár alatt (tehát a gépemen a böngészőt a ➲ http://localhost/mason/alzabo/schema címre kellett irányítani). Ez felhozza a sémakészítő és szerkesztő lapot, amely lehetővé teszi, hogy egy már létező sémát átszerkesszünk, újat készítsünk vagy a már létezőt visszafejtsük. Az utolsó lehetőség a legérdekesebb, hiszen megengedi, hogy az Alzabo segítségével az öröklött adatbázishoz is hozzányúljunk. Mi most új sémát fogunk készíteni. Be kell gépelünk a nevet (jobb ötlet híján az addressbook – címjegyzék – nevet választottam), illetve jelölnünk kell, hogy a PostgreSQL-t fogjuk háttéradatbázisként használni.
A ***Submit*** gombra kattintás után több lehetőség közül választhatunk: hozzáadhatunk egy új táblát a sémához, törölhetjük az egész sémát vagy megnézhetjük az SQL-sort, amelyet az Alzabo önműködően hozott létre. Egyelőre természetesen még semmilyen megjeleníthető SQL-kód nincsen, idővel azonban láthatjuk majd, ahogy az SQL-kód egyre növekszik.
Az Alzabo ugyan semmilyen SQL-sort nem készített még, de ez nem azt jelenti, hogy a háttérben nem is végzett semmilyen előkészítő munkát. Valójában az Alzabo a ***/usr/local/alzabo/schemas***-ban önműködően elkészítette a címtárkönyvtárat, amely három fájlt tartalmaz: az ***addressbook.create.alz***-t, az ***addressbook.runtime.alz***-t (mindkettőt futtatható formátumban) és az ***addressbook.rdbms***-t, amely egyetlen „PostgreSQL" szót tartalmaz. Ezzel a módszerrel az Alzabo nyomon követheti az adatbázis-kiszolgálót, ahol a séma tárolódik.
A címjegyzéksémába a belépést követően az ***Add a table*** szövegmezőbe az „Emberek" szót begépelve, majd a ***Submit***

gombra kattintva az „Emberek” táblát hozzáadtam az adatbázishoz. (A PostgreSQL ugyan figyelmen kívül hagyja a kis- és nagybetű különbségeit a tábla- és mezőnevekben, de én a Táblanevek kezdőbetűit illetőleg ***Joe Celko*** szabályait szeretem alkalmazni, illetve a mezőneveket csupa kisbetűvel írom, és csupa nagybetűvel az SQL FENNTARTOTT SZAVAIT.)
Az Emberek táblában létrehoztam néhány különböző típusú mezőt. Az Alzabo menüben ajánlja fel a leggyakrabban használt adattípusokat, de akár saját készítésűt is begépelhetünk, ha kívánjuk; ez különösképpen PostgreSQL alatt lehet hasznos, mivel lehetővé teszi a számunkra, hogy saját adattípusokat hozzunk létre.
Az ilyen táblák esetében általában mesterséges elsődleges kulcsokkal szeretek dolgozni, minden egyes sorhoz egyedi értéket rendelve. PostgreSQL alatt ezt a `SERIAL` adattípussal adhatjuk meg. Rögtön észrevehetjük, hogy az Alzabo választéklistájában nincs ilyen típusú mező. Esetleg kísértést érezhetünk, hogy a mezőt `INTEGER` típusúnak jelöljük meg, és bekapcsoljuk hozzá a „sequenced” jelölőnégyzetet a mezőszerkesztő alsó részén. Ilyenkor ugyanis egy `INTEGER`-típusú oszlop, és egy ettől teljesen független PostgreSQL-szekvencia jön létre. Ezért a mesterséges elsődleges kulcshoz a `<select>` mezőtípuslista alatti szövegmezőbe a `SERIAL` kulcsszót inkább kézzel gépeljük be.
Egy további jelölőnégyzet segítségével az oszlopot elsődleges kulcsként adhatjuk meg, amely ezt követően `pk` jelöléssel szerepel a mezőlistában. Végül a harmadik jelölőnégyzet mutatja meg, hogy a mező tartalmazhat-e `NULL` értéket. Figyelmeztetek minden kezdő adatbázis-tervezőt, hogy a `NULL`-ok, azaz az üres mezők túlbonyolítják az életünket, és amikor csak lehetséges, célszerű elkerülnünk a használatukat.
Az idegen kulcsok (`REFERENCES` vagy `CHECK` záradék) megadásához a megfelelő sort a HTML-űrlap alsó részén adjuk az ***attributes*** szövegmezőhöz. Ne feledjük, a Perlben ezen a ponton még csak modellezzük a sémát, ami azt jelenti, hogy bármilyen záradékot később is hozzáadhatunk vagy eltávolíthatunk, anélkül, hogy az adatbázisnak `ALTER TABLE` lekérdezéseket kellene küldenünk. Az Alzabo szerkesztőjével egy vagy több oszlopra indexeket is készíthetünk.
Az Alzabo tábla- és oszlopszerkesztőjével több táblát is gyárthatunk, amelyek között a többszintű menü- és listarendszer segítségével mozoghatunk. Az Alzabo minden egyes oszlop mellett megjelenít egy „<” és „>” jelet, amelyekkel egymáshoz képest az adott meghatározásnak megfelelően mozdíthatjuk el őket.
Ha böngészőalapú sémaszerkesztővel dolgozunk, javaslom, vessünk néha egy pillantást az Alzabo által készített SQL-kódra is. Nemcsak azért, hogy meggyőződjünk róla, hogy az Alzabo helyesen építi-e fel (emlékezzünk a `SERIAL` oszlopra), hanem mert így jobb képet kapunk az éppen készülő séma alacsony szintű szerkezetéről.
Ha elkészültünk a sémával, használjuk az ***execute SQL*** gombot az ***SQL preview*** űrlapon, és az SQL-lekérdezés(eke)t küldjük el az adatbázis-kiszolgálónak. Ha az adatbázis-kiszolgáló hibaüzenetet küld, az Alzabo hosszú és részletes hibaüzenetet fog visszaadni, amely leírja, mi is történt tulajdonképpen.
Néhány esetben a tábla- vagy mezőmeghatározásokat kell módosítanunk, ám az is előfordulhat, hogy a kiszolgáló megfelelő jogosultságbeállításaival akadt gond. Arról is győződjünk meg, hogy létrehoztunk-e olyan PostgreSQL-felhasználót (a parancssoros `createuser` programmal), akinek a neve megegyezik azzal a névvel, ami alatt az Apache fut – hacsak nem akarunk a HTML-űrlapon közvetlen módon más felhasználót megadni.

Sémahasználat programból

Miután az SQL-t futtattuk a sémaszerkesztőben, két lehetőségünk kínálkozik az adatok elérésére. Természetesen elérhetjük őket közvetlenül a DBI-n keresztül (vagy hasonló felületen más nyelvek esetében), ha létrehozzuk és végrehajtjuk az SQL-lekérdezéseket.
Például a címjegyzéksémát a következő táblának megfelelően építettem fel:

```
CREATE TABLE Emberek (
    szemely_id    SERIAL    NOT NULL,
    first_name     TEXT      NOT NULL,
    last_name    TEXT      NOT NULL,
    birthday          DATE      NOT NULL,
    PRIMARY KEY(person_id)
);
```

Tegyük még érdekesebbé, ezért gépeljünk pár adatot is a táblába:

```
INSERT INTO Emberek (first_name, last_name,
                     ➥birthday)
VALUES (·Reuven·, ·Lerner·, ·1970-Jul-14·);

INSERT INTO People (first_name, last_name,
                    ➥birthday)
VALUES (·Atara Margalit·, ·Lerner-Friedman·,
       ➥·2000-Dec-16·);
```

Az *1. lista* egy egyszerű Perl-programot tartalmaz, amely DBI-t használ a nevek (és születésnapok) lekéréséhez, tehát azon emberek neveinek (és születésnapjainak) a lekérésére a címtárból, akik megegyeznek a parancssorba gépelt SQL-mintával. (Az SQL-minták sokkal egyszerűbbek, mint a Unix szabványos kifejezései és mindössze két karaktert használnak: a % minden nulla vagy hosszabb karaktersorozattal, a _ pedig pontosan egyetlen karakterrel egyezik.)
A felhasználó által beírt bemenetet beolvassuk a parancssorból, elé és utána % jeleket helyezünk, így a karaktersorozat találatot ér el, függetlenül attól, hogy hol található meg a `first_name` (keresztnév) vagy a `last_name` (vezetéknév) oszlopban.
Ezután csatlakozunk az adatbázishoz, bekapcsoljuk az `AutoCommit` szolgáltatást (miként azt a DBI-leírás ismerteti), és felélesztjük a `RaiseError` és `PrintError` hibaelemző eszközöket.
Végül a `$sql` változóban előállítjuk az SQL-lekérdezést, nem feledve, hogy a közvetlen változóbehelyettesítés helyett inkább helyettesítő karaktert (?) használunk . Ez nemcsak annak a veszélyét csökkenti, hogy valaki belerondít az SQL-ünkbe, de néhány adatbáziskezelő-vezérlő a következő lekérdezésben ki is használja a helyettesítő karaktereket, ez pedig jelentős sebességnövekedést eredményezhet.

Ugyanez Alzabóval

Most írjuk újra a fenti programot a közvetlen DBI-használat helyett az Alzabo segítségével. Nem fogunk közvetlenül SQL-t írni, és az adatbázishoz sem csatlakozunk. Ehelyett új sémaobjektumot készítünk, elnevezve az Alzabo interaktív eszközével készített sémát. Ez az objektum számos eljárással rendelkezik, a segítségükkel csomó olyan feladatot is elvégezhetünk, amelyekhez egyébként a DBI-t kellene használnunk.
A két változat között nem sok különbséget találunk – miként

1. lista retrieve-today-birthday.pl, amely DBI segítségével keresi vissza a címjegyzéktáblából azoknak az embereknek a neveit, akiknek ma van a születésnapjuk

```
#!/usr/bin/perl

use warnings;     # Perl 5.6.x version of
                  # "-w" option
use strict;
use DBI;

# beÆll tunk nØhÆny alapvÆltoz t
my $dbname = ·addressbook·;
my $username = ·reuven·;
my $password = ··;

# milyen nØvre keres nk?
my $look_for_name = $ARGV[0];
die "Nem  rtÆl be keresendı nevet!"
    unless $look_for_name;

# hozzÆkapcsol dunk az adatbÆzishoz
my $dbh = DBI->connect
               ("dbi:Pg:dbname=addressbook",
                $username, $password,
                {RaiseError => 1,
                PrintError => 1,
                AutoCommit => 1});

# Ellenırizz k, hogy sikeresen csatlakoztunk-e
# az adatbÆzishoz
die "No connection to the database:
    ➥·$DBI::errstr· "
    unless $dbh;

# %-jel ragasztÆsa el lre Øs hÆtra az SQL
# regexp elıÆll tÆsÆhoz
$look_for_name = "%" . $look_for_name . "%";

# SQL-lekØrdezØs kØsz tØse
my $sql = "SELECT first_name, last_name,
          ➥birthday ";
$sql .= "  FROM Emberek ";
$sql .= "  WHERE (first_name LIKE ?)
           ➥OR (last_name LIKE ?) ";

# felkØsz lØs a lekØrdezØshez
my $sth = $dbh->prepare($sql);

# lekØrdezØs vØgrehajtÆsa
$sth->execute($look_for_name, $look_for_name);

# vØgiglØpked nk az eredmØnysorokon
while (my $row_ref = $sth->fetchrow_arrayref)
{
    my ($first_name, $last_name, $birthday)
      ➥= @{$row_ref};
    print "$first_name $last_name
           ➥(birthday: $birthday)\n";
}

# Ellenırizz k, hogyan j ttek vissza a sorok
if ($sth->rows == 0)
{
    print "Nem volt talÆlat a c mtÆrban.\n";
}

# vØgezt nk az SQL-rØsszel
$sth->finish;

# lezÆrjuk a kapcsolatot Øs az adatbÆzist
$dbh->disconnect;
```

azt a 2. *listában* láthatjuk –, amíg hozzá nem kapcsolódunk az adatforráshoz. A DBI programváltozatban az adatforráshoz a `DBI->connect` paranccsal kapcsolódtunk; az Alzabóban ezzel szemben a sémához, amely viszont feltételezhetően az adatbázishoz kapcsolódik, és hozzárendeljük a `$schema` objektumhoz.
A `$schema` használatával az egyik táblánkhoz tartozó tábla-objektumot szerezhetjük meg:

```
my $emberek = $schema->table("Emberek");
```

Most, hogy az Emberek táblához tartozó objektumunk már létezik, könnyűszerrel hozzáférhetünk a tábla kiválasztott soraihoz. A sorok lekérésére a legegyszerűbb mód a `rows_where` eljárás használata, ezáltal egyetlen `Alzabo::Runtime::RowCursor:` típusú objektumot kapunk vissza.

```
my $row_cursor = $people->rows_where
   ➥(where => [[$people->column(·first_name·),
   ➥·LIKE·, $look_for_name], ·or·,
   ➥[$people->column(·last_name·), ·LIKE·,
   ➥$look_for_name]]);
```

Az Alzabo `WHERE`-szerkezetei általában három elemet tartalmaznak: az oszlopobjektumot, az összehasonlító műveleti jelet és az értéket, vagy egy másik oszlopobjektumot. Végigvizsgálhatjuk például a keresztnévoszlopot, hogy akad-e valahol ***Ferenc*** értékű mező:

```
where => [$table->column(·first_name·),
         ➥·=·, ·Ferenc·]
```

A 2. listában ezt egy kicsit tovább bonyolítottuk: a két vektorhivatkozást az `OR` logikai műveletjellel kapcsoltuk össze:

```
where => [[$people->column(·first_name·),
         ➥·LIKE·, $look_for_name], ·or·,
         ➥[$people->column(·last_name·),
         ➥·LIKE·, $look_for_name]]
```

Az Alzabo elég okos ahhoz, hogy felismerje: `WHERE` meghatározása első és harmadik eleme vektorhivatkozás, így a fenti kódot a megfelelő SQL-utasítássá formálja.
Ha hozzáfértünk a `RowCursor` objektumhoz, a `next_row` eljárással játszi módon lépdelhetünk végig az egyes sorokon:

2. lista retrieve-birthday-alzabo.pl, az 1. lista programjának Alzabo-változata

```
#!/usr/bin/perl

use warnings;
use strict;
use Alzabo::Runtime::Schema;

# beÆll tunk nØhÆny alapvÆltoz t
my $schema_name = ·addressbook·;
my $username = ·reuven·;
my $password = ··;

# milyen nØvre keres nk?
my $look_for_name = $ARGV[0];
die "Nem  rtÆl be keresendı nevet!"
    unless $look_for_name;

# %-jel ragasztÆsa el lre Øs hÆtra az SQL
# regexp elıÆll tÆsÆhoz
$look_for_name = "%" . $look_for_name . "%";

# sØma bet ltØse a lemezrıl
my $schema =
    ➥Alzabo::Runtime::Schema->load_from_file
    ➥( name => $schema_name );

$schema->set_user( $username );
$schema->set_password( $password );
$schema->connect;

# megszerezz k a tÆblaobjektumot, amin
# dolgozni szeretnØnk
my $people = $schema->table("Emberek");

# az  sszes, a lekØrdezØs nkkel egyezı sor
# begyßjtØse
my $row_cursor = $emberek->rows_where
    (where => [[$emberek->column(·first_name·),
                ➥·LIKE·, $look_for_name],
                ➥·or·,
               ➥[$emberek->column(·last_name·),
                  ➥·LIKE·,
                  ➥$look_for_name]]);

my $rows_returned = 0;

# vØgiglØpked nk a sorokon a cursor seg tsØgØvel
while (my $row = $row_cursor->next_row)
{
    my $first_name = $row->select(·first_name·);
    my $last_name = $row->select(·last_name·);
    my $birthday = $row->select(·birthday·);

    print "$first_name $last_name
           ➥(birthday: $birthday)\n";

    $rows_returned++;
}

# jelezz k, ha volt hiba
if ($rows_returned == 0)
{
     print "Nem volt talÆlat a c mtÆrban.\n";
}
```

```
while (my $row = $row_cursor->next_row)
{
   my $first_name = $row->select(·first_name·);
   my $last_name = $row->select(·last_name·);
   my $birthday = $row->select(·birthday·);

   print "$first_name $last_name
         ➥(birthday: $birthday)\n";

   $rows_returned++;
}
```

Gyorstárak és kivételek

Ha az Alzabo mindössze csak néhány SQL-készítő eljárást nyújtana, nem volna valami hatékony eszköz. Az Alzabo azonban a csomag részeként lehetőséget ad a gyorstárazásra és a kivételkezelésre is, ami megkönnyíti az adatbázisokkal való munkát.

Az Alzabo gyorstárazó eszköze a memóriában tartja a táblát ahelyett, hogy minden egyes lekérést rögtön az adatbázis-kiszolgálóhoz továbbítana. Nyilvánvaló, hogy a gyorstár használata az olyan táblák esetében nem megfelelő, amelyek rendszeresen megváltoznak, viszont ritkán változó táblák esetén nyugodtan beindíthatjuk, és élvezhetjük a kellemes sebességnövekedést.

A gyorstárazást az `Alzabo::ObjectCache` modul programba töltésével indíthatjuk be. A `RowCursor` objektum, amelyet a *2. listában* a sorok lekérésére használtunk, a `next_row` eljárás minden egyes újabb ismétlésénél `Row` objektumokat ad vissza. A különféle elérhető gyorstártípusokról és a velük kapcsolatos tudnivalókról az ***Alzabo::Runtime::Row*** és ***Alzabo::ObjectCache*** leírásában további adatok találhatók.

Az Alzabo a Perl beépített kivételkezelő rendszerét is használja, azaz ha valami nem megfelelően működik, meghívja a `die` parancsot. Így Alzabo- programjainkat (vagy akár a bennük található egyedi hívásokat) `eval`-blokkokba csomagolhatjuk be:

```
# pr bÆljuk meg futtatni ezt a k dot
eval {
    my $row_cursor = $people->rows_where(
    ➥where => [[$people->column(·first_name·),
    ➥·LIKE·, $look_for_name], ·or·,
    ➥[$people->column(·last_name·), ·LIKE·,
    ➥$look_for_name]]);
};
```

A `$@` Perl-változó vizsgálatával megtudhatjuk, ha valami nem megfelelően működött, mivel ez a változó akkor kerül beállításra, ha az előző `eval` során valamilyen hiba lépett fel. Az Alzabo használja az `Exception::Class` objektumot is

(amely a CPAN-on érhető el), így még kifinomultabb Perl-kivételkezelést képes nyújtani. A `$@` változóba nem a hibát leíró karaktersorozat kerül, hanem a megfelelő kivételosztály példánya. Így aztán a `$@`-t kipróbálhatjuk az `UNIVERSAL::isa` feltétellel, hogy megállapítsuk, miféle objektum is ez, illetve milyen hiba történt a kódunkban. Az Mason által irányított Apache tartalomkönyvtárban található ***Alzabo*** könyvtárba telepített ***common/exception*** Mason-elemek részletesen bemutatják, hogyan is kell ezt megtenni.

Tudnivalók

Nyilvánvalóan hátránya is akad az Alzabo használatának, akárcsak bármely olyan programnak, amely az objektumrelációs szakadékot próbálja meg áthidalni. Először is az SQL meglehetősen jó általános módszer a relációs adatbázisokkal végzett munkához.

Az Alzabo használata azt is jelenti, hogy a szabványtól egy másik megoldás felé távolodunk el, amely semmi mással nem egyeztethető össze. Természetesen nem vagyok ellensége az ilyen megoldásoknak, és számos jelentős előnye létezik az Alzabo használatának, azonban óvatos vagyok, amikor egy régi, bevált megoldás új, szokatlan módszerrel történő helyettesítéséről esik szó.

Bár a tábláimat egyébként kézzel összerakott SQL-utasításokkal szoktam megalkotni, ez a módszer nem alkalmazható tíz vagy húsz tábla felett anélkül, hogy az Emacs átmeneti tárban vad görgetésre ne késztetne. Az Alzabo webalapú sémaszerkesztője megkönnyíti a nagy számú táblák nyomon követését, a köztük lévő kapcsolatok készítését, illetve módosításukat. Előfordult már, hogy félórát töltöttem azzal, hogy megpróbáltam visszaemlékezni, miben különböznek az Oracle formai követelményei a PostgreSQL-éitől ilyenkor nagyon örültem volna egy Alzabo-szerű eszköznek.

Amint korábban láttuk, az Alzabóval könnyen előállíthatók egyenlőségen alapuló összetett lekérdezések, még akkor is, ha `OR` és `AND` műveleteket is tartalmaznak.

Az `Alzabo::Runtime::Table` objektum egy eljárást tartalmaz, amely tetszőleges SQL-függvény futtatására szolgál, viszont az Alzabo `WHERE` részének elkészítését meglehetősen nehéznek, néha egyenesen lehetetlennek találtam. Ennek segítségével többszörös függvényhívásokon alapuló SQL-lekérdezést lehetett volna összeállítani. Bevallom, eléggé újonc vagyok még az Alzabo világában, és mindössze egy vagy két órát próbálkoztam, de ha egy lekérdezést húsz másodpercig tart SQL-ben megírni, akkor Alzabo alatt sem szabadna sokkal tovább készülnie.

Az egyik legnehezebb dolog az objektumok relációs adatbázisának leképezésekor a `join`-ok kezelése. A `join`-ok igen fontosak a táblák esetében, de jelentésük sokkal kevésbé nyilvánvaló, amikor objektumokkal dolgozunk. Az Alzabo rendelkezik ugyan néhány beépített `join`-támogatással, de ezt a részt meglehetősen újként és kísérleti jellegűként jelölték meg.

Végül itt van még a sebesség kérdése, amely minden középréteg esetében felmerül. Az 1. és 2. lista közötti különbség erősen érzékelhető volt, amikor parancssorból hajtottam végre őket, ami jórészt annak köszönhető, hogy az Alzabo igen nagy számú Perl-osztályt olvas be. A ***mod_perl*** környezetben (ahol az Alzabót tervezték), a sebességkülönbség már jóval kisebb, mivel a legtöbb idő a modulok lemezről való betöltésével telik el. Mivel a ***mod_perl*** futtatás előtt csak egyszer fordítja le a programokat, a sebességkülönbség az Alzabo és a nyers DBI-hívások között valószínűleg már nem olyan nagy.

Összegzés

Az Alzabo viszonylag egyszerű módszert nyújt a relációs adatbázistáblák objektumokba burkolására. Számos jó hírt közölhetek ennek kapcsán: az adatmodellező eszköz meglehetősen kifinomult, ügyes képességekkel rendelkezik, az eljárások eléggé értelmesek, a leírás vaskos és jól megfogalmazott. Ahogy a Nyílt Forráskód Közössége régen hangoztatja: ugyanannak a feladatnak a megoldására egy régi, jól bevált, hadviselt és nyílt eszköz majdnem mindig jobb, mint egy új üzleti csomag bevezetése.

Az relációs adatbázistáblák objektumokba csomagolása azonban veszélyekkel és hibákkal terhes, ami alól az Alzabo sem kivétel: a `join`-ok kezelése még mindig hagy némi kívánnivalót maga után, és az sem látható tisztán, hogy néhány lekérdezést miképpen lehet összeállítani. Ez nem az Alzabo hibája: kikerülhetetlen kérdés, ha két olyan módszerrel dolgozunk, amely a világot teljesen eltérő módon látja.

Teljesen nyilvánvaló, hogy az Alzabót a jövőben valamilyen kiszolgálóoldali munkámban fel fogom használni, különösen ha kifinomult gyorstárazásra és kivételkezelésre lesz szükségem, amit egyébként nekem kellene megírnom.

A következő hónapban visszatérünk eredeti, kiszolgálóoldali Java-körutunkhoz, és az Enhydra DODS csomagját az Alzabóval és rokonaival hasonlítjuk össze.

Reuven M. Lerner (reuven@lerner.co.il) kisebb webes és internetes módszerekkel foglalkozó tanácsadó cég tulajdonosa és vezetője. A cikk megjelenésének időpontjában valószínűleg már végleg elkészült Core Perl című könyvével, melyet idén jelentet meg a Prentice-Hall. Az ATF honlapon érhető el (➲ http://www.lerner.co.il/atf/).

További érdekességek

Az Alzabo-modulok elérhetők a CPAN-on keresztül („Alzabo" néven) vagy a projekt honlapján, a SourceForge-on (➲ http://sourceforge.net/projects/alzabo). Ugyanezen a honlapon bőséges leírás is elérhető modulonkénti bontásban, hagyományos perldoc formátumban. Majdnem minden Alzabóval kapcsolatos fejlesztés *Dave Rolsky* nevéhez fűződik, aki igen tevékenyen vesz részt a *HTML::Mason* és a *mod_perl* világában, akárcsak ebben a projektben is.

Az Alzabo nem az első és nem is az egyetlen Perl-rendszer, amely táblákat objektumoknak feletet meg. Egy másik megoldás a Tangram, amit JLL készített és a ➲ http://www.tangram-persistence.org címen érhető el. A Tangram néhány ügyes lehetőséget tartalmaz, az adatbázis-eléréshez pedig kizárólag DBI-t használ, így a Tangram bármely DBD-hez használható, nem csak azokhoz, amelyekhez egyedi alkalmazásvezérővel bír, mint az Alzabo esetében. A Tangram ugyanakkor nem tartalmaz olyan adatbázis-visszafejtő képességet, mint az Alzabo, amelynek segítségével korábbról örökölt adatbázisokkal is dolgozni tudunk.