Télapó, gyere már...

Mindenkit utolért már a decemberi láz. Mindenki az ünnepekkel foglalkozik, gyorsan befejezi hosszú távú terveit és felkészül az ünnepekre. Valahogy így vagyunk mi is: lapzárta után kiakasztjuk a „telelünk" táblát, a következő megjelenésünk ugyanis egy összevont január–februári szám keretében történik, amelyet január legelején lehet majd kézbe venni. Természetesen a január–február is egy számnak minősül, de megnövelt oldalszámmal és 3 CD-melléklettel bővítve – szerencsére azonos áron.

Térjünk azonban vissza a karácsonyhoz! Mi is igyekeztünk megtömni a Mikulás puttonyát mindenféle jóval: most is foglalkozunk majd a bevezető témákkal, ***Szaló István*** például a kezdők számára indított egy sorozatot, amelyben nem kisebb feladatra vállalkozott, minthogy az Emacs-környezetet mutassa be. ***Garzó András*** is kedveskedik az első szárnycsapásokkal próbálkozóknak: a lelkes vállalkozókat egy háromoldalas cikkel igyekszik felkészíteni a rendszermag újrafordítására. Szeretném elmondani, hogy a rendszermag – ahogyan neve is mutatja – a rendszer lelke, emellett rendkívül bonyolult valami, tehát készüljünk fel, hogy elsőre nem mindig járunk teljes sikerrel. Bár olyan ez, mint a tűzkeresztség... ha valaki komolyan szeretne linuxozni, előbb-utóbb át kell esnie rajta! És célszerű némi időt hagyni rá – mondjuk két-három napot.

A haladóknak is igyekszünk kedvezni. Manapság kedvenc témáink egyike a vírusvédelem: nem véletlen, hogy újra előkerült, hiszen ismét rendkívül sok vírus ütötte fel a fejét. Bár hallottam olyan véleményt is, hogy miért szenvedünk? Ha találkozunk egy vírusos géppel, adjunk a felhasználónak egy linuxos telepítőlemezt, és lelkesítsük az átállásra. Ezt azonban a legtöbb cégnél sajnos nem lehet ilyen könnyen megtenni.

A vállalat ügymenetét veszélyeztetett operációs rendszereken végzik, csupán egy-két hozzáértő gépén fut GNU/Linux, és természetesen a kiszolgálókon. Hogy ilyen helyeken miként védhetjük meg mégis a belső hálózatot, erről szól két írásunk is: ***Borkuti Péter*** a 24. oldalon a proxy felvértezéséről ír, ***Dave Jones*** a 30. oldalon pedig egy második megoldást javasol a levezés szűrésére. Ám ez csak szemezgetés volt az e havi cikkekből.

Miről szóljunk?

Ismét szeretnénk megkérni olvasóinkat, hogy mondják meg, milyen témájú és szintű cikkeket látnának még szívesen a lap hasábjain! Az újsághoz mellékeltünk egy kérdőívet, kérek mindenkit, szánjon rá néhány percet, hogy elmondja igényeit, véleményét és ötleteit. A kérdőív kitölthető a ➲ http://www.linuxvilag.hu/kerdoiv címen is. Külön kíváncsi vagyok rá, hogy vajon a Linux-változat tárgykörben ki is nyer. Mióta a Linuxvilág él, folyamatosan kapom a véleményeket, mindegyik oldalról. Három típuslevelet már könnyedén megkülönböztetek:

a) „Kedves Linuxvilág! Örömmel lapozgatom újságjukat, de szeretném megkérdezni, miért nem foglalkoznak többet az amerikai felmérések szerint is legjobb Linux-változattal, a RedHat Linuxszal? Nem véletlen, hogy olyan sokan használják, könnyű felügyelni és egyszerűen csodás!"

b) „Kedves Linuxvilág! A lap szép, jó, de nagyon hiányolom belőle, hogy nem foglalkozik eleget a SuSE Linux (igény szerint a Mandrake szó is behelyettesíthető) képességeivel, lehetőségeivel. Rendkívül könnyen használhatónak találom, szerintem mindenkinek ezt kellene ajánlaniuk!"

c) „Kedves Linuxvilág! A lap érdekes, tartalmas, de! Elképesztőnek tartom, hogy egy szabad és nyílt világban olyan rendszereket is támogattok és ajánlotok a felhasználóknak, amelyek mögött cégek állnak, és mások közösségi munkájából akarnak nyereséghez jutni! Szerintem csak a Debian GNU/Linuxot szabadna bárhol is használni, szinte mindegyik komoly kiszolgálón ez fut!"

Úgy látszik, a háború már a vérünkké vált. Jómagam is rendelkezem már tapasztalatokkal, viszont nem érzem úgy, hogy lándzsát kellene törnöm bármelyik mellett is. Mindegyiknek van előnye is, hátránya is. A tisztánlátás kedvéért: én most egy Debianról kapott X-felületen dolgozom, amit egy Mandrake GNU/Linuxot futtató gépen látok. Mostanában ezzel ütöm el az időmet: teljes értékű X-munkagépeket alakítgatok ki csacsi öreg P1-esekből. A legnagyobb bajom, hogy csak rendkívül drágán lehet manapság hozzájutni rendes PCI csatolós monitorkártyához. Sőt, mivel a merevlemezeink egyre-másra adják be a kulcsot, hamarosan utána kell néznem, hogyan is lehet lemezmentes munkaállomásokat kialakítani.

Búcsúzóul

A karácsonyi szünet elejének, bár a szeretet jegyében illenék eltelnie, amennyire saját tapasztalatomból le tudom vonni, általában az ajándékok után rohangálással, a felhalmozott „majd később" munkák elvégzésével és a rég nem látott ismerősök felkeresésével telik el. Erre a rövid időszakra még egy kis kitartást kívánok mindenkinek, és remélem, a Jézuska mindenkinek számos kellemes meglepetést hoz! Boldog karácsonyt és kellemesen másnapos boldog új esztendőt 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 Woody 3. CD

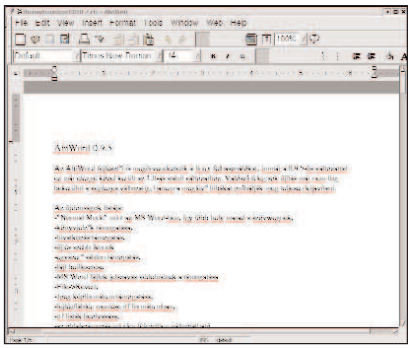
A múlt hónapban elkezdett Debian Woody változat első két korongja után most a harmadik CD kerül kiadásra. Ez lesz az utolsó a Woody terjesztés 5–6. korongjából, aminek egyszerű oka az, hogy mire az összes korongot kiadnánk, addigra már sokkalta újabbak jelennének meg, tehát nem lenne friss. Sokan kérdezik, hogyha bizonyos programok fordításához, futtatásához hiányzik egy szükséges fájl, és a rendszer nem találja, akkor honnan tudható meg, hogy melyik csomagot kell feltelepíteni (amelyik tartalmazza a hiányolt fájlt). Nos, ehhez is megvan a segítség: a 24-es CD Woody könyvtárában található ***Contents-i386.gz*** fájlban rákeresheünk a fájlra. Ha van ilyen fájl, akkor mellette rögtön láthatjuk a telepítendő csomag nevét. Példa egy keresésre: keressünk rá a startx szóra, amit a legkülönfélébb módokon tehetünk meg. A legegyszerűbb talán a `grep` program használata, lásd a listát. Mivel tömörített fájlról van szó, nem lehet egyszerűen csak keresni a fájlban. Segítségül hívtuk a `gunzip` programot, kicsomagoltuk a `-c` kapcsoló segítségével a képernyőre, és a kibontott szöveget egy csővezetékben átadjuk a `grep` programnak. Az eredmény jól látszik: maga a `startx` parancs az ***x11/xbase-clients*** csomag része, így ha valóban ezt kerestük, az `apt-get install xbase-clients` paranccsal telepíthetjük is.

Keresés a fájlban

```
csontos@sharon:~/woody$ gunzip -c Contents-i386.gz | grep startx
usr/X11R6/bin/startx                        x11/xbase-clients
usr/X11R6/man/ja/man1/startx.1x.gz          x11/xmanpages-ja
usr/X11R6/man/man1/startx.1x.gz             x11/xbase-clients
usr/bin/startxfce                           x11/xfce
usr/share/X11R6/man/it/man1/startx.1x       doc/manpages-it
usr/share/apps/ksysguard/icons/locolor/16x16/apps/startx.png
   ➥x11/kdebase
usr/share/man/man1/startxfce.1.gz           x11/xfce
usr/share/man/pl/man1/startx.1X.gz             doc/manpages-pl
csontos@sharon:~/woody$
```

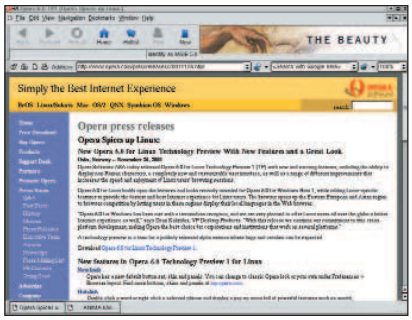
AbiWord 0.9.5

Az AbiWord fejlesztői is megörvendeztetik a Linux-felhasználókat – immár a 0.9.5-ös változattal, amely már meglehetősen közel került az 1.0-s üzembiztos változathoz. Valószínűleg sok újítás már nem fog bekerülni a végleges változatig, inkább a meglévő hibákat próbálják meg kijavítani.



Az újdonságok listája:

- Normal Mode, mint az MS Wordben, így több hely marad a szövegnek,
- a könyvjelzők támogatása,
- hivatkozástámogatás,
- új és szebb ikonok,
- egyszerű sablontámogatás,
- fájl beillesztése,
- MS Word-fájlok jelszavas védelmének támogatása,
- File –>Revert,
- jpeg képformátum támogatása,
- fejléc, illetve lábléc mentése rtf formátumban,
- rtf-listák beolvasása,
- az oldalszámozás minden fejezetben változtatható,
- Perl-héjkiegészítés,
- az állapotjelző sor kiegészítése,
- számos bővítmény (plug-in) és képkezelő-kiegészítés,
- MS-Write fájlok olvasása,
- parancssoros nyomtatás,
- a KWord-fájlok beolvasásának javítása,
- számtalan hibajavítás.



Bár nem tartozik szorosan a programhoz (igaz, ez nézőpont kérdése), de a honlapjukat is teljesen átalakították.
➲ http://www.abisource.com

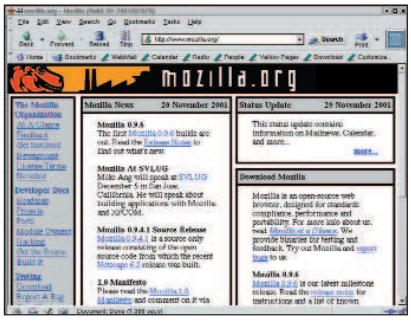
Opera 6.0

Az Opera böngésző ismét megnövelte eggyel a változatszámot, immáron a 6.0 TP (Technology Preview), azaz „szakmai előnézet" a letölthető próbaváltozat fedőneve. Hirtelen váltás ez, hiszen az előző a 5.05 TP volt, az ugrás bizonnyal a sok kisebb-nagyobb változásnak köszönhető. Felhívnám azonban mindenkinek a figyelmét, hogy ez a változat korántsem minősíthető üzembiztosnak! Működik minden általam kipróbált szolgáltatás, de ez nem jelenti azt, hogy ***minden*** és hibátlanul működik – tehát ajánlom a körültekintő használatot.
Az újdonságok közül talán a legszembetűnőbb a külső megváltozása, amely szerintem sokkal barátságosabb lett: az ikonokat lecserélték (szép kékek lettek), de természetesen a régi felületet kedvelők nyugodtan visszacserélhetik az eredetiekre, mivel a program készítői nem száműzték őket örökre (a ***File,***

Preferences, General, Look-ban tudjuk megtenni). A másik nagy újdonság a *Hotclick*: ennek segítségével, ha a böngésző ablakában duplán kattintunk egy szóra, egy menüt kapunk, ahol kiválaszthatjuk, hogy mit is szeretnénk vele csinálni: másolni, keresni (valamelyik keresőmotorral), kaphatunk szótárat, vagy lefordíthatjuk valamilyen nyelvre (sajnos a magyar nem szerepel köztük). A *Personal Bar*-ban egy helyen tudjuk a kereséseinket és a könyvjelzőket kezelni, ezeket sorba rendezhetjük, kereshetünk bennük vagy saját elrendezést is készíthetünk. Új a süti- (cookie) kezelő rész, a *Quick Preferences* F12 segítségével pedig könnyedén elérhetjük Operánk legfontosabb beállítási lehetőségeit. Ez az a változat az Opera életében, amelyik támogatja a Unicode-karaktereket, és amelyben egyszerűbb lett az ablakok kezelése – a CTRL+TAB billentyűkkel válthatunk közöttük. Kapunk egy *Contact List* nevű eszközt is, amelyben a barátaink, munkatársaink adatait tárolhatjuk weboldalukkal és levélcímükkel együtt. A bővítmények támogatása pedig sokat javult a 5.05 TP-hez képest, most már a legtöbb Netscape-bővítmény is működik, mint a Macromedia Flash, az Acrobat Reader, a Real Player, a Java, a Plugger, a TCL 2.0 és a Codeweaver Crossover (Apple Quicktime).
A teljes súgót újraírták, és a saját ablakban böngészhetjük.
➲ http://www.opera.com

Mozilla 0.9.6

A Mozilla böngésző is egyenes úton halad a megdicsőülés felé, amit szintén az 1.0-s változatszám elérése jelent majd. Érdemes összehasonlítgatni a linuxos programok változatszámait: amint láthatjuk, ebben a hónapban szinte minden programnak a 0.9.x-es változatát tudjuk a mellékletre tenni.



Ez reményeim szerint azt jelenti, hogy még ebben az évben (sajnos, így akkor is csak jövőre kerülhetnek a CD-re), de ha nem is idén, akkor a jövő év elején mindhárom programnak megjelenik a megbízható változata (azt hiszem, a megbízhatósággal e programok utóbbi változatainál sem volt sok gond). A Mozilla mostani változata képes megjeleníteni a Windows .BMP és .ICO formátumú képeit, működik a nyomtatási előkép (*Print Preview*), sokat változott a levelezőrendszer, továbbá lehetőség nyílik, hogy a weboldalon kijelölt szóra rákeressünk az Interneten.
➲ http://www.mozilla.org

Mandrake-frissítések

Mostani CD-mellékletünkön az októberi korongon található Mandrake Linux 8.1-es rendszerhez eddig megjelent összes frissítést közreadjuk, hogy olvasóink biztonságban érezhessék magukat és a rendszerüket.
Érdemes azoknak is frissíteniük, akik nem kapcsolódnak a Világhálóhoz. Valószínű, hogy időközben újabb és újabb frissítések jelennek meg, amelyeket a ➲ http://www.linux-mandrake.com/en/security/mdk-updates.php3?dis=8.1 címen nézhetünk meg, innen a megfelelő fájlokat a frissítéshez azonnal le is tölthetjük.

- Wu-FTPd – kijavították a távolról történő jogosulatlan hozzáférést;
- Postfix – a lehetséges távoli DoS-támadások kiküszöbölése;
- Apache – számos hibát kijavítottak;
- Squid – a lehetséges DoS-támadások kiküszöbölése;
- Expect – kijavították a lehetséges jogosulatlan hozzáférést;
- pspell;
- tetex – biztonságossá tették az ideiglenes fájlok hozzáférését;
- rendszermag 2.4 és 2.2 – számos biztonsági hibát kijavítottak;
- Mozilla – az Euro karaktereket most már helyesen jeleníti meg.

Ez a lista közel sem teljes, ezért érdemes szétnézni a 24-es CD Mandrake_frissitesek/RPMS könyvtárában, ahol több mint 150 MB-nyi anyag szerepel – ez, mint már említettem, a CD lezárásának időpontjáig megjelent összes frissítést és hibajavítást tartalmazza.

Rendszermag

Úgy tűnik, a 2.4-es rendszermagsorozat kettesével szedi a lépcsőt, mivel már másodjára történik meg, hogy két változat szinte ugyanakkor jelenik meg. Általában olyan hibákat kell sürgősen kijavítani, amelyek eléggé érzékeny részei a rendszernek. A jelenlegi változatszám a 2.4.16-os, a két lépésben

végrehajtott változások elolvashatóak az alábbi címeken:
➲ http://www.kernel.org/pub/linux/kernel/v2.4/ChangeLog-2.4.15
➲ http://www.kernel.org/pub/linux/kernel/v2.4/ChangeLog-2.4.16
Immáron a 2.5-ös fejlesztői sorozat első „lépései” is elérhetők – CD-mellékletünkön szintén megtalálhatják őket. Figyelem: szigorúan csak hozzáértőknek ajánlom őket! Egyelőre még nincs olyan sok változás a 2.4-es sorozathoz képest, ez azonban nagyon gyorsan meg fog változni és számos újdonság kerül bele, aminek a vége természetesen a 2.6-os, esetleg a 3.0-s sorozat lesz.

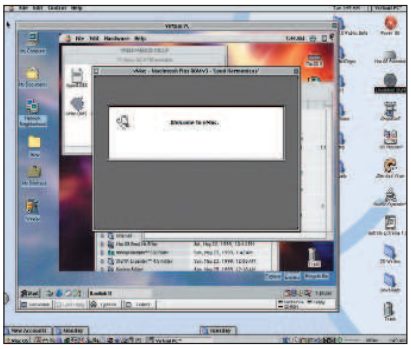
OpenOffice 641c

A Sun Microsystem már elég érettnek találta az OpenOffice-fejlesztést ahhoz, hogy elkészítse a StarOffice 6.0 próbaváltozatát, ami letölthető a Sun honlapjáról. Így ez a fejlesztés is egyre megbízhatóbban, egyre kényelmesebben – és ami nagyon fontos, a linuxos irodai csomagokban egyre „magyarabbul” használható. A fő kényelmi szempont az ékezetes betűk helyes kezelése, mindenféle külön beállítás nélkül lehet vele *ŰűŐőÁáÉéÖöÜüÍí* betűket gépelgetni, és nem kell varázslóvá vagy Linux-guruvá válnunk ahhoz sem, hogy ezeket a betűket is elérhessük (ez a legtöbb Linuxra érvényes, legalábbis a Debian Woodyra, a Mandrake 8.1-re és a RedHat 7.1-re biztosan). Ezek a karakterek nyomtatáskor is megjelennek a papíron, tehát már azt a kellemetlen tulajdonságát sem őrizte meg, hogy a képernyőn minden szépnek és jónak látszik, a papíron azonban ? vagy valamilyen más „érdekes” karakter bukkan fel a vágyott helyett.

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

MacOS Linuxon

A Basilisk II nevezetű programmal Macre készült programokat vagyunk képesek futtatni a legkülönfélébb gépeken és operációs rendszereken, közéjük tartoznak az x86-os gépek is. A MacOS összes változatát képes futtatni egészen a MacOS 8.1-ig a hozzá tartozó programok nagy részével együtt. Megoldást jelenthet mindenki számára, akik különböző gépeket és operációs rendszereket használnak, mivel így mindenhol egységes felületet kapnak kézhez.



A honlapján a következő támogatott rendszereket sorolták fel:

- BeOS R4 (PowerPC és x86, valamint már BeOS R5-höz is);
- Unix X11 grafikus felülettel (Linux, Solaris 2.5, FreeBSD 3.x és IRIX 6.5);
- AmigaOS 3.x;
- Windows NT 4.0 (Windows 9x).

A *képen* látható helyzet az emulátorok használatának egészen furcsa összetételét szemlélteti: egy MacOS futtat VirtualPC-t Windowszal, amelyben egy BasiliskII MacOS-sel fut, ebben pedig egy vMac (szintén Mac-emulátor). Bár ez kissé értelmetlennek tűnhet, mindenesetre bizonyítja, hogy a különféle rendszereket megbízhatóan „össze lehet keverni".
Használatbavételéhez mindössze az emulátor állományainak beszerzésére, valamint egy olyan fájlra lesz szükségünk, amely az eredeti ROM-ot hivatott helyettesíteni (ezt hivatalosan sajnos nem lehet letölteni, mivel nem terjeszthető szabadon, így mindenképpen egy eredeti Macre lesz szükségünk). Végül a ➲ http://www.apple.com oldaláról letölthetjük a teljes MacOS 7.5.3-as operációs rendszert. A telepítés folyamatáról a program honlapjáról kiindulva számos segítséget kaphatunk.

➲ http://www.uni-mainz.de/~bauec002/B2Main.html
➲ http://www.kearney.net/~mhoffman/basiliskii.html

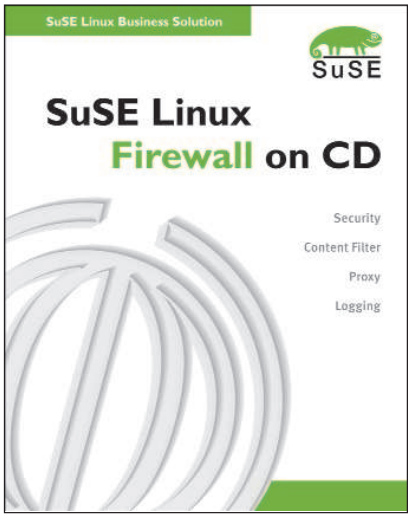
Linux két lemezen?

Egyre nagyobbak az operációs rendszerek – és egyre több helyet foglalnak el a merevlemezen. Talán pont emiatt próbálkoznak sokan az egy-két kislemezes rendszerekkel. Íme egy újabb, két kislemezes Linux-rendszer, ráadásul X-es felülettel. Készítői bebizonyították, hogy lehetséges apró, de használható rendszert is építeni.

➲ http://www.mungkie.btinternet.co.uk/projects/2diskXwin.htm

SuSE VPN

A SuSE VPN (Virtual Private Network – virtuális magánhálózat) segítségével biztonságosan köthetők össze a vállalatok különböző telephelyei az Interneten keresztül.



A SuSE Linux Firewall különlegessége az úgynevezett Live- (élő) rendszer. A módszer lényege, hogy a rendszer nem települ a számítógép merevlemezére, hanem CD-ről működik. Ez nagy előnyt jelent a biztonság tekintetében, mivel a CD-ROM-on lévő rendszerbe nem nyúlhatunk bele. A tűzfal beállítása – például a csomagszűrőé – írásvédett hajlékonylemezen tárolható. A csomag részét alkotó forráskód az egyéni igényekhez szabást is lehetővé teszi.
A csomag második CD-jén a tűzfalkarbantartási rendszer (FAS) található, amely akár több tűzfal felügyeletét is képes grafikus felületen keresztül ellátni. A SuSE Linux Firewall on CD a standard kiadás mellett mostantól VPN-kiadásban is kapható. A csomag a rendszer mellett 3 CD-t, átfogó leírást, 30 napos telepítési segítséget és egyéves programkarbantartást is magában foglal.

➲ http://www.suselinux.hu

Zorp 1.4

A BalaBit IT Biztonságtechnikai Kft. terméke, a Zorp moduláris alkalmazásszintű tűzfal, legújabb változata az 1.4-es. A fejlesztés célja, hogy nagyon érzékeny környezetben is jól alkalmazható tűzfalat hozzanak létre. Segítségével a beágyazott protokollok (például a HTTPS) forgalmának felügyelete és szűrése is lehetővé válik. A cég honlapjáról ingyenesen letölthető a program GPL-es változata is, természetesen ebben nincs benne minden lehetőség.

➲ http://www.balabit.hu
➲ http://www.balabit.hu/downloads/

Járjunk könyvtárba a Weben

Széles kínálatú számítástechnikai könyvtárat találhatunk az alábbi címen, amelyet gyors feliratkozás után azonnal használatba is vehetünk. Fontos, hogy mindenképpen létező és számunkra hozzáférhető címet adjunk meg, mert erre kapjuk meg a jelszavunkat.

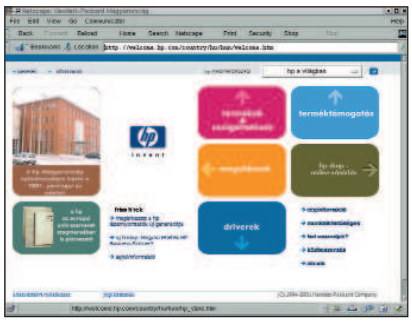
Mindenki kedvére válogathat a különféle könyvek közül, akár Java, Unix, Perl vagy bármilyen más témában keresgél is szakirodalmat. Témakörökre bontva több mint ötszáz kötet anyaga közül válogathatunk, közülük némelyek letölthetők, némelyek csupán on-line olvashatók. Letölthető például az ígéretes című „100 Linux tipp és trükk" is – nem csak vállalkozó kedvűeknek!

➲ http://www.syndik.at/harald/regal/buecher/bibliothek.asp

HP DVD+RW

A Hewlet-Packard bemutatta az első DVD+RW módszerre épülő, újraírható DVD- és CD-lemez meghajtóját. Az új HP DVD-Writer dvd100i üzleti és magánfelhasználói gyorsan és egyszerűen készíthetnek saját DVD- és CD-felvételeket. A DVD+RW módszer kifejlesztésekor kezdettől fogva szem előtt tartották a különféle szórakoztatóelektronikai és személyi számítógépes környezetek tökéletes együttműködésének fontosságát. A HP DVD-Writer dvd100i ráadásul az interaktív, dinamikus videofelvételek készítését, másolását és lejátszását is támogatja, és a nagy mennyiségű adattárolást is lehetővé teszi.
A DVD+RW lemezek 4,7 gigabájtos adattárolási képességgel rendelkeznek, ami háromórányi videofelvételi, vagyis -visszajátszási időnek, illetve 7 CD méretének felel meg. A DVD+RW adathordozók a legtöbb jelenleg kapható DVD-ROM és DVD-videolejátszóval képesek együttműködni. A CD-R adathordozók több mint egymilliárd CD-ROM- és CD-lejátszóval használhatók világszerte. Néhány jellemző:

- kombinált újraírható DVD-, illetve DVD-lemezmeghajtó;
- DVD – legalább 2,4× újraírási, illetve 8× olvasási sebesség;
- CD – legalább 12× írási, 10x újraírási, illetve 32× olvasási sebesség;
- 4,7 GB DVD;
- 600–700 MB CD;
- IDE, illetve ATAPI DVD+RW/CD-RW belső meghajtó.



➲ http://www.hp.hu

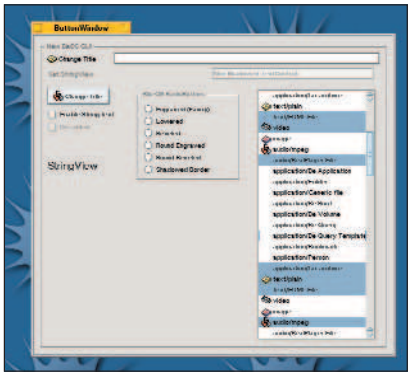
A HP termékekről az alábbi címeken tudhatunk meg részletes adatokat:

➲ http://www.hp.hu
➲ http://www.hp.comhttp://www.hp.hu/sajto/default_media.asp

A Be és a Palm

Befejeződött a Be Inc. felvásárlása, amelyre a Palm összesen 11 millió dollárt költött. Néhány fejlesztő megtartásával valószínűleg az eddigi megoldásokat szeretnék a saját rendszerükben is hasznosítani. Arról, hogy a Palmnak milyen szándékai vannak a sokak által kedvelt BeOS operációs rendszerrel, nem szólnak a hírek. A bizonytalanság azonban nem tesz (tett) jót a rendszernek, mivel senki sem fejleszt szívesen „halott rendszerre". Szerencsére a ➲ http://www.bebits.com-on látható változások nem ezt igazolják! Emellett különféle ötletek is napvilágot láttak, köztük szerepel egy másik operációs rendszer magjára átültetett BeOS-felület – ez lehetne a Linux, az ATheOS; de ilyen fejlesztés például a ScaraBeOS és a BlueOS is. A BlueOS különösen érdekes, mivel Linux-rendszermagra épül – a *képen* látható, mire is jutottak a fejlesztők. Forráskódszinten a BeOS- és a Linux-kódokat is képes lesz kezelni, így a felhasználói programok nagy részének kérdése megoldódik.

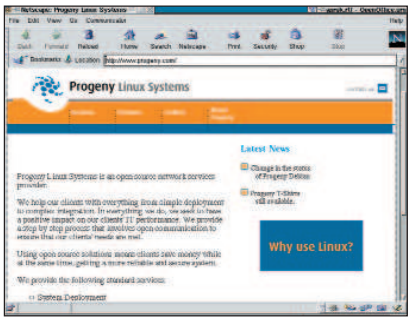
➲ http://www.beos.com



➲ http://blueos.free.fr
➲ http://www.bebits.com
➲ http://http://www.palm.com
➲ http://open-beos.sourceforge.net/

Viszlát, Progeny!

Újabb üzleti Debian-változat dobta be a törölközőt. Sajnos a Progeny Linux is feladta a versenyt: 1.0-s Newton-változatuk ugyan még mindig letölthető a honlapjukról, támogatást is adnak hozzá, de nem fejlesztik tovább, mivel



üzletileg nem éri meg. Október elsején hagyták abba a fejlesztést, a közvetlen eladások október 25-ig folytak, most már a teljes változatot csak néhány viszonteladótól lehet beszerezni. December 31-ig a termékhez járó ingyenes harmincnapos telefonos és három hónapos levélbeni támogatás természetesen még él. A Woodyra átálló felhasználók a ➲ http://www.progeny.com/archive/debian/support/conversion.html oldalon kaphatnak segítséget, ahol részletes leírást olvashatnak a programok frissítéséről, telepítéséről és az átállásról.

➲ http://www.progeny.com

Sun Cobalt

Hol volt, hol nem volt, volt egyszer Cobalt nevű cég, amely hetedhét erdőn túl szép kék dobozokat gyártott. E dobozok világítottak a sötétben, vezetékek kapcsolódtak hozzájuk, és a következő célfeladatokat látták el: nagy teherbírású proxykiszolgáló, vállalati tűzfal és belsőhálózat-kiépítés. Annak idején e dobozok nagy erősségének számított, hogy a testreszabott Linuxot webes felületen keresztül is el lehetett érni, és akár a kezdő rendszergazdák is pillanatok alatt beállíthattak egy Apache- vagy proxykiszolgálót. Már akkortájt látszott, hogy sikeresen megállja a helyét a piacon, amennyiben néhány hiányosságát kiküszöbölik, például a proxykiszolgálót nem IDE-csatolófelületű lemezekkel szerelik (a párhuzamos terhelést ugyanis nehezen bírja), valamint a futtatott Linux-terjesztést biztonsági szempontból folyamatosan karbantartják…

A termék piacra kerülése óta eltelt két év folyamán a Sun megvásárolta a Cobaltot.

Lássuk, hogy ez a hasznos kis masina mára mennyit változott! Jelenleg szinte minden vállalati felhasználási területet sikerült lefedniük, a különböző célfeladatoktól kezdve (behívókiszolgáló) az általános felhasználásig (Web). Időközben a gyerekbetegségeit is kinőtte, a proxyba SCSI-merevlemez került, és a programot is továbbfejlesztették. Már az üzembe helyezés során kiderül, milyen kényelmes eszközt kaptunk kézhez: a Sun Cobalt-termékcsaládra nagyon egyszerűen telepíthetünk operációs rendszert. A készüléket keresztkábel segítségével összekötjük a saját gépünkkel, majd a gépünket CD-ről újraindítjuk. Ilyenkor a két gép között a kapcsolatot egy átmenetileg működő DHCP-kiszolgáló biztosítja. A telepítés menetébe nem szükséges beavatkozni – miután elkészült, a Cobalt kijelzőjén megjelenik a felirat: újraindíthatjuk. Ezentúl gépünket már webes felületen keresztül is felügyelhetjük. Ez azonban rögvest egy gondot is felvet: a 80-as kapun átmenő forgalom alapesetben kódolás nélkül kerül átvitelre, így például beíráskor egy figyelő lefülelheti a jelszavunkat, érdemes tehát már az elején bekapcsolni a biztonságos HTTPS-támogatást. Amennyiben az alaptelepítést a gépünkön felejtjük, számtalan rést hagy rajta (Telnet-kapu, `finger`, `time`, `portmap` stb.), de szerencsére saját magunk is könnyen ki tudjuk kapcsolni őket. Továbbá célszerű a Telnet protokollt azonnal a jóval megbízhatóbb SSH-ra cserélni (cobalt formátumban érhető el hozzá). A fenti leírásból következik, hogy mind webes felügyeletre, mind héjhozzáférésre lehetőségünk nyílik. Dicséretes újítás, hogy a biztonsági frissítéseket kényelmes módon a webes felügyeleti felületen tudjuk beállítani, sőt, akár önműködővé is tehetjük. Ez a lehetőség leginkább a Debian GNU/Linuxban már megismert `apt-get update/upgrade` szolgáltatásra emlékeztet. Fontos figyelembe venni, hogy ezeket az eszközöket úgy alakították ki, hogy a felhasználó által könnyen felügyelhető legyen, ezért mindenképpen ajánlott vagy egy jól beállított tűzfal mögé (lásd *Linuxvilág* 5. szám, 40. oldal) helyezni, vagy magára a gépre tűzfalat telepíteni.

Alapvetően három fő lehetőséggel gazdálkodhatunk:

1. A Sun *Qube* termékcsaládjához úgynevezett „Adaptive Firewall" csomagot (állapottartó csomagszűrő) biztosít, amelyet egy biztonságos webböngészőből könnyedén beállíthatunk. Ha a cél az, hogy megvédjünk egy webkiszolgálót egy kiszolgálózónában, és nem szándékozunk újabb gépet kötni a Cobalt elé, ez nagyon jó megoldás lehet. Sajnos a program nem szabad, csak ingyenesen használható.
2. Astaro Security Linux: önálló Linux-terjesztés, amely a rendszergazdák álmát valósítja meg egy CD-n. A nevéből is kitűnik, hogy fejlesztése során a biztonságra törekvés volt az elsődleges szempont. Bár az Astaro cég nemcsak Cobaltra készít terjesztést, hanem például saját tűzfalat is előállít, az 1.8-as és a 2.0-s (nem tesztelt, de üzembiztos) változatok azonban Cobaltra is elérhetők, és együttműködnek a cobaltossal. A telepítése a cobaltoséhoz hasonló módon zajlik, felügyeletet már csak HTTPS-kapcsolaton keresztül gyakorolhatunk. Szinte mindent kattintva tudunk beállítani, még a VPN-t is. Jó megoldást kínál olyan cégeknek, amelyek 1 hüvelyk magas tűzfalat szeretnének gyorsan és megbízhatóan üzemeltetni, és ne feledkezzünk meg róla: a tűzfalrész ebben is csupán állapottartó csomagszűrő!
2. Zorp OS: lehetőségünk nyílik rá, hogy egy Debian GNU/Linux-alapú moduláris proxytűzfalat használjunk. Ez jelen pillanatban még kísérleti állapotban található, de a Zorp GNU-s változatát a Cobaltra tölthetjük és beüzemelhetjük. Ebben az esetben mind a cobaltos készülék, mind a Zorp tűzfalprogram összes jó tulajdonságát ki tudjuk használni. Amennyiben a Cobaltot kifejezetten és egyedül tűzfalként akarjuk használni, tehát semmilyen kiszolgálót (például Apache) nem szeretnénk rajta üzemeltetni, a Cobaltra is hamarosan megjelenő Zorp OS-t használhatjuk. Ezzel kapcsolatban a legfrissebb hírekért a ➲ http://www.balabit.hu címet keressük fel.

Kapcsolódó címek

A **Sun Cobalt** magyarországi forgalmazója ➲ http://www.avnet.com
Zorp tűzfal ➲ http://www.balabit.hu
Astaro security Linux ➲ http://www.astaro.com
Cobalt oldal ➲ http://www.suncobalt.hu

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.

Séta a LinuxWorld beágyazott oldalán

A CodeWeavers képes olyan megoldást biztosítani, amellyel beágyazott alkalmazások futtathatók Linux alatt, a Wine segítségével…

Annak ellenére, hogy az ez évi nagy linuxos esemény, a West Coast LinuxWorld hangulata a szokásosnál visszafogottabb volt, egyvalamiben szinte teljes volt az egyetértés: a beágyazott Linux az utóbbi 6–12 hónapban hatalmas fejlődésen ment keresztül. Ugyan a rendezvény nem különösebben összpontosít a beágyazott rendszerek piacára, mégis rengeteg olyan új terméket, műszaki megoldást és fejlesztési tervet fedtek fel és mutattak be, amelyek a beágyazott rendszerek és a különféle okos eszközök fejlesztőinek igényeit hivatottak kielégíteni. Az ember bármerre is fordult, mindenhol olyan cégekbe botlott, amelyek zsebtitkárokba, szórakoztató készülékekbe (elsősorban tévé set-top-boxokba), telematikai eszközökbe és vékony ügyfelekbe ágyazták be a Linuxot.

A korábbi kiállításokon a beágyazott Linuxszal kapcsolatos termékek és megoldások a bemutatott termékeknek körülbelül tíz százalékát tették ki. A legutóbbi rendezvényen ez az arány 15–20 százalékra nőtt, ami cseppet sem meglepő, hiszen a VDC, az Evans Data Corporation, az Embedded Systems Programming magazin, a LinuxDevices.com és még sokan mások jelentéseikben számottevő piaci növekedést jeleztek a beágyazott Linux iránt.

Azoknak, akik nem tudtak eljönni a rendezvényre, vagy más kötötte le a figyelmüket, következzék hagyományos beszámolóm a LinuxWorld beágyazott oldaláról, amelyben ábécésorrendben haladva az összes beágyazott és beágyazható dologról említést teszek.

1.

1. kép Kisméretű linuxos számítógép az Advantage Business Computer Systemstől

2. kép Az Earthlink K+F telematikai kutatófelülete Linuxot futtat

3. kép Linux a nappaliban? A HP új Digital Entertainment Centere

4. kép Az IBM koncepcióautójában Linux fut

5. kép Nézd, mire képes a Tiny StrongARM SBC Plus beágyazott Linuxszal!

Applied Data Systems

Az ADS standjának látogatói meghökkentő mennyiségű LCD-kijelzős, StrongARM processzoros, egyetlen áramköri lapból álló számítógéppel találkozhattak – a kütyürajongók paradicsoma. Az ADS saját beágyazott Linuxokat készített, amelyek választható kiegészítőként bármelyik SBC-jéhez elérhetők, de emellett három Java-képességekkel is rendelkező, grafikus alkalmazáskörnyezetet is bemutatott: az Insignia Jeode VM fejlesztése a Lineo Embedix fejlesztésén, az IBM Visual Age Micro Edition a MontaVista Hard Hat Linuxán, a Blackdown JDK-ja pedig az ADS saját beágyazott Linuxán fut. Az ADS augusztusban jelentette be, hogy az Intel a StrongARM és XScale processzoraihoz a céget választotta külső felületfejlesztőjéül, és kifejleszti jelenlegi Graphics Master SBC-jének módosított változatát, amely hivatkozási alaprendszerként fog szolgálni az Intel ügyfelei számára.

➲ http://www.applieddata.net

Advantage Business Computer Systems

A Linux Terminal Server Project (LTSP) standján elrejtőzve mutatta be *David Anders* saját cégének apró Linux-rendszerét – egy valóban remek kis linuxos gépet. Anders elmondta, hogy a hamarosan megjelenő újabb változat már CompactFlash aljzattal is rendelkezni fog, így lehetővé válik a gép bővítése, valamint az eltávolítható adathordozók használata. Szép darab.

➲ http://www.abcsinc.com

Century Embedded Technologies

A Century megszokott helyét foglalta el a RedHat hatalmas pavilonjának aljában. A Century nagy újdonsága a rendezvényen a PIXIL volt, amely korábbi kulcstermékeit (Microwindows és ViewML) feljavítva és továbbfejlesztve egyesíti egy új grafikus környezetből, eszközökből és alkalmazásokból álló készletbe, amelynek célterülete a Linux-alapú zsebtitkárok, webtáblák és vékony ügyfelek piaca. A bemutatók során a PIXIL-t Compaq iPAQ zsebtitkáron láthattuk futni, valamint megtekinthettünk egy PIXIL-alapú set-top-boxot, amely a National Semiconductor SP1SC10 típusú fejlesztői készülékén alapult. *Greg Haerr* – a Century elnöke – kiemelte, hogy ugyan a TV Linux Alliance Project csak a szabványosítási munka kezdetén tart, a Century/National újonnan bejelentett Linux4.TV fejlesztése már elérhető a felhasználók számára. Támogatja a digitális és analóg tévéket, a filmrögzítést, továbbá rendszermag- és középszintű programozási felületet is biztosít.

➲ http://embedded.censoft.com

CodeWeavers

A Codeweavers bemutatta a nemrég bejelentett CrossOver megoldást, amely lehetővé teszi, hogy linuxos rendszereken windowsos böngészőket és alkalmazássegítőket használjunk. *Jeremy White* elnök szerint a CrossOver beágyazott változatának fejlesztése már folyamatban van, átlagos esetben 1 MB memó-

riára és 4,5 MB tárhelyre lesz szüksége. A windowsos beépülő modulok időnként sajnos meglehetősen sok erőforrást igényelnek, ha CrossOver alatt futtatjuk őket. A CodeWeavers, amely jelentős hátteret jelent a Wine számára, képes olyan megoldást biztosítani, amellyel beágyazott alkalmazások futtathatók Linux alatt, a Wine segítségével, valamint közvetlenül is át tudja ültetni az alkalmazásokat Linux alkalmazásfejlesztői felületekre.
➲ http://www.codeweavers.com

Az Earthlink kutató- és fejlesztőcsoportja

A csoport egy nyílt szabványokra épülő, járművek számára készített helymeghatározó telematikai felület mintapéldányát mutatta be, amelyet közlekedési alkalmazások bemutatására, valamint arra terveztek, hogy tanulmányozni lehessen a linuxos alkalmazások és internetes megoldások használatát gépjárművekben, valamint távoli hibafelderítő, m-kereskedelmi és helyzetfüggő alkalmazások megvalósíthatóságát és hatékonyságát. Az Earthlink egy pályázatot is kiírt a Linux, az XML, a Java, a vezeték nélküli és a webes alkalmazások fejlesztői számára. Elég egy jó ötlet, és máris valamelyik érdekes gépükkel játszadozhatunk.
➲ http://www.research.earthlink.net

Embedded Linux Consortium

Az ELC bejelentette, hogy a közelmúltban számos új taggal bővült, köztük található a Future Sound Technologies, az Intel Corporation, az American Megatrends, Inc., az Aleph One, Ltd. és a Vibren Technologies is. Az ELC honlapján található lista szerint jelenleg 68 vállalati tagot számlálnak (36 igazgatási és 32 támogató). Fontos apróság a hírek zuhatagában, hogy az ELC igazgatótanácsa nemrég elfogadta az ELC szabályzatának azon módosítását, amelynek értelmében a szervezet a jövőben szellemi tulajdonnal is rendelkezhet. Ezzel szabaddá vált az út afelé, hogy az ELC beágyazott Linux-szabványokat fejlesszen ki, és engedélyeztessen mások számára.
➲ http://www.embedded-linux.org

Embedded Linux Journal

A LinuxWorld nyitásának napján az ELJ kihirdette az első beágyazott Linux tervezési verseny győzteseit. A győztesek Costa Ricába utazhatnak, minden költségüket a szervező állta.
➲ http://embedded.linuxjournal.com

2.


3.

4.

5.

Empower Technologies

Az Empower bemutatta új LinuxDA nevű „Linux-frissítését” Palm III és V készülékekhez. Úgyszintén bejelentették, hogy a LinuxDA használati jogát megvette két, a Palmokkal egyenértékű készülékeket gyártó tajvani vállalkozás: az Elitegroup Computer Systems és az APlux Communications.
➲ http://www.linuxda.com

Hewlett-Packard Company

A Hewlett-Packard jelentős beágyazott Linux cselekvési tervet jelentett be a LinuxWorld rendezvény kapcsán. A bejelentés szerint a Linuxot választották az összes jövőbeni HP-fejlesztésű eszköz operációs rendszeréül. A HP beágyazott Linux-rendszerének neve Chai-LX. A HP bemutatta új Digital Entertainment Center hangrendszerét is, amely x86-os típusú processzort tartalmaz, és beágyazott operációs rendszerként Linuxot futtat.
➲ http://www.hp.com/linux

IBM

Az IBM alphaWorks új, kísérleti TechMobile fejlesztését mutatta be, egy módosított Ford Explorer 2002 Limited Edition terepjárót, amelyet Linuxot használó, különféle webes alkalmazásokat futtató számítógépekkel szereltek fel. A bemutató során egy egyszerű felhasználói felülettel és Bluetooth-kapcsolattal rendelkező zsebtitkárról vezérelték a gépkocsi fényszóróit, ajtózárait és indítómotorját. A rendezvény különböző pontjain további, a beágyazott Linuxszal kapcsolatos műszaki megoldásokkal lehetett találkozni az IBM háza tájáról, így például DB2 adatbázisokkal, VisualAge Micro Edition (a MontaVista standján) és Embedded ViaVoice (szintén MontaVista) fejlesztésekkel.
➲ http://www.alphaworks.com

Kapcsolódó cím

➲ http://www.empowerthechnologies.com

Rick Lehrbaum (rick@linuxdevices.com) hozta létre a LinuxDevices.com „beágyazott Linuxok portálját”, amely nemrég visszatért alapítójához, miután az LLC a Device Forge-ot. 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.

Egy Bluetooth-kapcsolattal rendelkező zsebtitkár vezérelte az autó fényszóróit, ajtózárait és indítómotorját

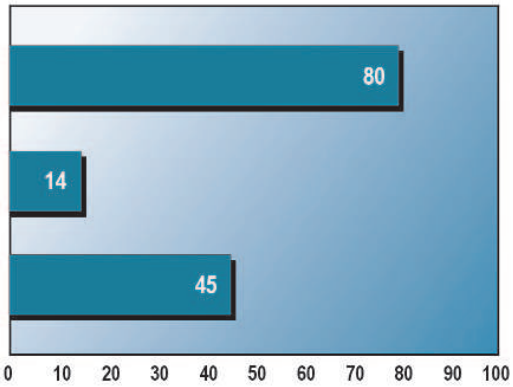
Biztató a jövő a beágyazott Linuxot illetően

Nem nagy újság, hogy a Linux mindenhol megállja a helyét, a „háztartási gépektől a szórakoztató elektronikai cikkeken át az iparban, a repülőkben és az autókban használt vezérlőberendezésekig” – olvashatjuk az Evans Data Corporation (➲ http://www.evansdata.com) jelentésében. Az viszont újdonságszámba megy, hogy milyen magas a Linux részesedése a beágyazott rendszerek területén.

Állításuk szerint a Linux fontos a beágyazott rendszereket fejlesztők közössége számára.

Már van linuxos alkalmazásuk.

Az elkövetkező évben linuxos alkalmazás megjelentetését tervezik.



A „Beágyazott rendszerek fejlesztői körében végzett felmérés” alapján az Evans Data „megdöbbentő változás”-ra hívja fel a figyelmünket „a munkahelyi és otthoni mikroprocesszoros eszközök programjai körében”. Ennek eredményeképpen a cég a beágyazott rendszerekkel kapcsolatos linuxos projektek megháromszorozódására számít a következő év során, ami véleményük szerint „része annak a folyamatnak, amelynek eredményeképpen a házilag fejlesztett operációs rendszerek helyét a kereskedelmi rendszerek veszik át”. Gondot okoz a megfelelő eszközök hiánya. Pontos számadatok közlése nélkül elmondhatjuk, hogy „élénk érdeklődés mutatkozott a fejlesztők munkáját elősegítő eszközök és termékek iránt, amelyek segítségével időre befejezhetik a munkájukat”. A megkérdezett fejlesztők kétharmada elmondta, hogy sok beágyazottrendszer-projekt elindul, de sohasem fejeződik be, még többnek a határideje jelentősen kitolódik.

Mit szeretnek a fejlesztők a Linuxban? Ahogyan várható volt, „a nyílt forráskódot, a felhasználási szerződésdíjak hiányát és a nagy szakértelemmel rendelkező, népes linuxos fejlesztői közösséget” jelölték meg a Linux legnagyobb előnyeiként. „A Linux lassú elfogadásának okai között kiemelkedő helyen szerepel az egyes kártyákat támogató csomagok és az eszközmeghajtók hiánya, valamint a kód széttöredezése.”

Doc Searls
(doc@ssc.com)
a Linux Journal szerkesztője és a Cluetrain Manifesto társszerzője

Ők mondták

Mutass egy olyan webes alkalmazást, amelyet nem lehet egy 100-as Pentiumról kiszolgálni, és én mutatok neked egy halott dot-com céget. (*Kevin Jamieson*)

Ha rendszereket szeretnél gyártani és eladni, és azt akarod, hogy a fejlesztők is melléd álljanak, a rendszerednek nyílt forrásúnak kell lennie. Ez a tény az utóbbi pár évben igaznak bizonyult. (*Dave Winer*)

Az operációs rendszerek háborúja véget ért: az operációs rendszerek vesztettek. Most olyan környezetek fejlődésének lehetünk a tanúi, amelyeknél az alkalmazások több operációs rendszert ölelnek át, és nem fordítva. A személyi számítógép mindössze egyetlen objektum a többi közül egy maroknyi kifelé látszó csatlakozási felülettel és összetett belsővel, amelyet azonban szabványos protokollok rejtenek el a hívó elől. (*Clay Shirky*)

A Webbel nem az a gond, hogy meg kell értenünk a szokatlant, hanem az, hogy rá kell jönnünk, mennyire szokatlan a hétköznapi. (*David Weinberger*)

Milyen kart kellett egy láthatatlan kéznek meghúznia, hogy felélessze a gépben lakozó szellemet? (*Christopher Locke*)

A műszaki fejlődés olyan, mint a balta az elmebeteg gyilkos kezében. (*Albert Einstein*)

Linux-index

1. A Szilícium-völgyben működő 807, tőkében legerősebb cég felsővezetésének az elmúlt pénzügyi évben kifizetett prémium összege: 4,8 milliárd dollár
2. A fenti szám az előző évihez viszonyítva: 2-szeres
3. A fenti cégek részvényárfolyamának esése – az MN Index alapján – ugyanebben az időszakban (százalék): 24
4. A „shit” szó ennyiszer fordult elő a Comedy Centralon sugárzott South Park című műsor első részében a képernyőn futó számláló alapján: 142
5. A Comedy Central a fenti South Park-résszel kapcsolatban ennyi elektronikus levelet kapott: 4
6. A káromkodást támogató levelek százaléka: 100
7. Az Egyesült Államok Szabadalmi Hivatala által jóváhagyott szabadalmak száma 2000-ben: 158 118
8. Az IBM helyezése a 2000-ben jóváhagyott szabadalmak száma alapján: 1
9. Az IBM-nek 2000-ben megítélt szabadalmak száma: 2 886
10. Az amerikai cégek száma a 10 legtöbb amerikai szabadalmat kapott cég között (2000-ben): 4
11. A japán cégek száma a 10 legtöbb amerikai szabadalmat kapott cég között (2000-ben): 6
12. A Webvan vesztesége 2001 júliusában, amikor csődbe ment: 860 millió dollár
13. A Google által bejárt weblapok száma: 1 346 966 000
14. A Google találatai alapján ennyi weblapon szerepel a „sun” kifejezés: 25 500 000
15. A Google találatai alapján ennyi weblapon szerepel a „microsoft” kifejezés: 20 200 000
16. A Google találatai alapján ennyi weblapon szerepel a „dell” kifejezés: 14 700 000
17. A Google találatai alapján ennyi weblapon szerepel a „solution” kifejezés: 13 300 000
18. A Google találatai alapján ennyi weblapon szerepel az „ibm” kifejezés: 11 200 000
19. A Google találatai alapján ennyi weblapon szerepel a „unix” kifejezés: 10 900 000
20. A Google találatai alapján ennyi weblapon szerepel a „perl” kifejezés: 7 650 000
21. A Google találatai alapján ennyi weblapon szerepel a „python” kifejezés: 2 070 000
22. A Google találatai alapján ennyi weblapon szerepel a „linux” kifejezés: 31 600 000
23. A Google által indexelt 1000 lapból ennyin szerepel a „linux” kifejezés: 2,35
24. Azok százaléka, akik az elsők között alkalmazták a PVR-t (Linux-alapú videorögzítő megoldás, mint például a TiVo), és ma többet néznek tévét, mint korábban: 63
25. Azok százaléka, akik használják a PVR-t, és nem tudják, hogy a felvett, majd később megnézett műsorok eredetileg melyik csatornán voltak láthatók: 12
26. Azok százaléka, akik PVR-rendszert vagy hagyományos videomagnót használnak és átugorják a hirdetéseket: 25
27. Az Open Source Initiative által elfogadott felhasználói engedélyek száma: 22
28. A vezető 74 márkából ennyi százalék vesztett márkaértékéből az elmúlt tíz esztendőben: 55
29. A 74 vezető márka ez idő alatt ennyit vesztett az értékéből: 5%
30. A múlt év során megjelent új termékek száma: 31 432
31. A zene felvételére alkalmas készülékek gyártói bevételük ennyi százalékát fizetik ki szerzői jogdíjként az American Home Recording Act (AHRA) értelmében: 2
32. Az AHRA szerzői jogdíjakból ekkora arányban részesedik a Sound Recording Found (Hangrögzítési Alap): kétharmad
33. A fent említett alapból ennyi százalék jut a „nem jegyzett zenészeknek és a vokálénekeseknek”: 4
34. A fennmaradó részből ennyi százalék jut a „jegyzett, a felvételt készítő művésznek”: 40
35. A fennmaradó részből ennyi százalék jut a „hangfelvételek másolására és terjesztésére az adott évben kizárólagosan jogosult tulajdonosnak”: 60
36. A RedHat aránya a Linux Counter által nyilvántartott gépek között: 28,39 %
37. A Slackware aránya a Linux Counter által nyilvántartott gépek között: 22,40 %
38. A Debian aránya a Linux Counter által nyilvántartott gépek között: 19,30 %

Forrás

1–3: San Jose Mercury News
4–6: The New Yorker
7–11: United States P atent and Trademark Office
12: The Wall Street Journal
13–23: Google, 2001. július 12.
24–25: Electronic Media a NextResearchből
26: The Wall Street Journal
27: Open Source Initiative (➲ http://www.opensource.org)
28–30: Financial Times
31–35: RIAA
36–38: Linux Counter

Cégcsokor (4. rész)

Sorozatunkban olyan cégeket gyűjtünk csokorba, amelyek huzamosabb ideje számos területen Linuxot alkalmaznak.

MT-Telecom Kft.

Elsőként lássuk, hogy milyen kiépítést használ ez a cég: három linuxos kiszolgáló üzemel náluk, amelyek Debian Linuxot futtatnak.
Az első gép fájlkiszolgálóként működik: 400 MHz-es Celeron processzoros, 128 MB RAM-mal, egy 4 GB-os és két 13 GB-os merevlemezt tartalmaz programból megvalósított RAID-del. Az alábbi alkalmazások futnak rajta:

- fájlkiszolgáló – Samba,
- faxkiszolgáló – Hylafax,
- DNS-kiszolgáló – Bind,
- DHCP-kiszolgáló – DHCP.

A második vas tűzfalként üzemel: 400 MHz-es Celeron processzorral van felszerelve és 64 MB RAM-mal. Proxykiszolgálónak Squidet, valamint POP3-proxyt használnak. A jövőben FTP-proxy és Socks bevezetését is tervezik.
A harmadik masina webkiszolgálóként dolgozik: 500 MHz-es Celeron processzorral és 128 MB RAM-mal. Webkiszolgálóként Apache-ot, levelezőkiszolgálóként pedig `qmail`-t és `vpopmail`-t használnak.
Az elképzeléseik között szerepel egy FTP-kiszolgáló (proftpd) és egy Mapguide kiszolgáló kialakítása is. Linuxot a munkaállomásokon is használnak az irodai alkalmazások futtatására és kipróbálására.

Interware Kft.

A cégnél Compaq Proliant, illetve AlphaServer DS20, ES40-es gépeket használnak Debian Linux alatt. A legnagyobb gépük egy Alphaserver ES40, 1 GB RAM-mal és 36 GB merevlemezzel. Ezen jelenleg átmenetileg körülbelül 5500 felhasználó levelezése fut, a közeljövőben azonban komolyabb igénybevételét is tervezik, ugyanis sokszorosan túlméretezték mind processzor, mind memória tekintetében. Az átlagos kiszolgáló Compaq Proliant DL380, 256 MB RAM-mal és körülbelül 36 GB merevlemezzel. A Linuxra bízott feladatok a következők: proxykiszolgáló – Squid; webkiszolgáló – Roxen Challenger; levelezőkiszolgáló – Exim + Courier + IMHO; hálózatfelügyelet – snmpd + mrtg; a felhasználók karbantartása MySQL adatbázis-kezelőn keresztül történik; végezetül DNS-szolgáltatás – Bind. A Linuxok, mint az már fentebb szóba került, megközelítőleg 5500 felhasználót szolgálnak ki. Nagyon sok saját fejlesztésű webes programot használnak, amelyek előállítását megkönnyítette a Linux alatt elérhető programozási nyelvek és eszközök bőséges tárháza. Az összes kiszolgáló fontosabb adatainak biztonsági mentése jelenleg hálózaton keresztül történik (rsync segítségével) külön kiszolgálóra, RAID5-be kötött merevlemezekre, de DLT vásárlását tervezik. Amennyiben ez sikerül, szalagra történő mentésre fognak átállni.

Országos Állategészségügyi Intézet

Célkitűzésük a levelezés, a dokumentum- és ügyiratkezelés, valamint az egységes felületen történő személyzeti nyilvántartás. Az intézet által megvalósított megoldást az alábbiakban vázoljuk fel.
Az állategészségügy PHARE-keretből körülbelül 800 felhasználós Lotus Notest vásárolt 2000-ben. Ez a következő helyszínekre bontható:

- 20 megyei állategészségügyi állomás;
- 5 állategészségügyi intézet;
- 40 állategészségügyi határállomás;
- 1 FVM állategészségügyi főosztály.

A megyei állomások, intézetek és az FVM-beli főosztály helyi hálózatokkal (átlagosan 25 felhasználó) rendelkezik, a határok pedig egygépes végpontok. Ezek mindegyike csillagszerűen kapcsolódik az Országos Állategészségügyi Intézetben (OÁI) kialakított Országos Állategészségügyi Informatikai Központhoz (OÁIK). Minden helyi hálózatra egy SuSE 7.0-t futtató IBM Netfinity 3000-es gép van telepítve 500 MHz-es Pentium III processzorral, 128 MB RAM-mal és 9 GB-os UltraWide-os SCSI merevlemezzel felszerelve ezek a helyi Lotus Notes Domino kiszolgálót futtatják. Éjszakánként mindegyikük összehangolja egymással az adatait. A központban egy kétkiszolgálós Notes Enterprise Cluster van telepítve RAID5-ös merevlemez alrendszerrel. A Lotus Notes saját telepmegoldását alkalmazzák. Mindkét kiszolgálóban három IBM UltraWideSCSI3 merevlemez található, és a rajtuk lévő lemezrészeken ReiserFS fájlrendszert használnak.
Az üzembiztonság további növelésére azonban a merevlemezek tükrözésének kipróbálását is tervezik (a helyi Notes-kiszolgálókon is).
Az eltelt idő alatt láthatóvá vált, hogy a lotusos rendszer jól működtethető, amelyben a Linuxnak jelentős szerep jutott. Régóta foglalkoznak Unixokkal (főleg SGI IRIX-szel és PC-s Linuxszal), és véleményük szerint komoly kiszolgálórendszert nem is lehet Microsoft-alapokra helyezni: mind az ár-teljesítmény viszony tekintetében, mind megbízhatóságban a Linux lényegesen jobb.

Carnation Internet Consulting Rt.

A részvénytársaságnál kilenc PC-alapú kiszolgáló alkotja az ügyfeleket kiszolgáló gépparkot. A kiszolgálókban Celeron és Pentium II processzorok találhatók, a memóriájuk 32-től egészen 256 MB-ig terjed. 8 GB-os IBM SCSI, illetve 10 GB-os WD és Quantum merevlemezek tárolják a programokat és az adatokat. Az Abit és az Asus alaplapokat részesítik előnyben, ezek közül is a következő típusokat: BH-6, BX-6, P2B-F és P3-BF. Minden kiszolgálón Linux fut, egyelőre RedHat, Slackware és Debian vegyesen. Szándékukban áll azonban a kiszolgálópark egységesítése, és mivel a Slackware Linuxot kedvelik, valószínű, hogy rá fog esni a választásuk.
A fájlkiszolgáló és a tartománykiszolgáló feladatokat két gép látja el Samba segítségével. Egy levelezőkiszolgáló Sendmailen keresztül biztosítja a cég, illetve az egyéb tartományok kapcsolatát a külvilággal. A listákhoz egy

levelezőlista-kiszolgálót használnak a mailman program segítségével. Egy proxy- és három webkiszolgáló is dolgozik a cégnél: a proxyn Squid fut, a webkiszolgálókon pedig Apache. Az Apache PHP4 és MySQL segítségével adatbázisból is képes webes felületen keresztül adatokat szolgáltatni. Egy csomagszűrő tűzfalat ugyancsak üzemeltetnek.

Az első pillanatban talán túl soknak tűnhet a kiszolgálók száma, és kevésnek az egy gép által nyújtott szolgáltatás, de azt a filozófiát vallják, miszerint egy nagy kiszolgáló helyett sokkal hasznosabb az egyes feladatok elkülönített gépeken történő megvalósítása. Így egy gép esetleges kiesése nem veszélyezteti az összes ember munkáját. A kiszolgálók változó fájljairól 2–3 naponta biztonsági másolat készül, hetente egyszer pedig teljes mentés.

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 Linux új piacokat hódít meg

Két évvel ezelőtt fejlesztési igazgatóhelyettes voltam egy nagy, távmarketinggel foglalkozó cégnél. A feladatom rendkívül nagy adatbázisok építése és karbantartása volt, ezekben tároltuk a célzott marketinghez szükséges adatokat. Jelentős kiadásaink voltak: egyrészt rengeteg pénzbe került a fejlesztői eszközök felhasználási jogainak a megvásárlása, a cégnél alkalmazott egyéb programokra pedig a gatyánk is rament. Néhány megoldásunk jól bevált, drága programjaink azonban még mindig nem látták el a vezetőséget olyan megfelelő adatokkal, amelyek alapján megfontolt üzleti döntéseket hozhattak volna. Például túl későn fértünk hozzá a kampányok sikerességét vagy sikertelenségét mutató adatokhoz, és a készpénzáramlásról is csak elképzeléseink voltak. Elhagytam a céget, amikor bezárták a fejlesztői részleget, és rosszul esett, hogy a munkám kárba veszett. Engem is zavart, hogy a cég alapvető üzleti gondjaira nem született kielégítő megoldás, és arról is volt elképzelésem, hogy a cég vezetőit mennyire elkeserítette programok felhasználási engedélyeire kifizetett tetemes összeg. Néhány hónapig a saját vállalkozásom keretében foglalkoztam adatbázis-tervezéssel, mígnem állásajánlatot kaptam egy közepes méretű cég, az Action Target ügyvezetőjétől. Az Action Target lőtereket rendez be, és ehhez gyárt felszereléseket. Először meghökkentett, amit tapasztaltam. Az ügyvezető (egy vállalkozó szellemű villamosmérnök) megismertetett a cégével, és mindenütt csak Linuxszal találkoztam. Minden egyes felhasználó – az értékesítéstől a szállítmányozáson és beszerzésen át a gyártásig, a fejlesztésig és a könyvelésig – linuxos munkaállomáson dolgozott, amelyek többsége lemezmentes volt. A munkaállomásokat egyetlen kiszolgálócsoport látta el, amelyek között webkiszolgáló, adatbázis-kiszolgáló és egy maroknyi alkalmazáskiszolgáló is szerepelt. Az alkalmazáskiszolgálók a programok mellett a felhasználók könyvtárainak is otthont adtak. Az alkalmazások többsége nyílt forrású vagy szabad program volt, így a StarOffice, Netscape vagy TGIF, de használtak néhány zárt forrású programot is, például Applixot. Az adatbázist PostgreSQL-kiszolgálóval valósították meg. A hálózati megosztásokat NFS segítségével működtették, a lemezmentes munkaállomások `bootp` protokollon keresztül csatlakoztak az alkalmazáskiszolgálókhoz.

Amit ezek után láttam, még jobban megdöbbentett. Nem elég, hogy az egész cég Linuxot használt, az ügyvezető saját ERP (Enterprise Resource Planning – vállalati erőforrás-tervező) programot írt. Elmondta, hogy több más megoldáson is gondolkodott, de mind drágának bizonyultak és költséges, zárt forrású RDBMS-megoldásokon futottak, ráadásul, ami a legfontosabb, nem támogatták megfelelő mértékben a testreszabást. Ezért úgy döntött, saját kezűleg oldja meg a gondot. A megvalósításhoz Wylibet használt, amely szintén saját fejlesztés volt. A Wylib egy Tcl/Tk-ban írt programkönyvtár-gyűjtemény. A Tcl/Tk-t könnyű használni, rendszerfüggetlen, héjként is alkalmazható, valamint a nyílt forrás közössége is elismeri és elfogadja. Az adatbázis-kezelést a PostgreSQL-re bízták.

Ez az ERP-rendszer működteti a céget. Az alkalmazottak használják, nélküle nem tudják elvégezni a munkájukat. Az ERP nyílt forrású programokon alapul, és teljes mértékben testreszabható. Ha valami nem tetszik benne, egyszerűen megváltoztatjuk. A cég vezetése úgy becsüli, hogy ez a rendszer egymillió dollárt takarított meg, ha csak a működési költségeket vesszük figyelembe, és ekkor a megtakarításba még nem számoltunk bele a programok felhasználási engedélyeinek szükségtelensége által nyert összegeket. Egy évet dolgoztam ennél a cégnél, a Wylib segítségével újraírtam az ERP-t és átszerveztem az adatbázist, majd saját céget hoztam létre WyattERP néven. A WyattERP célja kettős: egyrészről meg szeretné mutatni a cégeknek, hogyan takaríthatnak meg rengeteg pénzt a nyílt forrású megoldásokkal, másrészről azt is, hogy miként hozhatnak létre olcsó és az üzleti igényeikhez tökéletesen alkalmazkodó ERP-megoldást a Wylib segítségével.

A Wylib és a példaforráskód a ➲ http://www.wyatt-erp.com címről letölthető a Webről. A Wylib nyílt forrású, és az OPL-ben (Open Public License) foglaltak alapján terjeszthető. Ha személyesen is kapcsolatba szeretnél lépni velem, használd a weblapon megtalálható címemet.

Merrill Oveson

Egy nyomozás története

Ugye, mindenki elképzelte már a következő helyzetet: valami gond van a kiszolgálóval, amelyet üzemeltetünk – szerencsésebb, ha a karbantartását nem magunk végezzük, hiszen így nem mi hibáztunk –, és nekünk ki kell nyomoznunk a tettest.

Egyből rohanunk, hogy megoldjuk az esetet: esetleg már otthonról elkezdhetjük a munkát, de semmiképpen sem fejezhetjük be. El kell mennünk a kiszolgálóhoz, mert a gépeknél is szükség van „személyes kapcsolat"-ra, továbbá jobban látszik, hogy mennyire súlyos a gond.
A helyszínen azután belefutunk egy-két ügyes félrevezetésbe és csapdába, melyeket azonban megoldunk. Egy-két leleményes eljárás – amikor már a remény is elfogyott, természetesen csakis ekkor! –, és váratlan fordulatokkal dűlőre visszük az ügyet. A rossz ember lebukik, mi pedig besöpörhetjük az elismerést, a kollégák elismerő pillantásai nyomán pedig jó érzés tölt el minket.
Na, eddig tartott a mese.

```
[ago@mdk ago]$ telnet mail.ceg.hu 25
Trying X.Y.W.Z
Connected to X.Y.W.Z
Escape character is '^]'.
220 mail.ceg.hu ESMTP Postfix
MAIL FROM: ago@lsc.hu
250 Ok
RCPT TO: bill@microsoft.com
551 Reciepent address rejected. Relay access denied...
QUIT
```

A valóság: kétórás modemes kapcsolat otthonról, sok káromkodás és infarktusközeli élmény, egy fél tábla Boci csoki elfogyasztása szigorúan a stresszoldás végett, és befejezésképpen az áhított siker is megérkezik, amelyre azonban a frissen szerzett tapasztalatok birtokában már nem is vágyakoztunk annyira. Amennyiben választani lehetne, szívesebben szavaznék a nyugodt életre. A továbbiakban elolvashatjuk, mi is történt pontosan, milyen intézkedéseket foganatosítottam, és mi lett a következménye. A szereplők természetesen névtelenek maradnak, hogy senki személyiségi jogait ne sértsük meg.

Első felvonás

Éppen egy tanároknak szóló tanácskozáson tartózkodtam és az előadásomra készültem. Egy órával a nevezetes esemény előtt hangüzenet érkezett a telefonomra, de nem volt térerő, amikor hívni próbáltak. Az üzenetben egy barátom közölte, hogy levélszemetet (spam) kapott, és mit tudok mondani az info@gepnev címről, ugyanis tudomása szerint minket bíztak meg a karbantartásával. Kicsit elcsodálkoztam, mert az említett gépen levelezőszolgáltatások nem futottak, csupán egy átjárógép volt két alhálózattal. Ezen keresztül érhetik el az adott cég számítógépei a külső címtartományban lévő levelező- és egyéb kiszolgálókat. Felhívtam az említett cég telephelyén tartózkodó rendszergazdánkat, hogy nézze meg, nem fut-e az átjárón levelezőkiszolgáló (MTA). Természetesen nem futott, viszont célszerű volt rákérdezni, hiszen sohasem árt. Ezután már megnyugodva hívtam fel a hangüzenethagyót és közöltem, hogy valószínűleg meghamisították a levél fejlécét – ezért gondolhatta, hogy a levelet tőlünk kapta. Aznap este könnyű szívvel tértem nyugovóra.

Második felvonás

Másnap hazaérés és rövid pihenés után modemen keresztül csatlakoztam a Világhálóra. Amint beléptem, és elolvastam a beérkezett leveleket, az ellazultság legalább annyira távol került tőlem, mint a gonosz kismalactól a jóindulat. Többen is visszajeleztek, hogy levélszemetet kaptak, amely a fejlécek tanulsága szerint valóban tőlünk származott. Ezt megerősítette a levelezési forgalomról készült előző napi kimutatás, ami egyébként mindig lefut. Valaki – akkor még ismeretlen személy – 7241 kéretlen levelet továbbított a kiszolgálón keresztül! Mi is erősítette ezt meg? A visszajelzések között akadt olyan, amely a teljes levelet idézte a fejléccel együtt. Az eredetileg kiküldött levélben a `From:` szó után az a levélcím állt, amelyet a küldő a levelezőprogramba írt be. A továbbítókiszolgálókat azonosító fejlécrészlet pedig az ügyfelet kiszolgáló átjárót, valamit a tényleges levelezőkiszolgálót azonosította. A jelentés és az eredeti levél tehát egyértelművé tette: tényleg rajtunk keresztül küldték a levelet. A következőt kellett kiderítenem: valóban nyílt levéltovábbító-e a kiszolgáló vagy egyéb módon került meg a levél? Az egyéb lehetőségek közé soroltam még: a kiszolgálót feltörték és egy telepített program segítségével küldték ki a leveleket vagy a törés után nyílt levéltovábbítót csináltak belőle. Amennyiben nyílt levéltovábbítóként működik, akkor is meg kell vizsgálnom, nem történt-e valami sokkal rosszabb a háttérben. Ha megtörték, és nyílt levéltovábbító lett, valaminek futnia kellett ott. A nyomozást a lehető legegyszerűbben kezdtem: kipróbáltam, lehet-e nem engedélyezett címre levelet küldeni. Ennek legegyszerűbb módja a Telnet program használata. A segítségével csak rá kell kapcsolódni a levelezőkiszolgáló 25-ös kapujára és ellenőrizni. Aki követte a cikkeimet, annak már ismerős az a folyamat, amit az *1. lista* szemléltet.
Ez kicsit megnyugtatott, eszerint tehát rajtunk keresztül nem lehet akárkinek levelet küldeni. Ekkor számba vettem a második lehetőséget: a számítógépet megtörték. Mit tesz ilyenkor az előrelátó ember? Elővarázsolja a kiszolgáló „tiszta állapotát" tartalmazó fájlt, és az AIDE program segítségével összehasonlítja a rendszer binárisait. Kicsit morogni kezdtem, ugyanis az elutazásom előtti napon általános programfrissítést hajtottak végre. A rendszeren Debian Woody fut, amelyhez az újabb csomagokat is letöltöttem. Az új fájlt viszont mindig helyileg szeretem elkészíteni, ami azonban az utazás miatt elmaradt. Ez bizony balszerencse volt, nem engedhettem volna meg – le is szidtam magam. A következő lépésben meghívtam a `debsum` nevű programot, ami a telepített programokhoz tartozó ellenőrzőösszegeket – az úgynevezett `MD5 summ`-okat – hasonlítja össze a binárisok jelenlegi állapotával. Ha bármelyik futtatható fájlt lecserélték,

ez a program kideríti. Ámde hogyan bízhatnánk meg az esetlegesen gyanúba keveredett gépen lévő ellenőrzőösszegekben? Sehogy. Mindenesetre egyelőre tisztának tűnt minden, tehát meghívtam a `netstat` programot. Ez a program a hálózathoz kapcsolódó programok állapotáról ad tájékoztatást. Megmutatja, hogy a kiszolgáló milyen kapcsolatot kezel jelenleg. Én a *2. listán* látható módon indítottam el. Természetesen a kimeneten kívül több más érték is szerepel itt. A program ezekkel a kapcsolókkal mutatja meg, hogy melyik program vár kapcsolatot melyik kapun, és milyen IP-címen teszi azt. Itt ért az első szívroham, ugyanis egy számomra ismeretlen kaput fedeztem fel ismeretlen démonnal. Majd megnyugodtam, mert csupán az otthoni gépemen futó kísérleti program várta a kapcsolatokat. Még otthon kezdtem el kipróbálni és a gépemen maradt. Másodszor már a kiszolgálón sikerült lefuttatnom a programot, ahol mindent rendben lévőnek találtam. A biztonság kedvéért azonban tovább piszkáltam a rendszert. Átnéztem az összes parancsfájlt, amely a `cron`, az `at` és a Postfix programokhoz kapcsolódik, ugyanis ezeket a szolgáltatásokat le akartam állítani. Azt azonban mindenképpen el szerettem volna kerülni, hogy amennyiben a kiszolgálót mégis feltörték, és esetleg huncut `rm -rf /` parancsot írtak a vezérlő parancsfájlba, annak következményei legyenek. Miután mindent rendben találtam, a szolgáltatásokat leállítottam. Miért volt ez fontos? Ha a rendszert feltörték, és egy olyan binárist módosítottak, amelyet a `cron` is használ, a rendszer esetleg önműködően újra meg újra megnyílik. Ráadásul a Postfix még számos helyre el akarta küldeni a leveleket. Ezeket a leveleket a leállítás után töröltem. Jó néhány akadt, ezért miután meggyőződtem róla, hogy „rendes" levél nincs a sorban, az egész várakozási sort töröltem (szerencsére hétvége volt, ezért ez csaknem természetesnek tekinthető). Ezután a ➲ http://www.debian.org-ról leszedtem a binárisokhoz tartozó MD5-ös ellenőrző összegeket és átmásoltam őket a gépre, majd így ellenőriztem a binárisokat. Ezt az átjárón is megismételtem. Mindkét rendszer teljesen rendben volt, tehát a törést és a nyílt levéltovábbító gondokat elfelejthettem. Már csak egyetlen lehetőség maradt.

2
```
[ago@mdk ago]$ netstat -antl
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address     Foreign Address         State
tcp        0      0     0.0.0.0:25          0.0.0.0:*           LISTEN
```

Harmadik felvonás

Mivel a kiszolgáló csak egy helyről fogadott el leveleket továbbításra, nem volt nehéz dolgom, hogy behatároljam a következő keresési területet. Ez a gép az átjáró volt – az egyetlen gép, amely felől a küldés engedélyezett. Ezen az átjárón igazából nem fut semmi, csak címeket fordít és kapcsolatokat továbbít. A támadás tehát a mögötte lévő egész belső hálózatról jött. Hogyan lehet megkeresni a saját berkeinkben megbúvó támadót? Miután ismét elindítottam minden szolgáltatást a levelezőkiszolgálón, figyelmemet az átjáróra, pontosabban a naplófájlokra összpontosítottam. Mivel a levelezőkiszolgáló eléréséhez címet kell fordítanom, vagyis álcáznom (masquerading) kell, a csomagszűrővel az erre vonatkozó szabály volt beállítva. Csakhogy a szabály végén megadtam, hogy fordításnál *minden* kimenő csomagról írjon jelentést a naplófájlba. A parancs hasonlóan néz ki:

```
/sbin/ipchains -A forward -i -p tcp
➥-s 192.168.1.0/24 -d mail.ceg.hu 25
➥-j MASQ -l
```

Ez fordította át a belső hálózat címeit az átjáró IP-címére, amennyiben az ügyfelek kapcsolatba akartak kerülni a levelezőkiszolgálóval. A `-l` kapcsolta be a naplózást. Az előző napi naplófájlból kikerestem – természetesen a segédprogramok használatával –, hogy melyik ügyfél kapcsolódott kiemelkedően sokszor a levelezőkiszolgálóhoz. A segédprogram saját készítésű volt, ezért a kiosztott címtartomány minden ügyfelére összesítést készített, és hogy hány bejegyzést talál az adott IP-címhez. Majd a kiemelkedően nagy számú küldő címét felhasználva kikerestem a listából, hogy kihez tartozik a cím. Ekkor csörrent meg a telefon. Maga az elkövető volt. Honnan tudta meg, hogy nyomozok? Az egyik felháborodott partner ugyanis nemcsak nekem küldte el panaszát, hanem a levél küldőjének is – mármint arra a címre, amit az elkövető írt be, és ahova a válaszleveleket a megrendelésekkel együtt várta (ugyanis egy szolgáltatást hirdetett). Mivel csoportos választ nyomtam, ő is megkapta válaszomat – ez idegességemben fel sem tűnt nekem. Nos, próbálta magát mentegetni, hogy nem gondolta, mekkora baj lesz ebből, és különben is otthonról akarta végezni. Ennek azonban jócskán ellentmondott, hogy a saját bevallása szerint is mailbomber programot használt. A következő héten hétfőn leadtam a jelentést a cég vezetőjének és szóban is tájékoztattam az eseményekről. Ennek eredményeképpen az illetőt még aznap elbocsátották.

Miért is? A cég erőforrásait használta és mivel levélszemét jött a levelezőkiszolgálóról, néhány másik levelezőkiszolgálóról kitiltották a tőle érkező *összes* levelet. Ezenkívül a céget erkölcsi kár is érte.

Összegzés

Egyszer minden „csínytevésre" fény derül, és egy ilyen nyomozás inkább fárasztó és idegesítő, semmint jó móka. Gondolom, jövője ismeretében a vétkes is másképp cselekedett volna.

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 2001. évi LinuxJournal szerkesztői szavazásának eredménye

Sokat változott a Linux világa a tavalyi szerkesztői szavazás óta. Az idő tájt a technológiai és .com-fellendülés korát éltük, most viszont a gazdasági hanyatlásában tengődünk. Múlt évben panaszkodtunk, milyen nehéz kiválasztani a számos jelentkező közül a győztest, s felvetettük, kevesebb versenytárs esetén nyilván könnyebb lenne a döntés. Most már tudjuk, hogy ez nem feltétlenül igaz.

Noha az egyes tárgykörökben kevesebb gyártó (főleg gépet, alkatrészt termelő) indult, a kínálat változatlanul bőséges (mind a szabad, mind a kereskedelmi termékek terén), és jelentős minőségjavulást értek el az elmúlt év során, mind a termékek, mind a Linux-rendszermag fejlesztése terén.

A *Linux Journal szerkesztői díjaira* nyílt forrású és védett programok egyaránt pályázhattak, s a programok idei kiválasztottjai között mindkét típus képviselői egyformán szerepelnek. Bár jelenleg mindannyiunkat nagy elégedettséggel tölt el, hogy olyan sok nyílt forrású termék közül válogathattunk, ne feledkezzünk meg róla, hogy nem sokáig marad ez így, amennyiben nem védekezünk minden erőnkkel az SSSCA-féle törvényi szabályozás veszélyei ellen. Ez ugyanis a Digital Rights Management (amolyan digitális jogvédelem) kötelező alkalmazását vezetné be, amelynek eredményeként a szabad, sőt a nyílt forrású operációs rendszerek is törvénybe ütközők lennének.

Amennyiben valamelyik nyertesünk a végsőkig felbosszantana titeket, nézzétek meg előző számunk 20. oldalán az olvasói szavazás eredményeit is.

Kiszolgáló

Filanet Interjak 200 802.11b

Kellene egy 802.11b hálózat nagyteljesítményű antennával az olcsó WAN-kapcsolatok létesítésére? Szeretnétek egy VPN-szolgáltatást nyújtó 802.11b alapállomást, amely a felhasználók noteszgépeinek biztonságáról gondoskodik? Központilag kezelt postarendszerrel, Samba- és VPN-kiszolgálókkal kellene ellátni a vállalat minden otthon dolgozó munkatársát? A Filanet egy sor kis költségű, ventilátor nélküli, beágyazott Linuxszal ellátott hálózati eszközt gyárt, amelyeket saját ASIC alaplapra épít ARM processzormaggal és 3DES-titkosítást támogató alkatrészekkel. Segítségükkel a vállalatok és ISP-szervezetek számos gondja megoldható, s alig kerülnek többe, mint egy ostoba DSL-doboz.

Biztonsági eszköz

Nmap

Egy program elterjedésének biztos jele, ha a nevét igeként kezdik használni. Mára bevett adatbiztonsági gyakorlat, hogy minden új Linux-kiszolgáló felállításakor az Nmapet futtatják le, és rendszeresen ellenőrzik vele a hálózati változásokat. Nem véletlen, hogy az Nmap ugyanakkor terjedt el, amikor a különböző Linux-változatok megritkították az alapértelmezés szerint felkínált szolgáltatások listáját. Üdvözöljük az Nmapet mint a rendszergazdák és Linux-változatok könnyen kezelhető „biztonsági jelzőfényét"!

Webkiszolgáló

APPRO1124

Mi e rendszer kétprocesszoros Athlon MP alaplapját tettük a linuxos csúcsgépünkbe. Az APPRO azonban – a VA Linux Systems eredeti tervei alapján – egy egyszerű IU-kiszolgálóba helyezte négy működés közben cserélhető SCSI-meghajtó és egy vékony CD-ROM-meghajtó társaságában. A nagy teljesítményű ventilátoroknak és az egyedi tápegységnek köszönhetően olyan webkiszolgálót alkottak, amit annak idején nagyon szívesen láttunk volna az alkatrészpiacon, amikor a webkiszolgálókra még külön keretünk volt.

Webügyfél

Konqueror

Linuxosok, itt az ideje eldobni a Netscape 4.x böngészőt, mert mára már ósdi kacatnak számít! Mostanra mind a Mozilla, mind a Konqueror elérte a megfelelő megbízhatósági szintet, és eléggé széles körű szolgáltatásokat kínál ahhoz, hogy örökre megszabadulhassunk az avítt Motif-alapú Netscape-től. A szerkesztői díjat végül a Konquerornak ítéltük a KDE-környezetbe való tökéletes illeszkedés elismeréséül, a sebességéért, és amiért lehetővé teszi a Flash-animációk vagy -mozik egyszerű lejátszását.

3D-s modellezőeszköz

Maya 4

Ahogyan *Robin Rowe* a legutóbbi írásában már beszámolt róla (*Linuxvilág* 6–7. szám, 44. oldal), a Linux szó szerint meghódítja a filmipart: segíti a különleges képi hatások és animációk gyors előállítását. Egyetlen más iparágban sem tapasztalható ilyen tömeges áttérés Linuxra. A Maya ugyancsak kiveszi ebből saját részét, amikor ügyfelei igényének megfelelően termékét Linuxra ülteti. Ezzel *Linus* elismerését is elnyerték, aki szerint ez „minden idők legösszetettebb és leglenyűgözőbb Linuxon futó 3D-s grafikus alkalmazása".

Biztonságimentés-készítő eszköz

BRU-Pro

Már azt hittük, hogy a BRU örökre elveszett a vállalati útvesztőkben, de szerencsére *Tim Jones*, a BRU régi pártfogója nem hagyta elenyészni e jól bevált, régimódi mentési eszközt. Tim korábban a BRU eredeti gyártójának, az EST-nek volt a fejlesztési alelnöke, most pedig a Tolis Group márkanév alatt kínálja a BRU-t. Ezzel a segédeszközzel a biztonsági mentésterv és az ügyfél által használt szalagegységek egyszerűen állíthatók be. Az sem mellékes, hogy a BRU támogatja a ➲ http://www.linuxtapecert.org weboldalt, ahol a Linux alatt kipróbált és jóváhagyott szalagmeghajtók listája található.

Egyéb segédprogramok

Acronis OS Selector

Íme egy remek betöltés- és lemezterületkezelő, amelynek nagy előnye, hogy a fokozott adatvédelem érdekében a ReiserFS fájlrendszert is támogatja.

Hordozható eszköz

Compaq iPAQ

A Linux futtatására alkalmas, titkárprogrammal ellátott zsebgépek (PDA) két csoportra oszthatók: az egyik csak a legszükségesebb programok futtatására alkalmas, a másik kellően gyors és elegendő teret enged a kísérletezésnek. Az iPAQ az utóbbiak közé sorolható. Az ipari tervezés szép példája, eltekintve a majdnem szimmetrikus íróvesszőtől. Sok fejlesztőt megnyert magának, így a linuxos PDA-tulajdonosok számos alkalmazás és leírás közül válogathatnak. A tartozékokat tekintve, mint például a PCMCIA kártyabővítő-csomag és a hamarosan elérhető kamera, vagy a gyorsulásmérő további kellemes lehetőségeket tartogat számunkra. Az iPAQ tehát ígéretes felületnek tűnik a jövőbeni Linux-fejlesztések számára.

Könyvek

Linus Torvalds–Davis Diamond: Just for Fun: The Story of an Accidental Revolution

(A móka kedvéért: egy véletlen forradalom története)

Az Internet időszerűségét jelzi, hogy legbefolyásosabb személyiségei már 31 évesen megírták az első önéletrajzukat. Azért az elsőt, mert a móka nyilvánvalóan még csak most kezdődött. *Linus Torvalds* tehát véletlenül lett forradalmár, akárcsak író. Bizonyos értelemben véve éppen ez a lényeg. A könyvet nem „kiadták”, inkább csak úgy „jött” – ahogyan a cím is mondja: a móka kedvéért. Egyoldalú párbeszéd valamiről, amiről talán érdemes beszélni, s ha mégsem, a szerzőt az sem érdekli különösebben. A Linushoz hasonlók szemszögéből nézve ez a könyv is egy kezdeti állapotban levő „hack”, amit idővel tökéletesíteni és javítani lehet. Hasonló szemszögből nézve vehetjük programleírásnak vagy hibalistának is. Érdekes, hogy ennek az operációs rendszernek, amelyet leginkább a rendszermag alapítójáról ismernek, ennyire sok szerzője létezik. A könyvet elolvasva megérthetjük, hogy mi hozta össze ezeket az embereket, s hogy egy olyan hétköznapi dolog, mint egy operációs rendszer, miként lehet annyira szórakoztató.

Hálózati program

OpenSSH

A Linux Journal szerkesztőségi iroda egyik kiszolgálóján huszonegynéhány OpenSSH folyamat fut egyszerre, egy munkaállomáson pedig hat. Adatátviteli csatornákat nyitunk, nyakra-főre használjuk az scp-t, és alapjában véve az SSH-kapcsolatok bűvöletében élünk. Kényelmes, megbízható – a beállítása és a kezelése pedig kész öröm. Igazából azért esett a választásunk az OpenSSH-ra, mert ha nem lenne, mindannyian Seattle-be költözhetnénk.

Fejlesztőeszköz

KDevelop

Az egyre több felhasználót megnyerő KDevelop nemcsak hibakereső és elsőrangú böngészőeszközöket kínál, hanem a GNU-tól megszokott módon az új fejlesztések indítását is megkönnyíti. A védett IDE-alkalmazások világából érkezők örömmel tapasztalhatják, hogy a KDevelop számos népszerű kezelőfelület utánzására képes. A beágyazott változatokat gyártó REDSonic szintén a KDevelopot választotta a REDICE Linux egyesített fejlesztőkörnyezetéül.

Irodai alkalmazás

Abi Word

Nem akármilyen szövegszerkesztő: valamirevaló rendszeren három másodperc alatt elindul és 5 MB memóriát igényel egy üres dokumentumhoz. Nem tévedés: megbízható, alapszolgáltatást nyújtó szövegszerkesztő minden fölösleges sallang nélkül. Már nyomtatni is tud, sőt a Microsoft Word-dokumentumokat is képes behozni. Ha kipróbáltad, két eset lehetséges: 1. megtetszik; 2. nem vesztettél sok időt a letöltésével.

Asztali környezet

KDE2

Az új KDE asztali környezetnek ugyan még fejlődnie kell, ami az erőforrás-felhasználást és az üzembiztonságot illeti, de minden megjelenő változat azt ígéri, hogy a legjobb úton halad a tökéletesség felé. Jobb lett a felépítése és komoly fejlesztésen ment keresztül. Az egyik legkellemesebb módosítás a KDE böngészőjének, a Konquerornak fájlkezelőként történő beépítése. A Google bevonásával keresést indít a címsorba beírt szavakra. A KDevelop szintén teljesen be van ágyazva, lásd a Fejlesztőeszköz tárgykört.

Valós idejű eszköz

Preemptible Kernel Patch,
Nigel Gamble és társai
– Montavista Software

Ez a mindössze ezersoros javítófájl a meglévő rendszermag SMP darabolási stratégiáját kihasználva figyelemreméltó eredményt ér el. Nemcsak a beágyazott rendszerek megszállottjai értékelhetik ezt, hanem bárki, aki egy terjedelmes tar-fájl kicsomagolása közben szeretne a számítógépén zenét hallgatni.

Kapcsolódó címek

APPRO 1124 ➲ appro.com/1124.html
BRU-Pro ➲ www.estinc.com/bruproinfo2.php
Filanet InterJak 200 802.11b ➲ www.filanet.com/index.php3?side=home&home=products_80211b
Konqueror www.konqueror.org
Linux Professional Institute ➲ www.lpi.org
Tribes 2 ➲ www.lokigames.com/products/tribes2
Velcro ➲ www.velcro.com
SuSE Linux 7.3 ➲ www.suse.com/us/products/suse_linux/i386/index.html

Honlapok

LinuxDevices

Miután az LLC nemrég megvette a DeviceForge-ot, a LinuxDevices visszatért alapítója, *Rick Lehrbaum* kezébe. E weboldal rendkívül sokrétű: hírekkel, útmutatókkal, termékismertetőkkel és összehasonlításokkal, valamint vitafórumokkal szolgál. Bár elsősorban beágyazott Linuxszal foglalkozik, az átlagos Linux-felhasználóknak is rengeteg ötletet kínál.

Adatbázis

Oracle

A Linuxon 1999-ben megjelent Oracle mára erős versenytárssá vált. Tavaly a PostgreSQL kapta a szerkesztők díját, s noha változatlanul komoly vetélytárs és nagy nyilvánosságot kapott az idei év folyamán, az Oracle lehengerlő teljesítményét egyszerűen nem lehet figyelmen kívül hagyni.

A lejáratás és a felhasználói szerződések

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.

Néhány, kereskedelmi programokat forgalmazó cég állítása szerint a szabad programban és a GPL-ben van valami, ami idegen az amerikai értékektől. Eme „aggódók kórusában" a Microsoft hangja a leghangosabbak egyike. A cég a honlapján így fogalmaz:
„A GNU General Public License (GPL) … jelentős fenyegetés azon szellemi tulajdonra alapozó cégek számára, amelyek a GPL hatálya alá tartozó programok köré próbálják meg üzletüket felépíteni. Még az olyan vállalkozások is, amelyek azt hiszik, hogy »pusztán felhasználói« a GPL-programoknak, veszélyeztetettek, mivel a GPL-es kódot kombinálják különállónak vélt alkalmazásokkal. Ez a szerződési modell előre meghatározza, hogy egy cég mely szellemi termékeit fogja megosztani a közösséggel és milyen feltételek mellett."
➲ http://www.microsoft.com/business/licensing/ssfaq.asp
Már másutt is bemutattam (lásd
➲ http://www.rosenlaw.com/html/GPL.PDF), hogy ez az állítólagos fenyegetés légből kapott. Ám az ellentmondás még mélyebb: a Microsoft saját közösségi (*shared-source*) szerződése sokkal veszélyesebb a programfejlesztők közössége számára. Olyan trójai típusú szerződés, amely – ha nem vigyázunk – tönkreteheti nyílt forrású vagy kereskedelmi fejlesztéseinket.
A Microsoft közösségi szerződésének legegyszerűbb változata az, amelyet a Windows CE 3.0 forráskódjával terjesztenek. Ennek a szerződésnek a második bekezdésében áll: „Ez a program felhasználható bármilyen nem üzletszerű tevékenység céljából, beleértve az ennek alapján készült változatok terjesztését is." Ezután a szerződés egyértelműsíti, hogy üzletmenetünk vezetése „nem minősül nem üzletszerű tevékenységnek".
Az üzleti felhasználóknak – úgy gondolom, a legtöbb nyílt forrású program fejlesztője ebbe a kategóriába tartozik – a szerződést a rájuk vonatkozó felhasználási korlátozások meghatározásához tovább kell vizsgálniuk.
A rossz hír a szerződés harmadik bekezdésében olvasható: „Üzletszerű tevékenység során e program kizárólag a Windows CE-felülethez készített további programok és eszközök fejlesztéséhez és ellenőrzéséhez használható. Ez a program sem forrás-, sem bináris kód formájában nem terjeszthető üzleti céllal".
Figyeljük meg, hogy a Microsoft nem engedélyezi a kód lemásolását vagy beillesztését a saját programunkba. Üzleti céllal a kódot csak referenciaként használhatjuk. Természetesen tilos a Microsoft-kód bármely részét lemásolnunk vagy a saját programunk kialakításához felhasználnunk. Mi történik azonban abban az esetben, ha később önállóan, anélkül, hogy tudatosan felidéznénk, amit a Microsoft kódjában láttunk, olyasvalamit hozunk létre, ami lényegi hasonlóságot mutat vele? Felelősségre vonhatnak-e a szerzői jog megsértéséért?
Itt jön a képbe a szerződés trójai faló jellege, ugyanis a bíróságok korábban már egyértelművé tették: „annak a bizonyítása, hogy a mi alkotásunk és egy másik között lényegi hasonlóság áll fenn, valamint az, hogy a másik alkotás hozzáférhető volt, elegendő lehet a szerzői jog megsértésének bizonyítására, még akkor is, ha a másolás nem tudatosodott bennünk".
Vajon mennyire könnyű elfelejteni valami fontosat, amit az ember olvas? Egy szerzői jogi per a 70-es évekből segít megvilágítani a kérdést. *George Harrison* cégét beperelték a szerzői jogok megsértéséért. Egy zenei kiadó azt állította, hogy Harrison „My Sweet Lord" című népszerű dala egy korábbi sláger, a „He's So Fine" másolata. A szerzői jog megsértésének bizonyításához a „He's So Fine" kiadójának nem csak azt kellett bizonyítania, hogy a két dal „meglepően hasonló", hanem azt is, hogy Harrison az eredeti dalt másolta a „My Sweet Lord" írásakor.
Harrison nem tagadta, hogy ismerte az eredeti számot, azonban azt állította, hogy a „My Sweet Lord" írásakor nem volt tudatában, hogy a „He's So Fine" dallamát használja. A bíróság döntése szerint:
„Amikor zenei anyagokat keresett gondolatai kifejezéséhez ... [Harrison] elméjében előkerült egy bizonyos kombináció, amellyel elégedett volt..." Vajon Harrison szándékosan másolta le a „He's So Fine" zenéjét? Nem gondolom, hogy szándékosan tette volna. Mindazonáltal világos, hogy a „My Sweet Lord" ugyanaz a szám, mint a „He's So Fine", csak más szöveggel, és a „He's So Fine" Harrison számára hozzáférhető volt. Ez a törvény értelmében kimeríti a szerzői jog megsértését, és ezen az sem változtat, hogy nem tudatosan történt.
[Bright Tunes Music Corp. kontra Harrisongs Music, Ltd., 420 F.Supp. 177 (S.D.N.Y. 1976.)]
Ezekután Harrison cégét hozzávetőleg 1,6 millió dollár kártérítés megfizetésére kötelezték.
Bárki, aki jártas a számítógép-programozás művészetében, tudja, hogy éppen úgy, mint a zenében, bizonyos gondolatok kifejezésére meglehetősen szokványossá vált formák állnak rendelkezésre. Amennyiben egyszer már látta a Microsoft kódját, vajon képes-e egy szakavatott programozó egyszerűen kitörölni az emlékezetéből? Mégha meg is próbálja tudatosan elfelejteni, és nem törekszik a másolására, kifejeződhetnek-e a nem tudatos emlékei az általa létrehozott programkódban olyan mértékben, hogy egy bíróság a hasonlóságot elegendőnek találja a szerzői jog megsértésének megállapításához?
Arra intenék minden nyílt forrású kódot fejlesztő programozót, hogy kerülje ezt a kockázatot. Aki nem tartozik azok közé a különleges emberek közé, akik képesek ellenőrizni a tudatalattijukat, az inkább meg se nézze a Microsoft közösségi forráskód-szerződés hatálya alá eső kódot.

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).

Új termékek

LinuxCAD 3.0

Piacra dobták a LinuxCAD 3.0-s változatát, amely már támogatja a háromdimenziós rajzok készítését. Az összes gyakran használt 2D és 3D Acad-rajzolóparancs a LinuxCAD-ben ugyanúgy működik, mint az Acadben. A LinuxCAD az X környezetbe épül be, ami lehetővé teszi, hogy a felhasználók ugyanazt a fájlt több ablakban és több képernyőn szerkesszék, a rajzok részeit másolhassák, és egyszerre tíznél is több rajzzal dolgozhassanak ugyanazon a számítógépen. A LinuxCAD támogatja a .DXF, .DWG, .DXS, .SLD és .SHX rajzformátumokat, és Intel-alapú, Solaris- és LinuxPPC-rendszereken fut.
Adatok: Software Forge, Inc., telefon: 913-663-1724, e-mail: sales@softwareforge.com,
➲ http://www.linuxcad.com

VXA AutoRak

Az Ecrix Corporation VXA Autorak nevű készüléke állványba szerelhető szalagtároló és -betöltő, amely akár 660 GB tömörített adatot is képes tárolni, és adatátviteli sebessége elérheti a 21,6 GB/óra értéket. Az AutoRak legfeljebb tíz adatkazettát tud használni, alakja szabványos 2U formájú, ezért könnyen beszerelhető a 19 hüvelykes állványokba. Az adatok mentése és helyreállítása az AutoRak vezérlőpultján keresztül állítható be és követhető figyelemmel. A ki- és bemeneti kapu biztonsági megfontolásokból lezárható. Intelligens vonalkódolvasók is beszerezhetők hozzá.
Adatok: Ecrix Corporation, 5525 Central Avenue, Boulder, Colorado 80301, telefon: 303-402-9262, e-mail: info@ecrix.com,
➲ http://www.ecrix.com

Plesk Control Panel

A Rackspace Managed Hosting bejelentette a Plesk Server Administrator (PSA) 2.0-t. Több felületen futó webalapú program, amellyel sokféle rendszerfelügyeleti feladat is ellátható, a Rackspace Linux- és Unix-kiszolgálóhoz már egyaránt elérhető. A Plesk egérrel vezérelhető felületén a felhasználók postafiókokat hozhatnak létre és elvégezhetik a tartományok karbantartását. A PSA a szolgáltató vállalatok számára lehetővé teszi, hogy a kiszolgáló-karbantartási feladatokat ügyfeleikkel megosszák – kihasználva a webes felügyeleti felület három szintjét: Admin, Reseller Client és Domain Owner.
Adatok: Rackspace Managed Hosting, Inc., 112 East Pecan, Suite 600, San Antonio, Texas 78205, telefon: 1-800-961-288,
➲ http://www.rackspace.com

Niveus 205

A Niveus 205 Pinguin Computing Intel-alapú munkaállomása, amelyet

háromdimenziós grafikai és alkalmazásfejlesztési feladatokra terveztek. A Niveus belsejében két legfeljebb 1,26 GHz órajelű Pentium III processzort, 133 MHz sebességű alaplapot, ATA-100 merevlemezt, legfeljebb 1,5 GB PC133 RAM-ot, öt PCI-csatlakozót, egy 4× AGP-csatlakozót, három 5,25" meghajtóhelyet, 52-szeres CD-ROM-ot és 3,5" lemezmeghajtót találunk. A Niveus munkaállomás előre telepített RedHat operációs rendszerrel kerül forgalomba. Sokféle kiegészítő kapható hozzá, például Klipsh ProMedia hangszórók, LCD képernyők és csúcsminőségű grafikus kártyák, beleértve a GeForce 3-at is.
Adatok: Penguin Computing, Inc. 965 Mission Street, Suite 600, San Francisco, California 94103, telefon: 1-888-736-4846, e-mail: info@penguincomputing.com,
➲ http://www.penguincomputing.com

e/pop Server

A WiredRed Software Corp. kis- és nagyvállalatok számára kiadta az azonnali üzenetváltást (instant messaging = IM) és a valós idejű kapcsolattartást lehetővé tevő programját, az e/pop Servert. Az e/pop Linux Server az e/pop Standard Server Edition részeként a központilag felügyelhető, méretezhető és biztonságos üzleti kapcsolattartást, az üzenetek tárolását és a hálózati, valamint az internetes útválasztást teszi lehetővé. A távkommunikáció behívásos módszerrel, VPN-en vagy internetkapcsolaton keresztül jön létre. A biztonságról a beépített 512 bites RSA-titkosítás gondoskodik, az AES-, DES-, Triple DES- vagy az RC4-módszer használható. Az e/pop segítségével szövegalapú csevegő- és VoIP-konferencia is tartható, valamint az alkalmazások is megoszthatók.
Adatok: WiredRed Software Corporation, 4669 Murphy Canyon Road, Suite 108, San Diego, California 92123, telefon: 858-715-0970,
➲ http://www.wiredred.com

IEMS6

Az International Messaging Associates (IMA) bejelentette az Internet Exchange Messaging Server (IEMS) 6.0-s változatának megjelenését. Az IEMS6 magja az üzenetkezelő alkalmazás-keretrendszer, amely olyan alkalmazásfejlesztő környezet, amelyben a rendszergazdák egyszerű üzenetkezelésre felkészített alkalmazásokat készíthetnek, és az elektronikus levelezőrendszerrel, GSM-mel, SMS-sel és az Internettel köthetik össze őket. A felhasználók az alkalmazásokhoz vállalati és kishálózati-környezetben is hozzáférhetnek – otthoni hálózaton, SMS-ezésre képes mobiltelefonon, illetve tetszőleges webböngészésre képes készüléken keresztül. Az IEMS6 olyan naptár- és határidőnapló-lehetőségeket tartalmaz, amelyek támogatják a Linux-, Solaris-, HP-UX- és az Outlook-felületet. Az IEMS6-nak része még az SMTP-hez való SSL továbbfejlesztett támogatása és az SMTP-hitelesítés támogatása, a csatolt mellékleteket eltávolító szűrő, valamint az üzenet titkosított tárolását biztosító modul.
Adatok: International Messaging Associates, Ltd., 27/F China Resources Building, 26 Harbour Road, Wan Chai, Hong Kong, e-mail: sales@ima.com,
➲ http://www.ima.com

A hónap szakmai tanácsai

Létezik PAM a Slackwaren?

Slackware-kiszolgálómon egyre több alkalmazást telepítek a felhasználóim számára. Azt tapasztaltam, hogy számos alkalmazás használatához hitelesítés szükséges, azonban akad néhány, amelyik az adatokat nem a *passwd* fájlból veszi, ennek következtében a felhasználóknak több helyen is meg kell változtatniuk a jelszavukat.

Úgy tűnik, hogy a világ a PAM és az LDAP használata felé halad, ezért ha átállhatnék a támogatásukra, a felhasználók egy helyről (például egy webalapú jelszóváltoztató alkalmazáson keresztül) meg tudnák változtatni az összes szolgáltatáshoz tartozó jelszavukat, beleértve a Samba, a levelezés, a pppd és phpgroupware szolgáltatásokat. A Slackware sajnos nem támogatja a PAM-ot, és nem találtam olyan leírást, amely a PAM telepítését tárgyalná.

Brian Johnson, bjohnson@jecinc.on.ca

A PAM-ot olyan terjesztésekre is telepíteni lehet, amelyek nem támogatják, ez azonban más egyebek mellett azt is magával vonja, hogy minden hitelesítést megkövetelő alkalmazást le kell cserélned a PAM-ot használó változatra (ha a terjesztésedben ezek nem érhetők el, meg kell szerezned a forráskódot, meg kell keresned a PAM-foltokat – amennyiben nem részei a programnak –, végül a rendszered beállításainak megfelelően mindent újra kell fordítanod.)

Ez rengeteg munkával jár, és amennyiben nem kifejezetten keresed a különleges kihívásokat, javaslom, térj át valamelyik korszerűbb terjesztésre, például a Debianra vagy a RedHat Linuxra (mindkettő alapból támogatja a PAM-ot). A fenti két változatot csupán példaként említettem, a PAM-ot számos más terjesztés is támogatja.

Marc Merlin, marc_bts@valinux.com

Nincs elég hely a telepítéshez

A Slackware állandóan azt írja ki, hogy nincs elegendő helyem a telepítés folytatásához. Ez hihetetlen, hiszen 10 GB helyet foglaltam le e célra. Merevlemezem felosztása a következőképpen fest:

5 GB – WinNT 4.0
512 MB – /root
512 MB – csereterület
4 GB – /usr
4 GB – /home

Cheppy, banggae@fisika.ui.ac.id

A lemezrészek formázása és befűzési pontjaik megadása után válts át a második virtuális konzolra (Alt+F2), és a `df` vagy a `mount` használatával ellenőrizd, hogy a lemezrészek be vannak-e fűzve. Amennyiben nem, az egész Slackware az 512 megabájtos saját lemezrészre települ. Ez a méret túl kevésnek bizonyulhat, ha az X-et vagy más nagyméretű alkalmazást telepítesz.

Chad Robinson, crobinson@rfgonline.com

Lefogadom, hogy rosszul címkézted fel a lemezrészeket, és a / helyett `/root` címkét használtál. A lemezrészeket így próbáld címkézni:

5 GB – WinNT 4.0
512 MB – /
512 MB – csereterület
4 GB – /usr
4 GB – /home

Ez elég helyet biztosít a telepítéshez.

Paul Christensen, pchristensen@penguincomputing.com

A frissítés óta hetente fagy a gépem

Miután az egyik RedHat-rendszerünket 7.0-sról 7.1-es változatra frissítettem, a gép körülbelül hetente egyszer lefagy. A fagyás mindig hajnali négy óra után nem sokkal következik be (a cron.daily végrehajtása után).

A rendszermag kimenete:

```
unable to handle kernel NULL pointer
➥dereference at virtual address
➥00000000
```

A rendszermagot a 2.4.3-12-es változatra frissítettem, de a helyzet nem sokat javult.

Atsuko Crum, acrum@hood.edu

Egy másik terjesztéssel – de szintén a 2.4.x rendszermagváltozattal – nekem is hasonló gondom akadt. Végül is az alaplapi BIOS frissítése a gondok nagy részét megoldotta – bár néha még előfordul fagyás, mindazonáltal sokkal ritkábban.

David Brown, david@caldera.com

A számítógép alkatrészei bármikor meghibásodhatnak, ha a fagyások azonban a frissítés után kezdődtek, a rendszert érdemes azzal a rendszermaggal kipróbálni, amelyet a 7.0-s változattal használtál. Amennyiben nem riadsz vissza a rendszermag újrafordításától, tégy egy próbát a legfrissebb 2.4 rendszermaggal. A 2.4 sorozat első változataiban számos hibát kijavítottak. Amennyiben a számítógép meghibásodásának lehetőségét ki szeretnéd zárni, kipróbálhatod a Cerberust, amely erős terhelés alatt ellenőrzi a gép alkatrészeit. A Cerberus a SourceForge-ról a

➲ http://sorceforge.net/projects/va-ctcs címről tölthető le.

Marc Merlin, marc_bts@valinux.com

SCSI-utánzás csak egy meghajtóra

Egy HP IDE CD-íróval rendelkezem, ezért SCSI-utánzást kell használnom, hogy működjön a cdrecord programmal. A 2.2.18 rendszermag alatt meg tudtam mondani az `ide-scsi` modulnak, hogy csak az íróval foglalkozzon, és hagyja békén az ATAPI CD-ROM-meghajtómat. Ezt a *lilo.conf*-ba írt `append` sorral értem el:

```
append="hdc=ide-scsi"
```

Remekül működött, mivel a */dev/hdc* az író és a */dev/hdd* az ATAPI CD-ROM. Ez a 2.4-es rendszermaggal sajnos nem működik többé. Az `ide-scsi` modul mindkét eszközt megragadja, ennek következtében a */dev/hdd* elérhetetlenné válik, és a cdparanoia nem tud dolgozni vele, engem meg arra kényszerít, hogy a */dev/scd1*

használatával fűzzem be. Hogyan érhetném el, hogy az `ide-scsi` modul a 2.4 rendszermag alatt is csak a */dev/hdc*-t használja?
Michael Soulier, michael.soulier@home.com

Ha jól értem, azt szeretnéd, hogy a `hdc` SCSI-utánzást használjon, míg a `hdd` továbbra is IDE-eszköz maradjon. Általában az IDE CD-támogatása tiltott, a SCSI-utánzás pedig engedélyezett, ezért látszik mindkettő SCSI-eszközként. Olvasd el a
➲ http://www.wizball.co.uk/linux/cd_rewriter.php és a
➲ http://wwwteknospy.com/pages/howtos/cdburn.php
oldalakon található leírásokat.
Paul Christensen, pchristensen@penguincomputing.com

Merevlemez beépítése

Beépítettem egy második merevlemezt a gépembe, fel is osztottam lemezrészekre, de nem tudok rajta fájlrendszert létrehozni.
Kevin Williams, williams_kevin@btconnect.com

Először is győződj meg róla, hogy melyik eszközről van szó. Az alábbi egyszerű lista az IDE-felületű egységeket foglalja össze, talán segít az eligazodásban:

/dev/hda1	az elsődleges csatolón lévő mester (primary master) első lemezrésze
/dev/hda2	az elsődleges mester második lemezrésze
/dev/hdb1	az elsődleges szolga (slave) első lemezrésze
/dev/hdb2	az elsődleges szolga második lemezrésze
/dev/hdc1	a másodlagos csatolón lévő mester (secondary master) első lemezrésze
/dev/hdc2	a másodlagos mester második lemezrésze
/dev/hdd1	a másodlagos szolga első lemezrésze
/dev/hdd2	a másodlagos szolga második lemezrésze

Most válaszd ki a használni kívánt fájlrendszert. A SuSE-terjesztésben a ReiserFS található – én ezt javaslom, mivel használata esetén egy hibás rendszerleállás utáni újraindításkor nem kell kivárnunk a hosszadalmas ellenőrzést. Ezt követően formázd meg a lemezt. A ReiserFS lemezrész létrehozásához szükséges parancs az `mkreiserfs <eszk znØv>`, ahol az `<eszk znØv>` a lemezrész neve. Ha ragaszkodsz az ext2-höz, ugyanezt a parancsot használd, de az `mkreiserfs`-t cseréld le `mke2fs`-re. Létrejött tehát egy használható lemezrész, melyet már csak be kell fűznöd. Válassz vagy hozz létre egy befűzési pontot. Ebben a példában én a */mnt/storage* könyvtárat használom. Hozd létre ezt az `mkdir /mnt/storage` paranccsal. Amint láthatod, a befűzési pont igazából egy könyvtár. Most fűzd be a meghajtót:

```
mount <eszk znØv> /mnt/storage -t
➥<fÆjlrendszer_t pusa>
```

Itt az `<eszk znØv>` a lemezrész által használt eszköznév, a `<fÆjlrendszer_t pusa>` pedig `reiserfs` vagy `ext2`.

Most már egy második használható linuxos merevlemezed is van. Csak egy lépés maradt hátra. Feltételezem, hogy az új lemezt minden rendszerindítás után használni akarod, tehát az új lemezrészt a */etc/fstab* táblázatba is be kell vezetni. Írj ehhez a fájlhoz egy sort:

```
<eszk znØv>              <befßzØsi pont>
➥<fÆjlrendszer_t pusa> defaults 0 0
```

Az `<eszk znØv>` itt is a lemezrész eszközneve, a többi szintén egyértelmű. Ha a ReiserFS-t használod, a sor végén `0 0`, ha az ext2-t, akkor pedig `1 2` szerepeljen.
Ben Ford, ben@kalifornia.com

128 bites pontosság GCC-vel

Az `xlC` parancsot használom C++ programok lefordítására Unix-felületen. Amennyiben 64 bitről 128 bitre kell növelnem a matematikai számítások pontosságát, a következő parancsot alkalmazom:

```
xlC128 -qldbl128 <fÆjlnØv> [-lm]
```

A `-lm` kapcsolóval fűzöm be a matematikai programkönyvtárakat (amennyiben hiányoznak). Ezeket a számításokat ugyanolyan pontossággal Linux alatt is el szeretném végezni. Mivel próbálkozzam? Ha a matematikai programkönyvtárakat is fel kell használnom, kérem, adjátok meg, honnan szerezhetem meg őket.
Pramod, l_pramod@hotmail.com

Szerezd be a GMP-t (Gnu Math Precision)! Ez a szabad forrású programkönyvtár tetszőleges pontosságú aritmetikát valósít meg előjeles egészekkel, racionális és lebegőpontos számokkal. A ➲ http://www.swox.com/gpm címről tölthető le.

Mac-lemezrészek befűzése

Van egy 2001-es iMacDV számítógépem (400 MHz, 128 MB RAM), melyen Mac OS 9.1 és a Yellow Dog 2.0 fut. A Mac OS 9.1 lemezrészen lévő fájlokat sajnos nem tudom elérni. Azt hiszem, hogy a Mac OS lemezrész a `hda1`, de lehet, hogy tévedek.
Bill MacKay, w.mackay1@ntlworld.com

Valóban tévedsz, nem a `hda1`-ről van szó. A Mac-gépek a lemezrészeket valamivel bonyolultabban tartják nyilván. A lemezrészeket a `cat /proc/partitions` paranccsal nézheted meg. Remélem, sikerült könnyebbé tenni a helyzetet.

Hogyan indítsam el önműködően a webkiszolgálót?

A HTTPD a gép indításakor nem indul el magától, így nekem kell megtennem a */etc/rc.d/init.d/* könyvtárból. Melyik beállításfájlt változtassam meg, hogy ez önműködően is megtörténjen?
W. Huang, whuang53@excite.com

Add ki a `chkconfig --level 5 httpd on` parancsot. Ez azt feltételezi, hogy az 5. futásszintet használod (GUI a RedHatben). Ha más futásszintet akarsz, csak írd át a számot.
Ben Ford, ben@kalifornia.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.

Vírusellenőrzés a Squid proxykiszolgálón

A rendszergazdák természetes igénye, hogy minél több vírusellenőrzési pontot iktassanak be a számítógépes hálózatokba. Az egyik ilyen pont a HTTP-proxykiszolgáló, ugyanis ezen haladnak keresztül azok az Internetről letöltött fájlok, amelyeket a felhasználók böngészői szednek le.

Amennyiben e ponton sikerül kiszűrni a vírusos állományokat, el sem jutnak a felhasználók gépeihez.
A Squid általánosan használt proxykiszolgáló, vele működik együtt a `viralator` program, amelynek segítségével elvégezhetjük a víruskeresést.
A `viralator` Perlben írt CGI-héjprogram, amely képes a bemeneti értékként megadott fájlokat letölteni a kiszolgálóra, és a letöltött állományokon egy külső vírusirtó programot futtat. Eközben a felhasználóval az ügyfélgépen futó internetböngészőn keresztül tartja a kapcsolatot, azaz tájékoztat a letöltés menetéről, és arról, hogy vírusos-e a fájl. Nézzük meg működés közben!
A böngésző ablakában a kívánt hivatkozásra kattintva kezdjük meg a letöltést. A böngésző a letöltési kérést elküldi a távoli kiszolgálóhoz – ezt a kérést kapja el a viralator program (*1. kép*). A böngészőnek visszaküld egy „downloading…” tartalmú oldalt, ezután nyit egy kék hátterű ablakot, ahol a letöltés menetét láthatjuk – az ablak alján a vírusellenőrzés eredményével. A kék ablakban egy ***Stop*** gomb segítségével a folyamatot megállíthatjuk. Ha a teljes állomány a kiszolgálóra került és nem volt vírusos, előugrik a böngésző letöltési ablaka és menthetjük a fájlt. Miután az állomány az ügyfélgépre is megérkezett, térjünk vissza a viralator kék ablakához, és nyomjuk meg a ***Close window*** gombot. Ezután egy ablak tájékoztat arról, hogy a program a kiszolgálóról letörli az állományt, majd el is tűnik.

A viralator működéséhez szükséges programok

Vírusirtó program: a viralatorba nincs vírusirtó beépítve, külső programot indít el. A múlt havi számban ismertettem a Sophos `sweep` telepítését, a viralator képes vele együttműködni.

`wget`: a böngésző által kért állományt a `wget` programmal tölti le. Mindenképpen telepítsük, nagyon hasznos program.

HTTP-proxykiszolgáló: esetünkben a Squid, azonban most sem a telepítésére, sem a beállítására nem térek ki (lásd még ***Linuxvilág*** 2001. február–márciusi számának 74. oldalát). Az ügyfelek böngészőinek az Internetet a Squiden keresztül kell elérniük.

Átirányító program: ez a Squidhez intézett letöltési kéréseket egy külső programnak küldi át. A viralatorhoz ajánlott átirányító a Squirm.

Webkiszolgáló: a `wget`-tel letöltött fájl a webkiszolgálón keresztül jut el az ügyfélgépre. Apache-kiszolgáló működését feltételezem, a telepítést pedig Debian Potato rendszerre írom le.

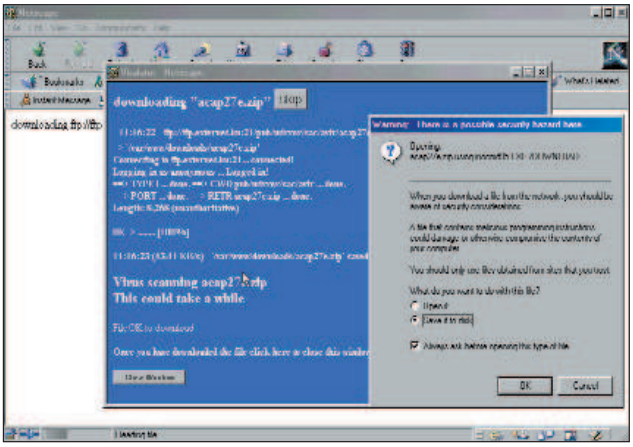
A Squirm telepítése

A Squirm honlapján a ➲ http://squirm.foote.com.au címen részletes telepítési leírást találunk, a forrást pedig a ➲ http://squirm.foote.com.au/squirm-1.0betaB.tar.gz címről tölthetjük le. Amennyiben ezt megtettük, csomagoljuk ki a */usr/src*-be, majd lépjünk be a ***squirm-1.0betaB*** könyvtárba. Adjuk ki a

```
cd regex
./configure
make clean
make
```

parancsot. A ***regex*** könyvtárban létrejövő két fájlt az eggyel feljebb lévő könyvtárba kell másolnunk:

```
cp -p regex.o regex.h ..
```



1. kép Letöltés Netscape-pel

Meg kell tudnunk, hogy a Squid milyen felhasználóként fut, amit a

```
grep cache_effective_user /etc/squid.conf
```

utasítással tehetünk meg. Debianon ez a `proxy` felhasználó, míg a Squirm a `squid` felhasználóra van beállítva. Ennek szellemében kell a Squirm Makefile-ját az `install` résznél módosítani:

```
install -m 755 -o root -g root
➥-d /usr/local/squirm \
/usr/local/squirm/bin
install -m 770 -o root -g proxy
➥-d /etc/squirm
install -m 750 -o proxy -g proxy
➥-d /var/log/squirm
install -m 660 -o root -g proxy
➥squirm.local.dist squirm.patterns.dist \
```

```
/etc/squirm
install -m 755 -o root -g root --strip squirm
➥/usr/local/squirm/bin
```

A program a beállítóállományokat eredetileg a */usr/local/squirm/etc*, a naplófájlokat pedig a */usr/local/squirm/log* könyvtárba tette. Mivel erre nem találtam elégséges indokot

2. kép A Viralator weboldala

és zavaró is lehet, ezeket is átírtam, így a fenti mintában már */etc/squirm* és */var/log/squirm* szerepel.
Ezután a ***paths.h*** fájlt is módosítani kell ott, ahol az eredeti elérési útvonalak voltak:

```
#define LOG_MATCH "/var/log/squirm/squirm.match"
#define LOG_FAIL "/var/log/squirm/squirm.fail"
#define LOG_ERROR "/var/log/squirm/squirm.error"
#define LOG_WHERE "/var/log/squirm/squirm.where"
#define LOG_DEBUG "/var/log/squirm/squirm.debug"
#define LOG_INFO "/var/log/squirm/squirm.info"

/****** Configuration file locations *******/
#define LOCAL_ADDRESSES "/etc/squirm/squirm.local"
#define REDIRECT_PATTERNS "/etc/squirm/squirm.patterns"
```

Adjuk ki a

```
make
make install
```

parancsokat és próbáljuk ki, hogy az átirányító fut-e rendszergazdaként a rendszerünkön:

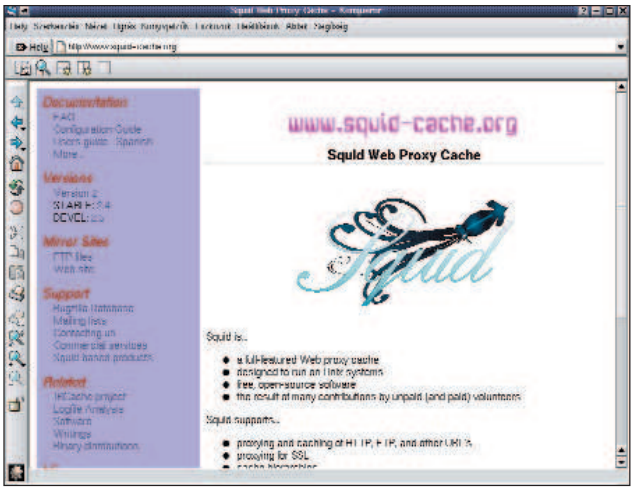
```
/usr/local/squirm/bin/squirm

Squirm running as UID 0: writing logs to stderr
Wed Nov 21 10:55:01 2001:unable to open local
➥addresses file [/etc/squirm/squirm.local]
Wed Nov 21 10:55:01 2001:unable to open
➥redirect patterns file
Wed Nov 21 10:55:01 2001:Invalid condition
➥- continuing in DODO mode
Wed Nov 21 10:55:01 2001:Squirm (PID 24924)
➥started
```

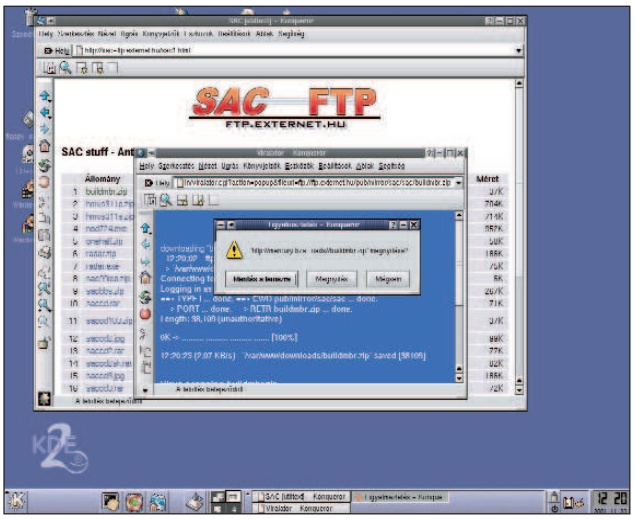
Mivel a Squirmnek a */etc/squirm* könyvtárban még nem készítettünk beállítóállományokat és csak dodo-módban indult el, CTRL+C-vel lépjünk ki.
A */etc/squid.conf* fájlban keressük meg a `redirect_program` részt és módosítsuk:

```
#redirect_program none

redirect_program /usr/local/squirm/bin/squirm
redirect_children 10
```



3. kép Hasznos programok tárháza



4. kép Letöltés Konquerorral

A Squidet indítsuk újra. A */var/log/squid/cache.log* állományban megjelenő

```
helperOpenServers: Starting 10 'squirm'
➥processes
```

bejegyzés tájékoztat arról, hogy a Squirm elindult.

A Squirm beállítása

A Squirm telepítésekor a */etc/squirm* könyvtárba két mintafájlt hoz létre: a ***squirm.local.dist***-t és a ***squirm.patterns.dist***-t. Másoljuk át őket ***squirm.local*** és ***squirm.patterns*** néven ugyanebbe a könyvtárba.

A ***squirm.local***-ba be kell azokat az ügyfeleket írni, akiknek a kéréseit át akarjuk irányítani. Az átirányítás szabályait soronként egyesével a ***squirm.patterns***-be kell írnunk. Próbáljuk ki a következőt:

```
regexi ^http://www\.playboy\.com/.
➥* http://www.disney.com
```

Amennyiben a kis- és nagybetűket nem akarjuk megkülönböztetni, a szabályt a `regexi`-vel vezessük be, egyébként a `regex`-et alkalmazzuk. A szabályok részeit szóköz választja el egymástól. Az első rész egy szabályos kifejezés, amelyet a Squirm a kért URL-re illeszt. Ha a szabály illeszkedett, a Squid a második részben leírt című állományt küldi el az ügyfél böngészőjéhez. Vigyázzunk, a szabályos kifejezésben a pontokat fontos fordított perjellel (backslash) védeni, ezzel szemben a szabály második részében a fordított perjelet ne használjuk a pont előtt! A ***local*** vagy a ***patterns*** állományok módosításai a

```
killall -HUP squirm
```

parancs kiadása után lépnek életbe. A böngészővel próbáljuk meg elérni a ***www.playboy.com*** gépet. Ha minden működik, a Disney oldala jelenik meg.

A viralator telepítése

A program főoldala a ➲ http://viralator.loddington.com/ címen található. Töltsük le a ***http://viralator.loddington.com/downloads/viralator-09pre2.zip*** állományt (ha valakinek rokonszenvesebb a suEXEC-es telepítés, a weboldalon közzétett útmutatót kövesse, bár én nem ezt választottam). Bontsuk ki (egyetlen állományt tartalmaz), és tegyük a webkiszolgálónk ***cgi-bin*** könyvtárába (ami Debianon a ***/usr/lib/cgi-bin***) ***viralator.cgi*** néven. A jogokat állítsuk be, például

```
chown root.www-data /usr/lib/cgi-bin/
➥viralator.cgi
chmod 750 /usr/lib/cgi-bin/viralator.cgi
```

A webkiszolgáló gyökerében ***downloads*** néven készítsünk könyvtárat, és olyan jogokkal ruházzuk fel, hogy a ***viralator.cgi*** képes legyen benne fájlokat elhelyezni, olvasni és törölni:

```
mkdir /var/www/downloads
chown root.www-data /var/www/downloads
chmod 770 /var/www/downloads
```

A viralator a kért állományokat a `wget` segítségével ide fogja letölteni, a vírusirtót szintén itt futtatja majd, és az ügyfelek böngészői is ebből a könyvtárból fognak letölteni. Továbbá a naplóállományát ***viralator.log*** néven ugyancsak ebben a könyvtárban hozza létre. A ***viralator.cgi*** állományt a kívánt nyelvnek és vírusirtónak megfelelően módosítsuk, például:

```
# A little housekeeping first

$default_lang = "en";
$antivirus="SOPHOS";
```

Ezután változtassuk meg az elérési útvonalakat:

```
$downloads = "/var/www/downloads";
$downloadsdir = "/downloads/";
$logfile = "$downloads/viralator.log";
$wget = "/usr/bin/";
$deleteaction = "deletefile";
#$deleteaction = "mantain";
```

A `$deletaction`-nel kezdődő sorok közül válasszuk a nekünk megfelelőt. A `"deletefile"`-nál a ***downloads*** könyvtárból törli az állományt, míg a `"mantain"`-nel nem.
Ezt követően a ***/etc/squirm/squirm.patterns*** fájlt át kell írnunk a *listán* látható módon.
Természetesen a példában lévő IP-cím és gépnév helyett saját gépünk adatait adjuk meg. Az `abortregexi` rész állítja meg az átirányítást, ha tehát a letöltést a Squidet futtató gépről végezzük, a viralator nem indul el.

```
abortregexi (^http://10.0.0.1/.*)
abortregexi (^^http://proxy.webhely.hu/.*)
regexi (^.*\.zip$) ^http://10.0.0.1/.* cgi-bin/viralator.cgi?url=|\1
regexi (^.*\.doc$) ^http://10.0.0.1/.* cgi-bin/viralator.cgi?url=|\1
regexi (^.*\.exe$) ^http://10.0.0.1/.* cgi-bin/viralator.cgi?url=|\1
```

A három `regexi`-s sor a viralatort csupán akkor indítja el, ha .ZIP, .DOC vagy .EXE állományt szeretnénk letölteni.
A `regexi`-s rész szabályos kifejezése azért található gömbölyű zárójelek között, hogy a ***viralator.cgi*** a letölteni kívánt URL-t átadott értékként kaphassa.
Indítsuk újra a Squirmet a `killall -HUP squirm` paranccsal. Néhány állomány letöltésével próbáljuk is ki. Ehhez keressük fel a ➲ http://viralator.loddington.com/ downloads/leicar.zip címet, mert az állomány próbavírust tartalmaz.

Hibák

Bár az alkotó a programot folyamatosan fejleszti, sajnos akadnak benne hibák. Netscape-pel, Konquerorral tökéletesen működik, de Opera vagy Internet Explorer alatt folyamatos letöltési ciklusba kerül. A javítás megjelenéséig a 432. sort tegyük megjegyzésbe:

```
#print "<META HTTP-EQUIV=\"refresh\"
➥CONTENT=\"5 \;URL\=$requestpage\">\n";
```

Ekkor a ***downloading ...*** ablakról kézzel kell visszaléptetnünk, de a végtelen ciklus elmarad. A ➲ http://viralator.loddington.com/-on olyan fórum is működik, ahol felhívják a figyelmet a hibákra, sőt a program használói megoldásokat is kínálnak rájuk. Az oldal GYK-t is tartalmaz, böngészését melegen ajánlom.

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

Kapcsolódó címek

➲ http://viralator.loddington.com
➲ http://www.squid-cache.org/related-software.html

Webmin

Jól használható eszköz mind a guruk, mind a Linuxszal éppen most ismerkedők számára – a modulrendszerű tervezés és a háttéradatbázis nélküli működés hatékony, könnyen alakítható felületet biztosít.

Hadd kezdjem a program bemutatását a készítők honlapjáról vett idézettel: „A Webmin webalapú felület unixos rendszergazdák számára". Néhány éve bukkantam rá a Webminre, amikor jómagam éppen a ***webmin.com*** névtartományt szerettem volna bejegyeztetni. Ragyogó irányítópultról álmodoztam, amelyről a kiszolgálóparkom működését irányíthattam volna, és nagyon felháborított, hogy valaki már megszerezte a kiötölt nevet. Ám nyomban megfeledkeztem a bosszúságomról, mihelyt megláttam a honlapon a letöltésre szánt programokat – innentől kezdve már rajongással böngésztem tovább.

A Webmin ama célkitűzése, hogy webalapú felületet kínáljon a feladatok megoldásához, nem egyedi. A Világhálón számos ilyen eszköz található, amelyek között mind nyílt forrású, mind kereskedelmi termék megtalálható.

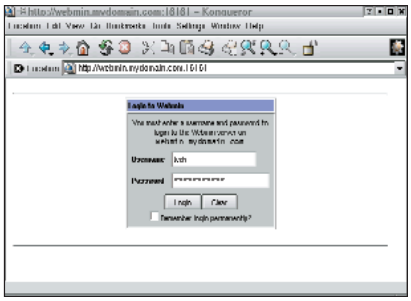
A Webmin azért tűnik ki a többi közül, mert tapasztalt és újdonsült rendszergazdák számára egyaránt célravezető választás lehet. Legelőször tekintsük át a telepítését, és azt is, hogy milyen feladatok megoldására alkalmas.

Az első lépések

Mielőtt elmélyednénk a Webmin szolgáltatásainak taglalásában, ejtek néhány szót a program kialakításáról. A Webmin alapvetően Perl nyelven íródott CGI-héjprogramok óriási gyűjteménye. Saját webkiszolgálóját azon a kapun működteti, amilyet a telepítés során megadunk neki, s ez teszi lehetővé a számunkra, hogy a Webmin biztonságát a tényleges webkiszolgálótól függetlenül kezeljük.

A támogatott felületek teljes listája a honlapon tekinthető meg, és ezek között az alább felsorolt rendszerek is fellelhetők: RedHat, Solaris, Debian, OpenBSD, HP-UX, IRIX, AIX, DEC, SCO és Mac OS X.

A Webmint a modulrendszerű tervezési séma teszi egyedülállóvá. Az összes szolgáltatás és program által biztosított lehetőség a modulok rendszerén keresztül valósul meg, ami a következőt jelenti: amennyiben a használni kívánt alkalmazást a Webmin nem támogatja, alkalmazói programfelületével (API) létrehozhatjuk a megfelelő modult, és ezzel már a kiválasztott program működésének irányítására is képes lesz. A Webminhez ötven szabványos modul tartozik, sőt, rajtuk kívül számos további modul is létezik, így hát temérdek forráskód segíti az első lépések megtételét.



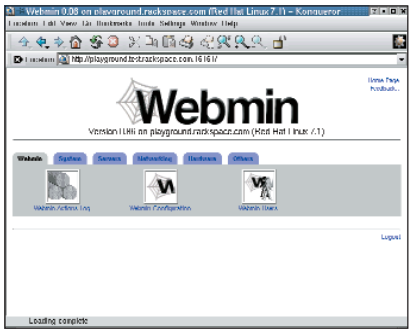
1. kép Bejelentkezés a Webminbe

Annak ellenére, hogy a Webmint magát a BSD-szerződés feltételeinek megfelelően terjesztik, a saját fejlesztésű modulok már tetszőlegesen választott szerződésfajta alapján terjeszthetők. Ez a megoldás a további programfejlesztés lehetőségét a Nyílt Forráskód Közössége és a kereskedelmi programok előállítói számára egyaránt nyitva hagyja.

A program telepítése enyhe fuvallathoz hasonlítható: elsőként látogassunk el a ➲ http://www.webmin.com/webmin címre, majd töltsük le az RPM-csomagot vagy a tar-állományt. Amennyiben az utóbbit választjuk, kicsomagolás után a `setup.sh` telepítő héjprogramot kell futtatnunk – ez fogja kezelni a telepítési folyamatot. A tar-t választók azonban győződjenek meg róla, hogy abban a könyvtárban végezték-e el a programok kicsomagolását, ahol azok működni is fognak! A Webmin kicsomagolása után ügyeljünk rá, hogy az éppen kicsomagolt ***webmin*** könyvtárat ne töröljük le, máskülönben a program nem fog működni. Amennyiben a Webmint a későbbiek folyamán törölni szeretnénk, a kezdeti könyvtárat a telepítő programcsomagban szereplő eltávolítóprogram fogja törölni az összes többi állománnyal együtt.

Miután befejeztük a Webmin telepítését, indítsunk egy böngészőprogramot, és adjuk meg az IP-címet- vagy egy feloldható kiszolgálónevet a hozzá tartozó kapucímmel – alapértelmezés szerint 10 000 – együtt (*1. kép*).

A képernyőn a bejelentkezést követően a Webmin üdvözlőoldala jelenik meg a kategóriafülekkel. A program moduljait a *2. képen* látható módon csoportosították, például a ***Webmin*** menüfülön található az általános jellemzőkre,



2. kép A Webmin felülete

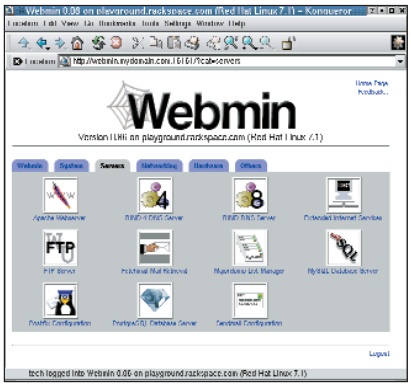
felhasználókra, modulokra stb. vonatkozó összes beállítási lehetőség – a program ezeken a helyeken az alapértelmezés szerinti számokkal fog működni.

A további füleket a ***System***, ***Servers***, ***Networking***, ***Hardware*** és ***Others*** címkékkel látták el.

A ***System*** kategória alatt rejtőző modulok olyan feladatok kezelését végzik, amelyek magának az asztali számítógépnek vagy a kiszolgálógépnek a működését érintik. E feladatok között szerepel többek között a lemezterület-felhasználás, az NFS- és NIS-jellemzők beállítása, a PAM-jellemzők módosítása, a rendszernapló megtekintése, új felhasználók felvétele, a cron működésének és a rendszerbetöltés idejére időzített szolgáltatásoknak a szabályozása, de a gép újraindítása is.

A ***Servers*** lap tartalmazza az összes

kiszolgálóbeállító modult, vagyis itt találhatjuk meg az Apache-, BIND-, DHCP-, Sendmail-, Squid- és még számtalan egyéb modult (*3. kép*).
A ***Networking*** az egyik újdonság, ami Linuxon a remek grafikus felületű IP Chains beállító modult tartalmazza, továbbá egy csokorravalót a hálózati segédprogramokból, így például a `ping`-et, a `traceroute`-ot, a `whois`-t és a `dig`-et.

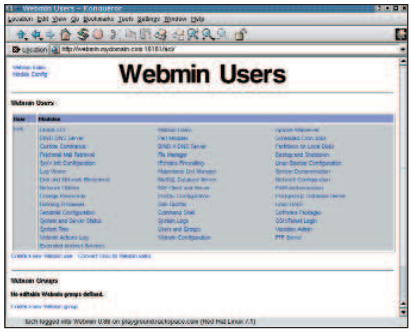


3. kép Webmin-modulok a Servers részben

A ***Hardware*** lapon természetesen közvetlenül a számítógép belső felépítésével kapcsolatos jellemzőkkel találkozhatunk. E helyen szerepelnek olyan adatok, mint a merevlemezek felosztása, a rendszeridő, a hálózati csatolókártya beállításai, a LILO és a RAID-kezelő programrendszer.
Az utolsó lapot joggal tekinthetjük a „mindent bele” kategóriának, hiszen teljes körű szolgáltatást nyújtó Java-alapú SSH/Telnet ügyfélprogram; Java-alapú állománykezelő; különleges, a felhasználó által meghatározott modul, amelyet ***Custom Commands***-nak neveznek; rendszernapló-olvasó program; Perl-modulok; valamint webalapú kiszolgáló- és rendszerállapot-figyelő modul található benne. A ***Custom Commands*** modullal egyszerű fogadófelületet készíthetünk a Webmin számára, amely tetszőleges parancs futtatását teszi lehetővé. Ez igen előnyös, amennyiben a Webmin csupán olyan célfeladat elvégzéséhez szükséges, amely önálló modulhasználatot nem igényel. A *4. kép* ilyen könnyen és gyorsan elkészíthető fogadófelületet mutat be.

A lépésenkénti útmutató

Az új, virtuális gépen futó Apache-kiszolgáló beállításához lépésenkénti útmutató áll rendelkezésünkre. Ez a leírás megmutatja, miképpen kell saját dokumentumot (főoldalt) szerkeszteni, továbbá hogyan kell a `cgi-bin`-t és a naplózást beállítani. A leírás a valóságosnál sokkalta bonyolultabbnak tűnik. Megállapítottam, hogy a „kattintós” grafikus felületen az irányok pontos meghatározása sokkal nagyobb erőfeszítést követel, mint az egyes objektumok kijelölése és az egérrel való kattintás.



4. kép Saját parancsok létrehozása

Jelenlegi példánkban a „dirk” felhasználói azonosító már előzetesen is létezett, de a ***test.com*** virtuális gépet és a / névtartományt csak most fogjuk létrehozni. A honlapfeltöltésre a Wu-FTPd-t használjuk, és feltételezzük, hogy a DNS már képes feloldani az új nevet. Először hozzunk létre HTML-állományokat, naplóállományokat és CGI-héjprogramokat tartalmazó könyvtárakat a webhely számára.
Látogassunk el ismét a Webmin főoldalára, és válasszuk az ***Others*** lehetőséget, ezt követően indítsuk el az ***Állománykezelőt***. Az Állománykezelő segítségével jussunk el a webhely rendszergazdájának saját könyvtárához. A felhasználó ugyanis ide kerül, amikor FTP-n keresztül kapcsolódik a kiszolgálóhoz. Így ennek a felhasználónak a saját könyvtára: ***/home/felh_nev***. Kattintsunk duplán a bal oldalon a ***/home*** könyvtáron, majd ismét duplán a rendszergazda nevét viselő könyvtáron. Elérkezett az ideje, hogy létrehozzuk a webhely számára szükséges könyvtárakat. Az ***New Folders*** ikonra történő kattintással – pontosabban az ***New*** felirattal és könyvtárikonnal ellátott gombon – a névtartomány számára létrehozzuk a főkönyvtárat, a ***/home/dirk/test.com***-ot. A könyvtárba való belépéshez az újonnan létrehozott könyvtáron, vagyis a ***test.com***-on kell duplán kattintani. Ezen belül három további könyvtár létrehozása szükséges: ***htdocs***, ***logs*** és ***cgi-bin*** névvel. Amennyiben a munkánkat jól végeztük, az alábbi könyvtárak jöttek létre:

```
/home/dirk/test.com,
/home/dirk/test.com/htdocs,
/home/dirk/test.com/logs,
/home/dirk/test.com/cgi-bin
```

Most térjünk vissza a felhasználó saját könyvtárához. Kattintsunk a jobb oldalon a webhelyhez tartozó könyvtáron (például a ***test.com***-on), majd a lap tetején levő ***Info*** gombon. Az ***Info*** ablakban a felhasználó- és csoportazonosítót állítsuk be a valóságnak megfelelően. Az ***Apply changes to*** szakaszban lévő lenyíló listából válasszuk a ***This directory and all subdirectories*** lehetőséget, majd a beállítást mentsük.
Lépjünk vissza a Webmin főoldalához a lap tetején található ***Return to index*** feliraton kattintva. Válasszuk a ***Servers*** fület, és az egérrel kattintsunk az Apache-kiszolgálón. Amennyiben az Apache-kiszolgálót első alkalommal indítottuk a Webminben, a Webmin engedélyt fog kérni, hogy ellenőrizhesse az Apache beállításait, ekkor egyszerűen kattintsunk a ***Configure*** gombon. Ezután pedig menjünk a lap aljára, ahol egy beviteli mező található – ez szolgál az újonnan létrehozott webhelyek megadására.
A szövegmezőbe gépeljük be a webhely IP-címét. Amennyiben névalapú weblap elhelyezése mellett döntünk, ne feledkezzünk meg róla, hogy az ***Add name virtual server address*** lehetőség be legyen kapcsolva. A következő helyen található: ***/home/felh_nev/tartomany/htdocs***, vagyis valóságos névvel: ***/home/dirk/test.com/htdocs***. A szöveges típusú ***Server Name*** mezőt értelemszerűen töltsük ki: ***www.test.com*** és kattintsunk a ***Create*** gombon.
A virtuális kiszolgálók névsorát addig görgessük, amíg rá nem bukkanunk az éppen létrehozott webhelyre. Kattintsunk a webhelyhez tartozó virtuális kiszolgálón. A `cgi-bin` könyvtár kialakításához kattintsunk a ***CGI Programs***-ra és a ***Form*** mezőbe gépeljük be: ***/cgi-bin/***, a szövegmezőbe pedig írjunk ***/home/felh_nev/nevtartomany/cgi-bin/***-t. Fontos, hogy ez utóbbi útvonal perjelre (/) végződjék, vagyis most már névvel együtt: ***/home/dirk/nevtartomany/cgi-bin/***. Végül mentsük ezt is lemezre.
Az egérrel kattintsunk a ***Log Files***-ra. A ***File or Program*** feliratú mezőben szereplő naplóállományhoz való hozzáférés naplózásához gépeljük be a ***/home/felh_nev/tartomany/logs/access_log*** szöveget, azaz a felhasználói azonosítóval és a példaként választott tartománnyal kiegészítve: ***/home/dirk/test.com/logs/access_log***. Amennyiben azt szeretnénk, hogy

Kapcsolódó címek

Rackspace-szel kapcsolatos cikkek a ➲ http://support.rackspace.com/kbsearch.php3 címen érhetők el. Válasszuk a *Platform=Linux* és *Details=Webmin* lehetőségeket.
„A Webmin és a rendszergazdai feladatok" címmel használati útmutatóra bukkanhatunk a ➲ http://www.swelltech.com/support/webminguide/index.html címen.
A Webmin saját honlapja a ➲ http://www.webmin.com/webmin címen olvasható, maga a program is innen tölthető le.

a napló gyakori keresési adatokat tartalmazzon a kérő nevével együtt, az ***Access log files row*** lehetőségnél a ***Format*** oszlopban váltsunk át az alapértelmezésről a szövegmezőre úgy, hogy bejelöljük az előtte található jelölőnégyzetet.
Ezt követően a szövegmezőbe írjuk be a ***combined*** szót. A beállítást mentsük, majd az egérrel kattintsunk a ***Networking and Addresses*** lehetőségen. A ***Server admin email address*** mezőbe e hely webmesterének a levélcímét gépeljük, majd a mező előtti négyzet bejelölésével végezzük el a mentést.
Ezzel a webhely alapvető beállításait tulajdonképpen be is fejeztük, de mielőtt megkezdené önálló életét, még egy végső és nem kevésbé fontos lépést is meg kell tennünk. A jobb felső sarokban ugyanis egy ***Apply changes*** felirattal ellátott gomb található: kattintsunk ezen a gombon, ezzel juttatva érvényre a változtatásokat.

Biztonság

A Webmin számos biztonsági szolgáltatást nyújt. Az első védelmi vonal olyan felhasználói azonosító és jelszóhitelesítő rendszer, amely teljesen független a */etc/passwd* állományban tárolt felhasználói azonosítótól. Ez azt jelenti, hogy valakinek anélkül is jogot adhatunk a Webminhez történő hozzáférésre, hogy bármilyen más operációsrendszer-szintű jogosultságot kellene adnunk neki.
A Webmin teljes mértékben támogatja az SSL-t. Amennyiben a gépünkön Perl SSL-modul található, a Webmin-kapcsolatokat teljes mértékben titkosítani lehet, ily módon a támadókat megakadályozza abban, hogy lehallgatással adatokat szerezzenek.
A Webmin a különböző, már hozzáférhető modulok finomhangolását is lehetővé teszi, például a felhasználók számára a teljes DNS-kiszolgálóhoz hozzáférési jogot biztosíthatunk anélkül, hogy ez a hozzáférési jog az Apache-beállításokra is kiterjedne, vagy éppen ellenkezőleg: a hozzáférési jogokat kizárólag arra a névtartományra korlátozhatjuk, amely fölött ők maguk rendelkeznek.
Amennyiben igényeljük az egyes feladatoknak más-más rendszergazdákra való átruházását, jól kihasználható a vezérléskorlátozási és -újraelosztási lehetőség. Végezetül arra is módunk van, hogy a Webmint úgy állítsuk be, hogy az összes, a felületén keresztül végzett módosítást naplózza – hibakeresés során ez a szolgáltatás rendkívül jól használható.

Mi teszi a Webmint nagyszerűvé?

Amint már bizonyára kitalálták, nagyon kedvelem a Webmint. Szeretem, hogy a szerződés alapján hozzájuthatok a forráskódhoz és módosíthatom is, amennyiben éppen erre van szükségem. Azt is élvezem, hogy a modulrendszer révén akár magam is új dolgokat hozhatok létre, vagy beépíthetem a mások által készített modulokat. Jelenleg a Webmin számára készített LTSP-modul próbálgatásával foglalkozom, hogy néhány rakoncátlan I-Opener megszelídítésében segédkezzem. A lehetséges feladatok kevésbé tapasztalt rendszergazdákra (szobatársakra) való átruházása, valamint az a tudat, hogy a számukra kijelölt területtől nem térhetnek el, jelentősen csökkenti a rám váró feladatok mennyiségét.
Ha a Webmin csak ezt tudná nyújtani, már akkor is el lennék ragadtatva, de akad egy további előnye is: a rendszerállományokat közvetlen módon éri el, vagyis sem adatbázist, sem más szabványostól eltérő adattárolási módot nem használ. Ennek következtében anélkül módosíthatom kézzel az Apache-hoz tartozó ***httpd.conf***-ot , hogy az elkövethető hibák miatt kellene aggódnom.
Az ügyféltámogatás tekintetében ez azt jelenti, hogy a Webmint telepíthetem a kiszolgálóra, és a működtetését nyugodtan másra bízhatom. Amennyiben a gondokkal nem tudna megbirkózni, a hiba elhárítására még mindig használhatom a héjprogramjaimat és vi-ismereteimet.
A beállítást a barátságos parancssor és a háttérbeli adatbázis nélkülözése miatt a közönséges állományokra bízza, amit a vezérlőpultok tervezői túlságosan gyakran hajlamosak figyelmen kívül hagyni. Így végezetül olyan rendszereket állítanak elő, amelyekben mindent a vezérlőpulton keresztül kell beállítani, máskülönben a program befejezi a működését. A Webmin szabad kezet ad nekem rendszergazdai feladataim intézési módjának kiválasztásában. Az Apache beállításait például jobban szeretem közvetlenül módosítani, a BIND-dal viszont teljesen más a helyzet. A BIND hírhedten szőrszálhasogató program, ezért kényelmes beállítófelületként a Webmint használom hozzá. A program az összes – különben rejtett – lehetőséget felajánlja, és nagymértékben csökkenti az elírásból adódó névfeloldási hibák arányát. Örömmel tölt el, hogy a Webmin mennyire jól beleillik rendszergazdai eszköztáramba.
A kezdő rendszergazdák szolgáltatásai mélységének köszönhetően hamar meg fogják szeretni a Webmint. Az egérkattintásokkal működő grafikus felület biztosítja, hogy nem kell mindent fejben tartani, ez azonban a kiszolgálók újdonsült rendszergazdái számára akár elrettentő feladatnak is bizonyulhat.
A Webmin magmoduljai az általuk támogatott szolgáltatások szinte minden képességét és jellemzőjét felvonultatják. Ez azt jelenti, hogy könnyedén vehetünk fel olyan új beállítási lehetőségeket, amelyek létezéséről korábban nem is tudtunk.
A Webmin jólszervezettsége és szolgáltatásbeli gazdagsága ellenére mindenki figyelmét szeretném felhívni rá, hogy a program mégsem teljesen kezdők számára készült. Ha valakinek fogalma sincs róla, mi a DNS-ben a bejegyzés, a Webmin nem fog segíteni rajta.
A Webmin a háttérben futó Linuxot webfelületen jeleníti meg, így amikor egyszerre nyerünk ennyi rugalmasságot és erőt, cserébe fel kell áldoznunk valamennyit a hatékonyságból. Ha valaki részletes ismeretekre tesz szert a szolgáltatások alapelveiben, annak kezében a Webmin nagyszerű eszköz lehet – azt azonban már senki ne várja el, hogy a program az O'Reilly kiadó által megjelentetett nagy BIND-kézikönyv összefoglalóját is megadja.

Dirk J. Elmendorf a Rackspace Managed Hosting cég egyik alapítótagja. Kutatásfejlesztési vezetőként az új termékek fejlesztésében és értékelésében is közreműködik, amelyeket egy hittérítő buzgalmával népszerűsít.

Vírusos levelek? Kizárva!

A levélalapú vírusok megállításának legjobb módja az, ha be sem eresztjük őket a hálózatunkra.

Múlt hónapban láthattunk egy módszert, hogy miként alkalmazhatjuk az Amavis csomagot a levélforgalom vírusellenőrzésére. Nagyobb hálózatoknál komoly gondot jelenthetnek a vírusok, főleg azért, mert ha egyszer bejutottak, akkor kegyetlen gyorsasággal végigfertőzhetik a munkagépeket. Manapság rendkívül fontos kérdés ez, és bár szerencsére a linuxos gépek esetében nem hallunk sokat vírusokról, a munkagépek védelmét is jellemzően kiszolgálóinkkal kell elősegítenünk, hiszen ezáltal rengeteg felesleges munkától mentjük meg magunkat. A belső hálózatot érő támadások leggyakoribb formái a levélvírusok. Az első lépés, amit egy rendszergazda általában tenni szokott ellenük: vírusvédelmi rendszert telepít a munkaállomásokra. Ez bölcs dolog, de számomra járhatóbb útnak tűnik, ha a vírusok rendszerbe jutását mindjárt a bejáratnál meggátoljuk. A vírusok, különösen a makróvírusok, messze leggyakoribb belépési pontja a szervezet levelezőrendszere. Mégis többnyire ez a vírusvédelmi rendszerek legelhanyagoltabb része. A piacon jelenleg kapható levélvírus-védelmi rendszerek gyakran alkalmazáshoz kötöttek, drágák, vagy mindkét tulajdonságot felmutatják (nem beszélve a megbízhatatlanságukról). Közepes méretű vállalkozás lévén a miénk, az idei költségvetésbe vírusirtó csomag beszerzését nem tervezték, így aztán csupán a Linuxhoz és a nyílt forráshoz fordulhattunk. Találtam is az Interneten néhány igen érdekes projektet, amely esetleg megfelelhetett volna az igényeinknek, de végül mégis a saját változat megírása mellett döntöttem. Azt szerettem volna elérni, hogy bármely felhasználó könnyen nyomon követhesse a rendszerünket, és egyszerűen bővíthesse, anélkül, hogy C- vagy Perl-guru lenne. A másik célom az volt, hogy a rendszer ki tudja azokat a hatékony eszközöket használni, amelyek általában minden linuxos alapterjesztésben megtalálhatók. E két tényező biztosítja a program hordozhatóságát, illetve, hogy más is képes legyen kezelni a rendszert a segítségem nélkül.

A rendszer alapjait Bash-héjprogramok, a `metamail`, a `grep`, az Obtuse Systems SMTPd termékei, a Samba és egy parancssoros víruskereső alkotják. Az *1. ábrán* egy folyamatábra stílusú vázlatot láthatunk. Az Obtuse Systems SMTP-tároló és -továbbító csomagja ingyenesen hozzáférhető a ➲ http://www.obtuse.com/smtpd.html címen. E sorok írásának idejében a legfrissebb változat a 2.0. Az általam választott víruskereső a McAfee Virus Scan for UNIX/Linux volt, de számos másikat is választhattam volna. Ezek közül némelyek ingyenesek, mások nem. Mindenféleképpen olyat válasszunk, amelyik a keresés eredményének megfelelően állítja be a kilépési értékét, és amelyhez rendszeresen letölthetünk az ujjlenyomat-frissítéseket. A rendszert felépíthetjük egy már meglévő linuxos tűzfalon, de akár egy külön gépen is, amennyiben linuxos tűzfal esetleg nincs kéznél. Ha erre a célra külön gépet használunk, nem kell túlságosan nagy teljesítményűnek lennie, egy 200 MHz-es 586-os 32 MB memóriával tökéletesen megfelel. A hálózatunk SDSL segítségével kapcsolódik az Internethez, a védelmet pedig egy IP-álcázást (masquerading) futtató Mandrake linuxos gép biztosítja. Ez a felépítés megkönnyíti a tűzfal telepítését.

A belső levelezőrendszer nem annyira fontos, elegendő, ha SMTP vagy ESMPT alatt működik. Mi például a Novell Groupwise termékét használjuk. Minden SMTP-forgalmat (25-ös kapu), ami a tűzfal SMTP-kapujára érkezik, át kell ahhoz a belső géphez irányítanunk, amelyen az SMTP-tűzfalat kiépítettük (vagy magához a tűzfalgéphez, mint a mi esetünkben is). Most lépjünk tovább a tulajdonképpeni beállításokhoz! Az első lépés a könyvtárszerkezet felállítása. Az ide vonatkozó vázlatot a *2. ábrán* találhatjuk. Rendszerünket a */var/spool/smtpd* könyvtár alatt fogjuk felépíteni. Amennyiben átlagos levélforgalmunk meghaladja a napi 25 000 levelet, javaslom, külön lemezrészt fűzzünk be a */var/spool/smtpd* könyvtár alá. Az alapkönyvtár tehát a */var/spool/smtpd* lesz. Ebben a könyvtárban öt alkönyvtárat hozunk létre: ***incoming*** (bejövő), ***outgoing*** (kimenő), ***etc***, ***bin*** és ***quarantine*** (karantén). Először is váltsunk át rendszergazdai jogosultsággal rendelkező felhasználóra, majd gépeljük be a következő parancsot:

```
mkdir -p  /var/spool/
➥smtpd/{etc,bin,incoming,outgoing,quarantine}
```

Következő lépésként állítsuk be a jogosultságokat, hogy a teljes könyvtárrendszert csak a `uucp`-felhasználó érhesse el, mivel az összes program e felhasználó jogosultságával fog futni. A következő parancsok megoldják számunkra a gondot:

```
chown -R uucp.uucp /var/spool/smtpd
chmod 700  /var/spool/smtpd
```

Most már beállíthatjuk a rendszer első összetevőjét. A korábban említett Obtuse Systems honlapjáról le kell töltenünk a ***smtpd*** csomagot. A letöltött fájl könyvtárában adjuk ki a következő parancsot:

```
tar -xzvf smtpd-2.0.tar.gz
```

Ezután váltsunk az ***smtp-2.0*** könyvtárba és a ***Makefile***-t szerkesszük át a következők szerint:

```
SPOOLDIR = /var/spool/smptd

SPOOLSUBDIR = incoming

POLL_TIME = 300

PARANOID_SMTP = 1

JUNIPER_SUPPORT = 0

CHECK_IDENT = 0
```

Azt szeretnénk elérni, hogy az `smtpd` a leveleket az ***incoming*** alkönyvtárban tárolja, a `smtpfwdd` pedig az ***outgoing*** alkönyv-

tárból olvassa őket. Hogy ezt lehetővé tegyük, az ***smtpfwdd.c*** fájlba a 75. sornál szúrjuk be a következő két sort:

```
// levelek let ltØse az outgoing alk nyvtÆrb l
#define SPOOLSUBDIR "outgoing"
```

Befejezésül fordítsuk le és telepítsük a csomagot a következő parancsokkal:

```
make

make install
```

A következő lépés az */var/spool/smtpd/etc* könyvtár benépesítése néhány, az `smtpd` helyes működéséhez szükséges állománnyal. Másoljuk a ***resolv.conf*** fájlt a */etc* könyvtárból a */var/spool/smtpd/etc* könyvtárba, majd a */etc* könyvtárból a ***localtime*** fájlt is másoljuk ide. Az smtpd-2.0 terjesztés könyvtárából az ***antirelay_check_rules_example*** fájlt átmásolhatjuk a */var/spool/smtpd/etc* könyvtárba. Amennyiben szükségünk van ilyesmire, az Obtuse System honlapján nézhetünk körbe további ellenőrző szabályokkal kapcsolatos utasításokért. Az `smtpd` program önműködő indításához a következő sort kell a */etc/inetd.conf* fájlba helyezni:

```
smtp    stream        tcp    nowait root
➥/usr/local/sbin/smtpd     smtpd
```

Ezt a sort az esetleg már meglévő `smtp`-sorok helyére kell írni. Az `smtpfwdd` programot a */etc/rc.d/rc.local* fájlból (vagy ahogy az *rc* fájlunkat éppen nevezik) kézzel kell elindítanunk. Fogjunk hozzá, és a következő sort írjuk be:

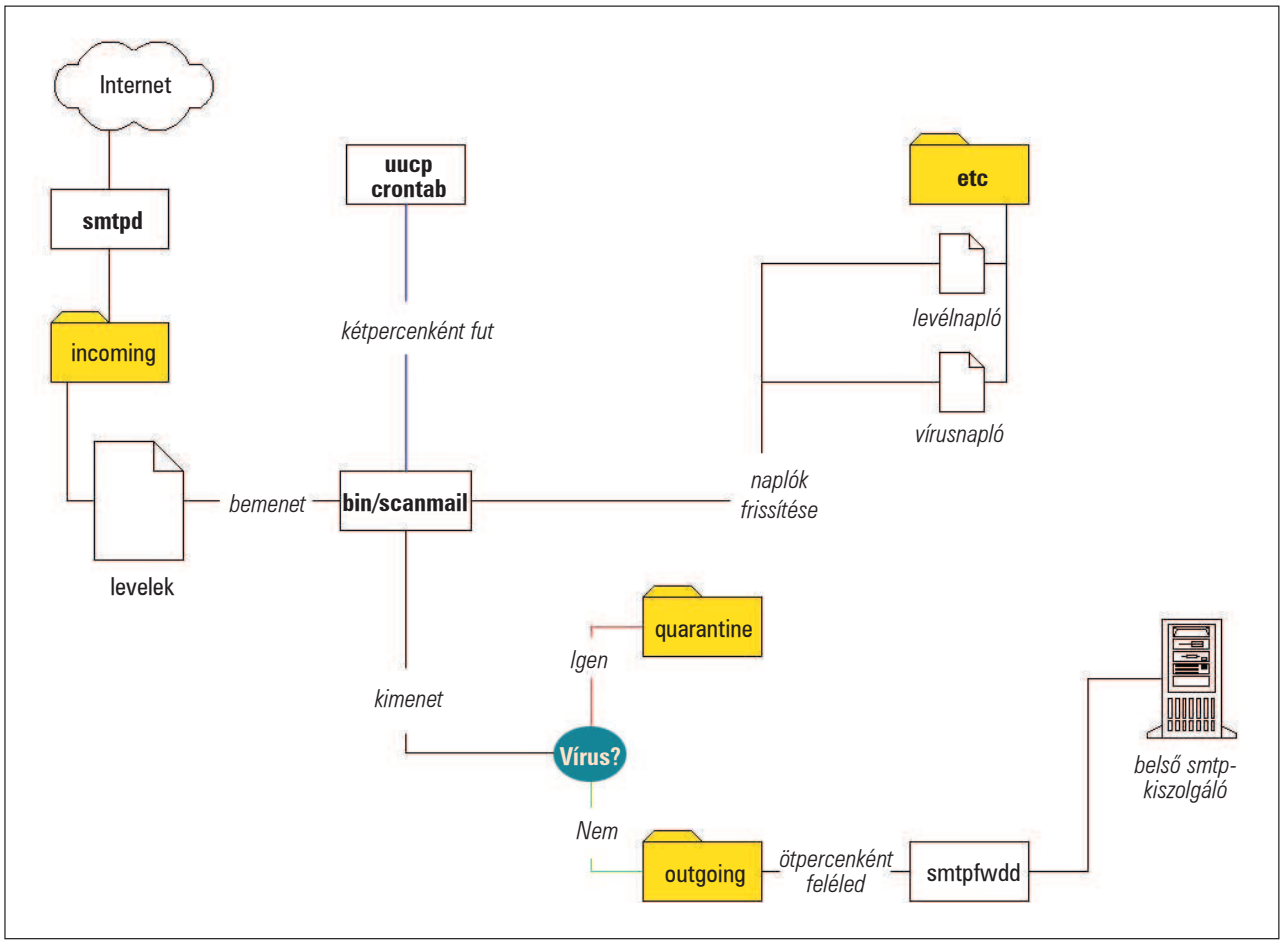
```
### Az smtpfwdd tovÆbb t  dØmon ind tÆsa
/usr/local/sbin/smtpfwdd
```

Végül le kell állítanunk minden esetleg még futó levéltovábbító ügynökprogramot (MTA-t). Ide értendők a ***Postfix***, ***Sendmail***, ***Qmail*** és társai. RedHat-rendszereken ezt egyszerűen a beállítóprogram segítségével is megtehetjük, amennyiben az összes MTA-t leállítjuk. Figyeljünk arra, hogy némely MTA például a Postfix vagy más folyamatok gyermekeiként futnak, így közvetlenül a `kill` paranccsal nem lehet őket „meggyilkolni”. A következő két parancs kiadásával indítsuk be az `smtpd` és az `smtpfwdd` démonokat:

```
kill -HUP `cat /var/run/inetd.pid`
➥/usr/local/sbin/smtpfwdd
```

Mikor már futnak a démonok, ki is próbálhatjuk őket, ha levélszűrő tűzfalunk 25-ös kapuján (ez az `smtp`-kapu) elindítunk egy `telnet`-kapcsolatot:

```
telnet email.firewall.com 25
```



1. ábra A hálózati forgalom és a tűzfal felépítése

ahol az `email.firewall.com` a levélszűrő tűzfalunk neve. A következő visszajelzést kell kapnunk:

```
220 email.firewall.com SMTP ready,
Who are you gonna pretend to be today?
```

Ha bármilyen más üzenetet kapunk, valószínűleg elfelejtettük a kiszolgálón futó MTA-t kikapcsolni. A `ps -e` segítségével ezt könnyen kideríthetjük.

Nézzük csak, hol is tartunk? Ha minden jól ment, most van egy gépünk, amelyen az `smtpd` démon fut és leveleket fogad. Minden beérkezett levél egyszerű szöveges fájlként a ***/var/spool/smtpd/incoming*** könyvtárban tárolódik.
Hogy valóban így is van-e, a következő parancsokkal nézhetjük meg:

```
$ telnet email.firewall.com 25
helo   firewall.com
mail from: joe@firewall.com
rcpt to: fred@firewall.hu
data
Ez egy pr ba.
.
quit
```

Ne felejtsük el a levelünk törzsét egy egyetlen pontot tartalmazó sorral zárni! Ez mondja meg ugyanis a kiszolgálónak, hogy a levelet el akarjuk küldeni.
Ha minden jól megy, a ***/var/spool/smtpd/incoming*** könyvtárban most egy szöveges fájlt fogunk találni. Néhány percen belül a fájlnak el kell tűnnie, mi pedig egy levelet találunk a levelesládánkban. Az `smtd` a leveleket smtpd formátumban menti, ahol smtpd egy véletlenszerűen készített üzenetazonosító.

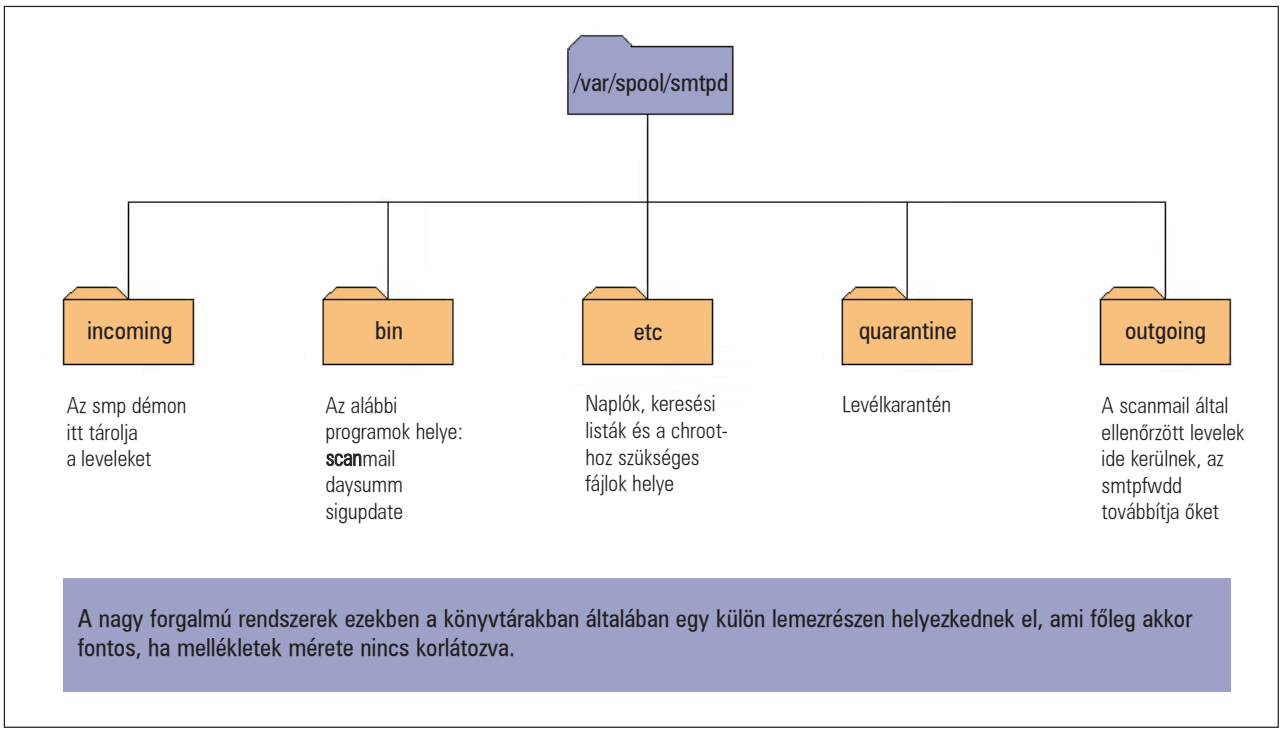
Az `smtpfwdd` ezeket a szöveges fájlokat beolvassa, és továbbítja őket a célkiszolgálónak. Sikeres továbbítás után a fájl törlődik.
De hogy is kerül a levél az ***incoming*** könyvtárból az ***outgoing*** könyvtárba? Nos, ez az a pont, ahol levélvizsgáló parancsfájlunk belép a képbe. A parancsfájl az ***incoming*** könyvtárban található fájlokat vírusra utaló nyomokat keresve egytől-egyig végignézi. Ha a fájl nem tartalmaz vírust, az ***outgoing*** könyvtárba kerül, ha viszont igen, a ***quarantine*** könyvtárba helyeződik át.
Ilyen egyszerű az egész. A levélvizsgáló parancsfájlhoz azonban előbb szükségünk lesz egy működő víruskeresőre, tehát előbb erre a feladatra összpontosítsunk.
Mindenféleképpen parancssoralapú víruskeresőre van szükségünk. Én a magam részéről – mint már említettem – a McAfee Virus Scan for UNIX/Linux rendszert választottam, így itt most erről fogok írni. A McAfee termék igen széles kilépőkód-listával rendelkezik, ezáltal a parancsfájlokba meglehetősen könnyen beilleszthető. A terméket egyszerűen beszerezhetjük ➲ http://www.nai.com honlap webboltján keresztül. A termék telepítése után bizonyosodjunk meg arról, hogy a `uucp` felhasználó által végrehajtható legyen. Ehhez be kell lépnünk a ***/usr/local/uvscan*** könyvtárba, majd ki kell adnunk a következő parancsot:

```
chmod -R 755
```

Próbáljuk ki, hogy működik-e! Váltsuk át `uucp`-felhasználóra, és írjuk be a következő parancsot:

```
/usr/local/uvscan/uvscan -version
```

Amennyiben a próba sikeresnek bizonyult, továbbléphetünk, és felfrissíthetjük a vírusujjlenyomatokat. Az ujjlenyomatfájlok (avagy a „definíciós” fájlok – ahogyan gyakran emlegetik őket) alkotják az összes víruskereső velejét. Ezért fontos, hogy



2. ábra A könyvtárszerkezet

a frissítés időről időre önműködő módon megtörténjen. Nagyon fontos, hogy ezeket a frissítéseket legalább kéthetente ellenőrizzük, mivel havonta legalább három új változat jelenik meg. Ehhez a megoldáshoz először egy `sigupdate` Bash-parancsfájlt fogunk alkotni, amelyet majd a */var/spool/smtpd/bin* könyvtárba helyezünk el. Akárcsak eddig, most is győződjünk meg arról, hogy az `uucp`-felhasználó-e a parancsfájl birtokosa, és hogy a fájlt végrehajthatónak jelöltük-e be. A parancsfájl tartalmát az 1. listában (24. CD Magazin/Virus könyvtár) találjuk, elég könnyen követhető. A `sigupdate` fájlt hetente egyszer fogjuk futtatni, lehetőség szerint a nap valamilyen kevésbé terhelt időszakában. Ezt később tesszük meg egy `crontab` sor beillesztésével. Továbbá készítenünk kell a `uucp`-felhasználó saját könyvtárában egy ***.netrc*** nevű fájlt. Szerkesszük át úgy, hogy a következőképpen nézzen ki:

```
machine ftp.nai.com
login anonymous
password admin@domain.com
macdef init
cd pub/antivirus/datfiles/4.x
bin
prompt
mget dat-*.tar
close
bye
```

A ***.netrc*** fájl előre meghatározott gazdagépek FTP-elérését vezérli. Ez a legjobb módja annak, hogy FTP-folyamatunkat önműködővé tegyük. A .netrc írásmódról az FTP súgóoldalon olvashatunk. Lássunk neki, és amint elkészült a két fájl, futtassuk le a `sigupdate`-et.
A frissítés végén futtassuk le a következő parancsot:

```
/usr/local/uvscan/uvscan -version
```

Nézzük meg a vírusadatfájl létrejöttének dátumát. Valamilyen közeli időpontot kell látnunk, általában egy hónapnál nem régebbit. Ha nem így lenne, parancssorból kell ellenőriznünk, hogy a `sigupdate` helyesen működik-e.
Most lépjünk tovább a fő levélvizsgáló parancsfájlra. Ezt a fájlt `scanmail`-nek fogjuk hívni és a */var/spool/smtpd/bin* könyvtárba kerül. Ez a parancsfájl fogja végrehajtani az összes közvetlen műveletet az `smtpd` által létrehozott levélszövegfájlok között. Készítsünk egy fájlt, majd tegyük végrehajthatóvá és adjuk át az `uucp`-felhasználónak. A teljes parancsfájlt a 2. listában találhatjuk meg (elérhető a 24. CD Magazin/Virus könyvtárában). Minthogy a fájl meglehetősen sok megjegyzést tartalmaz, itt csak nagy vonalakban térünk ki rá.
A 19. sortól kezdve a `scanmail` előbb belép az ***incoming*** könyvtárba, és az ott található fájlneveket egy vektorban helyezi el. Ezután minden egyes fájlnéven végiglépdel, és a `grep` segítségével a fájlokban adott mintákat keres. Minden levél tartalmát kétszer ellenőrizzük. Első körben azokat a csatolt fájlokat szűrjük ki, amelyek nyilvánvalóan rosszak. Az ehhez a kereséshez tartozó minták a ***matches.bad*** fájlban tárolódnak, amelyet később fogunk létrehozni. Ha a `grep` talál valamit, a levél a karanténba kerül, és egy levelet küldünk a rendszergazdának, amelyben megtalálható a dátum, a fájlnév, és hogy a levél kinek, illetve kitől érkezett.
Ha nem volt találat, jöhet a második kör. Ez esetben a `grep` a ***matches.doc*** nevű fájlt fogja használni, hogy kiszűrje azokat a csatolt állományokat, melyek makróvírusokat vagy beágyazott vírusokat tartalmazhatnak. Ha talál valamit, a csatolt részt a `metamail` program segítségével egy dinamikusan létrejövő ideiglenes könyvtárba bontja ki. Az ideiglenes könyvtár neve a csatolt állomány neve lesz `"_d"` utótaggal kiegészítve.
Az ideiglenes könyvtár tartalmát ezután a parancssoros víruskeresőnkkel végignézetjük.
Ha valamilyen vírust találtunk (ezt a kereső visszatérési értékéből tudjuk meg), a `scanmail` a levelet és a csatolt állományokat karantén alá helyezi, majd figyelmeztető levelet küld a rendszergazdának. Egyúttal udvarias levelet küldünk a levél feladójának, amelyben értesítjük, hogy érdemes lenne végignézni a rendszerét, illetve megadjuk a talált vírus nevét.
Ha ez idáig nem találtunk vírust, a levél az ***outgoing*** könyvtárba kerül, ahonnan az `smtpfwdd` a belső levélkiszolgálóhoz továbbítja majd. Az `smtpfwdd` a kimenő könyvtárat ötpercenként egyszer ellenőrzi.
A következő lépés annak a fájlnévlistának az elkészítése, amelyet a `scanmail` a gyanús csatolt állományok felderítéséhez fog használni. A `scanmail` a fájlok közt a `grep` eszköz segítségével keres. Kihasználjuk a `-f` kapcsoló nyújtotta előnyöket, mivel így a `grep` a kereséshez használt mintákat egy megadott szöveges fájlból fogja kiolvasni. A szöveges fájl szerkezete igen egyszerű, minden sorban egy minta található. A `grep` a fájlban felsorolt bármely mintával való egyezést találatként fogja értékelni. Váltsunk a */var/spool/smtpd/etc* könyvtárba és hozzunk létre két fájlt ***matches.bad*** és ***matches.doc*** néven. A ***matches.bad***-be az olyan fájlok névmintáit helyezzük, amelyeket semmiféleképpen nem szeretnénk anélkül a rendszerbe ereszteni, hogy a rendszergazda meg ne vizsgálta volna őket. A ***matches.doc*** fájlnak ezzel szemben azokat a dokumentummintákat kell tartalmaznia, amelyek beágyazott vírusokat tartalmazhatnak, ilyenek például a Word-dokumentumok és a táblázatkezelők állományai. Amikor ezeket a fájlneveket hozzuk létre, minden sorban a `filename=.*\.exe` formátumot használjuk. Ez azért szükséges, hogy ne kapjunk hamis riasztásokat a mimekódolás olyan véletlen karaktersorozatai miatt, amelyek történetesen megegyeznek a `grep` által keresett mintával. Figyeljünk arra is, hogy ez a fájl semmiképpen ne tartalmazzon üres sort, hiszen a `grep` az üres sort is keresendő mintának fogja venni, és minden levélre találatot fogunk kapni. A Vim jó szerkesztőprogram e célra, mivel könnyen láthatjuk a benne szereplő üres sorokat. Az általam használt fájlok tartalmát a 3. listában (24. CD Magazin/Virus köyvtár) lelhetjük fel.
A másik felhasznált parancsfájl neve `daysumm` lesz és a */var/spool/smtpd/bin* könyvtárban helyezzük el. Hozzuk létre ezt az állományt a *4. listának* megfelelően.
A `daysumm` a napi tevékenységről értesítőt küld a rendszergazdának. Megmutatja, hány levél érkezett aznap, közülük hány volt vírusos, illetve melyek voltak ezek a vírusok. A cronban állítjuk be, hogy minden este 11:59-kor fusson le.
A `daysumm` parancsfájl a */var/spool/smtpd/etc* könyvtárban elhelyezett ***virus.$date*** és ***email.$date*** fájloktól függ. Ezek szöveges fájlok, amelyek dinamikusan jönnek létre, a `scanmail` frissíti őket és dátumfüggők.
Emiatt a `daysumm` programot mindig éjfél előtt kell lefuttatnunk, különben az időbélyeg (timestamp) megváltozik és rossz fájlok kerülnek beolvasásra.
Biztos észrevették már, hogy valahányszor magáról a tűzfalról küldtünk ki üzenetet – például a parancsfájlokban is –, mindig egy `sendmail -q` parancsot is kiadunk. A `-q` ugyanis azt mondja meg a `sendmail`-nek, hogy induljon el, és nézzen körül, van-e kimenő üzenet, ha van, küldje el, majd lépjen ki. Ez hatékonyan kiüríti az összes kimeneti sort, ami azért szük-

4. lista A daysumm parancsfájl

```
#!/bin/sh
mailto="admin@domain.com"
timestamp='date '+%A - %B %d %Y
etc="/var/spool/smtpd/etc"
date='date '+%m%d%Y
email_total='cat $etc/email-log.$date'
virus_total='cat $etc/virus-log.$date'


mail -s "DAILY E-MAIL SUMMARY" $mailto <<eoi


  Date:     $timestamp
  E-Mail total: $email_total
  Virus total:  $virus_total

eoi

/usr/sbin/sendmail -q

exit 0
```

séges, mert többé semmilyen MTA nem fut a gépünkön.
Az ***smtpd*** csomag nem MTA, hanem egy tároló- és továbbítócsomag. Úgy is elképzelhetjük, mint egy kifejezetten levéltovábbításra kihegyezett programot. E külön parancs nélkül tehát soha egyetlen levelet sem kaphatnánk meg a tűzfalról.
Itt az ideje, hogy az egész folyamatot a cron démon segítségével önműködővé tegyük. Ezt a uucp-felhasználó személyes crontab állományának felhasználásával fogjuk megtenni. Lépjünk be uucp-felhasználóként, majd adjuk ki a `crontab -e` parancsot, ami az uucp-felhasználó cron tábláját nyitja meg szerkesztéshez. A `scanmail`, `daysumm` és `sigupdate` parancsfájlok részére hozzuk létre a következő bejegyzéseket:

```
MAILTO=""

# A scanmail parancsfÆjl minden kØt percben
# fusson le
*/2 * * * * /var/spool/smtpd/bin/scanmail

# A daysumm parancsfÆjl minden nap 11:59-kor
# induljon el
59 23 * * * /var/spool/smtpd/bin/daysumm

# A sigupdate minden cs t rt k n 4:00-kor
# fusson le.
0 16  * * 4 /var/spool/smtpd/bin/sigupdate
```

Természetesen a futásidőpontokat megváltoztathatjuk úgy, hogy megfeleljenek az igényeinknek. Az, hogy a `scanmail`-t milyen sűrűn futtassuk le, főként a napi levélforgalmunktól függ. Ha a napi mennyiség tízezer levél felett van, a magam részéről az időközt két percben határoznám meg, az `smtpfwdd`-t pedig ötpercenkénti futásra állítanám be. Így nem küldünk egyszerre hatalmas levélcsomagokat a belső kiszolgálóra. Ha naponta 1000 vagy ennél kevesebb levéllel kell csak számolnunk, elég, ha a `scanmail` minden tizedik percben fut le, az `smtpfwd` pedig ötpercenként néz körül. Ne feledjük el a `MAILTO=""` kifejezést kitenni a crontab-bejegyzések elejére! Ez azért szükséges, hogy a crond a végrehajtott cron-feladatokról ne küldjön levelet az uucp-felhasználónak. Ha minden két percben egy levél érkezik, az gyorsan felgyülemlik, és az uucp-felhasználó soha nem ellenőrzi a leveleit.
Felállítottam egy Samba-megosztást is, hogy a windowsos gépemről is hozzáférhessek a ***/var/spool/smtpd*** könyvtárszerkezethez. A főként Windowst használó rendszergazdáknak ez a beállítás hasznos lehet – így nem kell mindig SSH-kapcsolatot nyitnom, valahányszor vírusra figyelmeztető levelet ellenőrzök.
A következő sorokat kell a ***/etc/smb.conf*** fájlba illeszteni:

```
[mail-gate]
       Comment = LevØltßzfal k nyvtÆrak
       Path = /var/spool/smtpd
       Valid users = nev nk
       Admin users = nev nk
       Browseable = no
       Read only = no
```

ahol a „nevünk” természetesen a Samba-felhasználói nevünket jelenti. Ami még hiányzik ahhoz, hogy tűzfalunkról az összes bejövő SMTP-kapcsolatot az új levélszűrő kiszolgálónkra irányítsuk, az, hogy tűzfalunknak IP-álcázást kell használnia. Egyszerűen adjuk ki a következő parancsot:

```
ipmasqadm portfw -a -P tcp -L tßzfal 25
➥-R cØlgØp 25
```

ahol a `tßzfal` a tűzfalunk címe, a `cØlgØp` pedig az új levélszűrő gépünk címe. Amennyiben levélszűrő tűzfalunkat közvetlenül a már meglévő tűzfalunkon szeretnénk futtatni, semmin sem kell változtatnunk.
Ha más típusú tűzfalrendszert használunk, olvassuk el a leírást, hogy megtudjuk, miképpen állíthatjuk be a kapuátirányítást. Remélhetőleg nem okoz nagy gondot, de előfordulhat, hogy kapcsolatba kell lépnünk a termék készítőjével. Ne feledjük el ezt az átirányító parancsot betenni az indító parancsfájlokba, hogy túlélje a rendszerindításokat.
Ha minden jól ment, végre működő levélvírusszűrő tűzfallal rendelkezünk. Próbáljunk meg küldeni magunknak néhány próbaüzenetet valamelyik ingyenes webes levelezőszolgáltatótól, hogy lássuk, minden jól működik-e. Én például adott időközönként küldök magamnak egy makróvírussal fertőzött állományt, hogy lássam, a rendszer még mindig helyesen működik-e. Sok dologgal lehetne még bővíteni ezt az alaprendszert. Különösen ígéretes a `daysumm` parancsfájl, amelyet jócskán fel lehetne még fejleszteni. Jelenleg épp egy CGI-parancsfájlon dolgozom, amely a pillanatnyi átlagokat – például az átlagos napi mennyiséget – az Interneten keresztül jelenítené meg. Természetesen ez csak egy út a több százból, ahogyan ez a rendszer levelezőrendszerünket megvédheti, anélkül, hogy befolyásolná a cég költségvetését. Ha valaki esetleg kitalálna valamilyen ügyes továbbfejlesztést a rendszerhez, kérem, tudassa velem. Nagyon szeretnék hallani róla.

Dave Jones
(davidashleyjones@hotmail.com)
három évig volt hálózati rendszergazda az alabamai Birminghamben. Amikor éppen nem a számítógép előtt ül, egy-egy szál jófajta dohányt szív el, vagy a feleségével és a lányával X-aktákat néz a tévében.

Gyanús adatforgalom felderítése

Használjunk psad-t IP Chains, illetve IP Tables szabálykészletünkhöz, hogy felfedezhessük a TCP- és UDP-kapuvizsgálatokat, és más hálózati gonoszkodásokat.

Alig fél éve végre kiadásra került Linux 2.4.0 rendszermag által a GNU/Linux hatalmas lépést tett a vállalati operációs rendszerek világa felé. Több magrész is fejlődött ugyan a 2.2.x sorozat óta, de egyik sem olyan jelentős mértékben, mint a tűzfalkód. A 2.4.x rendszermagsorozatban megjelent a Netfilter (lásd a ***Linuxvilág*** 2001. októberi számát, 27–31. oldal), amely a 2.2.x sorozat régi IP Chains tűzfalkódjának helyére lépett, és több olyan képességgel is rendelkezik, amelyre egy valódi kereskedelmi tűzfalnak szüksége lehet. Alaposságát képességei bizonyítják a leginkább: a DoS-védelem (DoS = Denial of Service, szolgáltatásmegtagadás alapú támadás, azaz tömeges kérelemmel való bombázás a kiszolgáló lebénítása céljából), illetve sűrűségkorlátozás, hálózati címátalakítás (NAT), MAC-címszűrés, és végezetül, de nem utolsó sorban, tetszés szerinti TCP-jelzőkombináción alapuló TCP-csomagszűrés és naplózás. Ezzel szemben, az IP Chainsnek szánalmas korlátja is akad (nem végez alapos vizsgálatot), ugyanis mindössze két fajta TCP-csomagot képes megkülönböztetni aszerint, hogy a csomag SYN jelzője be van-e állítva vagy sem. A Netfilter azon képessége, hogy bármilyen tetszőleges TCP jelzőkombinációt megenged, lehetővé teszi, hogy azokat a kifinomult kapuvizsgálatokat is felderíthessük, amelyeket az Nmap segítségével bárki könnyedén a gépre engedhet. A kapuvizsgálatok működésének bemutatásához és felderítésük módozatainak az ismertetéséhez először némi Nmap-háttérismeretre (lásd még a ***Linuxvilág*** 2001. májusi számát, 45–49. oldal) lesz szükségünk.

Nmap

Az Nmap a világ legismertebb, többfajta fejlett módszert egyesítő programja, amellyel a nyitott kapukat a célgépen, illetve a hálózaton egyaránt felderíthetjük. Az Nmap-rendszer a nyitott kapuk minél tökéletesebb meghatározásához kínál többek között ujjlenyomat-elemzést és TCP-sorozatszám előrejelzést (TCP sequence number prediction) is. Az Nmap által alkalmazott három legérdekesebb TCP-vizsgálati mód a FIN-, a NULL- és a XMAS-vizsgálat. A megszokott TCP-forgalomban a FIN-csomagokat (az olyan csomagokat, amelyeknél a FIN jelző be lett állítva) a TCP-kapcsolat bármely végéről küldeni lehet, jelezvén, hogy az üzenetváltásnak vége, pontosabban nincs több küldendő adat. A FIN-vizsgálat azon az elven alapul, hogyha egy „árva” FIN-csomagot (olyan FIN-csomagot, amely semmilyen létező TCP-üzenetváltásnak nem része) küldünk egy nyitott TCP-kapura, nem kapunk visszajelzést. Ha viszont egy ilyen csomagot egy zárt kapura küldünk, a hagyományos szűretlen TCP-verem várhatóan egy RST-csomaggal válaszol. Így aztán a FIN-csomagvizsgálatnál az Nmap egyszerűen minden egyes célkapura elküld egy magányos FIN-csomagot és várja, vajon visszaérkezik-e valahonnan RST-csomag. Azok a kapuk, amelyek nem válaszoltak RST-csomaggal, nyitva vannak (vagy tűzfallal szűrtek). A NULL- és a XMAS-vizsgálat is hasonló módszeren alapul, de ahelyett, hogy csak a FIN-jelzőt állítaná be, az XMAS-vizsgálat az URG- és PSH-jelzőket is beállítja, a NULL-vizsgálat pedig olyan csomagokat készít, amelyeknek egyetlen jelzőjük sincs beállítva.

A Netfilter beállítása

A biztonságos tűzfal készítésének alapgondolata az alapértelmezett tagadó hozzáállás. Eszerint, amely adatforgalom nincs kifejezetten engedélyezve, azt a tűzfalnak meg kell tagadnia vagy el kell utasítania.
Tűzfalak esetében három kifejezést használunk, amelyek ugyan rokon értelműek, de a tűzfalak esetében más és mást jelentenek:
REJECT – elutasít (ilyenkor a tűzfal egy „elutasítva” üzenetet küld vissza),
DENY – megtagad (a tűzfal kidobja a csomagot, nincs válasz),
DROP – elvet, eldob (a tűzfal kidobja a csomagot, nincs válasz).
Továbbá a tűzfalat úgy kell beállítani, hogy minden jogosulatlan csomagot egy naplófájlba naplózzon, így azt később a rendszergazda vagy a `psad`-hoz hasonlóan önműködő naplófájlfigyelő rendszer elemezheti.
Ha valakit bővebben érdekel, hogyan tudná az IP Tablest a rendszerén beindítani, melegen ajánlom ***Rusty Russell*** Netfilter HOWTO-ját a ➲ http://netfilter.samba.org-on.

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot source   destination
ACCEPT     all  anywhere anywhere state
           RELATED,ESTABLISHED
ACCEPT     TCP  anywhere anywhere TCP dpt:ssh
           flags:FIN,SYN,RST,PSH,ACK,URG/SYN
ACCEPT     TCP  anywhere anywhere TCP dpt:www
           flags:FIN, YN,RST,PSH,ACK,URG/SYN
LOG        TCP  anywhere anywhere LOG level
           warning prefix 'DENY ·
DROP       TCP  anywhere anywhere
```

Kapuvizsgálatok megállítása

Vajon megállíthatja-e egy ilyen IP Tables-szabályzat a kapuvizsgálatokat? Először minden TCP-csomagot (a TCP-jelzőtől függetlenül), amely a 80-as vagy a 22-es kaputól eltérő számú kapura érkezne, a négyes számú szabály szerint naplózzunk, majd az ötös számú szerint dobjuk el. Ez máris minden zajos vizsgálatnak ellátja a baját, amely a létező 65 535 TCP-kapu közül nem pont a 80-as vagy a 22-es kaput célozza. No de mi történjen azokkal a vizsgálatokkal, amelyek a 80-as vagy a 22-es kaput célozzák? Ez a vizsgálat típusától függ. A hármas és négyes szabályok elfogadják azokat a csomagokat, amelyeknek be van állítva a SYN-jelzőjük, de csak akkor, ha minden más jelző törölve van, így azok a vizsgálatok, amelyek kizárólag SYN-csomagokat használnak, sikeresek lesznek. Ugyanígy minden vizsgálat, amely rendes `TCP connect()` rendszerhívást használ, ahogyan azt minden rendes böngésző vagy SSH-ügyfél is tenné, szintén sikeres lesz, hiszen az egyes számú szabály engedélyezi a három TCP-kézfogás végrehajtását.
Az Nmap mindkét vizsgálati módszert támogatja a `-sS` (félig

1. lista. Tűzfalüzenetek beolvasása kmsgsd-vel

```
open LOG, ">> /var/log/psad/fwdata" or die
"$!\n";

while (1) {
  open FIFO, "< /var/log/psadfifo" or die "$!\n";
  $service = <FIFO>; # ne lass tsuk a chomp-pal
  if (($service =~ /Packet\slog/ || $service =~
/IN.+?OUT.+?MAC/) && $service =~
/DROP|REJECT|DENY/) {
    # a tiltott/eldobott csomag napl zÆsa
    # a fwdata fÆjlban
    my $old_fh = select LOG;
    $| = 1;  # a LOG-ba helyezz k a pufferelt
             # kimenetet
      print "$service";
      select $old_fh;
   }
}
```

3. lista Webkiszolgálóhoz csatlakozó TCP-folyamat IP Tables-üzenetei

```
Aug  4 12:16:56 myserver kernel: IN=eth1 OUT=
MAC=00:a0:cc:e2:1f:f2:00:20:78:1c:60:58:08:00
SRC=192.168.10.10 DST=10.10.10.50 LEN=60 TOS=0x00
PREC=0x00 TTL=64 ID=25415 DF PROTO=TCP SPT=2591
DPT=80 WINDOW=32120 RES=0x00 SYN URGP=0

Aug  4 12:16:56 myserver kernel: IN= OUT=eth1
SRC=10.10.10.50 DST=192.168.10.10 LEN=60 TOS=0x00
PREC=0x00 TTL=64 ID=0 DF PROTO=TCP SPT=80
DPT=2591 WINDOW=5792 RES=0x00 ACK SYN URGP=0

Aug  4 12:16:56 myserver kernel: IN=eth1 OUT=
MAC=00:a0:cc:e2:1f:f2:00:20:78:1c:60:58:08:00
SRC=192.168.10.10 DST=10.10.10.50 LEN=52 TOS=0x00
PREC=0x00 TTL=64 ID=25416 DF PROTO=TCP
SPT=2591 DPT=80 WINDOW=32120 RES=0x00 ACK URGP=0
```

nyílt vagy SYN-vizsgálat) és a `-sT` (TCP `connect()` vizsgálat) parancssori kapcsolókkal. Bármely más vizsgálati módszert, amely nem az ezekre a kapukra irányuló szabályos TCP-forgalmon alapul, megakadályozunk és naplózunk. Ide értendő a FIN-, XMAS- és NULL-vizsgálat, amit a korábbi Nmap-részben említettünk, illetve a SYN/FIN- és az ACK-vizsgálat is.
Most ugyan végre bizonyosak lehetünk abban, hogy az IP Tables tűzfalunk képes megállítani a kapuvizsgálatokat, de ne üldögéljünk boldogan a babérjainkon! Egyáltalán nem elég csak megállítani a vizsgálatokat, a lehető legmegbízhatóbban azonosítanunk is kell őket, hiszen a kapuvizsgálat gyakran egy sokkal komolyabb támadásnak lehet az előjele.

A psad bemutatása

A Port Scan Attack Detector (`psad`) Perl nyelven íródott program, amelyet arra terveztek, hogy a szigorú IP Chains, illetve IP Tables szabálykészletek használatával TCP- és UDP-kapuvizsgálatokat derítsen fel. Lehetőségünk nyílik néhány előre beállítható veszélyességi szint (természetesen rendelkezésre áll néhány célravezető beépített szint is) létrehozására, kérhetünk figyelmeztetést levélben is, igény szerint önműködően letilthatjuk a támadó IP-címet az IP Chains, illetve IP Tables tűzfalszabályok dinamikus átállításával. Ezenkívül bőbeszédű riasztásokat kaphatunk, amelyek tartalmazzák a forrást, a célt, a vizsgálat alá vont kaputartományt, a kezdés és a befejezés időpontját, és a `dns` és `whois`-keresések eredményét. IP Tables tűzfal használata esetén a `psad` képes kihasználni a továbbfejlesztett naplózási képességet, és fel tudja deríteni az olyan különösen gyanús vizsgálatokat, mint amilyen a FIN, XMAS vagy a NULL, azaz amelyeket valaki az Nmap segítségével könnyen a gép ellen fordíthat. Továbbá a `psad` több, a ***Snort*** behatolásfelderítő rendszerhez tartozó TCP- és UDP-aláírást tartalmaz, amelyek alapján fényt deríthet a vizsgálatokra, illetve a különféle hátsó ajtókra (backdoor) – például az EvilFTP, GirlFriend, SubSeven – és a DdoS-eszközök (mstream, shaft) adatforgalmára. A `psad` a GNU Public License alá tartozó ingyenes program és a ➲ http://www.cipherdyne.com honlapról tölthető le.
A `psad` alapfeladata, hogy a segítségével értelmezzük az IP Chains vagy IP Tables tűzfal által készített naplóüzeneteket, és felderíthessük a gyanús hálózati forgalmat. A feladat kivitelezéséhez a `psad`-nak hatékony módszerre van szüksége, amivel a tűzfal által a rendszernaplóba írt üzenetekből a számára fontos adatokat kigyűjtheti. A `psad` ezért telepítéskor egy `psadfifo` nevű nevesített csővezetéket (named pipe) készít a */var/log/* könyvtárban, és beállítja a rendszernaplódémont (`syslogd`), hogy a ***kern.info*** üzeneteket ebbe a vezetékbe írja. A syslog nyelvezetében a ***kern*** szolgáltatás által jelentett IP Chains és IP Tables naplóüzenetek ***info*** naplószinten jelennek meg. A `psad` által végzett feladatok oroszlánrészét két külön démon, a `kmsgsd` és a `psad` végzi.

kmsgsd

A `kmsgsd` démon viszonylag egyszerű felépítésű. Feladata mindössze annyiból áll, hogy megnyitja a `psadfifo` vezetéket, majd minden olyan ***kern.info*** üzenetet beolvas, amely arra utalhat, hogy az IP Chains, illetve IP Tables egy csomagot elvetett vagy elutasított, végül minden ilyen üzenetet a */var/log/psad/fwdata* `psad` adatfájlba ír. Mivel a `kmsgsd`-be épített `regex` (szabványos kifejezés-kereső) csak az olyan csomagokat keresi ki, amelyeket elvetettek vagy elutasítottak, az ***fwdata*** fájl már megszűrt adatfolyam lesz, amely csak olyan adatot tartalmaz, ami biztonsági szempontból fontos lehet. Ez az adathalmaz azonban csak annyira lehet teljes és beszédes, amennyire a tűzfal naplózza őket, ezért van tehát szükség a lehető legszigorúbb tűzfalszabályokra.
Az IP Tables nem támogatja a naplózás lehetőségét semmilyen szabályhoz, amely csomagokat vet vagy utasít el. Ezt a gondot azonban könnyen áthidalhatjuk, ha az elvetést végző szabály elé egy `--log-prefix` kapcsolót tartalmazó naplózási szabályt teszünk. Ezt figyelhetjük meg a `simplefirewall.sh` negyedik szabályában. Az *1. listában* látható `kmsgsd`-kódrészlet azt mutatja be, hogyan olvas a `psadfifo` tűzfalüzeneteket nevesített csővezetékről, és miképpen lehet szabványos kifejezések segítségével az IP Chains vagy az IP Tables által elvetett csomagok üzeneteit kiszűrni.

psad

Miután a ***fwdata fájl*** megtelt az elutasított csomagokkal, kiértékelésük a psad démon feladata. Ez alapján dönthető el, hogy az adott csomag vajon része-e kapuvizsgálatnak vagy más gyanús hálózati forgalomnak. A psad ezt úgy végzi el, hogy meghatározott időnként megvizsgálja, vannak-e új sorok az ***fwdata*** fájlban, hiszen az azt jelenti, hogy a tűzfal nemrégiben csomagokat utasított el. A kapuvizsgálatok egytől ötig terjedő veszélyszinthez vannak rendelve attól függően, hogy a tűzfal hány csomagot utasított el egy megadott időn belül. Ezenkívül a vizsgálatot valamely veszélyszinthez aszerint is hozzá lehet rendelni, hogy tartalmaz-e valamit a ***psad_signatures*** fájlban található veszélyes minták közül. Ilyen mintára példa az ***NMAP Fingerprint attempt*** (azoknál a csomagoknál, ahol az URG-, PSH-, SYN- és FIN- jelzők be vannak állítva) és a ***DDoS – mstream client to handler*** (ahol a SYN-csomag a 15 104-es kaput célozza). Amikor a kapuvizsgálat elér egy meghatározott veszélyszintet, a rendszer a következő adatokat tartalmazó levelet küldi:

- a vizsgálat forrásának IP-címe,
- a cél IP-címe,
- a legutóbbi vizsgálati szakaszban végigvizsgált kapuk tartománya (TCP vagy UDP),
- a vizsgálat kezdete óta megvizsgált kapuk teljes tartománya (TCP és UDP),
- kezdési és befejezési időpont,
- a psad által hozzárendelt veszélyszint,
- a fordított DNS-adat (reverse dns information),
- a vizsgálat által használt TCP-jelzők (azokkal a megfelelő Nmap parancssori kapcsolókkal együtt, amelyek ilyen vizsgálatot eredményeznének),
- a whois-adat.

A psad ahelyett, hogy kérelmével a Linux-terjesztéseken alapértelmezés szerint telepített whois ügyfélhez fordulna, inkább a ***Marco d'Itri*** által írt kitűnő whois ügyfelet használja. Ez az ügyfél ugyanis rendelkezik azzal a képességgel, hogy csaknem minden vizsgálódó forrás IP-cím esetén mindig a megfelelő whois-adatbázist kérdezi le. A psad arra is képes, hogy az IP Chains, illetve az IP Tables szabálykészletet átállítsa: minden olyan IP-címet kitilt, amely elér egy bizonyos veszélyességi szintet. Ez a képesség alapértelmezés szerint nincs bekapcsolva, mivel sok rendszergazda nem szeretné, hogy hálózatszerte számos rendszergazda legyen képes a tűzfalat befolyásolni úgy, az elérést az adott weblaphoz kitiltva e gépek nevében (IP-cím átírással) veszélyes csomagokat küld a tűzfalhoz. Más rendszergazdák viszont valamilyen magas határértéket szeretnek a kapuvizsgálat vagy más veszélyes forgalom esetére beállítani, és a psad-ra bízzák az önműködő kitiltását. Figyeljük meg, hogy bármilyen adat, amit a psad megvizsgálhat, a tűzfal által már eleve tiltva van. Ha a vizsgálat elér egy megfelelően magas szintet, az önműködő kitiltó képesség a támadó forráscímről érkező valamennyi forgalmat letiltja, mivel a rendszergazda valószínűleg nem szeretné, ha egy ilyen IP-címről bármiféle csomag érkezne a helyi hálózatra, legyen az törvényes vagy törvénytelen.
A psad jelzőrendszerrel is rendelkezik, így ha a psad-nak USR1 jelzést küldünk, meghívja a Data::Dumper-t és a %Scan szerkezet-tartalmát a */var/log/psad/scan_hash.$$* fájlba írja, ahol a *$$* az adott psad-folyamat programazonosítóját jelenti. A %Scan szerkezet a psad központi adatszerkezete, az összes vizsgálati adatot tartalmazza, így kiíratása a vizsgált adatok rögzítésére és hibakeresésre egyaránt jó, kiterjesztve általa a psad képességkészletét. Jelenleg a psad is még fejlesztés alatt áll, és jó pár dolog szerepel a tervlistáján – ugyanakkor tevékenyen fejlesztik, és rövid időközönként frissül.

IP Chains vagy IP Tables napló?

Az IP Chains és IP Tables tűzfalak közötti különbség bemutatásához először is hasonlítsuk össze az Nmap XMAS-vizsgálat által készült naplókat.

IP Chains

A 2. listában (24. CD Magazin/Gyanus könyvtárban) látható IP Chains-üzenetek a 79 és 81 közötti TCP-kapuk Nmap XMAS-vizsgálatával készültek. Emlékezzünk, az XMAS-vizsgálat beállítja a FIN-, URG- és PSH-jelzőket. Előbb az nmap parancs és kimenete látható, ezt követi a megfelelő IP Chains-kimenet. Figyeljük meg, hogy az IP Chains egyáltalán nem említi, mely TCP-jelzők voltak beállítva.

IP Tables

Most ugyanezt a Nmap-vizsgálatot végezzük el (az nmap parancssor és kimenet azonos a fenti IP Chains példában találhatóval, így azt nem ismételjük meg), és megjelenítettük a megfelelő IP Tables-kimenetet (*3. lista*). Ebben az esetben a vizsgálatnál használt csomagokban tisztán láthatjuk a beállított FIN-, URG- és PSH-jelzőket.

Michael Rash
(mbr@cipherdyne) vezető biztonsági mérnökként dolgozik egy ASP-nél a marylandi Annapolisban. A marylandi egyetemen alkalmazott matematikából szerzett diplomát, és 1998 óta bütyköl Linuxon.

Kapcsolódó címek

A 2.2.x Linux rendszermagsorozathoz is léteznek olyan foltok, amelyek az IP Chains tűzfalkódot teljes körű TCP-jelző naplózási képességekkel ruházzák fel. Példákért nézzük át a Linuxmag irattárát 2000/12/01-től 2000-12/07-ig, amely a ➲ http://www.uwsg.indiana.edu/hypermail/linux/kernel/0012.0/index.html címen érhető el. Ebben a tárban találunk egy „[PATCH] IP Chains log will show all flags" című témát, amely a forráskód *diff*-et tartalmazza a *linux-2.2.x/net/ipv4/ip_fw.c* fájlhoz.

A legjobb hely, ahol első kézből juthatunk TCP-vel kapcsolatos adatokhoz: RFC: 793 -Transmission Control Protocol, a ➲ http://www.ibiblio.org/pub/docs/rfc/rfc793.txt címen.

A 2. és a 4. lista a ➲ 24. CD Magazin/adatforgalom könyvtárban található.

A psad tervlistájában jelenleg szereplő célkitűzések: IP Filter-támogatás a BSD-felületeken; a legfontosabb psad-összetevők újraírása C-ben a jobb hatásfok elérése érdekében; ICMP-támogatás; jobb aláírás-meghatározás, hogy több IP-, UDP-, TCP-fejlécmezőt is be lehessen venni; illetve a rendszer beépítése a Bastille Linuxba.
➲ http://www.bastille-linux.org.

A NIS és az NFS (1. rész)

A központi felhasználókezelés két segédeszköze

Két hárombetűs, bűvös rövidítés több ponton is hasonlít egymásra. Az első és legszembetűnőbb apróságon túllépve (miszerint ugyanazzal a betűvel kezdődnek), észreveheted, hogy mind a kettő mögött ugyanaz a cég áll, név szerint a Sun. Továbbá mindkettő ugyanazt a célt hivatott szolgálni: adatokat oszthatsz meg velük a hálózaton. Ha már hallottál róluk, most biztosan morogsz, amiért együtt említem őket, hiszen úgy vélheted, semmi közük egymáshoz. Igaz: a két rendszernél az állományok megosztásának módja nem ugyanolyan, viszont remélem, e kétrészes cikk végére kiderül, milyen remekül kiegészíti egymást a két rendszer. Vágjunk bele!

Mire jó a NIS?

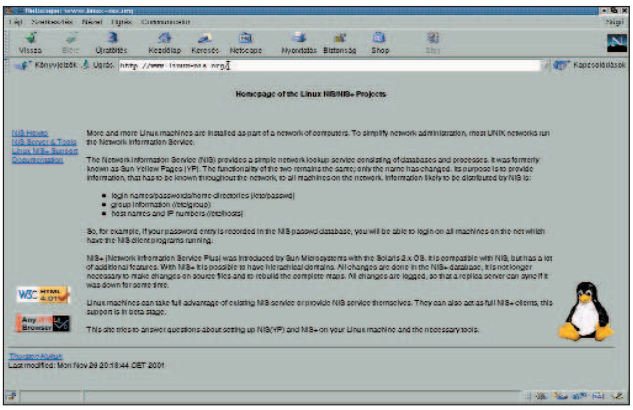
A NIS (Network Information Service) arra szolgál, hogy dbm formátumú adatbázisokat oszthass meg számítógépek között. Egy ilyen adatbázis rendkívül egyszerű felépítésű. Minden rekord egy kulcs- és egy értékpárból áll. Ezeket az adatbázisokat egyszerű szöveges állományokból készítheted a csomaghoz mellékelt segédprogrammal, a `makedbm`-mel. Ahhoz, hogy eloszlassam a témát körüllengő homályt, ismertetek egy egyszerű példát. Tegyük fel, létezik egy linuxos hálód, ahol azt szeretnéd, ha a felhasználókat központilag lehetne felügyelni. Örülnél, ha nem kellene az összes gép */etc/passwd* állományát módosítanod, amennyiben egy új felhasználót fel szeretnél venni a rendszerbe. Ebben az esetben a NIS a megoldás. Csak a központi gépen veszed fel a */etc/passwd*-be a felhasználókat, elkészítesz egy adatbázist, majd a NIS-sel megosztod. Ilyen egyszerű!

NIS, NIS+ és YP

Elnézést kérek mindenkitől a sok rövidítésért, tudom, hogy ezeket fel kell dolgozni, de egyedül ilyen „csúnya” szavakkal van tele az összes fellelhető leírás. Mivel ez a cikk nem lépheti túl a szabott keretet, kénytelen leszel hozzászokni ezekhez a „mozaikszavakhoz”. A NIS és az YP ugyanaz. Az YP (Yellow Pages – Sárga oldalak) a régebben használatos kifejezés. Mindaddig így hívták a NIS-t, amíg a British Telecom be nem jegyeztette az YP-t. A NIS+ pedig a NIS egy továbbfejlesztett változata. Minden megtalálható benne, amit a NIS-ből valaha is hiányolhattál: titkosított fájlátvitel, a faszerkezeteknek köszönhetően a nagy hálózatok rugalmas és megbízható kezelése. Akad azonban egy kis gond is a NIS+-szal: kiszolgálórésze Linux alá még csak próbaváltozatban létezik, így közel sem mondható üzembiztosnak. Másrészt a NISHOWTO is azt ajánlja, hogy a borzalmas kiszolgálóoldali beállítgatás gyötrelmeitől lehetőleg kíméljük meg magunkat, és használjunk NIS-t, ahol csak lehet.

Az RPC

Ha még nem barátkoztál meg a `portmap`-pal, itt az ideje, hiszen ez a NIS egyik alapköve. Az RPC (Remote Procedure Call) egy C-programozói könyvtár, amely lehetővé teszi, hogy egy folyamat egy függvényt egy másik számítógépen hívjon meg. Amikor az RPC-ről beszélünk, legtöbbször a Sun-féle



➲ http://www.linux-nis.org/

megvalósításra gondolunk, más néven a sunrpc-re. Amennyiben akad olyan kiszolgáló, amely távoli függvényhívást tesz lehetővé, először a `portmap`-hez fordul, és „elmondja”, hogy milyen számon és melyik kapun érhető el. Amikor az ügyfél kapcsolódik, a `portmap`-tól tudja meg, hogy az adott számú kiszolgáló éppen melyik kapun csücsül. Az egyetlen állandó kapu így a `portmap`-é, ami a 111-es, az összes RPC-kiszolgáló kapuja változhat. Összefoglalva: a `portmap` nem tesz mást, mint hogy azokat az egyedi számokat, amelyeken a függvények elérhetők, kapuszámokra fordítja. Ez a kialakítás természetesen azt is jelenti, hogy a `portmap`-nak kell legelőször elindulnia – még mielőtt bármilyen RPC-kiszolgálót indíthatnál a gépeden. Debian alatt például futási szinttől függetlenül a rendszerindítás után önműködően elindul a `portmap`. Amennyiben a `portmap` meghal, az összes RPC-kiszolgálót újra kell indítani. Meg kell jegyeznem, hogy velem ez még sosem fordult elő, de ki tudja, mit hoz a jövő. A biztonságmániásoknak jegyezném meg, hogy a `portmap` használja a `tcp_wrapper` függvénykönyvtárat, így a */etc/hosts.allow*, illetve a */etc/hosts.deny* állományokkal kényelmesen behatárolható azoknak a gépeknek a köre, amelyeknek engedélyezett a használat (lásd még: portmap(8)).

A NIS-tartományok

Egy NIS-re épülő hálózat több tartományból áll, de legalább egy szükséges. Ezt a tartományt még véletlenül sem szabad összetéveszteni a DNS-sel. Amennyiben a hálózatodon NIS és `named` is található, tartománynévként mindenképpen valamilyen a DNS-től különböző nevet válassz, ezzel is megnehezítve az esetleges külső behatolók dolgát. A tartománynevet a `domainname` paranccsal bármikor lekérdezheted és beállíthatod (lásd később). Ebben a tartományban lennie kell legalább egy úgynevezett fő (master) kiszolgálónak, ami a megosztandó adatbázisokat tartalmazza. Ezt tükrözheti egy vagy több alkiszolgáló (slave), amelyek a fő gép adatbázisairól mindig pillanatnyi másolatot tartanak fenn, és ha nagyon leterhelt lenne, ők is kiszolgálhatják az ügyfeleket.

A kiszolgálói oldal

Az ➲ ftp.kernel.org/pub/linux/utils/net/NIS címen a NIS-hez nemcsak a kiszolgálóalkalmazásokat, hanem mindenféle segédeszközt fellelhetsz. Jelenleg kétféle kiszolgáló érhető el, az egyik az `ypserv`, a másik az `yps`. Az yps nem tesz lehetővé fő- és alkiépítést, és hosszabb ideje már nem is támogatott. Debian alatt minden NIS-eszköz (kiszolgáló és ügyfél egyaránt) egy csomagban érhető el (mondanom sem kell, a csomag csak az `ypserv`-et tartalmazza, az `yps`-t nem). Ha tehát Debiant használsz, meglepően egyszerű dolgod lesz: telepítés közben csak a használandó tartománynevet kérdezi meg, minden mást önműködően beállít. Létrehozza a */var/yp* könyvtárat, amin belül egy a tartománynévvel megegyező nevű könyvtárban létrehozza az adatbázisokat. Ezt kézzel a `/usr/lib/yp/ypinit -m` parancs kiadásával érheted el. További alkiszolgálókat később az alkiszolgálón kiadott `/usr/lib/ypinit -s f1` paranccsal hozhatsz létre, ahol a `f1` helyén értelemszerűen egy név vagy IP-cím áll. Debian alatt a telepítő elkészíti még a */var/yp* alatt a *Makefile*-t, így ha az adatbázisokat újból létre akarod hozni, csupán rendszergazdaként ki kell adnod a `make` parancsot a */var/yp*-ben. A */etc/passwd* állomány végén a NIS-re vonatkozó bejegyzéseknek egy + (plusz) vagy egy – (mínusz) jellel kell kezdődniük. Így ha el akarod érni, hogy Alice távolról ne tudjon belépni, csak annyit kell tenned, hogy a */etc/passwd* fájl végére odaírod: `-alice:::::::`. Az üres mezők a „bármit” jelentik. Egy fontos megjegyzés: szokványos esetben mind a kiszolgálónak (ypserv), mind az ügyfélnek (ypbind) létrejön egy-egy parancsállomány a */etc/init.d*-ben. A Debian viszont megfordítja a dolgot: van egy */etc/init.d/nis*, amelynek első sora a `NISSERVER=false`. Ez az ügyféloldali beállítás, ezért ezt írd át masterre. Ezután az a legjobb, ha újraindítod a számítógépedet. Amennyiben ellenőrizni kívánod, hogy az `ypserv` (amely RPC-kiszolgáló) rendesen fut-e, írd be a következőt:

```
# rpcinfo -u localhost ypserv
```

ekkor valami az alábbihoz hasonlót kell látnod:

```
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

Lásd még: ypserv(8), ypserv.conf(5), rpcinfo(8).

Az ügyféloldal

Ügyfelet még ennél is egyszerűbben telepíthetsz. Az ügyfélprogramot, az `ypbind`-et – a könnyebb hibakeresés végett – érdemes a mestergépre is feltelepíteni és elindítani rajta. Ez Debian alatt ugyancsak alapértelmezés szerinti beállítás. Az ügyfelet a fentebb említett FTP-helyen vagy a már megismert NIS-csomag részeként érheted el. Amint az ypbind-et elindítod, megpróbálja megtalálni a kiszolgálót. Fontos, hogy a `NIS domain` helyesen legyen beállítva. Ellenőrizd a `domainname` paranccsal, és amennyiben szükséges, állítsd be. Az `ypbind` a NIS-kiszolgáló nevét először a */etc/yp.conf* állományban keresi; ha itt nem találja vagy nem éri el a felsorolt kiszolgálókat, üzenetszórásba kezd. Ezzel a módszerrel eltarthat egy darabig, amíg ráakad a kiszolgálóra, így ha már mindened működik, feltétlenül állítsd be itt a NIS-kiszolgálódat. Attól kezdve, hogy a kapcsolat létrejött, folyamatos lesz a kiszolgáló és az ügyfél között. És most jöjjön néhány fontosabb ügyféloldali alkalmazás, amelyek jól jöhetnek:

ypwhich

A segítségével megtalálható egy számítógép NIS-kiszolgálója vagy egy adott adatbázishoz tartozó mesterkiszolgáló, például:

```
$ ypwhich
turorudi.home
$ ypwhich -m passwd.byname
turorudi
```

Az első lekérdezés a használt NIS-kiszolgálóra vonatkozott, itt láthatod, hogy a teljes DNS-nevet visszakaptam. A második pedig a megosztott *passwd* adatbázis mesterkiszolgálójára irányult.

ypcat

Kiíratja egy adatbázis összes kulcsához tartozó értéket. Fontos, hogy az adatbázis megadásakor beceneveket is használhatsz. A fenti példában látott `passwd.byname` helyett most elég csak `passwd`-t írni. A teljes nevekhez tartozó becenevek listáját az `ypcat -x` paranccsal kapod meg.

```
$ ypcat passwd
identd:x:100:65534::/var/run/identd:/bin/false
nobody:x:65534:65534:nobody:/home:/bin/sh
telnetd:x:101:101::/usr/lib/telnetd:/bin/false
balazs:x:1000:100::/home/balazs:/bin/bash
```

ypmatch

A megadott kulcsokhoz tartozó értékeket adja vissza az adatbázisban – itt szintén használhatsz beceneveket.

```
$ ypmatch telnetd balazs passwd
telnetd:x:101:101::/usr/lib/telnetd:/bin/false
balazs:x:1000:100::/home/balazs:/bin/bash
```

Az nsswitch.conf

A */etc/nsswitch.conf* az egyes szolgáltatások esetében azt a sorrendet tartalmazza, ahogyan az adatokat le kell kérdezni, például a

```
hosts:      files dns
```

bejegyzés azt jelenti, hogy egy névhez tartozó IP-cím keresésekor először a */etc/hosts* állományt kell megnézni, és ha a bejegyzés nem szerepel, jöhet a DNS. Számunkra most csak annyi fontos, hogy az első három sor így nézzen ki:

```
passwd:         compat
group:          compat
shadow:         compat
```

Bővebben: nsswitch.conf(5)

Összegzés

Ugyanazzal a felhasználói névvel és jelszóval most már bármelyik gépen be tudunk lépni. No, de mi lesz a saját könyvtárunkkal? Ez még most sem érhető el! A következő részben ezt taglaljuk részletesebben.

Fülöp Balázs
(xut@freemail.hu) 17 éves, 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.

Átállás

Egy kiszolgáló átállításának nehézségei

Sokszor előfordul, hogy a vállalat egyik kiszolgálója eléri azt a kort, amikor már le kell cserélni. Három éve becsülettel szolgáló belső gépünk nálunk is eljutott erre a pontra. E cikkben áttekintést kívánok adni – elsősorban nem szakmai szempontból –, hogy egy ilyen átállás miféle módon zajlik.

A helyzet egyszerű: a központi kiszolgáló elavult, egy-két alkatrésze a végkimerülés szélén áll, az operációs rendszere esetleg olyan régi, hogy már egyszerűbb nulláról újratelepíteni, mint frissíteni. A vállalat eldönti: beruház egy új gépre, a rendszergazda megkapja a feladatot: itt a pénz, holnap kérem az új kiszolgálót! Amennyiben te is ilyen helyzetbe kerültél, csendben fordulj meg, ballagj el ezért a cikkért, majd dugd a főnök orra alá. Ez ugyanis nem ilyen egyszerű! Egy kiszolgáló helyett újat beállítani csak egy része a dolognak, ilyenkor számos járulékos kérdés és gond szokott a felszínre kerülni. Nézzük csak végig a jellemzőbbeket!

Egy kicsi vagy közepes méretű vállalatnál az alábbi fejlődés a jellemző: legelőször egy-két gép van. Egy idő után a gépeket (régebben koaxos, manapság már egyértelműen csavart érpáras) hálózattal összekötik. Később az egyik gépbe kerül egy modem, megszületett az internetkapcsolat, sőt, az adott gép faxok küldésére is alkalmas. Később rájönnek, hogy a munkagépek gyakran megfáradnak, ezért szükséges egy gép, amely jóval megbízhatóbb a többinél. Négy-öt gép felett kiderül: az volna a legáldásosabb, hogy ez a gép már ne legyen munkaállomás is egyben. Később az internetkapcsolatot meg kívánják osztani, majd az első komoly pusztítás után – amit mondjuk egy levélvírus vagy betörés okoz – komolyabb védelmi rendszerek üzembe állításával próbálkoznak meg. Emellett felmerül az igény a hálózatos faxküldésre, mindenki külön levélcímet szeretne, a gépek száma lassan két-három tucatnyira nő, és egyszer csak előkerül az útválasztás kérdése. Eleddig elég volt az az egy vagy két 16 kapus elosztó, most már viszont tisztán lehet látni, hogy a gépparkot külön szakaszokra érdemes szétválasztani. Ez teljesítményben és biztonságban is megtérül.

Ezen a ponton a vezetőség már gondban lehet, amennyiben a dolgozók többsége nem számítógépen nevelkedett. Szükség van egy profi rendszergazdára, de egy rendszergazda sokat kér. Ha megbíznak egy külsős céget, a gondok csak lassabban oldódnak meg, ha felvesznek egy új alkalmazottat, nem biztos, hogy megfelel az igényeknek…

Tehát ez a kissé áttekinthetetlen állapot fogad minket. Ne gondold, kedves olvasó, hogy ilyen vállalat nincs. Sőt! A legtöbb vállalat eljut ebbe az állapotba, van, hogy gyorsabban, van, hogy lassabban. Rendelkezünk egy kusza hálózattal, amelyben mindenféle ügyfélgép szerepel, egy-kettő kiszolgálóként működik, némelyikben van faxkártya, némelyik meg is osztja szolgáltatásait, a nyomtatók szanaszét helyezkednek el a hálózaton belül, majd a kegyelemdöfés: az egyik gépen bekopog egy vírus, mondjuk a Nimda.

A markunkban ott a pénz, de még mielőtt elköltenénk, gondolkozzunk, mire is érdemes. Először is gondoljuk végig, hogy a hálózat szerkezete (topológiája) megfelelő-e. Ha például három külön részleg van a cégnél, amelyeknek egymással nincs sok közös dolguk, jobban járunk, ha külön szakaszokba helyezzük őket. Ez több dolgot is maga után von. Először is több hálókártya kell az útválasztást végző gépbe, és mindegyik szakaszhoz külön elosztót vagy jelismétlőt kell üzembe helyezni. Ebből is érdemes olyat, amely a jelenlegi terhelést félvállról veszi – tervezve a leendő növekedéssel. Elképzelhető, hogy a kábelezésnél is szükséges lesz dolgozni. Kialakítunk tehát három külön szakaszt egy-egy elosztóval (eszünkbe ne jusson a világszerte rettegett ***Gagyi*** cég termékeit használni!), és az útválasztóba legalább három hálókártyát helyezünk. Kis cégeknél az útválasztó feladatát az új kiszolgáló vígan ellátja majd.

Most nézzük végig, még milyen feladatokat lát el általában egy ilyen gép. Központi fájlkiszolgáló, természetesen Samba-alapokon. Az internetkapcsolatot kezeli (a munkagépekből a modemeket az utolsó szálig ki kell imádkozni), ezzel együtt a levelezést is. Ehhez kapcsolódóan gyorstáraz és névfeloldást is végez. Esetleg központilag tárolhatja a dolgozók profiljait is, amennyiben erre van igény. Ha a modemeket kikönyörögtük, akkor joggal követelik a hálózati faxolást, főleg kifelé, de a bejövő faxokat is könnyedén kezelhetjük. A nyomtatás külön történet, valószínűleg ezt is gatyába kell ráznunk.

Most tervezzük meg az átállást – bár hiába tervezzük meg, előre szólok:

a, menet közben derül ki a legfontosabb, amire nem is gondoltunk,
b, a dolgozók a hajukat tépik majd és a pokolra kívánnak,
c, emellett újra előkerül egy még aljasabb, gyorsabban terjedő vírus.

De azért csak tervezzünk!

A kábelek rendberakása és az elosztók bekötése után a központi gépet kell összeraknunk. Ez a rendszergazda dol-

ga, egy ilyen gépet véletlenül se bízzunk olyan emberre, akinek nincs kellő tapasztalata, és nagyon fontos, hogy először élesszük fel rajta a szolgáltatásokat, alakítsuk ki a biztonsági rendet, működjön rendesen a levelezés, a fax, a fájlkiszolgálás, a nyomtatás, az internetkapcsolat és az útválasztás. Csak a szolgáltatások üzembiztos működése után kezdjük meg az átállást (nyugi, utána is lesz elég bajunk).

Az új rendszer üzembe helyezése szinte biztos, hogy az összes munkagép átállítását magával vonja (még ha DHCP-t használunk, akkor is elképzelhető). Számoljunk elég idővel, hiszen vagy minden felhasználónak végig kell mesélnünk a történetet, vagy készítenünk kell egy leírást, amelyben részletezzük, hogy mi és miért történik. A levélírással hatnyolc ügyfélgép fölött biztos, hogy időt takarítunk meg. Ha ügyesek vagyunk és bízunk a felhasználókban, a leírásban azt is leírhatjuk, hogy miként tudják saját maguk elősegíteni az átállást, és hogy milyen lépésekben és határidőkkel történik meg a váltás, valamint hogy milyen úton jelezzék, ha különleges igényük van.

Most jön a legfárasztóbb rész: a szakaszokat egyesével átállítjuk, az ügyfeleknél költözés, majd a régi szolgáltatások tiltása az adott szakaszra. Hogy miért fontos tiltani a régi szolgáltatásokat? Hogy nehogy nagyot koppanjunk, amikor (pár nap vagy hét) múlva nagy nyugalommal ízekre szedjük a régi gépet, a dolgozók pedig dörömbölnek, hogy nem tudnak dolgozni. Miután az első kis csoport átköltözött, adjunk nekik időt, hogy minden szolgáltatást élesben is kipróbáljanak. Ez órákat, napokat, de előfordulhat, hogy heteket jelent. Miután az első szakasz átállt és gond nélkül dolgoznak az új gépen, jön a tisztogatás. Ehhez szükséged lesz egy csavarhúzóra, kötélidegzetre és a főnök előzetes meggyőzésére. Ugyanis most jön, hogy az új szakasz összes gépéből ki kell követelni a modemeket, majd mindenkinek el kell magyarázni, hogy csak a központi gépen tárolja a fontos adatokat (mert az új rendszer legalábbis tükrözött lemezekkel dolgozik), hogy ne merjen hülyeségeket telepíteni (ismerek olyat is, aki a hajlékonylemezes és a CD-meghajtót is kiszereli), hogy véletlenül se osszon meg semmit a gépből (jellemzően a C: van megosztva, jelszó nélkül), hogy milyen levelezőt használjon és így tovább. Ez az a pont, amikor a rendszergazda még a főnökség tagjait is megelőzi a körözési listákon. Ezután csendben a háttérbe vonulunk és figyelünk. Ha minden felmerült kérdést megoldottunk, túl vagyunk a nehezén. Most már csupán ugyanezt kell végigzongoráznunk az összes szakaszon.

Miután az összes gép átállt és a régi kiszolgáló elérését mindenhonnan letiltottuk, még érdemes egy ideig – a biztonság kedvéért – működőképes állapotban tartani, és később, amikor már senki sem keresi, ráérünk szétszedni.

Igyekeztem röviden összefoglalni egy átállás menetét, természetesen minden környezet más és más, lehet, hogy több kiszolgáló van, előfordulhat, hogy az új gép csak különböző feladatokat vesz át, vagy a munkagépeket egytől egyig újra kell telepíteni… a változatok száma végtelen. Egy viszont biztos, ha alaposan átgondoljuk az átállást, nagyon sok keserűségtől kíméljük meg magunkat. És még egy gondolat: vegyünk legalább egy jó könyvet, ami a biztonság alapjaival foglalkozik és az összes felhasználóval olvastassuk el! Könnyebb a dolgunk, ha nem nekünk kell ötvenszer elmagyarázni, hogy miért ne használjon a munkatárs olyan levelezőt, ami önműködően elindítja a levelekben lévő vírusokat!

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

Hazai linuxos cégek gyűjteménye

Sokszor keresnek meg minket, hogy ismerünk-e jó rendszerfelügyelő céget, tudunk-e valakit, aki vállalja egy kiszolgáló telepítését, hol lehet linuxos tanfolyamokra jelentkezni, kihez fordulhatnak ügyesbajos kérdéseikkel stb. Mivel mi magunk sem ismerjük a hazai linuxos közösség minden tagját, arra gondoltunk, hogy indítunk egy sorozatot, amelyben mindig két-három, az adott területen dolgozó céget mutatunk be – a hálózatépítőtől a rendszerfelügyelőn át a programozóig. Igen ám, de ha ilyen rendszert szeretnénk indítani, jó volna, ha a cégeket egy pontos rendszerbe tudnánk beilleszteni. E rendszer kialakításához kérem most a segítségeteket.

Mit érdemes, és mit kötelező leírni egy ilyen cégről? Természetesen fontos, hogy a cég mikor alakult, hogy mióta foglalkozik Linuxszal, milyen típusú rendszerekkel dolgozik, hol található a székhelye, milyen ügyfélkört céloz meg. Emellett szerintem rendkívül fontos, hogy az adott cégen belül hány munkatárs ért az adott feladathoz, például egy rendszerfelügyeletet vállaló cégnél hány rendszergazda dolgozik, a cég mennyire van leterhelve, vállal-e 24 órás vagy hétvégi ügyeletet, gond esetén milyen határidővel dolgoznak, és természetesen milyen árkategóriában.

Az ár nehéz kérdés. Lehetetlen megmondani, hogy valaki mennyiért telepít egy rendszert, amíg nem tudja pontosan, mik is az igények. Az sem mindegy, hogy az adott munka elvégzése után hogyan adja át a készterméket, például egy gép szállítása után a cég vállalja-e, hogy amennyiben az általa összerakott gép valamelyik alkatrésze nem (vagy nem megfelelően) működik Linux alatt, azt kicseréli? Emellett egy vezető számára az is fontos, hogy a cég referenciákat tudjon adni. Sőt, az volna a legjobb, ha néhányuk véleményét ki lehetne kérni. Szeretném, ha létre tudnék hozni egy olyan listát, amely alapján az érdeklődő könnyen ki tudna választani mondjuk egy megbízható, elfogadható áron dolgozó rendszerfelügyelettel foglalkozó céget, vagy például egy komoly tapasztalatokkal rendelkező hálózatépítő társaságot.

Mivel mi ebben csak katalizátorszerepet játszhatunk, kérek mindenkit, írja meg nekem véleményét, a gondolatait, hogy milyen cégekre kíváncsi, milyen kérdéseket tenne fel, milyen szerkezetben látná örömmel a listát. Egyúttal ez felhívás a hazai linuxos cégek és vállalkozók felé is! Várjuk olyan társaságok és vállalkozók jelentkezését, akik ki mernek állni szolgáltatásaikkal a nagyvilág elé – vállalva a megmérettetést!

Szy György

A szerkesztők és az Emacs (1. rész)

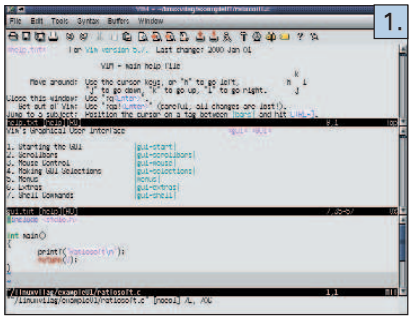
Ezt az írást nem a Linux-rendszert és a programozást jól ismerőknek szánom, hanem azoknak, akik mindkettővel most ismerkednek.

Elsőként a forráskód megírásához használható szerkesztőket mutatom be a teljesség igénye nélkül, majd az Emacs fejlesztőkörnyezettel foglalkozom hosszabban.

Szövegszerkesztők

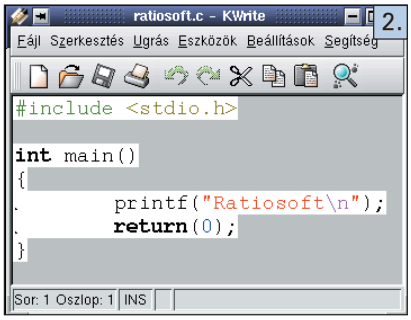
A programokat először meg kell írni, amihez kell egy nekünk tetsző egyszerű vagy kevésbé egyszerű szövegszerkesztő, amely ASCII formátumú állományt hoz létre. Ebből következik, hogy haszontalan lenne az OpenOffice nagy tudású szerkesztőjét használni, mert fölöslegesen sok helyet foglal el a memóriában, és semmiképpen sem tudjuk kihasználni a képességeit, hiszen a programírásban nem segítenek a kifinomult formázási lehetőségek, nem célszerű sokféle betűkészletet használni, és számos programozót zavar, ha változó szélességű és nagyságú betűket lát maga előtt bogarászás közben. Ne feledjük, hogy bármilyen szépre is formázzuk meg forráskódunkat, a formázó karakterek abban a pillanatban nyom nélkül eltűnnek, amikor forráskódunkat ASCII formátumban kimentjük, a fordítók pedig csak a formázatlan állományokat hajlandók elfogadni!

A fentiekből következik, hogy a merészebbek programozásra is használhatják a legendás hírű ***vi*** szerkesztőt. Erről a kezdőknek azt illik tudni, hogy a kellően tájékozottak „viáj”-nak ejtik, és hogy a `:q!` paranccsal lehet kilépni belőle. Ez nem segít mindig, ilyenkor nyomjuk meg az Esc billentyűt, és próbálkozzunk újra. Ha már fel vagyunk vértezve ennyi tudással, akkor legalább nem kell újraindítani az egész rendszert, hogy visszakaphassuk a parancssor készenléti jelét, ahogy azt egyik ismerősöm tette a `vi`-t próbálgatva. A mindenre elszántak próbálkozhatnak a `vi` többalakos változatával, az `xvi` (ejtsd: „eksz-vi-áj”) szerkesztővel, vagy az `xvi`-nál is fejlettebb `vim`-mel. A `vim` a „vi improved” (tökéletesített `vi`) szavak rövidítése. A `vi`-nak az elmúlt évtizedekben számos mutációja született, és ha telepítve vannak a gépünkön, kipróbálhatjuk őket. Írjuk be a parancssoron az `ed`, `ex`, `gex`, `vi`, `gvi`, `view`, `gview`, `vim`, `gvim`, `rvim`, `rview`, `rgvim` vagy `rgview`, `elvis`, `nvi` parancsokat és az olvasni vagy szerkeszteni kívánt állomány nevét! Ezek a parancsok többnyire ugyanazt a végrehajtható állományt indítják el, de különböző bővítményekkel vagy éppen korlátozásokkal. Például a `view` vagy a view-ra végződő parancsok használatakor a megnyitott fájlt csak olvasni lehet, a parancsok elején lévő r betű pedig a „restriction”, azaz megszorítás szóra utal. A korlátozott parancsok valamilyen

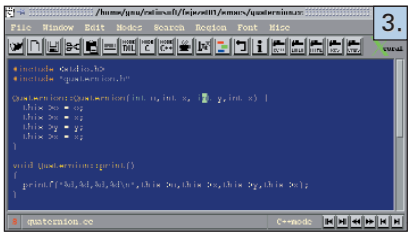

1.

módon szűkítik a `vi` lehetőségeit, például nem indíthatunk parancsokat a `vi`-ból, vagy nem függeszthetjük fel ideiglenesen a működését. A `gvim` (*1. kép*) a Windowsban megszokott menüket és ikonokat varázsolja elénk, és viszonylag jó leírással bír. Első pillanattól otthon fogjuk érezni magunkat benne, ha a telepítés után sikerül elindítanunk.

Ha nem, próbáljuk meg a *~/.gvimrc* fájlt testreszabni. A `vi` általában nem alkalmas bináris állományok szerkesztésére, ehhez inkább a `hex`, `vche`, `vche-nc`, `vche-raw`, `khexedit` vagy a `ghex` programokat használjuk. Szintén kéznél van a Midnight Commander belső fájlnézegetője, amely hexadecimális szerkesztő is egyben. A `vimtutor` program végigvezet bennünket a legfontosabb `vi`-parancsokon. Többek közt megtudhatjuk, hogy mely két üzemmód létezik a `vi`-ban, hogyan indíthatjuk el, hogyan mozgathatjuk a kurzort, hogyan szerkeszthetjük a fájlt, és miképpen léphetünk ki belőle. A `pico` szintén parancssori szövegszerkesztő, de könnyebb megtanulni, mint a `vi`-t. Ezt már az is mutatja, hogy mindjárt induláskor kiírja, hogyan lehet kilépni belőle. Megemlítem még a ***Joe*** szerkesztőt, amit ötféle üzemmódban használhatunk. Ha a `joe` helyett a `jpico` parancsot ütjük be, akkor a Joe a most említett Pico szerkesztőt utánozza,



2.

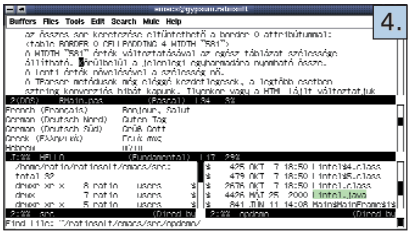

3.

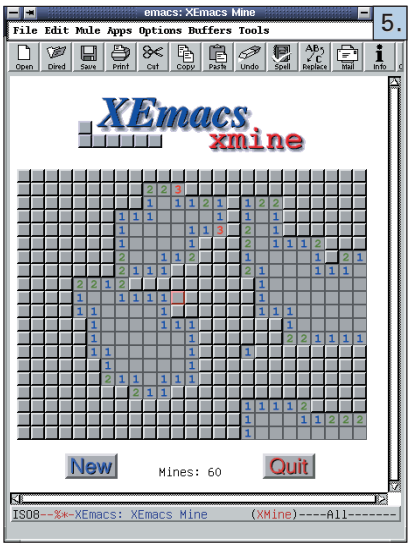
ha a `jstar` parancsot, akkor a hajdan jól ismert WordStar szövegszerkesztőt. A fenti programok mindegyike igen nagy tudású, de használatukhoz némi tapasztalatra van szükség. Az `xedit`, `kedit` és a `gedit` a Windows felől érkezők számára minden bizonnyal barátságosabbnak fognak tűnni, mint parancssori elődeik, de azt is rögtön tapasztalhatjuk, hogy ezek sem tudnak sokkal többet, mint a `notepad.exe`.

Az Emacs

A Linux-programozással foglalkozó szakkönyvek szinte kivétel nélkül a *GNU Emacs* fejlesztőkörnyezetet ajánlják. Figyeljük meg, hogy immár nem a „szerkesztő” szót használtam, hanem az IDE (Integrated Development Environment) mozaikszóra utaltam, ami magyarul összetett fejlesztőkörnyezetet jelent. Mégse gondoljunk azonban a *Linuxvilág* augusztusi számában ismertetett KDevelophoz vagy a Borland Kylixhoz hasonló RAD (Rapid Applica-

tion Development, azaz gyors alkalmazásfejlesztés) eszközökre. Az Emacs a Unix-hagyományokban gyökeredzik, nehézkes és barokkosan bonyolult. Némelyek szerint az Emacs gonosz, mások viszont három részre osztják a világban élő embereket: azokra, akik Emacsot használnak; olyanokra, akik inkább a


4.



5.

vi-t szeretik; és mindenki másra. Vannak, akik azt mondják, hogy az Emacs egy jó operációs rendszer, bár elismerik, hogy a Unixban több program van. Bonyolultságából következik, hogy hosszú időbe telik, amíg otthonosan mozoghatunk benne, és inkább azok számára ajánlom, akik naponta fogják használni, hiszen elég pár hetes kihagyás, és máris jó néhányat elfelejthetünk a számtalan billentyűkombinációból.

Az Emacs név a ***Richard M. Stallman*** által a TECO szerkesztőhöz írt „editing macros” (szerkesztőmakrók) szavak betűiből alkotott mozaikszó. Az Emacs történetét elolvashatjuk a ➲ http://www.gnu.org/philosophy/stallman-kth.html (a fájl másolata megtalálható a 24. CD Magazin/Emacs könyvtárában). A `vi`-hoz hasonlóan az Emacsnak (*4. kép*) is több változata létezik, például a ***Xemacs*** (*5. kép*), ami egy eszköztárral is rendelkezik. Az utóbbit korábban ***Lucid*** (azaz világos, érthető) ***Emacs***-nak nevezték.

Mindkét Emacs-fajta elfogadott, és gyakran együtt hivatkoznak rájuk az Emacson szóval. Ahogy az Emacs bejelentkező ablakában olvashatjuk, a GNU Emacs az egyik alkotóeleme a Linux-alapú GNU-rendszereknek, és mint ilyet egyetlen Linux-terjesztésből sem illik kihagyni. Akik GNU-programokat akarnak írni, azoknak otthonosan kell mozogniuk az Emacsban, mint ahogyan a rendszergazdáknak is ismerniük kell a `vi`-t, ami kötelezően ott van minden Unix-típusú rendszerben, és amit a legvadabb rendszerösszeomlások idején is el lehet indítani. Mindkét program elengedhetetlen kelléke a Unix-kultúrának, a Linux-életérzésnek és a GNU-mozgalomnak.

Ahogy a képeken is látjuk, az Emacs egyszerre több fájlt is meg tud nyitni – akár ugyanazon keretben több részre osztva fel a munkaterületet. Számos keretet vagy ablakot tarthatunk nyitva egyidejűleg, futtatható állományokat és képeket nézhetünk meg vele. Ha az Emacs nem támogatott képformátumot talál, akkor külső képnézőt (például ImageMagick) hív segítségül. Programozás közben intézhetjük levelezésünket, honlapokat látogathatunk meg és tölthetünk le akár PostScript formátumban is anélkül, hogy kilépnénk az Emacsból. Amikor elfáradunk, néhány beépített játékkal játszhatunk.

Az Emacsot nem a manapság igen divatos C vagy C++ nyelven írták és írják, hanem a Lisp nyelvet használják erre a feladatra, annak is egy különleges fajtáját, az Elispet, más néven Emacs Lispet. Az Elisp teljes értékű programozási környezet, amivel szöveget és fájlokat kezelhetünk, hálózati alkalmazásokat vagy új felhasználói felületeket építhetünk fel az Emacson belüli használatra. Az érdeklődők kipróbálhatják még a Jonathan Saját Emacs Változata (Jonathan's Own Version of Emacs) programot vagy annak menüsített, XJove nevű kiadását, ha beírják a `jove` vagy `xjove` parancsokat. Az `uemacs` vagy más néven MicroEmacs kis teljesítményű gépekre készült, és a 4.0-s változathoz maga ***Linus Torvalds*** írt javításokat. Utóbbit az `em` paranccsal indíthatjuk.

Az Emacs logikája

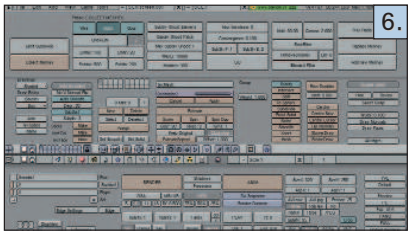
Talán igazságtalanul bántam az Emacscsal, amikor azt mondtam róla, hogy nehézkes és bonyolult, hiszen már rövid használat után beláthatjuk, hogy a fejlesztők egységes és igen egyszerű elvek alapján építették fel a programot. Ezeket az elveket így foglalhatnánk össze:

Ha a program használata közben új szempontok merülnek fel, amik megkönnyíthetik a mindennapi munkát, akkor meg kell valósítani azokat. Az új feladatok ellátásához egy vagy több Elisp-függvényt kell megírni, és ezeket a függvényeket a felhasználók számára hozzáférhetővé kell tenni, azaz meg kell adni számukra a lehetőséget, hogy programozás nélkül meghívhassák őket.

Mivel Richard Stallman és társai Unix-környezetben nevelkedtek, vérükké vált az a szemlélet, hogy egy program csak egy dolgot csináljon, de azt felettébb jól. Követelmény volt a Unixban az is, hogy ezek a programok tetszés szerint összefűzhetők legyenek. S valóban, az Emacsban ott van a mini átmeneti tár, ami lényegében egy miniatűr héj, ahová parancsokat gépelhetünk be. A mini átmeneti tárhoz hasonló parancssori beviteli eszközök eltűntek a mai szövegszerkesztőkből, ha egyáltalán voltak bennük, de a táblázatkezelőkben a szerkesztőlécek még most is megtalálhatók, annak ellenére, hogy az adatokat közvetlenül a cellákba is beírhatnánk. A Unix-követelmények szerint az Emacs-parancsok szintén kötegelhetők, és rendszerint csak egy dolgot tesznek, de azt felettébb jól, hiszen a közel két évtizedes fejlesztés következtében az Emacs igen jól átgondolt és hibamentes lett. Azt is tudjuk, hogy a programozó nem szeret sokat gépelni, annak ellenére, hogy a billentyűzetet mesteri módon tudja kezelni. Magától adódott tehát a felismerés, hogy az egyre szaporodó parancsokat billentyűkhöz és billentyűkombinációkhoz kössék. Tették ezt annál is inkább, mert majd húsz évvel ezelőtt még nem voltak ismertek a grafikus képernyők és a mutatóeszközök. A végeredmény az lett, hogy mára igen nagy számú Emacs-parancs és -változó áll a felhasználók rendelkezésére, akik így igen sok feladatot tudnak egyszerűen és hatékonyan megoldani. Cserébe viszont a kezdőknek viszonylag hosszú ideig kell ismerkedniük az Emacs-környezettel, és nemcsak a számtalan parancs nevét kell megtanulniuk, de a hozzájuk tartozó billentyűkombinációkat is készségszinten el kell sajátítaniuk.

Gondoljuk csak végig, hogy másképp van-e ez a többi, felhasználóbarátnak tartott alkalmazásban! Vajon a grafikus felületek kitalálói jobb receptet kínálnak a fenti dilemma orvoslására? Hatásos megoldás lehet, ha csökkentjük a felkínált parancsok számát – áttekinthetőbbé és könnyebben megtanulhatóvá téve a programot. Az ilyen lebutított változatok kielégítők lehetnek az átlagfelhasz-

nálók, de nem a szakemberek számára. Megpróbálkozhatunk a „szolgáltatáselrejtés” alkalmazásával, amit én a Microsoft Word 2000-ben láttam először.
A szakembereknek szánt programcsomagok viszont valósággal kérkednek az elérhető szolgáltatások százaival. Vessünk például egy pillantást a ***Blender*** nevű 3D-modellezőre, amely egyszerre több tucat nyomógombot rak ki a képernyőre, és ezek a gombok az üzemmódtól függően állandóan változnak, mindig újabbak bukkannak elő!


6.

Hiába grafikusak ezek a felületek, igazából nem lehet őket felhasználóbarátnak nevezni, hiszen előtanulmányok nélkül a zöldfülűek semmit sem tudnak kezdeni velük. Az ilyen programokat azonban az irodai programcsomagokkal ellentétben nem akarják mindenkinek eladni, beleértve az olvasni tudó kisdedeket és a homályosodó szemű aggastyánokat. A szakembereknek fejlesztő programozók számára tehát nem az a kérdés, hogy megmutassák-e a programban rejlő lehetőségeket, hanem az, hogy miképpen tegyék meg. Általánosan elfogadott módszer, hogy a szolgáltatásokat ilyen-olyan szempontok szerint csoportosítják, majd menükre, gyorsbillentyűkre, ikonsorokra vagy nyomógombokra fűzik fel őket. A párbeszédablakos megjelenítés kényelmes és felhasználóbarát, ha ritkán használt beállításokról van szó, de igen hátráltatja a munkát, ha gyakorta ismétlődő parancsok előhívására használjuk. Gondoljunk bele például, hogy a kijelölt szövegrész kivágásakor mennyire időigényes mozgássor az F10 funkcióbillentyű megnyomása, majd a kivágás menütétel kiválasztása a NYÍL billentyűkkel, és végezetül az ENTER leütése. Gyorsabb, ha a kivágás ikont nyomjuk meg gépeléskor, de még ilyenkor is oda kell vinnünk a kezünket az oldalt lévő egérhez, ami miatt meg kell szakítanunk a szövegbevitelt. Gépeléskor tehát a gyorsbillentyűk használata adja a legjobb eredményt, hiszen például az Emacsban elég megnyomni a CTRL-W billentyűkombinációt a kijelölt szövegrész kivágására, és az adatbevitel máris folytatható anélkül, hogy a kezünket egy pillanatra is arrébb kellett volna mozdítanunk. Mivel az Emacs fejlesztői tisztában voltak ezzel az ergonómiai törvényszerűséggel, nem nagyon törték magukat, hogy más, kevésbé hatékony, de mások által felhasználóbarátnak mondott megoldásokat keressenek. Érdekes módon az XEmacs sem törekszik erre, pedig azzal a céllal hozták létre, hogy könnyebben kezelhető legyen. Elég, ha egy pillantást vetünk rá, és rögtön látjuk, hogy a fejlesztők megelégedtek néhány ikonnal (összesen 15-tel), amelyeknek a felét én személy szerint fölöslegesnek tartom a mindennapi munka szempontjából. Miért kell például az Info ikont kirakni a szemünk elé, amikor nem tesz egyebet, mint megjeleníti a Súgót? Ezzel szemben a Microsoft Word szövegszerkesztőben nem 15 ikont, hanem 15 ikonsort találhatunk!
Az XEmacsban inkább a menük burjánzanak. Mégsem állítanám, hogy az XEmacs fejlesztői lusták lettek volna, amikor kispórolták az ikonokat. Inkább azt feltételezem, hogy nem látták értelmét annak, hogy ikonokkal vagy nyomógombokkal zsúfolják tele az új Emacs-változatot, hiszen az a programozói réteg, amelyik az Emacs-környezetet használja, nem igényel ilyen változtatásokat. A fejlesztők tehát a programozás és a leíráskészítés szempontjából a leghatékonyabb megoldást választották, azaz a gyorsbillentyűkkel való parancshívást, még akkor is, ha ez a kezdők számára első pillantásra elrettentőnek tűnik. Nincs királyi út! Legfeljebb az Emacs.

Az Emacs és az Xemacs

A kétfajta Emacs között nincs nagy különbség, de hamar észrevehetjük, hogy nem teljesen egyforma a kettő. Az XEmacsban hiába kerestem a szöveges állományokat átalakítás nélkül beolvasó `find-file-literally` parancsot, csak az Emacsban találtam meg. Bizonyos billentyűkombinációk is másként működnek a két változatban, de összességében nem nagyok az eltérések. Tanulás közben célszerű mindkettőt kipróbálni, és végül annál maradni, amelyik jobban tetszik nekünk.
Manapság a legtöbb alkalmazást valamilyen grafikus felhasználói felületre írják, ezért felmerülhet a kérdés, hogy van-e értelme az X Window nélkül valamelyik terminálról indítani az Emacsot, hiszen mindegyik ablakkezelőben van terminálemulátor, és abban már futhat az Emacs. De mi tehet az a programozó, aki éppenséggel egy ablakkezelő megírásába fog bele? Neki valószínűleg a parancssorról kényelmes tesztelnie új programját, hiszen éppen fejlesztés alatt álló ablakkezelője feltehetően még nem igazán használható. Ilyenkor csak valamelyik parancssori szerkesztőt futtathatja.
Ha nincs egerünk, akkor a terminálon indított Emacs még akkor is új élményt fog jelenteni számunkra, ha már jártasak vagyunk a terminálemulátorban futtatott Emacs használatában. Megszoktuk már, hogy a legtöbb alkalmazásban az F10 nyitja le a menüt, és nincs ez másként az Emacsban sem. A parancssorra tévedt felhasználó azonban meglepetten tapasztalja, hogy hiába nyomkodja az F10 funkcióbillentyűt, semmi sem változik a menüsorban. Minél feszültebben figyel azonban a képernyő felső részére, annál kevesebb esélye lesz arra, hogy észrevegye, mi történik az alsó fertályban. Az Emacs ugyanis a képernyő alján, a mini átmeneti tárban (pontosabban a visszhangterületen) írja ki a menüket, és ott is kell választanunk a megjelenő menütételek a balra vagy jobbra nyilak közül, majd az ENTER megnyomásával.
Hasonló furcsaságokkal bármikor találkozhatunk az Emacsnál, de ha már megismertük, többé nem fognak meglepődni.

Utószó

Nyilvánvaló, hogy a vi vagy az Emacs más, mint amit a többi operációs rendszerekben megszoktunk, de az a tény, hogy furcsa és szokatlan, még nem lehet a minősítés alapja. A szövegben szereplő kódbetűs szavak nemcsak a programok nevére utalnak, hanem megegyeznek a héjba beírandó programindító parancsokkal, a nagybetűvel kezdődő szavak viszont csak a programok neveit jelölik.
A következő részben röviden ismertetem az Emacs használatát.

Szaló István
(ratiosoft@freemail.hu)
tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Több írása megjelent már a hazai számítástechnikai lapokban. Ha néha feláll számítógépe mellől, rendszerint művészettörténész feleségével és kisiskolás lányával „találja szemben” magát.

Fordítsunk rendszermagot!

A nyílt operációs rendszerek egyik legnagyobb előnye, hogy a rendszermagot a felhasználó is bármikor újrafordíthatja, ezáltal csökkentheti a méretét és gyorsíthatja a rendszer futását.

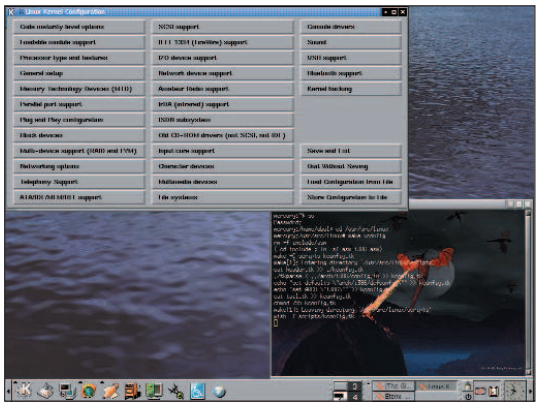
A rendszermag újrafordítása a Linux világában ugyan hétköznapi műveletnek számít, mégis számos nehézséget okozhat a kezdő felhasználók körében. Cikkünk nekik próbál segítséget nyújtani.

Bármely operációs rendszer legfontosabb része a rendszermag (kernel), amelynek legfontosabb feladata a felhasználói programok és a gépünk közötti kapcsolattartás biztosítása. Ezzel azonban még nem ért véget a tevékenységi köre, ugyanis szintén a rendszermag felelőssége a folyamatok (process) futásának felügyelete, továbbá a különböző biztonsági szabályok betartatása (nem engedi, hogy egy futó alkalmazás egy másik program által használt memóriaterületre írjon stb.).

A nyílt forráskódú rendszereknél a rendszermag forráskódjához is bárki hozzáférhet, ezért a felhasználók saját kezűleg fordíthatják újra, ha akarják. Milyen előnyökkel járhat a rendszermag újrafordítása? Először is várhatóan gyorsabb lesz, mint az előre lefordított „gyári" rendszermag, ugyanis a fordítóprogram a mi processzorunkhoz hangolja. Hasznos az is, hogy a fordítás előtt a felhasználónak lehetősége nyílik kiválasztani, mely összetevők szerepeljenek, illetve ne szerepeljenek a kész rendszermagban. Amennyiben például nincs és várhatóan nem is lesz semmiféle SCSI-eszközünk, a lefordított rendszermagból a teljes SCSI-támogatást ki lehet hagyni. Ezáltal jelentős méretbeli csökkenést érhetünk el. A rendszermag újrafordításával tehát elkészíthetjük a kifejezetten saját gépünkhöz illeszkedő rendszermagot.

A Windowshoz vagy OS/2-höz szokott felhasználók számára minden bizonnyal furcsán hangozhat, hogy a nyílt forrású operációs rendszerek világában a rendszermag újrafordítása teljesen hétköznapi műveletnek számít. Maga a fordítás nem nevezhető bonyolult műveletnek, sarokpontja egyedül az összetevők kiválasztása, amit azonban részletesebben is be fogunk mutatni. A rendszermag fordítása egyébként semmiféle programozói előképzettséget nem igényel.

Kezdjünk is neki! Legelső feladatunk magának a rendszermag forráskódjának a beszerzése lesz. A forrást Linuxunk telepítő CD-jén is megtalálhatjuk, de a legújabbat mindig fellelhetjük a CD-mellékleten, vagy letölthetjük az ➲ ftp://ftp.kernel.org-ról, illetve annak magyar tükréről, az ➲ ftp://ftp.hu.kernel.org címről. Jelen pillanatban a 2.2.x-es és a 2.4.x-es sorozatot párhuzamosan fejlesztik, e cikk írásának pillanatában az előzőből a 19-es, az utóbbiból pedig a 14-es a legfrissebb. Itt szeretnénk felhívni a figyelmet, hogyha régebbi Linux-változattal rendelkezünk, amely még a 2.2-es sorozatra épül, és át szeretnénk térni a 2.4-es rendszermagra, akkor bizonyos csomagokat frissítenünk kell. Ilyen például a telefonos internetelléréshez szükséges PPP-démon vagy a modulokat kezelő `modutils` (lásd később). E csomagok listáját a rendszermag leírásában találhatjuk meg.



A frissítéseket Linux-változatunk hivatalos honlapjáról is letölthetjük.

A magforrás becsomagolva körülbelül 22 MB, a lefordításához azonban körülbelül 60–80 MB szabad tárhelyre lesz szükségünk. Az Interneten elérhető rendszermagforrásokat általában gzippel vagy bzip2-vel tömörítik: az első esetben a kiterjesztés tar.gz, a másodikban pedig tar.bz2.

A forrást másoljuk a */usr/src* könyvtárba. Ha gzippel tömörítették, akkor a `gzip -d kernel-xxx.tar.gz` paranccsal csomagolhatjuk ki, bzip2 esetében a `bunzip2 kernel-xxx.tar.bz2` utasítást használhatjuk. (az xxx a rendszermagnak megfelelő változatszám) Ezt követően a létrejött .tar kiterjesztésű állomány kicsomagolásához adjuk ki a `tar -xvf kernel-xxx.tar` parancsot. Ez a parancs létrehoz egy Linux nevű könyvtárat. Ha már található ilyen könyvtár az adott helyen, előtte nevezzük át, például ***Linux-old***-ra (`mv Linux Linux-old`). Amennyiben mindent jól csináltunk, a forrást magát az újonnan létrejött ***linux*** könyvtárban találhatjuk meg.

A tar-állományra a továbbiakban nem lesz szükségünk, tehát nyugodtan letörölhetjük, ne foglalja fölöslegesen a helyet.

Most lépjünk be a magforrást tartalmazó ***linux*** könyvtárba! A rendszermag fordításának első és egyben legnehezebb lépése a kész rendszermag összetevőinek a kiválasztása. Mielőtt buzgón hozzálátnánk, meg kell beszélnünk egy fontos dolgot.

A Linux-rendszermag fontos tulajdonsága, hogy modularizált. Ez azt jelenti, hogy bizonyos alkatrészek és szolgáltatások támogatását nem feltétlenül kell közvetlenül a rendszermagba fordítanunk, hanem lehetőségünk nyílik rá, hogy modulokat készítsünk. Ezeket a modulokat bármikor kedvünkre betölthetjük a memóriába, illetve amennyiben feleslegessé váltak, el is távolíthatjuk őket onnan. Miért jó ez nekünk? Bizonyára akadnak olyan eszközeink, amelyeket nem használunk állandóan, a legjobb példa erre talán a hangkártya. A hangkártya szolgáltatásaira a rendszer mindennapi használatában nincs szükségünk, csupán abban az esetben, ha zenét akarunk hallgatni vagy lazítás-

képp valamely linuxos játékkal szeretnénk egy kicsit játszani. Ha jobban meggondoljuk, belátjuk, hogy teljesen felesleges a hangkártyatámogatást a rendszermagban „tárolnunk”, sokkal célszerűbb, ha modulba tesszük. Ennek köszönhetően a „hangos eszköz”-támogatás csak akkor kerül a memóriába, amikor hangkártyánkat ténylegesen „dalra fakasztjuk”.
Érdemes minél jobban kihasználnunk a Linux-rendszermag eme előnyét.
A bevált szokás az, hogy csak azokat az elemeket fordítjuk közvetlenül a rendszermagba, amelyekre a rendszer elindításához feltétlenül szükség van. Nem érdemes modulba tenni azoknak az egységeknek a támogatását, amelyeket a rendszer futása közben állandóan használunk: ilyen lehet például egy hálózati kiszolgáló esetében a hálókártya. Amelyik támogatást csak lehet, mind „dobáljuk” modulba.
A modulokat egyébként a `modprobe` parancs segítségével tölthetjük be, a feleslegessé vált modulok memóriából való eltávolítására pedig az `rmmod` utasítás szolgál. A betöltött modulokat az `lsmod` paranccsal listázhatjuk ki.
Egyes modulok betöltésekor értékeket is meg kell adnunk, hogy hol és mit, arról a rendszermag leírásában olvashatunk bővebben. A legtöbb modul betöltéséről azonban a rendszermag saját maga gondoskodik.
Lássunk neki az elemek kiválasztásának! Ehhez többféle út is kínálkozik: az egyik a konzolos menüvezérelt alkalmazás, amelyet a */usr/src/linux* könyvtárból a `make menuconfig` utasítás segítségével kelthetünk életre. Akik a grafikus környezetet kedvelik jobban, azok egy grafikus konzolból adják ki a `make xconfig` parancsot. A Linux-rendszermag fejlesztői az önsanyargatókról sem feledkeztek el: a `make config` parancsot az ő figyelmükbe ajánljuk.
A különböző elemeket különböző osztályokba csoportosítva találjuk. A továbbiakban helyhiány miatt csak a legfontosabbakra térhetünk ki (az összes támogatás részletes bemutatására a fél újság sem lenne elég). Mindenesetre bővebb tájékoztatásért nézzük át a rendszermag leírását vagy nyomjuk meg a ***Help*** gombot.

Code maturity level options

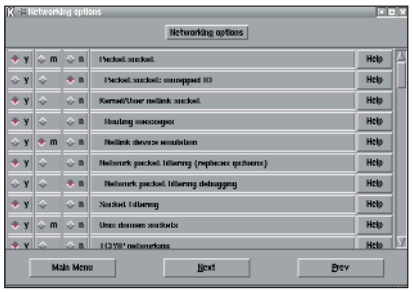
Ha bekapcsoljuk, elérhetjük a rendszermagban szereplő, de még csak kísérleti (EXPERIMENTAL) állapotban lévő támogatásokat is. Figyelem, az ilyen szolgáltatások nem üzembiztosak, ezért mindenki csak a saját felelőségére használja őket!

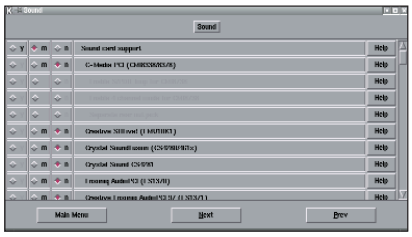
Loadable module support

Az ***Enable loadable module support***-ot mindenképpen fordítsuk be a rendszermagba, mivel nélküle nem élvezhetjük a modulok nyújtotta előnyöket.

Kernel module loader

A modulok önműködő betöltését teszi lehetővé.





Processor type and features

Itt adhatjuk meg processzorunk típusát. Ez azért fontos, mert a fordítóprogram a rendszermagot erre a processzorra fogja hangolni, ennek köszönhetően rendszerünk sebessége jelentősen nőhet.

High memory support

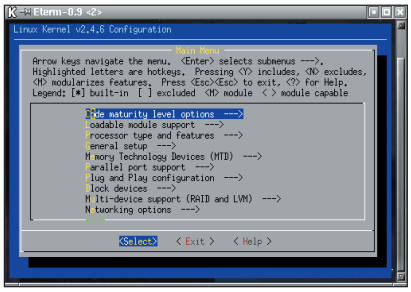
Ezt állítsuk ***Off***-ra, amennyiben 1 GB-nál kevesebb fizikai memóriával rendelkezünk.

Symmetric multi-processing support (SMP)

Csak abban az esetben kapcsoljuk be, ha egynél több processzorral bírunk, ugyanis ez a támogatás teszi lehetővé, hogy a rendszermag egynél több processzort használjon a folyamatok futtatására.

General setup

A ***PCI access mode***-nál adhatjuk meg, hogy a rendszermag milyen módon keresse meg a különböző PCI-os eszközöket. Ez történhet a BIOS segítségével, de közvetlenül (***Direct***) is. A legbiztosabb, ha az ***Any***-t választjuk; ebben az esetben a rendszermag PCI-os eszközeinket először a BIOS-on keresztül, majd ha ez sikertelen, közvetlenül próbálja meg elérni.
Az MCA-támogatás a PS2-es eszközök meghajtására szolgál. Ha PCMCIA-s kártyákkal is rendelkezünk, azok támogatását is itt kapcsolhatjuk be. A ***System V IPC***-t feltétlenül tegyük működővé, ugyanis ez teszi lehetővé a futó folyamatok közötti kapcsolattartást. Nagyon ügyeljünk rá, hogy a ***Support for ELF binaries***-t mindenképp magába a rendszermagba fordítsuk (és ne modulba!). Az elf a Linux futtatható binárisainak a formátuma, olyasmi, mint a Windows világában az exe. A többi bináris támogatását viszont nyugodtan modulba is helyezhetjük. Ha masinánk energiaellátását programból is szeretnénk szabályozni, ne felejtsük el bekapcsolni a ***Power management support***-ot sem!



Parallel port support

Ez a párhuzamos kapu támogatása, nyugodtan tegyük modulba. Figyelem, amennyiben PC-t használunk, a ***PC style hardware***-re is szükségünk lesz!

Block devices

Ide tartozik a PC-s hajlékonylemez-meghajtók támogatása, amit nyugodtan tegyünk modulba, még abban az esetben is, ha Linuxunkat lemezről indítjuk.
A rendszermagot ugyanis a lemezről a Linux indításvezérlője, a LILO tölti be, ami saját maga is tudja a lemezt, illetve a merevlemezeket kezelni.

Multi-device support

Ennél a menüpontnál a RAID-eszközök különböző támogatásait találjuk. A ***linear*** azt jelenti, hogy az összekötött lemezeket folyamatosan, egymás után töltjük meg. Az is megoldható, hogy több lemezt lássunk egy fájlrendszerként, ilyenkor a ***striping*** (csíkozás) nyújthat hasznos szolgáltatásokat. A ***mirroring***-gal (tükrözés) pedig megoldhatjuk, hogy két különálló lemezre pontosan ugyanazokat az adatokat írjuk fel. Ez nagymértékben növeli adataink biztonságát, ha ugyanis az egyik lemez megsérül, adataink a másikon még mindig elérhetők lesznek.
Az ***LVM support*** lehetővé teszi, hogy lemezterületeink összevonásával logikai kötetet hozzunk létre. Ezt a szolgáltatást csak a 2.4-es rendszermagok tartalmazzák. Akkor tehet jó szolgálatot, ha hirtelen lenne sok helyre szükségünk, de nem akarunk RAID-et használni.

Networking options

A ***Packet socket***-ra csak néhány egyedi hálózati alkalmazásnak van szüksége (például `tcpdump`), tehát nyugodtan modulba tehetjük. A ***Kernel/User netlink socket***-ra és a ***TCP/IP networking***-re azonban feltétlenül szükségünk lesz. A Linux-rendszermag ezenkívül számtalan hálózati szolgáltatást is tartalmaz, például teljes csomagszűrő tűzfalat. Ezt a 2.2-es rendszermagokban IP Chainsnek hívták, az alrendszert azonban a 2.4-esben kicserélték egy másikra, amelyet az IP Tables névre kereszteltek.

ATA/IDE/MFM/RLL support

Az IDE-eszközök és lapkakészletek támogatását tartalmazza. Figyelem, amennyiben Linuxunk fő lemezterülete IDE-me-

Példa a /etc/lilo.conf fájlra

```
boot=/dev/hda
prompt
timeout =200
# mennyi idı Ælljon
# rendelkezØsØre a
# felhasznÆl nak annak
# eld ntØsØre, hogy melyik
# rendszermaggal induljon
default=uj
# melyik rendszermag legyen
# az alapØrtelmezett
image =/boot/vmlinuz
   label = regi
   root=/dev/hda8
   read-only
image=/boot/œj kernel
   label=uj
   root=/dev/hda8
   read-only
```

revlemezen található, az IDE-támogatást semmiképpen se tegyük modulba, hanem közvetlenül a rendszermagba fordítsuk!

SCSI support

Amit az IDE-knél elmondtunk, az SCSI-nél is igaz. Ha nincs és nem is lesz SCSI-eszközünk, ezt a részt nyugodtan kihagyhatjuk. Egyvalamire hívjuk csak fel a figyelmet: a párhuzamos kapura köthető Iomega zipmeghajtó is az SCSI protokollt használja.

Network device support

Itt a különböző hálózati kártyák támogatását találhatjuk. Ki tudja miért, de a PPP is ide került, ez szükséges a telefonvonalas internetkapcsolat létrehozásához. Ha tehát modemmel kapcsolódunk a Világhálóra, a PPP-támogatást ne felejtsük el legalább modulba tenni.

ISDN subsystem

Ma már az analóg telefonvonal mellett az ISDN is egyre népszerűbb. A Linux természetesen ezt is gond nélkül támogatja. Meg kell jegyeznünk azonban, hogy az ISDN-en keresztüli internetezéshez más PPP-támogatás szükséges, mint a „hagyományos" telefonvonalaknál; előbbinél a ***Support synchronous PPP*** lesz a megfelelő.

Multimedia

Itt találhatjuk a videodigitalizáló és tv-tuner kártyák támogatását.

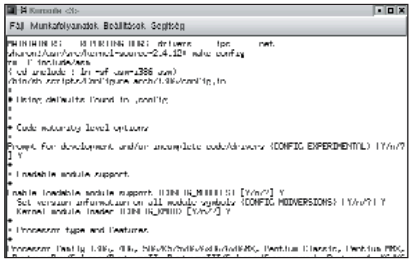
Filesystems

Megmondhatjuk, hogy rendszermagunk milyen fájlrendszereket támogasson. A legfontosabb a ***Second extended fs support***, amely nem más, mint az ext2, a Linux-lemezterületek fájlrendszerének a támogatása. Ezt sem szabad modulba tennünk.

Sound

A hangkártya-támogatást tartalmazza. A ***Sound card support***-ra mindenképpen szükségünk lesz. A rendszermag az általa ismert hangkártyákra a más Unix-rendszerekben is elterjedt OSS-t használja, ezzel azonban számos gond akadt, például kevés eszközt támogat. Így Linux alá kifejlesztették az ALSA-t, amely teljesen felváltja a rendszermagban lévő OSS-t. Ha készen vagyunk, nyomjuk meg a ***Save and Exit*** gombot, ezután visszatérünk a parancssorba. Most már kezdetét veheti a fordítás! Első feladatunk a függőségek beállítása, azaz az általunk kiválasztott elemeket elő kell készíteni a fordításra. Adjuk is ki tehát a `make dep` parancsot. Ha ez megtörtént, a `make clean`-nel eltávolíthatjuk a feleslegessé vált állományokat. Most következik maga a tényleges fordítás. Először a rendszermagot kell lefordítanunk, ezt követően pedig a modulokat. A lefordított rendszermagot egyébként rendszermaglenyomatnak (kernel image) nevezzük. A rendszermaglenyomatot a `make zImage` paranccsal készíthetjük el. Ha túl nagyra sikerült, két dolgot tehetünk: vagy több támogatást teszünk modulba, csökkentve ezzel a rendszermag méretét, vagy a `make zImage` helyett a `make bzImage` parancsot használjuk.
A következő lépés a modulok lefordítása a `make modules` paranccsal. Ha ezzel is kész vagyunk, a `make modules_install` paranccsal egyből a helyükre tehetjük őket, ilyenkor a ***/lib/modules/kernel-xxx/*** könyvtárba kerülnek.

Maga a fordítás általában körülbelül 10–15 percet vesz igénybe, de az időtartama természetesen a gép sebességétől függ, illetve attól, hogy mennyi elemet fordítottunk le. Ha a lenyomat vagy a modulok fordítása nem járna sikerrel, olvassuk el figyelmesen a hibaüzenetet, hátha az elemek kiválasztása során követtünk el valami hibát. Ha mégsem, próbálkozzunk a `make mrproper` pa-



ranccsal, ennek hatására a fordítást „tiszta lappal" kezdhetjük. Figyelem, e parancs kiadása után újból ki kell választanunk a lefordítandó elemeket! Utolsó lépésként telepítenünk kell a kész rendszermaglenyomatot. A lenyomatot az ***arch/i386/boot*** könyvtárban találjuk zImage vagy bzImage néven. Másoljuk a ***/boot*** könyvtárba, például ujkernel néven. Ezt követően az új rendszermagunkat hozzá kell adnunk a LILO-hoz.
Akik már behatóbban ismerik a LILO-t, bizonyára tudják, hogy lehetővé teszi, hogy ne csak a gépünkön lévő operációs rendszerek között, hanem a használni kívánt linuxos rendszermag között is választhassunk. Nem érdemes a régi rendszermagunktól egyből megválnunk, mivel elképzelhető, hogy az új rendszermaggal a rendszerünk nem fog elindulni. Az ebből adódó kellemetlenségeket kerülhetjük el, ha a LILO-ban a régi lenyomatunkat is meghagyjuk.
A LILO beállítófájlja a ***/etc/lilo.conf***. ***Listánkon*** példát láthatunk rá, miként adhatjuk hozzá a frissen fordított rendszermagunkat. Ha ezzel megvagyunk, adjuk ki a `lilo` parancsot, működőképessé téve az új beállításokat.
Indítsuk újra a rendszerünket, és ha mindent jól csináltunk, az új rendszermag fogad minket.

Garzó András
Körülbelül 3 éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.

Aerodinamika és a nyílt forrású alkalmazások

Steve bemutatja, miként lehet nyílt forrású programokat használni repülőgép-kísérleteknél.

Régen, még a Nyílt Forrású Programok mozgalma, a World Wide Web, a Szabad Program Alapítvány és a GNU megszületése előtt az Unsteady Aerodinamics Laboratory of the National Research Council of Canada alkalmazásában álltam. Feladatom a nagysebességű szélcsatornakísérletek adatainak összegyűjtése volt. Abban az időben a laborban egy különleges valós idejű miniszámítógép, egy Hewlett-Packard HP-1000 F sorozatú működött. Valamikor a nyolcvanas évek elején vagy közepén részt vettem a HP International Users Group tanácskozásán, és egy mágnesszalaggal tértem haza, ami a résztvevők által írt programokat és cikkeket tartalmazta. A szalagot a könyvtárrendszerbe fűztem és miután bepillantást nyertem az indexállományba, úgy éreztem magam, mint a kisgyerek, amikor karácsony este kicsomagolja az ajándékait. Először találkoztam forráskódmegosztással és fogalmam sem volt róla, mi mindent tartogathat a jövő egy ilyen nagyszerű és világos elképzelés számára.

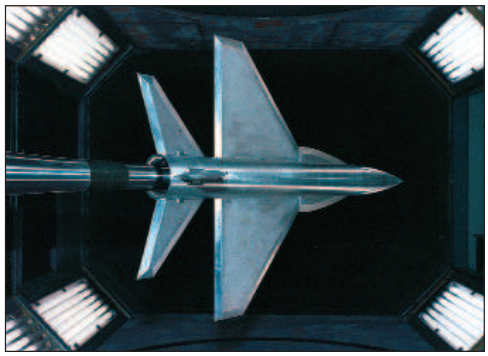
1990-ben kaptam meg az első unixos gépet: az akkor csúcstechnikának számító Silicon Graphics 4D/80GT-t saját T1-es internetkapcsolattal. A kicsi, hálózati kapcsolattal nem rendelkező, valós idejű operációs rendszerrel ellátott gépről az IRIX-alapú hálózati számítógépre történő váltás új, de kockázatos világra nyitott ablakot. Nagyon sok új ismeretet kellett elsajátítanom. 1992-ben vágtam bele a Byzantine nevű parancsállomány megírásába, amely a szélcsatorna adatállományainak kezelésére az `awk` és az `scd` kombinációit használta. Egy napon az egyik feladatom megoldásához tanácsot kértem az Useneten, és valaki megemlítette a Perlt. Bárcsak tudnám, ki volt az – most köszönetet mondhatnék neki, mert sokkal könnyebbé varázsolta az életemet. Egy éven belül a Perl mind az SGI-n, mind az otthoni Macintosh-gépemen nélkülözhetetlenné vált. A Perl a nyílt forrású programok legfontosabb darabja lett a laboratóriumban is.

Abban az időben a Mosaic programmal kísérletezgettem, amit – bár hasznosnak találtam – először igencsak alábecsültem. Nem sokkal később a NCSA által készített HTTPd-t telepítettem, és ekkor csodálkoztam rá a Web lehetőségeire is. A HTTPd később Apache néven született újjá, és a második legnagyobb OSS projektté lépett elő, amelytől egyébként a labor mindennapi munkája is függött.

1995-ben az Aerodinamikai Kutatóintézetet átszervezték, és a zűrzavar csillapultával a mostani helyemen találtam magam: az Aerodinamikai Laboratóriumban, ahol kis sebességű, 23 méteres szélcsatornájának felügyeletét látom el. Akkortájt kezdtem webalapú programokat készíteni, hogy kiterjesszem a már meglévő adatrendszer képességeit. Ezeket parancssoron és QNX, illetve AIX alatt futó X Window rendszeren keresztül lehetett elérni. Azért váltottam böngészőalapú programokra, mert a szélcsatorna adatbázisához kapcsolódó ügyfelek sokszínűsége ezt kívánta meg. Jóllehet munkánk nagy részét repülőgépek kipróbálása tette ki – olyan nagyvállalatok számára, mint a Bombraider Aerospace –, az utóbbi néhány évben azonban autókat, buszokat, teherautókat, motorkerékpárokat (a kedvenceim), villamos távvezetékeket, hidakat, antennákat, valamint olimpiai kerékpárosokat, síelőket és bobcsapatokat is próbára tettünk. Az ügyfelek ilyen széles köre – a számítástechnikához alig értőktől egészen a profikig – igazi kihívást jelentett a felhasználói felület megalkotásában. Mióta a vezetés is tudja, hogyan kell az Interneten keresgélni, elhatároztam, hogy alkalmazásaink közül egyet megpróbálok átültetni Perlre, amelyet azután webalapú felületen lehet elérni. Nagyon kedvező visszajelzéseket kaptam, mind a könnyű használhatóságának, mind az ügyfelek és a munkatársak által tapasztalt magas fokú kénye-

1. kép A kanadai sícsapat tagja 2 m × 3 m-es szélcsatornában

2. kép Jármű vizsgálata 2 m × 3 m-es szélcsatornában

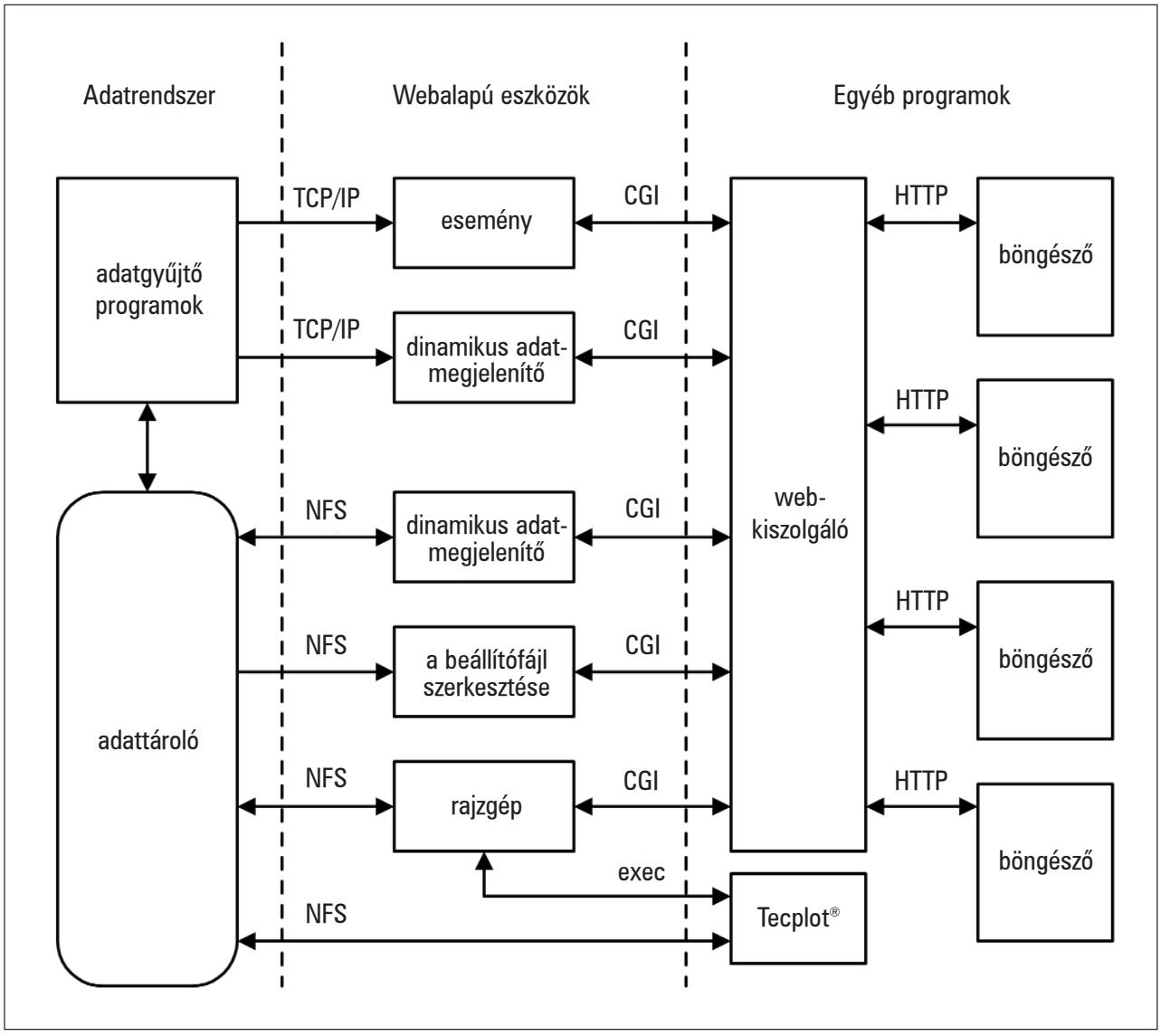


3. kép Harci repülőgép modellje 2 m × 3 m-es szélcsatornában

lemnek köszönhetően, ezért folytattam a webalapú eszközök készítését.
A harmadik kiemelkedő OSS-projektet jó pár évvel később fogadtuk örökbe. 1998-ban egy 24 csomópontos Alpha/Linux Beowulf telepet vásároltunk a folyadékok dinamikájával foglalkozó csoport számítási igényeinek kielégítésére. Ez kiváló lehetőségnek bizonyult az új módszer kiértékelésére, mert a feladat ugyan sokkal számításigényesebb volt, mint a szélcsatorna adatrendszeréé, a labor ügyfelei számára azonban nem bírt akkora horderővel. A sikeres alkalmazás meggyőzött bennünket, hogy az eddig használt kereskedelmi operációs rendszerek mellett a Linux életképes választási lehetőség.
Míg ezek a terjedelmes programok a helyükre kerültek, számos kisebb OSS-programot kezdtünk használni, például a Ghostscriptet, az Xmgr-t, a Vimet és a Neditet.

A jelen

A cikk további követhetőségéhez leírom az egyik, az adatfeldolgozás szempontjából jellemző repülőgép-kísérletet. Miután a modellt elhelyezik a szélcsatornában, 1–5 hét időtartam szükséges a próba befejezéséig. Ez idő alatt több mint ötszázezer mérést végeznek, ennek következtében 2000 X-Y plot, 4000 lemezállomány jön létre, és 500 MB szöveges adat jelenik meg a képernyőkön. Szükségszerű, hogy ennek az adattengernek a kezelésére gyors és egyszerű módszer kell, hogy rendelkezésre álljon. Az ügyfelek és a labor mérnökei az adatok gyűjtését, tárolását és megjelenítőrendszerét minden esetben Apache alatt futtatott Perl CGI programon keresztül érhetik el (lásd *ábránkon*). A szélcsatorna operátorai a kísérlet számos vezérlőelemét ugyanilyen módon kezelhetik. Amennyiben a vezérlőteremben a felhasználó bármelyik számítógépen megnyit egy webböngészőt, a szélcsatorna ügyfélhonlapja önműködően betöltődik. Ezen a lapon keresztül lehet elérni azt az öt webalapú programot, amelyeket az eddigiek folyamán írtam: ábrázoló, beállítási állományszerkesztőt, adatállomány-nézegetőt, eseménynaplózót és dinamikus adatmegjelenítőt. Ezenkívül léteznek hivatkozások a helyi segédeszközökhöz is, például a rendszerleíráshoz, illetve a mértékegység-átváltó számológéphez. Szeretném felhívni a figyelmet, hogy a laboratórium eléggé



Webalapú programrenszer szerkezete

korlátozó típusú helyi hálózati modellt használ, amely segíthet a webalapú rendszereknél előforduló biztonsággal kapcsolatos aggodalmak eloszlatásában. Elsőként az ábrázolórendszert fejlesztettük ki, és annak bizonyítására használtuk fel, hogy a szélcsatorna ügyfelei adataikat a Weben keresztül is el tudják érni. Amikor elkezdtük az Amtek Engineering által készített Tecplot kereskedelmi adatmegjelenítő programot használni, elhatároztam, hogy az ábrázolórendszert e program köré írom meg. A felhasználók az ábrázolómintákat – a beállítások kiválasztásával és szövegmezők kitöltésével – egyszerűen HTML-űrlapon keresztül állíthatják be. Ezeket a mintákat azután Tecplot-parancsállományok előállítására használjuk, ezáltal képernyőn vagy papíron lehetőség nyílik az eredmény megtekintésére. Egy démon (szintén Perlben íródott) ugyanezeket a mintákat használja a nyomtatásra, amely minden szélcsatorna-kísérlet végén önműködően zajlik. A beállítóállomány szerkesztését egy másik webalapú programmal valósítottam meg, amelyben a kísérleti adatokat gyűjtő és egyszerűsítő programok vezérlőállományait gyors és egyszerű módszerrel lehet módosítani. A felhasználók olyan űrlapot látnak, amelynek minden sora szövegmezőket és választógombokat, valamint a hozzá tartozó értékneveket tartalmazza. Az a Perl program, amely ezt a HTML-űrlapot hozza létre, dinamikusan előállít egy JavaScript-kódot is, amelynek az a feladata, hogy az űrlap benyújtása előtt ellenőrizze a kitöltött adatok érvényességét. Ha érvénytelen bejegyzést talál, a beviteli mező mellett villogó nyíl jelenik meg, és egy előugró párbeszédablakban a hiba jellege lesz olvasható.
Az adatállomány-megjelenítő egy egyszerű CGI program, amely egy adott szélcsatornapróba adataiért kutatja át a lemezterületet. Minden bejegyzéshez, amely a keresési mintára illeszkedik, egy HTML-gombot készít. Ezek a gombok táblázatban helyezkednek el, ahol minden sor a hasonló csatornapróbákat, illetve minden oszlop a megadott adattípust tartalmazza (például nyers, egyszerűsített). Bármely gomb megnyomására új böngészőablak nyílik meg, ahol a kiválasztott állomány formázott és elemzett alakban tekinthető meg. Ezután a felhasználók számára adott a lehetőség, hogy a saját gépükre CSV, Matlab vagy bármely más formátumban letöltsék. Minden új alkalmazás és számos régebbi kód állapotüzeneteket állít elő – eseményeket, amelyeket eseménynaplózó rendszerrel kezelünk. Ez a rendszer két fő részből tevődik össze: az első egyszerű Perl-démon, amely egy TCP/IP-kaput figyel, hogy érkezik-e rajta üzenet, s amennyiben igen, a naplóállományokban tárolja őket. A rendszer másik része egy webalapú megjelenítő, amellyel a felhasználók különböző szempontok

4. kép Forgó F18-as modell a vízcsatornában

alapján kereshetnek a naplóállományok között, mint például az esemény ideje, a számítógép neve, az esemény súlyossági szintje. Ez a program jelentéktelennek tűnik, pedig nélkülözhetetlen, mert az adatgyűjtő, -kezelő és -megjelenítő rendszer számos, különböző operációs rendszerrel működő számítógépből áll. Az ilyenfajta osztott rendszerekben a hibakeresés nagyon nehéz (különösen az összetettség miatti összehangolás okoz gondokat), általános eseménynaplózó rendszer nélkül csaknem lehetetlen. A felhasználók szempontjából az egyetlen nem interaktív eszköz a dinamikus adatmegjelenítő rendszer: Perl-kiszolgálón alapul, amely az adatgyűjtőrendszertől adatcsomagokat fogad. A felhasználók egy CGI-programon keresztül a kiszolgálóhoz kapcsolódva tudják megjeleníteni ezeket az adatokat. A CGI-program NPH-t (nonparsed header) vagy „server push”-t használ. A program egy adattáblázatot jelenít meg, amelyben az újabb adatokat dinamikusan a táblázat tetejére írja, a régi adatot pedig lefelé tolja. A program készítése folyamán aggódtam a memóriahézagok miatt, amelyek nemcsak Perlben vagy Apache-ban, de az ügyfelek böngészőjében is előfordulhatnak. Fölöslegesen, hiszen akadtak különleges NPH-ügyfelek, amelyek a kiszolgálóhoz folyamatosan kapcsolódva olyan anyagokért kutatták át a rendszert, amelyek több mint öt hete készültek. Eközben minden gond nélkül 500 MB-nál több adatot jelenített meg.
Ennek az öt programnak bármelyike külön-külön jól használható lenne, de aligha nevezhetők forradalminak. Mihelyt azonban egyesítjük őket, egyszerű, szilárd és nagyméretű környezetet kapunk. Nincs szükség nehezen érthető parancsokra, hosszú adatelérési útvonalakra, parancsbillentyű-kombinációkra vagy bármi másra, amely különféle típusú kezelőfelületekre jellemző. Nem kell mást tenni, mint kattintani, kijelölni, és kitölteni a mezőket – mindenki tudja, mit és hogyan tegyen.

A jövő

Teendőim hosszú listáján előkelő helyen szerepel azoknak a Perl-kódoknak a kétprocesszoros Intel/Linux-rendszerre történő ültetése, amelyeket az utolsó megmaradt SGI-rendszeren fejlesztettem. Bár a programok futtatása a jelenlegi formájukban jelentéktelen feladat, éltem az összes kód újrahangolásának lehetőségével. Még egy alkalmazás befejezése lenne különösen fontos: a modell viselkedését vezérlő rendszer felhasználói felületéé. Ráadásul figyelembe kell vennem adatformátumaink váltását – az arcane házilag fejlesztett formátumától az XML-ig. Ez szintén néhány új kód szükségességét eredményezi. Távolabbra pillantva a jövőbe, remélem, elég időm lesz kifejleszteni néhány VRML-alkalmazást is, amelyek a szélcsatornában elhelyezett modellek és szondák terhelés- és nyomásviszonyait képesek utánozni, és 3D-ben dinamikusan megjeleníteni. A műszeres csoport is vizsgálja a beágyazott, illetve valós idejű Linux-rendszerek felhasználásának lehetőségét.
Mire mindezen munkák elkészülnek, a nyílt forrású programok egyre növekvő fontosságú szerepe már vitathatatlan lesz az Aerodinamikai Laboratórium mindennapi munkájában.

Steve Jenkins az Aerodynamics Laboratory of the Institute for Aerospace Research vezető programozója és elemzője, a légi kutatások adatfeldolgozásában több mint húszéves tapasztalattal rendelkezik.

Amikor a Palm és a Linux beszélgetni kezd egymással...

Két egyetemi hallgató megoldotta a Palm és a linuxos számítógép összehangolását.

A Palm nagyszerű hordozható készülék: jegyzetelhetünk, találkozókat tervezhetünk vagy akár naplót is írhatunk vele. Csodálatos, hogy mindenhol velünk lehet útközben. Belső hálózatunk kiszolgálója szintén bámulatos masina. Vállalati terveinket, a megbeszélések napirendi pontjait, az érdekes feljegyzéseket, a címeket és a teljes üzleti adatbázist tárolja. Ez a kiszolgáló Linuxot, Apache-kiszolgálót és egy MySQL-adatbázist futtat, amelyeket egy, a célra tervezett alkalmazásmotor fog össze.

Ugye, csodálatos lenne, ha a két gépet össze tudnánk kapcsolni? Bárcsak sikerülne a Palmról elérni a Linux-kiszolgálón található adatbázist – máris jó úton haladnánk. A leírás szűkszavú, az Internet azonban hatalmas. ***Kevin*** és ***Jeffrey*** két egyetemista, akik kis erőráfordítással új megoldást fejlesztettek ki. Ennek alapján végezhetjük el a megfelelő változtatásokat a Palm-gépeken és a vállalati kiszolgálón, s e módosításokat a másik eszköz adatbázisaival is végrehajthatjuk.

A lejjebb látható kódokat egy RedHat 6.x Linuxot futtató Intel-alapú gépen és egy Palm OS 3.5-öt futtató, soros bölcsővel rendelkező Palm Vx-en próbáltuk ki, de más összeállítás sem okozhat gondot. Az alkalmazott könyvtárfájlok a Palm első piaci megjelenése óta (akkoriban még a 3Com gyártotta a gépet) nem változtak. Feltételezem (bár nem próbáltam ki), hogy a program a Visor-gépeken is gond nélkül működik, mivel ugyanazt az operációs rendszert használják.

Első lépésként az egyszerűbb résszel kezdjük: a Palmot kapcsoljuk össze a kiszolgálóval. Először is a Palm bölcsőjét kell összekötnünk a kiszolgáló soros csatlakozójával. Ezután létrehozunk egy pilot nevű eszközt, ami nem más, mint a soros kapu (esetünkben a */dev/ttyS0*) másodneve:

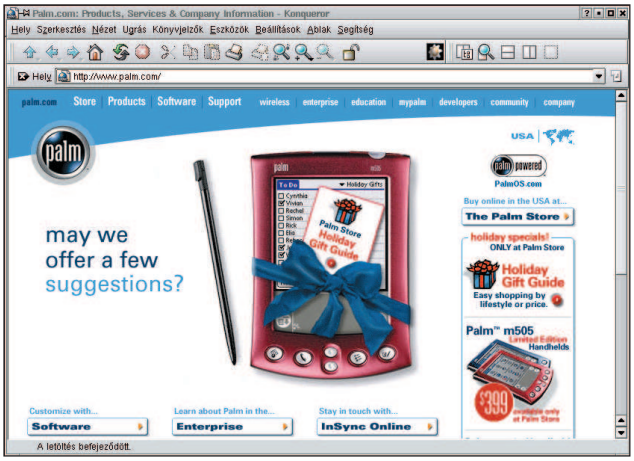
```
ln /dev/ttyS0 /dev/pilot
```

Most már egy C program és a ***HotSync*** gomb megnyomásával megnyithatjuk a kapcsolatot a Palmmal. Miután a kapcsolat létrejött, már csak a Palm adatbázisaiból kell a megfelelő adatokat kiolvasnunk.

A Palm és a számítógép közti kapcsolat és adatátvitel a ***pi*** könyvtárral egyszerűen megvalósítható. Ez a könyvtár a BSD-csatolófelületet utánozza: létrehoz egy foglalatot (socket), hozzákapcsolja az eszközhöz, figyel a bejövő kapcsolati kérelemre és elfogadja. A bejövő kapcsolatot a Palm és a bölcső kettőse váltja ki akkor, amikor a felhasználó megnyomja a HotSync gombot. Az *1. listán* láthatjuk, hogyan kell létrehozni egy Palm-kapcsolatra várakozó démont.

Miután a kapcsolat megvalósult, miként érhetjük el a Palm adatbázisait? Ezek mindegyike névvel rendelkezik. Az adatbázisokat a nevükkel nyithatjuk meg, majd meg kell adnunk az elérni kívánt rekordot, sőt az egész adatbázist is átnézhetjük. Macintosh- vagy Windows-gépeken ezt csővezetékek alkalmazásával oldják meg. A Palm is biztosít csővezetékeket e felületeken az összes, a Palm OS csomagba tartozó szabványos adatbázis számára. A Palm adatbázis-kezelő lehetővé teszi, hogy az adatbázisnak csak a módosított rekordjait tekintsük végig. Módosított – mióta is? Nos, a legutóbbi összehangolás óta, amikor a kiszolgáló utoljára érte el ezt az adatbázist. Ezt tehát programjainkban az összehangolás után kell megtennünk – a *2. listán* látható nyitott kapcsolatnál kell futtatni.

Amennyiben a Palm-adatbázisból rekordokat olvasunk ki, az nem számít összehangolásnak. Ennél többre van szükség, például arra, hogy a Palmra írunk, törlünk belőle és a saját MySQL-adatbázisunkból olvasunk. Mivel a MySQL-adatbázishoz való kapcsolódás ismertetése túlmutat e cikk keretein, most nem szólunk az összehangolás további részleteiről. ***Kevin Velghe*** nagyszerű leírást tett közzé a témáról, amelyre a ➲ http://www.duo.be/palm/mysql_palm.html címen bukkanhatunk rá.



1. lista A Palm összehangolása

```
Main() {
        int sd;
        struct pi_sockaddr addr;

        sd = pi_socket(PI_AF_SLP,
PI_SOCK_STREAM, PI_PF_PADP);
        addr.pi_family = PI_AF_SLP;
        strcpy(addr.pi_device, "/dev/pilot");
        pi_bind(sd, (struct sockaddr*)&addr,
➥sizeof(addr));

        sd = pi_listen(sd, 1);

        sd = pi_accept(sd, 0, 0);
        printf("HurrÆ! LØtrej tt a
              ➥kapcsolat...");
        pi_close(sd);
}
```

2. lista A módosított rekordok átfésülése

```
{
        int db, len, I, attr;
        recordid_t id;
        unsigned char buffer[4096];
...lØtrej n a kapcsolat...
        sd = pi_accept(sd, 0, 0);
        dlp_OpenDB(sd, 0, 0x40+0x80,
                ➥"DateBookDB", &db);
        for (;;) {
                len = dlp_ReadNextModifiedRec
                        ➥(sd, db, buffer, &id,
                        ➥&I, 0, &attr, 0);
                if (len < 0) break;
                printf(buffer); printf("\n");
        }
        dlp_ResetSyncFlags(sd, db);
        dlp_CleanUpDatabase(sd, db);
        dlp_CloseDB(sd, db);

        pi_close(sd);
}
```

További érdekességek

Michael J. Hammel a Linux Journal 1998 júniusában megjelent „Linux and the PalmPilot” című cikke nem a legfrissebb, de ismerteti a pilot-xfer használatát, ami nagyon hasznos segédprogram. Elolvashatjuk a ➲ http://www.linuxjournal.com/lj-issues/issue50/2711.html címen.
A „PalmOS Desktop HOWTO” című HOGYAN-ja a ➲ http://www.linuxdoc.org/HOWTO/PalmOS-HOWTO.html címen található meg. Az ➲ http://orbits.com-ra mutató hivatkozások legtöbbje már nem működik, hiszen ez a vállalat minden bizonnyal megszüntette az említett HOGYAN-ok fejlesztését. Ha valakinek sikerül megtalálnia az elveszett anyagokat, kérjük, küldje el őket a palm@duo.be címre, hogy a ➲ http://www.duo.be/palm címen mindenki számára elérhetővé tehessük.

3. lista A pack függvény

```
{
        int app_size, len;
        recordid_t pal_id, new_id;
        unsigned char buffer[512];
        struct Appointment app;

...
        strcpy(app.description,"Tennival ");
        app.begin = ...
...
        size = pack_Appointment(&app, buffer,
                ➥512);
        palm_id = 0;
        len = dlp_WriteRecord(sd, db, 0,
                ➥palm_id, 0, AppBuffer,
                ➥Appointment_size, &new_id);

}
```

A Palm-adatbázisban tárolt rekordok saját számokkal rendelkeznek. Amikor az eszközre rekordot írunk, ezt a számot kapjuk visszatérési értékként. Ezt a gépen vagy egy központi adatbázisban érdemes tárolnunk, így egy adott rekordot bármikor törölhetünk vagy frissíthetünk.
A `dlp_WriteRecord` egy Palm-rekordazonosítót fogad el. Amennyiben ez nulla, a Palm OS újat foglal le a számunkra; ha pedig létező azonosítót adunk át, akkor a megfelelő rekord kerül frissítésre. A legtöbb szabványos adatbázisrendszerben a rekordot a `pack` függvény csomagolja egy átmeneti tárolóba. Ez a folyamat a *3. listán* látható.

A Palm azonosítása

Amennyiben egy kiszolgálóval (tulajdonképpen egy bölcsővel) több Palmot is használunk (mint ebben az esetben is), meg kell határoznunk, hogy éppen melyik Palm csatlakozik a bölcsőre. Amint a kapcsolat létrejött, hívjuk meg a `ReadUserInfo` függvényt:

```
{
        int db, len, I, attr;
        recordid_t id;
        struct PilotUser U;
...lØtrej n a kapcsolat...
        sd = pi_accept(sd, 0, 0);
        dlp_ReadUserInfo(sd,&U);
        printf("Palm neve: %s", U.username);

        pi_close(sd);
}
```

Törölt rekordok

A Palm adatbázis-kezelő a rekordokat kiolvasás után nem törli – törlésre jelöli ki őket, de azt is megteheti, hogy a gépen vagy a kiszolgálón mentésre jelöli ki őket. Módosított rekord olvasásakor a fájltulajdonságokat ellenőrizni kell, amelyből megállapítható, hogy az adott rekordot törölni (vagy menteni) kell-e. Amint az adatbázis kitisztul, végérvényesen törlődik a Palmról:

```
{
    ...
    for (;;) {
       len = dlp_ReadNextModifiedRec(sd, db,
             ➥buffer, &id, &I, 0, &attr, 0);
       if (len < 0) break;
       if ((attr & dlpRecAttrDeleted) ||
            (attr & dlpRecAttrArchived))
          printf("T rlØsre kijel lve: %ld", id);
    }
}
```

Egy kicsi naplózás senkinek sem árt…

A Palm összehangolása után hasznos gyakorlat, hogyha a Palm naplófájljában megjegyzéseket is hagyunk róla. Akármit beleírhatunk, de az időpont és a dátum mindenképpen belekerül. Írjuk tehát az alábbi kódot programjaink végére:

```
{
       ...
       dlp_ResetSyncFlags(sd, db);
       dlp_CleanUpDatabase(sd, db);
       dlp_CloseDB(sd, db);

       dlp_AddSyncLogEntry(sd, "A Pilotr l
            ➥beolvastuk a m dos tÆsokat.\n");

       pi_close(sd);
}
```

Mi kell az induláshoz?

Ha a Pilotot linuxos géppel szeretnénk használni, szerezzük be a ***pilot-link*** csomagot. A kezelőfelületek számos rendszerhez elérhetők (Next, BSD, Solaris, OS/2, Linux stb.). A segítségükkel Python, Java, Perl, Tcl, C/C++ nyelvű programokat írhatunk. A szükséges fájl a ➲ http://ryeham.ee.ryerson.ca/pub/PalmOS címen elérhető FTP-kiszolgálón található meg, a neve: `pilot-link.0.9.3.tar.gz`, linuxos gépen minden gond nélkül lefordítható. A csomag tényleg több, mint csak egy csatolókönyvtár a használatát bemutató egyszerű példaprogramokkal. Ezek az egyszerű eszközök nagyon hasznosak: a segítségükkel teljes biztonsági mentést készíthetünk a Pilotról (és vissza is állíthatjuk), adatokat és adatbázisokat másolhatunk róla és vihetünk föl rá stb. A könyvtár C/C++, Perl, Python, Tcl és még néhány más nyelvből hívható meg. Aki rendelkezik némi programozási készséggel, az a csomagba tartozó példaprogramok és cikkünk listái segítségével könnyűszerrel összeállíthatja a számára szükséges eszközöket.
Az új programokat, a leírás kiegészítéseit, a megjegyzéseket, a HOGYAN-okat az olvasók a palm@duo.be címre küldhetik, mi pedig a ➲ http://www.duo.be/palm című honlapon mindenki számára elérhetővé tesszük őket.
Telepítsük a csomagot az alábbi paranccsal:

```
tar -xzvf pilot-link.0.9.3.tar.gz
```

Ez létrehozza a ***pilot-link.0.9.3*** könyvtárat, benne a forráskóddal. Lépjünk is bele.
Adjuk ki a `./configure` parancsot, ami átnézi a rendszert a fordításhoz szükséges kiegészítő fájlokat keresve. A `configure` a programot alapértelmezés szerint a ***/usr/local*** könyvtárba telepíti. Ha ez nem megfelelő számunkra, a `./configure --prefix=K NYVT`R` paranccsal állítsuk be a telepítés helyét.

Adjuk ki a `make` parancsot, ami lefordítja a csomagot. Ekkor a fájlok még nem kerülnek a helyükre, így a végleges telepítés előtt alkalmunk nyílik kipróbálni a programot. Ha egy régebbi változatra egy újabbat telepítünk, ellenőrizzük, hogy minden működik-e. Általában természetesen semmiféle gond nem szokott adódni.
Rendszergazdaként adjuk ki a `make install` parancsot, mely a fájlokat a megfelelő könyvtárakba helyezi. Ha rendszergazdaként nem tudunk belépni, akkor olyan helyre telepítsük a programot, amelyhez rendelkezünk írási jogosultsággal. Ne felejtsük el a futtatható fájlokat tartalmazó könyvtárakat beírni a rendszer alapértelmezett útvonalába (`PATH` változó). Nézzünk meg a csomag mellé kapott hasznos példaprogramokat is, amelyek leírását a ***További érdekességek*** részben említett helyen olvashatjuk el.

Johan Coppieters
(palm@duo.be) a Duo nv nevű cég vezetője. A belgiumi Brugesben székelő vállalat Belgium legnagyobb vállalatai számára készít weboldalakat, belső hálózatokat és internetes alkalmazásokat.

Kevin Velghe
írta a C nyelvű Palm–Linux összehangoló programot, s egy három hónapos iskolai gyakorlat cseretanfolyam közben a kutatás egy részét is ő végezte. A Duo nv-nél találhatjuk meg.

Könnyű álmok (10. rész)

A PAM használata a gyakorlatban

Sorozatunk előző cikkében (*Linuxvilág,* október 46. oldal) áttekintettük a felhasználók azonosítása kapcsán felmerülő kérdéseket, és általánosságban beszéltünk a Linux PAM rendszeréről. Írásunk célja, hogy megismertessük a PAM fontosabb alkotórészeinek használatát, és tanácsokkal szolgáljunk a beállításukkal kapcsolatban.

A PAM-rendszer fontosabb alapmoduljai

A PAM-rendszer az alapvető modulokat önműködően telepíti. Az alábbiakban felsoroljuk a leggyakrabban használt modulokat és fontosabb szolgáltatásaikat.

Általános PAM-hibakeresés

A „debug" kapcsoló

A PAM-modulok hibáinak felderítésére a `debug` kapcsoló használható. Beállításának hatására az adott modul hiba esetén a syslog(3) rendszerhíváson keresztül ír a rendszernaplóba. Mivel minden modul rendelkezik ezzel a kapcsolóval, a továbbiakban nem tárgyaljuk.

A PAM-rendszer fontosabb alkotórészei

A „pam_unix" modul

A `pam_unix` a legfontosabb és leggyakrabban használt modul a Linux-változatokban. A Unix-rendszerek hagyományos azonosítási (authentication) és feljogosítási (authorization) eljárásait biztosítja. A rendszer szabványos hívásait használja, tehát a */etc/passwd* és a */etc/shadow* állományokkal dolgozik. Érdemes megjegyezni, hogy a *pam_pwdb* modul is hasonló szolgáltatásokat nyújt, a felhasznált adatokat azonban adatbázisban tartja. Nagy felhasználószámú rendszereken érdemes alkalmazni.

- ***account***
  Kapcsolói: `debug`; `audit`.
  A felhasználói számla érvényességének ellenőrzését teszi lehetővé. A *shadow* állomány olyan mezőket tartalmaz (`expire`; `last_change`; `max_change`; `min_change`; `warn_change`), amelyek a felhasználó jelszavának kikényszerített cseréjét vagy a számla zárolását teszik lehetővé [1.]. Vigyázat, ha a *shadow* állomány a fenti mezők valamelyikét nem tartalmazza, az ellenőrzés nem hajtódik végre!
- ***auth***
  Kapcsolói: `debug`; `audit`; `use_first_pass`; `try_first_pass`; `nullok`; `nodelay`.
  A felhasználó jelszavas azonosítását tesz lehetővé. Amennyiben több jelszavas azonosítás is be van állítva (lásd később a `pam_ldap` modulnál), a `try_first_pass` érték használata célszerű. Ilyen esetben a rendszer a felhasználótól nem kérdezi meg újra a jelszavát, hanem az első modul által bekért jelszót használja. Ha azt szeretnénk, hogy a felhasználó több jelszóval lépjen be, ne használjuk. A `nullok` kapcsoló olyan felhasználók rendszerbe lépését teszi lehetővé, akiknek *shadow* állományában a kódolt jelszó mezője üres. A használata nem javasolt.
- ***password***
  Kapcsolói: `debug`; `audit`; `nullok`; `not_set_pass`; `use_authtok`; `try_first_pass`; `use_first_pass`; `md5`; `bigcrypt`; `shadow`; `nis`; `min`; `max`; `obscure`; `remember`.
  A felhasználók szabványos jelszócseréjét teszi lehetővé. Az `md5` és `bigcrypt` kapcsolók segítségével elérhető, hogy a jelszó ne a klasszikus `crypt` [2.] eljárással kódolva kerüljön a végleges helyére, hanem a megadottal. Ne felejtsük el beállítani, különben a rendszeren csak nyolc karakter hosszú jelszavakat lehet használni! Itt is alkalmazható a `try_first_pass`, ha a felhasználónak a különböző jelszótárakban egyforma jelszót szeretnénk beállítani. A `use_authtok` beállítás a modul számára kötelezővé teszi az előző modul által átadott jelszó beállítását. Erre a `pam_cracklib` használata esetén van szükség (lásd később). A `not_set_pass` kapcsoló segítségével letilthatjuk, hogy a bekért régi vagy új jelszó bármely más modulnak átadásra kerüljön. A `nis` kapcsoló hatására a rendszer a jelszó beállítására a NIS RPC-t használja. A `min` és `max` beállításával szabályozható a beállítható jelszó legkisebb és legnagyobb hossza. A `min`-t átlagos felhasználói rendszernél célszerű legalább nyolcra, erősen védett rendszernél pedig tízre állítani. Az `obscure` a beállítandó jelszón néhány alapvető ellenőrzést végez, amelyek a következők lehetnek: a jelszó nem hasonlíthat túlzottan az előzőhöz, nem lehet túl egyszerű (jelszóhossz, a használt karakterek típusa stb.), nem lehet az előző jelszó fordítottja vagy odavissza megegyező (például „qwerttrewq").
- ***session***
  Nincs kapcsolója.
  Használatával a felhasználó neve és a szolgáltatás a munkamenet (session) elején naplózódik (a leírás szerint a végén is, de a tapasztalat ennek gyakran ellentmond).

A „pam_deny" és a „pam_nologin" modul

A `deny` segítségével megakadályozható a felhasználó adott szolgáltatáshoz való hozzáférése. Az `auth` és `account` esetén a felhasználó azonosítását és hozzáférését teszi sikertelenné, és amennyiben a `password` elemben használjuk őket, a felhasználó nem tudja megváltoztatni a jelszavát. A `session` alatt használva lehetővé teszi, hogy a felhasználó ne hozhasson létre munkamenetet.
A `nologin` modul a PAM-rendszer `auth` elemében elérhető, és a szabványos unixos `nologin` használatát teszi lehetővé. Amennyiben a */etc/nologin* állomány létezik, az azonosítás sikertelen. Leggyakrabban a rendszer indulásakor alkalmazzák, többfelhasználós rendszeren azonban kényelmes lehetőséget nyújt a felhasználók belépésének időleges tiltására egy esetleges karbantartás idején.

A „pam_securetty" és a „pam_shells" modul

A `securetty` és a `shells` modul meghatározza, hogy az adott felhasználó által használt terminál szerepel-e a */etc/securetty* állományban, illetve a felhasználó bejelnetkező héjja benne van-e a */etc/shells* állományban. Amennyiben az állomány az adott bejegyzést nem tartalmazza, a felhasználót

mindkettő elutasítja. Mindkét modul a PAM-rendszer `auth` eleméből érhető el.

A „pam_listfile" modul

A `pam_listfile` az azonosítási szakaszban egy állomány tartalmán keresztül teszi lehetővé a karbantartás engedélyezését vagy tiltását.
Lehetséges kapcsolói:

- onerr=succeed|fail
- sense=allow|deny
- file=***állománynév***
- item=user|tty|rhost|ruser|group|shell
- apply=user|@group

A modul veszi az `item` által meghatározott elemet (ahol a `user` a felhasználó neve; a `tty` annak a terminálnak a neve, ahonnan a kérés érkezett; az `rhosts` a távoli gép neve – ha van; az `ruser` a távoli felhasználó nevét adja meg – ha van; a `group` pedig a felhasználó csoportja), és megnézi, hogy a `file` által meghatározott állomány tartalmazza-e. Ha tartalmazza és a `sense` értéke `allow`, a modul sikerrel tér vissza, ha `deny`, akkor elutasító választ ad. Amennyiben hiba lép fel (például a meghatározott állomány nem létezik), az `onerr` által beállított értékkel tér vissza. Ezt érdemes `fail`-re állítani. Az `apply` kapcsolót akkor célszerű használni, ha a vizsgált elem terminál, távoli gép vagy héj. Segítségével a sikeres visszatérés egy felhasználóhoz vagy egy csoporthoz köthető.
Így tehát egyszerűen korlátozható egy adott szolgáltatás elérése. Használatára a legjellemzőbb példa az FTP, amelynél a `listfile` modult használták fel, hogy bizonyos felhasználók számára megtiltsák a szolgáltatás elérését. A beállítóállományban ez a következőképpen néz ki:

```
auth  required  pam_listfile.so  item=user
➥sense=deny file=/etc/ftpusers onerr=fail
```

Egyéb felhasználására is mutatunk példát a későbbiekben.

A „pam_limits" modul

Lehetőséget ad a felhasználók által használható erőforrások korlátozására, amire azért van szükség, mert a többfelhasználós Linux-kiszolgálókon nem engedhető meg, hogy egy felhasználó olyan mértékben terhelje le a rendszert, hogy a többiek (különösen a rendszergazdák) ne tudják a munkájukat zavartalanul végezni. A Linux-rendszer e korlátozások használatát sajnos csak kis mértékben támogatja, így a rosszindulatú felhasználóktól csak a hagyományos módszerek védenek meg tökéletesen (időleges vagy végleges kizárás, vasalt orrú bakancs stb.).
Ezen keserű megállapítások a Linux 2.2.20-as és 2.4.14-es rendszermaggal folytatott hosszas kísérletezés után születtek. A felhasznált próbaprogramok az 1., 2. és 3. listán láthatók (24. CD Magazin/Konnyu könyvtár).
Amennyiben a felhasználók memóriafelhasználását korlátoztuk, a rendszer valamelyik fork-bombával túlterhelhetővé vált. Ha a belépésenkénti folyamatok (process) számát 4-re korlátoztuk, a fork-bombák akkor is szinte a teljes processzoridőt fel tudták használni, ráadásul az OpenSSH segítségével nem lehetett belépni a rendszerre. Tapasztalatunk szerint SSH-val csak akkor sikerült belépni a rendszerre, ha a lehetséges folyamatok száma legalább 40 volt. Mivel azonban a sikeres belépést követően a felhasználó 40 folyamatot futtathat, kedvezőtlen esetben a teljes processzoridőt le tudja foglalni. Az ésszerűtlen memóriafogyasztást meg lehet ugyan gátolni, de a sok memóriafoglalási kísérlet szintén megeszi a processzor idejének jelentős részét. Rendkívül kellemetlen, hogy ilyen esetben a legtöbbet a rendszermag dolgozik, így még korlátozni sem lehet.
A folyamatok számának korlátozása bizonyos esetekben a PAM-támogatás tökéletlen megvalósítása miatt nem megfelelő. Amennyiben a rendszeren a kifejezetten rosszindulatú felhasználókat ki tudjuk szűrni, van értelme a határok beállításának, mert ezzel csökkenthető a felhasználó akaratán kívül történő rendszertúlterhelés esélye. Sokat segíthet, ha a felhasználók `nice` szintjét csökkentjük, ezáltal hiba esetén a rendszergazdák nagyobb eséllyel tudnak sikeresen beavatkozni.
Eszményi az lenne, ha a felhasználónak általános határokat lehetne beállítani (jelenleg csak a belépésenkénti létezik), és a rendszermag lehetővé tenné annak beállítását, hogy egy bizonyos felhasználó által kezdeményezett (felhasználó vagy rendszermag által végzett) feladat legfeljebb mekkora részt kaphasson a rendelkezésre álló processzoridőből (fair share scheduling). Amíg a hivatalos rendszermagban ezek nem valósulnak meg, addig a felhasználók korlátozása csak részleges lehet.
Nem tartozik szorosan a témához, de itt érdemes megjegyezni, hogy a rendszermag lehetővé teszi annak a helynek a korlátozását, amit a felhasználók merevlemezen foglalhatnak. Így ésszerű mértékűre csökkenthető az egyes felhasználók területhasználata, és a levelesláda (mailbox) sem nőhet a többiek kárára egy adott méret fölé. Beállítása esetén azonban figyelni kell rá, mit tesz ilyen esetben a levelezőkiszolgáló.

Egyéb hasznos modulok

A `pam_env` modul (auth) környezeti változók előzetes beállítását vagy törlését teszi lehetővé. Többfelhasználós rendszeren célszerű alkalmazni, mivel a belépési héjjtól függetlenül teszi lehetővé a környezet egységes beállítását.
A `pam_rootok` modul (auth) azonosítja a felhasználót, ha a felhasználói azonosítója 0. Ennek akkor lehet értelme, ha a rendszergazdát nem akarjuk egy szolgáltatás minden egyes használatakor azonosítani. A Linux-változatok legtöbbjében a rendszergazdának megengedett a `su` használata jelszó nélkül, ami a beállítóállományban így fest:

```
auth    sufficient pam_rootok.so
auth    required   pam_unix.so
```

A rendszergazda úgy tevékenykedhet bármelyik felhasználó nevében, hogy nem adja meg annak a jelszavát. Ez felvet bizonyos erkölcsi kérdéseket, ami azonban szinte minden rendszergazdai jogosítványnál felmerül. Különleges esetben egy finoman hangolt, külső szakértők által is felülvizsgált rendszernél elérhető a rendszergazdák jogainak a szükségesre történő csökkentése, de ez komoly hozzáértést és erőforrás-ráfordítást igényel.
A `pam_chroot` modul (account, session, auth) segítségével lehetővé válik egy adott szolgáltatás root könyvtárának a PAM-rendszeren keresztüli beállítása. Hasznáról egy későbbi cikkben részletesebben írunk. A modullal jelenleg kissé nehézkes dolgozni, mivel a PAM-ot támogató programok egy része a munkamenetkezelést nem megfelelően valósítja meg. Jó példa erre az OpenSSH, ahol a PAM-megvalósítás félreérthetősége miatt a rendszer nem minden esetben működik helyesen. Többen kijavították az SSH hibáit, de a fejlesztők nem fogadták be a javításokat.
A `pam_motd`, `pam_mail`, `pam_lastlog` és `pam_issue` modulok a felhasználók tájékoztatását szolgálják. Belépéskor a terminálra a */etc/motd* és a */etc/issue* állományok tartalmát, az utolsó belépés idejét kiírják, továbbá jelzik, ha a felhasználónak új levele érkezett.
A `pam_radius` és `pam_krb4` modulok segítségével a felhasználók azonosítása egy RADIUS- [3.] vagy Kerberos- [4.] kiszolgáló segítségével történik.
Ezek az azonosítási eljárások általában nagyobb hálózatokon használatosak, és nagy biztonságú azonosítást tesznek lehető-

vé. A pam_securetty segítségével egy állományban meghatározható, hogy mely terminálok tekinthetők biztonságosnak. A pam_time modul használata lehetővé teszi a hozzáférés idő szerinti korlátozását. Minden biztonsági rendszer alapvető eleme a megszokott és elfogadott engedélyezése, és a kirívó esetek tiltása. Amennyiben például valószínűtlen, hogy a rendszergazda reggel tíz óra előtt belépjen a konzolról, e modul segítségével letiltható, vagy egy PAM-ot támogató játékalkalmazással megoldható, hogy csak munkaidőn kívül lehessen elindítani.

A PAM különleges moduljai

A „pam_cracklib" modul

A unix modul password elemének kiegészítésére szolgál. Jóval finomabb jelszóbonyolultság-ellenőrzést tesz lehetővé. A unix modul obscure kapcsolóknál említetteken kívül képes a szótári szavakon alapuló jelszavak kiszűrésére is. Kielégítő működéséhez egy megfelelő szavakat tartalmazó szótár szükséges, amit a legegyszerűbben oly módon állíthatunk elő, ha nagyobb mennyiségű levelezési listatárat szedünk össze, majd szavakra bontjuk. Célszerű olyan csomagokat is gyűjteni, amelyben a népek ékezettel leveleznek, mivel a felhasználók előszeretettel tesznek ékezetes szavakat a jelszavukba. Továbbá célravezető összeszedni a felhasználók, valamint kisállataik és szeretteik adatait. A 4. listán látható (24. CD Magazin/Konnyu könyvtár) egyszerű kis Perl-programocska segítségével a szövegállományokat szavakká daraboljuk.
A program a bemenetén a tiszta szövegállományokat várja (kismértékben akár HTML-lapokat is, bár ettől leendő adatbázisunk feleslegesen hízik), és az adatok a kimenetén szavakra darabolva érkeznek. Az adatbázis a Debianon az alábbi parancsösszetétellel állítható elő:

```
cat sok sz vegÆllomÆny neve |
      ➥mini_splitter | sort -u |
   crack_packer
      ➥/var/cache/cracklib/cracklib_dict
```

Ezzel előállítottuk a szóadatbázist, amelyet a későbbiekben ésszerű rendszeresen frissíteni. A modul ellenőrzi, hogy a megadott jelszó nem képezhető-e valamelyik szótári szóból a kis- és nagybetűk valamilyen kombinációjával.
A modul a PAM password elemében működik, használata egyszerűsítve a következő: a felhasználó által megadott jelszó minden karaktere egy pontot ér, továbbá minden különböző karakterosztályba tartozó karakter egy jutalompontnyit számít. A rendszer számára meghatározhatjuk, hogy egy adott karakterosztályra legfeljebb mennyi jutalompontot adjon. Az ismert osztályok: kisbetű (lower), nagybetű (upper), számjegy (digit) és egyéb (other). A jutalompontok hangolása a következő értékek beállításával történik:
dcredit=*N*; ucredit=*N*; lcredit=*N*; ocredit=*N*,
Az *N* az adott osztály karaktereire adható legmagasabb pluszpontok száma. Amennyiben a jelszóban megadott szám alatti vagy azzal megegyező számú adott osztályú karakter szerepel, mindegyikükért egy pluszpont jár. A karakterszámból adódó és a jutalompontok összegének legkisebbikét a minlen=*N* értékkel szabályozhatjuk. Az *N* értéke a megengedhető legkevesebb plusz egy. Így a következő beállításokkal:
dcredit=2 ucredit=1 lcredit=1 ocredit=2 minlen=12
csak olyan jelszó lesz elfogadható, amely vagy legkevesebb 10 kisbetűből áll, vagy ha van benne nagybetű, akkor nem rövidebb, mint 9 karakter; vagy ha van benne két számjegy és nagybetű, akkor nem rövidebb, mint 7 karakter és így tovább. A régi és új jelszó elvárt különbsége a difok=*N* értékkel állítható be. A segítségével megadható, hogy egy adott jelszóban hány karaktert kell mindenképpen lecserélni. Alapbeállítása tíz, de ehhez még egy újabb szabály adódik: amennyiben a jelszó karaktereinek legkevesebb a fele lecserélődik, felülbírálja az itteni beállítást, és a jelszó megfelel.

A „pam_ldap" modul

Nagyobb hálózatok estén gyakran felbukkanó gond, hogy a felhasználók a munkahelyek között vándorolnak, de mindenhol a megszokott munkakörnyezetet szeretnék látni. A rendszergazdák számára komoly nehézség lehet sok felhasználó együttes kezelése. Ilyen esetekben célszerű az LDAP-modul alkalmazása. A felhasználók adatait egy központi LDAP-kiszolgálón kell tárolni, így a tetszőleges munkaállomásra a megszokott jelszavukkal léphetnek be. A teljes támogatáshoz ne felejtsük el az nsswitch könyvtárakat sem áthangolni [5.]. A munkakönyvtárak átvitelére valamilyen hálózati állományrendszer is megfelel. Erre a célra jelenleg az NFS a legelterjedtebb megoldás, ami azonban biztonsági szempontból erősen megkérdőjelezhető, ezért használata kizárólag olyan környezetben fogadható el, ahol az ügyfelek tökéletesen megbízhatók – vagyis szinte sehol. Jelenleg a legésszerűbb a felhasználók munkakönyvtárait SSL-Sambán keresztül kiajánlani, így lehetővé válik a biztonságos csatlakozás. A kis kitérő után térjünk vissza a központi felhasználóazonosításhoz.
Az LDAP-modul használatához először is szükségünk lesz LDAP-kiszolgálóra, amelyen a felhasználók adatait tároljuk. Mi az OpenLDAP 2.0.14-es változatát használtuk. A telepítés Debian Woody rendszeren a megszokott apt-get parancs segítségével egyszerű (a csomag neve ***slapd***). Amennyiben a leendő LDAP szerkezetét előre megterveztük, telepítés közben létre lehet hozni a háttéradatbázist. A rendszer adatainak átemelésére tökéletesen alkalmas a PADL cég által fejlesztett MigrationTools nevű eszköz [6.]. Az OpenLDAP-nál az alapbeállítást kissé módosítani kellett, hogy a megfelelő sémameghatározásokat is betöltse.
Ésszerű a hozzáférést is szabályozni, mert az alaptelepítés bejelentkezés nélkül is olvasási jogot ad. A kissé paranoiásabb beállítás megfelelő része valahogy így fest:

```
access to attribute=userPassword
     by dn="cn=admin,o=Andrews,c=HU" write
     by anonymous auth
     by self write
     by * none

access to *
     by dn="cn=admin,o=Andrews,c=HU" write
     by self read
     by * none
```

Ezután megkezdődhet az LDAP-modul üzembeállítása, amely a */etc/ldap.conf* állományon keresztül zajlik. Lássuk az állomány tartalmát!

```
# Az LDAP-kiszolgÆl  neve
host tensor.andrews

# A keresØs kiindul pontjÆnak DN-je
base ou=People,o=Andrews,c=HU

# A rendszergazda DN-je (a jelszava a
# /etc/ldap.secret ÆllomÆnyban talÆlhat ,
# amely a libpam-ldap csomag telep tØsekor
# kit ltØsre ker l)
rootbinddn cn=root, o=Andrews, c=hu

# DN nevØben keres a pam_ldap modul.
```

```
binddn cn=admin, o=Andrews, c=hu
bindpw ubertitkosjelszo

# A jelszavak tÆrolÆsi formÆja
pam_password md5
```

A PAM beállítóállományába kerülő sorokat példánkban adjuk meg.

Egy példarendszer beállításai

Példánk tárgya egy kisebb hálózat egyik felhasználói gépe legyen. A felhasználók gyakran vándorolnak a gépek között, így azonosításukat LDAP-on keresztül oldjuk meg. A felhasználók a rendszert a konzolról való belépéssel érik el, távolról csak a rendszergazdák léphetnek be SSH segítségével. Megmutatjuk a login és az SSH PAM-modul beállításait.
A `login` modul beállításai csak annyiban térnek el a megszokottól, hogy a felhasználókat LDAP-ból is lehet azonosítani. Amennyiben a felhasználó az LDAP-modul segítségével azonosította magát, beengedjük, ha nem, megpróbáljuk a helyi felhasználói adatbázisból azonosítani. Ezt egészíti ki, hogy a `pam_listfile` modul használatával bizonyos felhasználók rendszerről való időleges kitiltását is lehetővé tesszük. Ennek kezelésére az 5. listában (24. CD Magazin/Konnyu könyvtár) található egyszerű kis Perl-program szolgál. A segítségével kilistázhatjuk a tiltott felhasználókat, felvehetünk és törölhetünk tiltást. Használatának megismeréséhez használjuk a `-h` kapcsolót.
Példarendszerünkön a */etc/pam.d/login* így néz ki:

```
# PAM beÆll t Ællomány a 'login' szolgÆltatÆshoz

auth    requisite  pam_listfile.so item=user
➥sense=deny file=/etc/security/deny_users
➥onerr=fail
auth    requisite  pam_securetty.so
auth    required   pam_nologin.so
auth    required   pam_env.so
auth    sufficient pam_ldap.so
auth    required   pam_unix.so try_first_pass

account     required   pam_unix.so

session     required   pam_unix.so
session     required   pam_limits.so
session     optional   pam_lastlog.so
session     optional   pam_motd.so
session     optional   pam_mail.so standard noenv

password    required   pam_cracklib.so retry=3
minlen=6 difok=3
password    required   pam_unix.so use_authtok md5
password    required   pam_ldap.so try_first_pass*
```

Az SSH beállítása a Debian által feltelepítettől annyiban tér el, hogy fel lett véve egy `listfile` modul, amely kizárólag a */etc/security/admins* állományban felsorolt felhasználókat engedi be. Az SSH PAM-állománya:

```
# PAM beÆll t Ællomány az 'ssh' szolgÆltatÆshoz
auth    requisite  pam_listfile.so item=user
➥sense=allow file=/etc/security/admins onerr=fail
auth    required   pam_nologin.so
auth    required   pam_env.so
auth    sufficient pam_ldap.so
auth    required   pam_unix.so

account     sufficient pam_ldap.so
account     required   pam_unix.so

session     required   pam_unix.so
session     optional   pam_lastlog.so
session     optional   pam_motd.so
session     optional   pam_mail.so standard
session     required   pam_limits.so

password    required   pam_cracklib.so retry=3
➥minlen=6 difok=3
password    required   pam_unix.so use_authtok md5
password    sufficient pam_ldap.so try_first_pass
```

Amennyiben a felhasználó a rendszeren jelszót változtat, az LDAP-ban is meg kell változtatni. Ehhez a `passwd` PAM-beállításait a következőképpen kellett módosítani:

```
# PAM beÆll t Ællomány a 'passwd' szolgÆltatÆshoz

password    required   pam_cracklib.so retry=3
➥minlen=6 difok=3
password    required   pam_unix.so use_authtok
➥md5
password    sufficient pam_ldap.so
➥try_first_pass
```

Ezzel az alapbeállításokat megválasztottuk, a finomhangolást mindenkinek az ízlésére bízzuk. A részletek és mellékhatások tekintetében nézzük meg a PAM-rendszer leírását [7.]. Azon szolgáltatásokról, amelyekről hely hiányában bővebben nem tudtunk szólni, elegendő adatot találunk a Linux PAM hivatalos honlapján [8.], továbbá számos egyéb hasznos modulra is ráakadhatunk. Mindenkinek hasznos keresgélést kívánunk!

Hivatkozásjegyzék

[1.] A ***shadow*** állomány formátuma - shadow (5)
[2.] A `crypt` eljárás leírása – crypt (3)
[3.] RADIUS ➲ http://www.gnu.org/software/radius/radius.html
[4.] Kerberos ➲ http://web.mit.edu/kerberos/www/
[5.] Authenticating with LDAP using Openldap and PAM
➲ http://www.imaginator.com/~simon/ldap/
[6.] plain to lpdap migration tools
➲ ftp://ftp.padl.com/pub/MigrationTools.tar.gz
[7.] A PAM-rendszer felhasználói kézikönyve
➲ http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/
[8.] A Linux PAM-rendszer hivatalos honlapja
➲ http://www.kernel.org/pub/linux/libs/pam/

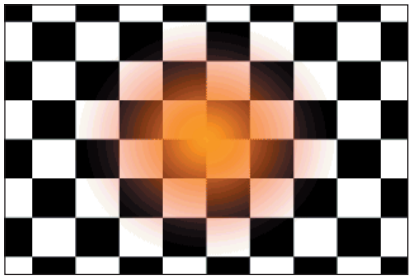
Mátó Péter (atya@andrews.hu), informatikus mérnök és tanár. Biztonsági rendszerek ellenőrzésével és telepítésével, valamint oktatással foglalkozik. 1995-ben találkozott először linuxos rendszerrel. Ha teheti, kirándul vagy olvas.

Borbély Zoltán (bozo@andrews.hu), okleveles mérnök-informatikus. Főként Linuxon futó számítógépes biztonsági rendszerek tervezésével és fejlesztésével foglalkozik.
A 1.0.9-es rendszermag ideje óta linuxozik. Szabadidejét barátaival tölti.

PoV-Ray ismeretek (6. rész)

Folytatjuk az anyagmódosítókkal való ismerkedést, hiszen most érkeztünk el az egyik olyan módosítóhoz, amellyel a leglátványosabb hatásokat hozhatjuk létre.

Részecskerendszerek segítségével modellezhetünk például robbanást, tüzet, füstöt és párát, továbbá felhőt és porfelhőt is. Az előbbieket természetesen különféle mintázatok felhasználásával is elkészíthetjük, de az igazán valósághű megjelenítéshez inkább a részecskerendszerek felelnek meg. Ismerkedjünk meg hát közelebbről a részecskerendszerek lelkivilágával – már amennyire e cikk keretei megengedik.
Első lépésben tűzhöz hasonló hatást próbálunk meg létrehozni, ehhez nyújt jó kiindulási alapot egy gömb. Az első fogalom, amivel meg kell ismerkednünk, a következő: tárolóobjektumnak nevezzük azt az objektumot, amit a részecskerendszer kiindulási formájaként határozunk meg. Minden részecskerendszernek szüksége van egy tárolóobjektumra, ami nem jelent nagy újdonságot, hiszen az anyagokat eddig is egy-egy tárgyhoz rendeltük hozzá. Fontosnak tartom megjegyezni, hogy a részecskerendszerek láthatóvá tételéhez a hordozó tárgynak áttetszőnek kell lennie, és a megfelelő hatás eléréséhez üreges test alkalmazása szükséges. Lássunk egy egyszerű példát, amiben a tűz kezdeti állapotait figyelhetjük meg. Itt részecskerendszereknek azt a típusát használjuk fel, amely egyszerűen csak fényt bocsát ki, ám az egyes részecskék a szomszédjukból sugárzó fényt nem nyelik el. Létrehozunk egy egyszerű jelenetet, amelyben a majdani tüzet megtestesítő gömb a tér középpontjában helyezkedik el, és fényforrással világítjuk meg. Természetesen ehhez a kamerát is meg kell alkotnunk. Láthatjuk tehát, hogy a gömb áttetsző, hiszen az `rgbt` kulcsszóval határoztuk meg a színét, amelynek utolsó értéke az adott szín átlátszóságát szabályozza. Ezenkívül a `hollow` kulcsszót is használtuk, amelynek hatására testünk üreges lesz. Felmerülhet a kérdés, hogy miért kellett a padlót alkotó síkot is üregessé tenni.
Ennek az az oka, hogy a részecskerendszer nem lehet egy másik tömör testen belül, mert ebben az esetben nem látható. Ha a síkot nem akarjuk tömörré tenni, a másik megoldás, hogy a kameránál a `negative` kulcsszót is megadjuk. Most tekintsük át, milyen ismeretlen szavak maradtak még számunkra a részecskerendszer meghatározásában.
A következő ilyen kifejezés az `emitting`, amelynek hatására a részecskék fényforrásként működnek. Célunk eléréséhez ez lesz a megfelelő típus, a további lehetőségek bemutatását a későbbiekben folytatom. A `spherical_mapping` adja a PoV-Ray tudtára, hogy ez esetben gömbszerű leképezés szükséges, ami nyilvánvaló is, hiszen a hordozótest is



1. kép Tűzfészek

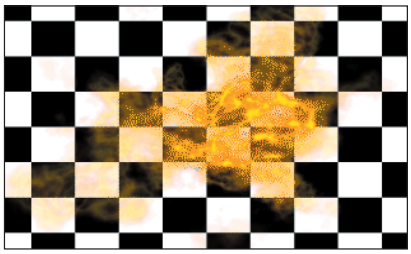
gömb. Egy részecskerendszernél az egyes részecskék eloszlását egy úgynevezett sűrűségfüggvény határozza meg, amely leírja, hogy mennyi részecskét találhatunk egy adott helyen. A PoV-Rayben pontos matematikai függvény helyett előre meghatározott eloszlásokat alkalmazhatunk. Ezek közül az egyik a `linear` kulcsszó által meghatározott egyenletes eloszlás. A részecskék száma minden esetben a koordinátarendszer középpontjában a legnagyobb, így előfordulhat, hogy egy tárgyat olyan koordinátákra helyezünk, ahol már nincsenek részecskék. Ilyenkor semmiféle tüzet, ködöt vagy felhőt nem láthatunk, de ennek tudatában a tárgyat szerencsére már a létrehozásnál a koordinátarendszer középpontjába helyezhetjük és később az egészet (a tárgyat az anyagokkal és a részecskerendszerekkel együtt) a kívánt helyzetbe tolhatjuk el. Az egyenletes eloszlásnak köszönhetően a részecskesűrűség a gömb középpontjában lesz a legnagyobb, a felszínen pedig 0.
A fenti példában gömbünk (ahogyan a „szófogadó” tűzhöz illik) középen átlátszóság nélkül sárga színű lesz, a felszínen pedig vörös és szinte alig látható. Az *1. kép* ezt az állapotot mutatja meg.
Első példánk utolsó ismeretlen szava a `samples` (a minták száma) névre hallgat és egy egész szám követi. Ezzel a számmal határozzuk meg, hogy a számítások során mennyi fénysugár haladjon keresztül a rendszeren, amíg a PoV-Ray kiszámítja a képet. Magasabb értékek esetén szebb eredményhez jutunk, kiszámítása azonban hosszabb időt vesz igénybe. Ha egy-egy részecskerendszer kiszámítása során teljesen szokatlan eredményt kapunk, érdemes ezt az értéket megnövelni – a legtöbb esetben a 10-es mintaszámérték megfelelő.
A fenti képen a színek meglehetősen halványak. Túl sok látszik a háttérből, így az első példánk szolgáltatta eredmény nem igazán hasonlít a tűzről, robbanásról alkotott képünkhöz. A színeket élénkebbé tehetjük, ha az objektum nagyobb sűrűségű részeinek átlátszóságát csökkentjük. Az alábbi részletbe a „-1”-es átlátszóság azért került bele, hogy a kapott eredmény jobban hasonlítson a tűzhöz.

```
Color_map {
   [ 0 color rgbt <1,0,0,1> ]
   [ 1 color rgbt <1,1,0,-1>]
}
```

Ennek az egyszerű változtatásnak fényesebb tűzkezdemény lesz az eredménye. A valóságban azonban ritkán láthatunk ilyen szabályos tűzgömböt, és én sem akadtam össze hasonló jelenséggel.
A valódi tűzgömb minden esetben rendezetlenebb formákat alkot, tehát a valósághűség érdekében a modellezés során nekünk is rendezetlenséget kell szimulálnunk. Erre már sorozatunk korábbi részeiben is láthattunk példát, ezért ismerősként köszönthetjük az alkalmazandó `turbulence` kulcsszót. Most nem ismétlem meg a listát, mivel mindössze egyetlen sort kell beszúrnunk a `linear` szó után (elé is megtehetjük: a `halo`-n belül és a `color_map`-on kívül), amivel kissé áttekinthetetlenebb kinézetet adhatunk a készülő

tűzgömbnek. Amennyiben a `turbulence 1.5` tartalmú sort a megfelelő helyre szúrjuk be, egy újabb leképezés után máris a világító labda nagymértékű átalakulásának lehetünk szemtanúi. Ennek kiszámításához több idő szükséges, mert minden részecske véletlen, örvényszerű elmozdulást kap. Amikor a részecskerendszert rendezetlenné tesszük, gyakran előfordul, hogy a részecskék a hordozó objektumon kívülre kerülnek. Elkerülésére magát a rendszert kell átméreteznünk, hiszen ha az objektum méretét változtatnánk meg, a részecskék elhelyezkedése nem sokat változna. A részecskerendszert a 2. listán (➲ 24. CD, Magazin/Pov-Ray könyvtár) látható módon méretezhetjük át, tehát úgy, hogy még a részecskerendszer meghatározásakor a PoV-Raynek a méretek megváltoztatására adunk utasítást.
A fenti kérdés megoldására más módszert is használhatunk: a gömböt készítsük nagyobb méretűre, és mielőtt még meghatároznánk a részecskerendszert, méretezzük is át. Így a részecskerendszer mérete már nem változik. Ezeket a megoldásokat csak akkor alkalmazhatjuk, ha a rendszert gömbszerű (`spherical_mapping`) vagy kockára történő leképezéssel készítjük el.
Amennyiben elsajátítjuk a `frequency` kulcsszó használatát, a tűzgömböt még élethűbbé varázsolhatjuk. Most sem érdemes a matematikai háttérrel foglalkoznunk, mert nem vinne közelebb a lehetőség gyakorlati használatához, elegendő annyit tudnunk, hogy ez az érték határozza meg, hogy a tárolóobjektumon belül a színek hányszor ismétlődjenek. Vigyáznunk kell azonban, mert ha a részecskerendszer színeit nem ismétlődőként adjuk meg (amikor az első és az utolsó érték megegyezik), a végeredményül kapott képen csúnya ugrásokat láthatunk.
Miután ilyen szép tüzet készítettünk (*2. kép*), megeshet, hogy az általunk alkotott képet valamilyen különleges hatással még tetszetősebbé szeretnénk varázsolni. Tegyük a képet egy kicsit szokatlanná: változtassuk meg a tűzgömb színét! Célunk, hogy a gömbön ne piros-sárga átmenetet lássunk, hanem zöld-pirosat. Ennek legegyszerűbb módja, hogy a részecskerendszer színét módosítjuk. Ekkor azonban azt fogjuk tapasztalni, hogy a gömb közepén nem a várva-várt zöld szín jelenik meg, hanem a zöld és piros keveréke: a sárga. Nem egészen ezt szerettük volna, de nem adhatjuk fel ilyen könnyen. Íme a kiváló alkalom a `glowing halo` megismerésére.

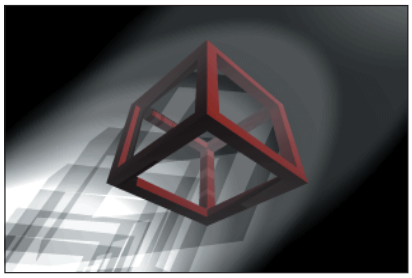
Ez a részecskerendszer-típus abban különbözik az előzőtől, hogy az egyes részecskék fényt bocsátanak ki – elkerülve ezzel a színek keveredését. Csak annyit kell az előző példánkon megváltoztatnunk, hogy az `emitting` kulcsszó helyett a `glowing`-ot használjuk, minden másban megegyezik az előbbiekben tárgyalt típussal, így korábbi ismereteinkre támaszkodva bátran kalandozhatunk a részecskék és részecskerendszerek világában.



2. kép Túl valósághűre sikeredett…

A következő egyszerű részecskerendszer a felhők és a párás környezet modellezésére szolgáló `attenuating halo`.
E típus azért alkalmas felhők leképezésére, mert a rajta áthaladó fény egy részét elnyeli, ugyanis a PoV-Ray a leképezés során egy adott képpont értékét nem a pontot körülvevő részecskék alapján számítja ki, hanem az objektumon áthaladó fénysugár mentén minden részecskét figyelembe vesz. Ezeket a színeket a `color_map` részben határozzuk meg. Most lássunk egy olyan példát, amit felhőink kiindulási alapjaként a későbbiekben is felhasználhatunk (2. lista).
A felhők általában nem vörös színűek, de példánkban a jobb láthatóság kedvéért piros-fehér átmenetet használunk, ugyanis fekete-fehér háttér előtt nem lenne célszerű fehér felhőket megjeleníteni.
Előbbi példákban csak a részecskerendszer méretét változtattuk, de most lássunk arra is megoldást, hogy a hordozóobjektumot méretezzük át. A tűzgömbnél alkalmazott minőségjavító megoldások (a fényesség növelése, rendezetlenség alkalmazása) az eredményt itt is vonzóbbá teszik, de a felhők általában nem gömbalakúak. Most a hordozóobjektumot kell átméreteznünk.
Ha valóban élethű felhőket szeretnénk előállítani, egy hordozóobjektumon belül több részecskerendszert is alkalmazhatunk. Nem kell mást tennünk, csak egymás után úgy meghatározni a rendszereket, hogy csak a helyzetükben különbözzenek egymástól. Természetesen ilyenkor a rendszerek alapjául szolgáló tárgyat is nagyobbra kell készítenünk.

A következő részecskerendszer a fénysugarak poros közegen történő áthaladásának megjelenítésére alkalmas. Gyakran láthatunk ilyet például a délutáni erdőben, amikor a fény átszűrődik a lombok között, vagy a padlásokon, amikor a napsugarak a cserepek közötti réseken szűrődnek be. Ez a részecskerendszer eléggé összetett, nem csupán elnyeli a fény egy részét, de az a benne található részecskéken szét is szóródik. Ennek eredményeképpen a tárgyon áthaladó fénysugarak láthatóvá válnak.
Ezt a részecskerendszert ugyancsak egy példán keresztül világíthatjuk meg a legjobban. Először meghatározunk egy irányított fényforrást és egy tárgyat, utóbbin fogjuk szemlélni a keresztülhaladó fényt.
Ezután létre kell hozni egy olyan tárgyat, ami a részecskerendszert fogja tartalmazni, jelen esetben erre a célra egy másik dobozt alkalmazunk. A részecskerendszer sűrűsége állandó, amit a rendszeren belül a `max_value` kulcsszóval határozhatunk meg. Ennek alapértéke 1, és amennyiben ezt a sűrűségfüggvényt használjuk, bármelyik leképezési mód (gömbszerű, kockára való



3. kép A kezdeti állapotok…

leképezés stb.) ugyanazt az eredményt adja. Lássuk, hogyan adhatjuk meg a részecskerendszert (7. lista ➲ 24. CD, Magazin/Pov-Ray könyvtár)!
Amint a *3. képen* is láthatjuk, az eredmény még nem tökéletes, mert a kép túl világos, a háttér alig látható és a por is túl sűrű.
A hordozótárgyként megadott kockában a részecskék sűrűsége állandó, alapértelmezetten pedig 1. Ez azt jelenti, hogy a színek meghatározásánál a részecskerendszer sűrűségét és színét csak az 1-es helyen lévő érték fogja befolyásolni.
A következő példában ezt az átlátszóságot 0,7-re állítjuk, így a por ritkább és az eredmény is szebb lesz.

```
...
...
    color_map {
```

```
      [ 0 <1, 1, 1, 1> ]
      [ 1 <1, 1, 1, 0.7> ]
    }
    samples 10
  }
...
...
```

Az eddigi módosítások ellenére még mindig akad egy kis gond az árnyékokkal. Tegyük egy kicsit elmosódottabbá az éles árnyékokat! Többféleképpen is megoldhatjuk: alkalmazhatunk véletlenszerű zajt a részecskerendszerben a `jitter` kulcsszó megadásával. A másik lehetőségünk a felül-mintavételezés, ekkor a nagyobb fényességváltozásokat finomítjuk (az `aa_treshold` és az `aa_level` kulcsszó segítségével).
A harmadik mód, amikor a teljes képre magasabb mintavételezési értéket alkalmazunk. Mivel ez utóbbi a leglassabb eljárás, először a többi módszert próbáljuk ki. A részecskerendszer meghatározásán belül általánosságban a véletlen zajt és a helyi felül-mintavételezést használjuk a következő módon:

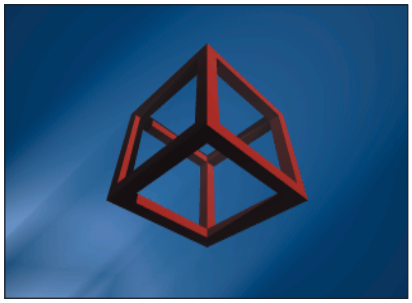
```
...
...
    }
    samples 10
    aa_level 3
    aa_threshold 0.2
    jitter 0.1
    }
  }
...
...
```

Most már szinte tökéletes a poron áthaladó fénysugár megjelenítése, de nagyon ritkán találkozunk olyan hellyel, ahol így áll a levegő, és a por ennyire egyenletesen oszlik el a térben. Kavarjunk egy kis szellőt a `turbulence` utasítás alkalmazásával, és máris elégedettek lehetünk az eredménnyel:

```
...
...   box_mapping
    linear
    turbulence 1
    color_map {
...
...
```

Vegyük észre, hogy nem használtuk az állandó sűrűséget meghatározó `constant` sűrűségfüggvényt, hanem helyette a `linear` kulcsszó meghatározta egyenletes eloszlást adtuk meg.

Ennek oka, hogy állandó sűrűségű térben nem lenne értelme a véletlenszerű változtatásoknak, a részecskesűrűség nem változna. Megjegyzendő, hogy a `turbulence` érték típusa vektor; a felhasználásával látványos hatásokat érhetünk el, ha az egyik irányban nagyobb értéket adunk meg, mint a másik két koordinátatengely mentén. Így készíthetünk például vízesést vagy más áramló rendszert.



4. kép Fény a porban…

Természetesen a por számára nemcsak fehér és szürke színeket adhatunk meg, hanem az alábbi részlet alapján akár a fehértől indulva – a képzelőerőnk szabta határokig – bármilyen színt. Fontos olyan színértékeket megadni, amelyek bizonyos színeket kiszűrnek, ezt a `rgbf` kulcsszóval tehetjük meg. Mivel azonban a részecskerendszerben a színeknek átlátszóknak is kell lenniük, a színek meghatározása során inkább a `rgbft` szót használjuk.

```
...
    color_map {
      [ 0 color rgbft <1, 0,
        ➥0, 0.5, 1.0> ]
      [ 1 color rgbft <1, 0,
        ➥0, 0.5, 0.7> ]
    }
...
...
```

Mielőtt a végére érnénk a részecskerendszerekkel való ismerkedésnek, fel kell hívnom a figyelmet néhány dologra: a részecskerendszert minden tárgyhoz a következő formában adjuk meg:

```
OBJETKUM {
  texture {
    pigment {...}
    normal {...}
    finish {...}
    halo {...}
  }
  hollow
}
```

Nem használhatók a `pigment`, `color_map`, `pigment_map`, `texture_map` és `material_map` utasításokon belül. Amennyiben többrétegű mintázatot szeretnénk használni, a részecskerendszert mindig a legalsó rétegben kell meghatároznunk, mely rétegnek természetesen átlátszónak kell lennie. Szintén ne feledkezzünk meg róla, hogy egymást átfedő tárolóobjektumok esetében a PoV-Ray az eredményt nem képes megfelelő módon kiszámítani. Ilyenkor minden tárolóobjektumot a többitől függetlenül számol ki, és az eredmény összeadódik. Az ebből származó hibák elkerülhetők, ha megfelelően nagy méretű tárolóobjektumot adunk meg.
További hiányosság, hogy a többféle színtérképpel (`color_map`) létrehozott `attenuating` típusú részecskerendszer, amelyet a felhők készítésénél tárgyaltunk, jelenleg nem használható.
Amint azt a leírás elején említettem, a kamera látóterében lévő objektumoknak üregesnek kell lenniük, ezt a `hollow` kulcsszó teszi lehetővé.
Fontos megjegyeznünk, hogy a `scale` kulcsszót a megfelelő helyen kell alkalmaznunk. Amennyiben a hordozó tárgyat át szeretnénk méretezni, például akkor, amikor a részecskéket rendezetlenné (`turbulence` kulcsszó) tesszük, az átméretezést még a rendszer meghatározása előtt el kell végezni; míg ha a részecskerendszer méretét szeretnénk megváltoztatni, a `scale` utasítást a rendszer meghatározásán belül kell használnunk.
Végül az ismétlés kedvéért jegyezzük meg, hogy a rendezetlenség nincs hatással az állandó sűrűségű rendszerre, tehát a `turbulence` és a `constant` kulcsszó együttes alkalmazása nincs hatással a részecskék eloszlásváltozására.
Végül nézzük meg a korábban elkezdett tárgyat, amely most poros térben lebeg, és szabadon alkalmazzuk rá új ismereteinket!

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.

Bevezetés a Tkinter használatába (2. rész)

Egy számológép elkészítése során is fedezhetünk fel új dolgokat: tudtad, hogyha véletlenül meglököd az egeredet, máris folyamatok egész sorát indíthatod el?

Tkinter-tanfolyamunk második részéhez érkeztünk. Az előző részben már láthattuk, hogyan néz ki egy egyszerű Tkinter-alkalmazás, ezúttal pedig egy kicsit bonyolultabb feladattal, egy számológépes példával folytatjuk. A „Szia Világ!"-os példa sokat elmond, amikor egy programnyelv vagy rendszer alapjaival ismerkedünk, a Tkinter viszont túlmutat ezen; és mivel terjedőben van a szokás, hogy a grafikus alkalmazások fejlesztésére szánt rendszereket egy-egy számológépes példán keresztül ismertetik meg a felhasználókkal, tegyünk így mi is. Eközben fény derül olyan titkokra, minthogy miként tudunk egy szövegbeviteli mezőt programsorból módosítani, de az is kiderül, mi minden történik olyankor, amikor látszólag nem történik semmi, csak az egerünkkel bóklászunk a képernyőn. Vágjunk bele!

Ismerkedés a számológéppel

Számológépet már mindenki látott, ezért a feladattal nagyjából tisztában vagyunk. Vegyük sorra, mi minden szükséges ahhoz, hogy az elképzeléseinket valóra váltva egy egyszerű számológép jelenjen meg a képernyőn, amely összead és kivon, továbbá a másik két alapművelettel is tisztában van. Ha a számítógép „fejével" gondolkodunk, mindjárt elakadunk, hiszen a „szegény" gép nem tudja, hogyan néz ki egy számológép, tehát pontról pontra mindent el kell neki magyaráznunk. Meg kell mondanunk például, hogy a gombok ne „csak úgy" megjelenjenek a képernyőn, hanem meg is lehessen őket nyomni; és azt is meg kell értetnünk, hogy olyankor mi történjen, ha meg is nyomjuk ezeket a gombokat. Tudnia kell, hol legyenek az egyes gombok, a többiről nem is beszélve. Nem olyan bonyolult ám ez, csak elsőre szokatlan, hiszen ezúttal olyan partnerrel akadtunk össze, aki nem biztos, hogy mindent azonnal ugyanúgy gondol, ahogyan mi elvárnánk.

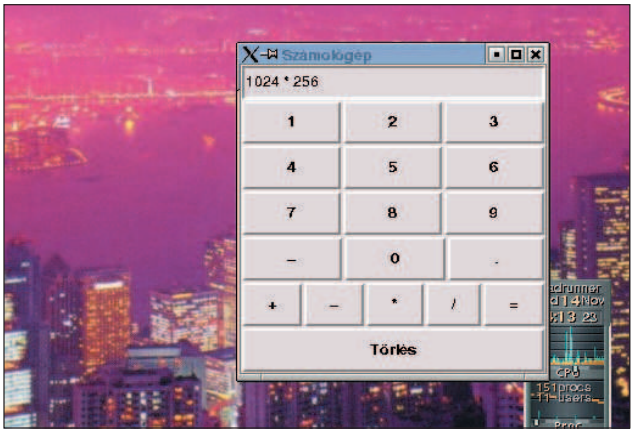
Első lépésben tehát az úgynevezett felhasználói felületet szükséges felépítenünk: ki kell találnunk, hogy milyen elemeket szeretnénk kitenni a képernyőre, és hol legyen a helyük. Esetünkben gombokra és kijelzőre lesz szükség. A következő lépésben pedig azt kell kifundálnunk, hogy az egyes gombokra kattintva mi történjék. Végül az sem árt, ha arra is felkészülünk, hogy mi legyen olyankor, ha a monitor előtt ülő felhasználó olyan dolgokat cselekszik, amelyre programunk nincs felkészülve: egyszerűen lépjünk ki valamilyen hibával a programból, ne is vegyük észre a hibát vagy tudassuk a felhasználóval, hogy rosszul csinált valamit?

Amennyiben mindezt kigondoltuk, a programtervezés nagy részén már túl is vagyunk, innentől már csak ujjgyakorlat az egész.

Az első nekifutás

Mint fentebb már említettem, számológépünkhöz két dologra lesz szükségünk: egy kijelzőre és sok-sok apró pici gombra, továbbá egy ablakra, ahol mindezt elhelyezhetjük. Az előző részben láthattuk, hogy egy gomb létrehozása csupán egy pillanatig tart: létrehozunk egy gombpéldányt és valamelyik felületkezelővel kitesszük a képernyőre. Szemléltetve:

```
w = Button(root, text= sz veg ,
           command=eljÆrÆs)
w.pack(side=LEFT, expand=YES, fill=BOTH)
```



Munkánk gyümölcse

Lambda kifejezések

Egy `lambda` kifejezés egy szokványos `def` függvénytől annyiban különbözik, hogy olyan helyeken is előfordulhat, ahol a `def` nem. Tartalma csak valamilyen egyszerű kifejezés lehet, bonyolultabb szerkezetek, mint `if` vagy `for`, nem. A `lambda` kifejezéseket elsősorban visszahívó függvényeknél használják. Visszatérési értéke mindig egy függvényobjektum, amelyet egy változóhoz rendelve hívhatunk meg. Az alábbi példa lefutása után a *b(18)* visszatérési értéke 36:

```
b = lambda a: a * 2
print b(18)
```

Minden `lambda` kifejezés saját névtérrel bír, amely nem azonos a hívó programrész névterével. Ezt a gondot egy trükkel szokták megoldani: a `lambda` értéklistájához azokat a változókat teszik hozzá, amelyekre a programrész névteréből szükségünk van. Számológépprogramunkban erre is találunk példát.

Az első sorban dől el, hogy mi lesz a gombunkra írva, és mi fog történni, ha valaki véletlenül rákattint. A `w` változó innentől kezdve az általunk megálmodott gombra mutat, ez azonban egyelőre csak a memóriában létezik. Ahhoz, hogy valódi, kattintható és látható gomb váljék belőle, ki kell tennünk a képernyőre. Ezt a nagyon fontos és cseppet sem elhanyagolható feladatot a Packer nevű felületkezelőre bízzuk. Ennek a `pack()` eljárásnak a dolga a gomb képernyőre történő helyezése, egészen pontosan az alkalmazásunk ablakába, lehetőség szerint a bal oldalra igazítva.

Számológép

```
1. from Tkinter import *
2.
3. class Calculator(Frame):
4.   def __init__(self):
5.   Frame.__init__(self)
6.   self.pack(expand=YES, fill=BOTH)
7.   self.master.title('SzÆmol gØp')
8.   self.error = 0
9.   self.ans = 0
10.
11.  display = StringVar()
12.  disp = Entry(self, relief=SUNKEN,
           textvariable=display)
13.  disp.pack(side=TOP, expand=YES,
     fill=BOTH)
14.
15.  for key in ("123", "456", "789", "-0."):
16.    fkey = frame(self, TOP)
17.    for char in key:
18.      button(fkey, char, lambda w=display,
         c=char, s=self: s.setscreen(w, c,
         1))
19.
20.  fops = frame(self, TOP)
21.  for char in "+-*/=":
22.    if char == '=':
23.      btn = button(fops, char)
24.      btn.bind('<ButtonRelease-1>',
25.        lambda e, s=self, w=display:
          s.calc(w))
26.    else:
27.      btn = button(fops, char,
28.        lambda w=display,
          c=' %s '%char, s=self:
          s.setscreen(w, c, 0))
29.
30.  clearF = frame(self, BOTTOM)
31.  button(clearF, 'T rlØs', lambda
     w=display, s=self: s.clear(w))
32.
33.  def setscreen(self, w, c, clear):
34.    if self.error == 0:
35.      if self.ans == 1:
36.        self.ans = 0
37.        if clear:
38.          w.set('')
39.      w.set(w.get() + c)
40.    else:
41.      self.clear(w)
42.      w.set(c)
43.
44.  def clear(self, w):
45.    w.set('')
46.    self.error = 0
47.
48.  def calc(self, display):
49.    try:
50.      if self.error == 0:
51.        display.set('eval(display.get())')
52.        self.ans = 1
53.      else:
54.        display.set("kattints a torlesre")
55.    except:
56.      display.set("HIBA")
57.      self.error = 1;
58.
59. def frame(root, side):
60.  w = Frame(root)
61.  w.pack(side=side, expand=YES, fill=BOTH)
62.  return w
63.
64. def button(root, text, command=None):
65.  w = Button(root, text=text,
         command=command)
66.  w.pack(side=LEFT, expand=YES, fill=BOTH)
67.  return w
68.
69. if __name__ == '__main__':
70.  Calculator().mainloop()
```

Egy felületkezelőnek ennél azonban jóval több dologra kell ügyelnie. Kezdetben az ablak mérete pontosan akkora kell legyen, hogy minden elem, amelyet belepakoltunk, elférjen benne és látszódjék, hacsak másképpen nem rendelkeztünk. Amennyiben az ablakunk méretét növeljük, gombjainknak több hely áll rendelkezésére, mint amennyire szükségünk van, és ilyenkor ugyancsak a felületkezelő feladata, hogy megmondja, miképpen változzon meg az ablakban található gomb vagy bármilyen más elem mérete és elhelyezkedése, hogy a felhasználók igényeinek a legjobban megfeleljen. Ebben az esetben feltételezzük, hogyha a felhasználó növeli az ablak méretét, bizonyára nagyobb gombokra van szüksége, így gombunk tulajdonságai közé vesszük, hogy a rendelkezésre álló helyet mindkét irányban töltse ki (`fill=BOTH`), és nőjön együtt az ablakkal (`expand=YES`). A `side` értékkel mondhatjuk meg, hogy a Packer az ablak melyik oldalára igazítsa a gombunkat.

A Packeren kívül két másik felületkezelő is létezik: a Grid és a Placer. A Grid az ablakot rácsokra osztja, és nekünk csak azt kell megadnunk, hogy az egyes elemek mely rácspontokba kerüljenek. Ez a felületkezelő a legjobban talán a HTML-ből ismert TABLE-höz fogható. Használata a Packerhez hasonlóan könnyű és egyszerű, és igazából csak szokás kérdése, hogy melyiket kedveljük jobban – viszont kétségkívül igaz, hogy olyan feladatok is akadnak, amelyek az egyikkel vagy a másikkal könnyebben kivitelezhetők.

A felületkezelők feketebáránya a legegyszerűbb, mégis a legtöbb odafigyelést igénylő ***Placer***, amelynek segítségével elemeinket pontosan igazítva helyezhetjük el az ablakban. Fontos, hogy egy kereten (vagy ablakon) belül csak egyetlen felületkezelő használható! Ez józan ésszel is belátható, hiszen mindannyian az elemek elhelyezéséért felelősek. Tegyük fel, hogy egy ablakot már rácsokra osztottunk, ilyenkor már nem pakolgathatunk benne ide-oda bármit!

Eseménykezelés

Bármilyen rendszer alatt dolgozunk is, legyen az X vagy Windows, a háttérben az egér egyetlen elmozdulását is események egész sora követi. Így van ez mindennel: esemény keletkezik, ha lenyomunk egy gombot, ha kattintunk valahol, sőt még akkor is, ha véletlenül meglökjük az egeret. A futó rendszer dönti el, hogy a pillanatnyilag zajló esemény az alkalmazásunkra tartozik-e vagy teljességgel lényegtelen. Az eseményeknek három fő típusa van: a billentyűzettel, az egérkezeléssel és az ablak helyével, illetve méretével kapcsolatosak. Ezek közül a billentyűzettel összefüggő eseményfajta az egyetlen, amelynek elfogásához az ablakunknak kell az aktív ablaknak, azaz a beviteli fókusszal bíró ablaknak lennie.
Egy eseményt kétféle módon lehet elfogni: vagy közvetlenül a `bind()` eljárással, vagy – amennyiben gombról van szó – a ***command*** tulajdonság megadásával.

```
btn.bind('<ButtonRelease-1>', eseménykezelő)
```

Ebben a példában gombunk objektumához egy eseményt rendelünk. Kikötjük, hogy amennyiben a gombunkon valaki a bal oldali egérgombot nyomja le, hajtsa végre az eseménykezelő által hivatkozott eljárást. Ennek legelső értéke kötelező érvényű, a `bind()` ezen keresztül tudatja az eljárással, hogy milyen esemény történt valójában. Ha függvényünket a ***command*** tulajdonságon keresztül hívatjuk vissza, erre a kötelező értékre nincs szükség.
Az alábbiakban felsorolunk néhány fontosabb eseményazonosítót:

ANY-ENTER	Az egérmutató az elem területére lépett.
BUTTON-1	Az egyes egérgombot lenyomták az objektum területén.
BUTTONRELEASE-2	Az objektum területén a kettes egérgombot felengedték.
KEYPRESS	Lenyomtak egy billentyűt.
CONTROL-SHIFT-F1	Az adott elemen lenyomták az adott billentyűkombinációt.
CONFIGURE	Az ablak mérete vagy helyzete megváltozott.
FOCUSIN	Ablakunk lett az aktív ablak.
DESTROY	Programunk bezárás alatt van.
A	Lenyomott A betű.

Ha gyakran végrehajtódó eseményhez rendelünk eseménykezelőt, ügyeljünk rá, hogy egy nagyobb függvény nagyon lefoglalhatja a processzorunkat, ezért igyekezzük elkerülni.
Az eseménykezelők egy alkalmazásban több szinten is beállíthatók. Alapértelmezésben a beállítás csak egy adott elemre vonatkozik. Ezenkívül még három szint létezik: az alkalmazásszint, amely egy adott alkalmazás minden ablakára és elemére vonatkozik; az osztályszint, ami egy osztály összes kezdeti példányára vonatkozik; illetve a héjszint, amely a szülő alkalmazásablakra vonatkozik.
Egy-egy létrehozott esemény legelőször azon az elemen jelentkezik, amelyen az esemény keletkezett, és attól halad lefelé egészen az alkalmazásszintig, kivéve, ha közben valamelyik szinten úgy rendelkezünk, hogy az eseményt nem engedjük tovább.

Mindez a gyakorlatban

Most, amikor az elmélettel már nagyjából tisztában vagyunk, vessünk néhány pillantást számológépünk forráskódjára.
Ha a kódsorokat begépeljük és .PY kiterjesztéssel mentjük, a `python` paranccsal meghívva fárasztó gépelésünk eredményében már gyönyörködhetünk is.
Amennyiben közelebbi pillantást vetünk a programra, látható, hogy a gerincét a ***Calculator*** osztály alkotja, amelyben a lényeg igazából az `__init__()` eljárásba van sűrítve. Azt már tudjuk, hogy ez az az eljárás, amely objektumpéldányunk létrehozásakor önmagától lefut, ennek megfelelően a benne rejlők már az alkalmazás indulásakor végrehajtódnak. Az első ismeretlenbe a 12. sorban ütközünk, itt hozzuk létre alkalmazásunk kijelzőjét, amely valójában egy egyszerű szövegbeviteli mező, és a következő sorban található `pack()` eljárással tesszük ki a képernyőre. A `relief` értékkel a kijelző stílusát adhatjuk meg, ami ebben az esetben SUNKEN, azaz süllyesztett.
A `textvariable` értékkel azt a változót jelöljük ki, amelyet összekötünk a kijelzővel. Amennyiben a változón módosítunk, változik a kijelző tartalma, ami fordítva is igaznak fog bizonyulni.
A 15. sortól kezdődően alkalmazásunk gombjait rajzoljuk ki, a 17. sorban pedig a ***key*** karaktersorozatot bontjuk szét karakterekre, amelyeket kirajzolásuk után egy lambda kifejezésen keresztül a `setscreen()` eljárásra csatolunk vissza, így ha bármelyiket lenyomjuk, az adott érték a kijelzőn jelenik meg.
A 18. sorban meghívott `button()` eljárás csak hivatkozás egy alább bevezetett függvényre, amelyet nem szabad a ***Button*** osztállyal összekevernünk, utóbbi ugyanis egy objektumpéldánnyal térne vissza.
A 16. és 20. sorokban megint csak egy saját függvényen keresztül hozunk létre egy keretet. Mivel a Packer felületkezelőt használjuk, erre azért van szükség, hogy a Packer egyértelműen eldönthesse, hova kell az adott elemet helyeznie.
A 24. és 25. sorokban a `bind()` függvényt hívjuk meg, amellyel az egérgomblenyomás eseményéhez rendelünk hozzá egy `lambda` eljárást. Az eljárás első értéke (`e`) látszólag használaton kívüli, valójában viszont azért szükséges, mert a `bind()` ezen keresztül küldi el az `event` eseményváltozót.
Az 51. sorban az `eval()` függvényen keresztül a kijelző tartalmát kiértékeljük és az eredményt kiíratjuk.

Összegzés

Ebben a részben egy számológépes példán keresztül a Tkinter alapvető lehetőségeivel ismerkedtünk meg, amelyek felhasználásával egyszerű alkalmazások készítésére leszünk képesek. A példaprogram részeit begépelés után ajánlatos módosítgatni vagy akár bővíteni. Így biztosak lehetünk benne, hogy a frissen elsajátított ismereteket nem felejtjük el azonnal, és később is fel tudjuk használni.

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

A Python hivatalos honlapja ➲ http://www.python.org/
Tkinter- tanfolyam
➲ http://www.pythonware.com/library/tkinter/introduction/

Az ablakkezelők és a Linux

Telepítéskor a legtöbb Linux-változat két ablakkezelőt helyez előtérbe: a KDE-t és a Gnome-t. Az összes többi ablakkezelő általában csak lehetőségként választható, amely sajnálatos módon azt eredményezi, hogy a felhasználók többsége – főleg a kezdők – nem, vagy csak kis mértékben próbálkozik megismerkedni a többivel. Néhány éve, amikor Linuxszal kezdtem foglalkozni (RedHat 5.2-sel), ha jól emlékszem, a KDE és a Gnome még egyáltalán nem volt üzembiztosnak nevezhető, ezért nem is választhattam őket. Akkoriban a manapság szinte ismeretlen fvwm ablakkezelő és az AfterStep, valamint a WindowMaker képviselték az elfogadható választást. Azóta több, a témával kapcsolatos fordítást is készítettem a levelezőlistákon megismert sorstársakkal. A legtöbb Linux-változat esetén néhány egyszerűbb ablakkezelő külön is választható. Jelenleg a Debian SID-ben található alapablakkezelőket szeretném bemutatni, terjedelmi és egyéb okokból kifolyólag azonban csak röviden. A cikksorozat első lépésben az igen egyszerű felépítésű, eszközkészlet nélküli ablakkezelők ismertetését veszi célba. Ehhez a Debian SID-változata szolgáltatta az alapot, amely a ***táblázatban*** látható ablakkezelőket tartalmazza.
A szóban forgó ablakkezelőkről, mint említettem, helyenként csak meglehetősen szűkszavúan fogok nyilatkozni, mint látni fogjuk részben azért, mert csekély mozgásteret engednek.

Az alapszintű ablakkezelők

Ebbe a csoportba azokat a grafikus felületeket soroltam, amelyek kinézet, tudás, valamint kezelhetőség tekintetében is igen puritánok.

A twm

A `twm` az egyik legegyszerűbb ablakkezelő, hiszen indítás után csupán egyetlen menüt tudunk megjeleníteni, jó esetben talán egy ikonkezelőt is. Az ablakkezelő igényeinknek megfelelő átformálását csakis valamilyen szövegszerkesztő program segítségével végezhetjük el. A beállítóállományok nálam két helyen találhatók: a */etc/X11/twm/system.twmrc*-ben (ez rendszerszintű fájl, amelyben a menü központilag frissül), valamint a *~/.twmrc* (ez pedig a saját beállításaimat tartalmazó fájl).
A helyi fájl csak a felhasználó számára használható beállításokat tartalmazza. Amennyiben a *~/.twmrc* fájl létezik, a rendszerszintű beállításokat tartalmazó fájl nem kerül feldolgozásra. Következzék néhány fontosabb twmrc-beállítás:

- a fájl elején található sorok a menü és az ablakfejlécek betűkészleteinek típusát tartalmazzák:
  ```
  TitleFont "-adobe-helvetica-bold-r-normal
  ➥--*-140-*-*-*-*-*-*"
  ```
- a fájl tartalmazza a menüket. Fontos, hogy először a menü könyvtárszerkezetét kell meghatároznunk, és csak ezután térhetünk rá az egyszerű menüpontok megadására:
  ```
  "Xcalc" f.exec "xcalc &"
  ```

A *twmrc* fájl felépítéséről további adatokat a `man twm` parancs kiadásával kaphatunk. A `twm` megbízható, gyors, üzembiztos ablakkezelő, amely az eddigiek során számtalan ablakkezelő alapjául szolgált.

9wm

Ha lehet az egyszerűséget tovább fokozni, a `9wm` még az előzőnél is egyszerűbb ablakkezelő. A grafikus felület indítása után a menüt az egér jobb gombjával érhetjük el, és öt elemet tartalmaz.
New – új X-terminálablakot nyit meg.
Reshape – feladata az ablakok átméretezése (az egér jobb gombjával használhatjuk, az első kattintással jelöljük ki az átméretezendő ablakot, a másodikkal adjuk meg az egyik sarkát, majd egy újabb kattintással az adott ablak átlós sarkát).
Move – az ablakok mozgatására szolgál, szintén az egér jobb gombjával használható. Az első kattintás és az egérgomb nyomva tartása az ablak kijelölésére szolgál, amit az elmozgatása után az egérgombot elengedve elhelyezhetünk.
Delete – bezárja az ablakot.
Hide – az ablak elrejtésére szolgál. Az elrejtett ablakok e jobbgombos menü további pontjaiként fognak megjelenni.
Az ablakkezelő bezárása a CTRL+ALT+BACKSPACE billentyűkkel vagy a `9wm exit` parancs X-terminálon történő kiadásával zajlik. Szintén megbízható és üzembiztos ablakkezelő.

aewm

Egyszerű ablakkezelő. Indítás után csak egy X-terminálablak jelenik meg, amelyről több alkalmazást is el tudunk indítani, például az alábbi paranccsal:

```
rxvt &
```

Az `&` jel visszaadja a parancsértelmezőnek a vezérlést, így több parancs is elindítható egy terminálról.
A fő ablakra kattintva a jobb egérgombbal újabb X-terminálok indíthatók el. A középső gombbal az asztalra kattintva egy ablaklistát hívhatunk elő. A megnyitott ablakokat az ablakok fejlécének jobb szélén lévő négyzetre kattintva a bal egérgombbal zárhatjuk be, és a jobb egérgombbal ugyanoda kattintva rejthetjük el.
A bal egérgombbal a fejlécre kattintva előtérbe hozhatjuk az ablakot, a jobbal pedig a háttérbe küldhetjük, a középső egérgomb folyamatos lenyomásával ugyanott mozgatni is tudjuk.
A CTRL+ALT+BACKSPACE billentyűkkel léphetünk ki az `aewm`-ből. Szintén megbízható ablakkezelő.

pwm

A `pwm` a `twm`-hez hasonlóan képes a rendszermenü használatára, és a beállítási lehetőségei is hasonlóak hozzá.
Indításkor lehetőségként megadható a pillanatnyi beállításokat tartalmazó fájl (`-pwm -cfgfile beÆll t fÆjl`).

larswm

Egyszerű felépítésű ablakkezelő, ennek megfelelően a kezelőparancsai is meglehetősen egyértelműek.
Fontos megjegyeznem, hogy mielőtt ebben az ablakkezelőben bárminek nekikezdenénk, másoljuk át a leírásban található *.larswm* fájlt a saját könyvtárunkba, és alaposan tanulmányozzuk át a **.ps.gz* fájlokat (szintén a leírásban lelhetők fel), amelyek az egérhasználatról és a billentyűkombinációkról nyújtanak tájékoztatást.
Amennyiben ezt a „műveletsorozatot” kihagyjuk, igen kevés

lehetőséget és játékteret hagyunk magunknak: mindössze a virtuális képernyőket váltogathatjuk az asztal alján található asztalváltó sávval.
A megnyitott ablakokat az előzőekben már szóba került PostScript-fájlok tartalmazta leírásokban található billentyűkombinációkkal is kezelni tudjuk. A `laswm` e téren számos lehetőséget nyújt.
A kilépés a CTRL+ALT+BACKSPACE billentyűkkel történik.

A Debian-Sidben található alapablakkezelők

twm	phluid	vtvm
9wm	pwm	qwm
amiwm	ratpoison	uwm
aewm	gwml	wm2
ion	ctwm	flwm
larswm	qvwm	failsafe

ctwm
Szintén a `twm`-hez hasonló tudású ablakkezelő. Leírását úgyszintén még indítás előtt célszerű elolvasni. Ezenkívül ajánlom a leírás mellett található példafájlok ***.twmrc*** valamelyikének saját könyvtárunkba történő másolását ***.twmrc*** néven, nélküle ugyanis nagyon kevés eszköz fog a rendelkezésünkre állni. Én a ***lynx.twmrc*** fájlt próbáltam ki, amivel már kellemes eszközökre tettem szert. Az így kapott beállításokkal elérhetjük, hogy a bal egérgombbal kattintva alkalmazásmenü bukkan fel, a középső gomb használata esetén az ablakkezelő vezérlőmenü (kilépés stb.) jelenik meg, a jobb egérgombbal kattintva pedig az ablakok kezelését segítő menü (ablakbezárás, képernyőanimáció) áll a rendelkezésünkre. Szomorú tapasztalat, hogy ez volt az egyetlen ablakkezelő, amely esetenként nem érzékelte az egér használatát.

vtvm
Első ránézésre szinte teljesen azonos a `twm`-mel: mind a menük, mind a beállítások felépítése megegyezik. A beállításokat a ***.vtvmrc*** fájl tartalmazza a sajátkönyvtárunkban. A rendszerszintű beállítások a ***/etc/X11/rtvm*** könyvtárban szerepelnek.

gwm
Alapvető képernyőkezelő program. A Debianban található változat nem tartalmaz beállítóeszközöket.

uwm
Saját menüvel is rendelkezik, amelyet mi magunk is szerkeszteni tudunk. A teljes rendszerre érvényes beállításokat a ***/etc/X11/ude*** tartalmazza (furcsa, hogy az `uwm` nevű ablakkezelő beállítófájljai az ***ude*** könyvtárba kerültek).

wm2
Alapszintű ablakkezelő, hiszen csak X-terminálok indítására alkalmas, illetve ikonná változtatásukra és újbóli megjelenítésükre képes.

flwm
A Debian saját menüjét veszi át. Beállítási lehetőségei igen szegényesek. Egyetlen saját lehetőséget tartalmazott: több virtuális képernyőt tudunk vele kezelni.

failsafe
Nem is tudom igazán eldönteni, hogy a failsafe külön ablakkezelőnek minősíthető-e, hiszen az X feltelepítésekor önműködően feltevődik a rendszerre. Önálló beállításokkal nem rendelkezik. Indítás után egy fejléc nélküli X-terminál nyílik meg, amelyből különböző alkalmazásokat indíthattunk el. Én annak idején a StarOffice 5.2 indításakor alkalmaztam, hiszen ha teljes képernyőn használtam, akkor az apró eltéréseket leszámítva Windows-szerű felületet adott.

Egy lépcsővel feljebb
Ebbe a csoportba azokat az ablakkezelőket soroltam, amelyek némiképpen egyediek, tehát valamilyen önálló ötletet tartalmaznak – és ennek köszönhetően az előbb fölsoroltaknál bizonyos szempontból jobbak. Természetesen ezek sem közelítik meg a komolyabb ablakkezelők szintjét, de a megfelelő helyen alkalmazva őket igen hasznosak lehetnek a felhasználók számára.

amiwm
A leírás alapján ez a képernyőkezelő felépítésében az Amiga ablakkezelőjére hasonlít. Ezt sajnos nem tudom eldönteni, mivel az Amiga felületét nem ismerem, de mindenképp érdekes felületet hoz létre, leginkább a MacOs-éra emlékeztetett (természetesen jóval egyszerűbb kivitelben). Az egér jobb gombjával a képernyő bal felső sarkában kattintva több menüt is kapunk, ezek közül a legelsőben található egy parancssor begépelését lehetővé tevő ablak. Ebből tudjuk elindítani az alkalmazásokat (ugyanitt lelhető fel az ablakkezelőből kiléptető ***Exit...*** menüpont is). Az így létrejövő ablakok teljes képernyőssé vagy ikonméretűvé tehetők, illetve az ablakok jobb felső sarkában lévő három ikonnal a háttérbe küldhetők. Az ablakok a bal felső sarokban szereplő ikonnal zárhatók be.

ion
Első látásra ez az ablakkezelő is nagyon egyszerűnek látszik. Amennyiben elolvassuk a súgóit, a leírását, illetve betekintünk a bennük hivatkozott fájlokba (***/etc/ion/***), rájöhetünk, a látszat igenis csal: az `ion` ugyanis számos billentyűkombinációt tartalmaz. Az általa megjelenített képernyő igen sajátos, mert minden ablakot teljes képernyős ablakként jelenítünk meg, még a legutolsó beviteli ablakot is. Néhány fontosabb billentyű a könnyed használathoz:

F1	a súgó-oldal kiválasztása és megtekintése;
Mod*+F1	a súgó-oldal kiválasztása és megtekintése;
F2	az X-terminálemulátor indítása;
F3	parancsfuttatás;
Mod*+F3	a lehetőség indítása;
F4	SSH-kapcsolat létrehozása valamilyen kiszolgálóval;
F5	fájl szerkesztése;
F6	a fájlnézegető indítása;
F9	munkaterület készítése, amennyiben már létezik, átlépés rá;
Mod*+F9	a munkaterület lekérdezése;
F11	megerősítés után újra azt kérdezi tőlünk, hogy újraindítsa-e az `ion`-t;
F12	megerősítés után újra azt kérdezi tőlünk, hogy kilépjünk-e az `ion`-ból.
*	(Módosító billentyű, ALT vagy CTRL változatfüggő.)

Nagyon zavaró, hogy a normál billentyűk is tartalmaznak ablakkezelő szolgáltatásokat, ami sok esetben ütközik az indított alkalmazással. Ez volt az egyik legegyedibb ablakkezelő.
Folytatjuk.

Tóth Béla (tothb1@freemail.hu)
Nős, két gyermek büszke atyja. Dolgozott földmérőként, majd térinformatikus szakmérnöki képesítést szerzett. Egyaránt otthonosan mozog a CAD és a térinformatikai programokban, valamint a DOS- és Windows-alkalmazásokban. Legkedveltebb elfoglaltsága már két és fél éve a Linux.

Váltás PostgeSQL adatbázisrendszerre

Néhány hasznos tanács arra vonatkozóan, hogy miként cserélhetjük le meglévő Microsoft Access adatbázisrendszereinket.

Egyre több vállalat kezd nyílt forrású üzleti rendszerekkel foglalkozni, sokan közülük teljes rendszert igyekeznek kiépíteni, amely a vásárlók számára hozzáférhető webes felülettől az alapként szolgáló adatbázisig terjed. Legtöbb esetben a Linux-PHP-Apache hármast, illetve egy nyílt forrású adatbázist (MySQL-t vagy PostgreSQL-t) foglal magában. A PostgreSQL népszerűsége főleg az utóbbi időben növekedett meg, hiszen mára a program elérte azt a minőséget, amelynél már megbízhatónak és jól használhatónak mondható. Több nagyvállalat biztosít állandóan elérhető támogatást, közöttük a RedHat is. Írásomban azt szeretném megmutatni, hogy mire számíthatunk, ha a Microsoft Accessről nyílt forrású adatbázisrendszerre (itt és most a BSD-típusú felhasználási szerződéssel rendelkező PostgreSQL-re) térünk át.

Michael Calabrese, a Bike Friday nevű kerékpárgyártó cég adatrendszereinek felelőse nemrégiben a változás mellett döntött. A Bike Friday egy villámgyorsan növekedő, túra- és hegyi kerékpárok gyártásával foglalkozó cég, székhelyük az Oregon állambeli Eugene-ben található. A vállalatnál PostgreSQL adatbázisban rögzítik az eladásokkal, a gyártással és a vásárlói támogatással kapcsolatos összes adatot. Calabrese mostanában épp a cég e-üzleti rendszerét szándékozik ingyenes programokra lecserélni: Linuxra, Apache-ra és PostgreSQL-re. Egyelőre azonban úgy tervezi, hogy megtartja a munkaállomásokon használt Microsoft Access 97-et annak érdekében, hogy az adatbázis lecserélése miatt szükséges szünet a lehető legrövidebb ideig tartson. Calabrese szerint: „Ha a munkaállomásokon használt kezelőprogramokat nem akarjuk megtartani, egyszerű a dolgunk: csupán le kell futtatni az átalakítóprogramokat és elkezdhetjük megírni az új kezelőrendszert. Ha a kezelőrendszer az Access, amelyet továbbra is szeretnénk használni a PostgreSQL adatbázissal, akkor a fejlődés új irányait úgy jelöltük ki, hogy az induláshoz nincs szükség az egész rendszer átalakítására. A kezelőfelület befagyasztása után nyugodtan számoljunk egy évet az átalakításra. Ha a változtatásokat fokozatosan vezetjük be, megmarad a választási lehetőség, hogy egy-egy új lehetőséget Accessben vagy PostgreSQL-ben valósítunk meg”.

A változtatás eszközei

Amennyiben a Microsoft Open DataBase Connectivity (ODBC) meghajtókat betöltjük a PostgreSQL-sablonadatbázisba, máris megtettük az első lépést az Access és a PostgreSQL összeházasításához. Az együttműködéshez az alapvető átalakító eszközök mellett (lásd a ***További érdekességek*** című részt) néha további ODBC kiszolgálóoldali szolgáltatásokra is szüksége van. Ezeket az ***src/interfaces/odbc/odbc.sql*** fájlban találhatjuk meg. A PostgreSQL felületfüggetlen 4-es típusú Java adatbáziskapcsolati felület (JDBC) meghajtót is tartalmaz. Továbbá a C számára készült beágyazott felület (ECPG) is a PostgreSQL részét képezi. A telepítés végeztével Calabrese adatösszesítő eszközöket választott, például a pgAccesst – ez Windows és Unix változatban, illetve az exSQL 3.1-es változatában egyaránt hozzáférhető.

Miután a meglévő adatbázisokról a rendelkezésre álló eszközökkel (vagy a `pg_dumpall` segédprogrammal, vagy pedig a `pg_dump` és a `pg_dumpaccounts` együttes használatával) biztonsági mentést készítettünk és a telepítőt (***Installer***) is lefuttattuk, az adatok átalakításának első lépéseként az Accessben használatos, azonban a PostgreSQL-ben nem megengedett fájlnevek levadászása következett. Az Access meglehetősen szabadelvű, hiszen a fájlnevekben számos olyan karakter használatát lehetővé teszi, amelyeket más adatbázisrendszerek (Oracle, Sybase, PostgreSQL stb.) nem ismernek fel. Így a Bike Friday adatbázisában szereplő nevek jó részét a PostgreSQL által is kezelhető formájúra kellett alakítani, például az ***Order Detail*** táblázatból ***Order_Detail***, a ***Shipped***? mezőnevekből pedig ***Shipped*** vagy ***ShippedYN*** lett.

Az alapvető átalakítóeszközök minden meg nem engedett karaktert eltávolítanak. Ez súlyos gondokat okozhat, hiszen a mit sem sejtő munkaállomási kezelőprogramok (nem értve a helyzetet) minden további nélkül megszakíthatják az adatbázissal való kapcsolatot.
Calabrese azt javasolja, hogy ha a kezelőprogramokat megtartjuk, ne írjuk át az adatbázis neveit, vagy pedig az adatbázist és a kezelőfelületeket is párhuzamosan módosítsuk. Ő maga ezt úgy oldotta meg, hogy az adatbázisban és a kezelőprogramokban saját kezűleg, egyesével írta át a kérdéses karaktereket.

1. lista Meglehetősen pazarló lekérdezés

```
SELECT Orders.SalesRepID
Bikes_ColorsAvailable.Color,
OrderDetails.BuildABikeID, Orders.OrderDate,
BuildABike.ColorID,
Bikes_BasicFrameTypes.FrameName,
Reps.Rep, BuildABike.Frame, Orders.CustID,
Orders.OrderID, OrderDetails.PartID,
Orders.ShipDate,
BuildABike.BikeState
FROM (Reps RIGHT JOIN Orders ON)
Reps.RepID = Orders.SalesRepID) INNER JOIN
(((Bikes_BasicFrameTypes INNER JOIN
(OrderDetails INNER JOIN BuildABike ON
OrderDetails.BuildABikeID =
BuildABike.BuildABikeID) ON
Bikes_BasicFrameTypes.FrameTypeID =
BuildABike.FrameTypeID) INNER JOIN
Bikes_ColorsAvailable ON BuildABike.ColorID =
Bikes_ColorsAvailable.ColorID)
INNER JOIN Contacts_CurrentFrame_BABID ON
BuildABike.BuildABikeID =
Contacts_CurrentFrame_BABID.BuildABikeID)
On Orders.OrderID = OrderDEtails.OrderID
WHERE (((OrderDEtails.PartID)=6502))
ORDER BY Orders.OrderDate DESC;
```

2. lista Ugyanaz a lekérdezés, de jóval gyorsabban

```
SELECT salesrepid, color, od.buildabikeid, o.orderdate,
            bab.colorid, framename, reps.rep, bab.frame,
            o.custid, o.orderid, partid, o.shipdate, bab.bikestate
FROM        reps, bikes_basicframetypes b_bft, orders as o,
            orderdetails as od,
            bikes_colorsavailable as b_co_a,
            buildabike as bab
WHERE  repid = salesrepid
      AND   od.buildabikeid = bab.buildabikeid
      AND   b_bft.frametypeid = bab.frametypeid
      AND   b_co_a.colorid = bab.colorid
      AND   o.orderid = od.orderid
      AND   bab.custid=[Forms]![Customers/Contacts]![ID]
      AND         entrydate = (
                        SELECT MAX(entrydate)
                                    FROM buildabike as b2
                              WHERE bab.frame = b2.frame
                              )     ;
```

Ez rendben is volt, hiszen a távolabbi tervek között a kezelőprogramok lecserélése is szerepelt. Ami a legfontosabb: a munka ezen szakaszában nagyon fontos, hogy folyamatosan ellenőrizzük, vajon minden működik-e. Ha a meg nem engedett karaktereket kiküszöböltük, az adatok máris készen állnak az átalakításra.

Az adatok átalakítása

Ha azt tervezzük, hogy az Accest megtartva kezelőfelületnek, első lépésként csak az adatokat ültetjük át PostgreSQL alá, a pgAdmint rendkívül hasznos eszköznek fogjuk találni. Calabrese az exSQL módosított változatát is felhasználta annak meghatározásához, hogy az Access és a PostgreSQL hogyan kezeli a táblázatok közötti kapcsolatokat. A ➲ http://www.geocities.com/musica_6898/postgresaccess_home.html címen elérhető honlapján a nyilvánosság elé tárt változat egy héjprogramot futtat le, mely számos esetben módosítja a mezőtípus-átalakítás menetét (például az Access pénznemkezelését is). A Bike Friday Access kezelőfelülete a PostgreSQL számszerű decimális mezőit szövegmezőként értelmezte. Annak érdekében, hogy az Access ezeket helyesen kezelje, Calabrese a mezőket ***Float4***-típusúvá változtatta (a PostgreSQL így nevezi a négybájtos lebegőpontos számokat).

A kezelőfelület kipróbálása

Száznál is több táblázatával a Bike Friday kezelőfelülete meglehetősen összetett. A felhasználó szemszögéből nézve a Bike Friday több mint nyolcvan képernyőt használ a megrendelések beírásához, az alkatrésztáblázat megtekintéséhez, a gyártás ütemezéséhez és a raktárkészlet ellenőrzéséhez. Ezért Calabresének biztosnak kellett lennie abban, hogy a rendszert több tucat felhasználó sem fagyaszthatja le. A kipróbálás jó néhány hétig tartott, közben az SQL-lekérdezéseket is szükség szerint módosítani kellett oly módon, hogy az Access-oldalról vagy (ha ez nem volt megoldható) az adatbázis oldalán újraírták őket, egészen addig, amíg elfogadható sebességet kaptak.
Az *1.* és *2.* listában egy jellegzetes és egy gyorsított lekérdezést láthatunk.
A PostgreSQL-lekérdezések hatékonyabbá tétele általában a `Create index`, `vacuum`, `vacuum analyze`, `cluster` és `explain` parancsok használatával történik. Calabrese azonban figyelmeztet, hogy az Access alapértelmezetten a lekérdezéseket nem közvetlenül továbbítja, hanem mindig az általa leghatékonyabbnak ítélt formára alakítja őket. Calabrese a program okoskodását közvetlen lekérdezés használatával kerülte meg, amely közölte az Access-szel, hogy hozzá ne nyúljon a lekérdezéshez, egyszerűen csak továbbítsa a kiszolgáló felé.
A Bike Friday PostgreSQL-adatbázisának egyszerűsítése során Calabrese úgy ért el sebességnövekedést, hogy a lekérdezésekben az adatbázisból kisebb, pontosabban körülhatárolt adatokat vett ki. Százezer rendelési adat egyidejű lekérdezése

helyett a lekérdezést úgy alakította ki, hogy az adatbázisnak csak 2000 adatra kellett figyelnie. „Az Access buta – mondja Calabrese –, az összes rekordot kézbe veszi és minden egyes alkalommal az összeset átnézi. Ez rendkívül pazarló módszer. Jelenleg harminc alkalmazottunk van, és ha történetesen minden számítógéppel egyszerre próbálnák meg elérni az adatbázist, ez igen hamar nagyon komoly sebességcsökkenést okozna."

A PostgreSQL ellenőrzése

A változtatás következő lépése a lekérdezések hibaellenőrzése, ahol mindjárt két út közül választhatunk. Az egyik a PostgreSQL ODBC-meghajtójához tartozó ellenőrzőeszközök használata. A meghajtóval készíttethetünk egy naplófájlt, és amikor az Access SQL-parancsot küld, a PostgreSQL azt azonnal bevezeti a naplóba, mely a C meghajtó gyökérkönyvtárában található. Ezzel elcsíphetjük az Access olyan ügyetlen próbálkozásait, amikor mondjuk százezer sort próbál egyszerre behívni. Ebben az esetben például a lekérdezést ezer kisebbre bonthatjuk szét. Ez a napló igazából egy nyomkövetés, mely segítségével gyorsan kiszúrhatjuk, ha valami hiba lépett fel, mint ahogy itt is történt.

```
conn=86311032, query=' '

CONN ERROR: func=SQLDriverConnect,
desc='Error from CC_Connect',
errnum=105, errmsg='The database does not
exist on thet server or user authentication
failed.'
```

Azt is megtehetjük, hogyha az Access lekérdezést küld, és a rendszer leáll, a kiszolgálóoldal hibakezelési szintjét (***Debug level***) átállítva a kérelmet csak azért is kiolvassuk. A finomhangolás lényege, hogy minden egyes képernyőn végig kell haladnunk és a kérelmek egyszerűsítésével, összevonásával gyorsítanunk kell őket. Az SQL-t jól ismerők tudják, hogy a rendszer milyen összetett, így elmondhatjuk, hogy fáradságos munka elébe nézünk. Ha azonban a fejlesztésnek ezen a pontján elvégezzük a megfelelő ellenőrzéseket, rengeteg későbbi fejfájástól kímélhetjük meg magunkat.
Mielőtt a rendszer működését visszaállítanánk, a kipróbálás következik. Calabrese folyamatosan figyelte a Bike Friday adatbázisrendszerét, miközben az irodákban már használták a rendszert. „Nemcsak azt kell kipróbálnunk, hogy akadnak-e a kezelőfelületnek hibái, hanem azt is, hogy mekkorára kell a kiszolgálót terveznünk" – mondja Calabrese. Írt egy lekérdező héjprogramot is, amely a három fő gondot okozó részegység (processzor, merevlemezek, hálózat) terheltségét kíséri figyelemmel.
Calabrese programja a processzor kihasználtságát aszerint ellenőrizte, hogy a terhelés hány másodpercig maradt 100, 50 és 0 százalékos. A lemez adatátvitelének értékelését úgy végezte, hogy hány olvasási és írási művelet zajlik éppen, illetve mérte az ezek során átvitt kilobájtokat is. A hálózat terhelését a másodpercenkénti csomagszámmal és a másodpercenként átvitt bájtok számával írta le. Calabrese azt is javasolja, hogy a lezárt hálózati szakaszban végezzünk árasztásos pingelést (`ping -F`), így meghatározhatjuk, hogy a kiszolgáló mekkora terhelésnél akad meg. A memóriával egyszerű a helyzet: minél több van belőle, a PostgreSQL annál többet használ föl, és így annál gyorsabb lesz az adatbázis működése.
Természetesen az adatbázis sebességéről a felhasználók véleménye árulkodik leginkább. A lépésenkénti apró várakozási idők a valóságban hatalmas késésekké adódhatnak össze.

Minden vállalatnál, szervezetnél kialakul egy vélemény arról, mi számít lassúnak és mi elfogadhatóan gyorsnak. Így a rendszer főpróbája mindenképpen az lesz, amikor a felhasználók pár órás használat után kimondják a végítéletet: „Csigalassú" vagy „Hm, nem is olyan rossz".
Végül, miután a rendszer megfelelőnek találtatott, mi pedig a kezelőfelület minden hibáját kijavítottuk, máris nekiláthatunk egy nyílt forrású alapokra építkező e-üzleti rendszer kiépítésének.

Chris Volpe
(chris@macnet2.com)
New Hampshire-ben él és technológiai leírásokat készít.

mascTovábbi érdekességek

Bruce Momjian: PostgreSQL: Introduction and Concepts (ISBN: 0-201-70331-9, 44,95 dollár, 544 oldal)
➲ http://www.ca.postgresql.org/docs/awbook.html címen érhető el.
A gép és az alkatrészek, valamint a PostgreSQL összehangolásáról olvassuk el *Momjian* írását „PostgreSQL Performance Tuning" címmel
➲ http://www.linuxjournal.com/lj-issues/issue88/4791.html
F. Scott Barker: Microsoft Access 2000 Power Programming (ISBN: 0-672-31506-8, 49,99 dollár, 1332 oldal, CD-ROM)
PostgreSQL 7.1 kézikönyv
➲ http://www.ca.postgresql.org/users-lounge/docs/7.1/reference
Postgres GYK
➲ http://www.ca.postgresql.org/users-lounge/docs/#7.1
Postgres/Access GYK
➲ http://joelburton.com/resources/pgaccess
PostgreSQL-leírás és Data Migration Tools
➲ http://postgresql.crimelabs.net/users-lounge/docs
Migration Tools: a csomagban a *pgAdmin* (grafikus PostgreSQL-vezérlőfelület), a *phpPgAdmin* (webalapú eszköz a pgAdminhoz hasonló feladatokra) és a PsqlODBC Windows-meghajtó található. Ez utóbbi lehetővé teszi, hogy a PostgreSQL-adatbázist az ODBC-meghajtókon keresztül elérő Windows-alkalmazásokat írjunk.
Még egy érdekesség: az exSQL új, nyilvános változata is elérhető. Az exSQL nagyszerű PostgreSQL-átalakító eszköz. Az új változat, amely az indexeket és az idegen kulcsokat megbízhatóbban kezeli, a
➲ http://www.geocities.com/musica_6898/postgresaccess_home.html címről tölthető le. A PostgreSQL az idegen kulcsokat a táblázatok összekapcsolására és kapcsolataik kódolására használja. A *Michael Calabrese* által írt exSQL-változat módosítja azokat a szabályokat, melyek meghatározzák, hogy az egyes Access-mezőkből milyen PostgreSQL-mezőtípusok készíthetők. További egyszerűsítések és hibajavítások mellett ez a változat tartalmaz egy parancsfájlt, amely az Access egyik hibáját küszöböli ki: alapértelmezés szerint a program szövegessé alakítja a pénzmezőket. A parancsfájl felülbírálja ezt az alapértelmezést, így futás közben ebből hibák adódhatnak.

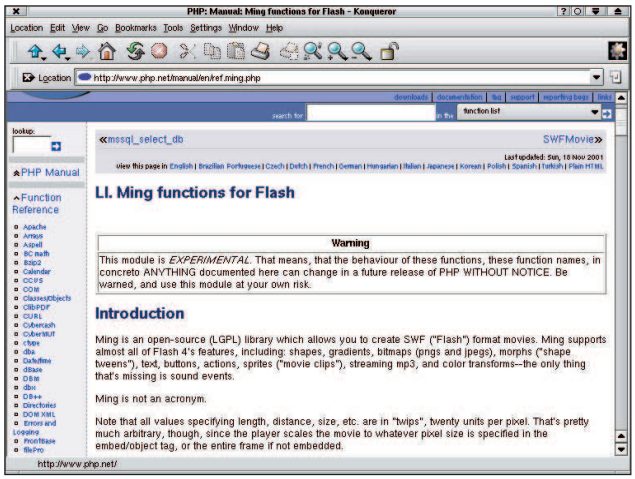
A PHP és a MING

Hogyan készítsünk weblapunkra röptében Flash-mozikat?

Napjainkban a Flash-féle pörgő-forgó csodát nehéz kikerülni a Világhálón. Nem is kell, hiszen a Flash-lejátszás az összes újabb linuxos böngészőben lehetséges. Kedvenc operációs rendszerünkön manapság az ilyen mozik létrehozása sem elérhetetlen cél. A ***MING*** könyvtár segítségével és PHP-támogatással weblapjainkat elláthatjuk röptében előállított animációkkal. Cikkünkben a MING Linuxra telepítéséről, valamint a PHP-vel történő összeházasításáról lesz szó. Emellett a cikk kínálta lehetőségekhez mérten igyekszem a MING működését is bemutatni.

A MING beszerzése és telepítése

A PHP MING kiegészítésének telepítését Debian Woody rendszeren a PHP 4.0.6-os változatához mutatom be. Kis módosításokkal természetesen más GNU/Linux-változatokon is üzembe helyezhető. Tekintettel arra, hogy a támogatás csak a 4.0.5-ös változattól került a PHP-ba, érdemes a gépünkön egy friss PHP-val próbálkozni.



1. kép A MING-függvények részletes leírása a PHP-kézikönyvben

A MING hivatalos weblapját megnyitva hamar szembesülhetünk vele, hogy nem vagyunk magunkra hagyatva. Innen a legfrissebb PHP-változatokhoz azonnal letölthetjük az előre lefordított ***php_ming.so*** modult. Jó esetben akár lusták is lehetünk, és a MING-kiegészítést fordítás nélkül beizzíthatjuk, csupán ezt a fájlt szükséges a PHP-kiterjesztéseket tartalmazó könyvtárba másolnunk. Ennek helyét könnyen megtudhatjuk, ha a parancssorban kiadjuk a `php-config --extension-dir` parancsot. A másolás mellett még a ***php.ini*** fájlba is bele kell nyúlnunk, és a következő bejegyzést kell beillesztenünk:

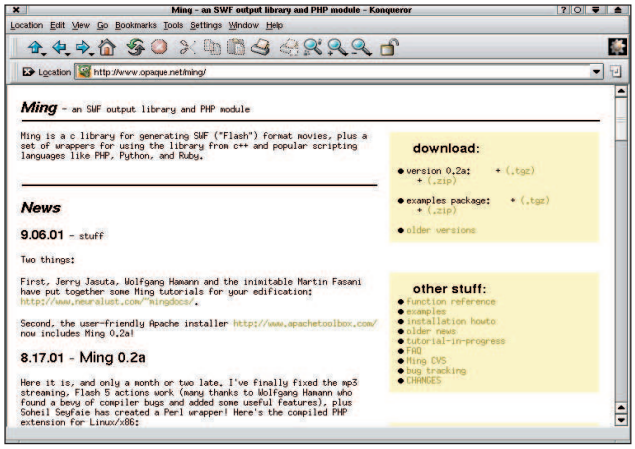
```
extension=php_ming.so
```

Amennyiben ezzel megvagyunk és minden egyezik, máris rendelkezünk a Flash-mozik létrehozásához szükséges eszköztárral. Ne feledkezzünk meg webkiszolgálónk újraindításáról, amennyiben az a beállításváltozások érvényre juttatásához szükséges.
Előfordulhat, hogy a folyamat nem ilyen egyszerűen zajlik le, ekkor sem kell kétségbe esni, a ***php_ming.so*** kiegészítést mi magunk is könnyen létrehozhatjuk. Ehhez először be kell szereznünk a MING forráskódját, majd le kell fordítanunk és telepítenünk (ehhez a parancsokat a MING forráskönyvtárából adjuk ki):

```
make
make install
```

Ezáltal a ***/usr/lib*** alá létrejön a ***limbing.so*** állomány, és egy ***ming.h*** is megfelelő helyére kerül a ***/usr/include*** könyvtárban. Ezáltal megteremtettük a feltételét, hogy a MING-támogatást a jelenlegi PHP-rendszerünkbe építsük. A további szükséges lépéseket már a PHP-forráskönyvtárából kell elvégeznünk:

```
./buildconf
./configure --with-ming <egyØb kapcsol k>
make
make install
```



2. kép A MING oldala ➲ http://www.opaque.net/ming/

A szükséges könyvtárat tehát létrehoztuk, és a helyére is került. A ***php.ini***-nek a fenti bejegyzéssel történő kiegészítése természetesen ekkor is szükséges.

Az ismerkedés

A nagy fejesugrás előtt nem árt néhány dolgot tisztázni: a pontosság és a jó nagyíthatóság érdekében bevezették a ***twip*** mértékegységet. Hogy értsük, mit is takar ez: húsz twip tesz ki egy képpontot. A mozi méretei, azon belül is minden elemnek a mérete, a távolságok mind ebben az egységben értelmezendők. Alapesetben tehát egy 200×200 képpontmérettel rendelkező mozihoz 4000×4000 twip méretű valódi munkafelület tartozik. A másik fontos tudnivaló, hogy a MING jelen pillanatban csak

az FDB-típusú betűkészletek megjelenítésére alkalmas. Ilyenek begyűjtésére a MING-forrás ***util*** alkönyvtárában található `makefdb` használható, először ezt sem árt lefordítanunk. Tekintsük át nagy léptékekben, hogyan is épül fel a MING-birodalom! A legtöbb PHP-kiegészítéssel ellentétben itt nem ömlesztett függvénykönyvtárat kapunk a nyakunkba, hanem 13 osztályt. Ezek mindegyike egy témát ölel fel, és a hozzá tartozó eljárásokat, függvényeket, tulajdonságokat hordozza magában. Lássuk, miből fogunk csipegetni!

`SWFShape()`	Ezzel hozhatjuk létre és rajzolhatjuk meg a különböző, a moziban megjelenő formákat.
`SWFBitmap()`	A moziba bevihető objektumokat JPEG-képekből hozhatjuk létre.
`SWFText()`	A szöveges elemek létrehozására való osztály.
`SWFTextField()`	Szöveges űrlapelemek létrehozásához.
`SWFSprite()`	Önálló animációs almozik hozhatók létre vele, amelyek saját időskálával rendelkeznek.
`SWFButton()`	Nyomógombok létrehozására szolgál.
`SWFFont()`	Különféle betűtípusokat tölthetünk be vele, valamint a szövegek megjelenítéséhez szükséges.
`SWFGradient()`	Színátmenetek létrehozására, továbbá a formák kifestésénél használható.
`SWFFill()`	Már meglévő kifestőobjektumok forgatására, mozgatására és átméretezésére szolgál.
`SWFDisplayItem()`	A moziban létrehozott, behúzott objektumokat itt tudjuk pörgetni, forgatni és nagyítgatni.
`SWFMorph()`	Alakjukat változtató látványelemek létrehozását teszi lehetővé.
`SWFAction()`	A Flash saját nyelvén írhatunk ActionScripteket.
`SWFMovie()`	A mozi maga: elemeket adhatunk hozzá, menthetjük vagy a kimenetre küldhetjük a tartalmát.

Az `SWFMovie()` osztályt mindig használni fogjuk, mert mind a mozi születésekor, mind a véglegesítésekor jelen van. Egy moziobjektumot a következő módon hozunk létre:

```
$mozi = new SWFMovie();
```

Innentől létezik is $mozi néven az objektumunk, ebbe szórjuk bele a mozgatnivaló elemeket. Ha létrehoztuk, nem árt néhány vele kapcsolatos dolgot beállítani:

```
$mozi->SetRate(20);
$mozi->SetDimension(4000,4000);
$mozi->SetBackground(0xff, 0xaa, 0x66);
```

A `SetRate()` által tudjuk megadni, hogy egy másodpercben hány képkockát játszunk le, ez lesz a teljes mozira érvényes beállítás. Megjegyzendő, hogy csak amolyan kívánatos értékről van szó, hiszen egy leterhelt gépen a képkockák megrajzolása a rendelkezésre álló időnél többet vehet igénybe. Ilyenkor egyszerű lassulásról van szó: a lejátszó nem hagy ki kockákat, csupán lassabban játssza le őket. Más eset áll fenn akkor, ha folyamatos MP3 zenei aláfestés is tartozik a mozinkhoz, mert

1. lista A gorillamozi.php – immár teljes pompájában

```
<?php

$mozi = new SWFMovie();

$mozi->SetRate(20.0);
$mozi->SetDimension(4000,4000);
$mozi->SetBackground(0xff, 0xaa, 0x66);

// A kimenetre k ldØs elıtt tudatnunk kell
// a b ngØszıvel az adathalmaz MIME-t pusÆt.

header('Content-type:
       ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

2. lista A gorilla.html – az első mozinkat beágyazó HTML-oldal

```
<html>
<head>
    <title>GorillÆk - rendezıi
      ➥vÆltozat</title>
</head>
<body bgcolor=#ffffff>

    <embed src=gorillamozi.php width=200
      ➥height=200></embed>

</body>
</html>
```

ilyenkor a lejátszónak biztosítania kell, hogy a hanglejátszás lehetőleg folyamatos legyen. Ezt a gondot képkockakihagyással küszöböli ki.

A befoglaló méreteket, azaz megjelenő munkaterületünk méretét a `SetDimension()` által adhatjuk meg. Mint korábban már említettem, itt nem képpontokban, hanem twipekben kell gondolkodnunk, azaz a fenti példa egy 200x200 képpont méretű Flash-objektumot hoz létre.

A méretek közül először a szélességet, másodjára a magasságot kell megadni. A mozinak kell háttérszín is. Ennek beállításához használatos a `SetBackground()`. A háttérszínt az RGB- (vörös, zöld, kék) összetevők keverésével tudjuk létrehozni. A színeket 3x8 biten képezhetjük le, vagyis az egyes értékek értéktartománya 0-tól 255-ig terjed, és csakis egész számokban adhatók meg. A példában éltem a PHP nyelv adta lehetőséggel, és az értékeket tizenhatos számrendszerben ábrázoltam (hiába no, én már csak hexadecimálisokban látom a színeket).

Örvendezzünk, ugyanis elkészítettük életünk első egész estés animációs filmjét! A címe „Gorillák a narancsos ködben" lehetne, tekintve az események letisztult egyszerűségét. Egy apróság hiányzik még: mozinkat láthatóvá is kellene tenni a közönség (legalábbis a böngészőnk) számára. Ehhez az

3. lista A haromszog.php már valami, de még nem mozog…

```
<?php

// A befoglal  mozi.

$mozi = new SWFMovie();
$mozi->SetDimension(6000,6000);
$mozi->SetBackground(0xff, 0xff, 0xff);

// a hÆromsz g megrajzolÆsa-

$valaki = new SWFShape();
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
      ➥0xe4, 0xec, 50));
$valaki->movePenTo(0,1600);
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);

// tegy k a moziba, Øs k ldj k a helyØre

$haromszog = $mozi->add($valaki);
$haromszog->move(3000,3000);
$mozi->nextFrame();

// kØsz van, mehet a vilÆg elØ...

header('Content-type:
      ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

`SWFMovie()` egy újabb fontos eljárását kell alkalmaznunk. Lássunk erre is példát!

```
$mozi->Output();
```

Munkánk eredményét ez a gyakorlatlanok számára így még eléggé emészthetetlen formában tálalja: krixkrax karakterek halmazaként. Mielőtt még e furcsa betűkben látni kezdenénk a jeleneteket, tegyük fogyaszthatóvá az adatokat. Ehhez több dologra is szükség lesz: először is tudatnunk kell a böngészővel, hogy a küldött adatfolyam Flash-mozit közvetít. Második lépésként pedig létre kell hoznunk egy, a művünket beágyazó HTML-oldalt.
Tapasztalat, hogy a Flash-mozik fejlesztése során böngészőnk gyorsítótárát érdemes kikapcsolni, ellenkező esetben hajlamos makacsul ragaszkodni egy korábbi állapothoz. Így pedig meglehetősen nehézkes a kódolás folyamán ellenőrizni, hogy az történik-e, amit valóban szeretnénk.

Lejátszó hiányában…

Előfordulhat, hogy jelenlegi böngészőnk nem alkalmas Flash-fájlok lejátszására, ilyenkor hamar átirányít a Macromedia letöltési oldalára. Amennyiben ez önműködően nem történne meg, magunknak kell ellátogatnunk oda (*3. kép*).

Forgatás előtt – a szereplők

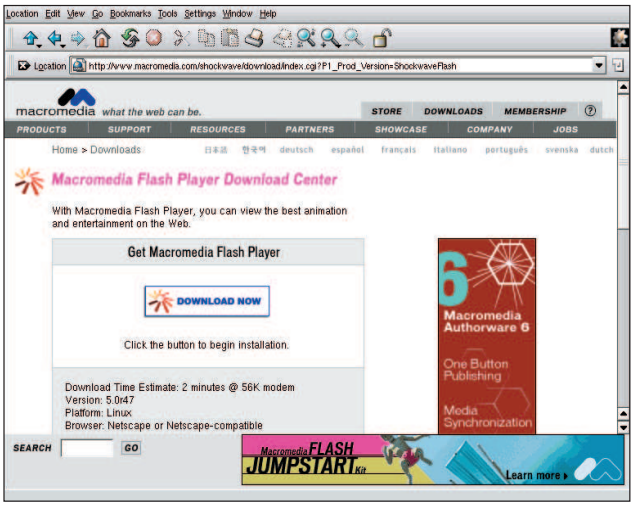
Most már feltehetően az összes szükséges eszközzel rendelkezünk a PHP-s Flash-fejlesztéshez. Beállítottuk a kiszolgálót és a böngészőnket is. Kíséreljünk meg összeállítani valami komolyabbat. Eddig csak az `SWFMovie()` osztály biztosította eszközkészlettel ügyeskedtünk. Itt az ideje megismerni egy másik, szintén elég alapvető osztályt, az `SWFShape()`-et.
A szemléltetést egy egyszerű háromszög alakjának megrajzolásával kezdem. Hozzuk létre a formát képviselő objektumot:

```
$valaki = new SWFShape();
```

Innentől $valaki néven létezik az objektum. Miután megrajzoltuk, az animációnkba számtalan példányban beilleszthetjük (így lehet fenyőfákból erdőt növeszteni). Természetesen minden ilyen egyed külön torzítható, forgatható, mozgatható. A forma megrajzolása előtt szükség lesz pár alapadat megadására, ilyen például a vonalvastagság és -szín. Tudatnunk kell azt is, ki akarjuk-e a rajzunkat festeni, és amennyiben igen, vajon mivel.

```
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
      ➥0xe4, 0xec, 50));
```

Az első sorral a rajzot megjelenítő vonal stílusát határozhatjuk meg. Először a vonal vastagságát, majd pedig a színét kell megszabnunk. A vastagság természetesen twipben értendő, ezért az 5-ös vastagság elég vékonynak számít. A vonal színének megadása a korábban megismerthez hasonló módon zajlik.



3. kép ➲ http://www.macromedia.com/shockwave/download

A vonalrajzolás beállítása után következik a kifestés meghatározása. Mint látható, egymásba ágyazott eljárásokról van szó. A belsővel, azaz az `SWFShape()` osztály `addFill()` függvényével különféle kifestési stílusokat hozhatunk létre. A jelenlegi egy puritán egyszínű festőstílus. Három adatot kell kötelezően megadnunk, amelyeket akár egy negyedikkel is kiegészíthetünk. A példára tekintve az első három talán kézenfekvő is, ezek a szokásos színösszetevők. A negyedik megadható adat pedig egy 0-tól 100-ig terjedő, áttetszőséget meghatározó érték. Ennek a `$valaki->addFill()` kifejezésnek a visszatérő értékét (egy azonosítót) kapja meg a `$valaki->setRightFill()`.
A `setRightFill()` parancsnak létezik egy társa, a

4. lista A mar_valami.php – mozgással ellátott mozi

```
<?php

// vØletlenszÆmgyÆrt  f ggvØny behœzÆsa

include 'random.inc';

// beÆll tÆsok egy helyen

define('NUM_TRIANGLES',20);
define('NUM_FRAMES',100);
define('FRAME_RATE',15);

// kezdı lØpØsek: mozi lØtrehozÆsa

$mozi = new SWFMovie();
$mozi->SetRate(FRAME_RATE);
$mozi->SetDimension(6000,6000);
$mozi->SetBackground(0xff, 0xff, 0xff);

// a mozi sorÆn rengetegszer felhasznÆlt
// hÆromsz g alakjÆnak megrajzolÆsa

$valaki = new SWFShape();
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
➥0xe4, 0xec, 50));
$valaki->movePenTo(0,1600);
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);

// A k vÆnt mennyisØgß hÆromsz g elhelyezØse,
// vØletlenszerß torz tÆsa, szØtsz rÆsa.
// Mindegyikhez k l nb zı,  sszevissza
// lØtrehozott forgÆsi sebessØg adÆsa.

for ($i=0; $i<NUM_TRIANGLES; $i++) {
    $hszog[$i] = $mozi->add($valaki);
    $hszog[$i]->move(2000+randomint(2000),
            ➥2000+randomint(2000));
    $hszog[$i]->scale(randomint(15)/10,
            ➥randomint(30)/10);
    $hszog[$i]->rotate(randomint(360));
    $hszogforg[$i] = randomint(15)-7.5;
}

// a k vÆnt mennyisØgß kØpkocka legyÆrtÆsa
// szØp sorban. Itt mÆr csak forgatni kell. :)

for ($j=0; $j<NUM_FRAMES; $j++) {
    for ($i=0; $i<NUM_TRIANGLES; $i++) {
      $hszog[$i]->rotate($hszogforg[$i]);
    }
    $mozi->nextFrame();
}

// kØsz van, mehet a vilÆg elØ...

header('Content-type:
            ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

`setLeftFill()`. Ha formánk pontjait sorrendben úgy adjuk meg, hogy a körbejárási sorrendjük az óramutató járásával megegyező irányú, akkor van szükség az elsőre. Fordított irányban haladva a „befelé” balra esik. Érdemes erre figyelmet fordítani, mert a lejátszó esetleg bedobhatja miatta a törülközőt. Érdekes adat, hogy ezeket jelenleg valamiért az `SWFMorph()` osztályon belül felcserélve kell használnunk. Apró következetlenség, majd „kinövi” a program. No igen, a PHP/MING-leírás minden egyes oldalon kihangsúlyozza, hogy ez a modul igencsak kísérleti állapotban található, az egyes elemek bármikor gyökeresen megváltozhatnak. Így jelenleg senki sem garantálja, hogy a mostani MING-objektumaink a következő kiadással is ugyanúgy fognak működni, tehát bánjunk velük óvatosan. Innentől kezdődik a forma megrajzolása, amihez vonalakat és íveket kell sorra megadnunk. Emellett rajzeszközünket vonal rajzolása nélkül is arrébb tudjuk pakolni, amire mindjárt az elején szükségünk is lesz, hiszen a tárgyunkat nem a 0,0 pontból kezdjük rajzolni. Irány a kiindulópont!

```
$valaki->movePenTo(0,1600);
```

Ezzel az eljárással „ceruzánkat” vonal rajzolása nélkül mozgatni tudjuk. A koordináták a szokásos módon X,Y sorrendben adandók meg. Vízszintes irányban egyértelmű a helyzet, hiszen ritka eset, ha valahol nem balról jobbra nő a koordinátaérték; függőleges irányban pedig számításba kell vennünk a Besenyő Pista bácsi-féle biorobotelmélet(et): „Egyet kell kérdezni: hogy mekkora, és hogy leerű-fee vagy feerű le?” Nos, kedves Boborján, a második. Tehát Y irányban a koordinátaérték lefelé növekszik, ami pont a körbejárási irány meghatározásánál a legfontosabb (csak megemlítem, hogy a méreteket itt is twipsben kell érteni). A rendszer legalább ebben végig következetes. Lehetőségünk nyílik közvetlen vagy viszonylagos helyzetmegadásra is. Példánkban egész idő alatt a közvetlen módszert választottam, amire az eljárások nevének végén található „To” szócska utal. Viszonylagos elmozdulást egy `$valaki->movePen()`-nel lehetett volna megadni. Innentől kezdve a háromszöget az óramutató járásával megegyező irányban rajzolom körbe:

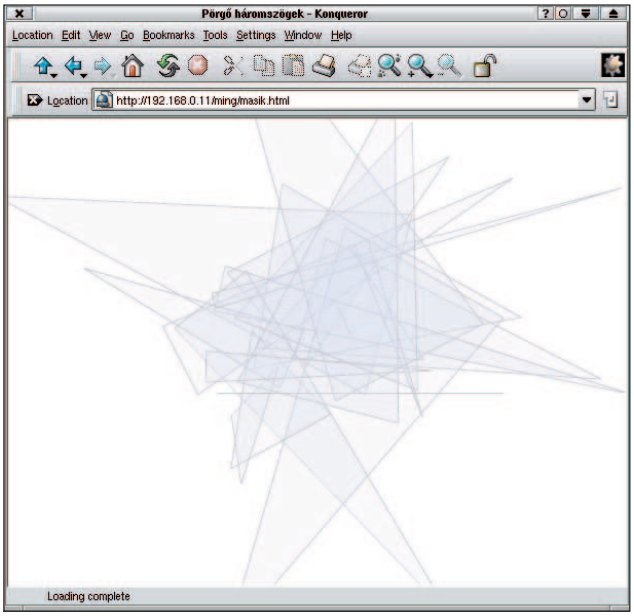
```
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);
```

Tárgyunk megrajzolásával ezennel végeztünk is. Háromszögünket helyezzük a munkatérbe, hogy láthatóvá váljon a kotyvasztásunk végeredménye. A befoglaló HTML-oldal elkészítését a nyájas olvasó/alkotó képzelőerejére bízom. A programlistában három ismeretlen sor található.

5. lista A random.inc a véletlenszám-előállításhoz

```
<?php
function randomint($max) {
  static $startseed = 0 ;
  if (!$startseed) {
    $startseed =
➥(double)microtime()*getrandmax() ;
    srand($startseed) ;
  }
  return (rand()%$max)+1 ;
}

?>>
```



4. kép Ez lesz a vége…

Az első helyezi tárgyunkat a mozi vásznára. Erre ezután a `$haromszog`-objektumon keresztül hivatkozhatunk. Érdemes tisztázni, hogy a `$valaki` csak a formát képviseli. Ebből számtalan egyedet helyezhetünk ki a mozivászonra, amelyeknek mind saját objektumuk lesz, hogy külön-külön lehessen őket módosítgatni. Egy ilyen egyed már az `SWFDisplayItem()` osztályra tartozik. A következő, `$haromszog->move(3000,3000)` paranccsal kerül a háromszög a munkaterület közepére. A `move` az első eljárás, amellyel az `SWFDisplayItem()` parancsai közül megismerkedhetünk. A példa kedvéért szokásomtól eltérve viszonylagos elmozdulást adtam meg. Ennek a `move()` utasításnak tehát létezik egy `moveTo()` testvére is. A formát természetesen eleve olyan módon is rajzolhattuk volna, hogy a közepe a 3000,3000 pontban legyen, de a saját 0,0 pontjának kitüntetett szerepe van: ekörül forog ugyanis, ha egy kicsit „megtekergetjük”.
A `$mozi->nextFrame()` kiadása akkor szükséges, amikor a képkockát befejeztük, azaz minden a helyén van. Mivel egyetlen kockánk készült el, véglegesíthetjük. Igaz, nem lesz több képkocka, „de úgy szép, ha kerek”, és így biztos nem adódik vele gond.

Csapó!

Az animációt kockáról kockára lépegetve kell megterveznünk. Amennyiben valami a mozivászonra került, azt ugyanabban az állapotában a következő kocka is tartalmazza. Csupán új elem szerepeltetésekor kerül sor újra az `SWFMovie()->add()` eljárás alkalmazására. Amennyiben valamit ki kell venni, az `SWFMovie()->remove(a tÆrgy azonos t ja)` parancsot kell kiadnunk.
Írásomat egy teljes animáció bemutatásával zárom, amely a *3. listában* látható példa továbbfejlesztésével készült. Forgatást is alkalmazok benne, amit az `SWFDisplayItem()->rotate()` parancsával tudok megtenni. A forgatási szög fokban értendő, és erre a feladatra a `rotateTo()` is használható. A pozitív forgatási irány az óramutató járásának megfelelő. A programban sok a véletlenszerű elem. Egyrészt a kirakott húsz háromszög sem mind a 3000,3000 ponton csücsül, hanem véletlenszerűen vannak szétszórva. A háromszögek objektumai a `$hszog` tömbbe kerülnek. Egy másik tömbbe helyezem a háromszögek forgási sebességét, amelynek neve `$hszogforg`. Szintén véletlenszerű adatokról van szó, értékük pozitív és negatív egyaránt lehet. Emellett a tárgyak kiinduló elfordulását is megadom, hogy induláskor se merevedjenek vigyázzállásba. Hab a tortán, hogy mindet torzítottam is, amire az `SWFDisplayItem()->scale()` eljárása ad lehetőséget. X és Y irányban a nagyítási arányt külön kell megadni.
A `scale()` használatával a jelenlegi mérethez képest történő nagyítás arányáról van szó. Amennyiben az eredetihez akarjuk viszonyítani, a `scaleTo()` is rendelkezésre áll.
A következő ciklus hivatott a százképkockás mozit legyártani. Minden kockán belül a háromszögeket egyesével meg kell forgatni a hozzá tartozó forgási sebesség szerint. Ha mind a húszat megmozgattuk, jöhet a következő kocka. Amint az összes kocka elkészült, jöhet a film bemutatása. A programot úgy írtam meg, hogy könnyedén lehessen játszadozni a lejátszási sebességgel, valamint a háromszögek számával és a mozi hosszával. A MING még számos haszonnal bír, ezeknek sajnos most nem jutott hely, de a PHP-kézikönyv MING-re vonatkozó részének értő olvasgatása már nem fog gondot okozni.
Kellemes ünnepeket, és jó MING-elést!

Heilig (Cece) Szabolcs (cece@mail.uti.hu) Veszprémben él, huszonhat éves fejjel már hatszoros nagybácsi. Több cégnek dolgozik PHP-programozóként, de PHP-távoktatást is végez. Linuxot először 1994-ben látott, kezdő perles szárnypróbálgatásai után 1997-ben szeretett bele a PHP-be. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalatoskodni.

Kapcsolódó címek

A PHP-kézikönyv MING-része ➲ http://www.php.net/ming

A MING hivatalos oldala ➲ http://www.opaque.net/ming

A Flash formátumának birtoklója, a Macromedia oldala ➲ http://www.macromedia.com

Minden, ami a Flash formátumáról tudható ➲ http://openswf.org

Modellezés DODS segítségével

Az Enhydra-kiszolgáló részeként a DODS az objektumok és a relációs adatbázisok között próbál kapcsolatot teremteni.

Mivel az adatok tárolását, lekérdezését egyszerűvé, rugalmassá és biztonságossá teszik, a legtöbb komoly webalkalmazás gerincét a relációs adatbázisok alkotják. Ez a felállás többnyire egészen addig tökéletesen működik, amíg a fejlesztők olyan objektumokkal nem kezdenek el dolgozni, amelyek teljesen más szemléletmódot követelnek. Vajon lehetőség van-e arra, hogy áthidaljuk az objektumközpontú és relációs világok közti szakadékot?

Valójában számos olyan módszer létezik, amellyel a relációs modellt objektumokká és eljárásokká formálhatjuk, és a legtöbb programozó már rég tervezett magának egy ilyen rendszert. Ahogyan a múlt hónapban láthattuk, a Perl-programozók egy kis segítséget kaphatnak az Alzabo modultól – lehetőséget ad nekik, hogy megtervezzék a táblákat, elérésükhöz pedig eljárásalapú csatolófelületet nyújt.

Ebben a hónapban a DODS-ra (Data Object Design Studio, azaz adatobjektum tervezőstúdió) vetünk egy pillantást, amely szellemében az Alzabóra hasonlít, ezt azonban Java-felhasználóknak szánták. A DODS az Enhydra központi eleme, amelynek hamarosan megjelenő változata (Enhydra Enterprise) várhatóan az első olyan nyílt forrású alkalmazáskiszolgáló lesz, amely támogatja a J2EE-t (Java 2, Enterprise Edition).

Jelenleg az Enhydra Enterprise még kipróbálás alatt áll, és bár úgy tűnik, a DODS-támogatás sokat fejlődött az utolsó változat óta, ***David Young*** a Lutris Enhydra-prófétája szerint az Enhydra 3.x DODS-változata azért üzembiztosabb. Hogy a dolgokat könnyen kipróbálhassam, a Lutris küldött nekem egy EAS (Enhydra Applicattion Service) példányt, amely az Enhydra bővített, kereskedelmi változata.

Nem vagyok benne teljesen biztos, mi a különbség az EAS és a nyílt forrású Enhydra-kiszolgáló között. Az ***Enhydra.org*** azt írja, hogy az EAS az Enhydrán alapul, de az EAS és egy Enhydra-példány vásárlása közötti különbség nem teljesen nyilvánvaló. Azt fogom feltételezni, hogy az általam telepített EAS nagyjából azonos a 3.1-változattal, bár meglehet, hogy ez nem teljesen pontos feltételezés.

A DODS áttekintése

A DODS-nak, akárcsak a múlt hónapban megismert Alzabo-rendszernek, kettős célja van: magas szintű felületet nyújt adatbázisok tervezéséhez, illetve eljárás- és objektumkészletet nyújt, amellyel azután az adatbázis elérhető. Míg az Alzabo kiszolgálóoldali, a DODS ügyféloldali Javában íródott alkalmazás, amelylyel adatbázisainkat építhetjük fel, illetve szerkeszthetjük.

A DODS elsődleges célja, hogy párhuzamosan készítsen SQL-meghatározásokat és Java-osztályokat, amelyek ugyanazt az adatbázist írják le. Ezután adatbázisba tölthetjük az SQL-meghatározásokat, a Java-osztályokkal pedig elérhetjük őket. Ezenkívül a DODS többfajta adatbázissal való munkára is fel lett készítve. Jelenleg PostgreSQL-, MySQL-, Sybase- és Oracle-rendszerekkel működik, de ez a kör a jövőben valószínűleg tovább bővül. Mivel a tényleges SQL-lekérdezések egy objektum-középrétegben íródtak, az Enhydra-programok átírás nélkül vihetők át egyik adatbázis-kezelőről a másikra. A valóságban természetesen kicsit bonyolultabb a dolog. Például az Enhydra PostgreSQL-támogatása nem éppen lenyűgöző, ugyanis figyelmen kívül hagyja a `SERIAL` adattípust (ez valójában egy számláló, más néven szekvencia), és nem kezeli a hivatkozásiépség-megkötéseket (referential integrity constrations), így például az idegen kulcsokat sem. Mindenesetre a cél becsülendő, és örömmel látnám, ha az Enhydra 4.x már kezelné ezt a gondot. Idővel a DODS várhatóan egyre több különféle adatbázist lesz képes kezelni, illetve a megfelelő lekérdezést az egyes SQL-nyelvjárásokhoz elkészíteni.

Az XMLC-dokumentum elkészítése

Hogy a működésükkel megismerkedhessünk, készítsünk az Enhydrával és a DODS-sal egyszerű adatbázis-, illetve webalkalmazás-együttest. Példáimban a PostgreSQL-t fogom használni, két okból kifolyólag is. Egyrészt, mert kitűnő nyílt forrású adatbázis-kezelő, másrészt mert DODS támogatja. Példánk, akárcsak a múlt hónapban, egy egyszerű webnapló lesz (más néven blog, egy olyan napló, amely az adatbázis bejegyzéseit fordított időrendben mutatja be). Egy ilyen program megírása nem különösképpen bonyolult, viszont annál inkább vonzó a DODS és az Enhydra kipróbálásaként.

Az első megállónk az Enhydra Appwizard lesz, amely elkészíti az alkalmazásunkhoz szükséges vázlatokat és könyvtárakat. Az Appwizard a ***$ENHYDRA/bin*** könyvtárban található, ahol az `ENHYDRA` az Enhydra telepítéskönyvtárának megfelelő környezeti változó. (Amikor RPM-csomagokból a saját RedHat-gépemhez telepítettem CD-ről, az `ENHYDRA` értéke ***/usr/local/lutrys-enhydra3.5.2*** volt.)

Az első `appwizard` képernyőn a hagyományos webalkalmazás és az Enhydra szuperservlet között választhattam, az utóbbi mellett döntöttem. A következő képen a HTML projektre voksoltam (a vezeték nélküli WML projekt helyett), majd a projektet elneveztem „blog”-nak és behelyeztem az ***il.co.lerner*** csomagosztályba. Elfogadtam az Enhydra-alkalmazásokhoz rendelt, alapértelmezett ***~/enhydraApps/*** alkalmazás saját könyvtárát. A forráskódomhoz nem szándékoztam szerzői jogi üzenetet rendelni, így végezetül a ***Finish***-re kattintottam, amely a ***~/enhydraApps/*** könyvtárban 18 új állományt hozott létre.

Most, hogy elkészítettük az alkalmazás vázát, módosíthatjuk az Enhydrával érkező alapértelmezett üdvözlő (Welcome) oldalt. Ezt két lépésben kell megtennünk: először a ***Welcome.html*** HTML-fájlt kell megváltoztatnunk, amely az én gépemen a ***~/enhydraApps/blog/src/il/co/lerner/presentation/Welcome.html*** helyen található.

Érdemes odafigyelni rá, mivel ez a fájl nem csak alap-HTML, hanem az XMLC által feldolgozandó további tagokat is tartalmaz (lásd ***Linuxvilág*** 2001. október, 67. oldal). Amint az *1. listában* látható, úgy fogjuk megváltoztatni, hogy az eredeti egyszerű oldal helyett blogunk legfrissebb adatait jelenítse

meg. Az XMLC és a hagyományos HTML-oldal közötti egyetlen különbség az, hogy a módosítani kívánt részeket `id` tulajdonsággal felruházott `<span>` tagok közé helyezzük. Például

```
<p><b><span id="date">Date</span></b></p>
<p><span id="text">Text</span></p>
```

amennyiben ezt a fájlt most egyszerűen csak megjelenítjük a böngészőben, a ***Date*** és ***Text*** szavakat fogjuk látni. A felhasználók azonban nem fogják ezt az oldalt közvetlenül elérni. Az XMLC ugyanis Java-osztállyá fogja őket fordítani. Ezután a ***WelcomePresentation*** osztályt használhatjuk a dokumentum egy példányának előállítására, miközben a ***text*** és a ***date*** mezők értékét önműködően létrehozott eljárásokkal állíthatjuk be.

A DODS használata

A ***WelcomePresentation*** a ***date*** és ***text*** mezőkbe szánt adatot a relációs adatbázistáblából kérdezi le. Mielőtt folytatnánk, készítenünk kell tehát egy táblát, és be kell népesítenünk néhány adattal.
Ez az a pont, ahol a DODS belép a képbe. A ***$ENHYDRA/bin/dods*** helyen található `dods` program szintén ügyféloldali, grafikus, Javában írt alkalmazás. Amennyiben a DODS-sal dolgozunk, soha ne feledjük, hogy olyan alkalmazást használunk, amely két különböző szemléletmódot köt össze, így a kifejezésmódja néha bizony furcsának tűnhet.
A DODS egy csomag készítésével indul, amely majd az általunk készített összes tábla és tulajdonság tárolására szolgál. Amint azt a kezdeti DODS-képernyőn is láthatjuk, e csomag neve alapértelmezés szerint ***root***. Ezt én ***blog***-ra változtattam, rákattintottam a ***root*** mappára, majd az ***Edit*** menüből a ***Package*** pontot választottam.
Az adattáblát (***BlogEntries***) két tulajdonsággal hozzuk létre: ***date*** (***dátum***), és ***text***, amelyek egyébként megegyeznek a ***Welcome.html*** változatunkban használt ***id*** tagokkal. Első lépésként az ***Insert*** menüponttal egy új táblát adunk hozzá a BlogEntrieshez: kiválasztjuk a ***data object*** pontot és a ***BlogEntries*** nevet adjuk neki.
Ezután a táblánkhoz két mezőt (***date*** és ***text***) kell adnunk. Ehhez a DODS-ablak bal felső sarkában kattintsunk a ***BlogEntries*** szóra és az ***Insert*** menüpont használatával szúrjuk be a két új tulajdonságot. Mindkét tulajdonságunk ***varchar*** típusú lesz – jelölvén, hogy szöveget szeretnénk bennük tárolni. Bár a mi esetünkben ez tökéletesen megfelel, kár, hogy DODS a PostgreSQL `TIMESTAMP` adattípusát nem kezeli, ez ugyanis ügyes és kifinomult idő- és dátumadat-kezelést tesz lehetővé. Így a dátumokat szöveges formában fogjuk tárolni, tudván, hogy az `ORDER BY` segítségével sorrendben kaphatjuk őket vissza – és nekünk ennyi elég is.
Mivel web-, illetve adatbázis-alkalmazásunkkal a lehető legnagyobb sebességet szeretnénk elérni, és mivel lényegesen kevesebbszer fogunk szöveget beszúrni, mint lekérdezni, közöljük a DODS-sal, hogy mindkét mezőnket tegye indexelhetővé és lekérdezhetővé. Az első az SQL-meghatározást fogja módosítani azáltal, hogy indexet készít a mezőkhöz. A második egy további eljárást készít, amivel az oszlopban tárolt adatokhoz férhetünk hozzá.
Végül az adatbázismenüből kiválasztjuk a PostgreSQL-t. Ezáltal a DODS kifejezetten PostgreSQL-stílusú SQL-t készít.
Most, hogy táblánkat elkészítettük a DODS-sal, nincs más hátra, minthogy (a ***File, Save as*** menüpont segítségével XML-formátumú DOML formában) mentsük, amely leírja a táblánkat, és mind a Java, mind az SQL elkészítéséhez felhasználható. Mentsük a DOML-fájlt a projektcsomag forráskönyvtárába; az én esetemben ez a ***blog/src/il/co/lerner*** könyvtárat jelenti. Ha DOML-fájlunk elkészült (lásd a *2. listát*), SQL- és Java-parancsokká alakíthatjuk át a ***File*** menü ***Build all*** parancsával. E választás eredményeképpen számos fájl keletkezik az adatkönyvtárban, ezért amikor válaszolnunk kell, hogy a fájlokat hova szeretnénk telepíteni, válasszuk azt az adatkönyvtárat, ahová a ***blog.doml*** fájlt helyeztük. Egy ablak fog megjelenni, amely tájékoztat bennünket, hogy a DODS mit is csinál éppen. Ha minden simán ment, a DODS-ból akár ki is léphetünk.

Az adatbázis elkészítése

A DODS `build all` parancsának futtatása a DOML-fájlt jó néhány új fájlra bontotta szét az adatkönyvtárban. A könyvtár immár nemcsak egy (korábban üres) Makefile-t tartalmaz, hanem egy `blog` alkönyvtárat is, amely a következő négy Javaosztályt tartalmazza: `BlogEntriesBO`, amely az Enhydra megjelenítést végző bemutató objektumához hasonlít; `BlogEntriesDataStruct`, amely tulajdonképpen táblánk adatait tárolja; `BlogEntriesDOI` amely a `BlogEntriesDO` objektumhoz tartozó csatolófelület; végül a `BlogEntriesQuery`, amely lehetővé teszi, hogy az előzőleg lekérdezhetőnek megjelölt mezőket lekérdezzük.
Az elkészült Java-forráskódon túl néhány olyan fájlt is találunk itt, amely SQL-utasításokat tartalmaz. Nevezetesen rálelhetünk egy ***create_tables.sql*** fájlra, amelynek segítségével az adatbázisunkat hozhatjuk létre.
Mindehhez a `CREATEDB` parancsot fogjuk használni, amelyet általában valamilyen erre jogosult felhasználó hajt végre Unix-héjprogramból (ami nem feltétlenül a rendszergazdát jelenti; nézzük végig a PostgreSQL leírását, hogy megtudjuk, miképpen készítsünk PostgreSQL-felügyelőket).
Kiadhatjuk a következő parancsot:

```
CREATEDB blog
```

Ezekután az adatbázisnak interaktív módon küldhetünk lekérdezéseket a következő paranccsal:

```
psql blog
```

Ha tábláinkat az önműködően létrehozott DODS parancsfájl segítségével szeretnénk létrehozni, a `psql \i` parancsát kell használnunk:

```
\i
➥/home/reuven/enhydraApps/blog/src/il/co/
➥lerner/data/create_tables.sql
```

Láthatunk néhány `CREATE` üzenetet, majd végül ismét a psql parancssora jelentkezik be. A `\d` parancs használatával kideríthetjük, hogy a DODS nem készítette el a ***BlogEntries*** táblát. Ehelyett két másik táblát hozott létre, az egyiket ***objectid***, a másik (elsődleges) táblát pedig ***newdbtable*** néven. Az ***objectid*** tábla a `PostgreSQL` sequence függvényeit hivatott helyettesíteni, jól szemléltetve az ilyesfajta általános eszközök korlátjait. A tábla egy ***next*** oszloppal rendelkezik, amely megmutatja, milyen azonosító lesz a következő. Ennek megfelelően az adatokat a ***newdbtable*** táblába szúrjuk be, ilyenkor az ***objectid*** táblát mindig egy-egy sorral bővítjük.
Adjunk is mindjárt néhány elemet a táblákhoz, hogy legyen mit lekérdezni:

1. lista A Welcome.html XMLC-bemeneti fájl, amit a Weblog megjelenítéséhez fogunk használni

```
<html>
  <head>
      <title>Weblog</title>
  </head>

  <body bgcolor="#FFFFFF">

      <H1>Weblog</H1>

      <P>Welcome to our Weblog!  Here is the
            ➥latest entry:</P>

      <p><b>
    <span id="date">Date</span>
    </b></p><p><span id="text">Text</span></p>

  </body>
</html>
```

2. lista A blog.doml, a DODS által önműködően készített fájl

```
<?xml version="1.0" encoding="UTF-8"?>
<doml>
  <database database="PostgreSQL"
   ➥legal_values="Standard,InstantDB,Oracle,
   ➥Informix,Msql,Sybase, PostgreSQL">
    <package id="blog">
      <table id="blog.BlogEntries"
       ➥dbTableName="NewDBTable">
        <column id="entrydate"
              ➥isIndex="true"
         ➥usedForQuery="true">
          <type dbType="VARCHAR"
              ➥javaType="String"/>
        </column>
        <column id="text" isIndex="true"
         ➥usedForQuery="true">
          <type dbType="VARCHAR"
              ➥javaType="String"/>
        </column>
      </table>
    </package>
  </database>
</doml>
```

```
INSERT INTO newdbtable (entrydate, text,
➥objectid, objectversion)
➥VALUES (NOW(),'First blog entry', 1, 1);
INSERT INTO objectid ( next ) VALUES ('2');

INSERT INTO newdbtable (entrydate, text,
➥objectid, objectversion)
➥VALUES (NOW(),'Second blog entry', 2, 1);
INSERT INTO objectid ( next ) VALUES ('3');
```

Most, hogy már van két blogbejegyzésünk, ki is léphetünk a psql-ből (\q), és elkezdhetjük módosítani a Java-osztályokat. Rakjuk össze mindezt!
A varázslás nagy része a ***WelcomePresentation.java*** fájlban zajlik. Itt fog elkészülni a ***Welcome.html*** és az adatbázisobjektumok egy-egy példánya, majd a lekérdezések eredményének megszerzése után a HTML-fájl itt töltődik fel az adatokkal. Miután a 3. listának (24. CD Magazin/DODS könyvtár) megfelelően módosítottuk a ***WelcomePresentation.java*** fájlt, futtassuk le a `make`-et a projekt sajátkönyvtárából. Az Enhydra lefordítja a Java-osztályainkat, ellenőrzi, hogy minden szükséges a helyén van-e, és futásra kész állapotba hozza az alkalmazásunkat.
Figyeljük meg, hogy a 3. listában módosítanunk kellett a `run` eljárást, hogy két új kivételt adjon vissza: a `NonUniqueQueryException`-t és `DataObjectException`-t. Ezeket az általunk létrehozott különféle adatobjektumok hozzák létre, és mivel ezeket a kivételeket nem szándékozunk elfogni, a hívónak jeleznünk kell, hogy lehet, hogy kivételt vált ki.
A 3. lista a DODS-eljárásokat használó SQL-lekérdezéseket az Enhydra QueryBuilder segítségével hozta létre. Első lépésként elkészítjük az egyik önműködően létrejövő osztály, a `BlogEntriesQuery` egy példányát:

```
BlogEntriesQuery blogq = new
      BlogEntriesQuery();
```

A jelen pillanatig az összes sort le szeretnénk kérdezni, fordított időrendben:

```
blogq.setQueryEntrydate("NOW()",
      ➥QueryBuilder("LESS_THAN");
blogq.addOrderByEntrydate(false);
```

Olyan eljárások is léteznek, amelyekkel a `WHERE` kifejezést is a lekérdezésünkhöz szúrhatjuk, így már meglehetősen összetett lekérdezéseket alkothatunk.
Végül az egyező sorok halmazát kapjuk vissza, amelyek mindegyike egy-egy `BlogEntriesDO` objektum formájában jelenik meg:

```
BlogEntriesDO[] blogEntries =
      ➥blogq.getDOArray();
```

Mivel csak a legfrissebb adatot szeretnénk megjeleníteni, a tömbnek egyszerűen az első elemét vesszük. A szöveget az XMLC által létrehozott „welcome” objektum eljárásának segítségével illesztjük a dokumentumunkba:

```
Welcome.setTextDate(blogEntries[0].getEntrydate());
Welcome.setTextText(blogEntries[0].getText());
```

Ha a módosítással készen vagyunk, futtassuk le a `make`-et a projekt legfelsőbb szintű könyvtárából. Ha Java programunkban bármilyen hibát találunk, ahányszor csak akarjuk, kijavíthatjuk, majd újrafuttathatjuk a `make`-et.
Elméletileg – a kimeneti könyvtárba lépve és a `./start` parancsot futtatva – most már elindíthatjuk a programot. Ha valóban megtesszük, láthatjuk, hogy próbálkozásunk kudarcot vall, mivel az alkalmazás még nem tudja, hol keresse a ***PostgreSQL .JAR*** fájlt. Az is hasznos, ha az első használatkor az Enhydrát (illetve bármilyen más alkalmazást) teljes körű hibakeresési kimenettel indítjuk, hiszen így az esetleges hibákat sokkal gyorsabban fel tudjuk deríteni és ki tudjuk javítani.

Három fájlt kell módosítanunk, hogy működésre bírjuk a dolgokat. Először módosítsuk az *$ENHYDRA/bin/multiserver* fájlt: a PostgreSQL JDBC meghajtó .JAR fájljára helyezzünk el egy hivatkozást. Ezt egyszerűen a multiserver fájl megváltoztatásával érhetjük el (ez tulajdonképpen egy héjprogram, ami a Java programot hívja meg). A `Build up classpath` megjegyzéssor alatti részeket a következőre változtatjuk:

```
# Where is the PostgreSQL JDBC .jar file?
PG_JDBC=/usr/share/pgsql/jdbc7.1-1.2.jar

if   [ X${CLASSPATH} = "X"] ; then
      ➥NEWCP=${ENHYDRA_CLASSES}${PS}${PG_JDBC}
else

NEWCP=${ENHYDRA_CLASSES}${PS}${CLASSPATH}${PS}
      ${PG_JDBC}
fi
```

Következő lépésként be kell állítanunk a *blog.conf* fájlt. Minden Enhydra-projekt tartalmaz egy beállításfájlt, amely számos más érték mellett meghatározza, hogy a rendszer milyen adatbázist használjon. Az én esetemben ez a beállításfájl a *blog/output/conf/blog.conf* névre hallgat, és az alkalmazással kapcsolatos rengeteg név-érték párost tartalmaz.
Módosítanunk kell a `Database manager` rész néhány részletét, hogy a mi programunkra mutasson. A 4. listában (24. CD Magazin/DODS könyvtár) megtaláljuk ezt a részt – abban a formában, ahogy annak lennie kell.
Végül módosítjuk a *servlet.conf*-ot. Annak ellenére, hogy nem kell feltétlenül megváltoztatnunk, hasznos dolognak tartom a következő két sorban a hibakereső részek bekapcsolását:

```
Server.LogToFile[] = EMERGENCY, ALERT,
      ➥CRITICAL, ERROR,WARNING, INFO, DEBUG
Server.LogToStderr[] = EMERGENCY, ALERT,
      ➥CRITICAL, ERROR,WARNING, INFO, DEBUG
```

A legfontosabb, amit a *blog.conf* és *servelet.conf* fájlokról tudni kell, hogy újra létrejönnek, valahányszor csak legfelsőbb szintű `make` parancsot adunk ki. Ezért ha ilyen módszerrel változtattuk meg a fájlokat, többet semmi esetre se adjuk ki a legfelsőbb szintű `make` parancsot. Ha megtesszük, meg fogjuk bánni (ahogy én is megbántam). Ehelyett inkább ezt a parancsot *presentation* könyvtárban adjuk ki.
Ha a beállításfájlokat megváltoztattuk, beléphetünk a *~/enhydraApps/blog/output* könyvtárba és futtathatjuk a `./start` parancsot. Láthatjuk, ahogy a kiszolgáló elindul, illetve (amennyiben a DEBUG lehetőséget beállítottuk a *servlet.conf*-ban, illetve a naplózást a *blog.conf*-ban) jó néhány hibakereső üzenetet tekinthetünk meg.
Ellenőrizhetjük is alkotásunkat. Állítsuk a böngészőt 9000-es kapura, amely az Enhydra-alkalmazások alapértelmezett kapuszáma: *http://localhost:9000/*. Ha minden jól megy, a böngészőben blogprogramunk kimenetét fogjuk látni.

Összegzés

A DODS jobb az Alzabónál, mivel kitűnő objektumréteget nyújt az SQL felett. Továbbá úgy tűnik, jobb és megbízhatóbb eljárásokat kínál az egyes lekérdezések összeállítására és az eredmények kezelésére. Ugyanakkor a DODS néhány szempontból ugyanazokban a betegségekben szenved, mint az Alzabo, vagy bármely egyéb relációs-objektum kapcsolatteremtő rendszer.
Az egyik ilyen gond, hogy új módszert kell megtanulnunk az (évek óta megszokott) SQL-lekérdezések elkészítésére, illetve a hosszabb lekérdezéseket meglehetősen kényelmetlen megírni, hiszen eljáráshívásokból kell összerakni őket. A DODS-szerű rendszerek általános használhatósága egyben azt is jelenti, hogy kedvenc adatbázis-kezelőnk különleges képességeit nem használhatjuk ki. Így például a PostgreSQL esetében a DODS úgy tűnik, teljesen figyelmen kívül hagyja az idegen kulcsokat és a számlálókat, amelyek pedig sokkal tömörebb adatszerkezetet eredményeznének.
Az Enhydra egyéb részeivel társítása a DODS azonban kitűnően működik. Akárcsak az XMLC és a superservletek esetében, a DODS-t is előbb letaglózónak és ügyetlennek éreztem, utóbb viszont hasznosnak és okosnak. Első pillantásra az Enhydra DODS eszköze jó próbálkozás az objektum- és a relációs világ közötti szakadék áthidalására. Már alig várom az Enhydra Enterprise végső változatát, amely kétségtelenül újabb lökést ad majd a dolgoknak.
A következő hónapban belenézünk abba az egyre inkább előtérbe kerülő szabványba, amely nemcsak, hogy összeköti a relációs és az objektumvilágot, de kiszolgálóoldali Java-alkalmazásainkat tranzakció-kezelési képességekkel is felruházza. Az Enterprise JavaBeans szolgáltatásokat és adatokat nyújt a webalkalmazásoknak, és egyre népszerűbbé válik azok közt a webfejlesztők között, akik objektumokat szeretnének alkotni, használni, illetve adatbázisban tárolni, anélkül, hogy meg kellene erőltetniük magukat.

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/).

Kapcsolódó címek

Az Enhydra fő honlapja a ➲ http://www.enhydra.org címen található. Az Enhydra 4.x más néven Enhydra Enterprise az ➲ http://www.enterprise.enhydra.org címen lelhető fel, a ➲ http://www.dods.enhydra.org címen pedig a DODS Projectről találunk néhány adatot.
Az Enhydráról szóló kitűnő bemutatót találunk a ➲ http://www.arsdigita.com/asj/enhydra címen, amelyet *Roger Metcalf*, az ArsDigita Corporation munkatársa írt.
Meglehetősen jó, bár kissé túlhaladott ismertetőt találunk az Enhydra, a DODS és a PostgreSQL használatáról az ➲ http://enhydra.enhydra.org/software/documentation/NewApps-DODS-Tutorial-PGSQL.html címen.
DODS és QueryBuilder témakörben két másik hasznos honlap: ➲ http://dods.enhydra.org/software/documentation/gettingStarted.html és ➲ http://www.dods.enhydra.org/project/faq/UsingTheGeneratedCode.html
Nagyon köszönöm a Lutris munkatársainak, különösképpen *David Young*-nak, hogy DODS-kérdéseimmel kapcsolatban ilyen segítőkészek voltak. Külön óriási köszönet illeti őket, amiért olyan nyugodtan tűrték a Federal Express és az izraeli iroda több mint egyhetes őrületét, akik egyszerűen nem hitték el, hogy egy program valóban ingyenes is lehet.

A mélység felfedezése

Marcel az fsv, a 3dfile és az XCruise segítségével a Linux furcsa világát egy egészen új nézőpontból mutatja be.

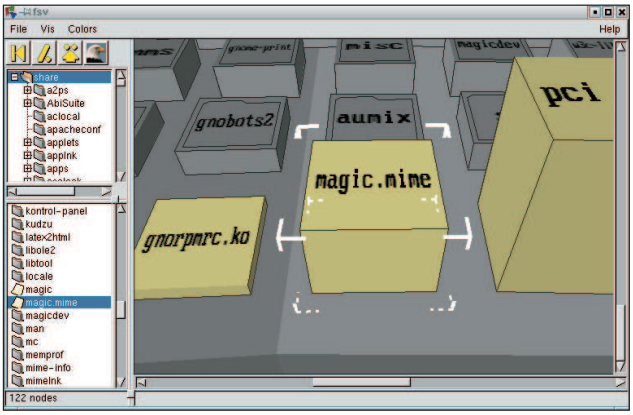
François, mon ami, le vagyok nyűgözve. El kell ismernem, amikor arra kértelek, hogy válassz olyan bort, amely kellemesen jellegzetes, teljesen megbíztam benned – és valóban elégedett vagyok. Az 1997-es Volnay-Santenots du Milineu remek választás. Kérlek, hozz fel annyit, hogy a vendégeinknek jusson bőven.
Már itt is vannak! Bienvenue, mes amis. Üdvözöllek titeket Chez Marcel kitűnő Linux-konyhájáról híres fogadójában. Foglaljatok helyet! François éppen a pincébe ment, hogy bort hozzon. Tudjátok, mes amis, mindig bámulatba ejt, amikor a finom ételek, a bor és a Linuxszal való főzés közötti párhuzamra gondolok. Nézzük például az e havi számot … François, épp jókor érkeztél. Kérlek, nyisd ki a bort és tölts a vendégeknek. Merci.
Miként a jó étteremtulajdonos eggyé válik a borospincéjével, úgy lesz eggyé a jó rendszergazda is a rendszerével. A rendszergazda minden könyvtárat és állományt úgy ismer, mint a tenyerét. Természetesen egy borospincéről sokkal könnyebb képet alkotni. Időről időre bejárom hűvös termeit, s hagyom, hogy a palackok, a címkék és az illatok elborítsák az érzékeimet. Linux-rendszergazdaként eddig csak a képzelőerőmre volt bízva e világ megjelenítése. De mától, mes amis, minden megváltozik! Ma Linux-rendszeretek valóságos hellyé változik át. Állományrendszeretek néhány nyílt forrású recept segítségével ezennel belép a harmadik dimenzióba.
A mai menü első fogása a ***Daniel Richard*** konyhájából származó, magát szerényen csak `fsv`-nek (filesystem viewer) nevező főzet. A programocska indulását követően először az egész állományrendszert végignézi, majd a tartalmát teljesen új és egyedi módon tárja elénk. Az állományok és a könyvtárak egyszerre csak különböző magasságú hasábokként jelennek meg, mintha egy idegen város utcáin járnánk. Ha rákattintasz egy könyvtárra, lehetővé válik, hogy megközelítve feltáruljon az alatta lévő „város”. A program kétféle megjelenítési módot kínál: térkép- illetve fanézetet. A fanézetben az az érdekes, hogy a könyvtárak úgy jelennek meg, mint egy távoli űrbéli város felhőkarcolói. Az *1. képen* az `fsv`-t működés közben láthatjátok. Az fsv a ➲ http://fsv.sourceforge.net címről tölthető le. Az elindításához olyan dolgokra lesz szükséged, mint az OpenGL vagy a Mesa GTK+ (lehetséges, hogy már mindkettő megtalálható a gépeden, vagy olyan közel vannak hozzád, mint Linuxod telepítőkorongjai), illetve a Löf's GtkGLArea OpenGL-készlet a GTK+-hoz, amely a ➲ http://www.student.oulu.fi/~jlot/gtkglarea címen érhető el. Amennyiben RPM-alapú rendszerrel rendelkezel, a ➲ http://ww.rpmfind.net címen előre fordított csomagokat is találhatsz. De térjünk vissza a tárgyhoz. Csomagold ki a forrást és fordítsd le:

```
tar -xzvf fsv-0.9.tar.gz
cd fsv-0.9
make
make install
```

A program futtatásához egyszerűen gépeld be az `fsv` parancsot. Az eredmény megjelenése az indulókönyvtár méretétől függően eltarthat néhány másodpercig. Amennyiben szeretnéd, parancssorban is megadhatod az induló könyvtárat. Ha például a Linux-rendszermag forrásának könyvtárából szeretnél kiindulni, a következő sort írd be:

```
fsv /usr/src/linux
```

Nem Daniel az egyetlen, aki Linuxát háromdimenziós rendszerként képzeli el. Egy másik érdekes projekt a találó ***3dfile*** nevet kapta. A 3D-látvány ismét az OpenGL, valamint a Mesa és ***Mircea Mitu*** ajándéka.
Mircea Mitu siet leszögezni, hogy a 3dfile nem kimondottan



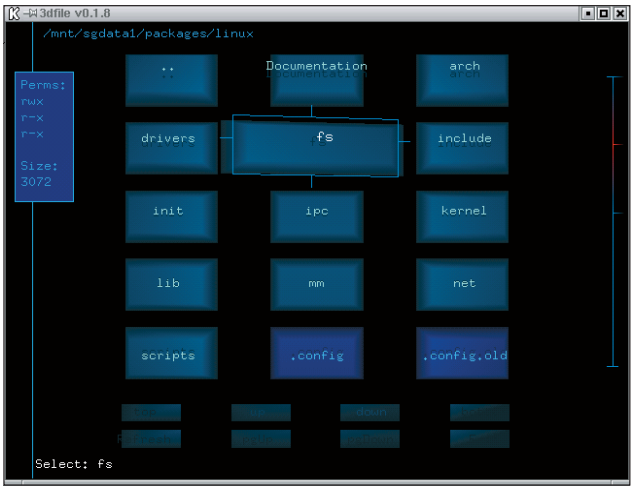
1. kép Íme, a város – állományrendszered így jelenik meg az fsv-ben

állománykezelő program, bár rendelkezik néhány ügyes megoldással (az előre-hátrahintázó ikonokon túl is). Ha például egy vörös tömbre kattintasz (amely a futtatható állományokat jeleníti meg), elindíthatod vele az alkalmazást. A Maelstrom nevű játékot is így indítottam – elvesztegetve vele egy csomó időt. Esetleg próbálj ki egy másik módszert: egy jobb egérkattintás az állomány nevén, és egy kis helyi menü jelenik meg. Amennyiben éppen szövegről van szó, szöveges állományként nézheted meg. A másik nézet hexadecimális, amely jó módszer kevésbé műszaki beállítottságú ismerőseink elkápráztatására. Nyisd ki az állományt a hexanézőkével, hosszasan tanulmányozd, végül mondj valami ilyesmit: „Á, igen, megvan a hiba”. A rendszergazdák feladata a mítosz életben tartása, non?
Vess egy pillantást a *2. képre*, majd dobjuk össze együtt ezt a kis receptet, és keltsük életre a 3dfile-t. Meg kell látogatnod a ➲ http://web.ss.pub.ro/~mrns/3dfile címet, ahonnan beszerezheted a forrást. A telepítés is zökkenőmentesen zajlik, bár a már hagyományosnak tekinthető `./configure` előtt akad egy érdekes lépés:

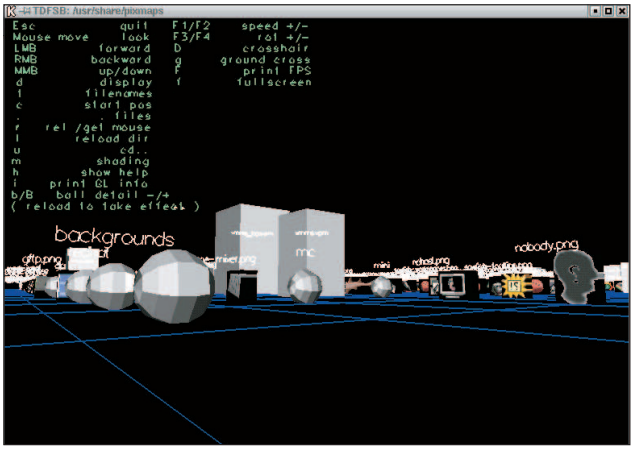
```
tar -xzvf 3dfile-0.1.8.tar.gz
```

```
cd 3dfile.0.1.8
./autogen.sh
./configure
make
make install
```

A program futtatása olyan egyszerű, mint a `3dfile` parancs beírása. A programban való eligazodáshoz elég, ha az ember az ösztöneire hagyatkozik, egyedül az alul található gombsor nem nyilvánvaló elsőre. A célkeresztet fölöttük mozgatva az irányítást még egyszerűbbnek fogod találni.



2. kép Forgó állománytömbök a 3dfile-ban



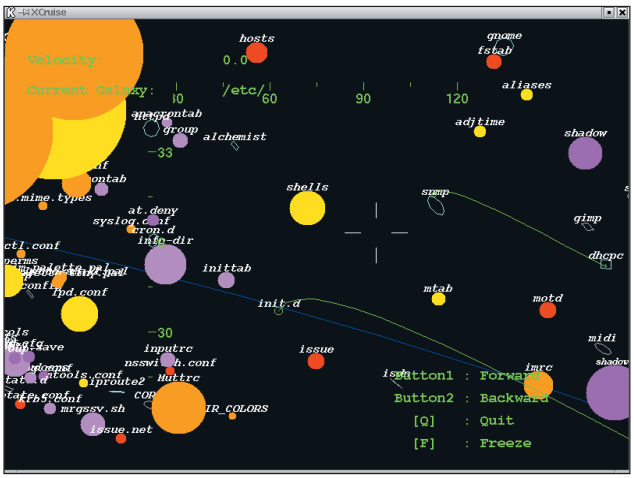
3. kép A TDFSB által elénk tárt háromdimenziós táj

A menü következő fogása a rendszer egy újabb lehetséges nézetét kínálja. ***Leander Seige*** TDFSB programjának érdekessége, hogy igen élvezetes módon teszi lehetővé utazásunkat az OpenGL előállította állományrendszerben. A könyvtárak az úszó rácstájkép felett ezüst gömbökként lebegnek. Amit különösen érdekesnek találtam, akkor válik láthatóvá, amikor az ember grafikus vagy képállományba botlik. Az állományt a program egy háromdimenziós képpé képezi le, amit körbe lehet járni. A *3. kép* bepillantást enged a TDFSB látványvilágába.
A TDFSB fordítása egyszerű, de figyelmeztetlek, hogy Leander forráskódja nem biztosít könnyen használható eszközt a programfordításra. A README csak azt tartalmazza, hogy mire lesz szükséged a fordításhoz, és azt is nagyon szűkszavúan – azért csak olvasd el. Most röviden elmondom, hogy a saját rendszeremen hogyan zajlott a beüzemelés.
Kezdd a TDBFS internetes oldalán, és töltsd le a forrást. A TDBFS kezdőlapja a ➲ http://www.hgb-leipzig.de/~leander/TDFSB címen található. Ezenkívül már csak a gdk-pixbuf csomagra volt szükségem (a Gnome része):

```
tar -xzvf tdfsb.tgz
cd tdfsb
```

Most kell elolvasnod a README állományt. Ennek alapján elkészítettem az alábbi egysoros telepítő parancsállományt. Ne felejtsd, hogy bár a szövegből ez első ránézésre nem nyilvánvaló, egyetlen sorról van szó. A parancs lényegében a README állomány részeinek másolásával állt elő. Ja, igen, a próbát egy hordozható gépen folytattam RedHat 7.1 alatt:

```
gcc -L/usr/X11R6/lib -I./ -lGL -lGLU -lglut
➥-lXmu -lXi -lXext -lX11 -lm -lgdk_pixbuf
➥-march=i686 -malign-loops=4 -malign-jumps=4
➥-malign-functions=4 -O6 -fomit-frame-pointer
➥-fno-strength-reduce -x c -o tdfsb tdb3.c
➥-I/usr/include/gdk-pixbuf
➥-I/usr/include/glib-1.2 -I/usr/include/gtk-1.2
➥-I/usr/lib/glib/include
```



4. kép A /etc galaxisban az XCruise-zal cirkálva

Ezután futtasd a programot a `./tdfsb` paranccsal. Rövid lebegő súgó megjelenítéséhez a ***h*** gomb megnyomását használhatod. Mivel a program világának navigációs eszköze az egér, használata az ***r*** gombbal váltogatható. A próbarendszerem nem rendelkezett 3D gyorsítókártyával, de nincs kétségem afelől, hogy egy ilyen kártya előnyei jól kiaknázhatók a TDFSB használatakor.
Bár a záróra vészesen közeleg, engedjétek meg, hogy egy teljesen új rendszerszemléleti lehetőséget mutassak be nektek. Linuxban még járatlan barátaink időről időre hallhatják a megjegyzést: a Linux olyan, mint egy teljesen új világ. A most felfedezett eszközök megmutattak néhányat azok közül a lenyűgöző megjelenési formák közül, amelyekben ez a világ megnyilvánulhat. Világ? Miért nem Naprendszer vagy Világegyetem? Kétségtelenül ilyen gondolatok járhattak ***Yusuke Shinyama*** fejében, amikor az XCruise-t megalkotta. Ez a nagyszerű kis csomag olyan állománykezelőt rejt, amelyben a lemezedet úgy járhatod be, mintha önálló világegyetem lenne.

Minden állományrendszer egy galaxis, az állományok csillagok (nagyobb állomány = nagyobb csillag), a közvetett hivatkozások pedig féreglyukként jelennek meg. Szerény szakácsotok és hűséges pincére így navigálva kellemes órákat töltött el a maga Linux-rendszerében.
Már alig várjátok, hogy a saját világegyetemeteket futtassátok? Pillantsatok a *4. képre*, majd látogassatok el a ➲ http://www.unixuser.org/~euske/pub/ címre.
A fordítás maga az egyszerűség. Kövessétek ezt a kis receptet és az ebéd csaknem el is készült.

```
tar -xzvf xcruise-0.24.tar.gz
cd xcruise-0.24
xmkmf
make
```

A kapott futtatható állomány a könyvtárunkban csücsül, és oda másolhatod, ahova akarod. A futtatáshoz gépeld be a `./cruise &` parancsot. Előfordulhat, hogy az XCruise kezdeti megjelenése túl nagy lesz a képernyődhöz, de az X beállításaival szerencsére kiválasztható a megjelenítődhöz illő méret.
Az én táskagépem 1024×768-as megjelenítővel bír, de én csak egy 800×600-as területet akartam az XCruise-zal kitölteni. A programot ehhez így indítottam el:

```
./xcruise -geometry 800x600+0+0
```

Magnifique! A bal egérgombot használva hihetetlen sebességgel száguldoztam előre, a középső gombbal (vagy a kettővel egyszerre, ha az egér kétgombos) pedig ki-bemozoghattam. A görgetőgombok a szöget és az irányt változtatták. Gazdag és felettébb izgalmas ez a Linux-univerzum, non?
Á, mes amis, sajnos itt a záróra. Ideje, hogy François utoljára töltse tele a poharaitokat és visszatérjünk a való világba, amelyre a mai alkalom után talán már nem is tudtok ugyanúgy tekinteni. A következő alkalomig au revoir, mes amis. A votré santé! Bon apétit!

Marcel Gagné (mggagne@salmar.com) Mississaugaban (Ontario, Kanada) él, a Salmar Consulting Inc. rendszerépítéssel és hálózati tanácsadással foglalkozó cég elnöke. Pilóta és sci-fi író egy személyben. A világhálón elérhető honlapján sok hasznos dolgot találhatunk. ➲ http://www.salmar.com/marcel/

Kapcsolódó címek

3dfile Web Site ➲ web.ss.pub.ro/~mms/3dfile

fsv (3-D Filesystem Visualizer) ➲ fsv.sourceforge.net

tdfsb (3-D Filesystem Browser) ➲ www.hgb-leipzig.de/~leander/

TDFSB The WINE Headquarters ➲ www.winehq.com

XCruise ➲ www.unixuser.org/~euske/pub

Rendszergazdák, vége az arany életnek!

Ahogyan a Linux egyre inkább megjelenik az asztali gépeken, úgy válik a rendszergazdák élete is egyre nehezebbé.

A linuxos rendszergazdák élete eddig tulajdonképpen egészen egyszerű volt, ám úgy tűnik, hamarosan megváltozik a helyzet. Miért? A Linux eddig alapvetően a kiszolgálók operációs rendszere volt, a grafikus felület és vele együtt az ablakkezelők is csupán bájos kiegészítők maradtak. Az asztali gépekre kerülése azonban a linuxos rendszergazdák életét jócskán megnehezíti, hiszen többé nem elegendő a Sendmail, a DNS- és az Apache-kiszolgáló beállításait ismerni, most már a felhasználók által „lerohasztott" grafikus felületek rendbe tételéhez is érteniük kell. Kívülről-belülről meg kell ismerni a KDE, a Gnome, az XFCE, a TWM, az FVVM, az MWM, a Blackbox stb. működését. A jelenlegi cégednél ugyan KDE-t használsz, a másiknál viszont már a Gnome-hoz, vagy a tucatnyi ablakkezelő valamelyikéhez ragaszkodnak. Azután ott vannak azok a huncut „tapasztalt felhasználók", akik elég veszélyesek, hiszen a saját ablakkezelőjüket természetesen maguk telepítik. Nemrég belenéztem a KDE-beállítófájlokba és elrémültem. A */opt/kde/share/config* könyvtárban több fájl van, mint a */etc*-ben! Tartok tőle, eltart majd egy ideig, amíg megismerem őket. Amennyiben végeztem, nekiállhatok a Gnome-nak. Tudom, ezekre a feladatokra grafikus eszközök is léteznek, de csak akkor lehet őket jól használni, ha a kérdéses gép előtt állunk. Amennyiben viszont egy 56 kb/s sebességű modemes kapcsolat túloldalán található, a `vi` marad az egyetlen választásunk.

GtkDiskFree

A lemezhasználatot általában inkább a parancssorból szeretem ellenőrizni, de ez az apróság nagyon megtetszett. A leginkább az fogott meg, hogy az összes gépemen elhelyezhetek belőle egy másolatot, majd az SSH távoli alkalmazásfuttatási képességét kihasználva az eredményt helyileg is megjeleníthetem (hogyha előtte beállítottam az X11 továbbítást). Egyszerre több példányt is kirakhatok a képernyőre, és egy időben több gépet is szemmel tarthatok. A futtatáshoz szükséges: libgtk, libfdk, libgmodule, libglic, libXext, libX11, libm, glibc.
➲ http://gtkdiskfree.sourceforge.net

Perltidy

Lehet, hogy tévedek – sokszor megesik velem ilyesmi –, de úgy vélem, a rendszergazdák túlnyomó többsége legalább egy kicsit ismeri a Perlt, hiszen annyira hasznos, és mégis viszonylag egyszerű programokat írni vele. Különféle modulokkal felszerelkezve pedig rengeteg feladatot oldhatunk meg igen kevés valós programozási tudással. A Perltidy alkalmazás úgy lehet segítségünkre, hogy a Perl programkódot valamivel olvashatóbbá teszi. Én például nem mindig igazítom a sorokat. A programozási munkák között (írás/kipróbálás/újraírás) időnként rájövök, hogy bizonyos dolgokat másképp kellene megoldani. A Perltidy segítségével rendbe szedhetjük a kódot, és még sok egyéb dologban is a hasznunkra lesz. Nem tökéletes program, de a kódot biztosan olvashatóbbá teszi. A futtatáshoz szükséges: Perl és az IO::File Perl modul.
➲ http://perltidy.sourceforge.net

vcheck

Amennyiben olyan rendszert kell karbantartanunk, amelyen mindenből a legújabb és legszebb található, a vcheck életet menthet. Nem tudok olyan terjesztésről, amely a könyvtárakat és a fájlokat – hacsak nem biztonsági okokból – frissen tartaná. Ha viszont a legújabb programokat szeretném a gépemen lefordítani, mindig a könyvtárak legújabb változatával kell rendelkeznem. Ezzel a segédprogrammal mindez gyerekjáték. Alapértelmezett beállítás szerint csak a rendszermagot fogja ellenőrizni, de pillanatok alatt készíthetsz újabb bejegyzéseket is. Némi munka a cronnal, és a legújabb programok máris készen állnak a telepítésre. A futtatáshoz szükséges: Perl és Perl-modulok: LWP::UserAgent, Getopt::Long, File::Copy, http:::Request és File::Basename.
➲ http://www.tu-ilmenau.de/~gomar/stuff/vcheck

gocr

A gocr fejlesztésének célja, hogy a Linux-változatokat optikai karakterfelismerő (OCR) képességekkel ruházza fel. A programot úgy tervezték, hogy PNG-fájlokat (faxokat) lehessen ASCII-szöveggé alakítani vele. Több különböző dokumentummal próbáltam ki, és úgy találtam, hogy működik, de rendkívül kényes a betűk méretére és típusára. Ha valaki csodás faxot küld különféle méretű és típusú betűkkel, a gocr valószínűleg nem sokra fog jutni; ha azonban egyszerű fax érkezik, Courier betűtípussal írva – nem lesz semmi gond. Nem ígérem, hogy elégedett leszel vele, de érdemes kipróbálni. A futtatáshoz szükséges: libpnm, libpgm, libppm, glibc.
➲ http://ftp://jocr.sourceforge.net

ettercap

Az ettercap jóval többet nyújt, mint az egyszerű – ámbár roppant hasznos – tcpdump, és jóval többet árul el arról, hogy mi is történik a helyi hálózaton. Képes arra, hogy például két állomás között folyó vagy a webes forgalmat mutassa stb. Elemezni tudja a titkosított forgalmat, sőt még kapcsolóval (switch) összekötött hálózatban is működik. Egyedül a telepítés utáni első indításkor kelt jelentősebb terhelést a hálózaton, kapcsolóval összekötött hálózatban ugyanis ARP-vihart keltve szerzi be a hálózatról a tájékoztató adatokat. Természetesen mindig rendelkezésünkre állnak olyan parancssori kapcsolók, amelyekkel e viselkedését felülbírálhatjuk. A futtatáshoz szükséges: libdl, libform, libncurses, libm, libssl, libcrypto, glibc.
➲ http://ettercap.sourceforge.net

Ennyit erre a hónapra.

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

Télapó, gyere már...

Mindenkit utolért már a decemberi láz. Mindenki az ünnepekkel foglalkozik, gyorsan befejezi hosszú távú terveit és felkészül az ünnepekre. Valahogy így vagyunk mi is: lapzárta után kiakasztjuk a „telelünk" táblát, a következő megjelenésünk ugyanis egy összevont január–februári szám keretében történik, amelyet január legelején lehet majd kézbe venni. Természetesen a január–február is egy számnak minősül, de megnövelt oldalszámmal és 3 CD-melléklettel bővítve – szerencsére azonos áron.

Térjünk azonban vissza a karácsonyhoz! Mi is igyekeztünk megtömni a Mikulás puttonyát mindenféle jóval: most is foglalkozunk majd a bevezető témákkal, ***Szaló István*** például a kezdők számára indított egy sorozatot, amelyben nem kisebb feladatra vállalkozott, minthogy az Emacs-környezetet mutassa be. ***Garzó András*** is kedveskedik az első szárnycsapásokkal próbálkozóknak: a lelkes vállalkozókat egy háromoldalas cikkel igyekszik felkészíteni a rendszermag újrafordítására. Szeretném elmondani, hogy a rendszermag – ahogyan neve is mutatja – a rendszer lelke, emellett rendkívül bonyolult valami, tehát készüljünk fel, hogy elsőre nem mindig járunk teljes sikerrel. Bár olyan ez, mint a tűzkeresztség... ha valaki komolyan szeretne linuxozni, előbb-utóbb át kell esnie rajta! És célszerű némi időt hagyni rá – mondjuk két-három napot.

A haladóknak is igyekszünk kedvezni. Manapság kedvenc témáink egyike a vírusvédelem: nem véletlen, hogy újra előkerült, hiszen ismét rendkívül sok vírus ütötte fel a fejét. Bár hallottam olyan véleményt is, hogy miért szenvedünk? Ha találkozunk egy vírusos géppel, adjunk a felhasználónak egy linuxos telepítőlemezt, és lelkesítsük az átállásra. Ezt azonban a legtöbb cégnél sajnos nem lehet ilyen könnyen megtenni.

A vállalat ügymenetét veszélyeztetett operációs rendszereken végzik, csupán egy-két hozzáértő gépén fut GNU/Linux, és természetesen a kiszolgálókon. Hogy ilyen helyeken miként védhetjük meg mégis a belső hálózatot, erről szól két írásunk is: ***Borkuti Péter*** a 24. oldalon a proxy felvértezéséről ír, ***Dave Jones*** a 30. oldalon pedig egy második megoldást javasol a levezés szűrésére. Ám ez csak szemezgetés volt az e havi cikkekből.

Miről szóljunk?

Ismét szeretnénk megkérni olvasóinkat, hogy mondják meg, milyen témájú és szintű cikkeket látnának még szívesen a lap hasábjain! Az újsághoz mellékeltünk egy kérdőívet, kérek mindenkit, szánjon rá néhány percet, hogy elmondja igényeit, véleményét és ötleteit. A kérdőív kitölthető a ➲ http://www.linuxvilag.hu/kerdoiv címen is. Külön kíváncsi vagyok rá, hogy vajon a Linux-változat tárgykörben ki is nyer. Mióta a Linuxvilág él, folyamatosan kapom a véleményeket, mindegyik oldalról. Három típuslevelet már könnyedén megkülönböztetek:

a) „Kedves Linuxvilág! Örömmel lapozgatom újságjukat, de szeretném megkérdezni, miért nem foglalkoznak többet az amerikai felmérések szerint is legjobb Linux-változattal, a RedHat Linuxszal? Nem véletlen, hogy olyan sokan használják, könnyű felügyelni és egyszerűen csodás!"

b) „Kedves Linuxvilág! A lap szép, jó, de nagyon hiányolom belőle, hogy nem foglalkozik eleget a SuSE Linux (igény szerint a Mandrake szó is behelyettesíthető) képességeivel, lehetőségeivel. Rendkívül könnyen használhatónak találom, szerintem mindenkinek ezt kellene ajánlaniuk!"

c) „Kedves Linuxvilág! A lap érdekes, tartalmas, de! Elképesztőnek tartom, hogy egy szabad és nyílt világban olyan rendszereket is támogattok és ajánlotok a felhasználóknak, amelyek mögött cégek állnak, és mások közösségi munkájából akarnak nyereséghez jutni! Szerintem csak a Debian GNU/Linuxot szabadna bárhol is használni, szinte mindegyik komoly kiszolgálón ez fut!"

Úgy látszik, a háború már a vérünkké vált. Jómagam is rendelkezem már tapasztalatokkal, viszont nem érzem úgy, hogy lándzsát kellene törnöm bármelyik mellett is. Mindegyiknek van előnye is, hátránya is. A tisztánlátás kedvéért: én most egy Debianról kapott X-felületen dolgozom, amit egy Mandrake GNU/Linuxot futtató gépen látok. Mostanában ezzel ütöm el az időmet: teljes értékű X-munkagépeket alakítgatok ki csacsi öreg P1-esekből. A legnagyobb bajom, hogy csak rendkívül drágán lehet manapság hozzájutni rendes PCI csatolós monitorkártyához. Sőt, mivel a merevlemezeink egyre-másra adják be a kulcsot, hamarosan utána kell néznem, hogyan is lehet lemezmentes munkaállomásokat kialakítani.

Búcsúzóul

A karácsonyi szünet elejének, bár a szeretet jegyében illenék eltelnie, amennyire saját tapasztalatomból le tudom vonni, általában az ajándékok után rohangálással, a felhalmozott „majd később" munkák elvégzésével és a rég nem látott ismerősök felkeresésével telik el. Erre a rövid időszakra még egy kis kitartást kívánok mindenkinek, és remélem, a Jézuska mindenkinek számos kellemes meglepetést hoz! Boldog karácsonyt és kellemesen másnapos boldog új esztendőt 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 Woody 3. CD

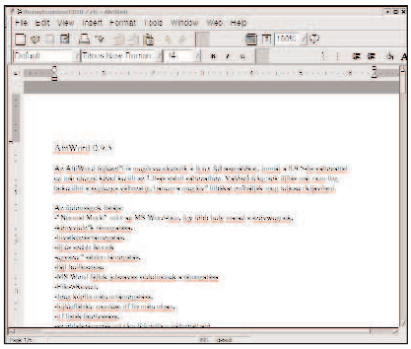
A múlt hónapban elkezdett Debian Woody változat első két korongja után most a harmadik CD kerül kiadásra. Ez lesz az utolsó a Woody terjesztés 5–6. korongjából, aminek egyszerű oka az, hogy mire az összes korongot kiadnánk, addigra már sokkalta újabbak jelennének meg, tehát nem lenne friss. Sokan kérdezik, hogyha bizonyos programok fordításához, futtatásához hiányzik egy szükséges fájl, és a rendszer nem találja, akkor honnan tudható meg, hogy melyik csomagot kell feltelepíteni (amelyik tartalmazza a hiányolt fájlt). Nos, ehhez is megvan a segítség: a 24-es CD Woody könyvtárában található ***Contents-i386.gz*** fájlban rákeresheünk a fájlra. Ha van ilyen fájl, akkor mellette rögtön láthatjuk a telepítendő csomag nevét. Példa egy keresésre: keressünk rá a startx szóra, amit a legkülönfélébb módokon tehetünk meg. A legegyszerűbb talán a `grep` program használata, lásd a listát. Mivel tömörített fájlról van szó, nem lehet egyszerűen csak keresni a fájlban. Segítségül hívtuk a `gunzip` programot, kicsomagoltuk a `-c` kapcsoló segítségével a képernyőre, és a kibontott szöveget egy csővezetékben átadjuk a `grep` programnak. Az eredmény jól látszik: maga a `startx` parancs az ***x11/xbase-clients*** csomag része, így ha valóban ezt kerestük, az `apt-get install xbase-clients` paranccsal telepíthetjük is.

Keresés a fájlban

```
csontos@sharon:~/woody$ gunzip -c Contents-i386.gz | grep startx
usr/X11R6/bin/startx                        x11/xbase-clients
usr/X11R6/man/ja/man1/startx.1x.gz          x11/xmanpages-ja
usr/X11R6/man/man1/startx.1x.gz             x11/xbase-clients
usr/bin/startxfce                           x11/xfce
usr/share/X11R6/man/it/man1/startx.1x       doc/manpages-it
usr/share/apps/ksysguard/icons/locolor/16x16/apps/startx.png
   ➥x11/kdebase
usr/share/man/man1/startxfce.1.gz           x11/xfce
usr/share/man/pl/man1/startx.1X.gz             doc/manpages-pl
csontos@sharon:~/woody$
```

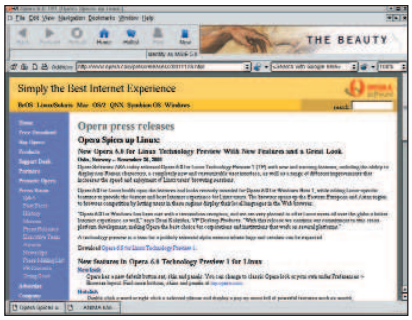
AbiWord 0.9.5

Az AbiWord fejlesztői is megörvendeztetik a Linux-felhasználókat – immár a 0.9.5-ös változattal, amely már meglehetősen közel került az 1.0-s üzembiztos változathoz. Valószínűleg sok újítás már nem fog bekerülni a végleges változatig, inkább a meglévő hibákat próbálják meg kijavítani.



Az újdonságok listája:

- Normal Mode, mint az MS Wordben, így több hely marad a szövegnek,
- a könyvjelzők támogatása,
- hivatkozástámogatás,
- új és szebb ikonok,
- egyszerű sablontámogatás,
- fájl beillesztése,
- MS Word-fájlok jelszavas védelmének támogatása,
- File –>Revert,
- jpeg képformátum támogatása,
- fejléc, illetve lábléc mentése rtf formátumban,
- rtf-listák beolvasása,
- az oldalszámozás minden fejezetben változtatható,
- Perl-héjkiegészítés,
- az állapotjelző sor kiegészítése,
- számos bővítmény (plug-in) és képkezelő-kiegészítés,
- MS-Write fájlok olvasása,
- parancssoros nyomtatás,
- a KWord-fájlok beolvasásának javítása,
- számtalan hibajavítás.



Bár nem tartozik szorosan a programhoz (igaz, ez nézőpont kérdése), de a honlapjukat is teljesen átalakították.
➲ http://www.abisource.com

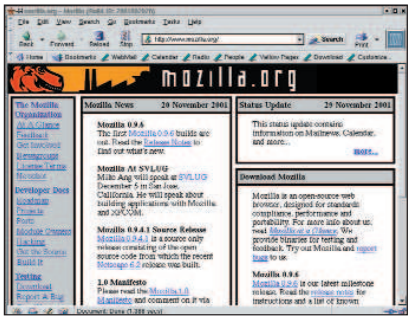
Opera 6.0

Az Opera böngésző ismét megnövelte eggyel a változatszámot, immáron a 6.0 TP (Technology Preview), azaz „szakmai előnézet" a letölthető próbaváltozat fedőneve. Hirtelen váltás ez, hiszen az előző a 5.05 TP volt, az ugrás bizonnyal a sok kisebb-nagyobb változásnak köszönhető. Felhívnám azonban mindenkinek a figyelmét, hogy ez a változat korántsem minősíthető üzembiztosnak! Működik minden általam kipróbált szolgáltatás, de ez nem jelenti azt, hogy ***minden*** és hibátlanul működik – tehát ajánlom a körültekintő használatot.
Az újdonságok közül talán a legszembetűnőbb a külső megváltozása, amely szerintem sokkal barátságosabb lett: az ikonokat lecserélték (szép kékek lettek), de természetesen a régi felületet kedvelők nyugodtan visszacserélhetik az eredetiekre, mivel a program készítői nem száműzték őket örökre (a ***File,***

Preferences, General, Look-ban tudjuk megtenni). A másik nagy újdonság a *Hotclick*: ennek segítségével, ha a böngésző ablakában duplán kattintunk egy szóra, egy menüt kapunk, ahol kiválaszthatjuk, hogy mit is szeretnénk vele csinálni: másolni, keresni (valamelyik keresőmotorral), kaphatunk szótárat, vagy lefordíthatjuk valamilyen nyelvre (sajnos a magyar nem szerepel köztük). A *Personal Bar*-ban egy helyen tudjuk a kereséseinket és a könyvjelzőket kezelni, ezeket sorba rendezhetjük, kereshetünk bennük vagy saját elrendezést is készíthetünk. Új a süti- (cookie) kezelő rész, a *Quick Preferences* F12 segítségével pedig könnyedén elérhetjük Operánk legfontosabb beállítási lehetőségeit. Ez az a változat az Opera életében, amelyik támogatja a Unicode-karaktereket, és amelyben egyszerűbb lett az ablakok kezelése – a CTRL+TAB billentyűkkel válthatunk közöttük. Kapunk egy *Contact List* nevű eszközt is, amelyben a barátaink, munkatársaink adatait tárolhatjuk weboldalukkal és levélcímükkel együtt. A bővítmények támogatása pedig sokat javult a 5.05 TP-hez képest, most már a legtöbb Netscape-bővítmény is működik, mint a Macromedia Flash, az Acrobat Reader, a Real Player, a Java, a Plugger, a TCL 2.0 és a Codeweaver Crossover (Apple Quicktime).
A teljes súgót újraírták, és a saját ablakban böngészhetjük.
➲ http://www.opera.com

Mozilla 0.9.6

A Mozilla böngésző is egyenes úton halad a megdicsőülés felé, amit szintén az 1.0-s változatszám elérése jelent majd. Érdemes összehasonlítgatni a linuxos programok változatszámait: amint láthatjuk, ebben a hónapban szinte minden programnak a 0.9.x-es változatát tudjuk a mellékletre tenni.



Ez reményeim szerint azt jelenti, hogy még ebben az évben (sajnos, így akkor is csak jövőre kerülhetnek a CD-re), de ha nem is idén, akkor a jövő év elején mindhárom programnak megjelenik a megbízható változata (azt hiszem, a megbízhatósággal e programok utóbbi változatainál sem volt sok gond). A Mozilla mostani változata képes megjeleníteni a Windows .BMP és .ICO formátumú képeit, működik a nyomtatási előkép (*Print Preview*), sokat változott a levelezőrendszer, továbbá lehetőség nyílik, hogy a weboldalon kijelölt szóra rákeressünk az Interneten.
➲ http://www.mozilla.org

Mandrake-frissítések

Mostani CD-mellékletünkön az októberi korongon található Mandrake Linux 8.1-es rendszerhez eddig megjelent összes frissítést közreadjuk, hogy olvasóink biztonságban érezhessék magukat és a rendszerüket.
Érdemes azoknak is frissíteniük, akik nem kapcsolódnak a Világhálóhoz. Valószínű, hogy időközben újabb és újabb frissítések jelennek meg, amelyeket a ➲ http://www.linux-mandrake.com/en/security/mdk-updates.php3?dis=8.1 címen nézhetünk meg, innen a megfelelő fájlokat a frissítéshez azonnal le is tölthetjük.

- Wu-FTPd – kijavították a távolról történő jogosulatlan hozzáférést;
- Postfix – a lehetséges távoli DoS-támadások kiküszöbölése;
- Apache – számos hibát kijavítottak;
- Squid – a lehetséges DoS-támadások kiküszöbölése;
- Expect – kijavították a lehetséges jogosulatlan hozzáférést;
- pspell;
- tetex – biztonságossá tették az ideiglenes fájlok hozzáférését;
- rendszermag 2.4 és 2.2 – számos biztonsági hibát kijavítottak;
- Mozilla – az Euro karaktereket most már helyesen jeleníti meg.

Ez a lista közel sem teljes, ezért érdemes szétnézni a 24-es CD Mandrake_frissitesek/RPMS könyvtárában, ahol több mint 150 MB-nyi anyag szerepel – ez, mint már említettem, a CD lezárásának időpontjáig megjelent összes frissítést és hibajavítást tartalmazza.

Rendszermag

Úgy tűnik, a 2.4-es rendszermagsorozat kettesével szedi a lépcsőt, mivel már másodjára történik meg, hogy két változat szinte ugyanakkor jelenik meg. Általában olyan hibákat kell sürgősen kijavítani, amelyek eléggé érzékeny részei a rendszernek. A jelenlegi változatszám a 2.4.16-os, a két lépésben

végrehajtott változások elolvashatóak az alábbi címeken:
➲ http://www.kernel.org/pub/linux/kernel/v2.4/ChangeLog-2.4.15
➲ http://www.kernel.org/pub/linux/kernel/v2.4/ChangeLog-2.4.16
Immáron a 2.5-ös fejlesztői sorozat első „lépései” is elérhetők – CD-mellékletünkön szintén megtalálhatják őket. Figyelem: szigorúan csak hozzáértőknek ajánlom őket! Egyelőre még nincs olyan sok változás a 2.4-es sorozathoz képest, ez azonban nagyon gyorsan meg fog változni és számos újdonság kerül bele, aminek a vége természetesen a 2.6-os, esetleg a 3.0-s sorozat lesz.

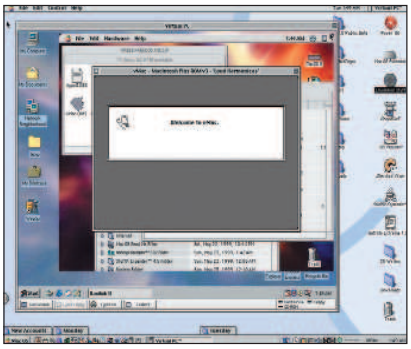
OpenOffice 641c

A Sun Microsystem már elég érettnek találta az OpenOffice-fejlesztést ahhoz, hogy elkészítse a StarOffice 6.0 próbaváltozatát, ami letölthető a Sun honlapjáról. Így ez a fejlesztés is egyre megbízhatóbban, egyre kényelmesebben – és ami nagyon fontos, a linuxos irodai csomagokban egyre „magyarabbul” használható. A fő kényelmi szempont az ékezetes betűk helyes kezelése, mindenféle külön beállítás nélkül lehet vele *ŰűŐőÁáÉéÖöÜüÍí* betűket gépelgetni, és nem kell varázslóvá vagy Linux-guruvá válnunk ahhoz sem, hogy ezeket a betűket is elérhessük (ez a legtöbb Linuxra érvényes, legalábbis a Debian Woodyra, a Mandrake 8.1-re és a RedHat 7.1-re biztosan). Ezek a karakterek nyomtatáskor is megjelennek a papíron, tehát már azt a kellemetlen tulajdonságát sem őrizte meg, hogy a képernyőn minden szépnek és jónak látszik, a papíron azonban ? vagy valamilyen más „érdekes” karakter bukkan fel a vágyott helyett.

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

MacOS Linuxon

A Basilisk II nevezetű programmal Macre készült programokat vagyunk képesek futtatni a legkülönfélébb gépeken és operációs rendszereken, közéjük tartoznak az x86-os gépek is. A MacOS összes változatát képes futtatni egészen a MacOS 8.1-ig a hozzá tartozó programok nagy részével együtt. Megoldást jelenthet mindenki számára, akik különböző gépeket és operációs rendszereket használnak, mivel így mindenhol egységes felületet kapnak kézhez.



A honlapján a következő támogatott rendszereket sorolták fel:

- BeOS R4 (PowerPC és x86, valamint már BeOS R5-höz is);
- Unix X11 grafikus felülettel (Linux, Solaris 2.5, FreeBSD 3.x és IRIX 6.5);
- AmigaOS 3.x;
- Windows NT 4.0 (Windows 9x).

A *képen* látható helyzet az emulátorok használatának egészen furcsa összetételét szemlélteti: egy MacOS futtat VirtualPC-t Windowszal, amelyben egy BasiliskII MacOS-sel fut, ebben pedig egy vMac (szintén Mac-emulátor). Bár ez kissé értelmetlennek tűnhet, mindenesetre bizonyítja, hogy a különféle rendszereket megbízhatóan „össze lehet keverni".
Használatbavételéhez mindössze az emulátor állományainak beszerzésére, valamint egy olyan fájlra lesz szükségünk, amely az eredeti ROM-ot hivatott helyettesíteni (ezt hivatalosan sajnos nem lehet letölteni, mivel nem terjeszthető szabadon, így mindenképpen egy eredeti Macre lesz szükségünk). Végül a ➲ http://www.apple.com oldaláról letölthetjük a teljes MacOS 7.5.3-as operációs rendszert. A telepítés folyamatáról a program honlapjáról kiindulva számos segítséget kaphatunk.

➲ http://www.uni-mainz.de/~bauec002/B2Main.html
➲ http://www.kearney.net/~mhoffman/basiliskii.html

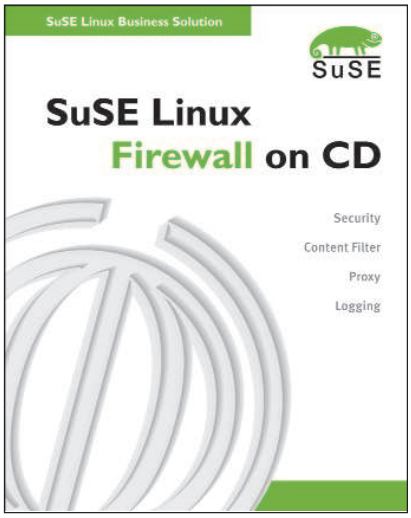
Linux két lemezen?

Egyre nagyobbak az operációs rendszerek – és egyre több helyet foglalnak el a merevlemezen. Talán pont emiatt próbálkoznak sokan az egy-két kislemezes rendszerekkel. Íme egy újabb, két kislemezes Linux-rendszer, ráadásul X-es felülettel. Készítői bebizonyították, hogy lehetséges apró, de használható rendszert is építeni.

➲ http://www.mungkie.btinternet.co.uk/projects/2diskXwin.htm

SuSE VPN

A SuSE VPN (Virtual Private Network – virtuális magánhálózat) segítségével biztonságosan köthetők össze a vállalatok különböző telephelyei az Interneten keresztül.



A SuSE Linux Firewall különlegessége az úgynevezett Live- (élő) rendszer. A módszer lényege, hogy a rendszer nem települ a számítógép merevlemezére, hanem CD-ről működik. Ez nagy előnyt jelent a biztonság tekintetében, mivel a CD-ROM-on lévő rendszerbe nem nyúlhatunk bele. A tűzfal beállítása – például a csomagszűrőé – írásvédett hajlékonylemezen tárolható. A csomag részét alkotó forráskód az egyéni igényekhez szabást is lehetővé teszi.
A csomag második CD-jén a tűzfalkarbantartási rendszer (FAS) található, amely akár több tűzfal felügyeletét is képes grafikus felületen keresztül ellátni. A SuSE Linux Firewall on CD a standard kiadás mellett mostantól VPN-kiadásban is kapható. A csomag a rendszer mellett 3 CD-t, átfogó leírást, 30 napos telepítési segítséget és egyéves programkarbantartást is magában foglal.

➲ http://www.suselinux.hu

Zorp 1.4

A BalaBit IT Biztonságtechnikai Kft. terméke, a Zorp moduláris alkalmazásszintű tűzfal, legújabb változata az 1.4-es. A fejlesztés célja, hogy nagyon érzékeny környezetben is jól alkalmazható tűzfalat hozzanak létre. Segítségével a beágyazott protokollok (például a HTTPS) forgalmának felügyelete és szűrése is lehetővé válik. A cég honlapjáról ingyenesen letölthető a program GPL-es változata is, természetesen ebben nincs benne minden lehetőség.

➲ http://www.balabit.hu
➲ http://www.balabit.hu/downloads/

Járjunk könyvtárba a Weben

Széles kínálatú számítástechnikai könyvtárat találhatunk az alábbi címen, amelyet gyors feliratkozás után azonnal használatba is vehetünk. Fontos, hogy mindenképpen létező és számunkra hozzáférhető címet adjunk meg, mert erre kapjuk meg a jelszavunkat.

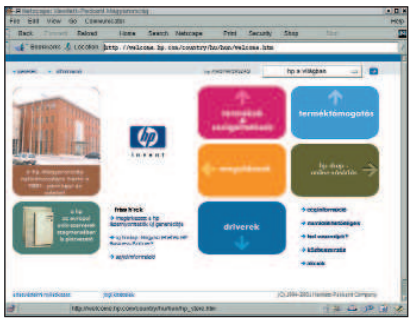
Mindenki kedvére válogathat a különféle könyvek közül, akár Java, Unix, Perl vagy bármilyen más témában keresgél is szakirodalmat. Témakörökre bontva több mint ötszáz kötet anyaga közül válogathatunk, közülük némelyek letölthetők, némelyek csupán on-line olvashatók. Letölthető például az ígéretes című „100 Linux tipp és trükk" is – nem csak vállalkozó kedvűeknek!

➲ http://www.syndik.at/harald/regal/buecher/bibliothek.asp

HP DVD+RW

A Hewlet-Packard bemutatta az első DVD+RW módszerre épülő, újraírható DVD- és CD-lemez meghajtóját. Az új HP DVD-Writer dvd100i üzleti és magánfelhasználói gyorsan és egyszerűen készíthetnek saját DVD- és CD-felvételeket. A DVD+RW módszer kifejlesztésekor kezdettől fogva szem előtt tartották a különféle szórakoztatóelektronikai és személyi számítógépes környezetek tökéletes együttműködésének fontosságát. A HP DVD-Writer dvd100i ráadásul az interaktív, dinamikus videofelvételek készítését, másolását és lejátszását is támogatja, és a nagy mennyiségű adattárolást is lehetővé teszi.
A DVD+RW lemezek 4,7 gigabájtos adattárolási képességgel rendelkeznek, ami háromórányi videofelvételi, vagyis -visszajátszási időnek, illetve 7 CD méretének felel meg. A DVD+RW adathordozók a legtöbb jelenleg kapható DVD-ROM és DVD-videolejátszóval képesek együttműködni. A CD-R adathordozók több mint egymilliárd CD-ROM- és CD-lejátszóval használhatók világszerte. Néhány jellemző:

- kombinált újraírható DVD-, illetve DVD-lemezmeghajtó;
- DVD – legalább 2,4× újraírási, illetve 8× olvasási sebesség;
- CD – legalább 12× írási, 10x újraírási, illetve 32× olvasási sebesség;
- 4,7 GB DVD;
- 600–700 MB CD;
- IDE, illetve ATAPI DVD+RW/CD-RW belső meghajtó.



➲ http://www.hp.hu

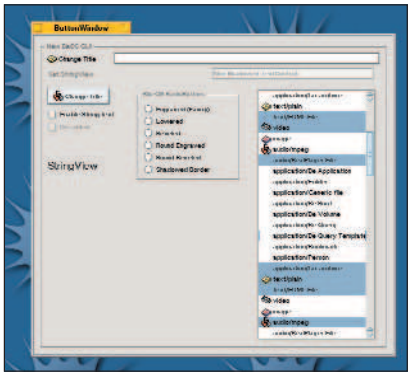
A HP termékekről az alábbi címeken tudhatunk meg részletes adatokat:

➲ http://www.hp.hu
➲ http://www.hp.comhttp://www.hp.hu/sajto/default_media.asp

A Be és a Palm

Befejeződött a Be Inc. felvásárlása, amelyre a Palm összesen 11 millió dollárt költött. Néhány fejlesztő megtartásával valószínűleg az eddigi megoldásokat szeretnék a saját rendszerükben is hasznosítani. Arról, hogy a Palmnak milyen szándékai vannak a sokak által kedvelt BeOS operációs rendszerrel, nem szólnak a hírek. A bizonytalanság azonban nem tesz (tett) jót a rendszernek, mivel senki sem fejleszt szívesen „halott rendszerre". Szerencsére a ➲ http://www.bebits.com-on látható változások nem ezt igazolják! Emellett különféle ötletek is napvilágot láttak, köztük szerepel egy másik operációs rendszer magjára átültetett BeOS-felület – ez lehetne a Linux, az ATheOS; de ilyen fejlesztés például a ScaraBeOS és a BlueOS is. A BlueOS különösen érdekes, mivel Linux-rendszermagra épül – a *képen* látható, mire is jutottak a fejlesztők. Forráskódszinten a BeOS- és a Linux-kódokat is képes lesz kezelni, így a felhasználói programok nagy részének kérdése megoldódik.

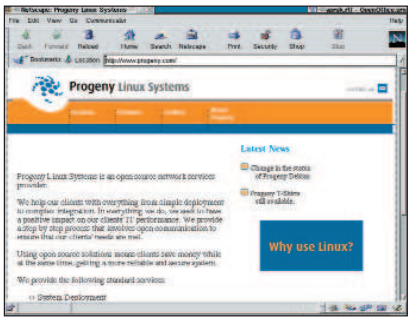
➲ http://www.beos.com



➲ http://blueos.free.fr
➲ http://www.bebits.com
➲ http://http://www.palm.com
➲ http://open-beos.sourceforge.net/

Viszlát, Progeny!

Újabb üzleti Debian-változat dobta be a törölközőt. Sajnos a Progeny Linux is feladta a versenyt: 1.0-s Newton-változatuk ugyan még mindig letölthető a honlapjukról, támogatást is adnak hozzá, de nem fejlesztik tovább, mivel



üzletileg nem éri meg. Október elsején hagyták abba a fejlesztést, a közvetlen eladások október 25-ig folytak, most már a teljes változatot csak néhány viszonteladótól lehet beszerezni. December 31-ig a termékhez járó ingyenes harmincnapos telefonos és három hónapos levélbeni támogatás természetesen még él. A Woodyra átálló felhasználók a ➲ http://www.progeny.com/archive/debian/support/conversion.html oldalon kaphatnak segítséget, ahol részletes leírást olvashatnak a programok frissítéséről, telepítéséről és az átállásról.

➲ http://www.progeny.com

Sun Cobalt

Hol volt, hol nem volt, volt egyszer Cobalt nevű cég, amely hetedhét erdőn túl szép kék dobozokat gyártott. E dobozok világítottak a sötétben, vezetékek kapcsolódtak hozzájuk, és a következő célfeladatokat látták el: nagy teherbírású proxykiszolgáló, vállalati tűzfal és belsőhálózat-kiépítés. Annak idején e dobozok nagy erősségének számított, hogy a testreszabott Linuxot webes felületen keresztül is el lehetett érni, és akár a kezdő rendszergazdák is pillanatok alatt beállíthattak egy Apache- vagy proxykiszolgálót. Már akkortájt látszott, hogy sikeresen megállja a helyét a piacon, amennyiben néhány hiányosságát kiküszöbölik, például a proxykiszolgálót nem IDE-csatolófelületű lemezekkel szerelik (a párhuzamos terhelést ugyanis nehezen bírja), valamint a futtatott Linux-terjesztést biztonsági szempontból folyamatosan karbantartják…

A termék piacra kerülése óta eltelt két év folyamán a Sun megvásárolta a Cobaltot.

Lássuk, hogy ez a hasznos kis masina mára mennyit változott! Jelenleg szinte minden vállalati felhasználási területet sikerült lefedniük, a különböző célfeladatoktól kezdve (behívókiszolgáló) az általános felhasználásig (Web). Időközben a gyerekbetegségeit is kinőtte, a proxyba SCSI-merevlemez került, és a programot is továbbfejlesztették. Már az üzembe helyezés során kiderül, milyen kényelmes eszközt kaptunk kézhez: a Sun Cobalt-termékcsaládra nagyon egyszerűen telepíthetünk operációs rendszert. A készüléket keresztkábel segítségével összekötjük a saját gépünkkel, majd a gépünket CD-ről újraindítjuk. Ilyenkor a két gép között a kapcsolatot egy átmenetileg működő DHCP-kiszolgáló biztosítja. A telepítés menetébe nem szükséges beavatkozni – miután elkészült, a Cobalt kijelzőjén megjelenik a felirat: újraindíthatjuk. Ezentúl gépünket már webes felületen keresztül is felügyelhetjük. Ez azonban rögvest egy gondot is felvet: a 80-as kapun átmenő forgalom alapesetben kódolás nélkül kerül átvitelre, így például beíráskor egy figyelő lefülelheti a jelszavunkat, érdemes tehát már az elején bekapcsolni a biztonságos HTTPS-támogatást. Amennyiben az alaptelepítést a gépünkön felejtjük, számtalan rést hagy rajta (Telnet-kapu, `finger`, `time`, `portmap` stb.), de szerencsére saját magunk is könnyen ki tudjuk kapcsolni őket. Továbbá célszerű a Telnet protokollt azonnal a jóval megbízhatóbb SSH-ra cserélni (cobalt formátumban érhető el hozzá). A fenti leírásból következik, hogy mind webes felügyeletre, mind héjhozzáférésre lehetőségünk nyílik. Dicséretes újítás, hogy a biztonsági frissítéseket kényelmes módon a webes felügyeleti felületen tudjuk beállítani, sőt, akár önműködővé is tehetjük. Ez a lehetőség leginkább a Debian GNU/Linuxban már megismert `apt-get update/upgrade` szolgáltatásra emlékeztet. Fontos figyelembe venni, hogy ezeket az eszközöket úgy alakították ki, hogy a felhasználó által könnyen felügyelhető legyen, ezért mindenképpen ajánlott vagy egy jól beállított tűzfal mögé (lásd *Linuxvilág* 5. szám, 40. oldal) helyezni, vagy magára a gépre tűzfalat telepíteni.

Alapvetően három fő lehetőséggel gazdálkodhatunk:

1. A Sun *Qube* termékcsaládjához úgynevezett „Adaptive Firewall" csomagot (állapottartó csomagszűrő) biztosít, amelyet egy biztonságos webböngészőből könnyedén beállíthatunk. Ha a cél az, hogy megvédjünk egy webkiszolgálót egy kiszolgálózónában, és nem szándékozunk újabb gépet kötni a Cobalt elé, ez nagyon jó megoldás lehet. Sajnos a program nem szabad, csak ingyenesen használható.
2. Astaro Security Linux: önálló Linux-terjesztés, amely a rendszergazdák álmát valósítja meg egy CD-n. A nevéből is kitűnik, hogy fejlesztése során a biztonságra törekvés volt az elsődleges szempont. Bár az Astaro cég nemcsak Cobaltra készít terjesztést, hanem például saját tűzfalat is előállít, az 1.8-as és a 2.0-s (nem tesztelt, de üzembiztos) változatok azonban Cobaltra is elérhetők, és együttműködnek a cobaltossal. A telepítése a cobaltoséhoz hasonló módon zajlik, felügyeletet már csak HTTPS-kapcsolaton keresztül gyakorolhatunk. Szinte mindent kattintva tudunk beállítani, még a VPN-t is. Jó megoldást kínál olyan cégeknek, amelyek 1 hüvelyk magas tűzfalat szeretnének gyorsan és megbízhatóan üzemeltetni, és ne feledkezzünk meg róla: a tűzfalrész ebben is csupán állapottartó csomagszűrő!
2. Zorp OS: lehetőségünk nyílik rá, hogy egy Debian GNU/Linux-alapú moduláris proxytűzfalat használjunk. Ez jelen pillanatban még kísérleti állapotban található, de a Zorp GNU-s változatát a Cobaltra tölthetjük és beüzemelhetjük. Ebben az esetben mind a cobaltos készülék, mind a Zorp tűzfalprogram összes jó tulajdonságát ki tudjuk használni. Amennyiben a Cobaltot kifejezetten és egyedül tűzfalként akarjuk használni, tehát semmilyen kiszolgálót (például Apache) nem szeretnénk rajta üzemeltetni, a Cobaltra is hamarosan megjelenő Zorp OS-t használhatjuk. Ezzel kapcsolatban a legfrissebb hírekért a ➲ http://www.balabit.hu címet keressük fel.

Kapcsolódó címek

A **Sun Cobalt** magyarországi forgalmazója ➲ http://www.avnet.com
Zorp tűzfal ➲ http://www.balabit.hu
Astaro security Linux ➲ http://www.astaro.com
Cobalt oldal ➲ http://www.suncobalt.hu

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.

Séta a LinuxWorld beágyazott oldalán

A CodeWeavers képes olyan megoldást biztosítani, amellyel beágyazott alkalmazások futtathatók Linux alatt, a Wine segítségével…

Annak ellenére, hogy az ez évi nagy linuxos esemény, a West Coast LinuxWorld hangulata a szokásosnál visszafogottabb volt, egyvalamiben szinte teljes volt az egyetértés: a beágyazott Linux az utóbbi 6–12 hónapban hatalmas fejlődésen ment keresztül. Ugyan a rendezvény nem különösebben összpontosít a beágyazott rendszerek piacára, mégis rengeteg olyan új terméket, műszaki megoldást és fejlesztési tervet fedtek fel és mutattak be, amelyek a beágyazott rendszerek és a különféle okos eszközök fejlesztőinek igényeit hivatottak kielégíteni. Az ember bármerre is fordult, mindenhol olyan cégekbe botlott, amelyek zsebtitkárokba, szórakoztató készülékekbe (elsősorban tévé set-top-boxokba), telematikai eszközökbe és vékony ügyfelekbe ágyazták be a Linuxot.

A korábbi kiállításokon a beágyazott Linuxszal kapcsolatos termékek és megoldások a bemutatott termékeknek körülbelül tíz százalékát tették ki. A legutóbbi rendezvényen ez az arány 15–20 százalékra nőtt, ami cseppet sem meglepő, hiszen a VDC, az Evans Data Corporation, az Embedded Systems Programming magazin, a LinuxDevices.com és még sokan mások jelentéseikben számottevő piaci növekedést jeleztek a beágyazott Linux iránt.

Azoknak, akik nem tudtak eljönni a rendezvényre, vagy más kötötte le a figyelmüket, következzék hagyományos beszámolóm a LinuxWorld beágyazott oldaláról, amelyben ábécésorrendben haladva az összes beágyazott és beágyazható dologról említést teszek.

1.

1. kép Kisméretű linuxos számítógép az Advantage Business Computer Systemstől

2. kép Az Earthlink K+F telematikai kutatófelülete Linuxot futtat

3. kép Linux a nappaliban? A HP új Digital Entertainment Centere

4. kép Az IBM koncepcióautójában Linux fut

5. kép Nézd, mire képes a Tiny StrongARM SBC Plus beágyazott Linuxszal!

Applied Data Systems

Az ADS standjának látogatói meghökkentő mennyiségű LCD-kijelzős, StrongARM processzoros, egyetlen áramköri lapból álló számítógéppel találkozhattak – a kütyürajongók paradicsoma. Az ADS saját beágyazott Linuxokat készített, amelyek választható kiegészítőként bármelyik SBC-jéhez elérhetők, de emellett három Java-képességekkel is rendelkező, grafikus alkalmazáskörnyezetet is bemutatott: az Insignia Jeode VM fejlesztése a Lineo Embedix fejlesztésén, az IBM Visual Age Micro Edition a MontaVista Hard Hat Linuxán, a Blackdown JDK-ja pedig az ADS saját beágyazott Linuxán fut. Az ADS augusztusban jelentette be, hogy az Intel a StrongARM és XScale processzoraihoz a céget választotta külső felületfejlesztőjéül, és kifejleszti jelenlegi Graphics Master SBC-jének módosított változatát, amely hivatkozási alaprendszerként fog szolgálni az Intel ügyfelei számára.

➲ http://www.applieddata.net

Advantage Business Computer Systems

A Linux Terminal Server Project (LTSP) standján elrejtőzve mutatta be *David Anders* saját cégének apró Linux-rendszerét – egy valóban remek kis linuxos gépet. Anders elmondta, hogy a hamarosan megjelenő újabb változat már CompactFlash aljzattal is rendelkezni fog, így lehetővé válik a gép bővítése, valamint az eltávolítható adathordozók használata. Szép darab.

➲ http://www.abcsinc.com

Century Embedded Technologies

A Century megszokott helyét foglalta el a RedHat hatalmas pavilonjának aljában. A Century nagy újdonsága a rendezvényen a PIXIL volt, amely korábbi kulcstermékeit (Microwindows és ViewML) feljavítva és továbbfejlesztve egyesíti egy új grafikus környezetből, eszközökből és alkalmazásokból álló készletbe, amelynek célterülete a Linux-alapú zsebtitkárok, webtáblák és vékony ügyfelek piaca. A bemutatók során a PIXIL-t Compaq iPAQ zsebtitkáron láthattuk futni, valamint megtekinthettünk egy PIXIL-alapú set-top-boxot, amely a National Semiconductor SP1SC10 típusú fejlesztői készülékén alapult. *Greg Haerr* – a Century elnöke – kiemelte, hogy ugyan a TV Linux Alliance Project csak a szabványosítási munka kezdetén tart, a Century/National újonnan bejelentett Linux4.TV fejlesztése már elérhető a felhasználók számára. Támogatja a digitális és analóg tévéket, a filmrögzítést, továbbá rendszermag- és középszintű programozási felületet is biztosít.

➲ http://embedded.censoft.com

CodeWeavers

A Codeweavers bemutatta a nemrég bejelentett CrossOver megoldást, amely lehetővé teszi, hogy linuxos rendszereken windowsos böngészőket és alkalmazássegítőket használjunk. *Jeremy White* elnök szerint a CrossOver beágyazott változatának fejlesztése már folyamatban van, átlagos esetben 1 MB memó-

riára és 4,5 MB tárhelyre lesz szüksége. A windowsos beépülő modulok időnként sajnos meglehetősen sok erőforrást igényelnek, ha CrossOver alatt futtatjuk őket. A CodeWeavers, amely jelentős hátteret jelent a Wine számára, képes olyan megoldást biztosítani, amellyel beágyazott alkalmazások futtathatók Linux alatt, a Wine segítségével, valamint közvetlenül is át tudja ültetni az alkalmazásokat Linux alkalmazásfejlesztői felületekre.
➲ http://www.codeweavers.com

Az Earthlink kutató- és fejlesztőcsoportja

A csoport egy nyílt szabványokra épülő, járművek számára készített helymeghatározó telematikai felület mintapéldányát mutatta be, amelyet közlekedési alkalmazások bemutatására, valamint arra terveztek, hogy tanulmányozni lehessen a linuxos alkalmazások és internetes megoldások használatát gépjárművekben, valamint távoli hibafelderítő, m-kereskedelmi és helyzetfüggő alkalmazások megvalósíthatóságát és hatékonyságát. Az Earthlink egy pályázatot is kiírt a Linux, az XML, a Java, a vezeték nélküli és a webes alkalmazások fejlesztői számára. Elég egy jó ötlet, és máris valamelyik érdekes gépükkel játszadozhatunk.
➲ http://www.research.earthlink.net

Embedded Linux Consortium

Az ELC bejelentette, hogy a közelmúltban számos új taggal bővült, köztük található a Future Sound Technologies, az Intel Corporation, az American Megatrends, Inc., az Aleph One, Ltd. és a Vibren Technologies is. Az ELC honlapján található lista szerint jelenleg 68 vállalati tagot számlálnak (36 igazgatási és 32 támogató). Fontos apróság a hírek zuhatagában, hogy az ELC igazgatótanácsa nemrég elfogadta az ELC szabályzatának azon módosítását, amelynek értelmében a szervezet a jövőben szellemi tulajdonnal is rendelkezhet. Ezzel szabaddá vált az út afelé, hogy az ELC beágyazott Linux-szabványokat fejlesszen ki, és engedélyeztessen mások számára.
➲ http://www.embedded-linux.org

Embedded Linux Journal

A LinuxWorld nyitásának napján az ELJ kihirdette az első beágyazott Linux tervezési verseny győzteseit. A győztesek Costa Ricába utazhatnak, minden költségüket a szervező állta.
➲ http://embedded.linuxjournal.com

2.


3.

4.

5.

Empower Technologies

Az Empower bemutatta új LinuxDA nevű „Linux-frissítését” Palm III és V készülékekhez. Úgyszintén bejelentették, hogy a LinuxDA használati jogát megvette két, a Palmokkal egyenértékű készülékeket gyártó tajvani vállalkozás: az Elitegroup Computer Systems és az APlux Communications.
➲ http://www.linuxda.com

Hewlett-Packard Company

A Hewlett-Packard jelentős beágyazott Linux cselekvési tervet jelentett be a LinuxWorld rendezvény kapcsán. A bejelentés szerint a Linuxot választották az összes jövőbeni HP-fejlesztésű eszköz operációs rendszeréül. A HP beágyazott Linux-rendszerének neve Chai-LX. A HP bemutatta új Digital Entertainment Center hangrendszerét is, amely x86-os típusú processzort tartalmaz, és beágyazott operációs rendszerként Linuxot futtat.
➲ http://www.hp.com/linux

IBM

Az IBM alphaWorks új, kísérleti TechMobile fejlesztését mutatta be, egy módosított Ford Explorer 2002 Limited Edition terepjárót, amelyet Linuxot használó, különféle webes alkalmazásokat futtató számítógépekkel szereltek fel. A bemutató során egy egyszerű felhasználói felülettel és Bluetooth-kapcsolattal rendelkező zsebtitkárról vezérelték a gépkocsi fényszóróit, ajtózárait és indítómotorját. A rendezvény különböző pontjain további, a beágyazott Linuxszal kapcsolatos műszaki megoldásokkal lehetett találkozni az IBM háza tájáról, így például DB2 adatbázisokkal, VisualAge Micro Edition (a MontaVista standján) és Embedded ViaVoice (szintén MontaVista) fejlesztésekkel.
➲ http://www.alphaworks.com

Kapcsolódó cím

➲ http://www.empowerthechnologies.com

Rick Lehrbaum (rick@linuxdevices.com) hozta létre a LinuxDevices.com „beágyazott Linuxok portálját”, amely nemrég visszatért alapítójához, miután az LLC a Device Forge-ot. 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.

Egy Bluetooth-kapcsolattal rendelkező zsebtitkár vezérelte az autó fényszóróit, ajtózárait és indítómotorját

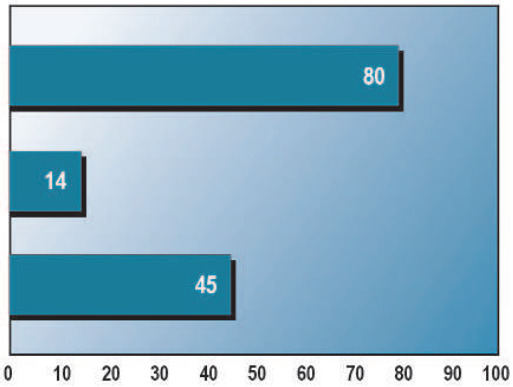
Biztató a jövő a beágyazott Linuxot illetően

Nem nagy újság, hogy a Linux mindenhol megállja a helyét, a „háztartási gépektől a szórakoztató elektronikai cikkeken át az iparban, a repülőkben és az autókban használt vezérlőberendezésekig” – olvashatjuk az Evans Data Corporation (➲ http://www.evansdata.com) jelentésében. Az viszont újdonságszámba megy, hogy milyen magas a Linux részesedése a beágyazott rendszerek területén.

Állításuk szerint a Linux fontos a beágyazott rendszereket fejlesztők közössége számára.

Már van linuxos alkalmazásuk.

Az elkövetkező évben linuxos alkalmazás megjelentetését tervezik.



A „Beágyazott rendszerek fejlesztői körében végzett felmérés” alapján az Evans Data „megdöbbentő változás”-ra hívja fel a figyelmünket „a munkahelyi és otthoni mikroprocesszoros eszközök programjai körében”. Ennek eredményeképpen a cég a beágyazott rendszerekkel kapcsolatos linuxos projektek megháromszorozódására számít a következő év során, ami véleményük szerint „része annak a folyamatnak, amelynek eredményeképpen a házilag fejlesztett operációs rendszerek helyét a kereskedelmi rendszerek veszik át”. Gondot okoz a megfelelő eszközök hiánya. Pontos számadatok közlése nélkül elmondhatjuk, hogy „élénk érdeklődés mutatkozott a fejlesztők munkáját elősegítő eszközök és termékek iránt, amelyek segítségével időre befejezhetik a munkájukat”. A megkérdezett fejlesztők kétharmada elmondta, hogy sok beágyazottrendszer-projekt elindul, de sohasem fejeződik be, még többnek a határideje jelentősen kitolódik.

Mit szeretnek a fejlesztők a Linuxban? Ahogyan várható volt, „a nyílt forráskódot, a felhasználási szerződésdíjak hiányát és a nagy szakértelemmel rendelkező, népes linuxos fejlesztői közösséget” jelölték meg a Linux legnagyobb előnyeiként. „A Linux lassú elfogadásának okai között kiemelkedő helyen szerepel az egyes kártyákat támogató csomagok és az eszközmeghajtók hiánya, valamint a kód széttöredezése.”

Doc Searls
(doc@ssc.com)
a Linux Journal szerkesztője és a Cluetrain Manifesto társszerzője

Ők mondták

Mutass egy olyan webes alkalmazást, amelyet nem lehet egy 100-as Pentiumról kiszolgálni, és én mutatok neked egy halott dot-com céget. (*Kevin Jamieson*)

Ha rendszereket szeretnél gyártani és eladni, és azt akarod, hogy a fejlesztők is melléd álljanak, a rendszerednek nyílt forrásúnak kell lennie. Ez a tény az utóbbi pár évben igaznak bizonyult. (*Dave Winer*)

Az operációs rendszerek háborúja véget ért: az operációs rendszerek vesztettek. Most olyan környezetek fejlődésének lehetünk a tanúi, amelyeknél az alkalmazások több operációs rendszert ölelnek át, és nem fordítva. A személyi számítógép mindössze egyetlen objektum a többi közül egy maroknyi kifelé látszó csatlakozási felülettel és összetett belsővel, amelyet azonban szabványos protokollok rejtenek el a hívó elől. (*Clay Shirky*)

A Webbel nem az a gond, hogy meg kell értenünk a szokatlant, hanem az, hogy rá kell jönnünk, mennyire szokatlan a hétköznapi. (*David Weinberger*)

Milyen kart kellett egy láthatatlan kéznek meghúznia, hogy felélessze a gépben lakozó szellemet? (*Christopher Locke*)

A műszaki fejlődés olyan, mint a balta az elmebeteg gyilkos kezében. (*Albert Einstein*)

Linux-index

1. A Szilícium-völgyben működő 807, tőkében legerősebb cég felsővezetésének az elmúlt pénzügyi évben kifizetett prémium összege: 4,8 milliárd dollár
2. A fenti szám az előző évihez viszonyítva: 2-szeres
3. A fenti cégek részvényárfolyamának esése – az MN Index alapján – ugyanebben az időszakban (százalék): 24
4. A „shit” szó ennyiszer fordult elő a Comedy Centralon sugárzott South Park című műsor első részében a képernyőn futó számláló alapján: 142
5. A Comedy Central a fenti South Park-résszel kapcsolatban ennyi elektronikus levelet kapott: 4
6. A káromkodást támogató levelek százaléka: 100
7. Az Egyesült Államok Szabadalmi Hivatala által jóváhagyott szabadalmak száma 2000-ben: 158 118
8. Az IBM helyezése a 2000-ben jóváhagyott szabadalmak száma alapján: 1
9. Az IBM-nek 2000-ben megítélt szabadalmak száma: 2 886
10. Az amerikai cégek száma a 10 legtöbb amerikai szabadalmat kapott cég között (2000-ben): 4
11. A japán cégek száma a 10 legtöbb amerikai szabadalmat kapott cég között (2000-ben): 6
12. A Webvan vesztesége 2001 júliusában, amikor csődbe ment: 860 millió dollár
13. A Google által bejárt weblapok száma: 1 346 966 000
14. A Google találatai alapján ennyi weblapon szerepel a „sun” kifejezés: 25 500 000
15. A Google találatai alapján ennyi weblapon szerepel a „microsoft” kifejezés: 20 200 000
16. A Google találatai alapján ennyi weblapon szerepel a „dell” kifejezés: 14 700 000
17. A Google találatai alapján ennyi weblapon szerepel a „solution” kifejezés: 13 300 000
18. A Google találatai alapján ennyi weblapon szerepel az „ibm” kifejezés: 11 200 000
19. A Google találatai alapján ennyi weblapon szerepel a „unix” kifejezés: 10 900 000
20. A Google találatai alapján ennyi weblapon szerepel a „perl” kifejezés: 7 650 000
21. A Google találatai alapján ennyi weblapon szerepel a „python” kifejezés: 2 070 000
22. A Google találatai alapján ennyi weblapon szerepel a „linux” kifejezés: 31 600 000
23. A Google által indexelt 1000 lapból ennyin szerepel a „linux” kifejezés: 2,35
24. Azok százaléka, akik az elsők között alkalmazták a PVR-t (Linux-alapú videorögzítő megoldás, mint például a TiVo), és ma többet néznek tévét, mint korábban: 63
25. Azok százaléka, akik használják a PVR-t, és nem tudják, hogy a felvett, majd később megnézett műsorok eredetileg melyik csatornán voltak láthatók: 12
26. Azok százaléka, akik PVR-rendszert vagy hagyományos videomagnót használnak és átugorják a hirdetéseket: 25
27. Az Open Source Initiative által elfogadott felhasználói engedélyek száma: 22
28. A vezető 74 márkából ennyi százalék vesztett márkaértékéből az elmúlt tíz esztendőben: 55
29. A 74 vezető márka ez idő alatt ennyit vesztett az értékéből: 5%
30. A múlt év során megjelent új termékek száma: 31 432
31. A zene felvételére alkalmas készülékek gyártói bevételük ennyi százalékát fizetik ki szerzői jogdíjként az American Home Recording Act (AHRA) értelmében: 2
32. Az AHRA szerzői jogdíjakból ekkora arányban részesedik a Sound Recording Found (Hangrögzítési Alap): kétharmad
33. A fent említett alapból ennyi százalék jut a „nem jegyzett zenészeknek és a vokálénekeseknek”: 4
34. A fennmaradó részből ennyi százalék jut a „jegyzett, a felvételt készítő művésznek”: 40
35. A fennmaradó részből ennyi százalék jut a „hangfelvételek másolására és terjesztésére az adott évben kizárólagosan jogosult tulajdonosnak”: 60
36. A RedHat aránya a Linux Counter által nyilvántartott gépek között: 28,39 %
37. A Slackware aránya a Linux Counter által nyilvántartott gépek között: 22,40 %
38. A Debian aránya a Linux Counter által nyilvántartott gépek között: 19,30 %

Forrás

1–3: San Jose Mercury News
4–6: The New Yorker
7–11: United States P atent and Trademark Office
12: The Wall Street Journal
13–23: Google, 2001. július 12.
24–25: Electronic Media a NextResearchből
26: The Wall Street Journal
27: Open Source Initiative (➲ http://www.opensource.org)
28–30: Financial Times
31–35: RIAA
36–38: Linux Counter

Cégcsokor (4. rész)

Sorozatunkban olyan cégeket gyűjtünk csokorba, amelyek huzamosabb ideje számos területen Linuxot alkalmaznak.

MT-Telecom Kft.

Elsőként lássuk, hogy milyen kiépítést használ ez a cég: három linuxos kiszolgáló üzemel náluk, amelyek Debian Linuxot futtatnak.
Az első gép fájlkiszolgálóként működik: 400 MHz-es Celeron processzoros, 128 MB RAM-mal, egy 4 GB-os és két 13 GB-os merevlemezt tartalmaz programból megvalósított RAID-del. Az alábbi alkalmazások futnak rajta:

- fájlkiszolgáló – Samba,
- faxkiszolgáló – Hylafax,
- DNS-kiszolgáló – Bind,
- DHCP-kiszolgáló – DHCP.

A második vas tűzfalként üzemel: 400 MHz-es Celeron processzorral van felszerelve és 64 MB RAM-mal. Proxykiszolgálónak Squidet, valamint POP3-proxyt használnak. A jövőben FTP-proxy és Socks bevezetését is tervezik.
A harmadik masina webkiszolgálóként dolgozik: 500 MHz-es Celeron processzorral és 128 MB RAM-mal. Webkiszolgálóként Apache-ot, levelezőkiszolgálóként pedig `qmail`-t és `vpopmail`-t használnak.
Az elképzeléseik között szerepel egy FTP-kiszolgáló (proftpd) és egy Mapguide kiszolgáló kialakítása is. Linuxot a munkaállomásokon is használnak az irodai alkalmazások futtatására és kipróbálására.

Interware Kft.

A cégnél Compaq Proliant, illetve AlphaServer DS20, ES40-es gépeket használnak Debian Linux alatt. A legnagyobb gépük egy Alphaserver ES40, 1 GB RAM-mal és 36 GB merevlemezzel. Ezen jelenleg átmenetileg körülbelül 5500 felhasználó levelezése fut, a közeljövőben azonban komolyabb igénybevételét is tervezik, ugyanis sokszorosan túlméretezték mind processzor, mind memória tekintetében. Az átlagos kiszolgáló Compaq Proliant DL380, 256 MB RAM-mal és körülbelül 36 GB merevlemezzel. A Linuxra bízott feladatok a következők: proxykiszolgáló – Squid; webkiszolgáló – Roxen Challenger; levelezőkiszolgáló – Exim + Courier + IMHO; hálózatfelügyelet – snmpd + mrtg; a felhasználók karbantartása MySQL adatbázis-kezelőn keresztül történik; végezetül DNS-szolgáltatás – Bind. A Linuxok, mint az már fentebb szóba került, megközelítőleg 5500 felhasználót szolgálnak ki. Nagyon sok saját fejlesztésű webes programot használnak, amelyek előállítását megkönnyítette a Linux alatt elérhető programozási nyelvek és eszközök bőséges tárháza. Az összes kiszolgáló fontosabb adatainak biztonsági mentése jelenleg hálózaton keresztül történik (rsync segítségével) külön kiszolgálóra, RAID5-be kötött merevlemezekre, de DLT vásárlását tervezik. Amennyiben ez sikerül, szalagra történő mentésre fognak átállni.

Országos Állategészségügyi Intézet

Célkitűzésük a levelezés, a dokumentum- és ügyiratkezelés, valamint az egységes felületen történő személyzeti nyilvántartás. Az intézet által megvalósított megoldást az alábbiakban vázoljuk fel.
Az állategészségügy PHARE-keretből körülbelül 800 felhasználós Lotus Notest vásárolt 2000-ben. Ez a következő helyszínekre bontható:

- 20 megyei állategészségügyi állomás;
- 5 állategészségügyi intézet;
- 40 állategészségügyi határállomás;
- 1 FVM állategészségügyi főosztály.

A megyei állomások, intézetek és az FVM-beli főosztály helyi hálózatokkal (átlagosan 25 felhasználó) rendelkezik, a határok pedig egygépes végpontok. Ezek mindegyike csillagszerűen kapcsolódik az Országos Állategészségügyi Intézetben (OÁI) kialakított Országos Állategészségügyi Informatikai Központhoz (OÁIK). Minden helyi hálózatra egy SuSE 7.0-t futtató IBM Netfinity 3000-es gép van telepítve 500 MHz-es Pentium III processzorral, 128 MB RAM-mal és 9 GB-os UltraWide-os SCSI merevlemezzel felszerelve ezek a helyi Lotus Notes Domino kiszolgálót futtatják. Éjszakánként mindegyikük összehangolja egymással az adatait. A központban egy kétkiszolgálós Notes Enterprise Cluster van telepítve RAID5-ös merevlemez alrendszerrel. A Lotus Notes saját telepmegoldását alkalmazzák. Mindkét kiszolgálóban három IBM UltraWideSCSI3 merevlemez található, és a rajtuk lévő lemezrészeken ReiserFS fájlrendszert használnak.
Az üzembiztonság további növelésére azonban a merevlemezek tükrözésének kipróbálását is tervezik (a helyi Notes-kiszolgálókon is).
Az eltelt idő alatt láthatóvá vált, hogy a lotusos rendszer jól működtethető, amelyben a Linuxnak jelentős szerep jutott. Régóta foglalkoznak Unixokkal (főleg SGI IRIX-szel és PC-s Linuxszal), és véleményük szerint komoly kiszolgálórendszert nem is lehet Microsoft-alapokra helyezni: mind az ár-teljesítmény viszony tekintetében, mind megbízhatóságban a Linux lényegesen jobb.

Carnation Internet Consulting Rt.

A részvénytársaságnál kilenc PC-alapú kiszolgáló alkotja az ügyfeleket kiszolgáló gépparkot. A kiszolgálókban Celeron és Pentium II processzorok találhatók, a memóriájuk 32-től egészen 256 MB-ig terjed. 8 GB-os IBM SCSI, illetve 10 GB-os WD és Quantum merevlemezek tárolják a programokat és az adatokat. Az Abit és az Asus alaplapokat részesítik előnyben, ezek közül is a következő típusokat: BH-6, BX-6, P2B-F és P3-BF. Minden kiszolgálón Linux fut, egyelőre RedHat, Slackware és Debian vegyesen. Szándékukban áll azonban a kiszolgálópark egységesítése, és mivel a Slackware Linuxot kedvelik, valószínű, hogy rá fog esni a választásuk.
A fájlkiszolgáló és a tartománykiszolgáló feladatokat két gép látja el Samba segítségével. Egy levelezőkiszolgáló Sendmailen keresztül biztosítja a cég, illetve az egyéb tartományok kapcsolatát a külvilággal. A listákhoz egy

Cégcsokor (4. rész)

Sorozatunkban olyan cégeket gyűjtünk csokorba, amelyek huzamosabb ideje számos területen Linuxot alkalmaznak.

MT-Telecom Kft.

Elsőként lássuk, hogy milyen kiépítést használ ez a cég: három linuxos kiszolgáló üzemel náluk, amelyek Debian Linuxot futtatnak.
Az első gép fájlkiszolgálóként működik: 400 MHz-es Celeron processzoros, 128 MB RAM-mal, egy 4 GB-os és két 13 GB-os merevlemezt tartalmaz programból megvalósított RAID-del. Az alábbi alkalmazások futnak rajta:

- fájlkiszolgáló – Samba,
- faxkiszolgáló – Hylafax,
- DNS-kiszolgáló – Bind,
- DHCP-kiszolgáló – DHCP.

A második vas tűzfalként üzemel: 400 MHz-es Celeron processzorral van felszerelve és 64 MB RAM-mal. Proxykiszolgálónak Squidet, valamint POP3-proxyt használnak. A jövőben FTP-proxy és Socks bevezetését is tervezik.
A harmadik masina webkiszolgálóként dolgozik: 500 MHz-es Celeron processzorral és 128 MB RAM-mal. Webkiszolgálóként Apache-ot, levelezőkiszolgálóként pedig `qmail`-t és `vpopmail`-t használnak.
Az elképzeléseik között szerepel egy FTP-kiszolgáló (proftpd) és egy Mapguide kiszolgáló kialakítása is. Linuxot a munkaállomásokon is használnak az irodai alkalmazások futtatására és kipróbálására.

Interware Kft.

A cégnél Compaq Proliant, illetve AlphaServer DS20, ES40-es gépeket használnak Debian Linux alatt. A legnagyobb gépük egy Alphaserver ES40, 1 GB RAM-mal és 36 GB merevlemezzel. Ezen jelenleg átmenetileg körülbelül 5500 felhasználó levelezése fut, a közeljövőben azonban komolyabb igénybevételét is tervezik, ugyanis sokszorosan túlméretezték mind processzor, mind memória tekintetében. Az átlagos kiszolgáló Compaq Proliant DL380, 256 MB RAM-mal és körülbelül 36 GB merevlemezzel. A Linuxra bízott feladatok a következők: proxykiszolgáló – Squid; webkiszolgáló – Roxen Challenger; levelezőkiszolgáló – Exim + Courier + IMHO; hálózatfelügyelet – snmpd + mrtg; a felhasználók karbantartása MySQL adatbázis-kezelőn keresztül történik; végezetül DNS-szolgáltatás – Bind. A Linuxok, mint az már fentebb szóba került, megközelítőleg 5500 felhasználót szolgálnak ki. Nagyon sok saját fejlesztésű webes programot használnak, amelyek előállítását megkönnyítette a Linux alatt elérhető programozási nyelvek és eszközök bőséges tárháza. Az összes kiszolgáló fontosabb adatainak biztonsági mentése jelenleg hálózaton keresztül történik (rsync segítségével) külön kiszolgálóra, RAID5-be kötött merevlemezekre, de DLT vásárlását tervezik. Amennyiben ez sikerül, szalagra történő mentésre fognak átállni.

Országos Állategészségügyi Intézet

Célkitűzésük a levelezés, a dokumentum- és ügyiratkezelés, valamint az egységes felületen történő személyzeti nyilvántartás. Az intézet által megvalósított megoldást az alábbiakban vázoljuk fel.
Az állategészségügy PHARE-keretből körülbelül 800 felhasználós Lotus Notest vásárolt 2000-ben. Ez a következő helyszínekre bontható:

- 20 megyei állategészségügyi állomás;
- 5 állategészségügyi intézet;
- 40 állategészségügyi határállomás;
- 1 FVM állategészségügyi főosztály.

A megyei állomások, intézetek és az FVM-beli főosztály helyi hálózatokkal (átlagosan 25 felhasználó) rendelkezik, a határok pedig egygépes végpontok. Ezek mindegyike csillagszerűen kapcsolódik az Országos Állategészségügyi Intézetben (OÁI) kialakított Országos Állategészségügyi Informatikai Központhoz (OÁIK). Minden helyi hálózatra egy SuSE 7.0-t futtató IBM Netfinity 3000-es gép van telepítve 500 MHz-es Pentium III processzorral, 128 MB RAM-mal és 9 GB-os UltraWide-os SCSI merevlemezzel felszerelve ezek a helyi Lotus Notes Domino kiszolgálót futtatják. Éjszakánként mindegyikük összehangolja egymással az adatait. A központban egy kétkiszolgálós Notes Enterprise Cluster van telepítve RAID5-ös merevlemez alrendszerrel. A Lotus Notes saját telepmegoldását alkalmazzák. Mindkét kiszolgálóban három IBM UltraWideSCSI3 merevlemez található, és a rajtuk lévő lemezrészeken ReiserFS fájlrendszert használnak.
Az üzembiztonság további növelésére azonban a merevlemezek tükrözésének kipróbálását is tervezik (a helyi Notes-kiszolgálókon is).
Az eltelt idő alatt láthatóvá vált, hogy a lotusos rendszer jól működtethető, amelyben a Linuxnak jelentős szerep jutott. Régóta foglalkoznak Unixokkal (főleg SGI IRIX-szel és PC-s Linuxszal), és véleményük szerint komoly kiszolgálórendszert nem is lehet Microsoft-alapokra helyezni: mind az ár-teljesítmény viszony tekintetében, mind megbízhatóságban a Linux lényegesen jobb.

Carnation Internet Consulting Rt.

A részvénytársaságnál kilenc PC-alapú kiszolgáló alkotja az ügyfeleket kiszolgáló gépparkot. A kiszolgálókban Celeron és Pentium II processzorok találhatók, a memóriájuk 32-től egészen 256 MB-ig terjed. 8 GB-os IBM SCSI, illetve 10 GB-os WD és Quantum merevlemezek tárolják a programokat és az adatokat. Az Abit és az Asus alaplapokat részesítik előnyben, ezek közül is a következő típusokat: BH-6, BX-6, P2B-F és P3-BF. Minden kiszolgálón Linux fut, egyelőre RedHat, Slackware és Debian vegyesen. Szándékukban áll azonban a kiszolgálópark egységesítése, és mivel a Slackware Linuxot kedvelik, valószínű, hogy rá fog esni a választásuk.
A fájlkiszolgáló és a tartománykiszolgáló feladatokat két gép látja el Samba segítségével. Egy levelezőkiszolgáló Sendmailen keresztül biztosítja a cég, illetve az egyéb tartományok kapcsolatát a külvilággal. A listákhoz egy

levelezőlista-kiszolgálót használnak a mailman program segítségével. Egy proxy- és három webkiszolgáló is dolgozik a cégnél: a proxyn Squid fut, a webkiszolgálókon pedig Apache. Az Apache PHP4 és MySQL segítségével adatbázisból is képes webes felületen keresztül adatokat szolgáltatni. Egy csomagszűrő tűzfalat ugyancsak üzemeltetnek.

Az első pillanatban talán túl soknak tűnhet a kiszolgálók száma, és kevésnek az egy gép által nyújtott szolgáltatás, de azt a filozófiát vallják, miszerint egy nagy kiszolgáló helyett sokkal hasznosabb az egyes feladatok elkülönített gépeken történő megvalósítása. Így egy gép esetleges kiesése nem veszélyezteti az összes ember munkáját. A kiszolgálók változó fájljairól 2–3 naponta biztonsági másolat készül, hetente egyszer pedig teljes mentés.

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 Linux új piacokat hódít meg

Két évvel ezelőtt fejlesztési igazgatóhelyettes voltam egy nagy, távmarketinggel foglalkozó cégnél. A feladatom rendkívül nagy adatbázisok építése és karbantartása volt, ezekben tároltuk a célzott marketinghez szükséges adatokat. Jelentős kiadásaink voltak: egyrészt rengeteg pénzbe került a fejlesztői eszközök felhasználási jogainak a megvásárlása, a cégnél alkalmazott egyéb programokra pedig a gatyánk is rament. Néhány megoldásunk jól bevált, drága programjaink azonban még mindig nem látták el a vezetőséget olyan megfelelő adatokkal, amelyek alapján megfontolt üzleti döntéseket hozhattak volna. Például túl későn fértünk hozzá a kampányok sikerességét vagy sikertelenségét mutató adatokhoz, és a készpénzáramlásról is csak elképzeléseink voltak. Elhagytam a céget, amikor bezárták a fejlesztői részleget, és rosszul esett, hogy a munkám kárba veszett. Engem is zavart, hogy a cég alapvető üzleti gondjaira nem született kielégítő megoldás, és arról is volt elképzelésem, hogy a cég vezetőit mennyire elkeserítette programok felhasználási engedélyeire kifizetett tetemes összeg. Néhány hónapig a saját vállalkozásom keretében foglalkoztam adatbázis-tervezéssel, mígnem állásajánlatot kaptam egy közepes méretű cég, az Action Target ügyvezetőjétől. Az Action Target lőtereket rendez be, és ehhez gyárt felszereléseket. Először meghökkentett, amit tapasztaltam. Az ügyvezető (egy vállalkozó szellemű villamosmérnök) megismertetett a cégével, és mindenütt csak Linuxszal találkoztam. Minden egyes felhasználó – az értékesítéstől a szállítmányozáson és beszerzésen át a gyártásig, a fejlesztésig és a könyvelésig – linuxos munkaállomáson dolgozott, amelyek többsége lemezmentes volt. A munkaállomásokat egyetlen kiszolgálócsoport látta el, amelyek között webkiszolgáló, adatbázis-kiszolgáló és egy maroknyi alkalmazáskiszolgáló is szerepelt. Az alkalmazáskiszolgálók a programok mellett a felhasználók könyvtárainak is otthont adtak. Az alkalmazások többsége nyílt forrású vagy szabad program volt, így a StarOffice, Netscape vagy TGIF, de használtak néhány zárt forrású programot is, például Applixot. Az adatbázist PostgreSQL-kiszolgálóval valósították meg. A hálózati megosztásokat NFS segítségével működtették, a lemezmentes munkaállomások `bootp` protokollon keresztül csatlakoztak az alkalmazáskiszolgálókhoz.

Amit ezek után láttam, még jobban megdöbbentett. Nem elég, hogy az egész cég Linuxot használt, az ügyvezető saját ERP (Enterprise Resource Planning – vállalati erőforrás-tervező) programot írt. Elmondta, hogy több más megoldáson is gondolkodott, de mind drágának bizonyultak és költséges, zárt forrású RDBMS-megoldásokon futottak, ráadásul, ami a legfontosabb, nem támogatták megfelelő mértékben a testreszabást. Ezért úgy döntött, saját kezűleg oldja meg a gondot. A megvalósításhoz Wylibet használt, amely szintén saját fejlesztés volt. A Wylib egy Tcl/Tk-ban írt programkönyvtár-gyűjtemény. A Tcl/Tk-t könnyű használni, rendszerfüggetlen, héjként is alkalmazható, valamint a nyílt forrás közössége is elismeri és elfogadja. Az adatbázis-kezelést a PostgreSQL-re bízták.

Ez az ERP-rendszer működteti a céget. Az alkalmazottak használják, nélküle nem tudják elvégezni a munkájukat. Az ERP nyílt forrású programokon alapul, és teljes mértékben testreszabható. Ha valami nem tetszik benne, egyszerűen megváltoztatjuk. A cég vezetése úgy becsüli, hogy ez a rendszer egymillió dollárt takarított meg, ha csak a működési költségeket vesszük figyelembe, és ekkor a megtakarításba még nem számoltunk bele a programok felhasználási engedélyeinek szükségtelensége által nyert összegeket. Egy évet dolgoztam ennél a cégnél, a Wylib segítségével újraírtam az ERP-t és átszerveztem az adatbázist, majd saját céget hoztam létre WyattERP néven. A WyattERP célja kettős: egyrészről meg szeretné mutatni a cégeknek, hogyan takaríthatnak meg rengeteg pénzt a nyílt forrású megoldásokkal, másrészről azt is, hogy miként hozhatnak létre olcsó és az üzleti igényeikhez tökéletesen alkalmazkodó ERP-megoldást a Wylib segítségével.

A Wylib és a példaforráskód a ➲ http://www.wyatt-erp.com címről letölthető a Webről. A Wylib nyílt forrású, és az OPL-ben (Open Public License) foglaltak alapján terjeszthető. Ha személyesen is kapcsolatba szeretnél lépni velem, használd a weblapon megtalálható címemet.

Merrill Oveson

Egy nyomozás története

Ugye, mindenki elképzelte már a következő helyzetet: valami gond van a kiszolgálóval, amelyet üzemeltetünk – szerencsésebb, ha a karbantartását nem magunk végezzük, hiszen így nem mi hibáztunk –, és nekünk ki kell nyomoznunk a tettest.

Egyből rohanunk, hogy megoldjuk az esetet: esetleg már otthonról elkezdhetjük a munkát, de semmiképpen sem fejezhetjük be. El kell mennünk a kiszolgálóhoz, mert a gépeknél is szükség van „személyes kapcsolat"-ra, továbbá jobban látszik, hogy mennyire súlyos a gond.
A helyszínen azután belefutunk egy-két ügyes félrevezetésbe és csapdába, melyeket azonban megoldunk. Egy-két leleményes eljárás – amikor már a remény is elfogyott, természetesen csakis ekkor! –, és váratlan fordulatokkal dűlőre visszük az ügyet. A rossz ember lebukik, mi pedig besöpörhetjük az elismerést, a kollégák elismerő pillantásai nyomán pedig jó érzés tölt el minket.
Na, eddig tartott a mese.

```
[ago@mdk ago]$ telnet mail.ceg.hu 25
Trying X.Y.W.Z
Connected to X.Y.W.Z
Escape character is '^]'.
220 mail.ceg.hu ESMTP Postfix
MAIL FROM: ago@lsc.hu
250 Ok
RCPT TO: bill@microsoft.com
551 Reciepent address rejected. Relay access denied...
QUIT
```

A valóság: kétórás modemes kapcsolat otthonról, sok káromkodás és infarktusközeli élmény, egy fél tábla Boci csoki elfogyasztása szigorúan a stresszoldás végett, és befejezésképpen az áhított siker is megérkezik, amelyre azonban a frissen szerzett tapasztalatok birtokában már nem is vágyakoztunk annyira. Amennyiben választani lehetne, szívesebben szavaznék a nyugodt életre. A továbbiakban elolvashatjuk, mi is történt pontosan, milyen intézkedéseket foganatosítottam, és mi lett a következménye. A szereplők természetesen névtelenek maradnak, hogy senki személyiségi jogait ne sértsük meg.

Első felvonás

Éppen egy tanároknak szóló tanácskozáson tartózkodtam és az előadásomra készültem. Egy órával a nevezetes esemény előtt hangüzenet érkezett a telefonomra, de nem volt térerő, amikor hívni próbáltak. Az üzenetben egy barátom közölte, hogy levélszemetet (spam) kapott, és mit tudok mondani az info@gepnev címről, ugyanis tudomása szerint minket bíztak meg a karbantartásával. Kicsit elcsodálkoztam, mert az említett gépen levelezőszolgáltatások nem futottak, csupán egy átjárógép volt két alhálózattal. Ezen keresztül érhetik el az adott cég számítógépei a külső címtartományban lévő levelező- és egyéb kiszolgálókat. Felhívtam az említett cég telephelyén tartózkodó rendszergazdánkat, hogy nézze meg, nem fut-e az átjárón levelezőkiszolgáló (MTA). Természetesen nem futott, viszont célszerű volt rákérdezni, hiszen sohasem árt. Ezután már megnyugodva hívtam fel a hangüzenethagyót és közöltem, hogy valószínűleg meghamisították a levél fejlécét – ezért gondolhatta, hogy a levelet tőlünk kapta. Aznap este könnyű szívvel tértem nyugovóra.

Második felvonás

Másnap hazaérés és rövid pihenés után modemen keresztül csatlakoztam a Világhálóra. Amint beléptem, és elolvastam a beérkezett leveleket, az ellazultság legalább annyira távol került tőlem, mint a gonosz kismalactól a jóindulat. Többen is visszajeleztek, hogy levélszemetet kaptak, amely a fejlécek tanulsága szerint valóban tőlünk származott. Ezt megerősítette a levelezési forgalomról készült előző napi kimutatás, ami egyébként mindig lefut. Valaki – akkor még ismeretlen személy – 7241 kéretlen levelet továbbított a kiszolgálón keresztül! Mi is erősítette ezt meg? A visszajelzések között akadt olyan, amely a teljes levelet idézte a fejléccel együtt. Az eredetileg kiküldött levélben a `From:` szó után az a levélcím állt, amelyet a küldő a levelezőprogramba írt be. A továbbítókiszolgálókat azonosító fejlécrészlet pedig az ügyfelet kiszolgáló átjárót, valamit a tényleges levelezőkiszolgálót azonosította. A jelentés és az eredeti levél tehát egyértelművé tette: tényleg rajtunk keresztül küldték a levelet. A következőt kellett kiderítenem: valóban nyílt levéltovábbító-e a kiszolgáló vagy egyéb módon került meg a levél? Az egyéb lehetőségek közé soroltam még: a kiszolgálót feltörték és egy telepített program segítségével küldték ki a leveleket vagy a törés után nyílt levéltovábbítót csináltak belőle. Amennyiben nyílt levéltovábbítóként működik, akkor is meg kell vizsgálnom, nem történt-e valami sokkal rosszabb a háttérben. Ha megtörték, és nyílt levéltovábbító lett, valaminek futnia kellett ott. A nyomozást a lehető legegyszerűbben kezdtem: kipróbáltam, lehet-e nem engedélyezett címre levelet küldeni. Ennek legegyszerűbb módja a Telnet program használata. A segítségével csak rá kell kapcsolódni a levelezőkiszolgáló 25-ös kapujára és ellenőrizni. Aki követte a cikkeimet, annak már ismerős az a folyamat, amit az *1. lista* szemléltet.
Ez kicsit megnyugtatott, eszerint tehát rajtunk keresztül nem lehet akárkinek levelet küldeni. Ekkor számba vettem a második lehetőséget: a számítógépet megtörték. Mit tesz ilyenkor az előrelátó ember? Elővarázsolja a kiszolgáló „tiszta állapotát" tartalmazó fájlt, és az AIDE program segítségével összehasonlítja a rendszer binárisait. Kicsit morogni kezdtem, ugyanis az elutazásom előtti napon általános programfrissítést hajtottak végre. A rendszeren Debian Woody fut, amelyhez az újabb csomagokat is letöltöttem. Az új fájlt viszont mindig helyileg szeretem elkészíteni, ami azonban az utazás miatt elmaradt. Ez bizony balszerencse volt, nem engedhettem volna meg – le is szidtam magam. A következő lépésben meghívtam a `debsum` nevű programot, ami a telepített programokhoz tartozó ellenőrzőösszegeket – az úgynevezett `MD5 summ`-okat – hasonlítja össze a binárisok jelenlegi állapotával. Ha bármelyik futtatható fájlt lecserélték,

ez a program kideríti. Ámde hogyan bízhatnánk meg az esetlegesen gyanúba keveredett gépen lévő ellenőrzőösszegekben? Sehogy. Mindenesetre egyelőre tisztának tűnt minden, tehát meghívtam a `netstat` programot. Ez a program a hálózathoz kapcsolódó programok állapotáról ad tájékoztatást. Megmutatja, hogy a kiszolgáló milyen kapcsolatot kezel jelenleg. Én a *2. listán* látható módon indítottam el. Természetesen a kimeneten kívül több más érték is szerepel itt. A program ezekkel a kapcsolókkal mutatja meg, hogy melyik program vár kapcsolatot melyik kapun, és milyen IP-címen teszi azt. Itt ért az első szívroham, ugyanis egy számomra ismeretlen kaput fedeztem fel ismeretlen démonnal. Majd megnyugodtam, mert csupán az otthoni gépemen futó kísérleti program várta a kapcsolatokat. Még otthon kezdtem el kipróbálni és a gépemen maradt. Másodszor már a kiszolgálón sikerült lefuttatnom a programot, ahol mindent rendben lévőnek találtam. A biztonság kedvéért azonban tovább piszkáltam a rendszert. Átnéztem az összes parancsfájlt, amely a `cron`, az `at` és a Postfix programokhoz kapcsolódik, ugyanis ezeket a szolgáltatásokat le akartam állítani. Azt azonban mindenképpen el szerettem volna kerülni, hogy amennyiben a kiszolgálót mégis feltörték, és esetleg huncut `rm -rf /` parancsot írtak a vezérlő parancsfájlba, annak következményei legyenek. Miután mindent rendben találtam, a szolgáltatásokat leállítottam. Miért volt ez fontos? Ha a rendszert feltörték, és egy olyan binárist módosítottak, amelyet a `cron` is használ, a rendszer esetleg önműködően újra meg újra megnyílik. Ráadásul a Postfix még számos helyre el akarta küldeni a leveleket. Ezeket a leveleket a leállítás után töröltem. Jó néhány akadt, ezért miután meggyőződtem róla, hogy „rendes" levél nincs a sorban, az egész várakozási sort töröltem (szerencsére hétvége volt, ezért ez csaknem természetesnek tekinthető). Ezután a ➲ http://www.debian.org-ról leszedtem a binárisokhoz tartozó MD5-ös ellenőrző összegeket és átmásoltam őket a gépre, majd így ellenőriztem a binárisokat. Ezt az átjárón is megismételtem. Mindkét rendszer teljesen rendben volt, tehát a törést és a nyílt levéltovábbító gondokat elfelejthettem. Már csak egyetlen lehetőség maradt.

2
```
[ago@mdk ago]$ netstat -antl
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address     Foreign Address         State
tcp        0      0     0.0.0.0:25          0.0.0.0:*           LISTEN
```

Harmadik felvonás

Mivel a kiszolgáló csak egy helyről fogadott el leveleket továbbításra, nem volt nehéz dolgom, hogy behatároljam a következő keresési területet. Ez a gép az átjáró volt – az egyetlen gép, amely felől a küldés engedélyezett. Ezen az átjárón igazából nem fut semmi, csak címeket fordít és kapcsolatokat továbbít. A támadás tehát a mögötte lévő egész belső hálózatról jött. Hogyan lehet megkeresni a saját berkeinkben megbúvó támadót? Miután ismét elindítottam minden szolgáltatást a levelezőkiszolgálón, figyelmemet az átjáróra, pontosabban a naplófájlokra összpontosítottam. Mivel a levelezőkiszolgáló eléréséhez címet kell fordítanom, vagyis álcáznom (masquerading) kell, a csomagszűrővel az erre vonatkozó szabály volt beállítva. Csakhogy a szabály végén megadtam, hogy fordításnál *minden* kimenő csomagról írjon jelentést a naplófájlba. A parancs hasonlóan néz ki:

```
/sbin/ipchains -A forward -i -p tcp
➥-s 192.168.1.0/24 -d mail.ceg.hu 25
➥-j MASQ -l
```

Ez fordította át a belső hálózat címeit az átjáró IP-címére, amennyiben az ügyfelek kapcsolatba akartak kerülni a levelezőkiszolgálóval. A `-l` kapcsolta be a naplózást. Az előző napi naplófájlból kikerestem – természetesen a segédprogramok használatával –, hogy melyik ügyfél kapcsolódott kiemelkedően sokszor a levelezőkiszolgálóhoz. A segédprogram saját készítésű volt, ezért a kiosztott címtartomány minden ügyfelére összesítést készített, és hogy hány bejegyzést talál az adott IP-címhez. Majd a kiemelkedően nagy számú küldő címét felhasználva kikerestem a listából, hogy kihez tartozik a cím. Ekkor csörrent meg a telefon. Maga az elkövető volt. Honnan tudta meg, hogy nyomozok? Az egyik felháborodott partner ugyanis nemcsak nekem küldte el panaszát, hanem a levél küldőjének is – mármint arra a címre, amit az elkövető írt be, és ahova a válaszleveleket a megrendelésekkel együtt várta (ugyanis egy szolgáltatást hirdetett). Mivel csoportos választ nyomtam, ő is megkapta válaszomat – ez idegességemben fel sem tűnt nekem. Nos, próbálta magát mentegetni, hogy nem gondolta, mekkora baj lesz ebből, és különben is otthonról akarta végezni. Ennek azonban jócskán ellentmondott, hogy a saját bevallása szerint is mailbomber programot használt. A következő héten hétfőn leadtam a jelentést a cég vezetőjének és szóban is tájékoztattam az eseményekről. Ennek eredményeképpen az illetőt még aznap elbocsátották.

Miért is? A cég erőforrásait használta és mivel levélszemét jött a levelezőkiszolgálóról, néhány másik levelezőkiszolgálóról kitiltották a tőle érkező *összes* levelet. Ezenkívül a céget erkölcsi kár is érte.

Összegzés

Egyszer minden „csínytevésre" fény derül, és egy ilyen nyomozás inkább fárasztó és idegesítő, semmint jó móka. Gondolom, jövője ismeretében a vétkes is másképp cselekedett volna.

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 2001. évi LinuxJournal szerkesztői szavazásának eredménye

Sokat változott a Linux világa a tavalyi szerkesztői szavazás óta. Az idő tájt a technológiai és .com-fellendülés korát éltük, most viszont a gazdasági hanyatlásában tengődünk. Múlt évben panaszkodtunk, milyen nehéz kiválasztani a számos jelentkező közül a győztest, s felvetettük, kevesebb versenytárs esetén nyilván könnyebb lenne a döntés. Most már tudjuk, hogy ez nem feltétlenül igaz.

Noha az egyes tárgykörökben kevesebb gyártó (főleg gépet, alkatrészt termelő) indult, a kínálat változatlanul bőséges (mind a szabad, mind a kereskedelmi termékek terén), és jelentős minőségjavulást értek el az elmúlt év során, mind a termékek, mind a Linux-rendszermag fejlesztése terén.

A *Linux Journal szerkesztői díjaira* nyílt forrású és védett programok egyaránt pályázhattak, s a programok idei kiválasztottjai között mindkét típus képviselői egyformán szerepelnek. Bár jelenleg mindannyiunkat nagy elégedettséggel tölt el, hogy olyan sok nyílt forrású termék közül válogathattunk, ne feledkezzünk meg róla, hogy nem sokáig marad ez így, amennyiben nem védekezünk minden erőnkkel az SSSCA-féle törvényi szabályozás veszélyei ellen. Ez ugyanis a Digital Rights Management (amolyan digitális jogvédelem) kötelező alkalmazását vezetné be, amelynek eredményeként a szabad, sőt a nyílt forrású operációs rendszerek is törvénybe ütközők lennének.

Amennyiben valamelyik nyertesünk a végsőkig felbosszantana titeket, nézzétek meg előző számunk 20. oldalán az olvasói szavazás eredményeit is.

Kiszolgáló

Filanet Interjak 200 802.11b

Kellene egy 802.11b hálózat nagyteljesítményű antennával az olcsó WAN-kapcsolatok létesítésére? Szeretnétek egy VPN-szolgáltatást nyújtó 802.11b alapállomást, amely a felhasználók noteszgépeinek biztonságáról gondoskodik? Központilag kezelt postarendszerrel, Samba- és VPN-kiszolgálókkal kellene ellátni a vállalat minden otthon dolgozó munkatársát? A Filanet egy sor kis költségű, ventilátor nélküli, beágyazott Linuxszal ellátott hálózati eszközt gyárt, amelyeket saját ASIC alaplapra épít ARM processzormaggal és 3DES-titkosítást támogató alkatrészekkel. Segítségükkel a vállalatok és ISP-szervezetek számos gondja megoldható, s alig kerülnek többe, mint egy ostoba DSL-doboz.

Biztonsági eszköz

Nmap

Egy program elterjedésének biztos jele, ha a nevét igeként kezdik használni. Mára bevett adatbiztonsági gyakorlat, hogy minden új Linux-kiszolgáló felállításakor az Nmapet futtatják le, és rendszeresen ellenőrzik vele a hálózati változásokat. Nem véletlen, hogy az Nmap ugyanakkor terjedt el, amikor a különböző Linux-változatok megritkították az alapértelmezés szerint felkínált szolgáltatások listáját. Üdvözöljük az Nmapet mint a rendszergazdák és Linux-változatok könnyen kezelhető „biztonsági jelzőfényét"!

Webkiszolgáló

APPRO1124

Mi e rendszer kétprocesszoros Athlon MP alaplapját tettük a linuxos csúcsgépünkbe. Az APPRO azonban – a VA Linux Systems eredeti tervei alapján – egy egyszerű IU-kiszolgálóba helyezte négy működés közben cserélhető SCSI-meghajtó és egy vékony CD-ROM-meghajtó társaságában. A nagy teljesítményű ventilátoroknak és az egyedi tápegységnek köszönhetően olyan webkiszolgálót alkottak, amit annak idején nagyon szívesen láttunk volna az alkatrészpiacon, amikor a webkiszolgálókra még külön keretünk volt.

Webügyfél

Konqueror

Linuxosok, itt az ideje eldobni a Netscape 4.x böngészőt, mert mára már ósdi kacatnak számít! Mostanra mind a Mozilla, mind a Konqueror elérte a megfelelő megbízhatósági szintet, és eléggé széles körű szolgáltatásokat kínál ahhoz, hogy örökre megszabadulhassunk az avítt Motif-alapú Netscape-től. A szerkesztői díjat végül a Konquerornak ítéltük a KDE-környezetbe való tökéletes illeszkedés elismeréséül, a sebességéért, és amiért lehetővé teszi a Flash-animációk vagy -mozik egyszerű lejátszását.

3D-s modellezőeszköz

Maya 4

Ahogyan *Robin Rowe* a legutóbbi írásában már beszámolt róla (*Linuxvilág* 6–7. szám, 44. oldal), a Linux szó szerint meghódítja a filmipart: segíti a különleges képi hatások és animációk gyors előállítását. Egyetlen más iparágban sem tapasztalható ilyen tömeges áttérés Linuxra. A Maya ugyancsak kiveszi ebből saját részét, amikor ügyfelei igényének megfelelően termékét Linuxra ülteti. Ezzel *Linus* elismerését is elnyerték, aki szerint ez „minden idők legösszetettebb és leglenyűgözőbb Linuxon futó 3D-s grafikus alkalmazása".

Biztonságimentés-készítő eszköz

BRU-Pro

Már azt hittük, hogy a BRU örökre elveszett a vállalati útvesztőkben, de szerencsére *Tim Jones*, a BRU régi pártfogója nem hagyta elenyészni e jól bevált, régimódi mentési eszközt. Tim korábban a BRU eredeti gyártójának, az EST-nek volt a fejlesztési alelnöke, most pedig a Tolis Group márkanév alatt kínálja a BRU-t. Ezzel a segédeszközzel a biztonsági mentésterv és az ügyfél által használt szalagegységek egyszerűen állíthatók be. Az sem mellékes, hogy a BRU támogatja a ➲ http://www.linuxtapecert.org weboldalt, ahol a Linux alatt kipróbált és jóváhagyott szalagmeghajtók listája található.

Egyéb segédprogramok

Acronis OS Selector

Íme egy remek betöltés- és lemezterületkezelő, amelynek nagy előnye, hogy a fokozott adatvédelem érdekében a ReiserFS fájlrendszert is támogatja.

Hordozható eszköz

Compaq iPAQ

A Linux futtatására alkalmas, titkárprogrammal ellátott zsebgépek (PDA) két csoportra oszthatók: az egyik csak a legszükségesebb programok futtatására alkalmas, a másik kellően gyors és elegendő teret enged a kísérletezésnek. Az iPAQ az utóbbiak közé sorolható. Az ipari tervezés szép példája, eltekintve a majdnem szimmetrikus íróvesszőtől. Sok fejlesztőt megnyert magának, így a linuxos PDA-tulajdonosok számos alkalmazás és leírás közül válogathatnak. A tartozékokat tekintve, mint például a PCMCIA kártyabővítő-csomag és a hamarosan elérhető kamera, vagy a gyorsulásmérő további kellemes lehetőségeket tartogat számunkra. Az iPAQ tehát ígéretes felületnek tűnik a jövőbeni Linux-fejlesztések számára.

Könyvek

Linus Torvalds–Davis Diamond: Just for Fun: The Story of an Accidental Revolution

(A móka kedvéért: egy véletlen forradalom története)

Az Internet időszerűségét jelzi, hogy legbefolyásosabb személyiségei már 31 évesen megírták az első önéletrajzukat. Azért az elsőt, mert a móka nyilvánvalóan még csak most kezdődött. *Linus Torvalds* tehát véletlenül lett forradalmár, akárcsak író. Bizonyos értelemben véve éppen ez a lényeg. A könyvet nem „kiadták”, inkább csak úgy „jött” – ahogyan a cím is mondja: a móka kedvéért. Egyoldalú párbeszéd valamiről, amiről talán érdemes beszélni, s ha mégsem, a szerzőt az sem érdekli különösebben. A Linushoz hasonlók szemszögéből nézve ez a könyv is egy kezdeti állapotban levő „hack”, amit idővel tökéletesíteni és javítani lehet. Hasonló szemszögből nézve vehetjük programleírásnak vagy hibalistának is. Érdekes, hogy ennek az operációs rendszernek, amelyet leginkább a rendszermag alapítójáról ismernek, ennyire sok szerzője létezik. A könyvet elolvasva megérthetjük, hogy mi hozta össze ezeket az embereket, s hogy egy olyan hétköznapi dolog, mint egy operációs rendszer, miként lehet annyira szórakoztató.

Hálózati program

OpenSSH

A Linux Journal szerkesztőségi iroda egyik kiszolgálóján huszonegynéhány OpenSSH folyamat fut egyszerre, egy munkaállomáson pedig hat. Adatátviteli csatornákat nyitunk, nyakra-főre használjuk az scp-t, és alapjában véve az SSH-kapcsolatok bűvöletében élünk. Kényelmes, megbízható – a beállítása és a kezelése pedig kész öröm. Igazából azért esett a választásunk az OpenSSH-ra, mert ha nem lenne, mindannyian Seattle-be költözhetnénk.

Fejlesztőeszköz

KDevelop

Az egyre több felhasználót megnyerő KDevelop nemcsak hibakereső és elsőrangú böngészőeszközöket kínál, hanem a GNU-tól megszokott módon az új fejlesztések indítását is megkönnyíti. A védett IDE-alkalmazások világából érkezők örömmel tapasztalhatják, hogy a KDevelop számos népszerű kezelőfelület utánzására képes. A beágyazott változatokat gyártó REDSonic szintén a KDevelopot választotta a REDICE Linux egyesített fejlesztőkörnyezetéül.

Irodai alkalmazás

Abi Word

Nem akármilyen szövegszerkesztő: valamirevaló rendszeren három másodperc alatt elindul és 5 MB memóriát igényel egy üres dokumentumhoz. Nem tévedés: megbízható, alapszolgáltatást nyújtó szövegszerkesztő minden fölösleges sallang nélkül. Már nyomtatni is tud, sőt a Microsoft Word-dokumentumokat is képes behozni. Ha kipróbáltad, két eset lehetséges: 1. megtetszik; 2. nem vesztettél sok időt a letöltésével.

Asztali környezet

KDE2

Az új KDE asztali környezetnek ugyan még fejlődnie kell, ami az erőforrás-felhasználást és az üzembiztonságot illeti, de minden megjelenő változat azt ígéri, hogy a legjobb úton halad a tökéletesség felé. Jobb lett a felépítése és komoly fejlesztésen ment keresztül. Az egyik legkellemesebb módosítás a KDE böngészőjének, a Konquerornak fájlkezelőként történő beépítése. A Google bevonásával keresést indít a címsorba beírt szavakra. A KDevelop szintén teljesen be van ágyazva, lásd a Fejlesztőeszköz tárgykört.

Valós idejű eszköz

Preemptible Kernel Patch,
Nigel Gamble és társai
– Montavista Software

Ez a mindössze ezersoros javítófájl a meglévő rendszermag SMP darabolási stratégiáját kihasználva figyelemreméltó eredményt ér el. Nemcsak a beágyazott rendszerek megszállottjai értékelhetik ezt, hanem bárki, aki egy terjedelmes tar-fájl kicsomagolása közben szeretne a számítógépén zenét hallgatni.

Kapcsolódó címek

APPRO 1124 ➲ appro.com/1124.html
BRU-Pro ➲ www.estinc.com/bruproinfo2.php
Filanet InterJak 200 802.11b ➲ www.filanet.com/index.php3?side=home&home=products_80211b
Konqueror www.konqueror.org
Linux Professional Institute ➲ www.lpi.org
Tribes 2 ➲ www.lokigames.com/products/tribes2
Velcro ➲ www.velcro.com
SuSE Linux 7.3 ➲ www.suse.com/us/products/suse_linux/i386/index.html

Honlapok

LinuxDevices

Miután az LLC nemrég megvette a DeviceForge-ot, a LinuxDevices visszatért alapítója, *Rick Lehrbaum* kezébe. E weboldal rendkívül sokrétű: hírekkel, útmutatókkal, termékismertetőkkel és összehasonlításokkal, valamint vitafórumokkal szolgál. Bár elsősorban beágyazott Linuxszal foglalkozik, az átlagos Linux-felhasználóknak is rengeteg ötletet kínál.

Adatbázis

Oracle

A Linuxon 1999-ben megjelent Oracle mára erős versenytárssá vált. Tavaly a PostgreSQL kapta a szerkesztők díját, s noha változatlanul komoly vetélytárs és nagy nyilvánosságot kapott az idei év folyamán, az Oracle lehengerlő teljesítményét egyszerűen nem lehet figyelmen kívül hagyni.

A lejáratás és a felhasználói szerződések

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.

Néhány, kereskedelmi programokat forgalmazó cég állítása szerint a szabad programban és a GPL-ben van valami, ami idegen az amerikai értékektől. Eme „aggódók kórusában" a Microsoft hangja a leghangosabbak egyike. A cég a honlapján így fogalmaz:
„A GNU General Public License (GPL) … jelentős fenyegetés azon szellemi tulajdonra alapozó cégek számára, amelyek a GPL hatálya alá tartozó programok köré próbálják meg üzletüket felépíteni. Még az olyan vállalkozások is, amelyek azt hiszik, hogy »pusztán felhasználói« a GPL-programoknak, veszélyeztetettek, mivel a GPL-es kódot kombinálják különállónak vélt alkalmazásokkal. Ez a szerződési modell előre meghatározza, hogy egy cég mely szellemi termékeit fogja megosztani a közösséggel és milyen feltételek mellett."
➲ http://www.microsoft.com/business/licensing/ssfaq.asp
Már másutt is bemutattam (lásd
➲ http://www.rosenlaw.com/html/GPL.PDF), hogy ez az állítólagos fenyegetés légből kapott. Ám az ellentmondás még mélyebb: a Microsoft saját közösségi (*shared-source*) szerződése sokkal veszélyesebb a programfejlesztők közössége számára. Olyan trójai típusú szerződés, amely – ha nem vigyázunk – tönkreteheti nyílt forrású vagy kereskedelmi fejlesztéseinket.
A Microsoft közösségi szerződésének legegyszerűbb változata az, amelyet a Windows CE 3.0 forráskódjával terjesztenek. Ennek a szerződésnek a második bekezdésében áll: „Ez a program felhasználható bármilyen nem üzletszerű tevékenység céljából, beleértve az ennek alapján készült változatok terjesztését is." Ezután a szerződés egyértelműsíti, hogy üzletmenetünk vezetése „nem minősül nem üzletszerű tevékenységnek".
Az üzleti felhasználóknak – úgy gondolom, a legtöbb nyílt forrású program fejlesztője ebbe a kategóriába tartozik – a szerződést a rájuk vonatkozó felhasználási korlátozások meghatározásához tovább kell vizsgálniuk.
A rossz hír a szerződés harmadik bekezdésében olvasható: „Üzletszerű tevékenység során e program kizárólag a Windows CE-felülethez készített további programok és eszközök fejlesztéséhez és ellenőrzéséhez használható. Ez a program sem forrás-, sem bináris kód formájában nem terjeszthető üzleti céllal".
Figyeljük meg, hogy a Microsoft nem engedélyezi a kód lemásolását vagy beillesztését a saját programunkba. Üzleti céllal a kódot csak referenciaként használhatjuk. Természetesen tilos a Microsoft-kód bármely részét lemásolnunk vagy a saját programunk kialakításához felhasználnunk. Mi történik azonban abban az esetben, ha később önállóan, anélkül, hogy tudatosan felidéznénk, amit a Microsoft kódjában láttunk, olyasvalamit hozunk létre, ami lényegi hasonlóságot mutat vele? Felelősségre vonhatnak-e a szerzői jog megsértéséért?
Itt jön a képbe a szerződés trójai faló jellege, ugyanis a bíróságok korábban már egyértelművé tették: „annak a bizonyítása, hogy a mi alkotásunk és egy másik között lényegi hasonlóság áll fenn, valamint az, hogy a másik alkotás hozzáférhető volt, elegendő lehet a szerzői jog megsértésének bizonyítására, még akkor is, ha a másolás nem tudatosodott bennünk".
Vajon mennyire könnyű elfelejteni valami fontosat, amit az ember olvas? Egy szerzői jogi per a 70-es évekből segít megvilágítani a kérdést. *George Harrison* cégét beperelték a szerzői jogok megsértéséért. Egy zenei kiadó azt állította, hogy Harrison „My Sweet Lord" című népszerű dala egy korábbi sláger, a „He's So Fine" másolata. A szerzői jog megsértésének bizonyításához a „He's So Fine" kiadójának nem csak azt kellett bizonyítania, hogy a két dal „meglepően hasonló", hanem azt is, hogy Harrison az eredeti dalt másolta a „My Sweet Lord" írásakor.
Harrison nem tagadta, hogy ismerte az eredeti számot, azonban azt állította, hogy a „My Sweet Lord" írásakor nem volt tudatában, hogy a „He's So Fine" dallamát használja. A bíróság döntése szerint:
„Amikor zenei anyagokat keresett gondolatai kifejezéséhez ... [Harrison] elméjében előkerült egy bizonyos kombináció, amellyel elégedett volt..." Vajon Harrison szándékosan másolta le a „He's So Fine" zenéjét? Nem gondolom, hogy szándékosan tette volna. Mindazonáltal világos, hogy a „My Sweet Lord" ugyanaz a szám, mint a „He's So Fine", csak más szöveggel, és a „He's So Fine" Harrison számára hozzáférhető volt. Ez a törvény értelmében kimeríti a szerzői jog megsértését, és ezen az sem változtat, hogy nem tudatosan történt.
[Bright Tunes Music Corp. kontra Harrisongs Music, Ltd., 420 F.Supp. 177 (S.D.N.Y. 1976.)]
Ezekután Harrison cégét hozzávetőleg 1,6 millió dollár kártérítés megfizetésére kötelezték.
Bárki, aki jártas a számítógép-programozás művészetében, tudja, hogy éppen úgy, mint a zenében, bizonyos gondolatok kifejezésére meglehetősen szokványossá vált formák állnak rendelkezésre. Amennyiben egyszer már látta a Microsoft kódját, vajon képes-e egy szakavatott programozó egyszerűen kitörölni az emlékezetéből? Mégha meg is próbálja tudatosan elfelejteni, és nem törekszik a másolására, kifejeződhetnek-e a nem tudatos emlékei az általa létrehozott programkódban olyan mértékben, hogy egy bíróság a hasonlóságot elegendőnek találja a szerzői jog megsértésének megállapításához?
Arra intenék minden nyílt forrású kódot fejlesztő programozót, hogy kerülje ezt a kockázatot. Aki nem tartozik azok közé a különleges emberek közé, akik képesek ellenőrizni a tudatalattijukat, az inkább meg se nézze a Microsoft közösségi forráskód-szerződés hatálya alá eső kódot.

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).

Új termékek

LinuxCAD 3.0

Piacra dobták a LinuxCAD 3.0-s változatát, amely már támogatja a háromdimenziós rajzok készítését. Az összes gyakran használt 2D és 3D Acad-rajzolóparancs a LinuxCAD-ben ugyanúgy működik, mint az Acadben. A LinuxCAD az X környezetbe épül be, ami lehetővé teszi, hogy a felhasználók ugyanazt a fájlt több ablakban és több képernyőn szerkesszék, a rajzok részeit másolhassák, és egyszerre tíznél is több rajzzal dolgozhassanak ugyanazon a számítógépen. A LinuxCAD támogatja a .DXF, .DWG, .DXS, .SLD és .SHX rajzformátumokat, és Intel-alapú, Solaris- és LinuxPPC-rendszereken fut.
Adatok: Software Forge, Inc., telefon: 913-663-1724, e-mail: sales@softwareforge.com,
➲ http://www.linuxcad.com

VXA AutoRak

Az Ecrix Corporation VXA Autorak nevű készüléke állványba szerelhető szalagtároló és -betöltő, amely akár 660 GB tömörített adatot is képes tárolni, és adatátviteli sebessége elérheti a 21,6 GB/óra értéket. Az AutoRak legfeljebb tíz adatkazettát tud használni, alakja szabványos 2U formájú, ezért könnyen beszerelhető a 19 hüvelykes állványokba. Az adatok mentése és helyreállítása az AutoRak vezérlőpultján keresztül állítható be és követhető figyelemmel. A ki- és bemeneti kapu biztonsági megfontolásokból lezárható. Intelligens vonalkódolvasók is beszerezhetők hozzá.
Adatok: Ecrix Corporation, 5525 Central Avenue, Boulder, Colorado 80301, telefon: 303-402-9262, e-mail: info@ecrix.com,
➲ http://www.ecrix.com

Plesk Control Panel

A Rackspace Managed Hosting bejelentette a Plesk Server Administrator (PSA) 2.0-t. Több felületen futó webalapú program, amellyel sokféle rendszerfelügyeleti feladat is ellátható, a Rackspace Linux- és Unix-kiszolgálóhoz már egyaránt elérhető. A Plesk egérrel vezérelhető felületén a felhasználók postafiókokat hozhatnak létre és elvégezhetik a tartományok karbantartását. A PSA a szolgáltató vállalatok számára lehetővé teszi, hogy a kiszolgáló-karbantartási feladatokat ügyfeleikkel megosszák – kihasználva a webes felügyeleti felület három szintjét: Admin, Reseller Client és Domain Owner.
Adatok: Rackspace Managed Hosting, Inc., 112 East Pecan, Suite 600, San Antonio, Texas 78205, telefon: 1-800-961-288,
➲ http://www.rackspace.com

Niveus 205

A Niveus 205 Pinguin Computing Intel-alapú munkaállomása, amelyet

háromdimenziós grafikai és alkalmazásfejlesztési feladatokra terveztek. A Niveus belsejében két legfeljebb 1,26 GHz órajelű Pentium III processzort, 133 MHz sebességű alaplapot, ATA-100 merevlemezt, legfeljebb 1,5 GB PC133 RAM-ot, öt PCI-csatlakozót, egy 4× AGP-csatlakozót, három 5,25" meghajtóhelyet, 52-szeres CD-ROM-ot és 3,5" lemezmeghajtót találunk. A Niveus munkaállomás előre telepített RedHat operációs rendszerrel kerül forgalomba. Sokféle kiegészítő kapható hozzá, például Klipsh ProMedia hangszórók, LCD képernyők és csúcsminőségű grafikus kártyák, beleértve a GeForce 3-at is.
Adatok: Penguin Computing, Inc. 965 Mission Street, Suite 600, San Francisco, California 94103, telefon: 1-888-736-4846, e-mail: info@penguincomputing.com,
➲ http://www.penguincomputing.com

e/pop Server

A WiredRed Software Corp. kis- és nagyvállalatok számára kiadta az azonnali üzenetváltást (instant messaging = IM) és a valós idejű kapcsolattartást lehetővé tevő programját, az e/pop Servert. Az e/pop Linux Server az e/pop Standard Server Edition részeként a központilag felügyelhető, méretezhető és biztonságos üzleti kapcsolattartást, az üzenetek tárolását és a hálózati, valamint az internetes útválasztást teszi lehetővé. A távkommunikáció behívásos módszerrel, VPN-en vagy internetkapcsolaton keresztül jön létre. A biztonságról a beépített 512 bites RSA-titkosítás gondoskodik, az AES-, DES-, Triple DES- vagy az RC4-módszer használható. Az e/pop segítségével szövegalapú csevegő- és VoIP-konferencia is tartható, valamint az alkalmazások is megoszthatók.
Adatok: WiredRed Software Corporation, 4669 Murphy Canyon Road, Suite 108, San Diego, California 92123, telefon: 858-715-0970,
➲ http://www.wiredred.com

IEMS6

Az International Messaging Associates (IMA) bejelentette az Internet Exchange Messaging Server (IEMS) 6.0-s változatának megjelenését. Az IEMS6 magja az üzenetkezelő alkalmazás-keretrendszer, amely olyan alkalmazásfejlesztő környezet, amelyben a rendszergazdák egyszerű üzenetkezelésre felkészített alkalmazásokat készíthetnek, és az elektronikus levelezőrendszerrel, GSM-mel, SMS-sel és az Internettel köthetik össze őket. A felhasználók az alkalmazásokhoz vállalati és kishálózati-környezetben is hozzáférhetnek – otthoni hálózaton, SMS-ezésre képes mobiltelefonon, illetve tetszőleges webböngészésre képes készüléken keresztül. Az IEMS6 olyan naptár- és határidőnapló-lehetőségeket tartalmaz, amelyek támogatják a Linux-, Solaris-, HP-UX- és az Outlook-felületet. Az IEMS6-nak része még az SMTP-hez való SSL továbbfejlesztett támogatása és az SMTP-hitelesítés támogatása, a csatolt mellékleteket eltávolító szűrő, valamint az üzenet titkosított tárolását biztosító modul.
Adatok: International Messaging Associates, Ltd., 27/F China Resources Building, 26 Harbour Road, Wan Chai, Hong Kong, e-mail: sales@ima.com,
➲ http://www.ima.com

A hónap szakmai tanácsai

Létezik PAM a Slackwaren?

Slackware-kiszolgálómon egyre több alkalmazást telepítek a felhasználóim számára. Azt tapasztaltam, hogy számos alkalmazás használatához hitelesítés szükséges, azonban akad néhány, amelyik az adatokat nem a *passwd* fájlból veszi, ennek következtében a felhasználóknak több helyen is meg kell változtatniuk a jelszavukat.

Úgy tűnik, hogy a világ a PAM és az LDAP használata felé halad, ezért ha átállhatnék a támogatásukra, a felhasználók egy helyről (például egy webalapú jelszóváltoztató alkalmazáson keresztül) meg tudnák változtatni az összes szolgáltatáshoz tartozó jelszavukat, beleértve a Samba, a levelezés, a pppd és phpgroupware szolgáltatásokat. A Slackware sajnos nem támogatja a PAM-ot, és nem találtam olyan leírást, amely a PAM telepítését tárgyalná.

Brian Johnson, bjohnson@jecinc.on.ca

A PAM-ot olyan terjesztésekre is telepíteni lehet, amelyek nem támogatják, ez azonban más egyebek mellett azt is magával vonja, hogy minden hitelesítést megkövetelő alkalmazást le kell cserélned a PAM-ot használó változatra (ha a terjesztésedben ezek nem érhetők el, meg kell szerezned a forráskódot, meg kell keresned a PAM-foltokat – amennyiben nem részei a programnak –, végül a rendszered beállításainak megfelelően mindent újra kell fordítanod.)

Ez rengeteg munkával jár, és amennyiben nem kifejezetten keresed a különleges kihívásokat, javaslom, térj át valamelyik korszerűbb terjesztésre, például a Debianra vagy a RedHat Linuxra (mindkettő alapból támogatja a PAM-ot). A fenti két változatot csupán példaként említettem, a PAM-ot számos más terjesztés is támogatja.

Marc Merlin, marc_bts@valinux.com

Nincs elég hely a telepítéshez

A Slackware állandóan azt írja ki, hogy nincs elegendő helyem a telepítés folytatásához. Ez hihetetlen, hiszen 10 GB helyet foglaltam le e célra. Merevlemezem felosztása a következőképpen fest:

5 GB – WinNT 4.0
512 MB – /root
512 MB – csereterület
4 GB – /usr
4 GB – /home

Cheppy, banggae@fisika.ui.ac.id

A lemezrészek formázása és befűzési pontjaik megadása után válts át a második virtuális konzolra (Alt+F2), és a `df` vagy a `mount` használatával ellenőrizd, hogy a lemezrészek be vannak-e fűzve. Amennyiben nem, az egész Slackware az 512 megabájtos saját lemezrészre települ. Ez a méret túl kevésnek bizonyulhat, ha az X-et vagy más nagyméretű alkalmazást telepítesz.

Chad Robinson, crobinson@rfgonline.com

Lefogadom, hogy rosszul címkézted fel a lemezrészeket, és a / helyett `/root` címkét használtál. A lemezrészeket így próbáld címkézni:

5 GB – WinNT 4.0
512 MB – /
512 MB – csereterület
4 GB – /usr
4 GB – /home

Ez elég helyet biztosít a telepítéshez.

Paul Christensen, pchristensen@penguincomputing.com

A frissítés óta hetente fagy a gépem

Miután az egyik RedHat-rendszerünket 7.0-sról 7.1-es változatra frissítettem, a gép körülbelül hetente egyszer lefagy. A fagyás mindig hajnali négy óra után nem sokkal következik be (a cron.daily végrehajtása után).

A rendszermag kimenete:

```
unable to handle kernel NULL pointer
➥dereference at virtual address
➥00000000
```

A rendszermagot a 2.4.3-12-es változatra frissítettem, de a helyzet nem sokat javult.

Atsuko Crum, acrum@hood.edu

Egy másik terjesztéssel – de szintén a 2.4.x rendszermagváltozattal – nekem is hasonló gondom akadt. Végül is az alaplapi BIOS frissítése a gondok nagy részét megoldotta – bár néha még előfordul fagyás, mindazonáltal sokkal ritkábban.

David Brown, david@caldera.com

A számítógép alkatrészei bármikor meghibásodhatnak, ha a fagyások azonban a frissítés után kezdődtek, a rendszert érdemes azzal a rendszermaggal kipróbálni, amelyet a 7.0-s változattal használtál. Amennyiben nem riadsz vissza a rendszermag újrafordításától, tégy egy próbát a legfrissebb 2.4 rendszermaggal. A 2.4 sorozat első változataiban számos hibát kijavítottak. Amennyiben a számítógép meghibásodásának lehetőségét ki szeretnéd zárni, kipróbálhatod a Cerberust, amely erős terhelés alatt ellenőrzi a gép alkatrészeit. A Cerberus a SourceForge-ról a

➲ http://sorceforge.net/projects/va-ctcs címről tölthető le.

Marc Merlin, marc_bts@valinux.com

SCSI-utánzás csak egy meghajtóra

Egy HP IDE CD-íróval rendelkezem, ezért SCSI-utánzást kell használnom, hogy működjön a cdrecord programmal. A 2.2.18 rendszermag alatt meg tudtam mondani az `ide-scsi` modulnak, hogy csak az íróval foglalkozzon, és hagyja békén az ATAPI CD-ROM-meghajtómat. Ezt a *lilo.conf*-ba írt `append` sorral értem el:

```
append="hdc=ide-scsi"
```

Remekül működött, mivel a */dev/hdc* az író és a */dev/hdd* az ATAPI CD-ROM. Ez a 2.4-es rendszermaggal sajnos nem működik többé. Az `ide-scsi` modul mindkét eszközt megragadja, ennek következtében a */dev/hdd* elérhetetlenné válik, és a cdparanoia nem tud dolgozni vele, engem meg arra kényszerít, hogy a */dev/scd1*

használatával fűzzem be. Hogyan érhetném el, hogy az `ide-scsi` modul a 2.4 rendszermag alatt is csak a */dev/hdc*-t használja?
Michael Soulier, michael.soulier@home.com

Ha jól értem, azt szeretnéd, hogy a `hdc` SCSI-utánzást használjon, míg a `hdd` továbbra is IDE-eszköz maradjon. Általában az IDE CD-támogatása tiltott, a SCSI-utánzás pedig engedélyezett, ezért látszik mindkettő SCSI-eszközként. Olvasd el a
➲ http://www.wizball.co.uk/linux/cd_rewriter.php és a
➲ http://wwwteknospy.com/pages/howtos/cdburn.php
oldalakon található leírásokat.
Paul Christensen, pchristensen@penguincomputing.com

Merevlemez beépítése

Beépítettem egy második merevlemezt a gépembe, fel is osztottam lemezrészekre, de nem tudok rajta fájlrendszert létrehozni.
Kevin Williams, williams_kevin@btconnect.com

Először is győződj meg róla, hogy melyik eszközről van szó. Az alábbi egyszerű lista az IDE-felületű egységeket foglalja össze, talán segít az eligazodásban:

/dev/hda1	az elsődleges csatolón lévő mester (primary master) első lemezrésze
/dev/hda2	az elsődleges mester második lemezrésze
/dev/hdb1	az elsődleges szolga (slave) első lemezrésze
/dev/hdb2	az elsődleges szolga második lemezrésze
/dev/hdc1	a másodlagos csatolón lévő mester (secondary master) első lemezrésze
/dev/hdc2	a másodlagos mester második lemezrésze
/dev/hdd1	a másodlagos szolga első lemezrésze
/dev/hdd2	a másodlagos szolga második lemezrésze

Most válaszd ki a használni kívánt fájlrendszert. A SuSE-terjesztésben a ReiserFS található – én ezt javaslom, mivel használata esetén egy hibás rendszerleállás utáni újraindításkor nem kell kivárnunk a hosszadalmas ellenőrzést. Ezt követően formázd meg a lemezt. A ReiserFS lemezrész létrehozásához szükséges parancs az `mkreiserfs <eszk znØv>`, ahol az `<eszk znØv>` a lemezrész neve. Ha ragaszkodsz az ext2-höz, ugyanezt a parancsot használd, de az `mkreiserfs`-t cseréld le `mke2fs`-re. Létrejött tehát egy használható lemezrész, melyet már csak be kell fűznöd. Válassz vagy hozz létre egy befűzési pontot. Ebben a példában én a */mnt/storage* könyvtárat használom. Hozd létre ezt az `mkdir /mnt/storage` paranccsal. Amint láthatod, a befűzési pont igazából egy könyvtár. Most fűzd be a meghajtót:

```
mount <eszk znØv> /mnt/storage -t
➥<fÆjlrendszer_t pusa>
```

Itt az `<eszk znØv>` a lemezrész által használt eszköznév, a `<fÆjlrendszer_t pusa>` pedig `reiserfs` vagy `ext2`.

Most már egy második használható linuxos merevlemezed is van. Csak egy lépés maradt hátra. Feltételezem, hogy az új lemezt minden rendszerindítás után használni akarod, tehát az új lemezrészt a */etc/fstab* táblázatba is be kell vezetni. Írj ehhez a fájlhoz egy sort:

```
<eszk znØv>              <befßzØsi pont>
➥<fÆjlrendszer_t pusa> defaults 0 0
```

Az `<eszk znØv>` itt is a lemezrész eszközneve, a többi szintén egyértelmű. Ha a ReiserFS-t használod, a sor végén `0 0`, ha az ext2-t, akkor pedig `1 2` szerepeljen.
Ben Ford, ben@kalifornia.com

128 bites pontosság GCC-vel

Az `xlC` parancsot használom C++ programok lefordítására Unix-felületen. Amennyiben 64 bitről 128 bitre kell növelnem a matematikai számítások pontosságát, a következő parancsot alkalmazom:

```
xlC128 -qldbl128 <fÆjlnØv> [-lm]
```

A `-lm` kapcsolóval fűzöm be a matematikai programkönyvtárakat (amennyiben hiányoznak). Ezeket a számításokat ugyanolyan pontossággal Linux alatt is el szeretném végezni. Mivel próbálkozzam? Ha a matematikai programkönyvtárakat is fel kell használnom, kérem, adjátok meg, honnan szerezhetem meg őket.
Pramod, l_pramod@hotmail.com

Szerezd be a GMP-t (Gnu Math Precision)! Ez a szabad forrású programkönyvtár tetszőleges pontosságú aritmetikát valósít meg előjeles egészekkel, racionális és lebegőpontos számokkal. A ➲ http://www.swox.com/gpm címről tölthető le.

Mac-lemezrészek befűzése

Van egy 2001-es iMacDV számítógépem (400 MHz, 128 MB RAM), melyen Mac OS 9.1 és a Yellow Dog 2.0 fut. A Mac OS 9.1 lemezrészen lévő fájlokat sajnos nem tudom elérni. Azt hiszem, hogy a Mac OS lemezrész a `hda1`, de lehet, hogy tévedek.
Bill MacKay, w.mackay1@ntlworld.com

Valóban tévedsz, nem a `hda1`-ről van szó. A Mac-gépek a lemezrészeket valamivel bonyolultabban tartják nyilván. A lemezrészeket a `cat /proc/partitions` paranccsal nézheted meg. Remélem, sikerült könnyebbé tenni a helyzetet.

Hogyan indítsam el önműködően a webkiszolgálót?

A HTTPD a gép indításakor nem indul el magától, így nekem kell megtennem a */etc/rc.d/init.d/* könyvtárból. Melyik beállításfájlt változtassam meg, hogy ez önműködően is megtörténjen?
W. Huang, whuang53@excite.com

Add ki a `chkconfig --level 5 httpd on` parancsot. Ez azt feltételezi, hogy az 5. futásszintet használod (GUI a RedHatben). Ha más futásszintet akarsz, csak írd át a számot.
Ben Ford, ben@kalifornia.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.

Vírusellenőrzés a Squid proxykiszolgálón

A rendszergazdák természetes igénye, hogy minél több vírusellenőrzési pontot iktassanak be a számítógépes hálózatokba. Az egyik ilyen pont a HTTP-proxykiszolgáló, ugyanis ezen haladnak keresztül azok az Internetről letöltött fájlok, amelyeket a felhasználók böngészői szednek le.

Amennyiben e ponton sikerül kiszűrni a vírusos állományokat, el sem jutnak a felhasználók gépeihez.
A Squid általánosan használt proxykiszolgáló, vele működik együtt a `viralator` program, amelynek segítségével elvégezhetjük a víruskeresést.
A `viralator` Perlben írt CGI-héjprogram, amely képes a bemeneti értékként megadott fájlokat letölteni a kiszolgálóra, és a letöltött állományokon egy külső vírusirtó programot futtat. Eközben a felhasználóval az ügyfélgépen futó internetböngészőn keresztül tartja a kapcsolatot, azaz tájékoztat a letöltés menetéről, és arról, hogy vírusos-e a fájl. Nézzük meg működés közben!
A böngésző ablakában a kívánt hivatkozásra kattintva kezdjük meg a letöltést. A böngésző a letöltési kérést elküldi a távoli kiszolgálóhoz – ezt a kérést kapja el a viralator program (*1. kép*). A böngészőnek visszaküld egy „downloading…” tartalmú oldalt, ezután nyit egy kék hátterű ablakot, ahol a letöltés menetét láthatjuk – az ablak alján a vírusellenőrzés eredményével. A kék ablakban egy ***Stop*** gomb segítségével a folyamatot megállíthatjuk. Ha a teljes állomány a kiszolgálóra került és nem volt vírusos, előugrik a böngésző letöltési ablaka és menthetjük a fájlt. Miután az állomány az ügyfélgépre is megérkezett, térjünk vissza a viralator kék ablakához, és nyomjuk meg a ***Close window*** gombot. Ezután egy ablak tájékoztat arról, hogy a program a kiszolgálóról letörli az állományt, majd el is tűnik.

A viralator működéséhez szükséges programok

Vírusirtó program: a viralatorba nincs vírusirtó beépítve, külső programot indít el. A múlt havi számban ismertettem a Sophos `sweep` telepítését, a viralator képes vele együttműködni.

`wget`: a böngésző által kért állományt a `wget` programmal tölti le. Mindenképpen telepítsük, nagyon hasznos program.

HTTP-proxykiszolgáló: esetünkben a Squid, azonban most sem a telepítésére, sem a beállítására nem térek ki (lásd még ***Linuxvilág*** 2001. február–márciusi számának 74. oldalát). Az ügyfelek böngészőinek az Internetet a Squiden keresztül kell elérniük.

Átirányító program: ez a Squidhez intézett letöltési kéréseket egy külső programnak küldi át. A viralatorhoz ajánlott átirányító a Squirm.

Webkiszolgáló: a `wget`-tel letöltött fájl a webkiszolgálón keresztül jut el az ügyfélgépre. Apache-kiszolgáló működését feltételezem, a telepítést pedig Debian Potato rendszerre írom le.

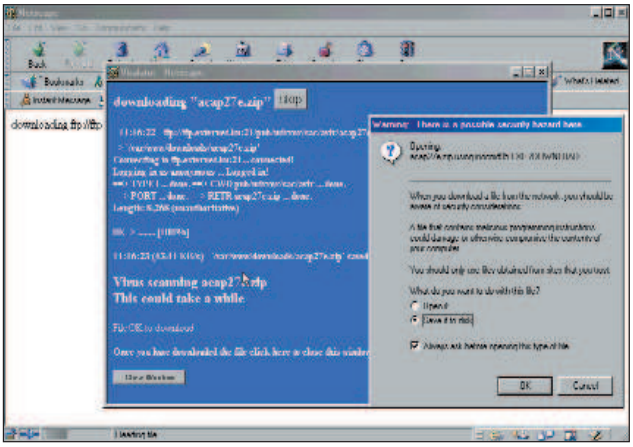
A Squirm telepítése

A Squirm honlapján a ➲ http://squirm.foote.com.au címen részletes telepítési leírást találunk, a forrást pedig a ➲ http://squirm.foote.com.au/squirm-1.0betaB.tar.gz címről tölthetjük le. Amennyiben ezt megtettük, csomagoljuk ki a */usr/src*-be, majd lépjünk be a ***squirm-1.0betaB*** könyvtárba. Adjuk ki a

```
cd regex
./configure
make clean
make
```

parancsot. A ***regex*** könyvtárban létrejövő két fájlt az eggyel feljebb lévő könyvtárba kell másolnunk:

```
cp -p regex.o regex.h ..
```



1. kép Letöltés Netscape-pel

Meg kell tudnunk, hogy a Squid milyen felhasználóként fut, amit a

```
grep cache_effective_user /etc/squid.conf
```

utasítással tehetünk meg. Debianon ez a `proxy` felhasználó, míg a Squirm a `squid` felhasználóra van beállítva. Ennek szellemében kell a Squirm Makefile-ját az `install` résznél módosítani:

```
install -m 755 -o root -g root
➥-d /usr/local/squirm \
/usr/local/squirm/bin
install -m 770 -o root -g proxy
➥-d /etc/squirm
install -m 750 -o proxy -g proxy
➥-d /var/log/squirm
install -m 660 -o root -g proxy
➥squirm.local.dist squirm.patterns.dist \
```

```
/etc/squirm
install -m 755 -o root -g root --strip squirm
➥/usr/local/squirm/bin
```

A program a beállítóállományokat eredetileg a */usr/local/squirm/etc*, a naplófájlokat pedig a */usr/local/squirm/log* könyvtárba tette. Mivel erre nem találtam elégséges indokot

2. kép A Viralator weboldala

és zavaró is lehet, ezeket is átírtam, így a fenti mintában már */etc/squirm* és */var/log/squirm* szerepel.
Ezután a ***paths.h*** fájlt is módosítani kell ott, ahol az eredeti elérési útvonalak voltak:

```
#define LOG_MATCH "/var/log/squirm/squirm.match"
#define LOG_FAIL "/var/log/squirm/squirm.fail"
#define LOG_ERROR "/var/log/squirm/squirm.error"
#define LOG_WHERE "/var/log/squirm/squirm.where"
#define LOG_DEBUG "/var/log/squirm/squirm.debug"
#define LOG_INFO "/var/log/squirm/squirm.info"

/****** Configuration file locations *******/
#define LOCAL_ADDRESSES "/etc/squirm/squirm.local"
#define REDIRECT_PATTERNS "/etc/squirm/squirm.patterns"
```

Adjuk ki a

```
make
make install
```

parancsokat és próbáljuk ki, hogy az átirányító fut-e rendszergazdaként a rendszerünkön:

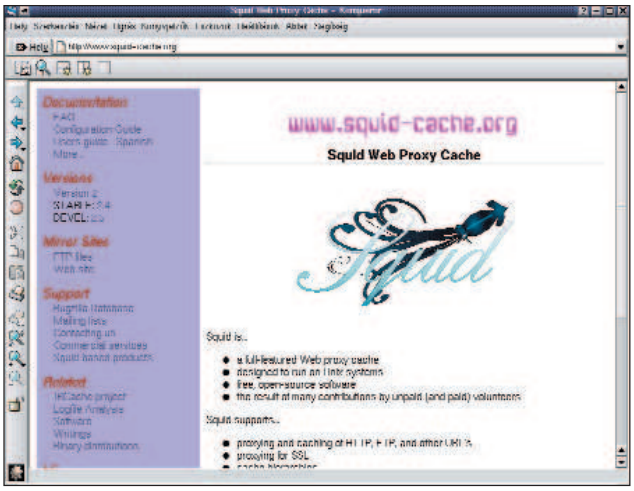
```
/usr/local/squirm/bin/squirm

Squirm running as UID 0: writing logs to stderr
Wed Nov 21 10:55:01 2001:unable to open local
➥addresses file [/etc/squirm/squirm.local]
Wed Nov 21 10:55:01 2001:unable to open
➥redirect patterns file
Wed Nov 21 10:55:01 2001:Invalid condition
➥- continuing in DODO mode
Wed Nov 21 10:55:01 2001:Squirm (PID 24924)
➥started
```

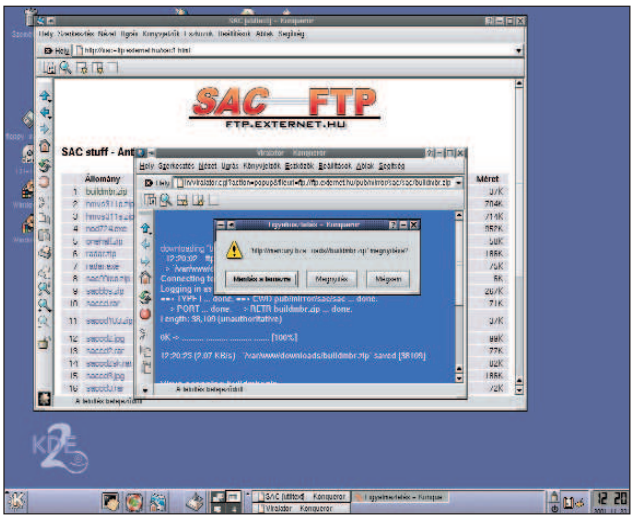
Mivel a Squirmnek a */etc/squirm* könyvtárban még nem készítettünk beállítóállományokat és csak dodo-módban indult el, CTRL+C-vel lépjünk ki.
A */etc/squid.conf* fájlban keressük meg a `redirect_program` részt és módosítsuk:

```
#redirect_program none

redirect_program /usr/local/squirm/bin/squirm
redirect_children 10
```



3. kép Hasznos programok tárháza



4. kép Letöltés Konquerorral

A Squidet indítsuk újra. A */var/log/squid/cache.log* állományban megjelenő

```
helperOpenServers: Starting 10 'squirm'
➥processes
```

bejegyzés tájékoztat arról, hogy a Squirm elindult.

A Squirm beállítása

A Squirm telepítésekor a */etc/squirm* könyvtárba két mintafájlt hoz létre: a ***squirm.local.dist***-t és a ***squirm.patterns.dist***-t. Másoljuk át őket ***squirm.local*** és ***squirm.patterns*** néven ugyanebbe a könyvtárba.

A ***squirm.local***-ba be kell azokat az ügyfeleket írni, akiknek a kéréseit át akarjuk irányítani. Az átirányítás szabályait soronként egyesével a ***squirm.patterns***-be kell írnunk. Próbáljuk ki a következőt:

```
regexi ^http://www\.playboy\.com/.
➥* http://www.disney.com
```

Amennyiben a kis- és nagybetűket nem akarjuk megkülönböztetni, a szabályt a `regexi`-vel vezessük be, egyébként a `regex`-et alkalmazzuk. A szabályok részeit szóköz választja el egymástól. Az első rész egy szabályos kifejezés, amelyet a Squirm a kért URL-re illeszt. Ha a szabály illeszkedett, a Squid a második részben leírt című állományt küldi el az ügyfél böngészőjéhez. Vigyázzunk, a szabályos kifejezésben a pontokat fontos fordított perjellel (backslash) védeni, ezzel szemben a szabály második részében a fordított perjelet ne használjuk a pont előtt! A ***local*** vagy a ***patterns*** állományok módosításai a

```
killall -HUP squirm
```

parancs kiadása után lépnek életbe. A böngészővel próbáljuk meg elérni a ***www.playboy.com*** gépet. Ha minden működik, a Disney oldala jelenik meg.

A viralator telepítése

A program főoldala a ➲ http://viralator.loddington.com/ címen található. Töltsük le a ***http://viralator.loddington.com/downloads/viralator-09pre2.zip*** állományt (ha valakinek rokonszenvesebb a suEXEC-es telepítés, a weboldalon közzétett útmutatót kövesse, bár én nem ezt választottam). Bontsuk ki (egyetlen állományt tartalmaz), és tegyük a webkiszolgálónk ***cgi-bin*** könyvtárába (ami Debianon a ***/usr/lib/cgi-bin***) ***viralator.cgi*** néven. A jogokat állítsuk be, például

```
chown root.www-data /usr/lib/cgi-bin/
➥viralator.cgi
chmod 750 /usr/lib/cgi-bin/viralator.cgi
```

A webkiszolgáló gyökerében ***downloads*** néven készítsünk könyvtárat, és olyan jogokkal ruházzuk fel, hogy a ***viralator.cgi*** képes legyen benne fájlokat elhelyezni, olvasni és törölni:

```
mkdir /var/www/downloads
chown root.www-data /var/www/downloads
chmod 770 /var/www/downloads
```

A viralator a kért állományokat a `wget` segítségével ide fogja letölteni, a vírusirtót szintén itt futtatja majd, és az ügyfelek böngészői is ebből a könyvtárból fognak letölteni. Továbbá a naplóállományát ***viralator.log*** néven ugyancsak ebben a könyvtárban hozza létre. A ***viralator.cgi*** állományt a kívánt nyelvnek és vírusirtónak megfelelően módosítsuk, például:

```
# A little housekeeping first

$default_lang = "en";
$antivirus="SOPHOS";
```

Ezután változtassuk meg az elérési útvonalakat:

```
$downloads = "/var/www/downloads";
$downloadsdir = "/downloads/";
$logfile = "$downloads/viralator.log";
$wget = "/usr/bin/";
$deleteaction = "deletefile";
#$deleteaction = "mantain";
```

A `$deletaction`-nel kezdődő sorok közül válasszuk a nekünk megfelelőt. A `"deletefile"`-nál a ***downloads*** könyvtárból törli az állományt, míg a `"mantain"`-nel nem.
Ezt követően a ***/etc/squirm/squirm.patterns*** fájlt át kell írnunk a *listán* látható módon.
Természetesen a példában lévő IP-cím és gépnév helyett saját gépünk adatait adjuk meg. Az `abortregexi` rész állítja meg az átirányítást, ha tehát a letöltést a Squidet futtató gépről végezzük, a viralator nem indul el.

```
abortregexi (^http://10.0.0.1/.*)
abortregexi (^^http://proxy.webhely.hu/.*)
regexi (^.*\.zip$) ^http://10.0.0.1/.* cgi-bin/viralator.cgi?url=|\1
regexi (^.*\.doc$) ^http://10.0.0.1/.* cgi-bin/viralator.cgi?url=|\1
regexi (^.*\.exe$) ^http://10.0.0.1/.* cgi-bin/viralator.cgi?url=|\1
```

A három `regexi`-s sor a viralatort csupán akkor indítja el, ha .ZIP, .DOC vagy .EXE állományt szeretnénk letölteni.
A `regexi`-s rész szabályos kifejezése azért található gömbölyű zárójelek között, hogy a ***viralator.cgi*** a letölteni kívánt URL-t átadott értékként kaphassa.
Indítsuk újra a Squirmet a `killall -HUP squirm` paranccsal. Néhány állomány letöltésével próbáljuk is ki. Ehhez keressük fel a ➲ http://viralator.loddington.com/ downloads/leicar.zip címet, mert az állomány próbavírust tartalmaz.

Hibák

Bár az alkotó a programot folyamatosan fejleszti, sajnos akadnak benne hibák. Netscape-pel, Konquerorral tökéletesen működik, de Opera vagy Internet Explorer alatt folyamatos letöltési ciklusba kerül. A javítás megjelenéséig a 432. sort tegyük megjegyzésbe:

```
#print "<META HTTP-EQUIV=\"refresh\"
➥CONTENT=\"5 \;URL\=$requestpage\">\n";
```

Ekkor a ***downloading ...*** ablakról kézzel kell visszaléptetnünk, de a végtelen ciklus elmarad. A ➲ http://viralator.loddington.com/-on olyan fórum is működik, ahol felhívják a figyelmet a hibákra, sőt a program használói megoldásokat is kínálnak rájuk. Az oldal GYK-t is tartalmaz, böngészését melegen ajánlom.

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

Kapcsolódó címek

➲ http://viralator.loddington.com
➲ http://www.squid-cache.org/related-software.html

Webmin

Jól használható eszköz mind a guruk, mind a Linuxszal éppen most ismerkedők számára – a modulrendszerű tervezés és a háttéradatbázis nélküli működés hatékony, könnyen alakítható felületet biztosít.

Hadd kezdjem a program bemutatását a készítők honlapjáról vett idézettel: „A Webmin webalapú felület unixos rendszergazdák számára". Néhány éve bukkantam rá a Webminre, amikor jómagam éppen a ***webmin.com*** névtartományt szerettem volna bejegyeztetni. Ragyogó irányítópultról álmodoztam, amelyről a kiszolgálóparkom működését irányíthattam volna, és nagyon felháborított, hogy valaki már megszerezte a kiötölt nevet. Ám nyomban megfeledkeztem a bosszúságomról, mihelyt megláttam a honlapon a letöltésre szánt programokat – innentől kezdve már rajongással böngésztem tovább.

A Webmin ama célkitűzése, hogy webalapú felületet kínáljon a feladatok megoldásához, nem egyedi. A Világhálón számos ilyen eszköz található, amelyek között mind nyílt forrású, mind kereskedelmi termék megtalálható.

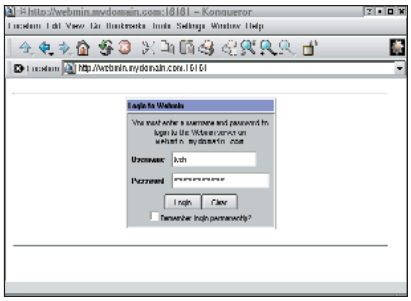
A Webmin azért tűnik ki a többi közül, mert tapasztalt és újdonsült rendszergazdák számára egyaránt célravezető választás lehet. Legelőször tekintsük át a telepítését, és azt is, hogy milyen feladatok megoldására alkalmas.

Az első lépések

Mielőtt elmélyednénk a Webmin szolgáltatásainak taglalásában, ejtek néhány szót a program kialakításáról. A Webmin alapvetően Perl nyelven íródott CGI-héjprogramok óriási gyűjteménye. Saját webkiszolgálóját azon a kapun működteti, amilyet a telepítés során megadunk neki, s ez teszi lehetővé a számunkra, hogy a Webmin biztonságát a tényleges webkiszolgálótól függetlenül kezeljük.

A támogatott felületek teljes listája a honlapon tekinthető meg, és ezek között az alább felsorolt rendszerek is fellelhetők: RedHat, Solaris, Debian, OpenBSD, HP-UX, IRIX, AIX, DEC, SCO és Mac OS X.

A Webmint a modulrendszerű tervezési séma teszi egyedülállóvá. Az összes szolgáltatás és program által biztosított lehetőség a modulok rendszerén keresztül valósul meg, ami a következőt jelenti: amennyiben a használni kívánt alkalmazást a Webmin nem támogatja, alkalmazói programfelületével (API) létrehozhatjuk a megfelelő modult, és ezzel már a kiválasztott program működésének irányítására is képes lesz. A Webminhez ötven szabványos modul tartozik, sőt, rajtuk kívül számos további modul is létezik, így hát temérdek forráskód segíti az első lépések megtételét.



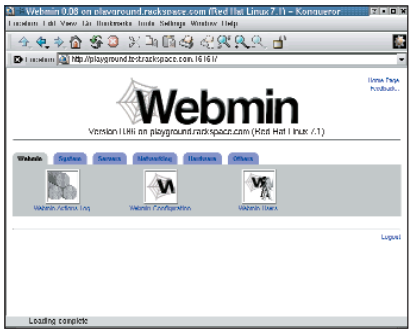
1. kép Bejelentkezés a Webminbe

Annak ellenére, hogy a Webmint magát a BSD-szerződés feltételeinek megfelelően terjesztik, a saját fejlesztésű modulok már tetszőlegesen választott szerződésfajta alapján terjeszthetők. Ez a megoldás a további programfejlesztés lehetőségét a Nyílt Forráskód Közössége és a kereskedelmi programok előállítói számára egyaránt nyitva hagyja.

A program telepítése enyhe fuvallathoz hasonlítható: elsőként látogassunk el a ➲ http://www.webmin.com/webmin címre, majd töltsük le az RPM-csomagot vagy a tar-állományt. Amennyiben az utóbbit választjuk, kicsomagolás után a `setup.sh` telepítő héjprogramot kell futtatnunk – ez fogja kezelni a telepítési folyamatot. A tar-t választók azonban győződjenek meg róla, hogy abban a könyvtárban végezték-e el a programok kicsomagolását, ahol azok működni is fognak! A Webmin kicsomagolása után ügyeljünk rá, hogy az éppen kicsomagolt ***webmin*** könyvtárat ne töröljük le, máskülönben a program nem fog működni. Amennyiben a Webmint a későbbiek folyamán törölni szeretnénk, a kezdeti könyvtárat a telepítő programcsomagban szereplő eltávolítóprogram fogja törölni az összes többi állománnyal együtt.

Miután befejeztük a Webmin telepítését, indítsunk egy böngészőprogramot, és adjuk meg az IP-címet- vagy egy feloldható kiszolgálónevet a hozzá tartozó kapucímmel – alapértelmezés szerint 10 000 – együtt (*1. kép*).

A képernyőn a bejelentkezést követően a Webmin üdvözlőoldala jelenik meg a kategóriafülekkel. A program moduljait a *2. képen* látható módon csoportosították, például a ***Webmin*** menüfülön található az általános jellemzőkre,



2. kép A Webmin felülete

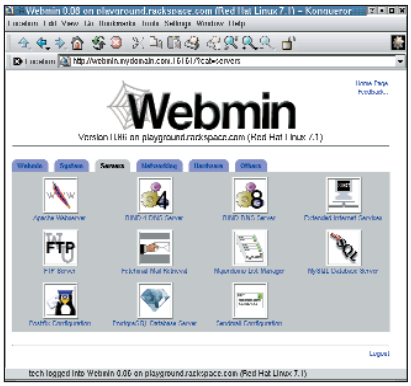
felhasználókra, modulokra stb. vonatkozó összes beállítási lehetőség – a program ezeken a helyeken az alapértelmezés szerinti számokkal fog működni.

A további füleket a ***System***, ***Servers***, ***Networking***, ***Hardware*** és ***Others*** címkékkel látták el.

A ***System*** kategória alatt rejtőző modulok olyan feladatok kezelését végzik, amelyek magának az asztali számítógépnek vagy a kiszolgálógépnek a működését érintik. E feladatok között szerepel többek között a lemezterület-felhasználás, az NFS- és NIS-jellemzők beállítása, a PAM-jellemzők módosítása, a rendszernapló megtekintése, új felhasználók felvétele, a cron működésének és a rendszerbetöltés idejére időzített szolgáltatásoknak a szabályozása, de a gép újraindítása is.

A ***Servers*** lap tartalmazza az összes

kiszolgálóbeállító modult, vagyis itt találhatjuk meg az Apache-, BIND-, DHCP-, Sendmail-, Squid- és még számtalan egyéb modult (*3. kép*).
A ***Networking*** az egyik újdonság, ami Linuxon a remek grafikus felületű IP Chains beállító modult tartalmazza, továbbá egy csokorravalót a hálózati segédprogramokból, így például a `ping`-et, a `traceroute`-ot, a `whois`-t és a `dig`-et.

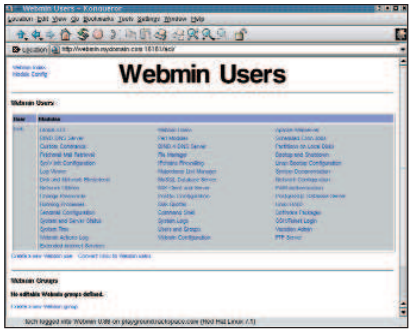


3. kép Webmin-modulok a Servers részben

A ***Hardware*** lapon természetesen közvetlenül a számítógép belső felépítésével kapcsolatos jellemzőkkel találkozhatunk. E helyen szerepelnek olyan adatok, mint a merevlemezek felosztása, a rendszeridő, a hálózati csatolókártya beállításai, a LILO és a RAID-kezelő programrendszer.
Az utolsó lapot joggal tekinthetjük a „mindent bele” kategóriának, hiszen teljes körű szolgáltatást nyújtó Java-alapú SSH/Telnet ügyfélprogram; Java-alapú állománykezelő; különleges, a felhasználó által meghatározott modul, amelyet ***Custom Commands***-nak neveznek; rendszernapló-olvasó program; Perl-modulok; valamint webalapú kiszolgáló- és rendszerállapot-figyelő modul található benne. A ***Custom Commands*** modullal egyszerű fogadófelületet készíthetünk a Webmin számára, amely tetszőleges parancs futtatását teszi lehetővé. Ez igen előnyös, amennyiben a Webmin csupán olyan célfeladat elvégzéséhez szükséges, amely önálló modulhasználatot nem igényel. A *4. kép* ilyen könnyen és gyorsan elkészíthető fogadófelületet mutat be.

A lépésenkénti útmutató

Az új, virtuális gépen futó Apache-kiszolgáló beállításához lépésenkénti útmutató áll rendelkezésünkre. Ez a leírás megmutatja, miképpen kell saját dokumentumot (főoldalt) szerkeszteni, továbbá hogyan kell a `cgi-bin`-t és a naplózást beállítani. A leírás a valóságosnál sokkalta bonyolultabbnak tűnik. Megállapítottam, hogy a „kattintós” grafikus felületen az irányok pontos meghatározása sokkal nagyobb erőfeszítést követel, mint az egyes objektumok kijelölése és az egérrel való kattintás.



4. kép Saját parancsok létrehozása

Jelenlegi példánkban a „dirk” felhasználói azonosító már előzetesen is létezett, de a ***test.com*** virtuális gépet és a / névtartományt csak most fogjuk létrehozni. A honlapfeltöltésre a Wu-FTPd-t használjuk, és feltételezzük, hogy a DNS már képes feloldani az új nevet. Először hozzunk létre HTML-állományokat, naplóállományokat és CGI-héjprogramokat tartalmazó könyvtárakat a webhely számára.
Látogassunk el ismét a Webmin főoldalára, és válasszuk az ***Others*** lehetőséget, ezt követően indítsuk el az ***Állománykezelőt***. Az Állománykezelő segítségével jussunk el a webhely rendszergazdájának saját könyvtárához. A felhasználó ugyanis ide kerül, amikor FTP-n keresztül kapcsolódik a kiszolgálóhoz. Így ennek a felhasználónak a saját könyvtára: ***/home/felh_nev***. Kattintsunk duplán a bal oldalon a ***/home*** könyvtáron, majd ismét duplán a rendszergazda nevét viselő könyvtáron. Elérkezett az ideje, hogy létrehozzuk a webhely számára szükséges könyvtárakat. Az ***New Folders*** ikonra történő kattintással – pontosabban az ***New*** felirattal és könyvtárikonnal ellátott gombon – a névtartomány számára létrehozzuk a főkönyvtárat, a ***/home/dirk/test.com***-ot. A könyvtárba való belépéshez az újonnan létrehozott könyvtáron, vagyis a ***test.com***-on kell duplán kattintani. Ezen belül három további könyvtár létrehozása szükséges: ***htdocs***, ***logs*** és ***cgi-bin*** névvel. Amennyiben a munkánkat jól végeztük, az alábbi könyvtárak jöttek létre:

```
/home/dirk/test.com,
/home/dirk/test.com/htdocs,
/home/dirk/test.com/logs,
/home/dirk/test.com/cgi-bin
```

Most térjünk vissza a felhasználó saját könyvtárához. Kattintsunk a jobb oldalon a webhelyhez tartozó könyvtáron (például a ***test.com***-on), majd a lap tetején levő ***Info*** gombon. Az ***Info*** ablakban a felhasználó- és csoportazonosítót állítsuk be a valóságnak megfelelően. Az ***Apply changes to*** szakaszban lévő lenyíló listából válasszuk a ***This directory and all subdirectories*** lehetőséget, majd a beállítást mentsük.
Lépjünk vissza a Webmin főoldalához a lap tetején található ***Return to index*** feliraton kattintva. Válasszuk a ***Servers*** fület, és az egérrel kattintsunk az Apache-kiszolgálón. Amennyiben az Apache-kiszolgálót első alkalommal indítottuk a Webminben, a Webmin engedélyt fog kérni, hogy ellenőrizhesse az Apache beállításait, ekkor egyszerűen kattintsunk a ***Configure*** gombon. Ezután pedig menjünk a lap aljára, ahol egy beviteli mező található – ez szolgál az újonnan létrehozott webhelyek megadására.
A szövegmezőbe gépeljük be a webhely IP-címét. Amennyiben névalapú weblap elhelyezése mellett döntünk, ne feledkezzünk meg róla, hogy az ***Add name virtual server address*** lehetőség be legyen kapcsolva. A következő helyen található: ***/home/felh_nev/tartomany/htdocs***, vagyis valóságos névvel: ***/home/dirk/test.com/htdocs***. A szöveges típusú ***Server Name*** mezőt értelemszerűen töltsük ki: ***www.test.com*** és kattintsunk a ***Create*** gombon.
A virtuális kiszolgálók névsorát addig görgessük, amíg rá nem bukkanunk az éppen létrehozott webhelyre. Kattintsunk a webhelyhez tartozó virtuális kiszolgálón. A `cgi-bin` könyvtár kialakításához kattintsunk a ***CGI Programs***-ra és a ***Form*** mezőbe gépeljük be: ***/cgi-bin/***, a szövegmezőbe pedig írjunk ***/home/felh_nev/nevtartomany/cgi-bin/***-t. Fontos, hogy ez utóbbi útvonal perjelre (/) végződjék, vagyis most már névvel együtt: ***/home/dirk/nevtartomany/cgi-bin/***. Végül mentsük ezt is lemezre.
Az egérrel kattintsunk a ***Log Files***-ra. A ***File or Program*** feliratú mezőben szereplő naplóállományhoz való hozzáférés naplózásához gépeljük be a ***/home/felh_nev/tartomany/logs/access_log*** szöveget, azaz a felhasználói azonosítóval és a példaként választott tartománnyal kiegészítve: ***/home/dirk/test.com/logs/access_log***. Amennyiben azt szeretnénk, hogy

Kapcsolódó címek

Rackspace-szel kapcsolatos cikkek a ➲ http://support.rackspace.com/kbsearch.php3 címen érhetők el. Válasszuk a *Platform=Linux* és *Details=Webmin* lehetőségeket.
„A Webmin és a rendszergazdai feladatok" címmel használati útmutatóra bukkanhatunk a ➲ http://www.swelltech.com/support/webminguide/index.html címen.
A Webmin saját honlapja a ➲ http://www.webmin.com/webmin címen olvasható, maga a program is innen tölthető le.

a napló gyakori keresési adatokat tartalmazzon a kérő nevével együtt, az ***Access log files row*** lehetőségnél a ***Format*** oszlopban váltsunk át az alapértelmezésről a szövegmezőre úgy, hogy bejelöljük az előtte található jelölőnégyzetet.
Ezt követően a szövegmezőbe írjuk be a ***combined*** szót. A beállítást mentsük, majd az egérrel kattintsunk a ***Networking and Addresses*** lehetőségen. A ***Server admin email address*** mezőbe e hely webmesterének a levélcímét gépeljük, majd a mező előtti négyzet bejelölésével végezzük el a mentést.
Ezzel a webhely alapvető beállításait tulajdonképpen be is fejeztük, de mielőtt megkezdené önálló életét, még egy végső és nem kevésbé fontos lépést is meg kell tennünk. A jobb felső sarokban ugyanis egy ***Apply changes*** felirattal ellátott gomb található: kattintsunk ezen a gombon, ezzel juttatva érvényre a változtatásokat.

Biztonság

A Webmin számos biztonsági szolgáltatást nyújt. Az első védelmi vonal olyan felhasználói azonosító és jelszóhitelesítő rendszer, amely teljesen független a */etc/passwd* állományban tárolt felhasználói azonosítótól. Ez azt jelenti, hogy valakinek anélkül is jogot adhatunk a Webminhez történő hozzáférésre, hogy bármilyen más operációsrendszer-szintű jogosultságot kellene adnunk neki.
A Webmin teljes mértékben támogatja az SSL-t. Amennyiben a gépünkön Perl SSL-modul található, a Webmin-kapcsolatokat teljes mértékben titkosítani lehet, ily módon a támadókat megakadályozza abban, hogy lehallgatással adatokat szerezzenek.
A Webmin a különböző, már hozzáférhető modulok finomhangolását is lehetővé teszi, például a felhasználók számára a teljes DNS-kiszolgálóhoz hozzáférési jogot biztosíthatunk anélkül, hogy ez a hozzáférési jog az Apache-beállításokra is kiterjedne, vagy éppen ellenkezőleg: a hozzáférési jogokat kizárólag arra a névtartományra korlátozhatjuk, amely fölött ők maguk rendelkeznek.
Amennyiben igényeljük az egyes feladatoknak más-más rendszergazdákra való átruházását, jól kihasználható a vezérléskorlátozási és -újraelosztási lehetőség. Végezetül arra is módunk van, hogy a Webmint úgy állítsuk be, hogy az összes, a felületén keresztül végzett módosítást naplózza – hibakeresés során ez a szolgáltatás rendkívül jól használható.

Mi teszi a Webmint nagyszerűvé?

Amint már bizonyára kitalálták, nagyon kedvelem a Webmint. Szeretem, hogy a szerződés alapján hozzájuthatok a forráskódhoz és módosíthatom is, amennyiben éppen erre van szükségem. Azt is élvezem, hogy a modulrendszer révén akár magam is új dolgokat hozhatok létre, vagy beépíthetem a mások által készített modulokat. Jelenleg a Webmin számára készített LTSP-modul próbálgatásával foglalkozom, hogy néhány rakoncátlan I-Opener megszelídítésében segédkezzem. A lehetséges feladatok kevésbé tapasztalt rendszergazdákra (szobatársakra) való átruházása, valamint az a tudat, hogy a számukra kijelölt területtől nem térhetnek el, jelentősen csökkenti a rám váró feladatok mennyiségét.
Ha a Webmin csak ezt tudná nyújtani, már akkor is el lennék ragadtatva, de akad egy további előnye is: a rendszerállományokat közvetlen módon éri el, vagyis sem adatbázist, sem más szabványostól eltérő adattárolási módot nem használ. Ennek következtében anélkül módosíthatom kézzel az Apache-hoz tartozó ***httpd.conf***-ot , hogy az elkövethető hibák miatt kellene aggódnom.
Az ügyféltámogatás tekintetében ez azt jelenti, hogy a Webmint telepíthetem a kiszolgálóra, és a működtetését nyugodtan másra bízhatom. Amennyiben a gondokkal nem tudna megbirkózni, a hiba elhárítására még mindig használhatom a héjprogramjaimat és vi-ismereteimet.
A beállítást a barátságos parancssor és a háttérbeli adatbázis nélkülözése miatt a közönséges állományokra bízza, amit a vezérlőpultok tervezői túlságosan gyakran hajlamosak figyelmen kívül hagyni. Így végezetül olyan rendszereket állítanak elő, amelyekben mindent a vezérlőpulton keresztül kell beállítani, máskülönben a program befejezi a működését. A Webmin szabad kezet ad nekem rendszergazdai feladataim intézési módjának kiválasztásában. Az Apache beállításait például jobban szeretem közvetlenül módosítani, a BIND-dal viszont teljesen más a helyzet. A BIND hírhedten szőrszálhasogató program, ezért kényelmes beállítófelületként a Webmint használom hozzá. A program az összes – különben rejtett – lehetőséget felajánlja, és nagymértékben csökkenti az elírásból adódó névfeloldási hibák arányát. Örömmel tölt el, hogy a Webmin mennyire jól beleillik rendszergazdai eszköztáramba.
A kezdő rendszergazdák szolgáltatásai mélységének köszönhetően hamar meg fogják szeretni a Webmint. Az egérkattintásokkal működő grafikus felület biztosítja, hogy nem kell mindent fejben tartani, ez azonban a kiszolgálók újdonsült rendszergazdái számára akár elrettentő feladatnak is bizonyulhat.
A Webmin magmoduljai az általuk támogatott szolgáltatások szinte minden képességét és jellemzőjét felvonultatják. Ez azt jelenti, hogy könnyedén vehetünk fel olyan új beállítási lehetőségeket, amelyek létezéséről korábban nem is tudtunk.
A Webmin jólszervezettsége és szolgáltatásbeli gazdagsága ellenére mindenki figyelmét szeretném felhívni rá, hogy a program mégsem teljesen kezdők számára készült. Ha valakinek fogalma sincs róla, mi a DNS-ben a bejegyzés, a Webmin nem fog segíteni rajta.
A Webmin a háttérben futó Linuxot webfelületen jeleníti meg, így amikor egyszerre nyerünk ennyi rugalmasságot és erőt, cserébe fel kell áldoznunk valamennyit a hatékonyságból. Ha valaki részletes ismeretekre tesz szert a szolgáltatások alapelveiben, annak kezében a Webmin nagyszerű eszköz lehet – azt azonban már senki ne várja el, hogy a program az O'Reilly kiadó által megjelentetett nagy BIND-kézikönyv összefoglalóját is megadja.

Dirk J. Elmendorf a Rackspace Managed Hosting cég egyik alapítótagja. Kutatásfejlesztési vezetőként az új termékek fejlesztésében és értékelésében is közreműködik, amelyeket egy hittérítő buzgalmával népszerűsít.

Vírusos levelek? Kizárva!

A levélalapú vírusok megállításának legjobb módja az, ha be sem eresztjük őket a hálózatunkra.

Múlt hónapban láthattunk egy módszert, hogy miként alkalmazhatjuk az Amavis csomagot a levélforgalom vírusellenőrzésére. Nagyobb hálózatoknál komoly gondot jelenthetnek a vírusok, főleg azért, mert ha egyszer bejutottak, akkor kegyetlen gyorsasággal végigfertőzhetik a munkagépeket. Manapság rendkívül fontos kérdés ez, és bár szerencsére a linuxos gépek esetében nem hallunk sokat vírusokról, a munkagépek védelmét is jellemzően kiszolgálóinkkal kell elősegítenünk, hiszen ezáltal rengeteg felesleges munkától mentjük meg magunkat. A belső hálózatot érő támadások leggyakoribb formái a levélvírusok. Az első lépés, amit egy rendszergazda általában tenni szokott ellenük: vírusvédelmi rendszert telepít a munkaállomásokra. Ez bölcs dolog, de számomra járhatóbb útnak tűnik, ha a vírusok rendszerbe jutását mindjárt a bejáratnál meggátoljuk. A vírusok, különösen a makróvírusok, messze leggyakoribb belépési pontja a szervezet levelezőrendszere. Mégis többnyire ez a vírusvédelmi rendszerek legelhanyagoltabb része. A piacon jelenleg kapható levélvírus-védelmi rendszerek gyakran alkalmazáshoz kötöttek, drágák, vagy mindkét tulajdonságot felmutatják (nem beszélve a megbízhatatlanságukról). Közepes méretű vállalkozás lévén a miénk, az idei költségvetésbe vírusirtó csomag beszerzését nem tervezték, így aztán csupán a Linuxhoz és a nyílt forráshoz fordulhattunk. Találtam is az Interneten néhány igen érdekes projektet, amely esetleg megfelelhetett volna az igényeinknek, de végül mégis a saját változat megírása mellett döntöttem. Azt szerettem volna elérni, hogy bármely felhasználó könnyen nyomon követhesse a rendszerünket, és egyszerűen bővíthesse, anélkül, hogy C- vagy Perl-guru lenne. A másik célom az volt, hogy a rendszer ki tudja azokat a hatékony eszközöket használni, amelyek általában minden linuxos alapterjesztésben megtalálhatók. E két tényező biztosítja a program hordozhatóságát, illetve, hogy más is képes legyen kezelni a rendszert a segítségem nélkül.

A rendszer alapjait Bash-héjprogramok, a `metamail`, a `grep`, az Obtuse Systems SMTPd termékei, a Samba és egy parancssoros víruskereső alkotják. Az *1. ábrán* egy folyamatábra stílusú vázlatot láthatunk. Az Obtuse Systems SMTP-tároló és -továbbító csomagja ingyenesen hozzáférhető a ➲ http://www.obtuse.com/smtpd.html címen. E sorok írásának idejében a legfrissebb változat a 2.0. Az általam választott víruskereső a McAfee Virus Scan for UNIX/Linux volt, de számos másikat is választhattam volna. Ezek közül némelyek ingyenesek, mások nem. Mindenféleképpen olyat válasszunk, amelyik a keresés eredményének megfelelően állítja be a kilépési értékét, és amelyhez rendszeresen letölthetünk az ujjlenyomat-frissítéseket. A rendszert felépíthetjük egy már meglévő linuxos tűzfalon, de akár egy külön gépen is, amennyiben linuxos tűzfal esetleg nincs kéznél. Ha erre a célra külön gépet használunk, nem kell túlságosan nagy teljesítményűnek lennie, egy 200 MHz-es 586-os 32 MB memóriával tökéletesen megfelel. A hálózatunk SDSL segítségével kapcsolódik az Internethez, a védelmet pedig egy IP-álcázást (masquerading) futtató Mandrake linuxos gép biztosítja. Ez a felépítés megkönnyíti a tűzfal telepítését.

A belső levelezőrendszer nem annyira fontos, elegendő, ha SMTP vagy ESMPT alatt működik. Mi például a Novell Groupwise termékét használjuk. Minden SMTP-forgalmat (25-ös kapu), ami a tűzfal SMTP-kapujára érkezik, át kell ahhoz a belső géphez irányítanunk, amelyen az SMTP-tűzfalat kiépítettük (vagy magához a tűzfalgéphez, mint a mi esetünkben is). Most lépjünk tovább a tulajdonképpeni beállításokhoz! Az első lépés a könyvtárszerkezet felállítása. Az ide vonatkozó vázlatot a *2. ábrán* találhatjuk. Rendszerünket a */var/spool/smtpd* könyvtár alatt fogjuk felépíteni. Amennyiben átlagos levélforgalmunk meghaladja a napi 25 000 levelet, javaslom, külön lemezrészt fűzzünk be a */var/spool/smtpd* könyvtár alá. Az alapkönyvtár tehát a */var/spool/smtpd* lesz. Ebben a könyvtárban öt alkönyvtárat hozunk létre: ***incoming*** (bejövő), ***outgoing*** (kimenő), ***etc***, ***bin*** és ***quarantine*** (karantén). Először is váltsunk át rendszergazdai jogosultsággal rendelkező felhasználóra, majd gépeljük be a következő parancsot:

```
mkdir -p  /var/spool/
➥smtpd/{etc,bin,incoming,outgoing,quarantine}
```

Következő lépésként állítsuk be a jogosultságokat, hogy a teljes könyvtárrendszert csak a `uucp`-felhasználó érhesse el, mivel az összes program e felhasználó jogosultságával fog futni. A következő parancsok megoldják számunkra a gondot:

```
chown -R uucp.uucp /var/spool/smtpd
chmod 700  /var/spool/smtpd
```

Most már beállíthatjuk a rendszer első összetevőjét. A korábban említett Obtuse Systems honlapjáról le kell töltenünk a ***smtpd*** csomagot. A letöltött fájl könyvtárában adjuk ki a következő parancsot:

```
tar -xzvf smtpd-2.0.tar.gz
```

Ezután váltsunk az ***smtp-2.0*** könyvtárba és a ***Makefile***-t szerkesszük át a következők szerint:

```
SPOOLDIR = /var/spool/smptd

SPOOLSUBDIR = incoming

POLL_TIME = 300

PARANOID_SMTP = 1

JUNIPER_SUPPORT = 0

CHECK_IDENT = 0
```

Azt szeretnénk elérni, hogy az `smtpd` a leveleket az ***incoming*** alkönyvtárban tárolja, a `smtpfwdd` pedig az ***outgoing*** alkönyv-

tárból olvassa őket. Hogy ezt lehetővé tegyük, az ***smtpfwdd.c*** fájlba a 75. sornál szúrjuk be a következő két sort:

```
// levelek let ltØse az outgoing alk nyvtÆrb l
#define SPOOLSUBDIR "outgoing"
```

Befejezésül fordítsuk le és telepítsük a csomagot a következő parancsokkal:

```
make

make install
```

A következő lépés az */var/spool/smtpd/etc* könyvtár benépesítése néhány, az `smtpd` helyes működéséhez szükséges állománnyal. Másoljuk a ***resolv.conf*** fájlt a */etc* könyvtárból a */var/spool/smtpd/etc* könyvtárba, majd a */etc* könyvtárból a ***localtime*** fájlt is másoljuk ide. Az smtpd-2.0 terjesztés könyvtárából az ***antirelay_check_rules_example*** fájlt átmásolhatjuk a */var/spool/smtpd/etc* könyvtárba. Amennyiben szükségünk van ilyesmire, az Obtuse System honlapján nézhetünk körbe további ellenőrző szabályokkal kapcsolatos utasításokért. Az `smtpd` program önműködő indításához a következő sort kell a */etc/inetd.conf* fájlba helyezni:

```
smtp    stream        tcp    nowait root
➥/usr/local/sbin/smtpd     smtpd
```

Ezt a sort az esetleg már meglévő `smtp`-sorok helyére kell írni. Az `smtpfwdd` programot a */etc/rc.d/rc.local* fájlból (vagy ahogy az *rc* fájlunkat éppen nevezik) kézzel kell elindítanunk. Fogjunk hozzá, és a következő sort írjuk be:

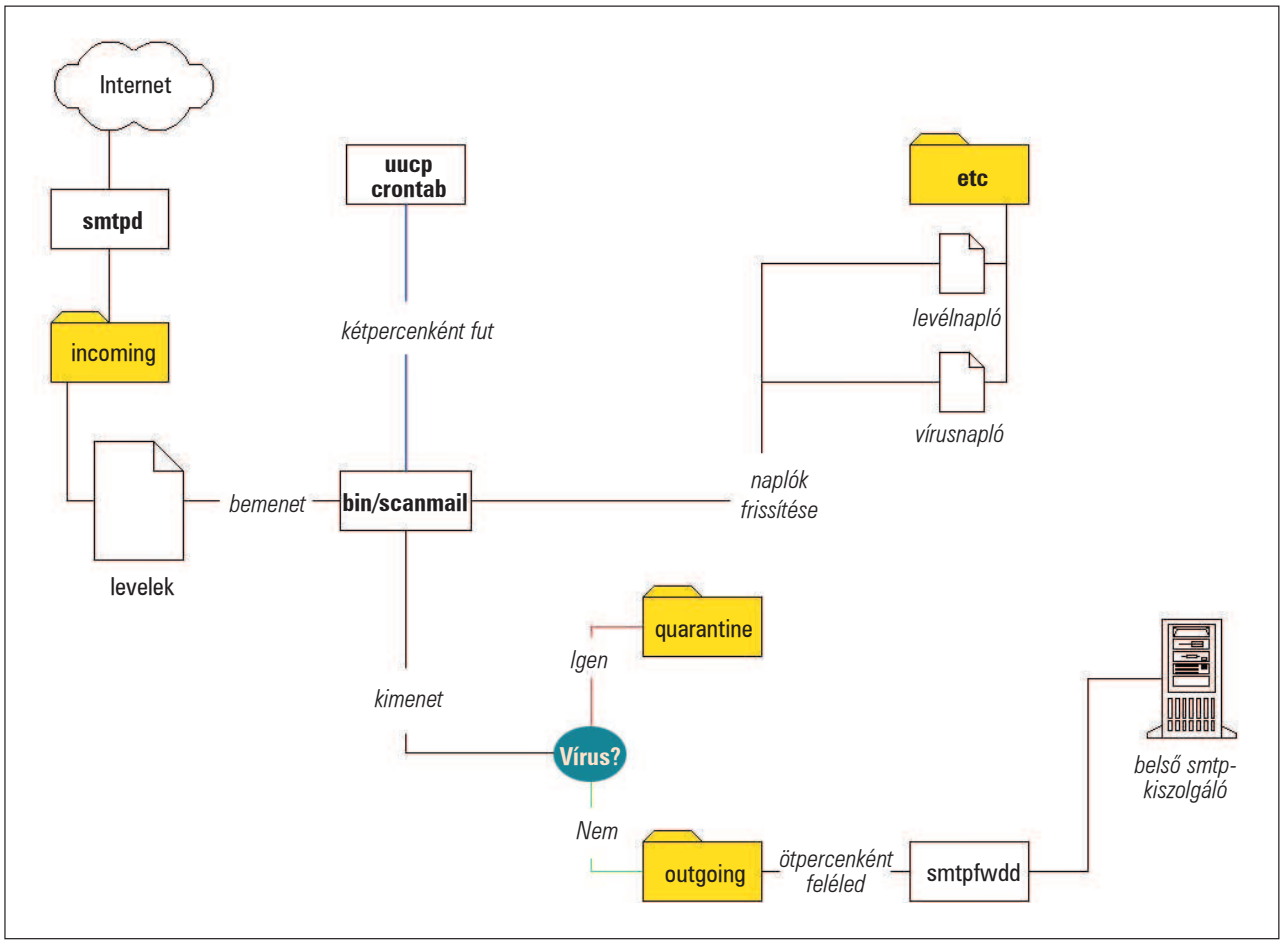
```
### Az smtpfwdd tovÆbb t  dØmon ind tÆsa
/usr/local/sbin/smtpfwdd
```

Végül le kell állítanunk minden esetleg még futó levéltovábbító ügynökprogramot (MTA-t). Ide értendők a ***Postfix***, ***Sendmail***, ***Qmail*** és társai. RedHat-rendszereken ezt egyszerűen a beállítóprogram segítségével is megtehetjük, amennyiben az összes MTA-t leállítjuk. Figyeljünk arra, hogy némely MTA például a Postfix vagy más folyamatok gyermekeiként futnak, így közvetlenül a `kill` paranccsal nem lehet őket „meggyilkolni”. A következő két parancs kiadásával indítsuk be az `smtpd` és az `smtpfwdd` démonokat:

```
kill -HUP `cat /var/run/inetd.pid`
➥/usr/local/sbin/smtpfwdd
```

Mikor már futnak a démonok, ki is próbálhatjuk őket, ha levélszűrő tűzfalunk 25-ös kapuján (ez az `smtp`-kapu) elindítunk egy `telnet`-kapcsolatot:

```
telnet email.firewall.com 25
```



1. ábra A hálózati forgalom és a tűzfal felépítése

ahol az `email.firewall.com` a levélszűrő tűzfalunk neve. A következő visszajelzést kell kapnunk:

```
220 email.firewall.com SMTP ready,
Who are you gonna pretend to be today?
```

Ha bármilyen más üzenetet kapunk, valószínűleg elfelejtettük a kiszolgálón futó MTA-t kikapcsolni. A `ps -e` segítségével ezt könnyen kideríthetjük.

Nézzük csak, hol is tartunk? Ha minden jól ment, most van egy gépünk, amelyen az `smtpd` démon fut és leveleket fogad. Minden beérkezett levél egyszerű szöveges fájlként a ***/var/spool/smtpd/incoming*** könyvtárban tárolódik.
Hogy valóban így is van-e, a következő parancsokkal nézhetjük meg:

```
$ telnet email.firewall.com 25
helo   firewall.com
mail from: joe@firewall.com
rcpt to: fred@firewall.hu
data
Ez egy pr ba.
.
quit
```

Ne felejtsük el a levelünk törzsét egy egyetlen pontot tartalmazó sorral zárni! Ez mondja meg ugyanis a kiszolgálónak, hogy a levelet el akarjuk küldeni.
Ha minden jól megy, a ***/var/spool/smtpd/incoming*** könyvtárban most egy szöveges fájlt fogunk találni. Néhány percen belül a fájlnak el kell tűnnie, mi pedig egy levelet találunk a levelesládánkban. Az `smtd` a leveleket smtpd formátumban menti, ahol smtpd egy véletlenszerűen készített üzenetazonosító.

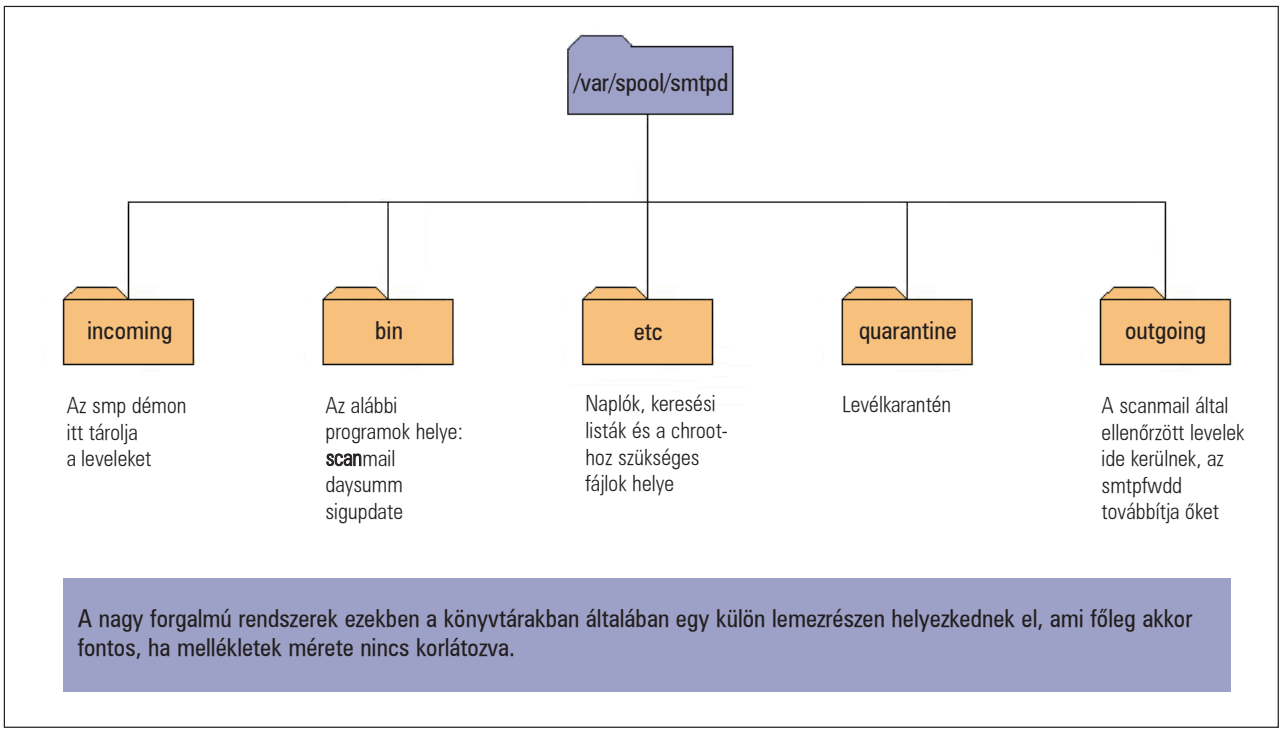
Az `smtpfwdd` ezeket a szöveges fájlokat beolvassa, és továbbítja őket a célkiszolgálónak. Sikeres továbbítás után a fájl törlődik.
De hogy is kerül a levél az ***incoming*** könyvtárból az ***outgoing*** könyvtárba? Nos, ez az a pont, ahol levélvizsgáló parancsfájlunk belép a képbe. A parancsfájl az ***incoming*** könyvtárban található fájlokat vírusra utaló nyomokat keresve egytől-egyig végignézi. Ha a fájl nem tartalmaz vírust, az ***outgoing*** könyvtárba kerül, ha viszont igen, a ***quarantine*** könyvtárba helyeződik át.
Ilyen egyszerű az egész. A levélvizsgáló parancsfájlhoz azonban előbb szükségünk lesz egy működő víruskeresőre, tehát előbb erre a feladatra összpontosítsunk.
Mindenféleképpen parancssoralapú víruskeresőre van szükségünk. Én a magam részéről – mint már említettem – a McAfee Virus Scan for UNIX/Linux rendszert választottam, így itt most erről fogok írni. A McAfee termék igen széles kilépőkód-listával rendelkezik, ezáltal a parancsfájlokba meglehetősen könnyen beilleszthető. A terméket egyszerűen beszerezhetjük ➲ http://www.nai.com honlap webboltján keresztül. A termék telepítése után bizonyosodjunk meg arról, hogy a `uucp` felhasználó által végrehajtható legyen. Ehhez be kell lépnünk a ***/usr/local/uvscan*** könyvtárba, majd ki kell adnunk a következő parancsot:

```
chmod -R 755
```

Próbáljuk ki, hogy működik-e! Váltsuk át `uucp`-felhasználóra, és írjuk be a következő parancsot:

```
/usr/local/uvscan/uvscan -version
```

Amennyiben a próba sikeresnek bizonyult, továbbléphetünk, és felfrissíthetjük a vírusujjlenyomatokat. Az ujjlenyomatfájlok (avagy a „definíciós” fájlok – ahogyan gyakran emlegetik őket) alkotják az összes víruskereső velejét. Ezért fontos, hogy



2. ábra A könyvtárszerkezet

a frissítés időről időre önműködő módon megtörténjen. Nagyon fontos, hogy ezeket a frissítéseket legalább kéthetente ellenőrizzük, mivel havonta legalább három új változat jelenik meg. Ehhez a megoldáshoz először egy `sigupdate` Bash-parancsfájlt fogunk alkotni, amelyet majd a */var/spool/smtpd/bin* könyvtárba helyezünk el. Akárcsak eddig, most is győződjünk meg arról, hogy az `uucp`-felhasználó-e a parancsfájl birtokosa, és hogy a fájlt végrehajthatónak jelöltük-e be. A parancsfájl tartalmát az 1. listában (24. CD Magazin/Virus könyvtár) találjuk, elég könnyen követhető. A `sigupdate` fájlt hetente egyszer fogjuk futtatni, lehetőség szerint a nap valamilyen kevésbé terhelt időszakában. Ezt később tesszük meg egy `crontab` sor beillesztésével. Továbbá készítenünk kell a `uucp`-felhasználó saját könyvtárában egy ***.netrc*** nevű fájlt. Szerkesszük át úgy, hogy a következőképpen nézzen ki:

```
machine ftp.nai.com
login anonymous
password admin@domain.com
macdef init
cd pub/antivirus/datfiles/4.x
bin
prompt
mget dat-*.tar
close
bye
```

A ***.netrc*** fájl előre meghatározott gazdagépek FTP-elérését vezérli. Ez a legjobb módja annak, hogy FTP-folyamatunkat önműködővé tegyük. A .netrc írásmódról az FTP súgóoldalon olvashatunk. Lássunk neki, és amint elkészült a két fájl, futtassuk le a `sigupdate`-et.
A frissítés végén futtassuk le a következő parancsot:

```
/usr/local/uvscan/uvscan -version
```

Nézzük meg a vírusadatfájl létrejöttének dátumát. Valamilyen közeli időpontot kell látnunk, általában egy hónapnál nem régebbit. Ha nem így lenne, parancssorból kell ellenőriznünk, hogy a `sigupdate` helyesen működik-e.
Most lépjünk tovább a fő levélvizsgáló parancsfájlra. Ezt a fájlt `scanmail`-nek fogjuk hívni és a */var/spool/smtpd/bin* könyvtárba kerül. Ez a parancsfájl fogja végrehajtani az összes közvetlen műveletet az `smtpd` által létrehozott levélszövegfájlok között. Készítsünk egy fájlt, majd tegyük végrehajthatóvá és adjuk át az `uucp`-felhasználónak. A teljes parancsfájlt a 2. listában találhatjuk meg (elérhető a 24. CD Magazin/Virus könyvtárában). Minthogy a fájl meglehetősen sok megjegyzést tartalmaz, itt csak nagy vonalakban térünk ki rá.
A 19. sortól kezdve a `scanmail` előbb belép az ***incoming*** könyvtárba, és az ott található fájlneveket egy vektorban helyezi el. Ezután minden egyes fájlnéven végiglépdel, és a `grep` segítségével a fájlokban adott mintákat keres. Minden levél tartalmát kétszer ellenőrizzük. Első körben azokat a csatolt fájlokat szűrjük ki, amelyek nyilvánvalóan rosszak. Az ehhez a kereséshez tartozó minták a ***matches.bad*** fájlban tárolódnak, amelyet később fogunk létrehozni. Ha a `grep` talál valamit, a levél a karanténba kerül, és egy levelet küldünk a rendszergazdának, amelyben megtalálható a dátum, a fájlnév, és hogy a levél kinek, illetve kitől érkezett.
Ha nem volt találat, jöhet a második kör. Ez esetben a `grep` a ***matches.doc*** nevű fájlt fogja használni, hogy kiszűrje azokat a csatolt állományokat, melyek makróvírusokat vagy beágyazott vírusokat tartalmazhatnak. Ha talál valamit, a csatolt részt a `metamail` program segítségével egy dinamikusan létrejövő ideiglenes könyvtárba bontja ki. Az ideiglenes könyvtár neve a csatolt állomány neve lesz `"_d"` utótaggal kiegészítve.
Az ideiglenes könyvtár tartalmát ezután a parancssoros víruskeresőnkkel végignézetjük.
Ha valamilyen vírust találtunk (ezt a kereső visszatérési értékéből tudjuk meg), a `scanmail` a levelet és a csatolt állományokat karantén alá helyezi, majd figyelmeztető levelet küld a rendszergazdának. Egyúttal udvarias levelet küldünk a levél feladójának, amelyben értesítjük, hogy érdemes lenne végignézni a rendszerét, illetve megadjuk a talált vírus nevét.
Ha ez idáig nem találtunk vírust, a levél az ***outgoing*** könyvtárba kerül, ahonnan az `smtpfwdd` a belső levélkiszolgálóhoz továbbítja majd. Az `smtpfwdd` a kimenő könyvtárat ötpercenként egyszer ellenőrzi.
A következő lépés annak a fájlnévlistának az elkészítése, amelyet a `scanmail` a gyanús csatolt állományok felderítéséhez fog használni. A `scanmail` a fájlok közt a `grep` eszköz segítségével keres. Kihasználjuk a `-f` kapcsoló nyújtotta előnyöket, mivel így a `grep` a kereséshez használt mintákat egy megadott szöveges fájlból fogja kiolvasni. A szöveges fájl szerkezete igen egyszerű, minden sorban egy minta található. A `grep` a fájlban felsorolt bármely mintával való egyezést találatként fogja értékelni. Váltsunk a */var/spool/smtpd/etc* könyvtárba és hozzunk létre két fájlt ***matches.bad*** és ***matches.doc*** néven. A ***matches.bad***-be az olyan fájlok névmintáit helyezzük, amelyeket semmiféleképpen nem szeretnénk anélkül a rendszerbe ereszteni, hogy a rendszergazda meg ne vizsgálta volna őket. A ***matches.doc*** fájlnak ezzel szemben azokat a dokumentummintákat kell tartalmaznia, amelyek beágyazott vírusokat tartalmazhatnak, ilyenek például a Word-dokumentumok és a táblázatkezelők állományai. Amikor ezeket a fájlneveket hozzuk létre, minden sorban a `filename=.*\.exe` formátumot használjuk. Ez azért szükséges, hogy ne kapjunk hamis riasztásokat a mimekódolás olyan véletlen karaktersorozatai miatt, amelyek történetesen megegyeznek a `grep` által keresett mintával. Figyeljünk arra is, hogy ez a fájl semmiképpen ne tartalmazzon üres sort, hiszen a `grep` az üres sort is keresendő mintának fogja venni, és minden levélre találatot fogunk kapni. A Vim jó szerkesztőprogram e célra, mivel könnyen láthatjuk a benne szereplő üres sorokat. Az általam használt fájlok tartalmát a 3. listában (24. CD Magazin/Virus köyvtár) lelhetjük fel.
A másik felhasznált parancsfájl neve `daysumm` lesz és a */var/spool/smtpd/bin* könyvtárban helyezzük el. Hozzuk létre ezt az állományt a *4. listának* megfelelően.
A `daysumm` a napi tevékenységről értesítőt küld a rendszergazdának. Megmutatja, hány levél érkezett aznap, közülük hány volt vírusos, illetve melyek voltak ezek a vírusok. A cronban állítjuk be, hogy minden este 11:59-kor fusson le.
A `daysumm` parancsfájl a */var/spool/smtpd/etc* könyvtárban elhelyezett ***virus.$date*** és ***email.$date*** fájloktól függ. Ezek szöveges fájlok, amelyek dinamikusan jönnek létre, a `scanmail` frissíti őket és dátumfüggők.
Emiatt a `daysumm` programot mindig éjfél előtt kell lefuttatnunk, különben az időbélyeg (timestamp) megváltozik és rossz fájlok kerülnek beolvasásra.
Biztos észrevették már, hogy valahányszor magáról a tűzfalról küldtünk ki üzenetet – például a parancsfájlokban is –, mindig egy `sendmail -q` parancsot is kiadunk. A `-q` ugyanis azt mondja meg a `sendmail`-nek, hogy induljon el, és nézzen körül, van-e kimenő üzenet, ha van, küldje el, majd lépjen ki. Ez hatékonyan kiüríti az összes kimeneti sort, ami azért szük-

4. lista A daysumm parancsfájl

```
#!/bin/sh
mailto="admin@domain.com"
timestamp='date '+%A - %B %d %Y
etc="/var/spool/smtpd/etc"
date='date '+%m%d%Y
email_total='cat $etc/email-log.$date'
virus_total='cat $etc/virus-log.$date'


mail -s "DAILY E-MAIL SUMMARY" $mailto <<eoi


  Date:     $timestamp
  E-Mail total: $email_total
  Virus total:  $virus_total

eoi

/usr/sbin/sendmail -q

exit 0
```

séges, mert többé semmilyen MTA nem fut a gépünkön.
Az ***smtpd*** csomag nem MTA, hanem egy tároló- és továbbítócsomag. Úgy is elképzelhetjük, mint egy kifejezetten levéltovábbításra kihegyezett programot. E külön parancs nélkül tehát soha egyetlen levelet sem kaphatnánk meg a tűzfalról.
Itt az ideje, hogy az egész folyamatot a cron démon segítségével önműködővé tegyük. Ezt a uucp-felhasználó személyes crontab állományának felhasználásával fogjuk megtenni. Lépjünk be uucp-felhasználóként, majd adjuk ki a `crontab -e` parancsot, ami az uucp-felhasználó cron tábláját nyitja meg szerkesztéshez. A `scanmail`, `daysumm` és `sigupdate` parancsfájlok részére hozzuk létre a következő bejegyzéseket:

```
MAILTO=""

# A scanmail parancsfÆjl minden kØt percben
# fusson le
*/2 * * * * /var/spool/smtpd/bin/scanmail

# A daysumm parancsfÆjl minden nap 11:59-kor
# induljon el
59 23 * * * /var/spool/smtpd/bin/daysumm

# A sigupdate minden cs t rt k n 4:00-kor
# fusson le.
0 16  * * 4 /var/spool/smtpd/bin/sigupdate
```

Természetesen a futásidőpontokat megváltoztathatjuk úgy, hogy megfeleljenek az igényeinknek. Az, hogy a `scanmail`-t milyen sűrűn futtassuk le, főként a napi levélforgalmunktól függ. Ha a napi mennyiség tízezer levél felett van, a magam részéről az időközt két percben határoznám meg, az `smtpfwdd`-t pedig ötpercenkénti futásra állítanám be. Így nem küldünk egyszerre hatalmas levélcsomagokat a belső kiszolgálóra. Ha naponta 1000 vagy ennél kevesebb levéllel kell csak számolnunk, elég, ha a `scanmail` minden tizedik percben fut le, az `smtpfwd` pedig ötpercenként néz körül. Ne feledjük el a `MAILTO=""` kifejezést kitenni a crontab-bejegyzések elejére! Ez azért szükséges, hogy a crond a végrehajtott cron-feladatokról ne küldjön levelet az uucp-felhasználónak. Ha minden két percben egy levél érkezik, az gyorsan felgyülemlik, és az uucp-felhasználó soha nem ellenőrzi a leveleit.
Felállítottam egy Samba-megosztást is, hogy a windowsos gépemről is hozzáférhessek a ***/var/spool/smtpd*** könyvtárszerkezethez. A főként Windowst használó rendszergazdáknak ez a beállítás hasznos lehet – így nem kell mindig SSH-kapcsolatot nyitnom, valahányszor vírusra figyelmeztető levelet ellenőrzök.
A következő sorokat kell a ***/etc/smb.conf*** fájlba illeszteni:

```
[mail-gate]
       Comment = LevØltßzfal k nyvtÆrak
       Path = /var/spool/smtpd
       Valid users = nev nk
       Admin users = nev nk
       Browseable = no
       Read only = no
```

ahol a „nevünk” természetesen a Samba-felhasználói nevünket jelenti. Ami még hiányzik ahhoz, hogy tűzfalunkról az összes bejövő SMTP-kapcsolatot az új levélszűrő kiszolgálónkra irányítsuk, az, hogy tűzfalunknak IP-álcázást kell használnia. Egyszerűen adjuk ki a következő parancsot:

```
ipmasqadm portfw -a -P tcp -L tßzfal 25
➥-R cØlgØp 25
```

ahol a `tßzfal` a tűzfalunk címe, a `cØlgØp` pedig az új levélszűrő gépünk címe. Amennyiben levélszűrő tűzfalunkat közvetlenül a már meglévő tűzfalunkon szeretnénk futtatni, semmin sem kell változtatnunk.
Ha más típusú tűzfalrendszert használunk, olvassuk el a leírást, hogy megtudjuk, miképpen állíthatjuk be a kapuátirányítást. Remélhetőleg nem okoz nagy gondot, de előfordulhat, hogy kapcsolatba kell lépnünk a termék készítőjével. Ne feledjük el ezt az átirányító parancsot betenni az indító parancsfájlokba, hogy túlélje a rendszerindításokat.
Ha minden jól ment, végre működő levélvírusszűrő tűzfallal rendelkezünk. Próbáljunk meg küldeni magunknak néhány próbaüzenetet valamelyik ingyenes webes levelezőszolgáltatótól, hogy lássuk, minden jól működik-e. Én például adott időközönként küldök magamnak egy makróvírussal fertőzött állományt, hogy lássam, a rendszer még mindig helyesen működik-e. Sok dologgal lehetne még bővíteni ezt az alaprendszert. Különösen ígéretes a `daysumm` parancsfájl, amelyet jócskán fel lehetne még fejleszteni. Jelenleg épp egy CGI-parancsfájlon dolgozom, amely a pillanatnyi átlagokat – például az átlagos napi mennyiséget – az Interneten keresztül jelenítené meg. Természetesen ez csak egy út a több százból, ahogyan ez a rendszer levelezőrendszerünket megvédheti, anélkül, hogy befolyásolná a cég költségvetését. Ha valaki esetleg kitalálna valamilyen ügyes továbbfejlesztést a rendszerhez, kérem, tudassa velem. Nagyon szeretnék hallani róla.

Dave Jones
(davidashleyjones@hotmail.com)
három évig volt hálózati rendszergazda az alabamai Birminghamben. Amikor éppen nem a számítógép előtt ül, egy-egy szál jófajta dohányt szív el, vagy a feleségével és a lányával X-aktákat néz a tévében.

Gyanús adatforgalom felderítése

Használjunk psad-t IP Chains, illetve IP Tables szabálykészletünkhöz, hogy felfedezhessük a TCP- és UDP-kapuvizsgálatokat, és más hálózati gonoszkodásokat.

Alig fél éve végre kiadásra került Linux 2.4.0 rendszermag által a GNU/Linux hatalmas lépést tett a vállalati operációs rendszerek világa felé. Több magrész is fejlődött ugyan a 2.2.x sorozat óta, de egyik sem olyan jelentős mértékben, mint a tűzfalkód. A 2.4.x rendszermagsorozatban megjelent a Netfilter (lásd a ***Linuxvilág*** 2001. októberi számát, 27–31. oldal), amely a 2.2.x sorozat régi IP Chains tűzfalkódjának helyére lépett, és több olyan képességgel is rendelkezik, amelyre egy valódi kereskedelmi tűzfalnak szüksége lehet. Alaposságát képességei bizonyítják a leginkább: a DoS-védelem (DoS = Denial of Service, szolgáltatásmegtagadás alapú támadás, azaz tömeges kérelemmel való bombázás a kiszolgáló lebénítása céljából), illetve sűrűségkorlátozás, hálózati címátalakítás (NAT), MAC-címszűrés, és végezetül, de nem utolsó sorban, tetszés szerinti TCP-jelzőkombináción alapuló TCP-csomagszűrés és naplózás. Ezzel szemben, az IP Chainsnek szánalmas korlátja is akad (nem végez alapos vizsgálatot), ugyanis mindössze két fajta TCP-csomagot képes megkülönböztetni aszerint, hogy a csomag SYN jelzője be van-e állítva vagy sem. A Netfilter azon képessége, hogy bármilyen tetszőleges TCP jelzőkombinációt megenged, lehetővé teszi, hogy azokat a kifinomult kapuvizsgálatokat is felderíthessük, amelyeket az Nmap segítségével bárki könnyedén a gépre engedhet. A kapuvizsgálatok működésének bemutatásához és felderítésük módozatainak az ismertetéséhez először némi Nmap-háttérismeretre (lásd még a ***Linuxvilág*** 2001. májusi számát, 45–49. oldal) lesz szükségünk.

Nmap

Az Nmap a világ legismertebb, többfajta fejlett módszert egyesítő programja, amellyel a nyitott kapukat a célgépen, illetve a hálózaton egyaránt felderíthetjük. Az Nmap-rendszer a nyitott kapuk minél tökéletesebb meghatározásához kínál többek között ujjlenyomat-elemzést és TCP-sorozatszám előrejelzést (TCP sequence number prediction) is. Az Nmap által alkalmazott három legérdekesebb TCP-vizsgálati mód a FIN-, a NULL- és a XMAS-vizsgálat. A megszokott TCP-forgalomban a FIN-csomagokat (az olyan csomagokat, amelyeknél a FIN jelző be lett állítva) a TCP-kapcsolat bármely végéről küldeni lehet, jelezvén, hogy az üzenetváltásnak vége, pontosabban nincs több küldendő adat. A FIN-vizsgálat azon az elven alapul, hogyha egy „árva” FIN-csomagot (olyan FIN-csomagot, amely semmilyen létező TCP-üzenetváltásnak nem része) küldünk egy nyitott TCP-kapura, nem kapunk visszajelzést. Ha viszont egy ilyen csomagot egy zárt kapura küldünk, a hagyományos szűretlen TCP-verem várhatóan egy RST-csomaggal válaszol. Így aztán a FIN-csomagvizsgálatnál az Nmap egyszerűen minden egyes célkapura elküld egy magányos FIN-csomagot és várja, vajon visszaérkezik-e valahonnan RST-csomag. Azok a kapuk, amelyek nem válaszoltak RST-csomaggal, nyitva vannak (vagy tűzfallal szűrtek). A NULL- és a XMAS-vizsgálat is hasonló módszeren alapul, de ahelyett, hogy csak a FIN-jelzőt állítaná be, az XMAS-vizsgálat az URG- és PSH-jelzőket is beállítja, a NULL-vizsgálat pedig olyan csomagokat készít, amelyeknek egyetlen jelzőjük sincs beállítva.

A Netfilter beállítása

A biztonságos tűzfal készítésének alapgondolata az alapértelmezett tagadó hozzáállás. Eszerint, amely adatforgalom nincs kifejezetten engedélyezve, azt a tűzfalnak meg kell tagadnia vagy el kell utasítania.
Tűzfalak esetében három kifejezést használunk, amelyek ugyan rokon értelműek, de a tűzfalak esetében más és mást jelentenek:
REJECT – elutasít (ilyenkor a tűzfal egy „elutasítva” üzenetet küld vissza),
DENY – megtagad (a tűzfal kidobja a csomagot, nincs válasz),
DROP – elvet, eldob (a tűzfal kidobja a csomagot, nincs válasz).
Továbbá a tűzfalat úgy kell beállítani, hogy minden jogosulatlan csomagot egy naplófájlba naplózzon, így azt később a rendszergazda vagy a `psad`-hoz hasonlóan önműködő naplófájlfigyelő rendszer elemezheti.
Ha valakit bővebben érdekel, hogyan tudná az IP Tablest a rendszerén beindítani, melegen ajánlom ***Rusty Russell*** Netfilter HOWTO-ját a ➲ http://netfilter.samba.org-on.

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot source   destination
ACCEPT     all  anywhere anywhere state
           RELATED,ESTABLISHED
ACCEPT     TCP  anywhere anywhere TCP dpt:ssh
           flags:FIN,SYN,RST,PSH,ACK,URG/SYN
ACCEPT     TCP  anywhere anywhere TCP dpt:www
           flags:FIN, YN,RST,PSH,ACK,URG/SYN
LOG        TCP  anywhere anywhere LOG level
           warning prefix 'DENY ·
DROP       TCP  anywhere anywhere
```

Kapuvizsgálatok megállítása

Vajon megállíthatja-e egy ilyen IP Tables-szabályzat a kapuvizsgálatokat? Először minden TCP-csomagot (a TCP-jelzőtől függetlenül), amely a 80-as vagy a 22-es kaputól eltérő számú kapura érkezne, a négyes számú szabály szerint naplózzunk, majd az ötös számú szerint dobjuk el. Ez máris minden zajos vizsgálatnak ellátja a baját, amely a létező 65 535 TCP-kapu közül nem pont a 80-as vagy a 22-es kaput célozza. No de mi történjen azokkal a vizsgálatokkal, amelyek a 80-as vagy a 22-es kaput célozzák? Ez a vizsgálat típusától függ. A hármas és négyes szabályok elfogadják azokat a csomagokat, amelyeknek be van állítva a SYN-jelzőjük, de csak akkor, ha minden más jelző törölve van, így azok a vizsgálatok, amelyek kizárólag SYN-csomagokat használnak, sikeresek lesznek. Ugyanígy minden vizsgálat, amely rendes `TCP connect()` rendszerhívást használ, ahogyan azt minden rendes böngésző vagy SSH-ügyfél is tenné, szintén sikeres lesz, hiszen az egyes számú szabály engedélyezi a három TCP-kézfogás végrehajtását.
Az Nmap mindkét vizsgálati módszert támogatja a `-sS` (félig

1. lista. Tűzfalüzenetek beolvasása kmsgsd-vel

```
open LOG, ">> /var/log/psad/fwdata" or die
"$!\n";

while (1) {
  open FIFO, "< /var/log/psadfifo" or die "$!\n";
  $service = <FIFO>; # ne lass tsuk a chomp-pal
  if (($service =~ /Packet\slog/ || $service =~
/IN.+?OUT.+?MAC/) && $service =~
/DROP|REJECT|DENY/) {
    # a tiltott/eldobott csomag napl zÆsa
    # a fwdata fÆjlban
    my $old_fh = select LOG;
    $| = 1;  # a LOG-ba helyezz k a pufferelt
             # kimenetet
      print "$service";
      select $old_fh;
   }
}
```

3. lista Webkiszolgálóhoz csatlakozó TCP-folyamat IP Tables-üzenetei

```
Aug  4 12:16:56 myserver kernel: IN=eth1 OUT=
MAC=00:a0:cc:e2:1f:f2:00:20:78:1c:60:58:08:00
SRC=192.168.10.10 DST=10.10.10.50 LEN=60 TOS=0x00
PREC=0x00 TTL=64 ID=25415 DF PROTO=TCP SPT=2591
DPT=80 WINDOW=32120 RES=0x00 SYN URGP=0

Aug  4 12:16:56 myserver kernel: IN= OUT=eth1
SRC=10.10.10.50 DST=192.168.10.10 LEN=60 TOS=0x00
PREC=0x00 TTL=64 ID=0 DF PROTO=TCP SPT=80
DPT=2591 WINDOW=5792 RES=0x00 ACK SYN URGP=0

Aug  4 12:16:56 myserver kernel: IN=eth1 OUT=
MAC=00:a0:cc:e2:1f:f2:00:20:78:1c:60:58:08:00
SRC=192.168.10.10 DST=10.10.10.50 LEN=52 TOS=0x00
PREC=0x00 TTL=64 ID=25416 DF PROTO=TCP
SPT=2591 DPT=80 WINDOW=32120 RES=0x00 ACK URGP=0
```

nyílt vagy SYN-vizsgálat) és a `-sT` (TCP `connect()` vizsgálat) parancssori kapcsolókkal. Bármely más vizsgálati módszert, amely nem az ezekre a kapukra irányuló szabályos TCP-forgalmon alapul, megakadályozunk és naplózunk. Ide értendő a FIN-, XMAS- és NULL-vizsgálat, amit a korábbi Nmap-részben említettünk, illetve a SYN/FIN- és az ACK-vizsgálat is.
Most ugyan végre bizonyosak lehetünk abban, hogy az IP Tables tűzfalunk képes megállítani a kapuvizsgálatokat, de ne üldögéljünk boldogan a babérjainkon! Egyáltalán nem elég csak megállítani a vizsgálatokat, a lehető legmegbízhatóbban azonosítanunk is kell őket, hiszen a kapuvizsgálat gyakran egy sokkal komolyabb támadásnak lehet az előjele.

A psad bemutatása

A Port Scan Attack Detector (`psad`) Perl nyelven íródott program, amelyet arra terveztek, hogy a szigorú IP Chains, illetve IP Tables szabálykészletek használatával TCP- és UDP-kapuvizsgálatokat derítsen fel. Lehetőségünk nyílik néhány előre beállítható veszélyességi szint (természetesen rendelkezésre áll néhány célravezető beépített szint is) létrehozására, kérhetünk figyelmeztetést levélben is, igény szerint önműködően letilthatjuk a támadó IP-címet az IP Chains, illetve IP Tables tűzfalszabályok dinamikus átállításával. Ezenkívül bőbeszédű riasztásokat kaphatunk, amelyek tartalmazzák a forrást, a célt, a vizsgálat alá vont kaputartományt, a kezdés és a befejezés időpontját, és a `dns` és `whois`-keresések eredményét. IP Tables tűzfal használata esetén a `psad` képes kihasználni a továbbfejlesztett naplózási képességet, és fel tudja deríteni az olyan különösen gyanús vizsgálatokat, mint amilyen a FIN, XMAS vagy a NULL, azaz amelyeket valaki az Nmap segítségével könnyen a gép ellen fordíthat. Továbbá a `psad` több, a ***Snort*** behatolásfelderítő rendszerhez tartozó TCP- és UDP-aláírást tartalmaz, amelyek alapján fényt deríthet a vizsgálatokra, illetve a különféle hátsó ajtókra (backdoor) – például az EvilFTP, GirlFriend, SubSeven – és a DdoS-eszközök (mstream, shaft) adatforgalmára. A `psad` a GNU Public License alá tartozó ingyenes program és a ➲ http://www.cipherdyne.com honlapról tölthető le.
A `psad` alapfeladata, hogy a segítségével értelmezzük az IP Chains vagy IP Tables tűzfal által készített naplóüzeneteket, és felderíthessük a gyanús hálózati forgalmat. A feladat kivitelezéséhez a `psad`-nak hatékony módszerre van szüksége, amivel a tűzfal által a rendszernaplóba írt üzenetekből a számára fontos adatokat kigyűjtheti. A `psad` ezért telepítéskor egy `psadfifo` nevű nevesített csővezetéket (named pipe) készít a */var/log/* könyvtárban, és beállítja a rendszernaplódémont (`syslogd`), hogy a ***kern.info*** üzeneteket ebbe a vezetékbe írja. A syslog nyelvezetében a ***kern*** szolgáltatás által jelentett IP Chains és IP Tables naplóüzenetek ***info*** naplószinten jelennek meg. A `psad` által végzett feladatok oroszlánrészét két külön démon, a `kmsgsd` és a `psad` végzi.

kmsgsd

A `kmsgsd` démon viszonylag egyszerű felépítésű. Feladata mindössze annyiból áll, hogy megnyitja a `psadfifo` vezetéket, majd minden olyan ***kern.info*** üzenetet beolvas, amely arra utalhat, hogy az IP Chains, illetve IP Tables egy csomagot elvetett vagy elutasított, végül minden ilyen üzenetet a */var/log/psad/fwdata* `psad` adatfájlba ír. Mivel a `kmsgsd`-be épített `regex` (szabványos kifejezés-kereső) csak az olyan csomagokat keresi ki, amelyeket elvetettek vagy elutasítottak, az ***fwdata*** fájl már megszűrt adatfolyam lesz, amely csak olyan adatot tartalmaz, ami biztonsági szempontból fontos lehet. Ez az adathalmaz azonban csak annyira lehet teljes és beszédes, amennyire a tűzfal naplózza őket, ezért van tehát szükség a lehető legszigorúbb tűzfalszabályokra.
Az IP Tables nem támogatja a naplózás lehetőségét semmilyen szabályhoz, amely csomagokat vet vagy utasít el. Ezt a gondot azonban könnyen áthidalhatjuk, ha az elvetést végző szabály elé egy `--log-prefix` kapcsolót tartalmazó naplózási szabályt teszünk. Ezt figyelhetjük meg a `simplefirewall.sh` negyedik szabályában. Az *1. listában* látható `kmsgsd`-kódrészlet azt mutatja be, hogyan olvas a `psadfifo` tűzfalüzeneteket nevesített csővezetékről, és miképpen lehet szabványos kifejezések segítségével az IP Chains vagy az IP Tables által elvetett csomagok üzeneteit kiszűrni.

psad

Miután a ***fwdata fájl*** megtelt az elutasított csomagokkal, kiértékelésük a psad démon feladata. Ez alapján dönthető el, hogy az adott csomag vajon része-e kapuvizsgálatnak vagy más gyanús hálózati forgalomnak. A psad ezt úgy végzi el, hogy meghatározott időnként megvizsgálja, vannak-e új sorok az ***fwdata*** fájlban, hiszen az azt jelenti, hogy a tűzfal nemrégiben csomagokat utasított el. A kapuvizsgálatok egytől ötig terjedő veszélyszinthez vannak rendelve attól függően, hogy a tűzfal hány csomagot utasított el egy megadott időn belül. Ezenkívül a vizsgálatot valamely veszélyszinthez aszerint is hozzá lehet rendelni, hogy tartalmaz-e valamit a ***psad_signatures*** fájlban található veszélyes minták közül. Ilyen mintára példa az ***NMAP Fingerprint attempt*** (azoknál a csomagoknál, ahol az URG-, PSH-, SYN- és FIN- jelzők be vannak állítva) és a ***DDoS – mstream client to handler*** (ahol a SYN-csomag a 15 104-es kaput célozza). Amikor a kapuvizsgálat elér egy meghatározott veszélyszintet, a rendszer a következő adatokat tartalmazó levelet küldi:

- a vizsgálat forrásának IP-címe,
- a cél IP-címe,
- a legutóbbi vizsgálati szakaszban végigvizsgált kapuk tartománya (TCP vagy UDP),
- a vizsgálat kezdete óta megvizsgált kapuk teljes tartománya (TCP és UDP),
- kezdési és befejezési időpont,
- a psad által hozzárendelt veszélyszint,
- a fordított DNS-adat (reverse dns information),
- a vizsgálat által használt TCP-jelzők (azokkal a megfelelő Nmap parancssori kapcsolókkal együtt, amelyek ilyen vizsgálatot eredményeznének),
- a whois-adat.

A psad ahelyett, hogy kérelmével a Linux-terjesztéseken alapértelmezés szerint telepített whois ügyfélhez fordulna, inkább a ***Marco d'Itri*** által írt kitűnő whois ügyfelet használja. Ez az ügyfél ugyanis rendelkezik azzal a képességgel, hogy csaknem minden vizsgálódó forrás IP-cím esetén mindig a megfelelő whois-adatbázist kérdezi le. A psad arra is képes, hogy az IP Chains, illetve az IP Tables szabálykészletet átállítsa: minden olyan IP-címet kitilt, amely elér egy bizonyos veszélyességi szintet. Ez a képesség alapértelmezés szerint nincs bekapcsolva, mivel sok rendszergazda nem szeretné, hogy hálózatszerte számos rendszergazda legyen képes a tűzfalat befolyásolni úgy, az elérést az adott weblaphoz kitiltva e gépek nevében (IP-cím átírással) veszélyes csomagokat küld a tűzfalhoz. Más rendszergazdák viszont valamilyen magas határértéket szeretnek a kapuvizsgálat vagy más veszélyes forgalom esetére beállítani, és a psad-ra bízzák az önműködő kitiltását. Figyeljük meg, hogy bármilyen adat, amit a psad megvizsgálhat, a tűzfal által már eleve tiltva van. Ha a vizsgálat elér egy megfelelően magas szintet, az önműködő kitiltó képesség a támadó forráscímről érkező valamennyi forgalmat letiltja, mivel a rendszergazda valószínűleg nem szeretné, ha egy ilyen IP-címről bármiféle csomag érkezne a helyi hálózatra, legyen az törvényes vagy törvénytelen.
A psad jelzőrendszerrel is rendelkezik, így ha a psad-nak USR1 jelzést küldünk, meghívja a Data::Dumper-t és a %Scan szerkezet-tartalmát a */var/log/psad/scan_hash.$$* fájlba írja, ahol a *$$* az adott psad-folyamat programazonosítóját jelenti. A %Scan szerkezet a psad központi adatszerkezete, az összes vizsgálati adatot tartalmazza, így kiíratása a vizsgált adatok rögzítésére és hibakeresésre egyaránt jó, kiterjesztve általa a psad képességkészletét. Jelenleg a psad is még fejlesztés alatt áll, és jó pár dolog szerepel a tervlistáján – ugyanakkor tevékenyen fejlesztik, és rövid időközönként frissül.

IP Chains vagy IP Tables napló?

Az IP Chains és IP Tables tűzfalak közötti különbség bemutatásához először is hasonlítsuk össze az Nmap XMAS-vizsgálat által készült naplókat.

IP Chains

A 2. listában (24. CD Magazin/Gyanus könyvtárban) látható IP Chains-üzenetek a 79 és 81 közötti TCP-kapuk Nmap XMAS-vizsgálatával készültek. Emlékezzünk, az XMAS-vizsgálat beállítja a FIN-, URG- és PSH-jelzőket. Előbb az nmap parancs és kimenete látható, ezt követi a megfelelő IP Chains-kimenet. Figyeljük meg, hogy az IP Chains egyáltalán nem említi, mely TCP-jelzők voltak beállítva.

IP Tables

Most ugyanezt a Nmap-vizsgálatot végezzük el (az nmap parancssor és kimenet azonos a fenti IP Chains példában találhatóval, így azt nem ismételjük meg), és megjelenítettük a megfelelő IP Tables-kimenetet (*3. lista*). Ebben az esetben a vizsgálatnál használt csomagokban tisztán láthatjuk a beállított FIN-, URG- és PSH-jelzőket.

Michael Rash
(mbr@cipherdyne) vezető biztonsági mérnökként dolgozik egy ASP-nél a marylandi Annapolisban. A marylandi egyetemen alkalmazott matematikából szerzett diplomát, és 1998 óta bütyköl Linuxon.

Kapcsolódó címek

A 2.2.x Linux rendszermagsorozathoz is léteznek olyan foltok, amelyek az IP Chains tűzfalkódot teljes körű TCP-jelző naplózási képességekkel ruházzák fel. Példákért nézzük át a Linuxmag irattárát 2000/12/01-től 2000-12/07-ig, amely a ➲ http://www.uwsg.indiana.edu/hypermail/linux/kernel/0012.0/index.html címen érhető el. Ebben a tárban találunk egy „[PATCH] IP Chains log will show all flags" című témát, amely a forráskód *diff*-et tartalmazza a *linux-2.2.x/net/ipv4/ip_fw.c* fájlhoz.

A legjobb hely, ahol első kézből juthatunk TCP-vel kapcsolatos adatokhoz: RFC: 793 -Transmission Control Protocol, a ➲ http://www.ibiblio.org/pub/docs/rfc/rfc793.txt címen.

A 2. és a 4. lista a ➲ 24. CD Magazin/adatforgalom könyvtárban található.

A psad tervlistájában jelenleg szereplő célkitűzések: IP Filter-támogatás a BSD-felületeken; a legfontosabb psad-összetevők újraírása C-ben a jobb hatásfok elérése érdekében; ICMP-támogatás; jobb aláírás-meghatározás, hogy több IP-, UDP-, TCP-fejlécmezőt is be lehessen venni; illetve a rendszer beépítése a Bastille Linuxba.
➲ http://www.bastille-linux.org.

A NIS és az NFS (1. rész)

A központi felhasználókezelés két segédeszköze

Két hárombetűs, bűvös rövidítés több ponton is hasonlít egymásra. Az első és legszembetűnőbb apróságon túllépve (miszerint ugyanazzal a betűvel kezdődnek), észreveheted, hogy mind a kettő mögött ugyanaz a cég áll, név szerint a Sun. Továbbá mindkettő ugyanazt a célt hivatott szolgálni: adatokat oszthatsz meg velük a hálózaton. Ha már hallottál róluk, most biztosan morogsz, amiért együtt említem őket, hiszen úgy vélheted, semmi közük egymáshoz. Igaz: a két rendszernél az állományok megosztásának módja nem ugyanolyan, viszont remélem, e kétrészes cikk végére kiderül, milyen remekül kiegészíti egymást a két rendszer. Vágjunk bele!

Mire jó a NIS?

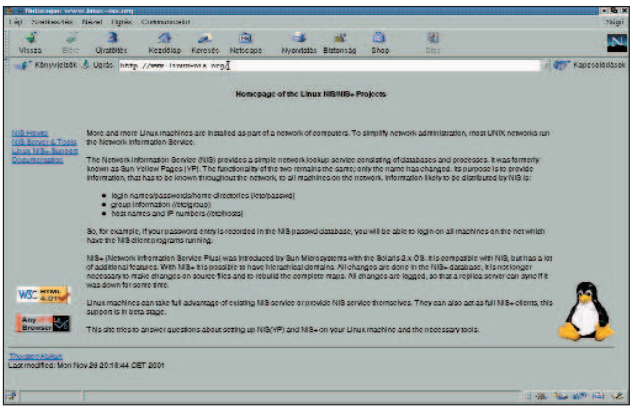
A NIS (Network Information Service) arra szolgál, hogy dbm formátumú adatbázisokat oszthass meg számítógépek között. Egy ilyen adatbázis rendkívül egyszerű felépítésű. Minden rekord egy kulcs- és egy értékpárból áll. Ezeket az adatbázisokat egyszerű szöveges állományokból készítheted a csomaghoz mellékelt segédprogrammal, a `makedbm`-mel. Ahhoz, hogy eloszlassam a témát körüllengő homályt, ismertetek egy egyszerű példát. Tegyük fel, létezik egy linuxos hálód, ahol azt szeretnéd, ha a felhasználókat központilag lehetne felügyelni. Örülnél, ha nem kellene az összes gép */etc/passwd* állományát módosítanod, amennyiben egy új felhasználót fel szeretnél venni a rendszerbe. Ebben az esetben a NIS a megoldás. Csak a központi gépen veszed fel a */etc/passwd*-be a felhasználókat, elkészítesz egy adatbázist, majd a NIS-sel megosztod. Ilyen egyszerű!

NIS, NIS+ és YP

Elnézést kérek mindenkitől a sok rövidítésért, tudom, hogy ezeket fel kell dolgozni, de egyedül ilyen „csúnya” szavakkal van tele az összes fellelhető leírás. Mivel ez a cikk nem lépheti túl a szabott keretet, kénytelen leszel hozzászokni ezekhez a „mozaikszavakhoz”. A NIS és az YP ugyanaz. Az YP (Yellow Pages – Sárga oldalak) a régebben használatos kifejezés. Mindaddig így hívták a NIS-t, amíg a British Telecom be nem jegyeztette az YP-t. A NIS+ pedig a NIS egy továbbfejlesztett változata. Minden megtalálható benne, amit a NIS-ből valaha is hiányolhattál: titkosított fájlátvitel, a faszerkezeteknek köszönhetően a nagy hálózatok rugalmas és megbízható kezelése. Akad azonban egy kis gond is a NIS+-szal: kiszolgálórésze Linux alá még csak próbaváltozatban létezik, így közel sem mondható üzembiztosnak. Másrészt a NISHOWTO is azt ajánlja, hogy a borzalmas kiszolgálóoldali beállítgatás gyötrelmeitől lehetőleg kíméljük meg magunkat, és használjunk NIS-t, ahol csak lehet.

Az RPC

Ha még nem barátkoztál meg a `portmap`-pal, itt az ideje, hiszen ez a NIS egyik alapköve. Az RPC (Remote Procedure Call) egy C-programozói könyvtár, amely lehetővé teszi, hogy egy folyamat egy függvényt egy másik számítógépen hívjon meg. Amikor az RPC-ről beszélünk, legtöbbször a Sun-féle



➲ http://www.linux-nis.org/

megvalósításra gondolunk, más néven a sunrpc-re. Amennyiben akad olyan kiszolgáló, amely távoli függvényhívást tesz lehetővé, először a `portmap`-hez fordul, és „elmondja”, hogy milyen számon és melyik kapun érhető el. Amikor az ügyfél kapcsolódik, a `portmap`-tól tudja meg, hogy az adott számú kiszolgáló éppen melyik kapun csücsül. Az egyetlen állandó kapu így a `portmap`-é, ami a 111-es, az összes RPC-kiszolgáló kapuja változhat. Összefoglalva: a `portmap` nem tesz mást, mint hogy azokat az egyedi számokat, amelyeken a függvények elérhetők, kapuszámokra fordítja. Ez a kialakítás természetesen azt is jelenti, hogy a `portmap`-nak kell legelőször elindulnia – még mielőtt bármilyen RPC-kiszolgálót indíthatnál a gépeden. Debian alatt például futási szinttől függetlenül a rendszerindítás után önműködően elindul a `portmap`. Amennyiben a `portmap` meghal, az összes RPC-kiszolgálót újra kell indítani. Meg kell jegyeznem, hogy velem ez még sosem fordult elő, de ki tudja, mit hoz a jövő. A biztonságmániásoknak jegyezném meg, hogy a `portmap` használja a `tcp_wrapper` függvénykönyvtárat, így a */etc/hosts.allow*, illetve a */etc/hosts.deny* állományokkal kényelmesen behatárolható azoknak a gépeknek a köre, amelyeknek engedélyezett a használat (lásd még: portmap(8)).

A NIS-tartományok

Egy NIS-re épülő hálózat több tartományból áll, de legalább egy szükséges. Ezt a tartományt még véletlenül sem szabad összetéveszteni a DNS-sel. Amennyiben a hálózatodon NIS és `named` is található, tartománynévként mindenképpen valamilyen a DNS-től különböző nevet válassz, ezzel is megnehezítve az esetleges külső behatolók dolgát. A tartománynevet a `domainname` paranccsal bármikor lekérdezheted és beállíthatod (lásd később). Ebben a tartományban lennie kell legalább egy úgynevezett fő (master) kiszolgálónak, ami a megosztandó adatbázisokat tartalmazza. Ezt tükrözheti egy vagy több alkiszolgáló (slave), amelyek a fő gép adatbázisairól mindig pillanatnyi másolatot tartanak fenn, és ha nagyon leterhelt lenne, ők is kiszolgálhatják az ügyfeleket.

A kiszolgálói oldal

Az ➲ ftp.kernel.org/pub/linux/utils/net/NIS címen a NIS-hez nemcsak a kiszolgálóalkalmazásokat, hanem mindenféle segédeszközt fellelhetsz. Jelenleg kétféle kiszolgáló érhető el, az egyik az `ypserv`, a másik az `yps`. Az yps nem tesz lehetővé fő- és alkiépítést, és hosszabb ideje már nem is támogatott. Debian alatt minden NIS-eszköz (kiszolgáló és ügyfél egyaránt) egy csomagban érhető el (mondanom sem kell, a csomag csak az `ypserv`-et tartalmazza, az `yps`-t nem). Ha tehát Debiant használsz, meglepően egyszerű dolgod lesz: telepítés közben csak a használandó tartománynevet kérdezi meg, minden mást önműködően beállít. Létrehozza a */var/yp* könyvtárat, amin belül egy a tartománynévvel megegyező nevű könyvtárban létrehozza az adatbázisokat. Ezt kézzel a `/usr/lib/yp/ypinit -m` parancs kiadásával érheted el. További alkiszolgálókat később az alkiszolgálón kiadott `/usr/lib/ypinit -s f1` paranccsal hozhatsz létre, ahol a `f1` helyén értelemszerűen egy név vagy IP-cím áll. Debian alatt a telepítő elkészíti még a */var/yp* alatt a *Makefile*-t, így ha az adatbázisokat újból létre akarod hozni, csupán rendszergazdaként ki kell adnod a `make` parancsot a */var/yp*-ben. A */etc/passwd* állomány végén a NIS-re vonatkozó bejegyzéseknek egy + (plusz) vagy egy – (mínusz) jellel kell kezdődniük. Így ha el akarod érni, hogy Alice távolról ne tudjon belépni, csak annyit kell tenned, hogy a */etc/passwd* fájl végére odaírod: `-alice:::::::`. Az üres mezők a „bármit” jelentik. Egy fontos megjegyzés: szokványos esetben mind a kiszolgálónak (ypserv), mind az ügyfélnek (ypbind) létrejön egy-egy parancsállomány a */etc/init.d*-ben. A Debian viszont megfordítja a dolgot: van egy */etc/init.d/nis*, amelynek első sora a `NISSERVER=false`. Ez az ügyféloldali beállítás, ezért ezt írd át masterre. Ezután az a legjobb, ha újraindítod a számítógépedet. Amennyiben ellenőrizni kívánod, hogy az `ypserv` (amely RPC-kiszolgáló) rendesen fut-e, írd be a következőt:

```
# rpcinfo -u localhost ypserv
```

ekkor valami az alábbihoz hasonlót kell látnod:

```
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

Lásd még: ypserv(8), ypserv.conf(5), rpcinfo(8).

Az ügyféloldal

Ügyfelet még ennél is egyszerűbben telepíthetsz. Az ügyfélprogramot, az `ypbind`-et – a könnyebb hibakeresés végett – érdemes a mestergépre is feltelepíteni és elindítani rajta. Ez Debian alatt ugyancsak alapértelmezés szerinti beállítás. Az ügyfelet a fentebb említett FTP-helyen vagy a már megismert NIS-csomag részeként érheted el. Amint az ypbind-et elindítod, megpróbálja megtalálni a kiszolgálót. Fontos, hogy a `NIS domain` helyesen legyen beállítva. Ellenőrizd a `domainname` paranccsal, és amennyiben szükséges, állítsd be. Az `ypbind` a NIS-kiszolgáló nevét először a */etc/yp.conf* állományban keresi; ha itt nem találja vagy nem éri el a felsorolt kiszolgálókat, üzenetszórásba kezd. Ezzel a módszerrel eltarthat egy darabig, amíg ráakad a kiszolgálóra, így ha már mindened működik, feltétlenül állítsd be itt a NIS-kiszolgálódat. Attól kezdve, hogy a kapcsolat létrejött, folyamatos lesz a kiszolgáló és az ügyfél között. És most jöjjön néhány fontosabb ügyféloldali alkalmazás, amelyek jól jöhetnek:

ypwhich

A segítségével megtalálható egy számítógép NIS-kiszolgálója vagy egy adott adatbázishoz tartozó mesterkiszolgáló, például:

```
$ ypwhich
turorudi.home
$ ypwhich -m passwd.byname
turorudi
```

Az első lekérdezés a használt NIS-kiszolgálóra vonatkozott, itt láthatod, hogy a teljes DNS-nevet visszakaptam. A második pedig a megosztott *passwd* adatbázis mesterkiszolgálójára irányult.

ypcat

Kiíratja egy adatbázis összes kulcsához tartozó értéket. Fontos, hogy az adatbázis megadásakor beceneveket is használhatsz. A fenti példában látott `passwd.byname` helyett most elég csak `passwd`-t írni. A teljes nevekhez tartozó becenevek listáját az `ypcat -x` paranccsal kapod meg.

```
$ ypcat passwd
identd:x:100:65534::/var/run/identd:/bin/false
nobody:x:65534:65534:nobody:/home:/bin/sh
telnetd:x:101:101::/usr/lib/telnetd:/bin/false
balazs:x:1000:100::/home/balazs:/bin/bash
```

ypmatch

A megadott kulcsokhoz tartozó értékeket adja vissza az adatbázisban – itt szintén használhatsz beceneveket.

```
$ ypmatch telnetd balazs passwd
telnetd:x:101:101::/usr/lib/telnetd:/bin/false
balazs:x:1000:100::/home/balazs:/bin/bash
```

Az nsswitch.conf

A */etc/nsswitch.conf* az egyes szolgáltatások esetében azt a sorrendet tartalmazza, ahogyan az adatokat le kell kérdezni, például a

```
hosts:      files dns
```

bejegyzés azt jelenti, hogy egy névhez tartozó IP-cím keresésekor először a */etc/hosts* állományt kell megnézni, és ha a bejegyzés nem szerepel, jöhet a DNS. Számunkra most csak annyi fontos, hogy az első három sor így nézzen ki:

```
passwd:         compat
group:          compat
shadow:         compat
```

Bővebben: nsswitch.conf(5)

Összegzés

Ugyanazzal a felhasználói névvel és jelszóval most már bármelyik gépen be tudunk lépni. No, de mi lesz a saját könyvtárunkkal? Ez még most sem érhető el! A következő részben ezt taglaljuk részletesebben.

Fülöp Balázs
(xut@freemail.hu) 17 éves, 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.

Átállás

Egy kiszolgáló átállításának nehézségei

Sokszor előfordul, hogy a vállalat egyik kiszolgálója eléri azt a kort, amikor már le kell cserélni. Három éve becsülettel szolgáló belső gépünk nálunk is eljutott erre a pontra. E cikkben áttekintést kívánok adni – elsősorban nem szakmai szempontból –, hogy egy ilyen átállás miféle módon zajlik.

A helyzet egyszerű: a központi kiszolgáló elavult, egy-két alkatrésze a végkimerülés szélén áll, az operációs rendszere esetleg olyan régi, hogy már egyszerűbb nulláról újratelepíteni, mint frissíteni. A vállalat eldönti: beruház egy új gépre, a rendszergazda megkapja a feladatot: itt a pénz, holnap kérem az új kiszolgálót! Amennyiben te is ilyen helyzetbe kerültél, csendben fordulj meg, ballagj el ezért a cikkért, majd dugd a főnök orra alá. Ez ugyanis nem ilyen egyszerű! Egy kiszolgáló helyett újat beállítani csak egy része a dolognak, ilyenkor számos járulékos kérdés és gond szokott a felszínre kerülni. Nézzük csak végig a jellemzőbbeket!

Egy kicsi vagy közepes méretű vállalatnál az alábbi fejlődés a jellemző: legelőször egy-két gép van. Egy idő után a gépeket (régebben koaxos, manapság már egyértelműen csavart érpáras) hálózattal összekötik. Később az egyik gépbe kerül egy modem, megszületett az internetkapcsolat, sőt, az adott gép faxok küldésére is alkalmas. Később rájönnek, hogy a munkagépek gyakran megfáradnak, ezért szükséges egy gép, amely jóval megbízhatóbb a többinél. Négy-öt gép felett kiderül: az volna a legáldásosabb, hogy ez a gép már ne legyen munkaállomás is egyben. Később az internetkapcsolatot meg kívánják osztani, majd az első komoly pusztítás után – amit mondjuk egy levélvírus vagy betörés okoz – komolyabb védelmi rendszerek üzembe állításával próbálkoznak meg. Emellett felmerül az igény a hálózatos faxküldésre, mindenki külön levélcímet szeretne, a gépek száma lassan két-három tucatnyira nő, és egyszer csak előkerül az útválasztás kérdése. Eleddig elég volt az az egy vagy két 16 kapus elosztó, most már viszont tisztán lehet látni, hogy a gépparkot külön szakaszokra érdemes szétválasztani. Ez teljesítményben és biztonságban is megtérül.

Ezen a ponton a vezetőség már gondban lehet, amennyiben a dolgozók többsége nem számítógépen nevelkedett. Szükség van egy profi rendszergazdára, de egy rendszergazda sokat kér. Ha megbíznak egy külsős céget, a gondok csak lassabban oldódnak meg, ha felvesznek egy új alkalmazottat, nem biztos, hogy megfelel az igényeknek…

Tehát ez a kissé áttekinthetetlen állapot fogad minket. Ne gondold, kedves olvasó, hogy ilyen vállalat nincs. Sőt! A legtöbb vállalat eljut ebbe az állapotba, van, hogy gyorsabban, van, hogy lassabban. Rendelkezünk egy kusza hálózattal, amelyben mindenféle ügyfélgép szerepel, egy-kettő kiszolgálóként működik, némelyikben van faxkártya, némelyik meg is osztja szolgáltatásait, a nyomtatók szanaszét helyezkednek el a hálózaton belül, majd a kegyelemdöfés: az egyik gépen bekopog egy vírus, mondjuk a Nimda.

A markunkban ott a pénz, de még mielőtt elköltenénk, gondolkozzunk, mire is érdemes. Először is gondoljuk végig, hogy a hálózat szerkezete (topológiája) megfelelő-e. Ha például három külön részleg van a cégnél, amelyeknek egymással nincs sok közös dolguk, jobban járunk, ha külön szakaszokba helyezzük őket. Ez több dolgot is maga után von. Először is több hálókártya kell az útválasztást végző gépbe, és mindegyik szakaszhoz külön elosztót vagy jelismétlőt kell üzembe helyezni. Ebből is érdemes olyat, amely a jelenlegi terhelést félvállról veszi – tervezve a leendő növekedéssel. Elképzelhető, hogy a kábelezésnél is szükséges lesz dolgozni. Kialakítunk tehát három külön szakaszt egy-egy elosztóval (eszünkbe ne jusson a világszerte rettegett ***Gagyi*** cég termékeit használni!), és az útválasztóba legalább három hálókártyát helyezünk. Kis cégeknél az útválasztó feladatát az új kiszolgáló vígan ellátja majd.

Most nézzük végig, még milyen feladatokat lát el általában egy ilyen gép. Központi fájlkiszolgáló, természetesen Samba-alapokon. Az internetkapcsolatot kezeli (a munkagépekből a modemeket az utolsó szálig ki kell imádkozni), ezzel együtt a levelezést is. Ehhez kapcsolódóan gyorstáraz és névfeloldást is végez. Esetleg központilag tárolhatja a dolgozók profiljait is, amennyiben erre van igény. Ha a modemeket kikönyörögtük, akkor joggal követelik a hálózati faxolást, főleg kifelé, de a bejövő faxokat is könnyedén kezelhetjük. A nyomtatás külön történet, valószínűleg ezt is gatyába kell ráznunk.

Most tervezzük meg az átállást – bár hiába tervezzük meg, előre szólok:

a, menet közben derül ki a legfontosabb, amire nem is gondoltunk,
b, a dolgozók a hajukat tépik majd és a pokolra kívánnak,
c, emellett újra előkerül egy még aljasabb, gyorsabban terjedő vírus.

De azért csak tervezzünk!

A kábelek rendberakása és az elosztók bekötése után a központi gépet kell összeraknunk. Ez a rendszergazda dol-

Átállás

Egy kiszolgáló átállításának nehézségei

Sokszor előfordul, hogy a vállalat egyik kiszolgálója eléri azt a kort, amikor már le kell cserélni. Három éve becsülettel szolgáló belső gépünk nálunk is eljutott erre a pontra. E cikkben áttekintést kívánok adni – elsősorban nem szakmai szempontból –, hogy egy ilyen átállás miféle módon zajlik.

A helyzet egyszerű: a központi kiszolgáló elavult, egy-két alkatrésze a végkimerülés szélén áll, az operációs rendszere esetleg olyan régi, hogy már egyszerűbb nulláról újratelepíteni, mint frissíteni. A vállalat eldönti: beruház egy új gépre, a rendszergazda megkapja a feladatot: itt a pénz, holnap kérem az új kiszolgálót! Amennyiben te is ilyen helyzetbe kerültél, csendben fordulj meg, ballagj el ezért a cikkért, majd dugd a főnök orra alá. Ez ugyanis nem ilyen egyszerű! Egy kiszolgáló helyett újat beállítani csak egy része a dolognak, ilyenkor számos járulékos kérdés és gond szokott a felszínre kerülni. Nézzük csak végig a jellemzőbbeket!

Egy kicsi vagy közepes méretű vállalatnál az alábbi fejlődés a jellemző: legelőször egy-két gép van. Egy idő után a gépeket (régebben koaxos, manapság már egyértelműen csavart érpáras) hálózattal összekötik. Később az egyik gépbe kerül egy modem, megszületett az internetkapcsolat, sőt, az adott gép faxok küldésére is alkalmas. Később rájönnek, hogy a munkagépek gyakran megfáradnak, ezért szükséges egy gép, amely jóval megbízhatóbb a többinél. Négy-öt gép felett kiderül: az volna a legáldásosabb, hogy ez a gép már ne legyen munkaállomás is egyben. Később az internetkapcsolatot meg kívánják osztani, majd az első komoly pusztítás után – amit mondjuk egy levélvírus vagy betörés okoz – komolyabb védelmi rendszerek üzembe állításával próbálkoznak meg. Emellett felmerül az igény a hálózatos faxküldésre, mindenki külön levélcímet szeretne, a gépek száma lassan két-három tucatnyira nő, és egyszer csak előkerül az útválasztás kérdése. Eleddig elég volt az az egy vagy két 16 kapus elosztó, most már viszont tisztán lehet látni, hogy a gépparkot külön szakaszokra érdemes szétválasztani. Ez teljesítményben és biztonságban is megtérül.

Ezen a ponton a vezetőség már gondban lehet, amennyiben a dolgozók többsége nem számítógépen nevelkedett. Szükség van egy profi rendszergazdára, de egy rendszergazda sokat kér. Ha megbíznak egy külsős céget, a gondok csak lassabban oldódnak meg, ha felvesznek egy új alkalmazottat, nem biztos, hogy megfelel az igényeknek…

Tehát ez a kissé áttekinthetetlen állapot fogad minket. Ne gondold, kedves olvasó, hogy ilyen vállalat nincs. Sőt! A legtöbb vállalat eljut ebbe az állapotba, van, hogy gyorsabban, van, hogy lassabban. Rendelkezünk egy kusza hálózattal, amelyben mindenféle ügyfélgép szerepel, egy-kettő kiszolgálóként működik, némelyikben van faxkártya, némelyik meg is osztja szolgáltatásait, a nyomtatók szanaszét helyezkednek el a hálózaton belül, majd a kegyelemdöfés: az egyik gépen bekopog egy vírus, mondjuk a Nimda.

A markunkban ott a pénz, de még mielőtt elköltenénk, gondolkozzunk, mire is érdemes. Először is gondoljuk végig, hogy a hálózat szerkezete (topológiája) megfelelő-e. Ha például három külön részleg van a cégnél, amelyeknek egymással nincs sok közös dolguk, jobban járunk, ha külön szakaszokba helyezzük őket. Ez több dolgot is maga után von. Először is több hálókártya kell az útválasztást végző gépbe, és mindegyik szakaszhoz külön elosztót vagy jelismétlőt kell üzembe helyezni. Ebből is érdemes olyat, amely a jelenlegi terhelést félvállról veszi – tervezve a leendő növekedéssel. Elképzelhető, hogy a kábelezésnél is szükséges lesz dolgozni. Kialakítunk tehát három külön szakaszt egy-egy elosztóval (eszünkbe ne jusson a világszerte rettegett ***Gagyi*** cég termékeit használni!), és az útválasztóba legalább három hálókártyát helyezünk. Kis cégeknél az útválasztó feladatát az új kiszolgáló vígan ellátja majd.

Most nézzük végig, még milyen feladatokat lát el általában egy ilyen gép. Központi fájlkiszolgáló, természetesen Samba-alapokon. Az internetkapcsolatot kezeli (a munkagépekből a modemeket az utolsó szálig ki kell imádkozni), ezzel együtt a levelezést is. Ehhez kapcsolódóan gyorstáraz és névfeloldást is végez. Esetleg központilag tárolhatja a dolgozók profiljait is, amennyiben erre van igény. Ha a modemeket kikönyörögtük, akkor joggal követelik a hálózati faxolást, főleg kifelé, de a bejövő faxokat is könnyedén kezelhetjük. A nyomtatás külön történet, valószínűleg ezt is gatyába kell ráznunk.

Most tervezzük meg az átállást – bár hiába tervezzük meg, előre szólok:

a, menet közben derül ki a legfontosabb, amire nem is gondoltunk,
b, a dolgozók a hajukat tépik majd és a pokolra kívánnak,
c, emellett újra előkerül egy még aljasabb, gyorsabban terjedő vírus.

De azért csak tervezzünk!

A kábelek rendberakása és az elosztók bekötése után a központi gépet kell összeraknunk. Ez a rendszergazda dol-

ga, egy ilyen gépet véletlenül se bízzunk olyan emberre, akinek nincs kellő tapasztalata, és nagyon fontos, hogy először élesszük fel rajta a szolgáltatásokat, alakítsuk ki a biztonsági rendet, működjön rendesen a levelezés, a fax, a fájlkiszolgálás, a nyomtatás, az internetkapcsolat és az útválasztás. Csak a szolgáltatások üzembiztos működése után kezdjük meg az átállást (nyugi, utána is lesz elég bajunk).

Az új rendszer üzembe helyezése szinte biztos, hogy az összes munkagép átállítását magával vonja (még ha DHCP-t használunk, akkor is elképzelhető). Számoljunk elég idővel, hiszen vagy minden felhasználónak végig kell mesélnünk a történetet, vagy készítenünk kell egy leírást, amelyben részletezzük, hogy mi és miért történik. A levélírással hatnyolc ügyfélgép fölött biztos, hogy időt takarítunk meg. Ha ügyesek vagyunk és bízunk a felhasználókban, a leírásban azt is leírhatjuk, hogy miként tudják saját maguk elősegíteni az átállást, és hogy milyen lépésekben és határidőkkel történik meg a váltás, valamint hogy milyen úton jelezzék, ha különleges igényük van.

Most jön a legfárasztóbb rész: a szakaszokat egyesével átállítjuk, az ügyfeleknél költözés, majd a régi szolgáltatások tiltása az adott szakaszra. Hogy miért fontos tiltani a régi szolgáltatásokat? Hogy nehogy nagyot koppanjunk, amikor (pár nap vagy hét) múlva nagy nyugalommal ízekre szedjük a régi gépet, a dolgozók pedig dörömbölnek, hogy nem tudnak dolgozni. Miután az első kis csoport átköltözött, adjunk nekik időt, hogy minden szolgáltatást élesben is kipróbáljanak. Ez órákat, napokat, de előfordulhat, hogy heteket jelent. Miután az első szakasz átállt és gond nélkül dolgoznak az új gépen, jön a tisztogatás. Ehhez szükséged lesz egy csavarhúzóra, kötélidegzetre és a főnök előzetes meggyőzésére. Ugyanis most jön, hogy az új szakasz összes gépéből ki kell követelni a modemeket, majd mindenkinek el kell magyarázni, hogy csak a központi gépen tárolja a fontos adatokat (mert az új rendszer legalábbis tükrözött lemezekkel dolgozik), hogy ne merjen hülyeségeket telepíteni (ismerek olyat is, aki a hajlékonylemezes és a CD-meghajtót is kiszereli), hogy véletlenül se osszon meg semmit a gépből (jellemzően a C: van megosztva, jelszó nélkül), hogy milyen levelezőt használjon és így tovább. Ez az a pont, amikor a rendszergazda még a főnökség tagjait is megelőzi a körözési listákon. Ezután csendben a háttérbe vonulunk és figyelünk. Ha minden felmerült kérdést megoldottunk, túl vagyunk a nehezén. Most már csupán ugyanezt kell végigzongoráznunk az összes szakaszon.

Miután az összes gép átállt és a régi kiszolgáló elérését mindenhonnan letiltottuk, még érdemes egy ideig – a biztonság kedvéért – működőképes állapotban tartani, és később, amikor már senki sem keresi, ráérünk szétszedni.

Igyekeztem röviden összefoglalni egy átállás menetét, természetesen minden környezet más és más, lehet, hogy több kiszolgáló van, előfordulhat, hogy az új gép csak különböző feladatokat vesz át, vagy a munkagépeket egytől egyig újra kell telepíteni… a változatok száma végtelen. Egy viszont biztos, ha alaposan átgondoljuk az átállást, nagyon sok keserűségtől kíméljük meg magunkat. És még egy gondolat: vegyünk legalább egy jó könyvet, ami a biztonság alapjaival foglalkozik és az összes felhasználóval olvastassuk el! Könnyebb a dolgunk, ha nem nekünk kell ötvenszer elmagyarázni, hogy miért ne használjon a munkatárs olyan levelezőt, ami önműködően elindítja a levelekben lévő vírusokat!

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

Hazai linuxos cégek gyűjteménye

Sokszor keresnek meg minket, hogy ismerünk-e jó rendszerfelügyelő céget, tudunk-e valakit, aki vállalja egy kiszolgáló telepítését, hol lehet linuxos tanfolyamokra jelentkezni, kihez fordulhatnak ügyesbajos kérdéseikkel stb. Mivel mi magunk sem ismerjük a hazai linuxos közösség minden tagját, arra gondoltunk, hogy indítunk egy sorozatot, amelyben mindig két-három, az adott területen dolgozó céget mutatunk be – a hálózatépítőtől a rendszerfelügyelőn át a programozóig. Igen ám, de ha ilyen rendszert szeretnénk indítani, jó volna, ha a cégeket egy pontos rendszerbe tudnánk beilleszteni. E rendszer kialakításához kérem most a segítségeteket.

Mit érdemes, és mit kötelező leírni egy ilyen cégről? Természetesen fontos, hogy a cég mikor alakult, hogy mióta foglalkozik Linuxszal, milyen típusú rendszerekkel dolgozik, hol található a székhelye, milyen ügyfélkört céloz meg. Emellett szerintem rendkívül fontos, hogy az adott cégen belül hány munkatárs ért az adott feladathoz, például egy rendszerfelügyeletet vállaló cégnél hány rendszergazda dolgozik, a cég mennyire van leterhelve, vállal-e 24 órás vagy hétvégi ügyeletet, gond esetén milyen határidővel dolgoznak, és természetesen milyen árkategóriában.

Az ár nehéz kérdés. Lehetetlen megmondani, hogy valaki mennyiért telepít egy rendszert, amíg nem tudja pontosan, mik is az igények. Az sem mindegy, hogy az adott munka elvégzése után hogyan adja át a készterméket, például egy gép szállítása után a cég vállalja-e, hogy amennyiben az általa összerakott gép valamelyik alkatrésze nem (vagy nem megfelelően) működik Linux alatt, azt kicseréli? Emellett egy vezető számára az is fontos, hogy a cég referenciákat tudjon adni. Sőt, az volna a legjobb, ha néhányuk véleményét ki lehetne kérni. Szeretném, ha létre tudnék hozni egy olyan listát, amely alapján az érdeklődő könnyen ki tudna választani mondjuk egy megbízható, elfogadható áron dolgozó rendszerfelügyelettel foglalkozó céget, vagy például egy komoly tapasztalatokkal rendelkező hálózatépítő társaságot.

Mivel mi ebben csak katalizátorszerepet játszhatunk, kérek mindenkit, írja meg nekem véleményét, a gondolatait, hogy milyen cégekre kíváncsi, milyen kérdéseket tenne fel, milyen szerkezetben látná örömmel a listát. Egyúttal ez felhívás a hazai linuxos cégek és vállalkozók felé is! Várjuk olyan társaságok és vállalkozók jelentkezését, akik ki mernek állni szolgáltatásaikkal a nagyvilág elé – vállalva a megmérettetést!

Szy György

A szerkesztők és az Emacs (1. rész)

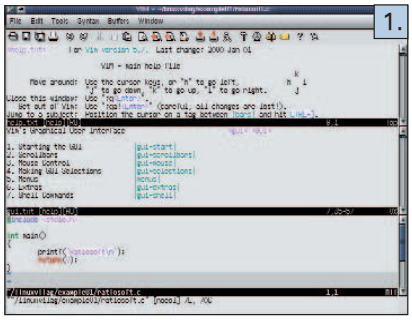
Ezt az írást nem a Linux-rendszert és a programozást jól ismerőknek szánom, hanem azoknak, akik mindkettővel most ismerkednek.

Elsőként a forráskód megírásához használható szerkesztőket mutatom be a teljesség igénye nélkül, majd az Emacs fejlesztőkörnyezettel foglalkozom hosszabban.

Szövegszerkesztők

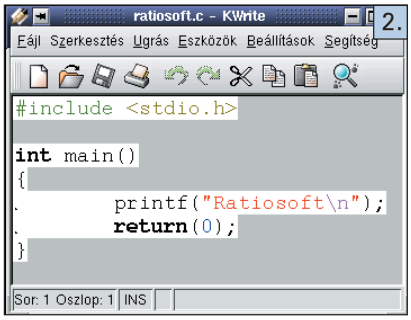
A programokat először meg kell írni, amihez kell egy nekünk tetsző egyszerű vagy kevésbé egyszerű szövegszerkesztő, amely ASCII formátumú állományt hoz létre. Ebből következik, hogy haszontalan lenne az OpenOffice nagy tudású szerkesztőjét használni, mert fölöslegesen sok helyet foglal el a memóriában, és semmiképpen sem tudjuk kihasználni a képességeit, hiszen a programírásban nem segítenek a kifinomult formázási lehetőségek, nem célszerű sokféle betűkészletet használni, és számos programozót zavar, ha változó szélességű és nagyságú betűket lát maga előtt bogarászás közben. Ne feledjük, hogy bármilyen szépre is formázzuk meg forráskódunkat, a formázó karakterek abban a pillanatban nyom nélkül eltűnnek, amikor forráskódunkat ASCII formátumban kimentjük, a fordítók pedig csak a formázatlan állományokat hajlandók elfogadni!

A fentiekből következik, hogy a merészebbek programozásra is használhatják a legendás hírű ***vi*** szerkesztőt. Erről a kezdőknek azt illik tudni, hogy a kellően tájékozottak „viáj”-nak ejtik, és hogy a `:q!` paranccsal lehet kilépni belőle. Ez nem segít mindig, ilyenkor nyomjuk meg az Esc billentyűt, és próbálkozzunk újra. Ha már fel vagyunk vértezve ennyi tudással, akkor legalább nem kell újraindítani az egész rendszert, hogy visszakaphassuk a parancssor készenléti jelét, ahogy azt egyik ismerősöm tette a `vi`-t próbálgatva. A mindenre elszántak próbálkozhatnak a `vi` többalakos változatával, az `xvi` (ejtsd: „eksz-vi-áj”) szerkesztővel, vagy az `xvi`-nál is fejlettebb `vim`-mel. A `vim` a „vi improved” (tökéletesített `vi`) szavak rövidítése. A `vi`-nak az elmúlt évtizedekben számos mutációja született, és ha telepítve vannak a gépünkön, kipróbálhatjuk őket. Írjuk be a parancssoron az `ed`, `ex`, `gex`, `vi`, `gvi`, `view`, `gview`, `vim`, `gvim`, `rvim`, `rview`, `rgvim` vagy `rgview`, `elvis`, `nvi` parancsokat és az olvasni vagy szerkeszteni kívánt állomány nevét! Ezek a parancsok többnyire ugyanazt a végrehajtható állományt indítják el, de különböző bővítményekkel vagy éppen korlátozásokkal. Például a `view` vagy a view-ra végződő parancsok használatakor a megnyitott fájlt csak olvasni lehet, a parancsok elején lévő r betű pedig a „restriction”, azaz megszorítás szóra utal. A korlátozott parancsok valamilyen

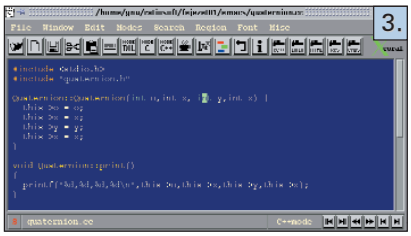

1.

módon szűkítik a `vi` lehetőségeit, például nem indíthatunk parancsokat a `vi`-ból, vagy nem függeszthetjük fel ideiglenesen a működését. A `gvim` (*1. kép*) a Windowsban megszokott menüket és ikonokat varázsolja elénk, és viszonylag jó leírással bír. Első pillanattól otthon fogjuk érezni magunkat benne, ha a telepítés után sikerül elindítanunk.

Ha nem, próbáljuk meg a *~/.gvimrc* fájlt testreszabni. A `vi` általában nem alkalmas bináris állományok szerkesztésére, ehhez inkább a `hex`, `vche`, `vche-nc`, `vche-raw`, `khexedit` vagy a `ghex` programokat használjuk. Szintén kéznél van a Midnight Commander belső fájlnézegetője, amely hexadecimális szerkesztő is egyben. A `vimtutor` program végigvezet bennünket a legfontosabb `vi`-parancsokon. Többek közt megtudhatjuk, hogy mely két üzemmód létezik a `vi`-ban, hogyan indíthatjuk el, hogyan mozgathatjuk a kurzort, hogyan szerkeszthetjük a fájlt, és miképpen léphetünk ki belőle. A `pico` szintén parancssori szövegszerkesztő, de könnyebb megtanulni, mint a `vi`-t. Ezt már az is mutatja, hogy mindjárt induláskor kiírja, hogyan lehet kilépni belőle. Megemlítem még a ***Joe*** szerkesztőt, amit ötféle üzemmódban használhatunk. Ha a `joe` helyett a `jpico` parancsot ütjük be, akkor a Joe a most említett Pico szerkesztőt utánozza,



2.

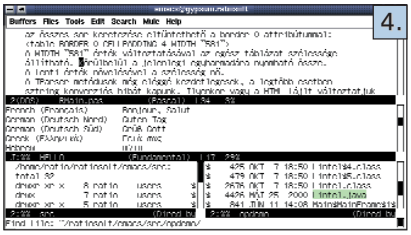

3.

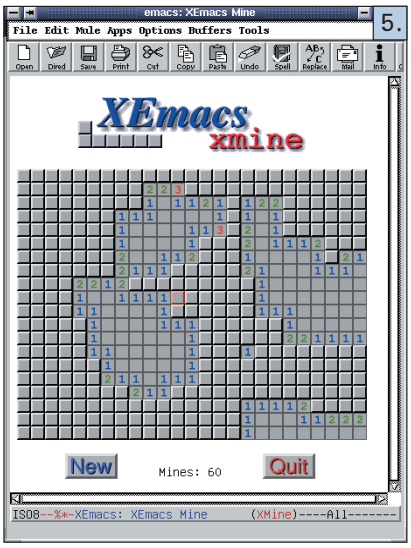
ha a `jstar` parancsot, akkor a hajdan jól ismert WordStar szövegszerkesztőt. A fenti programok mindegyike igen nagy tudású, de használatukhoz némi tapasztalatra van szükség. Az `xedit`, `kedit` és a `gedit` a Windows felől érkezők számára minden bizonnyal barátságosabbnak fognak tűnni, mint parancssori elődeik, de azt is rögtön tapasztalhatjuk, hogy ezek sem tudnak sokkal többet, mint a `notepad.exe`.

Az Emacs

A Linux-programozással foglalkozó szakkönyvek szinte kivétel nélkül a *GNU Emacs* fejlesztőkörnyezetet ajánlják. Figyeljük meg, hogy immár nem a „szerkesztő” szót használtam, hanem az IDE (Integrated Development Environment) mozaikszóra utaltam, ami magyarul összetett fejlesztőkörnyezetet jelent. Mégse gondoljunk azonban a *Linuxvilág* augusztusi számában ismertetett KDevelophoz vagy a Borland Kylixhoz hasonló RAD (Rapid Applica-

tion Development, azaz gyors alkalmazásfejlesztés) eszközökre. Az Emacs a Unix-hagyományokban gyökeredzik, nehézkes és barokkosan bonyolult. Némelyek szerint az Emacs gonosz, mások viszont három részre osztják a világban élő embereket: azokra, akik Emacsot használnak; olyanokra, akik inkább a


4.



5.

vi-t szeretik; és mindenki másra. Vannak, akik azt mondják, hogy az Emacs egy jó operációs rendszer, bár elismerik, hogy a Unixban több program van. Bonyolultságából következik, hogy hosszú időbe telik, amíg otthonosan mozoghatunk benne, és inkább azok számára ajánlom, akik naponta fogják használni, hiszen elég pár hetes kihagyás, és máris jó néhányat elfelejthetünk a számtalan billentyűkombinációból.

Az Emacs név a ***Richard M. Stallman*** által a TECO szerkesztőhöz írt „editing macros” (szerkesztőmakrók) szavak betűiből alkotott mozaikszó. Az Emacs történetét elolvashatjuk a ➲ http://www.gnu.org/philosophy/stallman-kth.html (a fájl másolata megtalálható a 24. CD Magazin/Emacs könyvtárában). A `vi`-hoz hasonlóan az Emacsnak (*4. kép*) is több változata létezik, például a ***Xemacs*** (*5. kép*), ami egy eszköztárral is rendelkezik. Az utóbbit korábban ***Lucid*** (azaz világos, érthető) ***Emacs***-nak nevezték.

Mindkét Emacs-fajta elfogadott, és gyakran együtt hivatkoznak rájuk az Emacson szóval. Ahogy az Emacs bejelentkező ablakában olvashatjuk, a GNU Emacs az egyik alkotóeleme a Linux-alapú GNU-rendszereknek, és mint ilyet egyetlen Linux-terjesztésből sem illik kihagyni. Akik GNU-programokat akarnak írni, azoknak otthonosan kell mozogniuk az Emacsban, mint ahogyan a rendszergazdáknak is ismerniük kell a `vi`-t, ami kötelezően ott van minden Unix-típusú rendszerben, és amit a legvadabb rendszerösszeomlások idején is el lehet indítani. Mindkét program elengedhetetlen kelléke a Unix-kultúrának, a Linux-életérzésnek és a GNU-mozgalomnak.

Ahogy a képeken is látjuk, az Emacs egyszerre több fájlt is meg tud nyitni – akár ugyanazon keretben több részre osztva fel a munkaterületet. Számos keretet vagy ablakot tarthatunk nyitva egyidejűleg, futtatható állományokat és képeket nézhetünk meg vele. Ha az Emacs nem támogatott képformátumot talál, akkor külső képnézőt (például ImageMagick) hív segítségül. Programozás közben intézhetjük levelezésünket, honlapokat látogathatunk meg és tölthetünk le akár PostScript formátumban is anélkül, hogy kilépnénk az Emacsból. Amikor elfáradunk, néhány beépített játékkal játszhatunk.

Az Emacsot nem a manapság igen divatos C vagy C++ nyelven írták és írják, hanem a Lisp nyelvet használják erre a feladatra, annak is egy különleges fajtáját, az Elispet, más néven Emacs Lispet. Az Elisp teljes értékű programozási környezet, amivel szöveget és fájlokat kezelhetünk, hálózati alkalmazásokat vagy új felhasználói felületeket építhetünk fel az Emacson belüli használatra. Az érdeklődők kipróbálhatják még a Jonathan Saját Emacs Változata (Jonathan's Own Version of Emacs) programot vagy annak menüsített, XJove nevű kiadását, ha beírják a `jove` vagy `xjove` parancsokat. Az `uemacs` vagy más néven MicroEmacs kis teljesítményű gépekre készült, és a 4.0-s változathoz maga ***Linus Torvalds*** írt javításokat. Utóbbit az `em` paranccsal indíthatjuk.

Az Emacs logikája

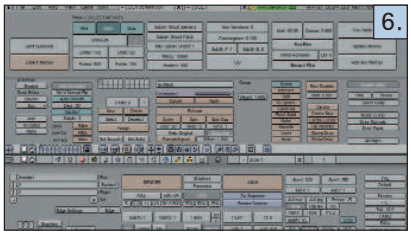
Talán igazságtalanul bántam az Emacscsal, amikor azt mondtam róla, hogy nehézkes és bonyolult, hiszen már rövid használat után beláthatjuk, hogy a fejlesztők egységes és igen egyszerű elvek alapján építették fel a programot. Ezeket az elveket így foglalhatnánk össze:

Ha a program használata közben új szempontok merülnek fel, amik megkönnyíthetik a mindennapi munkát, akkor meg kell valósítani azokat. Az új feladatok ellátásához egy vagy több Elisp-függvényt kell megírni, és ezeket a függvényeket a felhasználók számára hozzáférhetővé kell tenni, azaz meg kell adni számukra a lehetőséget, hogy programozás nélkül meghívhassák őket.

Mivel Richard Stallman és társai Unix-környezetben nevelkedtek, vérükké vált az a szemlélet, hogy egy program csak egy dolgot csináljon, de azt felettébb jól. Követelmény volt a Unixban az is, hogy ezek a programok tetszés szerint összefűzhetők legyenek. S valóban, az Emacsban ott van a mini átmeneti tár, ami lényegében egy miniatűr héj, ahová parancsokat gépelhetünk be. A mini átmeneti tárhoz hasonló parancssori beviteli eszközök eltűntek a mai szövegszerkesztőkből, ha egyáltalán voltak bennük, de a táblázatkezelőkben a szerkesztőlécek még most is megtalálhatók, annak ellenére, hogy az adatokat közvetlenül a cellákba is beírhatnánk. A Unix-követelmények szerint az Emacs-parancsok szintén kötegelhetők, és rendszerint csak egy dolgot tesznek, de azt felettébb jól, hiszen a közel két évtizedes fejlesztés következtében az Emacs igen jól átgondolt és hibamentes lett. Azt is tudjuk, hogy a programozó nem szeret sokat gépelni, annak ellenére, hogy a billentyűzetet mesteri módon tudja kezelni. Magától adódott tehát a felismerés, hogy az egyre szaporodó parancsokat billentyűkhöz és billentyűkombinációkhoz kössék. Tették ezt annál is inkább, mert majd húsz évvel ezelőtt még nem voltak ismertek a grafikus képernyők és a mutatóeszközök. A végeredmény az lett, hogy mára igen nagy számú Emacs-parancs és -változó áll a felhasználók rendelkezésére, akik így igen sok feladatot tudnak egyszerűen és hatékonyan megoldani. Cserébe viszont a kezdőknek viszonylag hosszú ideig kell ismerkedniük az Emacs-környezettel, és nemcsak a számtalan parancs nevét kell megtanulniuk, de a hozzájuk tartozó billentyűkombinációkat is készségszinten el kell sajátítaniuk.

Gondoljuk csak végig, hogy másképp van-e ez a többi, felhasználóbarátnak tartott alkalmazásban! Vajon a grafikus felületek kitalálói jobb receptet kínálnak a fenti dilemma orvoslására? Hatásos megoldás lehet, ha csökkentjük a felkínált parancsok számát – áttekinthetőbbé és könnyebben megtanulhatóvá téve a programot. Az ilyen lebutított változatok kielégítők lehetnek az átlagfelhasz-

nálók, de nem a szakemberek számára. Megpróbálkozhatunk a „szolgáltatáselrejtés” alkalmazásával, amit én a Microsoft Word 2000-ben láttam először.
A szakembereknek szánt programcsomagok viszont valósággal kérkednek az elérhető szolgáltatások százaival. Vessünk például egy pillantást a ***Blender*** nevű 3D-modellezőre, amely egyszerre több tucat nyomógombot rak ki a képernyőre, és ezek a gombok az üzemmódtól függően állandóan változnak, mindig újabbak bukkannak elő!


6.

Hiába grafikusak ezek a felületek, igazából nem lehet őket felhasználóbarátnak nevezni, hiszen előtanulmányok nélkül a zöldfülűek semmit sem tudnak kezdeni velük. Az ilyen programokat azonban az irodai programcsomagokkal ellentétben nem akarják mindenkinek eladni, beleértve az olvasni tudó kisdedeket és a homályosodó szemű aggastyánokat. A szakembereknek fejlesztő programozók számára tehát nem az a kérdés, hogy megmutassák-e a programban rejlő lehetőségeket, hanem az, hogy miképpen tegyék meg. Általánosan elfogadott módszer, hogy a szolgáltatásokat ilyen-olyan szempontok szerint csoportosítják, majd menükre, gyorsbillentyűkre, ikonsorokra vagy nyomógombokra fűzik fel őket. A párbeszédablakos megjelenítés kényelmes és felhasználóbarát, ha ritkán használt beállításokról van szó, de igen hátráltatja a munkát, ha gyakorta ismétlődő parancsok előhívására használjuk. Gondoljunk bele például, hogy a kijelölt szövegrész kivágásakor mennyire időigényes mozgássor az F10 funkcióbillentyű megnyomása, majd a kivágás menütétel kiválasztása a NYÍL billentyűkkel, és végezetül az ENTER leütése. Gyorsabb, ha a kivágás ikont nyomjuk meg gépeléskor, de még ilyenkor is oda kell vinnünk a kezünket az oldalt lévő egérhez, ami miatt meg kell szakítanunk a szövegbevitelt. Gépeléskor tehát a gyorsbillentyűk használata adja a legjobb eredményt, hiszen például az Emacsban elég megnyomni a CTRL-W billentyűkombinációt a kijelölt szövegrész kivágására, és az adatbevitel máris folytatható anélkül, hogy a kezünket egy pillanatra is arrébb kellett volna mozdítanunk. Mivel az Emacs fejlesztői tisztában voltak ezzel az ergonómiai törvényszerűséggel, nem nagyon törték magukat, hogy más, kevésbé hatékony, de mások által felhasználóbarátnak mondott megoldásokat keressenek. Érdekes módon az XEmacs sem törekszik erre, pedig azzal a céllal hozták létre, hogy könnyebben kezelhető legyen. Elég, ha egy pillantást vetünk rá, és rögtön látjuk, hogy a fejlesztők megelégedtek néhány ikonnal (összesen 15-tel), amelyeknek a felét én személy szerint fölöslegesnek tartom a mindennapi munka szempontjából. Miért kell például az Info ikont kirakni a szemünk elé, amikor nem tesz egyebet, mint megjeleníti a Súgót? Ezzel szemben a Microsoft Word szövegszerkesztőben nem 15 ikont, hanem 15 ikonsort találhatunk!
Az XEmacsban inkább a menük burjánzanak. Mégsem állítanám, hogy az XEmacs fejlesztői lusták lettek volna, amikor kispórolták az ikonokat. Inkább azt feltételezem, hogy nem látták értelmét annak, hogy ikonokkal vagy nyomógombokkal zsúfolják tele az új Emacs-változatot, hiszen az a programozói réteg, amelyik az Emacs-környezetet használja, nem igényel ilyen változtatásokat. A fejlesztők tehát a programozás és a leíráskészítés szempontjából a leghatékonyabb megoldást választották, azaz a gyorsbillentyűkkel való parancshívást, még akkor is, ha ez a kezdők számára első pillantásra elrettentőnek tűnik. Nincs királyi út! Legfeljebb az Emacs.

Az Emacs és az Xemacs

A kétfajta Emacs között nincs nagy különbség, de hamar észrevehetjük, hogy nem teljesen egyforma a kettő. Az XEmacsban hiába kerestem a szöveges állományokat átalakítás nélkül beolvasó `find-file-literally` parancsot, csak az Emacsban találtam meg. Bizonyos billentyűkombinációk is másként működnek a két változatban, de összességében nem nagyok az eltérések. Tanulás közben célszerű mindkettőt kipróbálni, és végül annál maradni, amelyik jobban tetszik nekünk.
Manapság a legtöbb alkalmazást valamilyen grafikus felhasználói felületre írják, ezért felmerülhet a kérdés, hogy van-e értelme az X Window nélkül valamelyik terminálról indítani az Emacsot, hiszen mindegyik ablakkezelőben van terminálemulátor, és abban már futhat az Emacs. De mi tehet az a programozó, aki éppenséggel egy ablakkezelő megírásába fog bele? Neki valószínűleg a parancssorról kényelmes tesztelnie új programját, hiszen éppen fejlesztés alatt álló ablakkezelője feltehetően még nem igazán használható. Ilyenkor csak valamelyik parancssori szerkesztőt futtathatja.
Ha nincs egerünk, akkor a terminálon indított Emacs még akkor is új élményt fog jelenteni számunkra, ha már jártasak vagyunk a terminálemulátorban futtatott Emacs használatában. Megszoktuk már, hogy a legtöbb alkalmazásban az F10 nyitja le a menüt, és nincs ez másként az Emacsban sem. A parancssorra tévedt felhasználó azonban meglepetten tapasztalja, hogy hiába nyomkodja az F10 funkcióbillentyűt, semmi sem változik a menüsorban. Minél feszültebben figyel azonban a képernyő felső részére, annál kevesebb esélye lesz arra, hogy észrevegye, mi történik az alsó fertályban. Az Emacs ugyanis a képernyő alján, a mini átmeneti tárban (pontosabban a visszhangterületen) írja ki a menüket, és ott is kell választanunk a megjelenő menütételek a balra vagy jobbra nyilak közül, majd az ENTER megnyomásával.
Hasonló furcsaságokkal bármikor találkozhatunk az Emacsnál, de ha már megismertük, többé nem fognak meglepődni.

Utószó

Nyilvánvaló, hogy a vi vagy az Emacs más, mint amit a többi operációs rendszerekben megszoktunk, de az a tény, hogy furcsa és szokatlan, még nem lehet a minősítés alapja. A szövegben szereplő kódbetűs szavak nemcsak a programok nevére utalnak, hanem megegyeznek a héjba beírandó programindító parancsokkal, a nagybetűvel kezdődő szavak viszont csak a programok neveit jelölik.
A következő részben röviden ismertetem az Emacs használatát.

Szaló István
(ratiosoft@freemail.hu)
tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Több írása megjelent már a hazai számítástechnikai lapokban. Ha néha feláll számítógépe mellől, rendszerint művészettörténész feleségével és kisiskolás lányával „találja szemben” magát.

Fordítsunk rendszermagot!

A nyílt operációs rendszerek egyik legnagyobb előnye, hogy a rendszermagot a felhasználó is bármikor újrafordíthatja, ezáltal csökkentheti a méretét és gyorsíthatja a rendszer futását.

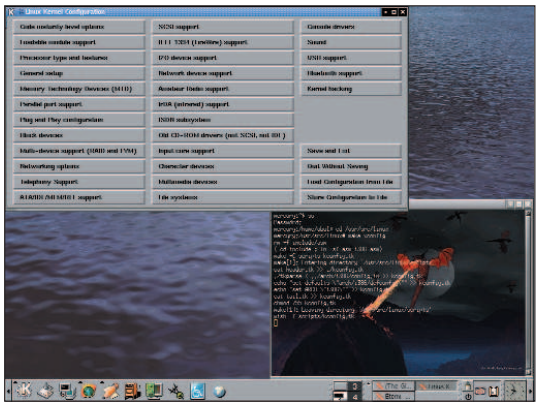
A rendszermag újrafordítása a Linux világában ugyan hétköznapi műveletnek számít, mégis számos nehézséget okozhat a kezdő felhasználók körében. Cikkünk nekik próbál segítséget nyújtani.

Bármely operációs rendszer legfontosabb része a rendszermag (kernel), amelynek legfontosabb feladata a felhasználói programok és a gépünk közötti kapcsolattartás biztosítása. Ezzel azonban még nem ért véget a tevékenységi köre, ugyanis szintén a rendszermag felelőssége a folyamatok (process) futásának felügyelete, továbbá a különböző biztonsági szabályok betartatása (nem engedi, hogy egy futó alkalmazás egy másik program által használt memóriaterületre írjon stb.).

A nyílt forráskódú rendszereknél a rendszermag forráskódjához is bárki hozzáférhet, ezért a felhasználók saját kezűleg fordíthatják újra, ha akarják. Milyen előnyökkel járhat a rendszermag újrafordítása? Először is várhatóan gyorsabb lesz, mint az előre lefordított „gyári" rendszermag, ugyanis a fordítóprogram a mi processzorunkhoz hangolja. Hasznos az is, hogy a fordítás előtt a felhasználónak lehetősége nyílik kiválasztani, mely összetevők szerepeljenek, illetve ne szerepeljenek a kész rendszermagban. Amennyiben például nincs és várhatóan nem is lesz semmiféle SCSI-eszközünk, a lefordított rendszermagból a teljes SCSI-támogatást ki lehet hagyni. Ezáltal jelentős méretbeli csökkenést érhetünk el. A rendszermag újrafordításával tehát elkészíthetjük a kifejezetten saját gépünkhöz illeszkedő rendszermagot.

A Windowshoz vagy OS/2-höz szokott felhasználók számára minden bizonnyal furcsán hangozhat, hogy a nyílt forrású operációs rendszerek világában a rendszermag újrafordítása teljesen hétköznapi műveletnek számít. Maga a fordítás nem nevezhető bonyolult műveletnek, sarokpontja egyedül az összetevők kiválasztása, amit azonban részletesebben is be fogunk mutatni. A rendszermag fordítása egyébként semmiféle programozói előképzettséget nem igényel.

Kezdjünk is neki! Legelső feladatunk magának a rendszermag forráskódjának a beszerzése lesz. A forrást Linuxunk telepítő CD-jén is megtalálhatjuk, de a legújabbat mindig fellelhetjük a CD-mellékleten, vagy letölthetjük az ➲ ftp://ftp.kernel.org-ról, illetve annak magyar tükréről, az ➲ ftp://ftp.hu.kernel.org címről. Jelen pillanatban a 2.2.x-es és a 2.4.x-es sorozatot párhuzamosan fejlesztik, e cikk írásának pillanatában az előzőből a 19-es, az utóbbiból pedig a 14-es a legfrissebb. Itt szeretnénk felhívni a figyelmet, hogyha régebbi Linux-változattal rendelkezünk, amely még a 2.2-es sorozatra épül, és át szeretnénk térni a 2.4-es rendszermagra, akkor bizonyos csomagokat frissítenünk kell. Ilyen például a telefonos internetelléréshez szükséges PPP-démon vagy a modulokat kezelő `modutils` (lásd később). E csomagok listáját a rendszermag leírásában találhatjuk meg.



A frissítéseket Linux-változatunk hivatalos honlapjáról is letölthetjük.

A magforrás becsomagolva körülbelül 22 MB, a lefordításához azonban körülbelül 60–80 MB szabad tárhelyre lesz szükségünk. Az Interneten elérhető rendszermagforrásokat általában gzippel vagy bzip2-vel tömörítik: az első esetben a kiterjesztés tar.gz, a másodikban pedig tar.bz2.

A forrást másoljuk a */usr/src* könyvtárba. Ha gzippel tömörítették, akkor a `gzip -d kernel-xxx.tar.gz` paranccsal csomagolhatjuk ki, bzip2 esetében a `bunzip2 kernel-xxx.tar.bz2` utasítást használhatjuk. (az xxx a rendszermagnak megfelelő változatszám) Ezt követően a létrejött .tar kiterjesztésű állomány kicsomagolásához adjuk ki a `tar -xvf kernel-xxx.tar` parancsot. Ez a parancs létrehoz egy Linux nevű könyvtárat. Ha már található ilyen könyvtár az adott helyen, előtte nevezzük át, például ***Linux-old***-ra (`mv Linux Linux-old`). Amennyiben mindent jól csináltunk, a forrást magát az újonnan létrejött ***linux*** könyvtárban találhatjuk meg.

A tar-állományra a továbbiakban nem lesz szükségünk, tehát nyugodtan letörölhetjük, ne foglalja fölöslegesen a helyet.

Most lépjünk be a magforrást tartalmazó ***linux*** könyvtárba! A rendszermag fordításának első és egyben legnehezebb lépése a kész rendszermag összetevőinek a kiválasztása. Mielőtt buzgón hozzálátnánk, meg kell beszélnünk egy fontos dolgot.

A Linux-rendszermag fontos tulajdonsága, hogy modularizált. Ez azt jelenti, hogy bizonyos alkatrészek és szolgáltatások támogatását nem feltétlenül kell közvetlenül a rendszermagba fordítanunk, hanem lehetőségünk nyílik rá, hogy modulokat készítsünk. Ezeket a modulokat bármikor kedvünkre betölthetjük a memóriába, illetve amennyiben feleslegessé váltak, el is távolíthatjuk őket onnan. Miért jó ez nekünk? Bizonyára akadnak olyan eszközeink, amelyeket nem használunk állandóan, a legjobb példa erre talán a hangkártya. A hangkártya szolgáltatásaira a rendszer mindennapi használatában nincs szükségünk, csupán abban az esetben, ha zenét akarunk hallgatni vagy lazítás-

képp valamely linuxos játékkal szeretnénk egy kicsit játszani. Ha jobban meggondoljuk, belátjuk, hogy teljesen felesleges a hangkártyatámogatást a rendszermagban „tárolnunk”, sokkal célszerűbb, ha modulba tesszük. Ennek köszönhetően a „hangos eszköz”-támogatás csak akkor kerül a memóriába, amikor hangkártyánkat ténylegesen „dalra fakasztjuk”.
Érdemes minél jobban kihasználnunk a Linux-rendszermag eme előnyét.
A bevált szokás az, hogy csak azokat az elemeket fordítjuk közvetlenül a rendszermagba, amelyekre a rendszer elindításához feltétlenül szükség van. Nem érdemes modulba tenni azoknak az egységeknek a támogatását, amelyeket a rendszer futása közben állandóan használunk: ilyen lehet például egy hálózati kiszolgáló esetében a hálókártya. Amelyik támogatást csak lehet, mind „dobáljuk” modulba.
A modulokat egyébként a `modprobe` parancs segítségével tölthetjük be, a feleslegessé vált modulok memóriából való eltávolítására pedig az `rmmod` utasítás szolgál. A betöltött modulokat az `lsmod` paranccsal listázhatjuk ki.
Egyes modulok betöltésekor értékeket is meg kell adnunk, hogy hol és mit, arról a rendszermag leírásában olvashatunk bővebben. A legtöbb modul betöltéséről azonban a rendszermag saját maga gondoskodik.
Lássunk neki az elemek kiválasztásának! Ehhez többféle út is kínálkozik: az egyik a konzolos menüvezérelt alkalmazás, amelyet a */usr/src/linux* könyvtárból a `make menuconfig` utasítás segítségével kelthetünk életre. Akik a grafikus környezetet kedvelik jobban, azok egy grafikus konzolból adják ki a `make xconfig` parancsot. A Linux-rendszermag fejlesztői az önsanyargatókról sem feledkeztek el: a `make config` parancsot az ő figyelmükbe ajánljuk.
A különböző elemeket különböző osztályokba csoportosítva találjuk. A továbbiakban helyhiány miatt csak a legfontosabbakra térhetünk ki (az összes támogatás részletes bemutatására a fél újság sem lenne elég). Mindenesetre bővebb tájékoztatásért nézzük át a rendszermag leírását vagy nyomjuk meg a ***Help*** gombot.

Code maturity level options

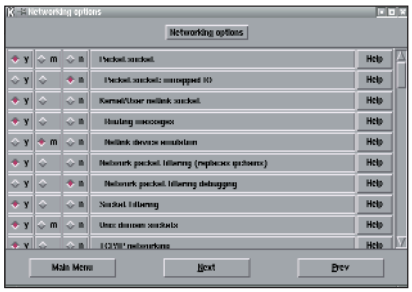
Ha bekapcsoljuk, elérhetjük a rendszermagban szereplő, de még csak kísérleti (EXPERIMENTAL) állapotban lévő támogatásokat is. Figyelem, az ilyen szolgáltatások nem üzembiztosak, ezért mindenki csak a saját felelőségére használja őket!

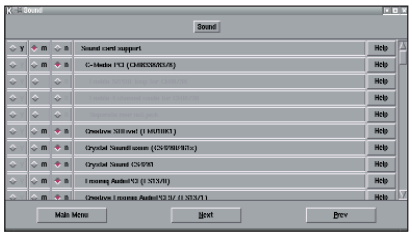
Loadable module support

Az ***Enable loadable module support***-ot mindenképpen fordítsuk be a rendszermagba, mivel nélküle nem élvezhetjük a modulok nyújtotta előnyöket.

Kernel module loader

A modulok önműködő betöltését teszi lehetővé.





Processor type and features

Itt adhatjuk meg processzorunk típusát. Ez azért fontos, mert a fordítóprogram a rendszermagot erre a processzorra fogja hangolni, ennek köszönhetően rendszerünk sebessége jelentősen nőhet.

High memory support

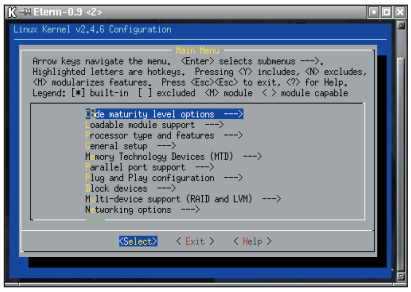
Ezt állítsuk ***Off***-ra, amennyiben 1 GB-nál kevesebb fizikai memóriával rendelkezünk.

Symmetric multi-processing support (SMP)

Csak abban az esetben kapcsoljuk be, ha egynél több processzorral bírunk, ugyanis ez a támogatás teszi lehetővé, hogy a rendszermag egynél több processzort használjon a folyamatok futtatására.

General setup

A ***PCI access mode***-nál adhatjuk meg, hogy a rendszermag milyen módon keresse meg a különböző PCI-os eszközöket. Ez történhet a BIOS segítségével, de közvetlenül (***Direct***) is. A legbiztosabb, ha az ***Any***-t választjuk; ebben az esetben a rendszermag PCI-os eszközeinket először a BIOS-on keresztül, majd ha ez sikertelen, közvetlenül próbálja meg elérni.
Az MCA-támogatás a PS2-es eszközök meghajtására szolgál. Ha PCMCIA-s kártyákkal is rendelkezünk, azok támogatását is itt kapcsolhatjuk be. A ***System V IPC***-t feltétlenül tegyük működővé, ugyanis ez teszi lehetővé a futó folyamatok közötti kapcsolattartást. Nagyon ügyeljünk rá, hogy a ***Support for ELF binaries***-t mindenképp magába a rendszermagba fordítsuk (és ne modulba!). Az elf a Linux futtatható binárisainak a formátuma, olyasmi, mint a Windows világában az exe. A többi bináris támogatását viszont nyugodtan modulba is helyezhetjük. Ha masinánk energiaellátását programból is szeretnénk szabályozni, ne felejtsük el bekapcsolni a ***Power management support***-ot sem!



Parallel port support

Ez a párhuzamos kapu támogatása, nyugodtan tegyük modulba. Figyelem, amennyiben PC-t használunk, a ***PC style hardware***-re is szükségünk lesz!

Block devices

Ide tartozik a PC-s hajlékonylemez-meghajtók támogatása, amit nyugodtan tegyünk modulba, még abban az esetben is, ha Linuxunkat lemezről indítjuk.
A rendszermagot ugyanis a lemezről a Linux indításvezérlője, a LILO tölti be, ami saját maga is tudja a lemezt, illetve a merevlemezeket kezelni.

Multi-device support

Ennél a menüpontnál a RAID-eszközök különböző támogatásait találjuk. A ***linear*** azt jelenti, hogy az összekötött lemezeket folyamatosan, egymás után töltjük meg. Az is megoldható, hogy több lemezt lássunk egy fájlrendszerként, ilyenkor a ***striping*** (csíkozás) nyújthat hasznos szolgáltatásokat. A ***mirroring***-gal (tükrözés) pedig megoldhatjuk, hogy két különálló lemezre pontosan ugyanazokat az adatokat írjuk fel. Ez nagymértékben növeli adataink biztonságát, ha ugyanis az egyik lemez megsérül, adataink a másikon még mindig elérhetők lesznek.
Az ***LVM support*** lehetővé teszi, hogy lemezterületeink összevonásával logikai kötetet hozzunk létre. Ezt a szolgáltatást csak a 2.4-es rendszermagok tartalmazzák. Akkor tehet jó szolgálatot, ha hirtelen lenne sok helyre szükségünk, de nem akarunk RAID-et használni.

Networking options

A ***Packet socket***-ra csak néhány egyedi hálózati alkalmazásnak van szüksége (például `tcpdump`), tehát nyugodtan modulba tehetjük. A ***Kernel/User netlink socket***-ra és a ***TCP/IP networking***-re azonban feltétlenül szükségünk lesz. A Linux-rendszermag ezenkívül számtalan hálózati szolgáltatást is tartalmaz, például teljes csomagszűrő tűzfalat. Ezt a 2.2-es rendszermagokban IP Chainsnek hívták, az alrendszert azonban a 2.4-esben kicserélték egy másikra, amelyet az IP Tables névre kereszteltek.

ATA/IDE/MFM/RLL support

Az IDE-eszközök és lapkakészletek támogatását tartalmazza. Figyelem, amennyiben Linuxunk fő lemezterülete IDE-me-

Példa a /etc/lilo.conf fájlra

```
boot=/dev/hda
prompt
timeout =200
# mennyi idı Ælljon
# rendelkezØsØre a
# felhasznÆl nak annak
# eld ntØsØre, hogy melyik
# rendszermaggal induljon
default=uj
# melyik rendszermag legyen
# az alapØrtelmezett
image =/boot/vmlinuz
   label = regi
   root=/dev/hda8
   read-only
image=/boot/œj kernel
   label=uj
   root=/dev/hda8
   read-only
```

revlemezen található, az IDE-támogatást semmiképpen se tegyük modulba, hanem közvetlenül a rendszermagba fordítsuk!

SCSI support

Amit az IDE-knél elmondtunk, az SCSI-nél is igaz. Ha nincs és nem is lesz SCSI-eszközünk, ezt a részt nyugodtan kihagyhatjuk. Egyvalamire hívjuk csak fel a figyelmet: a párhuzamos kapura köthető Iomega zipmeghajtó is az SCSI protokollt használja.

Network device support

Itt a különböző hálózati kártyák támogatását találhatjuk. Ki tudja miért, de a PPP is ide került, ez szükséges a telefonvonalas internetkapcsolat létrehozásához. Ha tehát modemmel kapcsolódunk a Világhálóra, a PPP-támogatást ne felejtsük el legalább modulba tenni.

ISDN subsystem

Ma már az analóg telefonvonal mellett az ISDN is egyre népszerűbb. A Linux természetesen ezt is gond nélkül támogatja. Meg kell jegyeznünk azonban, hogy az ISDN-en keresztüli internetezéshez más PPP-támogatás szükséges, mint a „hagyományos" telefonvonalaknál; előbbinél a ***Support synchronous PPP*** lesz a megfelelő.

Multimedia

Itt találhatjuk a videodigitalizáló és tv-tuner kártyák támogatását.

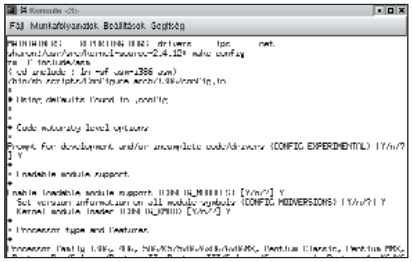
Filesystems

Megmondhatjuk, hogy rendszermagunk milyen fájlrendszereket támogasson. A legfontosabb a ***Second extended fs support***, amely nem más, mint az ext2, a Linux-lemezterületek fájlrendszerének a támogatása. Ezt sem szabad modulba tennünk.

Sound

A hangkártya-támogatást tartalmazza. A ***Sound card support***-ra mindenképpen szükségünk lesz. A rendszermag az általa ismert hangkártyákra a más Unix-rendszerekben is elterjedt OSS-t használja, ezzel azonban számos gond akadt, például kevés eszközt támogat. Így Linux alá kifejlesztették az ALSA-t, amely teljesen felváltja a rendszermagban lévő OSS-t. Ha készen vagyunk, nyomjuk meg a ***Save and Exit*** gombot, ezután visszatérünk a parancssorba. Most már kezdetét veheti a fordítás! Első feladatunk a függőségek beállítása, azaz az általunk kiválasztott elemeket elő kell készíteni a fordításra. Adjuk is ki tehát a `make dep` parancsot. Ha ez megtörtént, a `make clean`-nel eltávolíthatjuk a feleslegessé vált állományokat. Most következik maga a tényleges fordítás. Először a rendszermagot kell lefordítanunk, ezt követően pedig a modulokat. A lefordított rendszermagot egyébként rendszermaglenyomatnak (kernel image) nevezzük. A rendszermaglenyomatot a `make zImage` paranccsal készíthetjük el. Ha túl nagyra sikerült, két dolgot tehetünk: vagy több támogatást teszünk modulba, csökkentve ezzel a rendszermag méretét, vagy a `make zImage` helyett a `make bzImage` parancsot használjuk.
A következő lépés a modulok lefordítása a `make modules` paranccsal. Ha ezzel is kész vagyunk, a `make modules_install` paranccsal egyből a helyükre tehetjük őket, ilyenkor a ***/lib/modules/kernel-xxx/*** könyvtárba kerülnek.

Maga a fordítás általában körülbelül 10–15 percet vesz igénybe, de az időtartama természetesen a gép sebességétől függ, illetve attól, hogy mennyi elemet fordítottunk le. Ha a lenyomat vagy a modulok fordítása nem járna sikerrel, olvassuk el figyelmesen a hibaüzenetet, hátha az elemek kiválasztása során követtünk el valami hibát. Ha mégsem, próbálkozzunk a `make mrproper` pa-



ranccsal, ennek hatására a fordítást „tiszta lappal" kezdhetjük. Figyelem, e parancs kiadása után újból ki kell választanunk a lefordítandó elemeket! Utolsó lépésként telepítenünk kell a kész rendszermaglenyomatot. A lenyomatot az ***arch/i386/boot*** könyvtárban találjuk zImage vagy bzImage néven. Másoljuk a ***/boot*** könyvtárba, például ujkernel néven. Ezt követően az új rendszermagunkat hozzá kell adnunk a LILO-hoz.
Akik már behatóbban ismerik a LILO-t, bizonyára tudják, hogy lehetővé teszi, hogy ne csak a gépünkön lévő operációs rendszerek között, hanem a használni kívánt linuxos rendszermag között is választhassunk. Nem érdemes a régi rendszermagunktól egyből megválnunk, mivel elképzelhető, hogy az új rendszermaggal a rendszerünk nem fog elindulni. Az ebből adódó kellemetlenségeket kerülhetjük el, ha a LILO-ban a régi lenyomatunkat is meghagyjuk.
A LILO beállítófájlja a ***/etc/lilo.conf***. ***Listánkon*** példát láthatunk rá, miként adhatjuk hozzá a frissen fordított rendszermagunkat. Ha ezzel megvagyunk, adjuk ki a `lilo` parancsot, működőképessé téve az új beállításokat.
Indítsuk újra a rendszerünket, és ha mindent jól csináltunk, az új rendszermag fogad minket.

Garzó András
Körülbelül 3 éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.

Aerodinamika és a nyílt forrású alkalmazások

Steve bemutatja, miként lehet nyílt forrású programokat használni repülőgép-kísérleteknél.

Régen, még a Nyílt Forrású Programok mozgalma, a World Wide Web, a Szabad Program Alapítvány és a GNU megszületése előtt az Unsteady Aerodinamics Laboratory of the National Research Council of Canada alkalmazásában álltam. Feladatom a nagysebességű szélcsatornakísérletek adatainak összegyűjtése volt. Abban az időben a laborban egy különleges valós idejű miniszámítógép, egy Hewlett-Packard HP-1000 F sorozatú működött. Valamikor a nyolcvanas évek elején vagy közepén részt vettem a HP International Users Group tanácskozásán, és egy mágnesszalaggal tértem haza, ami a résztvevők által írt programokat és cikkeket tartalmazta. A szalagot a könyvtárrendszerbe fűztem és miután bepillantást nyertem az indexállományba, úgy éreztem magam, mint a kisgyerek, amikor karácsony este kicsomagolja az ajándékait. Először találkoztam forráskódmegosztással és fogalmam sem volt róla, mi mindent tartogathat a jövő egy ilyen nagyszerű és világos elképzelés számára.

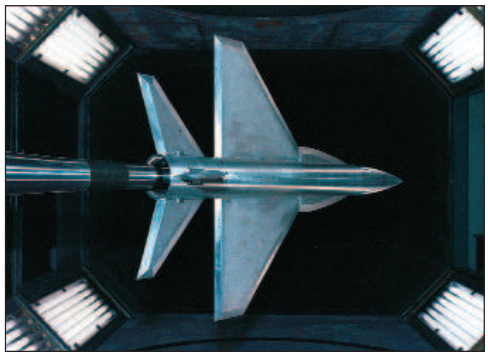
1990-ben kaptam meg az első unixos gépet: az akkor csúcstechnikának számító Silicon Graphics 4D/80GT-t saját T1-es internetkapcsolattal. A kicsi, hálózati kapcsolattal nem rendelkező, valós idejű operációs rendszerrel ellátott gépről az IRIX-alapú hálózati számítógépre történő váltás új, de kockázatos világra nyitott ablakot. Nagyon sok új ismeretet kellett elsajátítanom. 1992-ben vágtam bele a Byzantine nevű parancsállomány megírásába, amely a szélcsatorna adatállományainak kezelésére az `awk` és az `scd` kombinációit használta. Egy napon az egyik feladatom megoldásához tanácsot kértem az Useneten, és valaki megemlítette a Perlt. Bárcsak tudnám, ki volt az – most köszönetet mondhatnék neki, mert sokkal könnyebbé varázsolta az életemet. Egy éven belül a Perl mind az SGI-n, mind az otthoni Macintosh-gépemen nélkülözhetetlenné vált. A Perl a nyílt forrású programok legfontosabb darabja lett a laboratóriumban is.

Abban az időben a Mosaic programmal kísérletezgettem, amit – bár hasznosnak találtam – először igencsak alábecsültem. Nem sokkal később a NCSA által készített HTTPd-t telepítettem, és ekkor csodálkoztam rá a Web lehetőségeire is. A HTTPd később Apache néven született újjá, és a második legnagyobb OSS projektté lépett elő, amelytől egyébként a labor mindennapi munkája is függött.

1995-ben az Aerodinamikai Kutatóintézetet átszervezték, és a zűrzavar csillapultával a mostani helyemen találtam magam: az Aerodinamikai Laboratóriumban, ahol kis sebességű, 23 méteres szélcsatornájának felügyeletét látom el. Akkortájt kezdtem webalapú programokat készíteni, hogy kiterjesszem a már meglévő adatrendszer képességeit. Ezeket parancssoron és QNX, illetve AIX alatt futó X Window rendszeren keresztül lehetett elérni. Azért váltottam böngészőalapú programokra, mert a szélcsatorna adatbázisához kapcsolódó ügyfelek sokszínűsége ezt kívánta meg. Jóllehet munkánk nagy részét repülőgépek kipróbálása tette ki – olyan nagyvállalatok számára, mint a Bombraider Aerospace –, az utóbbi néhány évben azonban autókat, buszokat, teherautókat, motorkerékpárokat (a kedvenceim), villamos távvezetékeket, hidakat, antennákat, valamint olimpiai kerékpárosokat, síelőket és bobcsapatokat is próbára tettünk. Az ügyfelek ilyen széles köre – a számítástechnikához alig értőktől egészen a profikig – igazi kihívást jelentett a felhasználói felület megalkotásában. Mióta a vezetés is tudja, hogyan kell az Interneten keresgélni, elhatároztam, hogy alkalmazásaink közül egyet megpróbálok átültetni Perlre, amelyet azután webalapú felületen lehet elérni. Nagyon kedvező visszajelzéseket kaptam, mind a könnyű használhatóságának, mind az ügyfelek és a munkatársak által tapasztalt magas fokú kénye-

1. kép A kanadai sícsapat tagja 2 m × 3 m-es szélcsatornában

2. kép Jármű vizsgálata 2 m × 3 m-es szélcsatornában

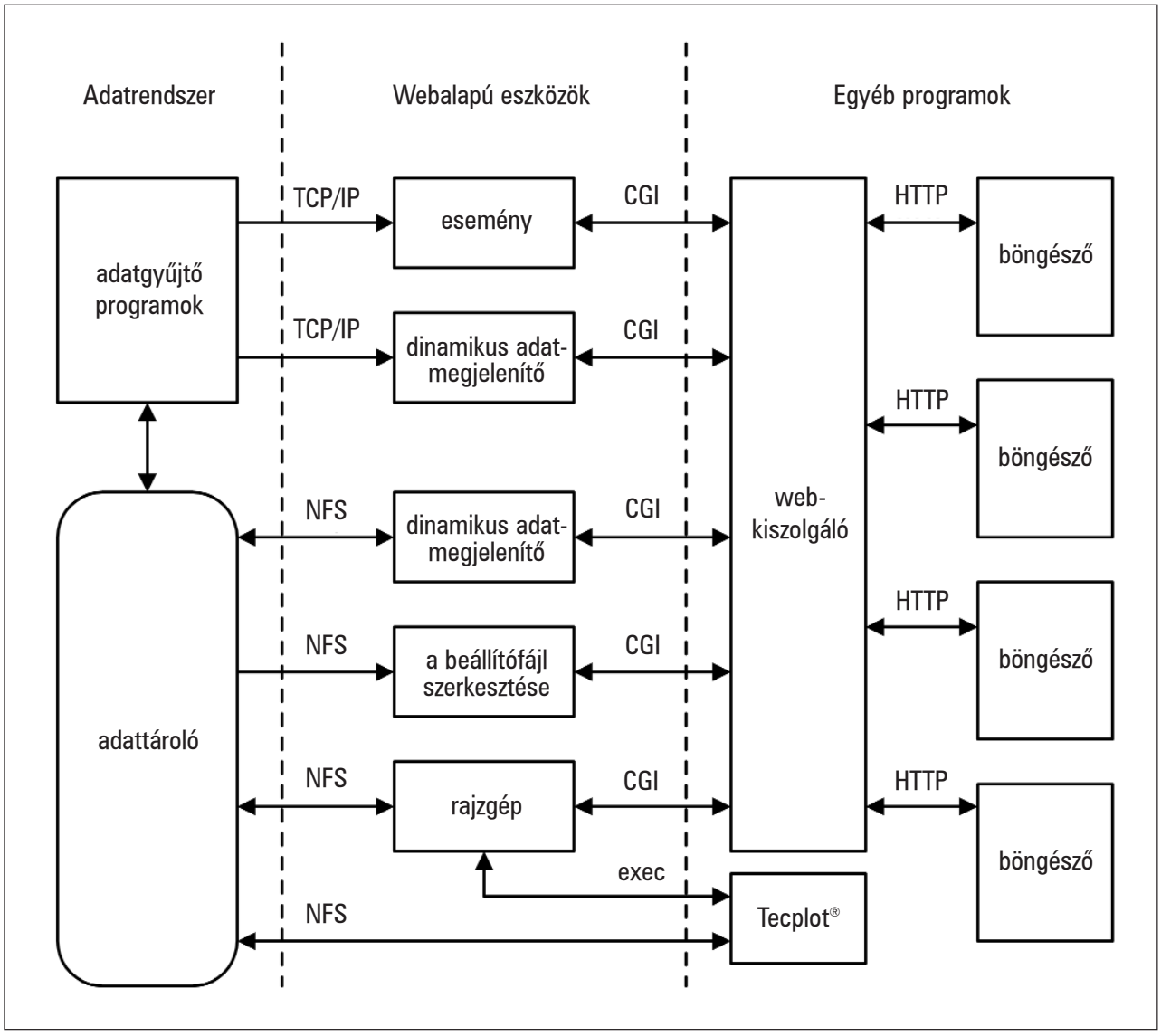


3. kép Harci repülőgép modellje 2 m × 3 m-es szélcsatornában

lemnek köszönhetően, ezért folytattam a webalapú eszközök készítését.
A harmadik kiemelkedő OSS-projektet jó pár évvel később fogadtuk örökbe. 1998-ban egy 24 csomópontos Alpha/Linux Beowulf telepet vásároltunk a folyadékok dinamikájával foglalkozó csoport számítási igényeinek kielégítésére. Ez kiváló lehetőségnek bizonyult az új módszer kiértékelésére, mert a feladat ugyan sokkal számításigényesebb volt, mint a szélcsatorna adatrendszeréé, a labor ügyfelei számára azonban nem bírt akkora horderővel. A sikeres alkalmazás meggyőzött bennünket, hogy az eddig használt kereskedelmi operációs rendszerek mellett a Linux életképes választási lehetőség.
Míg ezek a terjedelmes programok a helyükre kerültek, számos kisebb OSS-programot kezdtünk használni, például a Ghostscriptet, az Xmgr-t, a Vimet és a Neditet.

A jelen

A cikk további követhetőségéhez leírom az egyik, az adatfeldolgozás szempontjából jellemző repülőgép-kísérletet. Miután a modellt elhelyezik a szélcsatornában, 1–5 hét időtartam szükséges a próba befejezéséig. Ez idő alatt több mint ötszázezer mérést végeznek, ennek következtében 2000 X-Y plot, 4000 lemezállomány jön létre, és 500 MB szöveges adat jelenik meg a képernyőkön. Szükségszerű, hogy ennek az adattengernek a kezelésére gyors és egyszerű módszer kell, hogy rendelkezésre álljon. Az ügyfelek és a labor mérnökei az adatok gyűjtését, tárolását és megjelenítőrendszerét minden esetben Apache alatt futtatott Perl CGI programon keresztül érhetik el (lásd *ábránkon*). A szélcsatorna operátorai a kísérlet számos vezérlőelemét ugyanilyen módon kezelhetik. Amennyiben a vezérlőteremben a felhasználó bármelyik számítógépen megnyit egy webböngészőt, a szélcsatorna ügyfélhonlapja önműködően betöltődik. Ezen a lapon keresztül lehet elérni azt az öt webalapú programot, amelyeket az eddigiek folyamán írtam: ábrázoló, beállítási állományszerkesztőt, adatállomány-nézegetőt, eseménynaplózót és dinamikus adatmegjelenítőt. Ezenkívül léteznek hivatkozások a helyi segédeszközökhöz is, például a rendszerleíráshoz, illetve a mértékegység-átváltó számológéphez. Szeretném felhívni a figyelmet, hogy a laboratórium eléggé



Webalapú programrenszer szerkezete

korlátozó típusú helyi hálózati modellt használ, amely segíthet a webalapú rendszereknél előforduló biztonsággal kapcsolatos aggodalmak eloszlatásában. Elsőként az ábrázolórendszert fejlesztettük ki, és annak bizonyítására használtuk fel, hogy a szélcsatorna ügyfelei adataikat a Weben keresztül is el tudják érni. Amikor elkezdtük az Amtek Engineering által készített Tecplot kereskedelmi adatmegjelenítő programot használni, elhatároztam, hogy az ábrázolórendszert e program köré írom meg. A felhasználók az ábrázolómintákat – a beállítások kiválasztásával és szövegmezők kitöltésével – egyszerűen HTML-űrlapon keresztül állíthatják be. Ezeket a mintákat azután Tecplot-parancsállományok előállítására használjuk, ezáltal képernyőn vagy papíron lehetőség nyílik az eredmény megtekintésére. Egy démon (szintén Perlben íródott) ugyanezeket a mintákat használja a nyomtatásra, amely minden szélcsatorna-kísérlet végén önműködően zajlik. A beállítóállomány szerkesztését egy másik webalapú programmal valósítottam meg, amelyben a kísérleti adatokat gyűjtő és egyszerűsítő programok vezérlőállományait gyors és egyszerű módszerrel lehet módosítani. A felhasználók olyan űrlapot látnak, amelynek minden sora szövegmezőket és választógombokat, valamint a hozzá tartozó értékneveket tartalmazza. Az a Perl program, amely ezt a HTML-űrlapot hozza létre, dinamikusan előállít egy JavaScript-kódot is, amelynek az a feladata, hogy az űrlap benyújtása előtt ellenőrizze a kitöltött adatok érvényességét. Ha érvénytelen bejegyzést talál, a beviteli mező mellett villogó nyíl jelenik meg, és egy előugró párbeszédablakban a hiba jellege lesz olvasható.
Az adatállomány-megjelenítő egy egyszerű CGI program, amely egy adott szélcsatornapróba adataiért kutatja át a lemezterületet. Minden bejegyzéshez, amely a keresési mintára illeszkedik, egy HTML-gombot készít. Ezek a gombok táblázatban helyezkednek el, ahol minden sor a hasonló csatornapróbákat, illetve minden oszlop a megadott adattípust tartalmazza (például nyers, egyszerűsített). Bármely gomb megnyomására új böngészőablak nyílik meg, ahol a kiválasztott állomány formázott és elemzett alakban tekinthető meg. Ezután a felhasználók számára adott a lehetőség, hogy a saját gépükre CSV, Matlab vagy bármely más formátumban letöltsék. Minden új alkalmazás és számos régebbi kód állapotüzeneteket állít elő – eseményeket, amelyeket eseménynaplózó rendszerrel kezelünk. Ez a rendszer két fő részből tevődik össze: az első egyszerű Perl-démon, amely egy TCP/IP-kaput figyel, hogy érkezik-e rajta üzenet, s amennyiben igen, a naplóállományokban tárolja őket. A rendszer másik része egy webalapú megjelenítő, amellyel a felhasználók különböző szempontok

4. kép Forgó F18-as modell a vízcsatornában

alapján kereshetnek a naplóállományok között, mint például az esemény ideje, a számítógép neve, az esemény súlyossági szintje. Ez a program jelentéktelennek tűnik, pedig nélkülözhetetlen, mert az adatgyűjtő, -kezelő és -megjelenítő rendszer számos, különböző operációs rendszerrel működő számítógépből áll. Az ilyenfajta osztott rendszerekben a hibakeresés nagyon nehéz (különösen az összetettség miatti összehangolás okoz gondokat), általános eseménynaplózó rendszer nélkül csaknem lehetetlen. A felhasználók szempontjából az egyetlen nem interaktív eszköz a dinamikus adatmegjelenítő rendszer: Perl-kiszolgálón alapul, amely az adatgyűjtőrendszertől adatcsomagokat fogad. A felhasználók egy CGI-programon keresztül a kiszolgálóhoz kapcsolódva tudják megjeleníteni ezeket az adatokat. A CGI-program NPH-t (nonparsed header) vagy „server push”-t használ. A program egy adattáblázatot jelenít meg, amelyben az újabb adatokat dinamikusan a táblázat tetejére írja, a régi adatot pedig lefelé tolja. A program készítése folyamán aggódtam a memóriahézagok miatt, amelyek nemcsak Perlben vagy Apache-ban, de az ügyfelek böngészőjében is előfordulhatnak. Fölöslegesen, hiszen akadtak különleges NPH-ügyfelek, amelyek a kiszolgálóhoz folyamatosan kapcsolódva olyan anyagokért kutatták át a rendszert, amelyek több mint öt hete készültek. Eközben minden gond nélkül 500 MB-nál több adatot jelenített meg.
Ennek az öt programnak bármelyike külön-külön jól használható lenne, de aligha nevezhetők forradalminak. Mihelyt azonban egyesítjük őket, egyszerű, szilárd és nagyméretű környezetet kapunk. Nincs szükség nehezen érthető parancsokra, hosszú adatelérési útvonalakra, parancsbillentyű-kombinációkra vagy bármi másra, amely különféle típusú kezelőfelületekre jellemző. Nem kell mást tenni, mint kattintani, kijelölni, és kitölteni a mezőket – mindenki tudja, mit és hogyan tegyen.

A jövő

Teendőim hosszú listáján előkelő helyen szerepel azoknak a Perl-kódoknak a kétprocesszoros Intel/Linux-rendszerre történő ültetése, amelyeket az utolsó megmaradt SGI-rendszeren fejlesztettem. Bár a programok futtatása a jelenlegi formájukban jelentéktelen feladat, éltem az összes kód újrahangolásának lehetőségével. Még egy alkalmazás befejezése lenne különösen fontos: a modell viselkedését vezérlő rendszer felhasználói felületéé. Ráadásul figyelembe kell vennem adatformátumaink váltását – az arcane házilag fejlesztett formátumától az XML-ig. Ez szintén néhány új kód szükségességét eredményezi. Távolabbra pillantva a jövőbe, remélem, elég időm lesz kifejleszteni néhány VRML-alkalmazást is, amelyek a szélcsatornában elhelyezett modellek és szondák terhelés- és nyomásviszonyait képesek utánozni, és 3D-ben dinamikusan megjeleníteni. A műszeres csoport is vizsgálja a beágyazott, illetve valós idejű Linux-rendszerek felhasználásának lehetőségét.
Mire mindezen munkák elkészülnek, a nyílt forrású programok egyre növekvő fontosságú szerepe már vitathatatlan lesz az Aerodinamikai Laboratórium mindennapi munkájában.

Steve Jenkins az Aerodynamics Laboratory of the Institute for Aerospace Research vezető programozója és elemzője, a légi kutatások adatfeldolgozásában több mint húszéves tapasztalattal rendelkezik.

Amikor a Palm és a Linux beszélgetni kezd egymással...

Két egyetemi hallgató megoldotta a Palm és a linuxos számítógép összehangolását.

A Palm nagyszerű hordozható készülék: jegyzetelhetünk, találkozókat tervezhetünk vagy akár naplót is írhatunk vele. Csodálatos, hogy mindenhol velünk lehet útközben. Belső hálózatunk kiszolgálója szintén bámulatos masina. Vállalati terveinket, a megbeszélések napirendi pontjait, az érdekes feljegyzéseket, a címeket és a teljes üzleti adatbázist tárolja. Ez a kiszolgáló Linuxot, Apache-kiszolgálót és egy MySQL-adatbázist futtat, amelyeket egy, a célra tervezett alkalmazásmotor fog össze.

Ugye, csodálatos lenne, ha a két gépet össze tudnánk kapcsolni? Bárcsak sikerülne a Palmról elérni a Linux-kiszolgálón található adatbázist – máris jó úton haladnánk. A leírás szűkszavú, az Internet azonban hatalmas. ***Kevin*** és ***Jeffrey*** két egyetemista, akik kis erőráfordítással új megoldást fejlesztettek ki. Ennek alapján végezhetjük el a megfelelő változtatásokat a Palm-gépeken és a vállalati kiszolgálón, s e módosításokat a másik eszköz adatbázisaival is végrehajthatjuk.

A lejjebb látható kódokat egy RedHat 6.x Linuxot futtató Intel-alapú gépen és egy Palm OS 3.5-öt futtató, soros bölcsővel rendelkező Palm Vx-en próbáltuk ki, de más összeállítás sem okozhat gondot. Az alkalmazott könyvtárfájlok a Palm első piaci megjelenése óta (akkoriban még a 3Com gyártotta a gépet) nem változtak. Feltételezem (bár nem próbáltam ki), hogy a program a Visor-gépeken is gond nélkül működik, mivel ugyanazt az operációs rendszert használják.

Első lépésként az egyszerűbb résszel kezdjük: a Palmot kapcsoljuk össze a kiszolgálóval. Először is a Palm bölcsőjét kell összekötnünk a kiszolgáló soros csatlakozójával. Ezután létrehozunk egy pilot nevű eszközt, ami nem más, mint a soros kapu (esetünkben a */dev/ttyS0*) másodneve:

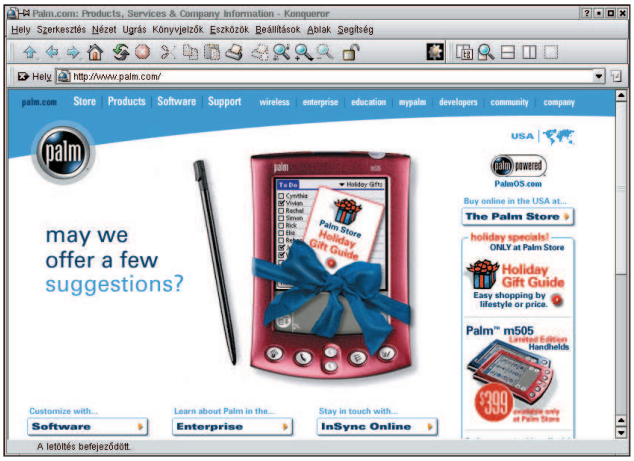
```
ln /dev/ttyS0 /dev/pilot
```

Most már egy C program és a ***HotSync*** gomb megnyomásával megnyithatjuk a kapcsolatot a Palmmal. Miután a kapcsolat létrejött, már csak a Palm adatbázisaiból kell a megfelelő adatokat kiolvasnunk.

A Palm és a számítógép közti kapcsolat és adatátvitel a ***pi*** könyvtárral egyszerűen megvalósítható. Ez a könyvtár a BSD-csatolófelületet utánozza: létrehoz egy foglalatot (socket), hozzákapcsolja az eszközhöz, figyel a bejövő kapcsolati kérelemre és elfogadja. A bejövő kapcsolatot a Palm és a bölcső kettőse váltja ki akkor, amikor a felhasználó megnyomja a HotSync gombot. Az *1. listán* láthatjuk, hogyan kell létrehozni egy Palm-kapcsolatra várakozó démont.

Miután a kapcsolat megvalósult, miként érhetjük el a Palm adatbázisait? Ezek mindegyike névvel rendelkezik. Az adatbázisokat a nevükkel nyithatjuk meg, majd meg kell adnunk az elérni kívánt rekordot, sőt az egész adatbázist is átnézhetjük. Macintosh- vagy Windows-gépeken ezt csővezetékek alkalmazásával oldják meg. A Palm is biztosít csővezetékeket e felületeken az összes, a Palm OS csomagba tartozó szabványos adatbázis számára. A Palm adatbázis-kezelő lehetővé teszi, hogy az adatbázisnak csak a módosított rekordjait tekintsük végig. Módosított – mióta is? Nos, a legutóbbi összehangolás óta, amikor a kiszolgáló utoljára érte el ezt az adatbázist. Ezt tehát programjainkban az összehangolás után kell megtennünk – a *2. listán* látható nyitott kapcsolatnál kell futtatni.

Amennyiben a Palm-adatbázisból rekordokat olvasunk ki, az nem számít összehangolásnak. Ennél többre van szükség, például arra, hogy a Palmra írunk, törlünk belőle és a saját MySQL-adatbázisunkból olvasunk. Mivel a MySQL-adatbázishoz való kapcsolódás ismertetése túlmutat e cikk keretein, most nem szólunk az összehangolás további részleteiről. ***Kevin Velghe*** nagyszerű leírást tett közzé a témáról, amelyre a ➲ http://www.duo.be/palm/mysql_palm.html címen bukkanhatunk rá.



1. lista A Palm összehangolása

```
Main() {
        int sd;
        struct pi_sockaddr addr;

        sd = pi_socket(PI_AF_SLP,
PI_SOCK_STREAM, PI_PF_PADP);
        addr.pi_family = PI_AF_SLP;
        strcpy(addr.pi_device, "/dev/pilot");
        pi_bind(sd, (struct sockaddr*)&addr,
➥sizeof(addr));

        sd = pi_listen(sd, 1);

        sd = pi_accept(sd, 0, 0);
        printf("HurrÆ! LØtrej tt a
              ➥kapcsolat...");
        pi_close(sd);
}
```

2. lista A módosított rekordok átfésülése

```
{
        int db, len, I, attr;
        recordid_t id;
        unsigned char buffer[4096];
...lØtrej n a kapcsolat...
        sd = pi_accept(sd, 0, 0);
        dlp_OpenDB(sd, 0, 0x40+0x80,
                ➥"DateBookDB", &db);
        for (;;) {
                len = dlp_ReadNextModifiedRec
                        ➥(sd, db, buffer, &id,
                        ➥&I, 0, &attr, 0);
                if (len < 0) break;
                printf(buffer); printf("\n");
        }
        dlp_ResetSyncFlags(sd, db);
        dlp_CleanUpDatabase(sd, db);
        dlp_CloseDB(sd, db);

        pi_close(sd);
}
```

További érdekességek

Michael J. Hammel a Linux Journal 1998 júniusában megjelent „Linux and the PalmPilot” című cikke nem a legfrissebb, de ismerteti a pilot-xfer használatát, ami nagyon hasznos segédprogram. Elolvashatjuk a ➲ http://www.linuxjournal.com/lj-issues/issue50/2711.html címen.
A „PalmOS Desktop HOWTO” című HOGYAN-ja a ➲ http://www.linuxdoc.org/HOWTO/PalmOS-HOWTO.html címen található meg. Az ➲ http://orbits.com-ra mutató hivatkozások legtöbbje már nem működik, hiszen ez a vállalat minden bizonnyal megszüntette az említett HOGYAN-ok fejlesztését. Ha valakinek sikerül megtalálnia az elveszett anyagokat, kérjük, küldje el őket a palm@duo.be címre, hogy a ➲ http://www.duo.be/palm címen mindenki számára elérhetővé tehessük.

3. lista A pack függvény

```
{
        int app_size, len;
        recordid_t pal_id, new_id;
        unsigned char buffer[512];
        struct Appointment app;

...
        strcpy(app.description,"Tennival ");
        app.begin = ...
...
        size = pack_Appointment(&app, buffer,
                ➥512);
        palm_id = 0;
        len = dlp_WriteRecord(sd, db, 0,
                ➥palm_id, 0, AppBuffer,
                ➥Appointment_size, &new_id);

}
```

A Palm-adatbázisban tárolt rekordok saját számokkal rendelkeznek. Amikor az eszközre rekordot írunk, ezt a számot kapjuk visszatérési értékként. Ezt a gépen vagy egy központi adatbázisban érdemes tárolnunk, így egy adott rekordot bármikor törölhetünk vagy frissíthetünk.
A `dlp_WriteRecord` egy Palm-rekordazonosítót fogad el. Amennyiben ez nulla, a Palm OS újat foglal le a számunkra; ha pedig létező azonosítót adunk át, akkor a megfelelő rekord kerül frissítésre. A legtöbb szabványos adatbázisrendszerben a rekordot a `pack` függvény csomagolja egy átmeneti tárolóba. Ez a folyamat a *3. listán* látható.

A Palm azonosítása

Amennyiben egy kiszolgálóval (tulajdonképpen egy bölcsővel) több Palmot is használunk (mint ebben az esetben is), meg kell határoznunk, hogy éppen melyik Palm csatlakozik a bölcsőre. Amint a kapcsolat létrejött, hívjuk meg a `ReadUserInfo` függvényt:

```
{
        int db, len, I, attr;
        recordid_t id;
        struct PilotUser U;
...lØtrej n a kapcsolat...
        sd = pi_accept(sd, 0, 0);
        dlp_ReadUserInfo(sd,&U);
        printf("Palm neve: %s", U.username);

        pi_close(sd);
}
```

Törölt rekordok

A Palm adatbázis-kezelő a rekordokat kiolvasás után nem törli – törlésre jelöli ki őket, de azt is megteheti, hogy a gépen vagy a kiszolgálón mentésre jelöli ki őket. Módosított rekord olvasásakor a fájltulajdonságokat ellenőrizni kell, amelyből megállapítható, hogy az adott rekordot törölni (vagy menteni) kell-e. Amint az adatbázis kitisztul, végérvényesen törlődik a Palmról:

```
{
    ...
    for (;;) {
       len = dlp_ReadNextModifiedRec(sd, db,
             ➥buffer, &id, &I, 0, &attr, 0);
       if (len < 0) break;
       if ((attr & dlpRecAttrDeleted) ||
            (attr & dlpRecAttrArchived))
          printf("T rlØsre kijel lve: %ld", id);
    }
}
```

Egy kicsi naplózás senkinek sem árt…

A Palm összehangolása után hasznos gyakorlat, hogyha a Palm naplófájljában megjegyzéseket is hagyunk róla. Akármit beleírhatunk, de az időpont és a dátum mindenképpen belekerül. Írjuk tehát az alábbi kódot programjaink végére:

```
{
       ...
       dlp_ResetSyncFlags(sd, db);
       dlp_CleanUpDatabase(sd, db);
       dlp_CloseDB(sd, db);

       dlp_AddSyncLogEntry(sd, "A Pilotr l
            ➥beolvastuk a m dos tÆsokat.\n");

       pi_close(sd);
}
```

Mi kell az induláshoz?

Ha a Pilotot linuxos géppel szeretnénk használni, szerezzük be a ***pilot-link*** csomagot. A kezelőfelületek számos rendszerhez elérhetők (Next, BSD, Solaris, OS/2, Linux stb.). A segítségükkel Python, Java, Perl, Tcl, C/C++ nyelvű programokat írhatunk. A szükséges fájl a ➲ http://ryeham.ee.ryerson.ca/pub/PalmOS címen elérhető FTP-kiszolgálón található meg, a neve: `pilot-link.0.9.3.tar.gz`, linuxos gépen minden gond nélkül lefordítható. A csomag tényleg több, mint csak egy csatolókönyvtár a használatát bemutató egyszerű példaprogramokkal. Ezek az egyszerű eszközök nagyon hasznosak: a segítségükkel teljes biztonsági mentést készíthetünk a Pilotról (és vissza is állíthatjuk), adatokat és adatbázisokat másolhatunk róla és vihetünk föl rá stb. A könyvtár C/C++, Perl, Python, Tcl és még néhány más nyelvből hívható meg. Aki rendelkezik némi programozási készséggel, az a csomagba tartozó példaprogramok és cikkünk listái segítségével könnyűszerrel összeállíthatja a számára szükséges eszközöket.
Az új programokat, a leírás kiegészítéseit, a megjegyzéseket, a HOGYAN-okat az olvasók a palm@duo.be címre küldhetik, mi pedig a ➲ http://www.duo.be/palm című honlapon mindenki számára elérhetővé tesszük őket.
Telepítsük a csomagot az alábbi paranccsal:

```
tar -xzvf pilot-link.0.9.3.tar.gz
```

Ez létrehozza a ***pilot-link.0.9.3*** könyvtárat, benne a forráskóddal. Lépjünk is bele.
Adjuk ki a `./configure` parancsot, ami átnézi a rendszert a fordításhoz szükséges kiegészítő fájlokat keresve. A `configure` a programot alapértelmezés szerint a ***/usr/local*** könyvtárba telepíti. Ha ez nem megfelelő számunkra, a `./configure --prefix=K NYVT`R` paranccsal állítsuk be a telepítés helyét.

Adjuk ki a `make` parancsot, ami lefordítja a csomagot. Ekkor a fájlok még nem kerülnek a helyükre, így a végleges telepítés előtt alkalmunk nyílik kipróbálni a programot. Ha egy régebbi változatra egy újabbat telepítünk, ellenőrizzük, hogy minden működik-e. Általában természetesen semmiféle gond nem szokott adódni.
Rendszergazdaként adjuk ki a `make install` parancsot, mely a fájlokat a megfelelő könyvtárakba helyezi. Ha rendszergazdaként nem tudunk belépni, akkor olyan helyre telepítsük a programot, amelyhez rendelkezünk írási jogosultsággal. Ne felejtsük el a futtatható fájlokat tartalmazó könyvtárakat beírni a rendszer alapértelmezett útvonalába (`PATH` változó). Nézzünk meg a csomag mellé kapott hasznos példaprogramokat is, amelyek leírását a ***További érdekességek*** részben említett helyen olvashatjuk el.

Johan Coppieters
(palm@duo.be) a Duo nv nevű cég vezetője. A belgiumi Brugesben székelő vállalat Belgium legnagyobb vállalatai számára készít weboldalakat, belső hálózatokat és internetes alkalmazásokat.

Kevin Velghe
írta a C nyelvű Palm–Linux összehangoló programot, s egy három hónapos iskolai gyakorlat cseretanfolyam közben a kutatás egy részét is ő végezte. A Duo nv-nél találhatjuk meg.

Könnyű álmok (10. rész)

A PAM használata a gyakorlatban

Sorozatunk előző cikkében (*Linuxvilág,* október 46. oldal) áttekintettük a felhasználók azonosítása kapcsán felmerülő kérdéseket, és általánosságban beszéltünk a Linux PAM rendszeréről. Írásunk célja, hogy megismertessük a PAM fontosabb alkotórészeinek használatát, és tanácsokkal szolgáljunk a beállításukkal kapcsolatban.

A PAM-rendszer fontosabb alapmoduljai

A PAM-rendszer az alapvető modulokat önműködően telepíti. Az alábbiakban felsoroljuk a leggyakrabban használt modulokat és fontosabb szolgáltatásaikat.

Általános PAM-hibakeresés

A „debug" kapcsoló

A PAM-modulok hibáinak felderítésére a `debug` kapcsoló használható. Beállításának hatására az adott modul hiba esetén a syslog(3) rendszerhíváson keresztül ír a rendszernaplóba. Mivel minden modul rendelkezik ezzel a kapcsolóval, a továbbiakban nem tárgyaljuk.

A PAM-rendszer fontosabb alkotórészei

A „pam_unix" modul

A `pam_unix` a legfontosabb és leggyakrabban használt modul a Linux-változatokban. A Unix-rendszerek hagyományos azonosítási (authentication) és feljogosítási (authorization) eljárásait biztosítja. A rendszer szabványos hívásait használja, tehát a */etc/passwd* és a */etc/shadow* állományokkal dolgozik. Érdemes megjegyezni, hogy a *pam_pwdb* modul is hasonló szolgáltatásokat nyújt, a felhasznált adatokat azonban adatbázisban tartja. Nagy felhasználószámú rendszereken érdemes alkalmazni.

- ***account***
  Kapcsolói: `debug`; `audit`.
  A felhasználói számla érvényességének ellenőrzését teszi lehetővé. A *shadow* állomány olyan mezőket tartalmaz (`expire`; `last_change`; `max_change`; `min_change`; `warn_change`), amelyek a felhasználó jelszavának kikényszerített cseréjét vagy a számla zárolását teszik lehetővé [1.]. Vigyázat, ha a *shadow* állomány a fenti mezők valamelyikét nem tartalmazza, az ellenőrzés nem hajtódik végre!
- ***auth***
  Kapcsolói: `debug`; `audit`; `use_first_pass`; `try_first_pass`; `nullok`; `nodelay`.
  A felhasználó jelszavas azonosítását tesz lehetővé. Amennyiben több jelszavas azonosítás is be van állítva (lásd később a `pam_ldap` modulnál), a `try_first_pass` érték használata célszerű. Ilyen esetben a rendszer a felhasználótól nem kérdezi meg újra a jelszavát, hanem az első modul által bekért jelszót használja. Ha azt szeretnénk, hogy a felhasználó több jelszóval lépjen be, ne használjuk. A `nullok` kapcsoló olyan felhasználók rendszerbe lépését teszi lehetővé, akiknek *shadow* állományában a kódolt jelszó mezője üres. A használata nem javasolt.
- ***password***
  Kapcsolói: `debug`; `audit`; `nullok`; `not_set_pass`; `use_authtok`; `try_first_pass`; `use_first_pass`; `md5`; `bigcrypt`; `shadow`; `nis`; `min`; `max`; `obscure`; `remember`.
  A felhasználók szabványos jelszócseréjét teszi lehetővé. Az `md5` és `bigcrypt` kapcsolók segítségével elérhető, hogy a jelszó ne a klasszikus `crypt` [2.] eljárással kódolva kerüljön a végleges helyére, hanem a megadottal. Ne felejtsük el beállítani, különben a rendszeren csak nyolc karakter hosszú jelszavakat lehet használni! Itt is alkalmazható a `try_first_pass`, ha a felhasználónak a különböző jelszótárakban egyforma jelszót szeretnénk beállítani. A `use_authtok` beállítás a modul számára kötelezővé teszi az előző modul által átadott jelszó beállítását. Erre a `pam_cracklib` használata esetén van szükség (lásd később). A `not_set_pass` kapcsoló segítségével letilthatjuk, hogy a bekért régi vagy új jelszó bármely más modulnak átadásra kerüljön. A `nis` kapcsoló hatására a rendszer a jelszó beállítására a NIS RPC-t használja. A `min` és `max` beállításával szabályozható a beállítható jelszó legkisebb és legnagyobb hossza. A `min`-t átlagos felhasználói rendszernél célszerű legalább nyolcra, erősen védett rendszernél pedig tízre állítani. Az `obscure` a beállítandó jelszón néhány alapvető ellenőrzést végez, amelyek a következők lehetnek: a jelszó nem hasonlíthat túlzottan az előzőhöz, nem lehet túl egyszerű (jelszóhossz, a használt karakterek típusa stb.), nem lehet az előző jelszó fordítottja vagy odavissza megegyező (például „qwerttrewq").
- ***session***
  Nincs kapcsolója.
  Használatával a felhasználó neve és a szolgáltatás a munkamenet (session) elején naplózódik (a leírás szerint a végén is, de a tapasztalat ennek gyakran ellentmond).

A „pam_deny" és a „pam_nologin" modul

A `deny` segítségével megakadályozható a felhasználó adott szolgáltatáshoz való hozzáférése. Az `auth` és `account` esetén a felhasználó azonosítását és hozzáférését teszi sikertelenné, és amennyiben a `password` elemben használjuk őket, a felhasználó nem tudja megváltoztatni a jelszavát. A `session` alatt használva lehetővé teszi, hogy a felhasználó ne hozhasson létre munkamenetet.
A `nologin` modul a PAM-rendszer `auth` elemében elérhető, és a szabványos unixos `nologin` használatát teszi lehetővé. Amennyiben a */etc/nologin* állomány létezik, az azonosítás sikertelen. Leggyakrabban a rendszer indulásakor alkalmazzák, többfelhasználós rendszeren azonban kényelmes lehetőséget nyújt a felhasználók belépésének időleges tiltására egy esetleges karbantartás idején.

A „pam_securetty" és a „pam_shells" modul

A `securetty` és a `shells` modul meghatározza, hogy az adott felhasználó által használt terminál szerepel-e a */etc/securetty* állományban, illetve a felhasználó bejelnetkező héjja benne van-e a */etc/shells* állományban. Amennyiben az állomány az adott bejegyzést nem tartalmazza, a felhasználót

mindkettő elutasítja. Mindkét modul a PAM-rendszer `auth` eleméből érhető el.

A „pam_listfile" modul

A `pam_listfile` az azonosítási szakaszban egy állomány tartalmán keresztül teszi lehetővé a karbantartás engedélyezését vagy tiltását.
Lehetséges kapcsolói:

- onerr=succeed|fail
- sense=allow|deny
- file=***állománynév***
- item=user|tty|rhost|ruser|group|shell
- apply=user|@group

A modul veszi az `item` által meghatározott elemet (ahol a `user` a felhasználó neve; a `tty` annak a terminálnak a neve, ahonnan a kérés érkezett; az `rhosts` a távoli gép neve – ha van; az `ruser` a távoli felhasználó nevét adja meg – ha van; a `group` pedig a felhasználó csoportja), és megnézi, hogy a `file` által meghatározott állomány tartalmazza-e. Ha tartalmazza és a `sense` értéke `allow`, a modul sikerrel tér vissza, ha `deny`, akkor elutasító választ ad. Amennyiben hiba lép fel (például a meghatározott állomány nem létezik), az `onerr` által beállított értékkel tér vissza. Ezt érdemes `fail`-re állítani. Az `apply` kapcsolót akkor célszerű használni, ha a vizsgált elem terminál, távoli gép vagy héj. Segítségével a sikeres visszatérés egy felhasználóhoz vagy egy csoporthoz köthető.
Így tehát egyszerűen korlátozható egy adott szolgáltatás elérése. Használatára a legjellemzőbb példa az FTP, amelynél a `listfile` modult használták fel, hogy bizonyos felhasználók számára megtiltsák a szolgáltatás elérését. A beállítóállományban ez a következőképpen néz ki:

```
auth  required  pam_listfile.so  item=user
➥sense=deny file=/etc/ftpusers onerr=fail
```

Egyéb felhasználására is mutatunk példát a későbbiekben.

A „pam_limits" modul

Lehetőséget ad a felhasználók által használható erőforrások korlátozására, amire azért van szükség, mert a többfelhasználós Linux-kiszolgálókon nem engedhető meg, hogy egy felhasználó olyan mértékben terhelje le a rendszert, hogy a többiek (különösen a rendszergazdák) ne tudják a munkájukat zavartalanul végezni. A Linux-rendszer e korlátozások használatát sajnos csak kis mértékben támogatja, így a rosszindulatú felhasználóktól csak a hagyományos módszerek védenek meg tökéletesen (időleges vagy végleges kizárás, vasalt orrú bakancs stb.).
Ezen keserű megállapítások a Linux 2.2.20-as és 2.4.14-es rendszermaggal folytatott hosszas kísérletezés után születtek. A felhasznált próbaprogramok az 1., 2. és 3. listán láthatók (24. CD Magazin/Konnyu könyvtár).
Amennyiben a felhasználók memóriafelhasználását korlátoztuk, a rendszer valamelyik fork-bombával túlterhelhetővé vált. Ha a belépésenkénti folyamatok (process) számát 4-re korlátoztuk, a fork-bombák akkor is szinte a teljes processzoridőt fel tudták használni, ráadásul az OpenSSH segítségével nem lehetett belépni a rendszerre. Tapasztalatunk szerint SSH-val csak akkor sikerült belépni a rendszerre, ha a lehetséges folyamatok száma legalább 40 volt. Mivel azonban a sikeres belépést követően a felhasználó 40 folyamatot futtathat, kedvezőtlen esetben a teljes processzoridőt le tudja foglalni. Az ésszerűtlen memóriafogyasztást meg lehet ugyan gátolni, de a sok memóriafoglalási kísérlet szintén megeszi a processzor idejének jelentős részét. Rendkívül kellemetlen, hogy ilyen esetben a legtöbbet a rendszermag dolgozik, így még korlátozni sem lehet.
A folyamatok számának korlátozása bizonyos esetekben a PAM-támogatás tökéletlen megvalósítása miatt nem megfelelő. Amennyiben a rendszeren a kifejezetten rosszindulatú felhasználókat ki tudjuk szűrni, van értelme a határok beállításának, mert ezzel csökkenthető a felhasználó akaratán kívül történő rendszertúlterhelés esélye. Sokat segíthet, ha a felhasználók `nice` szintjét csökkentjük, ezáltal hiba esetén a rendszergazdák nagyobb eséllyel tudnak sikeresen beavatkozni.
Eszményi az lenne, ha a felhasználónak általános határokat lehetne beállítani (jelenleg csak a belépésenkénti létezik), és a rendszermag lehetővé tenné annak beállítását, hogy egy bizonyos felhasználó által kezdeményezett (felhasználó vagy rendszermag által végzett) feladat legfeljebb mekkora részt kaphasson a rendelkezésre álló processzoridőből (fair share scheduling). Amíg a hivatalos rendszermagban ezek nem valósulnak meg, addig a felhasználók korlátozása csak részleges lehet.
Nem tartozik szorosan a témához, de itt érdemes megjegyezni, hogy a rendszermag lehetővé teszi annak a helynek a korlátozását, amit a felhasználók merevlemezen foglalhatnak. Így ésszerű mértékűre csökkenthető az egyes felhasználók területhasználata, és a levelesláda (mailbox) sem nőhet a többiek kárára egy adott méret fölé. Beállítása esetén azonban figyelni kell rá, mit tesz ilyen esetben a levelezőkiszolgáló.

Egyéb hasznos modulok

A `pam_env` modul (auth) környezeti változók előzetes beállítását vagy törlését teszi lehetővé. Többfelhasználós rendszeren célszerű alkalmazni, mivel a belépési héjjtól függetlenül teszi lehetővé a környezet egységes beállítását.
A `pam_rootok` modul (auth) azonosítja a felhasználót, ha a felhasználói azonosítója 0. Ennek akkor lehet értelme, ha a rendszergazdát nem akarjuk egy szolgáltatás minden egyes használatakor azonosítani. A Linux-változatok legtöbbjében a rendszergazdának megengedett a `su` használata jelszó nélkül, ami a beállítóállományban így fest:

```
auth    sufficient pam_rootok.so
auth    required   pam_unix.so
```

A rendszergazda úgy tevékenykedhet bármelyik felhasználó nevében, hogy nem adja meg annak a jelszavát. Ez felvet bizonyos erkölcsi kérdéseket, ami azonban szinte minden rendszergazdai jogosítványnál felmerül. Különleges esetben egy finoman hangolt, külső szakértők által is felülvizsgált rendszernél elérhető a rendszergazdák jogainak a szükségesre történő csökkentése, de ez komoly hozzáértést és erőforrás-ráfordítást igényel.
A `pam_chroot` modul (account, session, auth) segítségével lehetővé válik egy adott szolgáltatás root könyvtárának a PAM-rendszeren keresztüli beállítása. Hasznáról egy későbbi cikkben részletesebben írunk. A modullal jelenleg kissé nehézkes dolgozni, mivel a PAM-ot támogató programok egy része a munkamenetkezelést nem megfelelően valósítja meg. Jó példa erre az OpenSSH, ahol a PAM-megvalósítás félreérthetősége miatt a rendszer nem minden esetben működik helyesen. Többen kijavították az SSH hibáit, de a fejlesztők nem fogadták be a javításokat.
A `pam_motd`, `pam_mail`, `pam_lastlog` és `pam_issue` modulok a felhasználók tájékoztatását szolgálják. Belépéskor a terminálra a */etc/motd* és a */etc/issue* állományok tartalmát, az utolsó belépés idejét kiírják, továbbá jelzik, ha a felhasználónak új levele érkezett.
A `pam_radius` és `pam_krb4` modulok segítségével a felhasználók azonosítása egy RADIUS- [3.] vagy Kerberos- [4.] kiszolgáló segítségével történik.
Ezek az azonosítási eljárások általában nagyobb hálózatokon használatosak, és nagy biztonságú azonosítást tesznek lehető-

vé. A pam_securetty segítségével egy állományban meghatározható, hogy mely terminálok tekinthetők biztonságosnak. A pam_time modul használata lehetővé teszi a hozzáférés idő szerinti korlátozását. Minden biztonsági rendszer alapvető eleme a megszokott és elfogadott engedélyezése, és a kirívó esetek tiltása. Amennyiben például valószínűtlen, hogy a rendszergazda reggel tíz óra előtt belépjen a konzolról, e modul segítségével letiltható, vagy egy PAM-ot támogató játékalkalmazással megoldható, hogy csak munkaidőn kívül lehessen elindítani.

A PAM különleges moduljai

A „pam_cracklib" modul

A unix modul password elemének kiegészítésére szolgál. Jóval finomabb jelszóbonyolultság-ellenőrzést tesz lehetővé. A unix modul obscure kapcsolóknál említetteken kívül képes a szótári szavakon alapuló jelszavak kiszűrésére is. Kielégítő működéséhez egy megfelelő szavakat tartalmazó szótár szükséges, amit a legegyszerűbben oly módon állíthatunk elő, ha nagyobb mennyiségű levelezési listatárat szedünk össze, majd szavakra bontjuk. Célszerű olyan csomagokat is gyűjteni, amelyben a népek ékezettel leveleznek, mivel a felhasználók előszeretettel tesznek ékezetes szavakat a jelszavukba. Továbbá célravezető összeszedni a felhasználók, valamint kisállataik és szeretteik adatait. A 4. listán látható (24. CD Magazin/Konnyu könyvtár) egyszerű kis Perl-programocska segítségével a szövegállományokat szavakká daraboljuk.
A program a bemenetén a tiszta szövegállományokat várja (kismértékben akár HTML-lapokat is, bár ettől leendő adatbázisunk feleslegesen hízik), és az adatok a kimenetén szavakra darabolva érkeznek. Az adatbázis a Debianon az alábbi parancsösszetétellel állítható elő:

```
cat sok sz vegÆllomÆny neve |
      ➥mini_splitter | sort -u |
   crack_packer
      ➥/var/cache/cracklib/cracklib_dict
```

Ezzel előállítottuk a szóadatbázist, amelyet a későbbiekben ésszerű rendszeresen frissíteni. A modul ellenőrzi, hogy a megadott jelszó nem képezhető-e valamelyik szótári szóból a kis- és nagybetűk valamilyen kombinációjával.
A modul a PAM password elemében működik, használata egyszerűsítve a következő: a felhasználó által megadott jelszó minden karaktere egy pontot ér, továbbá minden különböző karakterosztályba tartozó karakter egy jutalompontnyit számít. A rendszer számára meghatározhatjuk, hogy egy adott karakterosztályra legfeljebb mennyi jutalompontot adjon. Az ismert osztályok: kisbetű (lower), nagybetű (upper), számjegy (digit) és egyéb (other). A jutalompontok hangolása a következő értékek beállításával történik:
dcredit=*N*; ucredit=*N*; lcredit=*N*; ocredit=*N*,
Az *N* az adott osztály karaktereire adható legmagasabb pluszpontok száma. Amennyiben a jelszóban megadott szám alatti vagy azzal megegyező számú adott osztályú karakter szerepel, mindegyikükért egy pluszpont jár. A karakterszámból adódó és a jutalompontok összegének legkisebbikét a minlen=*N* értékkel szabályozhatjuk. Az *N* értéke a megengedhető legkevesebb plusz egy. Így a következő beállításokkal:
dcredit=2 ucredit=1 lcredit=1 ocredit=2 minlen=12
csak olyan jelszó lesz elfogadható, amely vagy legkevesebb 10 kisbetűből áll, vagy ha van benne nagybetű, akkor nem rövidebb, mint 9 karakter; vagy ha van benne két számjegy és nagybetű, akkor nem rövidebb, mint 7 karakter és így tovább. A régi és új jelszó elvárt különbsége a difok=*N* értékkel állítható be. A segítségével megadható, hogy egy adott jelszóban hány karaktert kell mindenképpen lecserélni. Alapbeállítása tíz, de ehhez még egy újabb szabály adódik: amennyiben a jelszó karaktereinek legkevesebb a fele lecserélődik, felülbírálja az itteni beállítást, és a jelszó megfelel.

A „pam_ldap" modul

Nagyobb hálózatok estén gyakran felbukkanó gond, hogy a felhasználók a munkahelyek között vándorolnak, de mindenhol a megszokott munkakörnyezetet szeretnék látni. A rendszergazdák számára komoly nehézség lehet sok felhasználó együttes kezelése. Ilyen esetekben célszerű az LDAP-modul alkalmazása. A felhasználók adatait egy központi LDAP-kiszolgálón kell tárolni, így a tetszőleges munkaállomásra a megszokott jelszavukkal léphetnek be. A teljes támogatáshoz ne felejtsük el az nsswitch könyvtárakat sem áthangolni [5.]. A munkakönyvtárak átvitelére valamilyen hálózati állományrendszer is megfelel. Erre a célra jelenleg az NFS a legelterjedtebb megoldás, ami azonban biztonsági szempontból erősen megkérdőjelezhető, ezért használata kizárólag olyan környezetben fogadható el, ahol az ügyfelek tökéletesen megbízhatók – vagyis szinte sehol. Jelenleg a legésszerűbb a felhasználók munkakönyvtárait SSL-Sambán keresztül kiajánlani, így lehetővé válik a biztonságos csatlakozás. A kis kitérő után térjünk vissza a központi felhasználóazonosításhoz.
Az LDAP-modul használatához először is szükségünk lesz LDAP-kiszolgálóra, amelyen a felhasználók adatait tároljuk. Mi az OpenLDAP 2.0.14-es változatát használtuk. A telepítés Debian Woody rendszeren a megszokott apt-get parancs segítségével egyszerű (a csomag neve ***slapd***). Amennyiben a leendő LDAP szerkezetét előre megterveztük, telepítés közben létre lehet hozni a háttéradatbázist. A rendszer adatainak átemelésére tökéletesen alkalmas a PADL cég által fejlesztett MigrationTools nevű eszköz [6.]. Az OpenLDAP-nál az alapbeállítást kissé módosítani kellett, hogy a megfelelő sémameghatározásokat is betöltse.
Ésszerű a hozzáférést is szabályozni, mert az alaptelepítés bejelentkezés nélkül is olvasási jogot ad. A kissé paranoiásabb beállítás megfelelő része valahogy így fest:

```
access to attribute=userPassword
     by dn="cn=admin,o=Andrews,c=HU" write
     by anonymous auth
     by self write
     by * none

access to *
     by dn="cn=admin,o=Andrews,c=HU" write
     by self read
     by * none
```

Ezután megkezdődhet az LDAP-modul üzembeállítása, amely a */etc/ldap.conf* állományon keresztül zajlik. Lássuk az állomány tartalmát!

```
# Az LDAP-kiszolgÆl  neve
host tensor.andrews

# A keresØs kiindul pontjÆnak DN-je
base ou=People,o=Andrews,c=HU

# A rendszergazda DN-je (a jelszava a
# /etc/ldap.secret ÆllomÆnyban talÆlhat ,
# amely a libpam-ldap csomag telep tØsekor
# kit ltØsre ker l)
rootbinddn cn=root, o=Andrews, c=hu

# DN nevØben keres a pam_ldap modul.
```

```
binddn cn=admin, o=Andrews, c=hu
bindpw ubertitkosjelszo

# A jelszavak tÆrolÆsi formÆja
pam_password md5
```

A PAM beállítóállományába kerülő sorokat példánkban adjuk meg.

Egy példarendszer beállításai

Példánk tárgya egy kisebb hálózat egyik felhasználói gépe legyen. A felhasználók gyakran vándorolnak a gépek között, így azonosításukat LDAP-on keresztül oldjuk meg. A felhasználók a rendszert a konzolról való belépéssel érik el, távolról csak a rendszergazdák léphetnek be SSH segítségével. Megmutatjuk a login és az SSH PAM-modul beállításait.
A `login` modul beállításai csak annyiban térnek el a megszokottól, hogy a felhasználókat LDAP-ból is lehet azonosítani. Amennyiben a felhasználó az LDAP-modul segítségével azonosította magát, beengedjük, ha nem, megpróbáljuk a helyi felhasználói adatbázisból azonosítani. Ezt egészíti ki, hogy a `pam_listfile` modul használatával bizonyos felhasználók rendszerről való időleges kitiltását is lehetővé tesszük. Ennek kezelésére az 5. listában (24. CD Magazin/Konnyu könyvtár) található egyszerű kis Perl-program szolgál. A segítségével kilistázhatjuk a tiltott felhasználókat, felvehetünk és törölhetünk tiltást. Használatának megismeréséhez használjuk a `-h` kapcsolót.
Példarendszerünkön a */etc/pam.d/login* így néz ki:

```
# PAM beÆll t Ællomány a 'login' szolgÆltatÆshoz

auth    requisite  pam_listfile.so item=user
➥sense=deny file=/etc/security/deny_users
➥onerr=fail
auth    requisite  pam_securetty.so
auth    required   pam_nologin.so
auth    required   pam_env.so
auth    sufficient pam_ldap.so
auth    required   pam_unix.so try_first_pass

account     required   pam_unix.so

session     required   pam_unix.so
session     required   pam_limits.so
session     optional   pam_lastlog.so
session     optional   pam_motd.so
session     optional   pam_mail.so standard noenv

password    required   pam_cracklib.so retry=3
minlen=6 difok=3
password    required   pam_unix.so use_authtok md5
password    required   pam_ldap.so try_first_pass*
```

Az SSH beállítása a Debian által feltelepítettől annyiban tér el, hogy fel lett véve egy `listfile` modul, amely kizárólag a */etc/security/admins* állományban felsorolt felhasználókat engedi be. Az SSH PAM-állománya:

```
# PAM beÆll t Ællomány az 'ssh' szolgÆltatÆshoz
auth    requisite  pam_listfile.so item=user
➥sense=allow file=/etc/security/admins onerr=fail
auth    required   pam_nologin.so
auth    required   pam_env.so
auth    sufficient pam_ldap.so
auth    required   pam_unix.so

account     sufficient pam_ldap.so
account     required   pam_unix.so

session     required   pam_unix.so
session     optional   pam_lastlog.so
session     optional   pam_motd.so
session     optional   pam_mail.so standard
session     required   pam_limits.so

password    required   pam_cracklib.so retry=3
➥minlen=6 difok=3
password    required   pam_unix.so use_authtok md5
password    sufficient pam_ldap.so try_first_pass
```

Amennyiben a felhasználó a rendszeren jelszót változtat, az LDAP-ban is meg kell változtatni. Ehhez a `passwd` PAM-beállításait a következőképpen kellett módosítani:

```
# PAM beÆll t Ællomány a 'passwd' szolgÆltatÆshoz

password    required   pam_cracklib.so retry=3
➥minlen=6 difok=3
password    required   pam_unix.so use_authtok
➥md5
password    sufficient pam_ldap.so
➥try_first_pass
```

Ezzel az alapbeállításokat megválasztottuk, a finomhangolást mindenkinek az ízlésére bízzuk. A részletek és mellékhatások tekintetében nézzük meg a PAM-rendszer leírását [7.]. Azon szolgáltatásokról, amelyekről hely hiányában bővebben nem tudtunk szólni, elegendő adatot találunk a Linux PAM hivatalos honlapján [8.], továbbá számos egyéb hasznos modulra is ráakadhatunk. Mindenkinek hasznos keresgélést kívánunk!

Hivatkozásjegyzék

[1.] A ***shadow*** állomány formátuma - shadow (5)
[2.] A `crypt` eljárás leírása – crypt (3)
[3.] RADIUS ➲ http://www.gnu.org/software/radius/radius.html
[4.] Kerberos ➲ http://web.mit.edu/kerberos/www/
[5.] Authenticating with LDAP using Openldap and PAM
➲ http://www.imaginator.com/~simon/ldap/
[6.] plain to lpdap migration tools
➲ ftp://ftp.padl.com/pub/MigrationTools.tar.gz
[7.] A PAM-rendszer felhasználói kézikönyve
➲ http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/
[8.] A Linux PAM-rendszer hivatalos honlapja
➲ http://www.kernel.org/pub/linux/libs/pam/

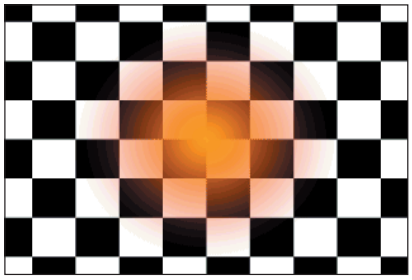
Mátó Péter (atya@andrews.hu), informatikus mérnök és tanár. Biztonsági rendszerek ellenőrzésével és telepítésével, valamint oktatással foglalkozik. 1995-ben találkozott először linuxos rendszerrel. Ha teheti, kirándul vagy olvas.

Borbély Zoltán (bozo@andrews.hu), okleveles mérnök-informatikus. Főként Linuxon futó számítógépes biztonsági rendszerek tervezésével és fejlesztésével foglalkozik.
A 1.0.9-es rendszermag ideje óta linuxozik. Szabadidejét barátaival tölti.

PoV-Ray ismeretek (6. rész)

Folytatjuk az anyagmódosítókkal való ismerkedést, hiszen most érkeztünk el az egyik olyan módosítóhoz, amellyel a leglátványosabb hatásokat hozhatjuk létre.

Részecskerendszerek segítségével modellezhetünk például robbanást, tüzet, füstöt és párát, továbbá felhőt és porfelhőt is. Az előbbieket természetesen különféle mintázatok felhasználásával is elkészíthetjük, de az igazán valósághű megjelenítéshez inkább a részecskerendszerek felelnek meg. Ismerkedjünk meg hát közelebbről a részecskerendszerek lelkivilágával – már amennyire e cikk keretei megengedik.
Első lépésben tűzhöz hasonló hatást próbálunk meg létrehozni, ehhez nyújt jó kiindulási alapot egy gömb. Az első fogalom, amivel meg kell ismerkednünk, a következő: tárolóobjektumnak nevezzük azt az objektumot, amit a részecskerendszer kiindulási formájaként határozunk meg. Minden részecskerendszernek szüksége van egy tárolóobjektumra, ami nem jelent nagy újdonságot, hiszen az anyagokat eddig is egy-egy tárgyhoz rendeltük hozzá. Fontosnak tartom megjegyezni, hogy a részecskerendszerek láthatóvá tételéhez a hordozó tárgynak áttetszőnek kell lennie, és a megfelelő hatás eléréséhez üreges test alkalmazása szükséges. Lássunk egy egyszerű példát, amiben a tűz kezdeti állapotait figyelhetjük meg. Itt részecskerendszereknek azt a típusát használjuk fel, amely egyszerűen csak fényt bocsát ki, ám az egyes részecskék a szomszédjukból sugárzó fényt nem nyelik el. Létrehozunk egy egyszerű jelenetet, amelyben a majdani tüzet megtestesítő gömb a tér középpontjában helyezkedik el, és fényforrással világítjuk meg. Természetesen ehhez a kamerát is meg kell alkotnunk. Láthatjuk tehát, hogy a gömb áttetsző, hiszen az `rgbt` kulcsszóval határoztuk meg a színét, amelynek utolsó értéke az adott szín átlátszóságát szabályozza. Ezenkívül a `hollow` kulcsszót is használtuk, amelynek hatására testünk üreges lesz. Felmerülhet a kérdés, hogy miért kellett a padlót alkotó síkot is üregessé tenni.
Ennek az az oka, hogy a részecskerendszer nem lehet egy másik tömör testen belül, mert ebben az esetben nem látható. Ha a síkot nem akarjuk tömörré tenni, a másik megoldás, hogy a kameránál a `negative` kulcsszót is megadjuk. Most tekintsük át, milyen ismeretlen szavak maradtak még számunkra a részecskerendszer meghatározásában.
A következő ilyen kifejezés az `emitting`, amelynek hatására a részecskék fényforrásként működnek. Célunk eléréséhez ez lesz a megfelelő típus, a további lehetőségek bemutatását a későbbiekben folytatom. A `spherical_mapping` adja a PoV-Ray tudtára, hogy ez esetben gömbszerű leképezés szükséges, ami nyilvánvaló is, hiszen a hordozótest is



1. kép Tűzfészek

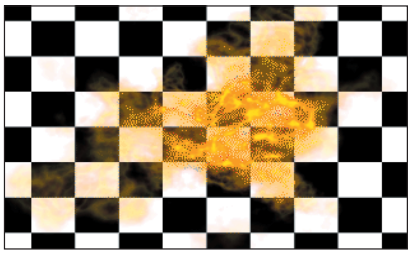
gömb. Egy részecskerendszernél az egyes részecskék eloszlását egy úgynevezett sűrűségfüggvény határozza meg, amely leírja, hogy mennyi részecskét találhatunk egy adott helyen. A PoV-Rayben pontos matematikai függvény helyett előre meghatározott eloszlásokat alkalmazhatunk. Ezek közül az egyik a `linear` kulcsszó által meghatározott egyenletes eloszlás. A részecskék száma minden esetben a koordinátarendszer középpontjában a legnagyobb, így előfordulhat, hogy egy tárgyat olyan koordinátákra helyezünk, ahol már nincsenek részecskék. Ilyenkor semmiféle tüzet, ködöt vagy felhőt nem láthatunk, de ennek tudatában a tárgyat szerencsére már a létrehozásnál a koordinátarendszer középpontjába helyezhetjük és később az egészet (a tárgyat az anyagokkal és a részecskerendszerekkel együtt) a kívánt helyzetbe tolhatjuk el. Az egyenletes eloszlásnak köszönhetően a részecskesűrűség a gömb középpontjában lesz a legnagyobb, a felszínen pedig 0.
A fenti példában gömbünk (ahogyan a „szófogadó” tűzhöz illik) középen átlátszóság nélkül sárga színű lesz, a felszínen pedig vörös és szinte alig látható. Az *1. kép* ezt az állapotot mutatja meg.
Első példánk utolsó ismeretlen szava a `samples` (a minták száma) névre hallgat és egy egész szám követi. Ezzel a számmal határozzuk meg, hogy a számítások során mennyi fénysugár haladjon keresztül a rendszeren, amíg a PoV-Ray kiszámítja a képet. Magasabb értékek esetén szebb eredményhez jutunk, kiszámítása azonban hosszabb időt vesz igénybe. Ha egy-egy részecskerendszer kiszámítása során teljesen szokatlan eredményt kapunk, érdemes ezt az értéket megnövelni – a legtöbb esetben a 10-es mintaszámérték megfelelő.
A fenti képen a színek meglehetősen halványak. Túl sok látszik a háttérből, így az első példánk szolgáltatta eredmény nem igazán hasonlít a tűzről, robbanásról alkotott képünkhöz. A színeket élénkebbé tehetjük, ha az objektum nagyobb sűrűségű részeinek átlátszóságát csökkentjük. Az alábbi részletbe a „-1”-es átlátszóság azért került bele, hogy a kapott eredmény jobban hasonlítson a tűzhöz.

```
Color_map {
   [ 0 color rgbt <1,0,0,1> ]
   [ 1 color rgbt <1,1,0,-1>]
}
```

Ennek az egyszerű változtatásnak fényesebb tűzkezdemény lesz az eredménye. A valóságban azonban ritkán láthatunk ilyen szabályos tűzgömböt, és én sem akadtam össze hasonló jelenséggel.
A valódi tűzgömb minden esetben rendezetlenebb formákat alkot, tehát a valósághűség érdekében a modellezés során nekünk is rendezetlenséget kell szimulálnunk. Erre már sorozatunk korábbi részeiben is láthattunk példát, ezért ismerősként köszönthetjük az alkalmazandó `turbulence` kulcsszót. Most nem ismétlem meg a listát, mivel mindössze egyetlen sort kell beszúrnunk a `linear` szó után (elé is megtehetjük: a `halo`-n belül és a `color_map`-on kívül), amivel kissé áttekinthetetlenebb kinézetet adhatunk a készülő

tűzgömbnek. Amennyiben a `turbulence 1.5` tartalmú sort a megfelelő helyre szúrjuk be, egy újabb leképezés után máris a világító labda nagymértékű átalakulásának lehetünk szemtanúi. Ennek kiszámításához több idő szükséges, mert minden részecske véletlen, örvényszerű elmozdulást kap. Amikor a részecskerendszert rendezetlenné tesszük, gyakran előfordul, hogy a részecskék a hordozó objektumon kívülre kerülnek. Elkerülésére magát a rendszert kell átméreteznünk, hiszen ha az objektum méretét változtatnánk meg, a részecskék elhelyezkedése nem sokat változna. A részecskerendszert a 2. listán (➲ 24. CD, Magazin/Pov-Ray könyvtár) látható módon méretezhetjük át, tehát úgy, hogy még a részecskerendszer meghatározásakor a PoV-Raynek a méretek megváltoztatására adunk utasítást.
A fenti kérdés megoldására más módszert is használhatunk: a gömböt készítsük nagyobb méretűre, és mielőtt még meghatároznánk a részecskerendszert, méretezzük is át. Így a részecskerendszer mérete már nem változik. Ezeket a megoldásokat csak akkor alkalmazhatjuk, ha a rendszert gömbszerű (`spherical_mapping`) vagy kockára történő leképezéssel készítjük el.
Amennyiben elsajátítjuk a `frequency` kulcsszó használatát, a tűzgömböt még élethűbbé varázsolhatjuk. Most sem érdemes a matematikai háttérrel foglalkoznunk, mert nem vinne közelebb a lehetőség gyakorlati használatához, elegendő annyit tudnunk, hogy ez az érték határozza meg, hogy a tárolóobjektumon belül a színek hányszor ismétlődjenek. Vigyáznunk kell azonban, mert ha a részecskerendszer színeit nem ismétlődőként adjuk meg (amikor az első és az utolsó érték megegyezik), a végeredményül kapott képen csúnya ugrásokat láthatunk.
Miután ilyen szép tüzet készítettünk (*2. kép*), megeshet, hogy az általunk alkotott képet valamilyen különleges hatással még tetszetősebbé szeretnénk varázsolni. Tegyük a képet egy kicsit szokatlanná: változtassuk meg a tűzgömb színét! Célunk, hogy a gömbön ne piros-sárga átmenetet lássunk, hanem zöld-pirosat. Ennek legegyszerűbb módja, hogy a részecskerendszer színét módosítjuk. Ekkor azonban azt fogjuk tapasztalni, hogy a gömb közepén nem a várva-várt zöld szín jelenik meg, hanem a zöld és piros keveréke: a sárga. Nem egészen ezt szerettük volna, de nem adhatjuk fel ilyen könnyen. Íme a kiváló alkalom a `glowing halo` megismerésére.

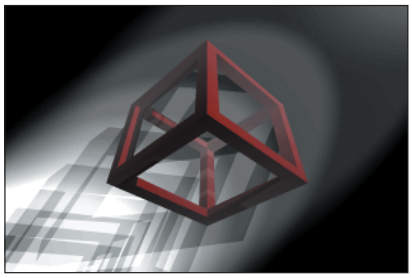
Ez a részecskerendszer-típus abban különbözik az előzőtől, hogy az egyes részecskék fényt bocsátanak ki – elkerülve ezzel a színek keveredését. Csak annyit kell az előző példánkon megváltoztatnunk, hogy az `emitting` kulcsszó helyett a `glowing`-ot használjuk, minden másban megegyezik az előbbiekben tárgyalt típussal, így korábbi ismereteinkre támaszkodva bátran kalandozhatunk a részecskék és részecskerendszerek világában.



2. kép Túl valósághűre sikeredett…

A következő egyszerű részecskerendszer a felhők és a párás környezet modellezésére szolgáló `attenuating halo`.
E típus azért alkalmas felhők leképezésére, mert a rajta áthaladó fény egy részét elnyeli, ugyanis a PoV-Ray a leképezés során egy adott képpont értékét nem a pontot körülvevő részecskék alapján számítja ki, hanem az objektumon áthaladó fénysugár mentén minden részecskét figyelembe vesz. Ezeket a színeket a `color_map` részben határozzuk meg. Most lássunk egy olyan példát, amit felhőink kiindulási alapjaként a későbbiekben is felhasználhatunk (2. lista).
A felhők általában nem vörös színűek, de példánkban a jobb láthatóság kedvéért piros-fehér átmenetet használunk, ugyanis fekete-fehér háttér előtt nem lenne célszerű fehér felhőket megjeleníteni.
Előbbi példákban csak a részecskerendszer méretét változtattuk, de most lássunk arra is megoldást, hogy a hordozóobjektumot méretezzük át. A tűzgömbnél alkalmazott minőségjavító megoldások (a fényesség növelése, rendezetlenség alkalmazása) az eredményt itt is vonzóbbá teszik, de a felhők általában nem gömbalakúak. Most a hordozóobjektumot kell átméreteznünk.
Ha valóban élethű felhőket szeretnénk előállítani, egy hordozóobjektumon belül több részecskerendszert is alkalmazhatunk. Nem kell mást tennünk, csak egymás után úgy meghatározni a rendszereket, hogy csak a helyzetükben különbözzenek egymástól. Természetesen ilyenkor a rendszerek alapjául szolgáló tárgyat is nagyobbra kell készítenünk.

A következő részecskerendszer a fénysugarak poros közegen történő áthaladásának megjelenítésére alkalmas. Gyakran láthatunk ilyet például a délutáni erdőben, amikor a fény átszűrődik a lombok között, vagy a padlásokon, amikor a napsugarak a cserepek közötti réseken szűrődnek be. Ez a részecskerendszer eléggé összetett, nem csupán elnyeli a fény egy részét, de az a benne található részecskéken szét is szóródik. Ennek eredményeképpen a tárgyon áthaladó fénysugarak láthatóvá válnak.
Ezt a részecskerendszert ugyancsak egy példán keresztül világíthatjuk meg a legjobban. Először meghatározunk egy irányított fényforrást és egy tárgyat, utóbbin fogjuk szemlélni a keresztülhaladó fényt.
Ezután létre kell hozni egy olyan tárgyat, ami a részecskerendszert fogja tartalmazni, jelen esetben erre a célra egy másik dobozt alkalmazunk. A részecskerendszer sűrűsége állandó, amit a rendszeren belül a `max_value` kulcsszóval határozhatunk meg. Ennek alapértéke 1, és amennyiben ezt a sűrűségfüggvényt használjuk, bármelyik leképezési mód (gömbszerű, kockára való



3. kép A kezdeti állapotok…

leképezés stb.) ugyanazt az eredményt adja. Lássuk, hogyan adhatjuk meg a részecskerendszert (7. lista ➲ 24. CD, Magazin/Pov-Ray könyvtár)!
Amint a *3. képen* is láthatjuk, az eredmény még nem tökéletes, mert a kép túl világos, a háttér alig látható és a por is túl sűrű.
A hordozótárgyként megadott kockában a részecskék sűrűsége állandó, alapértelmezetten pedig 1. Ez azt jelenti, hogy a színek meghatározásánál a részecskerendszer sűrűségét és színét csak az 1-es helyen lévő érték fogja befolyásolni.
A következő példában ezt az átlátszóságot 0,7-re állítjuk, így a por ritkább és az eredmény is szebb lesz.

```
...
...
    color_map {
```

```
      [ 0 <1, 1, 1, 1> ]
      [ 1 <1, 1, 1, 0.7> ]
    }
    samples 10
  }
...
...
```

Az eddigi módosítások ellenére még mindig akad egy kis gond az árnyékokkal. Tegyük egy kicsit elmosódottabbá az éles árnyékokat! Többféleképpen is megoldhatjuk: alkalmazhatunk véletlenszerű zajt a részecskerendszerben a `jitter` kulcsszó megadásával. A másik lehetőségünk a felül-mintavételezés, ekkor a nagyobb fényességváltozásokat finomítjuk (az `aa_treshold` és az `aa_level` kulcsszó segítségével).
A harmadik mód, amikor a teljes képre magasabb mintavételezési értéket alkalmazunk. Mivel ez utóbbi a leglassabb eljárás, először a többi módszert próbáljuk ki. A részecskerendszer meghatározásán belül általánosságban a véletlen zajt és a helyi felül-mintavételezést használjuk a következő módon:

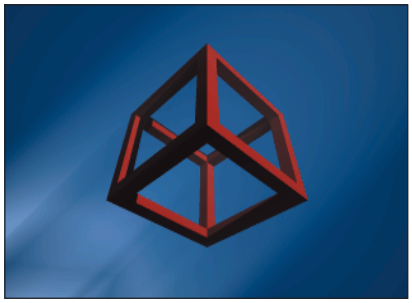
```
...
...
    }
    samples 10
    aa_level 3
    aa_threshold 0.2
    jitter 0.1
    }
  }
...
...
```

Most már szinte tökéletes a poron áthaladó fénysugár megjelenítése, de nagyon ritkán találkozunk olyan hellyel, ahol így áll a levegő, és a por ennyire egyenletesen oszlik el a térben. Kavarjunk egy kis szellőt a `turbulence` utasítás alkalmazásával, és máris elégedettek lehetünk az eredménnyel:

```
...
...   box_mapping
    linear
    turbulence 1
    color_map {
...
...
```

Vegyük észre, hogy nem használtuk az állandó sűrűséget meghatározó `constant` sűrűségfüggvényt, hanem helyette a `linear` kulcsszó meghatározta egyenletes eloszlást adtuk meg.

Ennek oka, hogy állandó sűrűségű térben nem lenne értelme a véletlenszerű változtatásoknak, a részecskesűrűség nem változna. Megjegyzendő, hogy a `turbulence` érték típusa vektor; a felhasználásával látványos hatásokat érhetünk el, ha az egyik irányban nagyobb értéket adunk meg, mint a másik két koordinátatengely mentén. Így készíthetünk például vízesést vagy más áramló rendszert.



4. kép Fény a porban…

Természetesen a por számára nemcsak fehér és szürke színeket adhatunk meg, hanem az alábbi részlet alapján akár a fehértől indulva – a képzelőerőnk szabta határokig – bármilyen színt. Fontos olyan színértékeket megadni, amelyek bizonyos színeket kiszűrnek, ezt a `rgbf` kulcsszóval tehetjük meg. Mivel azonban a részecskerendszerben a színeknek átlátszóknak is kell lenniük, a színek meghatározása során inkább a `rgbft` szót használjuk.

```
...
    color_map {
      [ 0 color rgbft <1, 0,
        ➥0, 0.5, 1.0> ]
      [ 1 color rgbft <1, 0,
        ➥0, 0.5, 0.7> ]
    }
...
...
```

Mielőtt a végére érnénk a részecskerendszerekkel való ismerkedésnek, fel kell hívnom a figyelmet néhány dologra: a részecskerendszert minden tárgyhoz a következő formában adjuk meg:

```
OBJETKUM {
  texture {
    pigment {...}
    normal {...}
    finish {...}
    halo {...}
  }
  hollow
}
```

Nem használhatók a `pigment`, `color_map`, `pigment_map`, `texture_map` és `material_map` utasításokon belül. Amennyiben többrétegű mintázatot szeretnénk használni, a részecskerendszert mindig a legalsó rétegben kell meghatároznunk, mely rétegnek természetesen átlátszónak kell lennie. Szintén ne feledkezzünk meg róla, hogy egymást átfedő tárolóobjektumok esetében a PoV-Ray az eredményt nem képes megfelelő módon kiszámítani. Ilyenkor minden tárolóobjektumot a többitől függetlenül számol ki, és az eredmény összeadódik. Az ebből származó hibák elkerülhetők, ha megfelelően nagy méretű tárolóobjektumot adunk meg.
További hiányosság, hogy a többféle színtérképpel (`color_map`) létrehozott `attenuating` típusú részecskerendszer, amelyet a felhők készítésénél tárgyaltunk, jelenleg nem használható.
Amint azt a leírás elején említettem, a kamera látóterében lévő objektumoknak üregesnek kell lenniük, ezt a `hollow` kulcsszó teszi lehetővé.
Fontos megjegyeznünk, hogy a `scale` kulcsszót a megfelelő helyen kell alkalmaznunk. Amennyiben a hordozó tárgyat át szeretnénk méretezni, például akkor, amikor a részecskéket rendezetlenné (`turbulence` kulcsszó) tesszük, az átméretezést még a rendszer meghatározása előtt el kell végezni; míg ha a részecskerendszer méretét szeretnénk megváltoztatni, a `scale` utasítást a rendszer meghatározásán belül kell használnunk.
Végül az ismétlés kedvéért jegyezzük meg, hogy a rendezetlenség nincs hatással az állandó sűrűségű rendszerre, tehát a `turbulence` és a `constant` kulcsszó együttes alkalmazása nincs hatással a részecskék eloszlásváltozására.
Végül nézzük meg a korábban elkezdett tárgyat, amely most poros térben lebeg, és szabadon alkalmazzuk rá új ismereteinket!

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.

Bevezetés a Tkinter használatába (2. rész)

Egy számológép elkészítése során is fedezhetünk fel új dolgokat: tudtad, hogyha véletlenül meglököd az egeredet, máris folyamatok egész sorát indíthatod el?

Tkinter-tanfolyamunk második részéhez érkeztünk. Az előző részben már láthattuk, hogyan néz ki egy egyszerű Tkinter-alkalmazás, ezúttal pedig egy kicsit bonyolultabb feladattal, egy számológépes példával folytatjuk. A „Szia Világ!"-os példa sokat elmond, amikor egy programnyelv vagy rendszer alapjaival ismerkedünk, a Tkinter viszont túlmutat ezen; és mivel terjedőben van a szokás, hogy a grafikus alkalmazások fejlesztésére szánt rendszereket egy-egy számológépes példán keresztül ismertetik meg a felhasználókkal, tegyünk így mi is. Eközben fény derül olyan titkokra, minthogy miként tudunk egy szövegbeviteli mezőt programsorból módosítani, de az is kiderül, mi minden történik olyankor, amikor látszólag nem történik semmi, csak az egerünkkel bóklászunk a képernyőn. Vágjunk bele!

Ismerkedés a számológéppel

Számológépet már mindenki látott, ezért a feladattal nagyjából tisztában vagyunk. Vegyük sorra, mi minden szükséges ahhoz, hogy az elképzeléseinket valóra váltva egy egyszerű számológép jelenjen meg a képernyőn, amely összead és kivon, továbbá a másik két alapművelettel is tisztában van. Ha a számítógép „fejével" gondolkodunk, mindjárt elakadunk, hiszen a „szegény" gép nem tudja, hogyan néz ki egy számológép, tehát pontról pontra mindent el kell neki magyaráznunk. Meg kell mondanunk például, hogy a gombok ne „csak úgy" megjelenjenek a képernyőn, hanem meg is lehessen őket nyomni; és azt is meg kell értetnünk, hogy olyankor mi történjen, ha meg is nyomjuk ezeket a gombokat. Tudnia kell, hol legyenek az egyes gombok, a többiről nem is beszélve. Nem olyan bonyolult ám ez, csak elsőre szokatlan, hiszen ezúttal olyan partnerrel akadtunk össze, aki nem biztos, hogy mindent azonnal ugyanúgy gondol, ahogyan mi elvárnánk.

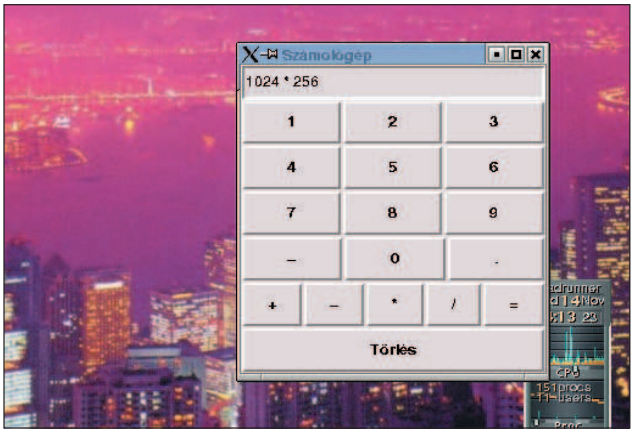
Első lépésben tehát az úgynevezett felhasználói felületet szükséges felépítenünk: ki kell találnunk, hogy milyen elemeket szeretnénk kitenni a képernyőre, és hol legyen a helyük. Esetünkben gombokra és kijelzőre lesz szükség. A következő lépésben pedig azt kell kifundálnunk, hogy az egyes gombokra kattintva mi történjék. Végül az sem árt, ha arra is felkészülünk, hogy mi legyen olyankor, ha a monitor előtt ülő felhasználó olyan dolgokat cselekszik, amelyre programunk nincs felkészülve: egyszerűen lépjünk ki valamilyen hibával a programból, ne is vegyük észre a hibát vagy tudassuk a felhasználóval, hogy rosszul csinált valamit?

Amennyiben mindezt kigondoltuk, a programtervezés nagy részén már túl is vagyunk, innentől már csak ujjgyakorlat az egész.

Az első nekifutás

Mint fentebb már említettem, számológépünkhöz két dologra lesz szükségünk: egy kijelzőre és sok-sok apró pici gombra, továbbá egy ablakra, ahol mindezt elhelyezhetjük. Az előző részben láthattuk, hogy egy gomb létrehozása csupán egy pillanatig tart: létrehozunk egy gombpéldányt és valamelyik felületkezelővel kitesszük a képernyőre. Szemléltetve:

```
w = Button(root, text= sz veg ,
           command=eljÆrÆs)
w.pack(side=LEFT, expand=YES, fill=BOTH)
```



Munkánk gyümölcse

Lambda kifejezések

Egy `lambda` kifejezés egy szokványos `def` függvénytől annyiban különbözik, hogy olyan helyeken is előfordulhat, ahol a `def` nem. Tartalma csak valamilyen egyszerű kifejezés lehet, bonyolultabb szerkezetek, mint `if` vagy `for`, nem. A `lambda` kifejezéseket elsősorban visszahívó függvényeknél használják. Visszatérési értéke mindig egy függvényobjektum, amelyet egy változóhoz rendelve hívhatunk meg. Az alábbi példa lefutása után a *b(18)* visszatérési értéke 36:

```
b = lambda a: a * 2
print b(18)
```

Minden `lambda` kifejezés saját névtérrel bír, amely nem azonos a hívó programrész névterével. Ezt a gondot egy trükkel szokták megoldani: a `lambda` értéklistájához azokat a változókat teszik hozzá, amelyekre a programrész névteréből szükségünk van. Számológépprogramunkban erre is találunk példát.

Az első sorban dől el, hogy mi lesz a gombunkra írva, és mi fog történni, ha valaki véletlenül rákattint. A `w` változó innentől kezdve az általunk megálmodott gombra mutat, ez azonban egyelőre csak a memóriában létezik. Ahhoz, hogy valódi, kattintható és látható gomb váljék belőle, ki kell tennünk a képernyőre. Ezt a nagyon fontos és cseppet sem elhanyagolható feladatot a Packer nevű felületkezelőre bízzuk. Ennek a `pack()` eljárásnak a dolga a gomb képernyőre történő helyezése, egészen pontosan az alkalmazásunk ablakába, lehetőség szerint a bal oldalra igazítva.

Számológép

```
1. from Tkinter import *
2.
3. class Calculator(Frame):
4.   def __init__(self):
5.   Frame.__init__(self)
6.   self.pack(expand=YES, fill=BOTH)
7.   self.master.title('SzÆmol gØp')
8.   self.error = 0
9.   self.ans = 0
10.
11.  display = StringVar()
12.  disp = Entry(self, relief=SUNKEN,
           textvariable=display)
13.  disp.pack(side=TOP, expand=YES,
     fill=BOTH)
14.
15.  for key in ("123", "456", "789", "-0."):
16.    fkey = frame(self, TOP)
17.    for char in key:
18.      button(fkey, char, lambda w=display,
         c=char, s=self: s.setscreen(w, c,
         1))
19.
20.  fops = frame(self, TOP)
21.  for char in "+-*/=":
22.    if char == '=':
23.      btn = button(fops, char)
24.      btn.bind('<ButtonRelease-1>',
25.        lambda e, s=self, w=display:
          s.calc(w))
26.    else:
27.      btn = button(fops, char,
28.        lambda w=display,
          c=' %s '%char, s=self:
          s.setscreen(w, c, 0))
29.
30.  clearF = frame(self, BOTTOM)
31.  button(clearF, 'T rlØs', lambda
     w=display, s=self: s.clear(w))
32.
33.  def setscreen(self, w, c, clear):
34.    if self.error == 0:
35.      if self.ans == 1:
36.        self.ans = 0
37.        if clear:
38.          w.set('')
39.      w.set(w.get() + c)
40.    else:
41.      self.clear(w)
42.      w.set(c)
43.
44.  def clear(self, w):
45.    w.set('')
46.    self.error = 0
47.
48.  def calc(self, display):
49.    try:
50.      if self.error == 0:
51.        display.set('eval(display.get())')
52.        self.ans = 1
53.      else:
54.        display.set("kattints a torlesre")
55.    except:
56.      display.set("HIBA")
57.      self.error = 1;
58.
59. def frame(root, side):
60.  w = Frame(root)
61.  w.pack(side=side, expand=YES, fill=BOTH)
62.  return w
63.
64. def button(root, text, command=None):
65.  w = Button(root, text=text,
         command=command)
66.  w.pack(side=LEFT, expand=YES, fill=BOTH)
67.  return w
68.
69. if __name__ == '__main__':
70.  Calculator().mainloop()
```

Egy felületkezelőnek ennél azonban jóval több dologra kell ügyelnie. Kezdetben az ablak mérete pontosan akkora kell legyen, hogy minden elem, amelyet belepakoltunk, elférjen benne és látszódjék, hacsak másképpen nem rendelkeztünk. Amennyiben az ablakunk méretét növeljük, gombjainknak több hely áll rendelkezésére, mint amennyire szükségünk van, és ilyenkor ugyancsak a felületkezelő feladata, hogy megmondja, miképpen változzon meg az ablakban található gomb vagy bármilyen más elem mérete és elhelyezkedése, hogy a felhasználók igényeinek a legjobban megfeleljen. Ebben az esetben feltételezzük, hogyha a felhasználó növeli az ablak méretét, bizonyára nagyobb gombokra van szüksége, így gombunk tulajdonságai közé vesszük, hogy a rendelkezésre álló helyet mindkét irányban töltse ki (`fill=BOTH`), és nőjön együtt az ablakkal (`expand=YES`). A `side` értékkel mondhatjuk meg, hogy a Packer az ablak melyik oldalára igazítsa a gombunkat.

A Packeren kívül két másik felületkezelő is létezik: a Grid és a Placer. A Grid az ablakot rácsokra osztja, és nekünk csak azt kell megadnunk, hogy az egyes elemek mely rácspontokba kerüljenek. Ez a felületkezelő a legjobban talán a HTML-ből ismert TABLE-höz fogható. Használata a Packerhez hasonlóan könnyű és egyszerű, és igazából csak szokás kérdése, hogy melyiket kedveljük jobban – viszont kétségkívül igaz, hogy olyan feladatok is akadnak, amelyek az egyikkel vagy a másikkal könnyebben kivitelezhetők.

A felületkezelők feketebáránya a legegyszerűbb, mégis a legtöbb odafigyelést igénylő ***Placer***, amelynek segítségével elemeinket pontosan igazítva helyezhetjük el az ablakban. Fontos, hogy egy kereten (vagy ablakon) belül csak egyetlen felületkezelő használható! Ez józan ésszel is belátható, hiszen mindannyian az elemek elhelyezéséért felelősek. Tegyük fel, hogy egy ablakot már rácsokra osztottunk, ilyenkor már nem pakolgathatunk benne ide-oda bármit!

Eseménykezelés

Bármilyen rendszer alatt dolgozunk is, legyen az X vagy Windows, a háttérben az egér egyetlen elmozdulását is események egész sora követi. Így van ez mindennel: esemény keletkezik, ha lenyomunk egy gombot, ha kattintunk valahol, sőt még akkor is, ha véletlenül meglökjük az egeret. A futó rendszer dönti el, hogy a pillanatnyilag zajló esemény az alkalmazásunkra tartozik-e vagy teljességgel lényegtelen. Az eseményeknek három fő típusa van: a billentyűzettel, az egérkezeléssel és az ablak helyével, illetve méretével kapcsolatosak. Ezek közül a billentyűzettel összefüggő eseményfajta az egyetlen, amelynek elfogásához az ablakunknak kell az aktív ablaknak, azaz a beviteli fókusszal bíró ablaknak lennie.
Egy eseményt kétféle módon lehet elfogni: vagy közvetlenül a `bind()` eljárással, vagy – amennyiben gombról van szó – a ***command*** tulajdonság megadásával.

```
btn.bind('<ButtonRelease-1>', eseménykezelő)
```

Ebben a példában gombunk objektumához egy eseményt rendelünk. Kikötjük, hogy amennyiben a gombunkon valaki a bal oldali egérgombot nyomja le, hajtsa végre az eseménykezelő által hivatkozott eljárást. Ennek legelső értéke kötelező érvényű, a `bind()` ezen keresztül tudatja az eljárással, hogy milyen esemény történt valójában. Ha függvényünket a ***command*** tulajdonságon keresztül hívatjuk vissza, erre a kötelező értékre nincs szükség.
Az alábbiakban felsorolunk néhány fontosabb eseményazonosítót:

ANY-ENTER	Az egérmutató az elem területére lépett.
BUTTON-1	Az egyes egérgombot lenyomták az objektum területén.
BUTTONRELEASE-2	Az objektum területén a kettes egérgombot felengedték.
KEYPRESS	Lenyomtak egy billentyűt.
CONTROL-SHIFT-F1	Az adott elemen lenyomták az adott billentyűkombinációt.
CONFIGURE	Az ablak mérete vagy helyzete megváltozott.
FOCUSIN	Ablakunk lett az aktív ablak.
DESTROY	Programunk bezárás alatt van.
A	Lenyomott A betű.

Ha gyakran végrehajtódó eseményhez rendelünk eseménykezelőt, ügyeljünk rá, hogy egy nagyobb függvény nagyon lefoglalhatja a processzorunkat, ezért igyekezzük elkerülni.
Az eseménykezelők egy alkalmazásban több szinten is beállíthatók. Alapértelmezésben a beállítás csak egy adott elemre vonatkozik. Ezenkívül még három szint létezik: az alkalmazásszint, amely egy adott alkalmazás minden ablakára és elemére vonatkozik; az osztályszint, ami egy osztály összes kezdeti példányára vonatkozik; illetve a héjszint, amely a szülő alkalmazásablakra vonatkozik.
Egy-egy létrehozott esemény legelőször azon az elemen jelentkezik, amelyen az esemény keletkezett, és attól halad lefelé egészen az alkalmazásszintig, kivéve, ha közben valamelyik szinten úgy rendelkezünk, hogy az eseményt nem engedjük tovább.

Mindez a gyakorlatban

Most, amikor az elmélettel már nagyjából tisztában vagyunk, vessünk néhány pillantást számológépünk forráskódjára.
Ha a kódsorokat begépeljük és .PY kiterjesztéssel mentjük, a `python` paranccsal meghívva fárasztó gépelésünk eredményében már gyönyörködhetünk is.
Amennyiben közelebbi pillantást vetünk a programra, látható, hogy a gerincét a ***Calculator*** osztály alkotja, amelyben a lényeg igazából az `__init__()` eljárásba van sűrítve. Azt már tudjuk, hogy ez az az eljárás, amely objektumpéldányunk létrehozásakor önmagától lefut, ennek megfelelően a benne rejlők már az alkalmazás indulásakor végrehajtódnak. Az első ismeretlenbe a 12. sorban ütközünk, itt hozzuk létre alkalmazásunk kijelzőjét, amely valójában egy egyszerű szövegbeviteli mező, és a következő sorban található `pack()` eljárással tesszük ki a képernyőre. A `relief` értékkel a kijelző stílusát adhatjuk meg, ami ebben az esetben SUNKEN, azaz süllyesztett.
A `textvariable` értékkel azt a változót jelöljük ki, amelyet összekötünk a kijelzővel. Amennyiben a változón módosítunk, változik a kijelző tartalma, ami fordítva is igaznak fog bizonyulni.
A 15. sortól kezdődően alkalmazásunk gombjait rajzoljuk ki, a 17. sorban pedig a ***key*** karaktersorozatot bontjuk szét karakterekre, amelyeket kirajzolásuk után egy lambda kifejezésen keresztül a `setscreen()` eljárásra csatolunk vissza, így ha bármelyiket lenyomjuk, az adott érték a kijelzőn jelenik meg.
A 18. sorban meghívott `button()` eljárás csak hivatkozás egy alább bevezetett függvényre, amelyet nem szabad a ***Button*** osztállyal összekevernünk, utóbbi ugyanis egy objektumpéldánnyal térne vissza.
A 16. és 20. sorokban megint csak egy saját függvényen keresztül hozunk létre egy keretet. Mivel a Packer felületkezelőt használjuk, erre azért van szükség, hogy a Packer egyértelműen eldönthesse, hova kell az adott elemet helyeznie.
A 24. és 25. sorokban a `bind()` függvényt hívjuk meg, amellyel az egérgomblenyomás eseményéhez rendelünk hozzá egy `lambda` eljárást. Az eljárás első értéke (`e`) látszólag használaton kívüli, valójában viszont azért szükséges, mert a `bind()` ezen keresztül küldi el az `event` eseményváltozót.
Az 51. sorban az `eval()` függvényen keresztül a kijelző tartalmát kiértékeljük és az eredményt kiíratjuk.

Összegzés

Ebben a részben egy számológépes példán keresztül a Tkinter alapvető lehetőségeivel ismerkedtünk meg, amelyek felhasználásával egyszerű alkalmazások készítésére leszünk képesek. A példaprogram részeit begépelés után ajánlatos módosítgatni vagy akár bővíteni. Így biztosak lehetünk benne, hogy a frissen elsajátított ismereteket nem felejtjük el azonnal, és később is fel tudjuk használni.

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

A Python hivatalos honlapja ➲ http://www.python.org/
Tkinter- tanfolyam
➲ http://www.pythonware.com/library/tkinter/introduction/

Az ablakkezelők és a Linux

Telepítéskor a legtöbb Linux-változat két ablakkezelőt helyez előtérbe: a KDE-t és a Gnome-t. Az összes többi ablakkezelő általában csak lehetőségként választható, amely sajnálatos módon azt eredményezi, hogy a felhasználók többsége – főleg a kezdők – nem, vagy csak kis mértékben próbálkozik megismerkedni a többivel. Néhány éve, amikor Linuxszal kezdtem foglalkozni (RedHat 5.2-sel), ha jól emlékszem, a KDE és a Gnome még egyáltalán nem volt üzembiztosnak nevezhető, ezért nem is választhattam őket. Akkoriban a manapság szinte ismeretlen fvwm ablakkezelő és az AfterStep, valamint a WindowMaker képviselték az elfogadható választást. Azóta több, a témával kapcsolatos fordítást is készítettem a levelezőlistákon megismert sorstársakkal. A legtöbb Linux-változat esetén néhány egyszerűbb ablakkezelő külön is választható. Jelenleg a Debian SID-ben található alapablakkezelőket szeretném bemutatni, terjedelmi és egyéb okokból kifolyólag azonban csak röviden. A cikksorozat első lépésben az igen egyszerű felépítésű, eszközkészlet nélküli ablakkezelők ismertetését veszi célba. Ehhez a Debian SID-változata szolgáltatta az alapot, amely a ***táblázatban*** látható ablakkezelőket tartalmazza.
A szóban forgó ablakkezelőkről, mint említettem, helyenként csak meglehetősen szűkszavúan fogok nyilatkozni, mint látni fogjuk részben azért, mert csekély mozgásteret engednek.

Az alapszintű ablakkezelők

Ebbe a csoportba azokat a grafikus felületeket soroltam, amelyek kinézet, tudás, valamint kezelhetőség tekintetében is igen puritánok.

A twm

A `twm` az egyik legegyszerűbb ablakkezelő, hiszen indítás után csupán egyetlen menüt tudunk megjeleníteni, jó esetben talán egy ikonkezelőt is. Az ablakkezelő igényeinknek megfelelő átformálását csakis valamilyen szövegszerkesztő program segítségével végezhetjük el. A beállítóállományok nálam két helyen találhatók: a */etc/X11/twm/system.twmrc*-ben (ez rendszerszintű fájl, amelyben a menü központilag frissül), valamint a *~/.twmrc* (ez pedig a saját beállításaimat tartalmazó fájl).
A helyi fájl csak a felhasználó számára használható beállításokat tartalmazza. Amennyiben a *~/.twmrc* fájl létezik, a rendszerszintű beállításokat tartalmazó fájl nem kerül feldolgozásra. Következzék néhány fontosabb twmrc-beállítás:

- a fájl elején található sorok a menü és az ablakfejlécek betűkészleteinek típusát tartalmazzák:
  ```
  TitleFont "-adobe-helvetica-bold-r-normal
  ➥--*-140-*-*-*-*-*-*"
  ```
- a fájl tartalmazza a menüket. Fontos, hogy először a menü könyvtárszerkezetét kell meghatároznunk, és csak ezután térhetünk rá az egyszerű menüpontok megadására:
  ```
  "Xcalc" f.exec "xcalc &"
  ```

A *twmrc* fájl felépítéséről további adatokat a `man twm` parancs kiadásával kaphatunk. A `twm` megbízható, gyors, üzembiztos ablakkezelő, amely az eddigiek során számtalan ablakkezelő alapjául szolgált.

9wm

Ha lehet az egyszerűséget tovább fokozni, a `9wm` még az előzőnél is egyszerűbb ablakkezelő. A grafikus felület indítása után a menüt az egér jobb gombjával érhetjük el, és öt elemet tartalmaz.
New – új X-terminálablakot nyit meg.
Reshape – feladata az ablakok átméretezése (az egér jobb gombjával használhatjuk, az első kattintással jelöljük ki az átméretezendő ablakot, a másodikkal adjuk meg az egyik sarkát, majd egy újabb kattintással az adott ablak átlós sarkát).
Move – az ablakok mozgatására szolgál, szintén az egér jobb gombjával használható. Az első kattintás és az egérgomb nyomva tartása az ablak kijelölésére szolgál, amit az elmozgatása után az egérgombot elengedve elhelyezhetünk.
Delete – bezárja az ablakot.
Hide – az ablak elrejtésére szolgál. Az elrejtett ablakok e jobbgombos menü további pontjaiként fognak megjelenni.
Az ablakkezelő bezárása a CTRL+ALT+BACKSPACE billentyűkkel vagy a `9wm exit` parancs X-terminálon történő kiadásával zajlik. Szintén megbízható és üzembiztos ablakkezelő.

aewm

Egyszerű ablakkezelő. Indítás után csak egy X-terminálablak jelenik meg, amelyről több alkalmazást is el tudunk indítani, például az alábbi paranccsal:

```
rxvt &
```

Az `&` jel visszaadja a parancsértelmezőnek a vezérlést, így több parancs is elindítható egy terminálról.
A fő ablakra kattintva a jobb egérgombbal újabb X-terminálok indíthatók el. A középső gombbal az asztalra kattintva egy ablaklistát hívhatunk elő. A megnyitott ablakokat az ablakok fejlécének jobb szélén lévő négyzetre kattintva a bal egérgombbal zárhatjuk be, és a jobb egérgombbal ugyanoda kattintva rejthetjük el.
A bal egérgombbal a fejlécre kattintva előtérbe hozhatjuk az ablakot, a jobbal pedig a háttérbe küldhetjük, a középső egérgomb folyamatos lenyomásával ugyanott mozgatni is tudjuk.
A CTRL+ALT+BACKSPACE billentyűkkel léphetünk ki az `aewm`-ből. Szintén megbízható ablakkezelő.

pwm

A `pwm` a `twm`-hez hasonlóan képes a rendszermenü használatára, és a beállítási lehetőségei is hasonlóak hozzá.
Indításkor lehetőségként megadható a pillanatnyi beállításokat tartalmazó fájl (`-pwm -cfgfile beÆll t fÆjl`).

larswm

Egyszerű felépítésű ablakkezelő, ennek megfelelően a kezelőparancsai is meglehetősen egyértelműek.
Fontos megjegyeznem, hogy mielőtt ebben az ablakkezelőben bárminek nekikezdenénk, másoljuk át a leírásban található *.larswm* fájlt a saját könyvtárunkba, és alaposan tanulmányozzuk át a **.ps.gz* fájlokat (szintén a leírásban lelhetők fel), amelyek az egérhasználatról és a billentyűkombinációkról nyújtanak tájékoztatást.
Amennyiben ezt a „műveletsorozatot” kihagyjuk, igen kevés

lehetőséget és játékteret hagyunk magunknak: mindössze a virtuális képernyőket váltogathatjuk az asztal alján található asztalváltó sávval.
A megnyitott ablakokat az előzőekben már szóba került PostScript-fájlok tartalmazta leírásokban található billentyűkombinációkkal is kezelni tudjuk. A `laswm` e téren számos lehetőséget nyújt.
A kilépés a CTRL+ALT+BACKSPACE billentyűkkel történik.

A Debian-Sidben található alapablakkezelők

twm	phluid	vtvm
9wm	pwm	qwm
amiwm	ratpoison	uwm
aewm	gwml	wm2
ion	ctwm	flwm
larswm	qvwm	failsafe

ctwm
Szintén a `twm`-hez hasonló tudású ablakkezelő. Leírását úgyszintén még indítás előtt célszerű elolvasni. Ezenkívül ajánlom a leírás mellett található példafájlok ***.twmrc*** valamelyikének saját könyvtárunkba történő másolását ***.twmrc*** néven, nélküle ugyanis nagyon kevés eszköz fog a rendelkezésünkre állni. Én a ***lynx.twmrc*** fájlt próbáltam ki, amivel már kellemes eszközökre tettem szert. Az így kapott beállításokkal elérhetjük, hogy a bal egérgombbal kattintva alkalmazásmenü bukkan fel, a középső gomb használata esetén az ablakkezelő vezérlőmenü (kilépés stb.) jelenik meg, a jobb egérgombbal kattintva pedig az ablakok kezelését segítő menü (ablakbezárás, képernyőanimáció) áll a rendelkezésünkre. Szomorú tapasztalat, hogy ez volt az egyetlen ablakkezelő, amely esetenként nem érzékelte az egér használatát.

vtvm
Első ránézésre szinte teljesen azonos a `twm`-mel: mind a menük, mind a beállítások felépítése megegyezik. A beállításokat a ***.vtvmrc*** fájl tartalmazza a sajátkönyvtárunkban. A rendszerszintű beállítások a ***/etc/X11/rtvm*** könyvtárban szerepelnek.

gwm
Alapvető képernyőkezelő program. A Debianban található változat nem tartalmaz beállítóeszközöket.

uwm
Saját menüvel is rendelkezik, amelyet mi magunk is szerkeszteni tudunk. A teljes rendszerre érvényes beállításokat a ***/etc/X11/ude*** tartalmazza (furcsa, hogy az `uwm` nevű ablakkezelő beállítófájljai az ***ude*** könyvtárba kerültek).

wm2
Alapszintű ablakkezelő, hiszen csak X-terminálok indítására alkalmas, illetve ikonná változtatásukra és újbóli megjelenítésükre képes.

flwm
A Debian saját menüjét veszi át. Beállítási lehetőségei igen szegényesek. Egyetlen saját lehetőséget tartalmazott: több virtuális képernyőt tudunk vele kezelni.

failsafe
Nem is tudom igazán eldönteni, hogy a failsafe külön ablakkezelőnek minősíthető-e, hiszen az X feltelepítésekor önműködően feltevődik a rendszerre. Önálló beállításokkal nem rendelkezik. Indítás után egy fejléc nélküli X-terminál nyílik meg, amelyből különböző alkalmazásokat indíthattunk el. Én annak idején a StarOffice 5.2 indításakor alkalmaztam, hiszen ha teljes képernyőn használtam, akkor az apró eltéréseket leszámítva Windows-szerű felületet adott.

Egy lépcsővel feljebb
Ebbe a csoportba azokat az ablakkezelőket soroltam, amelyek némiképpen egyediek, tehát valamilyen önálló ötletet tartalmaznak – és ennek köszönhetően az előbb fölsoroltaknál bizonyos szempontból jobbak. Természetesen ezek sem közelítik meg a komolyabb ablakkezelők szintjét, de a megfelelő helyen alkalmazva őket igen hasznosak lehetnek a felhasználók számára.

amiwm
A leírás alapján ez a képernyőkezelő felépítésében az Amiga ablakkezelőjére hasonlít. Ezt sajnos nem tudom eldönteni, mivel az Amiga felületét nem ismerem, de mindenképp érdekes felületet hoz létre, leginkább a MacOs-éra emlékeztetett (természetesen jóval egyszerűbb kivitelben). Az egér jobb gombjával a képernyő bal felső sarkában kattintva több menüt is kapunk, ezek közül a legelsőben található egy parancssor begépelését lehetővé tevő ablak. Ebből tudjuk elindítani az alkalmazásokat (ugyanitt lelhető fel az ablakkezelőből kiléptető ***Exit...*** menüpont is). Az így létrejövő ablakok teljes képernyőssé vagy ikonméretűvé tehetők, illetve az ablakok jobb felső sarkában lévő három ikonnal a háttérbe küldhetők. Az ablakok a bal felső sarokban szereplő ikonnal zárhatók be.

ion
Első látásra ez az ablakkezelő is nagyon egyszerűnek látszik. Amennyiben elolvassuk a súgóit, a leírását, illetve betekintünk a bennük hivatkozott fájlokba (***/etc/ion/***), rájöhetünk, a látszat igenis csal: az `ion` ugyanis számos billentyűkombinációt tartalmaz. Az általa megjelenített képernyő igen sajátos, mert minden ablakot teljes képernyős ablakként jelenítünk meg, még a legutolsó beviteli ablakot is. Néhány fontosabb billentyű a könnyed használathoz:

F1	a súgó-oldal kiválasztása és megtekintése;
Mod*+F1	a súgó-oldal kiválasztása és megtekintése;
F2	az X-terminálemulátor indítása;
F3	parancsfuttatás;
Mod*+F3	a lehetőség indítása;
F4	SSH-kapcsolat létrehozása valamilyen kiszolgálóval;
F5	fájl szerkesztése;
F6	a fájlnézegető indítása;
F9	munkaterület készítése, amennyiben már létezik, átlépés rá;
Mod*+F9	a munkaterület lekérdezése;
F11	megerősítés után újra azt kérdezi tőlünk, hogy újraindítsa-e az `ion`-t;
F12	megerősítés után újra azt kérdezi tőlünk, hogy kilépjünk-e az `ion`-ból.
*	(Módosító billentyű, ALT vagy CTRL változatfüggő.)

Nagyon zavaró, hogy a normál billentyűk is tartalmaznak ablakkezelő szolgáltatásokat, ami sok esetben ütközik az indított alkalmazással. Ez volt az egyik legegyedibb ablakkezelő.
Folytatjuk.

Tóth Béla (tothb1@freemail.hu)
Nős, két gyermek büszke atyja. Dolgozott földmérőként, majd térinformatikus szakmérnöki képesítést szerzett. Egyaránt otthonosan mozog a CAD és a térinformatikai programokban, valamint a DOS- és Windows-alkalmazásokban. Legkedveltebb elfoglaltsága már két és fél éve a Linux.

Váltás PostgeSQL adatbázisrendszerre

Néhány hasznos tanács arra vonatkozóan, hogy miként cserélhetjük le meglévő Microsoft Access adatbázisrendszereinket.

Egyre több vállalat kezd nyílt forrású üzleti rendszerekkel foglalkozni, sokan közülük teljes rendszert igyekeznek kiépíteni, amely a vásárlók számára hozzáférhető webes felülettől az alapként szolgáló adatbázisig terjed. Legtöbb esetben a Linux-PHP-Apache hármast, illetve egy nyílt forrású adatbázist (MySQL-t vagy PostgreSQL-t) foglal magában. A PostgreSQL népszerűsége főleg az utóbbi időben növekedett meg, hiszen mára a program elérte azt a minőséget, amelynél már megbízhatónak és jól használhatónak mondható. Több nagyvállalat biztosít állandóan elérhető támogatást, közöttük a RedHat is. Írásomban azt szeretném megmutatni, hogy mire számíthatunk, ha a Microsoft Accessről nyílt forrású adatbázisrendszerre (itt és most a BSD-típusú felhasználási szerződéssel rendelkező PostgreSQL-re) térünk át.

Michael Calabrese, a Bike Friday nevű kerékpárgyártó cég adatrendszereinek felelőse nemrégiben a változás mellett döntött. A Bike Friday egy villámgyorsan növekedő, túra- és hegyi kerékpárok gyártásával foglalkozó cég, székhelyük az Oregon állambeli Eugene-ben található. A vállalatnál PostgreSQL adatbázisban rögzítik az eladásokkal, a gyártással és a vásárlói támogatással kapcsolatos összes adatot. Calabrese mostanában épp a cég e-üzleti rendszerét szándékozik ingyenes programokra lecserélni: Linuxra, Apache-ra és PostgreSQL-re. Egyelőre azonban úgy tervezi, hogy megtartja a munkaállomásokon használt Microsoft Access 97-et annak érdekében, hogy az adatbázis lecserélése miatt szükséges szünet a lehető legrövidebb ideig tartson. Calabrese szerint: „Ha a munkaállomásokon használt kezelőprogramokat nem akarjuk megtartani, egyszerű a dolgunk: csupán le kell futtatni az átalakítóprogramokat és elkezdhetjük megírni az új kezelőrendszert. Ha a kezelőrendszer az Access, amelyet továbbra is szeretnénk használni a PostgreSQL adatbázissal, akkor a fejlődés új irányait úgy jelöltük ki, hogy az induláshoz nincs szükség az egész rendszer átalakítására. A kezelőfelület befagyasztása után nyugodtan számoljunk egy évet az átalakításra. Ha a változtatásokat fokozatosan vezetjük be, megmarad a választási lehetőség, hogy egy-egy új lehetőséget Accessben vagy PostgreSQL-ben valósítunk meg”.

A változtatás eszközei

Amennyiben a Microsoft Open DataBase Connectivity (ODBC) meghajtókat betöltjük a PostgreSQL-sablonadatbázisba, máris megtettük az első lépést az Access és a PostgreSQL összeházasításához. Az együttműködéshez az alapvető átalakító eszközök mellett (lásd a ***További érdekességek*** című részt) néha további ODBC kiszolgálóoldali szolgáltatásokra is szüksége van. Ezeket az ***src/interfaces/odbc/odbc.sql*** fájlban találhatjuk meg. A PostgreSQL felületfüggetlen 4-es típusú Java adatbáziskapcsolati felület (JDBC) meghajtót is tartalmaz. Továbbá a C számára készült beágyazott felület (ECPG) is a PostgreSQL részét képezi. A telepítés végeztével Calabrese adatösszesítő eszközöket választott, például a pgAccesst – ez Windows és Unix változatban, illetve az exSQL 3.1-es változatában egyaránt hozzáférhető.

Miután a meglévő adatbázisokról a rendelkezésre álló eszközökkel (vagy a `pg_dumpall` segédprogrammal, vagy pedig a `pg_dump` és a `pg_dumpaccounts` együttes használatával) biztonsági mentést készítettünk és a telepítőt (***Installer***) is lefuttattuk, az adatok átalakításának első lépéseként az Accessben használatos, azonban a PostgreSQL-ben nem megengedett fájlnevek levadászása következett. Az Access meglehetősen szabadelvű, hiszen a fájlnevekben számos olyan karakter használatát lehetővé teszi, amelyeket más adatbázisrendszerek (Oracle, Sybase, PostgreSQL stb.) nem ismernek fel. Így a Bike Friday adatbázisában szereplő nevek jó részét a PostgreSQL által is kezelhető formájúra kellett alakítani, például az ***Order Detail*** táblázatból ***Order_Detail***, a ***Shipped***? mezőnevekből pedig ***Shipped*** vagy ***ShippedYN*** lett.

Az alapvető átalakítóeszközök minden meg nem engedett karaktert eltávolítanak. Ez súlyos gondokat okozhat, hiszen a mit sem sejtő munkaállomási kezelőprogramok (nem értve a helyzetet) minden további nélkül megszakíthatják az adatbázissal való kapcsolatot.
Calabrese azt javasolja, hogy ha a kezelőprogramokat megtartjuk, ne írjuk át az adatbázis neveit, vagy pedig az adatbázist és a kezelőfelületeket is párhuzamosan módosítsuk. Ő maga ezt úgy oldotta meg, hogy az adatbázisban és a kezelőprogramokban saját kezűleg, egyesével írta át a kérdéses karaktereket.

1. lista Meglehetősen pazarló lekérdezés

```
SELECT Orders.SalesRepID
Bikes_ColorsAvailable.Color,
OrderDetails.BuildABikeID, Orders.OrderDate,
BuildABike.ColorID,
Bikes_BasicFrameTypes.FrameName,
Reps.Rep, BuildABike.Frame, Orders.CustID,
Orders.OrderID, OrderDetails.PartID,
Orders.ShipDate,
BuildABike.BikeState
FROM (Reps RIGHT JOIN Orders ON)
Reps.RepID = Orders.SalesRepID) INNER JOIN
(((Bikes_BasicFrameTypes INNER JOIN
(OrderDetails INNER JOIN BuildABike ON
OrderDetails.BuildABikeID =
BuildABike.BuildABikeID) ON
Bikes_BasicFrameTypes.FrameTypeID =
BuildABike.FrameTypeID) INNER JOIN
Bikes_ColorsAvailable ON BuildABike.ColorID =
Bikes_ColorsAvailable.ColorID)
INNER JOIN Contacts_CurrentFrame_BABID ON
BuildABike.BuildABikeID =
Contacts_CurrentFrame_BABID.BuildABikeID)
On Orders.OrderID = OrderDEtails.OrderID
WHERE (((OrderDEtails.PartID)=6502))
ORDER BY Orders.OrderDate DESC;
```

2. lista Ugyanaz a lekérdezés, de jóval gyorsabban

```
SELECT salesrepid, color, od.buildabikeid, o.orderdate,
            bab.colorid, framename, reps.rep, bab.frame,
            o.custid, o.orderid, partid, o.shipdate, bab.bikestate
FROM        reps, bikes_basicframetypes b_bft, orders as o,
            orderdetails as od,
            bikes_colorsavailable as b_co_a,
            buildabike as bab
WHERE  repid = salesrepid
      AND   od.buildabikeid = bab.buildabikeid
      AND   b_bft.frametypeid = bab.frametypeid
      AND   b_co_a.colorid = bab.colorid
      AND   o.orderid = od.orderid
      AND   bab.custid=[Forms]![Customers/Contacts]![ID]
      AND         entrydate = (
                        SELECT MAX(entrydate)
                                    FROM buildabike as b2
                              WHERE bab.frame = b2.frame
                              )     ;
```

Ez rendben is volt, hiszen a távolabbi tervek között a kezelőprogramok lecserélése is szerepelt. Ami a legfontosabb: a munka ezen szakaszában nagyon fontos, hogy folyamatosan ellenőrizzük, vajon minden működik-e. Ha a meg nem engedett karaktereket kiküszöböltük, az adatok máris készen állnak az átalakításra.

Az adatok átalakítása

Ha azt tervezzük, hogy az Accest megtartva kezelőfelületnek, első lépésként csak az adatokat ültetjük át PostgreSQL alá, a pgAdmint rendkívül hasznos eszköznek fogjuk találni. Calabrese az exSQL módosított változatát is felhasználta annak meghatározásához, hogy az Access és a PostgreSQL hogyan kezeli a táblázatok közötti kapcsolatokat. A ➲ http://www.geocities.com/musica_6898/postgresaccess_home.html címen elérhető honlapján a nyilvánosság elé tárt változat egy héjprogramot futtat le, mely számos esetben módosítja a mezőtípus-átalakítás menetét (például az Access pénznemkezelését is). A Bike Friday Access kezelőfelülete a PostgreSQL számszerű decimális mezőit szövegmezőként értelmezte. Annak érdekében, hogy az Access ezeket helyesen kezelje, Calabrese a mezőket ***Float4***-típusúvá változtatta (a PostgreSQL így nevezi a négybájtos lebegőpontos számokat).

A kezelőfelület kipróbálása

Száznál is több táblázatával a Bike Friday kezelőfelülete meglehetősen összetett. A felhasználó szemszögéből nézve a Bike Friday több mint nyolcvan képernyőt használ a megrendelések beírásához, az alkatrésztáblázat megtekintéséhez, a gyártás ütemezéséhez és a raktárkészlet ellenőrzéséhez. Ezért Calabresének biztosnak kellett lennie abban, hogy a rendszert több tucat felhasználó sem fagyaszthatja le. A kipróbálás jó néhány hétig tartott, közben az SQL-lekérdezéseket is szükség szerint módosítani kellett oly módon, hogy az Access-oldalról vagy (ha ez nem volt megoldható) az adatbázis oldalán újraírták őket, egészen addig, amíg elfogadható sebességet kaptak.
Az *1.* és *2.* listában egy jellegzetes és egy gyorsított lekérdezést láthatunk.
A PostgreSQL-lekérdezések hatékonyabbá tétele általában a `Create index`, `vacuum`, `vacuum analyze`, `cluster` és `explain` parancsok használatával történik. Calabrese azonban figyelmeztet, hogy az Access alapértelmezetten a lekérdezéseket nem közvetlenül továbbítja, hanem mindig az általa leghatékonyabbnak ítélt formára alakítja őket. Calabrese a program okoskodását közvetlen lekérdezés használatával kerülte meg, amely közölte az Access-szel, hogy hozzá ne nyúljon a lekérdezéshez, egyszerűen csak továbbítsa a kiszolgáló felé.
A Bike Friday PostgreSQL-adatbázisának egyszerűsítése során Calabrese úgy ért el sebességnövekedést, hogy a lekérdezésekben az adatbázisból kisebb, pontosabban körülhatárolt adatokat vett ki. Százezer rendelési adat egyidejű lekérdezése

helyett a lekérdezést úgy alakította ki, hogy az adatbázisnak csak 2000 adatra kellett figyelnie. „Az Access buta – mondja Calabrese –, az összes rekordot kézbe veszi és minden egyes alkalommal az összeset átnézi. Ez rendkívül pazarló módszer. Jelenleg harminc alkalmazottunk van, és ha történetesen minden számítógéppel egyszerre próbálnák meg elérni az adatbázist, ez igen hamar nagyon komoly sebességcsökkenést okozna."

A PostgreSQL ellenőrzése

A változtatás következő lépése a lekérdezések hibaellenőrzése, ahol mindjárt két út közül választhatunk. Az egyik a PostgreSQL ODBC-meghajtójához tartozó ellenőrzőeszközök használata. A meghajtóval készíttethetünk egy naplófájlt, és amikor az Access SQL-parancsot küld, a PostgreSQL azt azonnal bevezeti a naplóba, mely a C meghajtó gyökérkönyvtárában található. Ezzel elcsíphetjük az Access olyan ügyetlen próbálkozásait, amikor mondjuk százezer sort próbál egyszerre behívni. Ebben az esetben például a lekérdezést ezer kisebbre bonthatjuk szét. Ez a napló igazából egy nyomkövetés, mely segítségével gyorsan kiszúrhatjuk, ha valami hiba lépett fel, mint ahogy itt is történt.

```
conn=86311032, query=' '

CONN ERROR: func=SQLDriverConnect,
desc='Error from CC_Connect',
errnum=105, errmsg='The database does not
exist on thet server or user authentication
failed.'
```

Azt is megtehetjük, hogyha az Access lekérdezést küld, és a rendszer leáll, a kiszolgálóoldal hibakezelési szintjét (***Debug level***) átállítva a kérelmet csak azért is kiolvassuk. A finomhangolás lényege, hogy minden egyes képernyőn végig kell haladnunk és a kérelmek egyszerűsítésével, összevonásával gyorsítanunk kell őket. Az SQL-t jól ismerők tudják, hogy a rendszer milyen összetett, így elmondhatjuk, hogy fáradságos munka elébe nézünk. Ha azonban a fejlesztésnek ezen a pontján elvégezzük a megfelelő ellenőrzéseket, rengeteg későbbi fejfájástól kímélhetjük meg magunkat.
Mielőtt a rendszer működését visszaállítanánk, a kipróbálás következik. Calabrese folyamatosan figyelte a Bike Friday adatbázisrendszerét, miközben az irodákban már használták a rendszert. „Nemcsak azt kell kipróbálnunk, hogy akadnak-e a kezelőfelületnek hibái, hanem azt is, hogy mekkorára kell a kiszolgálót terveznünk" – mondja Calabrese. Írt egy lekérdező héjprogramot is, amely a három fő gondot okozó részegység (processzor, merevlemezek, hálózat) terheltségét kíséri figyelemmel.
Calabrese programja a processzor kihasználtságát aszerint ellenőrizte, hogy a terhelés hány másodpercig maradt 100, 50 és 0 százalékos. A lemez adatátvitelének értékelését úgy végezte, hogy hány olvasási és írási művelet zajlik éppen, illetve mérte az ezek során átvitt kilobájtokat is. A hálózat terhelését a másodpercenkénti csomagszámmal és a másodpercenként átvitt bájtok számával írta le. Calabrese azt is javasolja, hogy a lezárt hálózati szakaszban végezzünk árasztásos pingelést (`ping -F`), így meghatározhatjuk, hogy a kiszolgáló mekkora terhelésnél akad meg. A memóriával egyszerű a helyzet: minél több van belőle, a PostgreSQL annál többet használ föl, és így annál gyorsabb lesz az adatbázis működése.
Természetesen az adatbázis sebességéről a felhasználók véleménye árulkodik leginkább. A lépésenkénti apró várakozási idők a valóságban hatalmas késésekké adódhatnak össze.

Minden vállalatnál, szervezetnél kialakul egy vélemény arról, mi számít lassúnak és mi elfogadhatóan gyorsnak. Így a rendszer főpróbája mindenképpen az lesz, amikor a felhasználók pár órás használat után kimondják a végítéletet: „Csigalassú" vagy „Hm, nem is olyan rossz".
Végül, miután a rendszer megfelelőnek találtatott, mi pedig a kezelőfelület minden hibáját kijavítottuk, máris nekiláthatunk egy nyílt forrású alapokra építkező e-üzleti rendszer kiépítésének.

Chris Volpe
(chris@macnet2.com)
New Hampshire-ben él és technológiai leírásokat készít.

mascTovábbi érdekességek

Bruce Momjian: PostgreSQL: Introduction and Concepts (ISBN: 0-201-70331-9, 44,95 dollár, 544 oldal)
➲ http://www.ca.postgresql.org/docs/awbook.html címen érhető el.
A gép és az alkatrészek, valamint a PostgreSQL összehangolásáról olvassuk el *Momjian* írását „PostgreSQL Performance Tuning" címmel
➲ http://www.linuxjournal.com/lj-issues/issue88/4791.html
F. Scott Barker: Microsoft Access 2000 Power Programming (ISBN: 0-672-31506-8, 49,99 dollár, 1332 oldal, CD-ROM)
PostgreSQL 7.1 kézikönyv
➲ http://www.ca.postgresql.org/users-lounge/docs/7.1/reference
Postgres GYK
➲ http://www.ca.postgresql.org/users-lounge/docs/#7.1
Postgres/Access GYK
➲ http://joelburton.com/resources/pgaccess
PostgreSQL-leírás és Data Migration Tools
➲ http://postgresql.crimelabs.net/users-lounge/docs
Migration Tools: a csomagban a *pgAdmin* (grafikus PostgreSQL-vezérlőfelület), a *phpPgAdmin* (webalapú eszköz a pgAdminhoz hasonló feladatokra) és a PsqlODBC Windows-meghajtó található. Ez utóbbi lehetővé teszi, hogy a PostgreSQL-adatbázist az ODBC-meghajtókon keresztül elérő Windows-alkalmazásokat írjunk.
Még egy érdekesség: az exSQL új, nyilvános változata is elérhető. Az exSQL nagyszerű PostgreSQL-átalakító eszköz. Az új változat, amely az indexeket és az idegen kulcsokat megbízhatóbban kezeli, a
➲ http://www.geocities.com/musica_6898/postgresaccess_home.html címről tölthető le. A PostgreSQL az idegen kulcsokat a táblázatok összekapcsolására és kapcsolataik kódolására használja. A *Michael Calabrese* által írt exSQL-változat módosítja azokat a szabályokat, melyek meghatározzák, hogy az egyes Access-mezőkből milyen PostgreSQL-mezőtípusok készíthetők. További egyszerűsítések és hibajavítások mellett ez a változat tartalmaz egy parancsfájlt, amely az Access egyik hibáját küszöböli ki: alapértelmezés szerint a program szövegessé alakítja a pénzmezőket. A parancsfájl felülbírálja ezt az alapértelmezést, így futás közben ebből hibák adódhatnak.

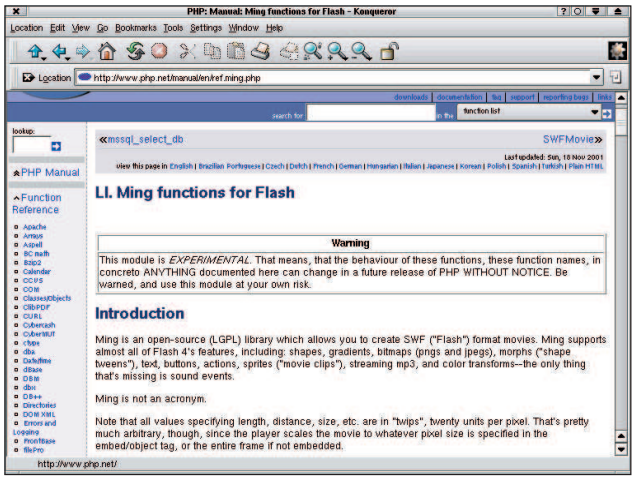
A PHP és a MING

Hogyan készítsünk weblapunkra röptében Flash-mozikat?

Napjainkban a Flash-féle pörgő-forgó csodát nehéz kikerülni a Világhálón. Nem is kell, hiszen a Flash-lejátszás az összes újabb linuxos böngészőben lehetséges. Kedvenc operációs rendszerünkön manapság az ilyen mozik létrehozása sem elérhetetlen cél. A ***MING*** könyvtár segítségével és PHP-támogatással weblapjainkat elláthatjuk röptében előállított animációkkal. Cikkünkben a MING Linuxra telepítéséről, valamint a PHP-vel történő összeházasításáról lesz szó. Emellett a cikk kínálta lehetőségekhez mérten igyekszem a MING működését is bemutatni.

A MING beszerzése és telepítése

A PHP MING kiegészítésének telepítését Debian Woody rendszeren a PHP 4.0.6-os változatához mutatom be. Kis módosításokkal természetesen más GNU/Linux-változatokon is üzembe helyezhető. Tekintettel arra, hogy a támogatás csak a 4.0.5-ös változattól került a PHP-ba, érdemes a gépünkön egy friss PHP-val próbálkozni.



1. kép A MING-függvények részletes leírása a PHP-kézikönyvben

A MING hivatalos weblapját megnyitva hamar szembesülhetünk vele, hogy nem vagyunk magunkra hagyatva. Innen a legfrissebb PHP-változatokhoz azonnal letölthetjük az előre lefordított ***php_ming.so*** modult. Jó esetben akár lusták is lehetünk, és a MING-kiegészítést fordítás nélkül beizzíthatjuk, csupán ezt a fájlt szükséges a PHP-kiterjesztéseket tartalmazó könyvtárba másolnunk. Ennek helyét könnyen megtudhatjuk, ha a parancssorban kiadjuk a `php-config --extension-dir` parancsot. A másolás mellett még a ***php.ini*** fájlba is bele kell nyúlnunk, és a következő bejegyzést kell beillesztenünk:

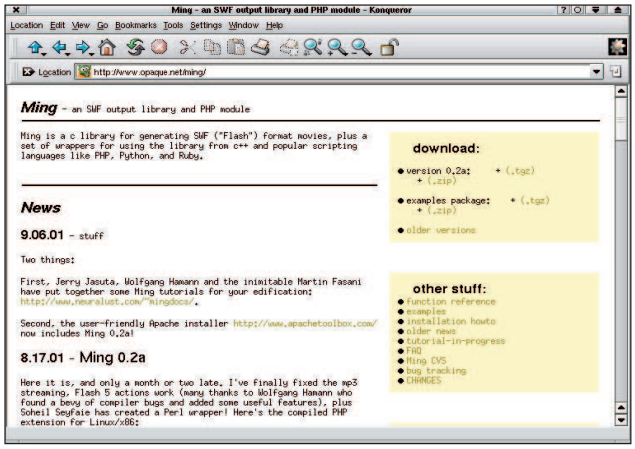
```
extension=php_ming.so
```

Amennyiben ezzel megvagyunk és minden egyezik, máris rendelkezünk a Flash-mozik létrehozásához szükséges eszköztárral. Ne feledkezzünk meg webkiszolgálónk újraindításáról, amennyiben az a beállításváltozások érvényre juttatásához szükséges.
Előfordulhat, hogy a folyamat nem ilyen egyszerűen zajlik le, ekkor sem kell kétségbe esni, a ***php_ming.so*** kiegészítést mi magunk is könnyen létrehozhatjuk. Ehhez először be kell szereznünk a MING forráskódját, majd le kell fordítanunk és telepítenünk (ehhez a parancsokat a MING forráskönyvtárából adjuk ki):

```
make
make install
```

Ezáltal a ***/usr/lib*** alá létrejön a ***limbing.so*** állomány, és egy ***ming.h*** is megfelelő helyére kerül a ***/usr/include*** könyvtárban. Ezáltal megteremtettük a feltételét, hogy a MING-támogatást a jelenlegi PHP-rendszerünkbe építsük. A további szükséges lépéseket már a PHP-forráskönyvtárából kell elvégeznünk:

```
./buildconf
./configure --with-ming <egyØb kapcsol k>
make
make install
```



2. kép A MING oldala ➲ http://www.opaque.net/ming/

A szükséges könyvtárat tehát létrehoztuk, és a helyére is került. A ***php.ini***-nek a fenti bejegyzéssel történő kiegészítése természetesen ekkor is szükséges.

Az ismerkedés

A nagy fejesugrás előtt nem árt néhány dolgot tisztázni: a pontosság és a jó nagyíthatóság érdekében bevezették a ***twip*** mértékegységet. Hogy értsük, mit is takar ez: húsz twip tesz ki egy képpontot. A mozi méretei, azon belül is minden elemnek a mérete, a távolságok mind ebben az egységben értelmezendők. Alapesetben tehát egy 200×200 képpontmérettel rendelkező mozihoz 4000×4000 twip méretű valódi munkafelület tartozik. A másik fontos tudnivaló, hogy a MING jelen pillanatban csak

az FDB-típusú betűkészletek megjelenítésére alkalmas. Ilyenek begyűjtésére a MING-forrás ***util*** alkönyvtárában található `makefdb` használható, először ezt sem árt lefordítanunk. Tekintsük át nagy léptékekben, hogyan is épül fel a MING-birodalom! A legtöbb PHP-kiegészítéssel ellentétben itt nem ömlesztett függvénykönyvtárat kapunk a nyakunkba, hanem 13 osztályt. Ezek mindegyike egy témát ölel fel, és a hozzá tartozó eljárásokat, függvényeket, tulajdonságokat hordozza magában. Lássuk, miből fogunk csipegetni!

`SWFShape()`	Ezzel hozhatjuk létre és rajzolhatjuk meg a különböző, a moziban megjelenő formákat.
`SWFBitmap()`	A moziba bevihető objektumokat JPEG-képekből hozhatjuk létre.
`SWFText()`	A szöveges elemek létrehozására való osztály.
`SWFTextField()`	Szöveges űrlapelemek létrehozásához.
`SWFSprite()`	Önálló animációs almozik hozhatók létre vele, amelyek saját időskálával rendelkeznek.
`SWFButton()`	Nyomógombok létrehozására szolgál.
`SWFFont()`	Különféle betűtípusokat tölthetünk be vele, valamint a szövegek megjelenítéséhez szükséges.
`SWFGradient()`	Színátmenetek létrehozására, továbbá a formák kifestésénél használható.
`SWFFill()`	Már meglévő kifestőobjektumok forgatására, mozgatására és átméretezésére szolgál.
`SWFDisplayItem()`	A moziban létrehozott, behúzott objektumokat itt tudjuk pörgetni, forgatni és nagyítgatni.
`SWFMorph()`	Alakjukat változtató látványelemek létrehozását teszi lehetővé.
`SWFAction()`	A Flash saját nyelvén írhatunk ActionScripteket.
`SWFMovie()`	A mozi maga: elemeket adhatunk hozzá, menthetjük vagy a kimenetre küldhetjük a tartalmát.

Az `SWFMovie()` osztályt mindig használni fogjuk, mert mind a mozi születésekor, mind a véglegesítésekor jelen van. Egy moziobjektumot a következő módon hozunk létre:

```
$mozi = new SWFMovie();
```

Innentől létezik is $mozi néven az objektumunk, ebbe szórjuk bele a mozgatnivaló elemeket. Ha létrehoztuk, nem árt néhány vele kapcsolatos dolgot beállítani:

```
$mozi->SetRate(20);
$mozi->SetDimension(4000,4000);
$mozi->SetBackground(0xff, 0xaa, 0x66);
```

A `SetRate()` által tudjuk megadni, hogy egy másodpercben hány képkockát játszunk le, ez lesz a teljes mozira érvényes beállítás. Megjegyzendő, hogy csak amolyan kívánatos értékről van szó, hiszen egy leterhelt gépen a képkockák megrajzolása a rendelkezésre álló időnél többet vehet igénybe. Ilyenkor egyszerű lassulásról van szó: a lejátszó nem hagy ki kockákat, csupán lassabban játssza le őket. Más eset áll fenn akkor, ha folyamatos MP3 zenei aláfestés is tartozik a mozinkhoz, mert

1. lista A gorillamozi.php – immár teljes pompájában

```
<?php

$mozi = new SWFMovie();

$mozi->SetRate(20.0);
$mozi->SetDimension(4000,4000);
$mozi->SetBackground(0xff, 0xaa, 0x66);

// A kimenetre k ldØs elıtt tudatnunk kell
// a b ngØszıvel az adathalmaz MIME-t pusÆt.

header('Content-type:
       ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

2. lista A gorilla.html – az első mozinkat beágyazó HTML-oldal

```
<html>
<head>
    <title>GorillÆk - rendezıi
      ➥vÆltozat</title>
</head>
<body bgcolor=#ffffff>

    <embed src=gorillamozi.php width=200
      ➥height=200></embed>

</body>
</html>
```

ilyenkor a lejátszónak biztosítania kell, hogy a hanglejátszás lehetőleg folyamatos legyen. Ezt a gondot képkockakihagyással küszöböli ki.

A befoglaló méreteket, azaz megjelenő munkaterületünk méretét a `SetDimension()` által adhatjuk meg. Mint korábban már említettem, itt nem képpontokban, hanem twipekben kell gondolkodnunk, azaz a fenti példa egy 200x200 képpont méretű Flash-objektumot hoz létre.

A méretek közül először a szélességet, másodjára a magasságot kell megadni. A mozinak kell háttérszín is. Ennek beállításához használatos a `SetBackground()`. A háttérszínt az RGB- (vörös, zöld, kék) összetevők keverésével tudjuk létrehozni. A színeket 3x8 biten képezhetjük le, vagyis az egyes értékek értéktartománya 0-tól 255-ig terjed, és csakis egész számokban adhatók meg. A példában éltem a PHP nyelv adta lehetőséggel, és az értékeket tizenhatos számrendszerben ábrázoltam (hiába no, én már csak hexadecimálisokban látom a színeket).

Örvendezzünk, ugyanis elkészítettük életünk első egész estés animációs filmjét! A címe „Gorillák a narancsos ködben" lehetne, tekintve az események letisztult egyszerűségét. Egy apróság hiányzik még: mozinkat láthatóvá is kellene tenni a közönség (legalábbis a böngészőnk) számára. Ehhez az

3. lista A haromszog.php már valami, de még nem mozog…

```
<?php

// A befoglal  mozi.

$mozi = new SWFMovie();
$mozi->SetDimension(6000,6000);
$mozi->SetBackground(0xff, 0xff, 0xff);

// a hÆromsz g megrajzolÆsa-

$valaki = new SWFShape();
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
      ➥0xe4, 0xec, 50));
$valaki->movePenTo(0,1600);
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);

// tegy k a moziba, Øs k ldj k a helyØre

$haromszog = $mozi->add($valaki);
$haromszog->move(3000,3000);
$mozi->nextFrame();

// kØsz van, mehet a vilÆg elØ...

header('Content-type:
      ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

`SWFMovie()` egy újabb fontos eljárását kell alkalmaznunk. Lássunk erre is példát!

```
$mozi->Output();
```

Munkánk eredményét ez a gyakorlatlanok számára így még eléggé emészthetetlen formában tálalja: krixkrax karakterek halmazaként. Mielőtt még e furcsa betűkben látni kezdenénk a jeleneteket, tegyük fogyaszthatóvá az adatokat. Ehhez több dologra is szükség lesz: először is tudatnunk kell a böngészővel, hogy a küldött adatfolyam Flash-mozit közvetít. Második lépésként pedig létre kell hoznunk egy, a művünket beágyazó HTML-oldalt.
Tapasztalat, hogy a Flash-mozik fejlesztése során böngészőnk gyorsítótárát érdemes kikapcsolni, ellenkező esetben hajlamos makacsul ragaszkodni egy korábbi állapothoz. Így pedig meglehetősen nehézkes a kódolás folyamán ellenőrizni, hogy az történik-e, amit valóban szeretnénk.

Lejátszó hiányában…

Előfordulhat, hogy jelenlegi böngészőnk nem alkalmas Flash-fájlok lejátszására, ilyenkor hamar átirányít a Macromedia letöltési oldalára. Amennyiben ez önműködően nem történne meg, magunknak kell ellátogatnunk oda (*3. kép*).

Forgatás előtt – a szereplők

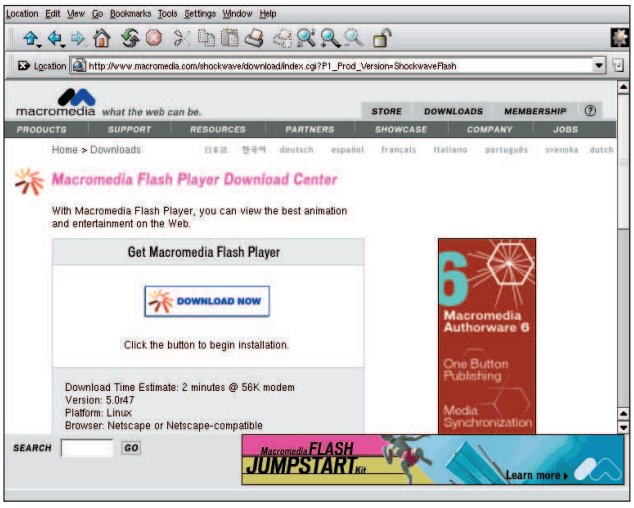
Most már feltehetően az összes szükséges eszközzel rendelkezünk a PHP-s Flash-fejlesztéshez. Beállítottuk a kiszolgálót és a böngészőnket is. Kíséreljünk meg összeállítani valami komolyabbat. Eddig csak az `SWFMovie()` osztály biztosította eszközkészlettel ügyeskedtünk. Itt az ideje megismerni egy másik, szintén elég alapvető osztályt, az `SWFShape()`-et.
A szemléltetést egy egyszerű háromszög alakjának megrajzolásával kezdem. Hozzuk létre a formát képviselő objektumot:

```
$valaki = new SWFShape();
```

Innentől $valaki néven létezik az objektum. Miután megrajzoltuk, az animációnkba számtalan példányban beilleszthetjük (így lehet fenyőfákból erdőt növeszteni). Természetesen minden ilyen egyed külön torzítható, forgatható, mozgatható. A forma megrajzolása előtt szükség lesz pár alapadat megadására, ilyen például a vonalvastagság és -szín. Tudatnunk kell azt is, ki akarjuk-e a rajzunkat festeni, és amennyiben igen, vajon mivel.

```
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
      ➥0xe4, 0xec, 50));
```

Az első sorral a rajzot megjelenítő vonal stílusát határozhatjuk meg. Először a vonal vastagságát, majd pedig a színét kell megszabnunk. A vastagság természetesen twipben értendő, ezért az 5-ös vastagság elég vékonynak számít. A vonal színének megadása a korábban megismerthez hasonló módon zajlik.



3. kép ➲ http://www.macromedia.com/shockwave/download

A vonalrajzolás beállítása után következik a kifestés meghatározása. Mint látható, egymásba ágyazott eljárásokról van szó. A belsővel, azaz az `SWFShape()` osztály `addFill()` függvényével különféle kifestési stílusokat hozhatunk létre. A jelenlegi egy puritán egyszínű festőstílus. Három adatot kell kötelezően megadnunk, amelyeket akár egy negyedikkel is kiegészíthetünk. A példára tekintve az első három talán kézenfekvő is, ezek a szokásos színösszetevők. A negyedik megadható adat pedig egy 0-tól 100-ig terjedő, áttetszőséget meghatározó érték. Ennek a `$valaki->addFill()` kifejezésnek a visszatérő értékét (egy azonosítót) kapja meg a `$valaki->setRightFill()`.
A `setRightFill()` parancsnak létezik egy társa, a

4. lista A mar_valami.php – mozgással ellátott mozi

```
<?php

// vØletlenszÆmgyÆrt  f ggvØny behœzÆsa

include 'random.inc';

// beÆll tÆsok egy helyen

define('NUM_TRIANGLES',20);
define('NUM_FRAMES',100);
define('FRAME_RATE',15);

// kezdı lØpØsek: mozi lØtrehozÆsa

$mozi = new SWFMovie();
$mozi->SetRate(FRAME_RATE);
$mozi->SetDimension(6000,6000);
$mozi->SetBackground(0xff, 0xff, 0xff);

// a mozi sorÆn rengetegszer felhasznÆlt
// hÆromsz g alakjÆnak megrajzolÆsa

$valaki = new SWFShape();
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
➥0xe4, 0xec, 50));
$valaki->movePenTo(0,1600);
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);

// A k vÆnt mennyisØgß hÆromsz g elhelyezØse,
// vØletlenszerß torz tÆsa, szØtsz rÆsa.
// Mindegyikhez k l nb zı,  sszevissza
// lØtrehozott forgÆsi sebessØg adÆsa.

for ($i=0; $i<NUM_TRIANGLES; $i++) {
    $hszog[$i] = $mozi->add($valaki);
    $hszog[$i]->move(2000+randomint(2000),
            ➥2000+randomint(2000));
    $hszog[$i]->scale(randomint(15)/10,
            ➥randomint(30)/10);
    $hszog[$i]->rotate(randomint(360));
    $hszogforg[$i] = randomint(15)-7.5;
}

// a k vÆnt mennyisØgß kØpkocka legyÆrtÆsa
// szØp sorban. Itt mÆr csak forgatni kell. :)

for ($j=0; $j<NUM_FRAMES; $j++) {
    for ($i=0; $i<NUM_TRIANGLES; $i++) {
      $hszog[$i]->rotate($hszogforg[$i]);
    }
    $mozi->nextFrame();
}

// kØsz van, mehet a vilÆg elØ...

header('Content-type:
            ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

`setLeftFill()`. Ha formánk pontjait sorrendben úgy adjuk meg, hogy a körbejárási sorrendjük az óramutató járásával megegyező irányú, akkor van szükség az elsőre. Fordított irányban haladva a „befelé” balra esik. Érdemes erre figyelmet fordítani, mert a lejátszó esetleg bedobhatja miatta a törülközőt. Érdekes adat, hogy ezeket jelenleg valamiért az `SWFMorph()` osztályon belül felcserélve kell használnunk. Apró következetlenség, majd „kinövi” a program. No igen, a PHP/MING-leírás minden egyes oldalon kihangsúlyozza, hogy ez a modul igencsak kísérleti állapotban található, az egyes elemek bármikor gyökeresen megváltozhatnak. Így jelenleg senki sem garantálja, hogy a mostani MING-objektumaink a következő kiadással is ugyanúgy fognak működni, tehát bánjunk velük óvatosan. Innentől kezdődik a forma megrajzolása, amihez vonalakat és íveket kell sorra megadnunk. Emellett rajzeszközünket vonal rajzolása nélkül is arrébb tudjuk pakolni, amire mindjárt az elején szükségünk is lesz, hiszen a tárgyunkat nem a 0,0 pontból kezdjük rajzolni. Irány a kiindulópont!

```
$valaki->movePenTo(0,1600);
```

Ezzel az eljárással „ceruzánkat” vonal rajzolása nélkül mozgatni tudjuk. A koordináták a szokásos módon X,Y sorrendben adandók meg. Vízszintes irányban egyértelmű a helyzet, hiszen ritka eset, ha valahol nem balról jobbra nő a koordinátaérték; függőleges irányban pedig számításba kell vennünk a Besenyő Pista bácsi-féle biorobotelmélet(et): „Egyet kell kérdezni: hogy mekkora, és hogy leerű-fee vagy feerű le?” Nos, kedves Boborján, a második. Tehát Y irányban a koordinátaérték lefelé növekszik, ami pont a körbejárási irány meghatározásánál a legfontosabb (csak megemlítem, hogy a méreteket itt is twipsben kell érteni). A rendszer legalább ebben végig következetes. Lehetőségünk nyílik közvetlen vagy viszonylagos helyzetmegadásra is. Példánkban egész idő alatt a közvetlen módszert választottam, amire az eljárások nevének végén található „To” szócska utal. Viszonylagos elmozdulást egy `$valaki->movePen()`-nel lehetett volna megadni. Innentől kezdve a háromszöget az óramutató járásával megegyező irányban rajzolom körbe:

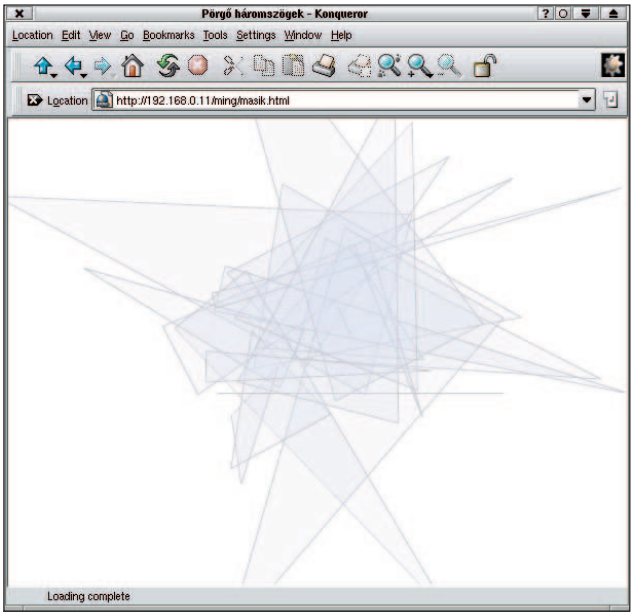
```
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);
```

Tárgyunk megrajzolásával ezennel végeztünk is. Háromszögünket helyezzük a munkatérbe, hogy láthatóvá váljon a kotyvasztásunk végeredménye. A befoglaló HTML-oldal elkészítését a nyájas olvasó/alkotó képzelőerejére bízom. A programlistában három ismeretlen sor található.

5. lista A random.inc a véletlenszám-előállításhoz

```
<?php
function randomint($max) {
  static $startseed = 0 ;
  if (!$startseed) {
    $startseed =
➥(double)microtime()*getrandmax() ;
    srand($startseed) ;
  }
  return (rand()%$max)+1 ;
}

?>>
```



4. kép Ez lesz a vége…

Az első helyezi tárgyunkat a mozi vásznára. Erre ezután a `$haromszog`-objektumon keresztül hivatkozhatunk. Érdemes tisztázni, hogy a `$valaki` csak a formát képviseli. Ebből számtalan egyedet helyezhetünk ki a mozivászonra, amelyeknek mind saját objektumuk lesz, hogy külön-külön lehessen őket módosítgatni. Egy ilyen egyed már az `SWFDisplayItem()` osztályra tartozik. A következő, `$haromszog->move(3000,3000)` paranccsal kerül a háromszög a munkaterület közepére. A `move` az első eljárás, amellyel az `SWFDisplayItem()` parancsai közül megismerkedhetünk. A példa kedvéért szokásomtól eltérve viszonylagos elmozdulást adtam meg. Ennek a `move()` utasításnak tehát létezik egy `moveTo()` testvére is. A formát természetesen eleve olyan módon is rajzolhattuk volna, hogy a közepe a 3000,3000 pontban legyen, de a saját 0,0 pontjának kitüntetett szerepe van: ekörül forog ugyanis, ha egy kicsit „megtekergetjük”.
A `$mozi->nextFrame()` kiadása akkor szükséges, amikor a képkockát befejeztük, azaz minden a helyén van. Mivel egyetlen kockánk készült el, véglegesíthetjük. Igaz, nem lesz több képkocka, „de úgy szép, ha kerek”, és így biztos nem adódik vele gond.

Csapó!

Az animációt kockáról kockára lépegetve kell megterveznünk. Amennyiben valami a mozivászonra került, azt ugyanabban az állapotában a következő kocka is tartalmazza. Csupán új elem szerepeltetésekor kerül sor újra az `SWFMovie()->add()` eljárás alkalmazására. Amennyiben valamit ki kell venni, az `SWFMovie()->remove(a tÆrgy azonos t ja)` parancsot kell kiadnunk.
Írásomat egy teljes animáció bemutatásával zárom, amely a *3. listában* látható példa továbbfejlesztésével készült. Forgatást is alkalmazok benne, amit az `SWFDisplayItem()->rotate()` parancsával tudok megtenni. A forgatási szög fokban értendő, és erre a feladatra a `rotateTo()` is használható. A pozitív forgatási irány az óramutató járásának megfelelő. A programban sok a véletlenszerű elem. Egyrészt a kirakott húsz háromszög sem mind a 3000,3000 ponton csücsül, hanem véletlenszerűen vannak szétszórva. A háromszögek objektumai a `$hszog` tömbbe kerülnek. Egy másik tömbbe helyezem a háromszögek forgási sebességét, amelynek neve `$hszogforg`. Szintén véletlenszerű adatokról van szó, értékük pozitív és negatív egyaránt lehet. Emellett a tárgyak kiinduló elfordulását is megadom, hogy induláskor se merevedjenek vigyázzállásba. Hab a tortán, hogy mindet torzítottam is, amire az `SWFDisplayItem()->scale()` eljárása ad lehetőséget. X és Y irányban a nagyítási arányt külön kell megadni.
A `scale()` használatával a jelenlegi mérethez képest történő nagyítás arányáról van szó. Amennyiben az eredetihez akarjuk viszonyítani, a `scaleTo()` is rendelkezésre áll.
A következő ciklus hivatott a százképkockás mozit legyártani. Minden kockán belül a háromszögeket egyesével meg kell forgatni a hozzá tartozó forgási sebesség szerint. Ha mind a húszat megmozgattuk, jöhet a következő kocka. Amint az összes kocka elkészült, jöhet a film bemutatása. A programot úgy írtam meg, hogy könnyedén lehessen játszadozni a lejátszási sebességgel, valamint a háromszögek számával és a mozi hosszával. A MING még számos haszonnal bír, ezeknek sajnos most nem jutott hely, de a PHP-kézikönyv MING-re vonatkozó részének értő olvasgatása már nem fog gondot okozni.
Kellemes ünnepeket, és jó MING-elést!

Heilig (Cece) Szabolcs (cece@mail.uti.hu) Veszprémben él, huszonhat éves fejjel már hatszoros nagybácsi. Több cégnek dolgozik PHP-programozóként, de PHP-távoktatást is végez. Linuxot először 1994-ben látott, kezdő perles szárnypróbálgatásai után 1997-ben szeretett bele a PHP-be. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalatoskodni.

Kapcsolódó címek

A PHP-kézikönyv MING-része ➲ http://www.php.net/ming

A MING hivatalos oldala ➲ http://www.opaque.net/ming

A Flash formátumának birtoklója, a Macromedia oldala ➲ http://www.macromedia.com

Minden, ami a Flash formátumáról tudható ➲ http://openswf.org

Modellezés DODS segítségével

Az Enhydra-kiszolgáló részeként a DODS az objektumok és a relációs adatbázisok között próbál kapcsolatot teremteni.

Mivel az adatok tárolását, lekérdezését egyszerűvé, rugalmassá és biztonságossá teszik, a legtöbb komoly webalkalmazás gerincét a relációs adatbázisok alkotják. Ez a felállás többnyire egészen addig tökéletesen működik, amíg a fejlesztők olyan objektumokkal nem kezdenek el dolgozni, amelyek teljesen más szemléletmódot követelnek. Vajon lehetőség van-e arra, hogy áthidaljuk az objektumközpontú és relációs világok közti szakadékot?

Valójában számos olyan módszer létezik, amellyel a relációs modellt objektumokká és eljárásokká formálhatjuk, és a legtöbb programozó már rég tervezett magának egy ilyen rendszert. Ahogyan a múlt hónapban láthattuk, a Perl-programozók egy kis segítséget kaphatnak az Alzabo modultól – lehetőséget ad nekik, hogy megtervezzék a táblákat, elérésükhöz pedig eljárásalapú csatolófelületet nyújt.

Ebben a hónapban a DODS-ra (Data Object Design Studio, azaz adatobjektum tervezőstúdió) vetünk egy pillantást, amely szellemében az Alzabóra hasonlít, ezt azonban Java-felhasználóknak szánták. A DODS az Enhydra központi eleme, amelynek hamarosan megjelenő változata (Enhydra Enterprise) várhatóan az első olyan nyílt forrású alkalmazáskiszolgáló lesz, amely támogatja a J2EE-t (Java 2, Enterprise Edition).

Jelenleg az Enhydra Enterprise még kipróbálás alatt áll, és bár úgy tűnik, a DODS-támogatás sokat fejlődött az utolsó változat óta, ***David Young*** a Lutris Enhydra-prófétája szerint az Enhydra 3.x DODS-változata azért üzembiztosabb. Hogy a dolgokat könnyen kipróbálhassam, a Lutris küldött nekem egy EAS (Enhydra Applicattion Service) példányt, amely az Enhydra bővített, kereskedelmi változata.

Nem vagyok benne teljesen biztos, mi a különbség az EAS és a nyílt forrású Enhydra-kiszolgáló között. Az ***Enhydra.org*** azt írja, hogy az EAS az Enhydrán alapul, de az EAS és egy Enhydra-példány vásárlása közötti különbség nem teljesen nyilvánvaló. Azt fogom feltételezni, hogy az általam telepített EAS nagyjából azonos a 3.1-változattal, bár meglehet, hogy ez nem teljesen pontos feltételezés.

A DODS áttekintése

A DODS-nak, akárcsak a múlt hónapban megismert Alzabo-rendszernek, kettős célja van: magas szintű felületet nyújt adatbázisok tervezéséhez, illetve eljárás- és objektumkészletet nyújt, amellyel azután az adatbázis elérhető. Míg az Alzabo kiszolgálóoldali, a DODS ügyféloldali Javában íródott alkalmazás, amelylyel adatbázisainkat építhetjük fel, illetve szerkeszthetjük.

A DODS elsődleges célja, hogy párhuzamosan készítsen SQL-meghatározásokat és Java-osztályokat, amelyek ugyanazt az adatbázist írják le. Ezután adatbázisba tölthetjük az SQL-meghatározásokat, a Java-osztályokkal pedig elérhetjük őket. Ezenkívül a DODS többfajta adatbázissal való munkára is fel lett készítve. Jelenleg PostgreSQL-, MySQL-, Sybase- és Oracle-rendszerekkel működik, de ez a kör a jövőben valószínűleg tovább bővül. Mivel a tényleges SQL-lekérdezések egy objektum-középrétegben íródtak, az Enhydra-programok átírás nélkül vihetők át egyik adatbázis-kezelőről a másikra. A valóságban természetesen kicsit bonyolultabb a dolog. Például az Enhydra PostgreSQL-támogatása nem éppen lenyűgöző, ugyanis figyelmen kívül hagyja a `SERIAL` adattípust (ez valójában egy számláló, más néven szekvencia), és nem kezeli a hivatkozásiépség-megkötéseket (referential integrity constrations), így például az idegen kulcsokat sem. Mindenesetre a cél becsülendő, és örömmel látnám, ha az Enhydra 4.x már kezelné ezt a gondot. Idővel a DODS várhatóan egyre több különféle adatbázist lesz képes kezelni, illetve a megfelelő lekérdezést az egyes SQL-nyelvjárásokhoz elkészíteni.

Az XMLC-dokumentum elkészítése

Hogy a működésükkel megismerkedhessünk, készítsünk az Enhydrával és a DODS-sal egyszerű adatbázis-, illetve webalkalmazás-együttest. Példáimban a PostgreSQL-t fogom használni, két okból kifolyólag is. Egyrészt, mert kitűnő nyílt forrású adatbázis-kezelő, másrészt mert DODS támogatja. Példánk, akárcsak a múlt hónapban, egy egyszerű webnapló lesz (más néven blog, egy olyan napló, amely az adatbázis bejegyzéseit fordított időrendben mutatja be). Egy ilyen program megírása nem különösképpen bonyolult, viszont annál inkább vonzó a DODS és az Enhydra kipróbálásaként.

Az első megállónk az Enhydra Appwizard lesz, amely elkészíti az alkalmazásunkhoz szükséges vázlatokat és könyvtárakat. Az Appwizard a ***$ENHYDRA/bin*** könyvtárban található, ahol az `ENHYDRA` az Enhydra telepítéskönyvtárának megfelelő környezeti változó. (Amikor RPM-csomagokból a saját RedHat-gépemhez telepítettem CD-ről, az `ENHYDRA` értéke ***/usr/local/lutrys-enhydra3.5.2*** volt.)

Az első `appwizard` képernyőn a hagyományos webalkalmazás és az Enhydra szuperservlet között választhattam, az utóbbi mellett döntöttem. A következő képen a HTML projektre voksoltam (a vezeték nélküli WML projekt helyett), majd a projektet elneveztem „blog”-nak és behelyeztem az ***il.co.lerner*** csomagosztályba. Elfogadtam az Enhydra-alkalmazásokhoz rendelt, alapértelmezett ***~/enhydraApps/*** alkalmazás saját könyvtárát. A forráskódomhoz nem szándékoztam szerzői jogi üzenetet rendelni, így végezetül a ***Finish***-re kattintottam, amely a ***~/enhydraApps/*** könyvtárban 18 új állományt hozott létre.

Most, hogy elkészítettük az alkalmazás vázát, módosíthatjuk az Enhydrával érkező alapértelmezett üdvözlő (Welcome) oldalt. Ezt két lépésben kell megtennünk: először a ***Welcome.html*** HTML-fájlt kell megváltoztatnunk, amely az én gépemen a ***~/enhydraApps/blog/src/il/co/lerner/presentation/Welcome.html*** helyen található.

Érdemes odafigyelni rá, mivel ez a fájl nem csak alap-HTML, hanem az XMLC által feldolgozandó további tagokat is tartalmaz (lásd ***Linuxvilág*** 2001. október, 67. oldal). Amint az *1. listában* látható, úgy fogjuk megváltoztatni, hogy az eredeti egyszerű oldal helyett blogunk legfrissebb adatait jelenítse

meg. Az XMLC és a hagyományos HTML-oldal közötti egyetlen különbség az, hogy a módosítani kívánt részeket `id` tulajdonsággal felruházott `<span>` tagok közé helyezzük. Például

```
<p><b><span id="date">Date</span></b></p>
<p><span id="text">Text</span></p>
```

amennyiben ezt a fájlt most egyszerűen csak megjelenítjük a böngészőben, a ***Date*** és ***Text*** szavakat fogjuk látni. A felhasználók azonban nem fogják ezt az oldalt közvetlenül elérni. Az XMLC ugyanis Java-osztállyá fogja őket fordítani. Ezután a ***WelcomePresentation*** osztályt használhatjuk a dokumentum egy példányának előállítására, miközben a ***text*** és a ***date*** mezők értékét önműködően létrehozott eljárásokkal állíthatjuk be.

A DODS használata

A ***WelcomePresentation*** a ***date*** és ***text*** mezőkbe szánt adatot a relációs adatbázistáblából kérdezi le. Mielőtt folytatnánk, készítenünk kell tehát egy táblát, és be kell népesítenünk néhány adattal.
Ez az a pont, ahol a DODS belép a képbe. A ***$ENHYDRA/bin/dods*** helyen található `dods` program szintén ügyféloldali, grafikus, Javában írt alkalmazás. Amennyiben a DODS-sal dolgozunk, soha ne feledjük, hogy olyan alkalmazást használunk, amely két különböző szemléletmódot köt össze, így a kifejezésmódja néha bizony furcsának tűnhet.
A DODS egy csomag készítésével indul, amely majd az általunk készített összes tábla és tulajdonság tárolására szolgál. Amint azt a kezdeti DODS-képernyőn is láthatjuk, e csomag neve alapértelmezés szerint ***root***. Ezt én ***blog***-ra változtattam, rákattintottam a ***root*** mappára, majd az ***Edit*** menüből a ***Package*** pontot választottam.
Az adattáblát (***BlogEntries***) két tulajdonsággal hozzuk létre: ***date*** (***dátum***), és ***text***, amelyek egyébként megegyeznek a ***Welcome.html*** változatunkban használt ***id*** tagokkal. Első lépésként az ***Insert*** menüponttal egy új táblát adunk hozzá a BlogEntrieshez: kiválasztjuk a ***data object*** pontot és a ***BlogEntries*** nevet adjuk neki.
Ezután a táblánkhoz két mezőt (***date*** és ***text***) kell adnunk. Ehhez a DODS-ablak bal felső sarkában kattintsunk a ***BlogEntries*** szóra és az ***Insert*** menüpont használatával szúrjuk be a két új tulajdonságot. Mindkét tulajdonságunk ***varchar*** típusú lesz – jelölvén, hogy szöveget szeretnénk bennük tárolni. Bár a mi esetünkben ez tökéletesen megfelel, kár, hogy DODS a PostgreSQL `TIMESTAMP` adattípusát nem kezeli, ez ugyanis ügyes és kifinomult idő- és dátumadat-kezelést tesz lehetővé. Így a dátumokat szöveges formában fogjuk tárolni, tudván, hogy az `ORDER BY` segítségével sorrendben kaphatjuk őket vissza – és nekünk ennyi elég is.
Mivel web-, illetve adatbázis-alkalmazásunkkal a lehető legnagyobb sebességet szeretnénk elérni, és mivel lényegesen kevesebbszer fogunk szöveget beszúrni, mint lekérdezni, közöljük a DODS-sal, hogy mindkét mezőnket tegye indexelhetővé és lekérdezhetővé. Az első az SQL-meghatározást fogja módosítani azáltal, hogy indexet készít a mezőkhöz. A második egy további eljárást készít, amivel az oszlopban tárolt adatokhoz férhetünk hozzá.
Végül az adatbázismenüből kiválasztjuk a PostgreSQL-t. Ezáltal a DODS kifejezetten PostgreSQL-stílusú SQL-t készít.
Most, hogy táblánkat elkészítettük a DODS-sal, nincs más hátra, minthogy (a ***File, Save as*** menüpont segítségével XML-formátumú DOML formában) mentsük, amely leírja a táblánkat, és mind a Java, mind az SQL elkészítéséhez felhasználható. Mentsük a DOML-fájlt a projektcsomag forráskönyvtárába; az én esetemben ez a ***blog/src/il/co/lerner*** könyvtárat jelenti. Ha DOML-fájlunk elkészült (lásd a *2. listát*), SQL- és Java-parancsokká alakíthatjuk át a ***File*** menü ***Build all*** parancsával. E választás eredményeképpen számos fájl keletkezik az adatkönyvtárban, ezért amikor válaszolnunk kell, hogy a fájlokat hova szeretnénk telepíteni, válasszuk azt az adatkönyvtárat, ahová a ***blog.doml*** fájlt helyeztük. Egy ablak fog megjelenni, amely tájékoztat bennünket, hogy a DODS mit is csinál éppen. Ha minden simán ment, a DODS-ból akár ki is léphetünk.

Az adatbázis elkészítése

A DODS `build all` parancsának futtatása a DOML-fájlt jó néhány új fájlra bontotta szét az adatkönyvtárban. A könyvtár immár nemcsak egy (korábban üres) Makefile-t tartalmaz, hanem egy `blog` alkönyvtárat is, amely a következő négy Javaosztályt tartalmazza: `BlogEntriesBO`, amely az Enhydra megjelenítést végző bemutató objektumához hasonlít; `BlogEntriesDataStruct`, amely tulajdonképpen táblánk adatait tárolja; `BlogEntriesDOI` amely a `BlogEntriesDO` objektumhoz tartozó csatolófelület; végül a `BlogEntriesQuery`, amely lehetővé teszi, hogy az előzőleg lekérdezhetőnek megjelölt mezőket lekérdezzük.
Az elkészült Java-forráskódon túl néhány olyan fájlt is találunk itt, amely SQL-utasításokat tartalmaz. Nevezetesen rálelhetünk egy ***create_tables.sql*** fájlra, amelynek segítségével az adatbázisunkat hozhatjuk létre.
Mindehhez a `CREATEDB` parancsot fogjuk használni, amelyet általában valamilyen erre jogosult felhasználó hajt végre Unix-héjprogramból (ami nem feltétlenül a rendszergazdát jelenti; nézzük végig a PostgreSQL leírását, hogy megtudjuk, miképpen készítsünk PostgreSQL-felügyelőket).
Kiadhatjuk a következő parancsot:

```
CREATEDB blog
```

Ezekután az adatbázisnak interaktív módon küldhetünk lekérdezéseket a következő paranccsal:

```
psql blog
```

Ha tábláinkat az önműködően létrehozott DODS parancsfájl segítségével szeretnénk létrehozni, a `psql \i` parancsát kell használnunk:

```
\i
➥/home/reuven/enhydraApps/blog/src/il/co/
➥lerner/data/create_tables.sql
```

Láthatunk néhány `CREATE` üzenetet, majd végül ismét a psql parancssora jelentkezik be. A `\d` parancs használatával kideríthetjük, hogy a DODS nem készítette el a ***BlogEntries*** táblát. Ehelyett két másik táblát hozott létre, az egyiket ***objectid***, a másik (elsődleges) táblát pedig ***newdbtable*** néven. Az ***objectid*** tábla a `PostgreSQL` sequence függvényeit hivatott helyettesíteni, jól szemléltetve az ilyesfajta általános eszközök korlátjait. A tábla egy ***next*** oszloppal rendelkezik, amely megmutatja, milyen azonosító lesz a következő. Ennek megfelelően az adatokat a ***newdbtable*** táblába szúrjuk be, ilyenkor az ***objectid*** táblát mindig egy-egy sorral bővítjük.
Adjunk is mindjárt néhány elemet a táblákhoz, hogy legyen mit lekérdezni:

1. lista A Welcome.html XMLC-bemeneti fájl, amit a Weblog megjelenítéséhez fogunk használni

```
<html>
  <head>
      <title>Weblog</title>
  </head>

  <body bgcolor="#FFFFFF">

      <H1>Weblog</H1>

      <P>Welcome to our Weblog!  Here is the
            ➥latest entry:</P>

      <p><b>
    <span id="date">Date</span>
    </b></p><p><span id="text">Text</span></p>

  </body>
</html>
```

2. lista A blog.doml, a DODS által önműködően készített fájl

```
<?xml version="1.0" encoding="UTF-8"?>
<doml>
  <database database="PostgreSQL"
   ➥legal_values="Standard,InstantDB,Oracle,
   ➥Informix,Msql,Sybase, PostgreSQL">
    <package id="blog">
      <table id="blog.BlogEntries"
       ➥dbTableName="NewDBTable">
        <column id="entrydate"
              ➥isIndex="true"
         ➥usedForQuery="true">
          <type dbType="VARCHAR"
              ➥javaType="String"/>
        </column>
        <column id="text" isIndex="true"
         ➥usedForQuery="true">
          <type dbType="VARCHAR"
              ➥javaType="String"/>
        </column>
      </table>
    </package>
  </database>
</doml>
```

```
INSERT INTO newdbtable (entrydate, text,
➥objectid, objectversion)
➥VALUES (NOW(),'First blog entry', 1, 1);
INSERT INTO objectid ( next ) VALUES ('2');

INSERT INTO newdbtable (entrydate, text,
➥objectid, objectversion)
➥VALUES (NOW(),'Second blog entry', 2, 1);
INSERT INTO objectid ( next ) VALUES ('3');
```

Most, hogy már van két blogbejegyzésünk, ki is léphetünk a psql-ből (\q), és elkezdhetjük módosítani a Java-osztályokat. Rakjuk össze mindezt!
A varázslás nagy része a ***WelcomePresentation.java*** fájlban zajlik. Itt fog elkészülni a ***Welcome.html*** és az adatbázisobjektumok egy-egy példánya, majd a lekérdezések eredményének megszerzése után a HTML-fájl itt töltődik fel az adatokkal. Miután a 3. listának (24. CD Magazin/DODS könyvtár) megfelelően módosítottuk a ***WelcomePresentation.java*** fájlt, futtassuk le a `make`-et a projekt sajátkönyvtárából. Az Enhydra lefordítja a Java-osztályainkat, ellenőrzi, hogy minden szükséges a helyén van-e, és futásra kész állapotba hozza az alkalmazásunkat.
Figyeljük meg, hogy a 3. listában módosítanunk kellett a `run` eljárást, hogy két új kivételt adjon vissza: a `NonUniqueQueryException`-t és `DataObjectException`-t. Ezeket az általunk létrehozott különféle adatobjektumok hozzák létre, és mivel ezeket a kivételeket nem szándékozunk elfogni, a hívónak jeleznünk kell, hogy lehet, hogy kivételt vált ki.
A 3. lista a DODS-eljárásokat használó SQL-lekérdezéseket az Enhydra QueryBuilder segítségével hozta létre. Első lépésként elkészítjük az egyik önműködően létrejövő osztály, a `BlogEntriesQuery` egy példányát:

```
BlogEntriesQuery blogq = new
      BlogEntriesQuery();
```

A jelen pillanatig az összes sort le szeretnénk kérdezni, fordított időrendben:

```
blogq.setQueryEntrydate("NOW()",
      ➥QueryBuilder("LESS_THAN");
blogq.addOrderByEntrydate(false);
```

Olyan eljárások is léteznek, amelyekkel a `WHERE` kifejezést is a lekérdezésünkhöz szúrhatjuk, így már meglehetősen összetett lekérdezéseket alkothatunk.
Végül az egyező sorok halmazát kapjuk vissza, amelyek mindegyike egy-egy `BlogEntriesDO` objektum formájában jelenik meg:

```
BlogEntriesDO[] blogEntries =
      ➥blogq.getDOArray();
```

Mivel csak a legfrissebb adatot szeretnénk megjeleníteni, a tömbnek egyszerűen az első elemét vesszük. A szöveget az XMLC által létrehozott „welcome” objektum eljárásának segítségével illesztjük a dokumentumunkba:

```
Welcome.setTextDate(blogEntries[0].getEntrydate());
Welcome.setTextText(blogEntries[0].getText());
```

Ha a módosítással készen vagyunk, futtassuk le a `make`-et a projekt legfelsőbb szintű könyvtárából. Ha Java programunkban bármilyen hibát találunk, ahányszor csak akarjuk, kijavíthatjuk, majd újrafuttathatjuk a `make`-et.
Elméletileg – a kimeneti könyvtárba lépve és a `./start` parancsot futtatva – most már elindíthatjuk a programot. Ha valóban megtesszük, láthatjuk, hogy próbálkozásunk kudarcot vall, mivel az alkalmazás még nem tudja, hol keresse a ***PostgreSQL .JAR*** fájlt. Az is hasznos, ha az első használatkor az Enhydrát (illetve bármilyen más alkalmazást) teljes körű hibakeresési kimenettel indítjuk, hiszen így az esetleges hibákat sokkal gyorsabban fel tudjuk deríteni és ki tudjuk javítani.

Három fájlt kell módosítanunk, hogy működésre bírjuk a dolgokat. Először módosítsuk az *$ENHYDRA/bin/multiserver* fájlt: a PostgreSQL JDBC meghajtó .JAR fájljára helyezzünk el egy hivatkozást. Ezt egyszerűen a multiserver fájl megváltoztatásával érhetjük el (ez tulajdonképpen egy héjprogram, ami a Java programot hívja meg). A `Build up classpath` megjegyzéssor alatti részeket a következőre változtatjuk:

```
# Where is the PostgreSQL JDBC .jar file?
PG_JDBC=/usr/share/pgsql/jdbc7.1-1.2.jar

if   [ X${CLASSPATH} = "X"] ; then
      ➥NEWCP=${ENHYDRA_CLASSES}${PS}${PG_JDBC}
else

NEWCP=${ENHYDRA_CLASSES}${PS}${CLASSPATH}${PS}
      ${PG_JDBC}
fi
```

Következő lépésként be kell állítanunk a *blog.conf* fájlt. Minden Enhydra-projekt tartalmaz egy beállításfájlt, amely számos más érték mellett meghatározza, hogy a rendszer milyen adatbázist használjon. Az én esetemben ez a beállításfájl a *blog/output/conf/blog.conf* névre hallgat, és az alkalmazással kapcsolatos rengeteg név-érték párost tartalmaz.
Módosítanunk kell a `Database manager` rész néhány részletét, hogy a mi programunkra mutasson. A 4. listában (24. CD Magazin/DODS könyvtár) megtaláljuk ezt a részt – abban a formában, ahogy annak lennie kell.
Végül módosítjuk a *servlet.conf*-ot. Annak ellenére, hogy nem kell feltétlenül megváltoztatnunk, hasznos dolognak tartom a következő két sorban a hibakereső részek bekapcsolását:

```
Server.LogToFile[] = EMERGENCY, ALERT,
      ➥CRITICAL, ERROR,WARNING, INFO, DEBUG
Server.LogToStderr[] = EMERGENCY, ALERT,
      ➥CRITICAL, ERROR,WARNING, INFO, DEBUG
```

A legfontosabb, amit a *blog.conf* és *servelet.conf* fájlokról tudni kell, hogy újra létrejönnek, valahányszor csak legfelsőbb szintű `make` parancsot adunk ki. Ezért ha ilyen módszerrel változtattuk meg a fájlokat, többet semmi esetre se adjuk ki a legfelsőbb szintű `make` parancsot. Ha megtesszük, meg fogjuk bánni (ahogy én is megbántam). Ehelyett inkább ezt a parancsot *presentation* könyvtárban adjuk ki.
Ha a beállításfájlokat megváltoztattuk, beléphetünk a *~/enhydraApps/blog/output* könyvtárba és futtathatjuk a `./start` parancsot. Láthatjuk, ahogy a kiszolgáló elindul, illetve (amennyiben a DEBUG lehetőséget beállítottuk a *servlet.conf*-ban, illetve a naplózást a *blog.conf*-ban) jó néhány hibakereső üzenetet tekinthetünk meg.
Ellenőrizhetjük is alkotásunkat. Állítsuk a böngészőt 9000-es kapura, amely az Enhydra-alkalmazások alapértelmezett kapuszáma: *http://localhost:9000/*. Ha minden jól megy, a böngészőben blogprogramunk kimenetét fogjuk látni.

Összegzés

A DODS jobb az Alzabónál, mivel kitűnő objektumréteget nyújt az SQL felett. Továbbá úgy tűnik, jobb és megbízhatóbb eljárásokat kínál az egyes lekérdezések összeállítására és az eredmények kezelésére. Ugyanakkor a DODS néhány szempontból ugyanazokban a betegségekben szenved, mint az Alzabo, vagy bármely egyéb relációs-objektum kapcsolatteremtő rendszer.
Az egyik ilyen gond, hogy új módszert kell megtanulnunk az (évek óta megszokott) SQL-lekérdezések elkészítésére, illetve a hosszabb lekérdezéseket meglehetősen kényelmetlen megírni, hiszen eljáráshívásokból kell összerakni őket. A DODS-szerű rendszerek általános használhatósága egyben azt is jelenti, hogy kedvenc adatbázis-kezelőnk különleges képességeit nem használhatjuk ki. Így például a PostgreSQL esetében a DODS úgy tűnik, teljesen figyelmen kívül hagyja az idegen kulcsokat és a számlálókat, amelyek pedig sokkal tömörebb adatszerkezetet eredményeznének.
Az Enhydra egyéb részeivel társítása a DODS azonban kitűnően működik. Akárcsak az XMLC és a superservletek esetében, a DODS-t is előbb letaglózónak és ügyetlennek éreztem, utóbb viszont hasznosnak és okosnak. Első pillantásra az Enhydra DODS eszköze jó próbálkozás az objektum- és a relációs világ közötti szakadék áthidalására. Már alig várom az Enhydra Enterprise végső változatát, amely kétségtelenül újabb lökést ad majd a dolgoknak.
A következő hónapban belenézünk abba az egyre inkább előtérbe kerülő szabványba, amely nemcsak, hogy összeköti a relációs és az objektumvilágot, de kiszolgálóoldali Java-alkalmazásainkat tranzakció-kezelési képességekkel is felruházza. Az Enterprise JavaBeans szolgáltatásokat és adatokat nyújt a webalkalmazásoknak, és egyre népszerűbbé válik azok közt a webfejlesztők között, akik objektumokat szeretnének alkotni, használni, illetve adatbázisban tárolni, anélkül, hogy meg kellene erőltetniük magukat.

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/).

Kapcsolódó címek

Az Enhydra fő honlapja a ➲ http://www.enhydra.org címen található. Az Enhydra 4.x más néven Enhydra Enterprise az ➲ http://www.enterprise.enhydra.org címen lelhető fel, a ➲ http://www.dods.enhydra.org címen pedig a DODS Projectről találunk néhány adatot.
Az Enhydráról szóló kitűnő bemutatót találunk a ➲ http://www.arsdigita.com/asj/enhydra címen, amelyet *Roger Metcalf*, az ArsDigita Corporation munkatársa írt.
Meglehetősen jó, bár kissé túlhaladott ismertetőt találunk az Enhydra, a DODS és a PostgreSQL használatáról az ➲ http://enhydra.enhydra.org/software/documentation/NewApps-DODS-Tutorial-PGSQL.html címen.
DODS és QueryBuilder témakörben két másik hasznos honlap: ➲ http://dods.enhydra.org/software/documentation/gettingStarted.html és ➲ http://www.dods.enhydra.org/project/faq/UsingTheGeneratedCode.html
Nagyon köszönöm a Lutris munkatársainak, különösképpen *David Young*-nak, hogy DODS-kérdéseimmel kapcsolatban ilyen segítőkészek voltak. Külön óriási köszönet illeti őket, amiért olyan nyugodtan tűrték a Federal Express és az izraeli iroda több mint egyhetes őrületét, akik egyszerűen nem hitték el, hogy egy program valóban ingyenes is lehet.

A mélység felfedezése

Marcel az fsv, a 3dfile és az XCruise segítségével a Linux furcsa világát egy egészen új nézőpontból mutatja be.

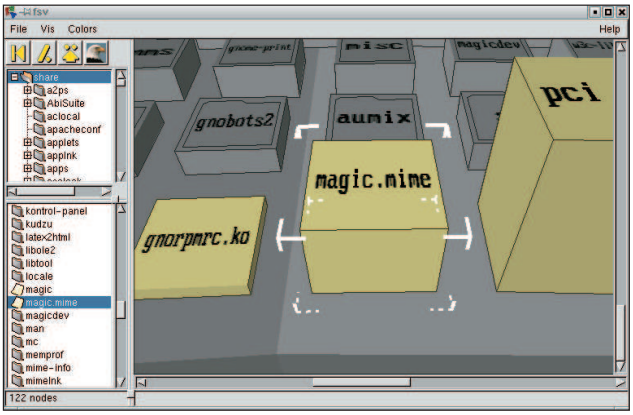
François, mon ami, le vagyok nyűgözve. El kell ismernem, amikor arra kértelek, hogy válassz olyan bort, amely kellemesen jellegzetes, teljesen megbíztam benned – és valóban elégedett vagyok. Az 1997-es Volnay-Santenots du Milineu remek választás. Kérlek, hozz fel annyit, hogy a vendégeinknek jusson bőven.
Már itt is vannak! Bienvenue, mes amis. Üdvözöllek titeket Chez Marcel kitűnő Linux-konyhájáról híres fogadójában. Foglaljatok helyet! François éppen a pincébe ment, hogy bort hozzon. Tudjátok, mes amis, mindig bámulatba ejt, amikor a finom ételek, a bor és a Linuxszal való főzés közötti párhuzamra gondolok. Nézzük például az e havi számot … François, épp jókor érkeztél. Kérlek, nyisd ki a bort és tölts a vendégeknek. Merci.
Miként a jó étteremtulajdonos eggyé válik a borospincéjével, úgy lesz eggyé a jó rendszergazda is a rendszerével. A rendszergazda minden könyvtárat és állományt úgy ismer, mint a tenyerét. Természetesen egy borospincéről sokkal könnyebb képet alkotni. Időről időre bejárom hűvös termeit, s hagyom, hogy a palackok, a címkék és az illatok elborítsák az érzékeimet. Linux-rendszergazdaként eddig csak a képzelőerőmre volt bízva e világ megjelenítése. De mától, mes amis, minden megváltozik! Ma Linux-rendszeretek valóságos hellyé változik át. Állományrendszeretek néhány nyílt forrású recept segítségével ezennel belép a harmadik dimenzióba.
A mai menü első fogása a ***Daniel Richard*** konyhájából származó, magát szerényen csak `fsv`-nek (filesystem viewer) nevező főzet. A programocska indulását követően először az egész állományrendszert végignézi, majd a tartalmát teljesen új és egyedi módon tárja elénk. Az állományok és a könyvtárak egyszerre csak különböző magasságú hasábokként jelennek meg, mintha egy idegen város utcáin járnánk. Ha rákattintasz egy könyvtárra, lehetővé válik, hogy megközelítve feltáruljon az alatta lévő „város”. A program kétféle megjelenítési módot kínál: térkép- illetve fanézetet. A fanézetben az az érdekes, hogy a könyvtárak úgy jelennek meg, mint egy távoli űrbéli város felhőkarcolói. Az *1. képen* az `fsv`-t működés közben láthatjátok. Az fsv a ➲ http://fsv.sourceforge.net címről tölthető le. Az elindításához olyan dolgokra lesz szükséged, mint az OpenGL vagy a Mesa GTK+ (lehetséges, hogy már mindkettő megtalálható a gépeden, vagy olyan közel vannak hozzád, mint Linuxod telepítőkorongjai), illetve a Löf's GtkGLArea OpenGL-készlet a GTK+-hoz, amely a ➲ http://www.student.oulu.fi/~jlot/gtkglarea címen érhető el. Amennyiben RPM-alapú rendszerrel rendelkezel, a ➲ http://ww.rpmfind.net címen előre fordított csomagokat is találhatsz. De térjünk vissza a tárgyhoz. Csomagold ki a forrást és fordítsd le:

```
tar -xzvf fsv-0.9.tar.gz
cd fsv-0.9
make
make install
```

A program futtatásához egyszerűen gépeld be az `fsv` parancsot. Az eredmény megjelenése az indulókönyvtár méretétől függően eltarthat néhány másodpercig. Amennyiben szeretnéd, parancssorban is megadhatod az induló könyvtárat. Ha például a Linux-rendszermag forrásának könyvtárából szeretnél kiindulni, a következő sort írd be:

```
fsv /usr/src/linux
```

Nem Daniel az egyetlen, aki Linuxát háromdimenziós rendszerként képzeli el. Egy másik érdekes projekt a találó ***3dfile*** nevet kapta. A 3D-látvány ismét az OpenGL, valamint a Mesa és ***Mircea Mitu*** ajándéka.
Mircea Mitu siet leszögezni, hogy a 3dfile nem kimondottan



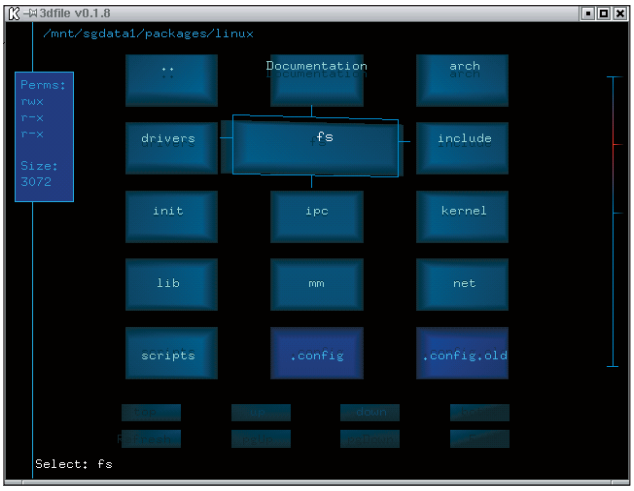
1. kép Íme, a város – állományrendszered így jelenik meg az fsv-ben

állománykezelő program, bár rendelkezik néhány ügyes megoldással (az előre-hátrahintázó ikonokon túl is). Ha például egy vörös tömbre kattintasz (amely a futtatható állományokat jeleníti meg), elindíthatod vele az alkalmazást. A Maelstrom nevű játékot is így indítottam – elvesztegetve vele egy csomó időt. Esetleg próbálj ki egy másik módszert: egy jobb egérkattintás az állomány nevén, és egy kis helyi menü jelenik meg. Amennyiben éppen szövegről van szó, szöveges állományként nézheted meg. A másik nézet hexadecimális, amely jó módszer kevésbé műszaki beállítottságú ismerőseink elkápráztatására. Nyisd ki az állományt a hexanézőkével, hosszasan tanulmányozd, végül mondj valami ilyesmit: „Á, igen, megvan a hiba”. A rendszergazdák feladata a mítosz életben tartása, non?
Vess egy pillantást a *2. képre*, majd dobjuk össze együtt ezt a kis receptet, és keltsük életre a 3dfile-t. Meg kell látogatnod a ➲ http://web.ss.pub.ro/~mrns/3dfile címet, ahonnan beszerezheted a forrást. A telepítés is zökkenőmentesen zajlik, bár a már hagyományosnak tekinthető `./configure` előtt akad egy érdekes lépés:

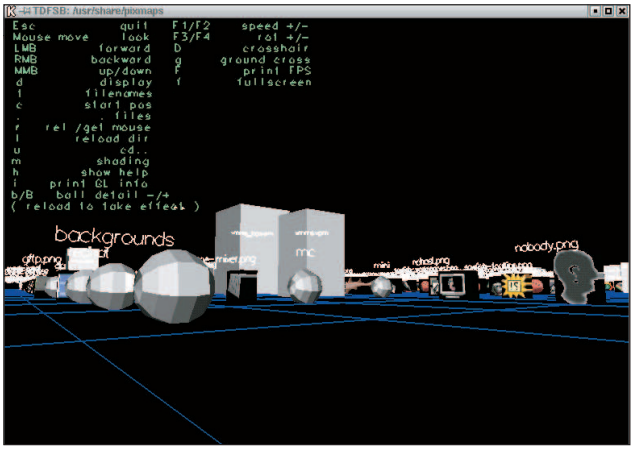
```
tar -xzvf 3dfile-0.1.8.tar.gz
```

```
cd 3dfile.0.1.8
./autogen.sh
./configure
make
make install
```

A program futtatása olyan egyszerű, mint a `3dfile` parancs beírása. A programban való eligazodáshoz elég, ha az ember az ösztöneire hagyatkozik, egyedül az alul található gombsor nem nyilvánvaló elsőre. A célkeresztet fölöttük mozgatva az irányítást még egyszerűbbnek fogod találni.



2. kép Forgó állománytömbök a 3dfile-ban



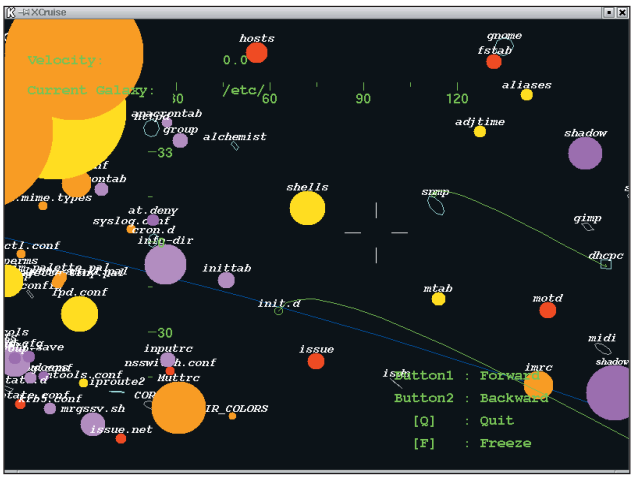
3. kép A TDFSB által elénk tárt háromdimenziós táj

A menü következő fogása a rendszer egy újabb lehetséges nézetét kínálja. ***Leander Seige*** TDFSB programjának érdekessége, hogy igen élvezetes módon teszi lehetővé utazásunkat az OpenGL előállította állományrendszerben. A könyvtárak az úszó rácstájkép felett ezüst gömbökként lebegnek. Amit különösen érdekesnek találtam, akkor válik láthatóvá, amikor az ember grafikus vagy képállományba botlik. Az állományt a program egy háromdimenziós képpé képezi le, amit körbe lehet járni. A *3. kép* bepillantást enged a TDFSB látványvilágába.
A TDFSB fordítása egyszerű, de figyelmeztetlek, hogy Leander forráskódja nem biztosít könnyen használható eszközt a programfordításra. A README csak azt tartalmazza, hogy mire lesz szükséged a fordításhoz, és azt is nagyon szűkszavúan – azért csak olvasd el. Most röviden elmondom, hogy a saját rendszeremen hogyan zajlott a beüzemelés.
Kezdd a TDBFS internetes oldalán, és töltsd le a forrást. A TDBFS kezdőlapja a ➲ http://www.hgb-leipzig.de/~leander/TDFSB címen található. Ezenkívül már csak a gdk-pixbuf csomagra volt szükségem (a Gnome része):

```
tar -xzvf tdfsb.tgz
cd tdfsb
```

Most kell elolvasnod a README állományt. Ennek alapján elkészítettem az alábbi egysoros telepítő parancsállományt. Ne felejtsd, hogy bár a szövegből ez első ránézésre nem nyilvánvaló, egyetlen sorról van szó. A parancs lényegében a README állomány részeinek másolásával állt elő. Ja, igen, a próbát egy hordozható gépen folytattam RedHat 7.1 alatt:

```
gcc -L/usr/X11R6/lib -I./ -lGL -lGLU -lglut
➥-lXmu -lXi -lXext -lX11 -lm -lgdk_pixbuf
➥-march=i686 -malign-loops=4 -malign-jumps=4
➥-malign-functions=4 -O6 -fomit-frame-pointer
➥-fno-strength-reduce -x c -o tdfsb tdb3.c
➥-I/usr/include/gdk-pixbuf
➥-I/usr/include/glib-1.2 -I/usr/include/gtk-1.2
➥-I/usr/lib/glib/include
```



4. kép A /etc galaxisban az XCruise-zal cirkálva

Ezután futtasd a programot a `./tdfsb` paranccsal. Rövid lebegő súgó megjelenítéséhez a ***h*** gomb megnyomását használhatod. Mivel a program világának navigációs eszköze az egér, használata az ***r*** gombbal váltogatható. A próbarendszerem nem rendelkezett 3D gyorsítókártyával, de nincs kétségem afelől, hogy egy ilyen kártya előnyei jól kiaknázhatók a TDFSB használatakor.
Bár a záróra vészesen közeleg, engedjétek meg, hogy egy teljesen új rendszerszemléleti lehetőséget mutassak be nektek. Linuxban még járatlan barátaink időről időre hallhatják a megjegyzést: a Linux olyan, mint egy teljesen új világ. A most felfedezett eszközök megmutattak néhányat azok közül a lenyűgöző megjelenési formák közül, amelyekben ez a világ megnyilvánulhat. Világ? Miért nem Naprendszer vagy Világegyetem? Kétségtelenül ilyen gondolatok járhattak ***Yusuke Shinyama*** fejében, amikor az XCruise-t megalkotta. Ez a nagyszerű kis csomag olyan állománykezelőt rejt, amelyben a lemezedet úgy járhatod be, mintha önálló világegyetem lenne.

Minden állományrendszer egy galaxis, az állományok csillagok (nagyobb állomány = nagyobb csillag), a közvetett hivatkozások pedig féreglyukként jelennek meg. Szerény szakácsotok és hűséges pincére így navigálva kellemes órákat töltött el a maga Linux-rendszerében.
Már alig várjátok, hogy a saját világegyetemeteket futtassátok? Pillantsatok a *4. képre*, majd látogassatok el a ➲ http://www.unixuser.org/~euske/pub/ címre.
A fordítás maga az egyszerűség. Kövessétek ezt a kis receptet és az ebéd csaknem el is készült.

```
tar -xzvf xcruise-0.24.tar.gz
cd xcruise-0.24
xmkmf
make
```

A kapott futtatható állomány a könyvtárunkban csücsül, és oda másolhatod, ahova akarod. A futtatáshoz gépeld be a `./cruise &` parancsot. Előfordulhat, hogy az XCruise kezdeti megjelenése túl nagy lesz a képernyődhöz, de az X beállításaival szerencsére kiválasztható a megjelenítődhöz illő méret.
Az én táskagépem 1024×768-as megjelenítővel bír, de én csak egy 800×600-as területet akartam az XCruise-zal kitölteni. A programot ehhez így indítottam el:

```
./xcruise -geometry 800x600+0+0
```

Magnifique! A bal egérgombot használva hihetetlen sebességgel száguldoztam előre, a középső gombbal (vagy a kettővel egyszerre, ha az egér kétgombos) pedig ki-bemozoghattam. A görgetőgombok a szöget és az irányt változtatták. Gazdag és felettébb izgalmas ez a Linux-univerzum, non?
Á, mes amis, sajnos itt a záróra. Ideje, hogy François utoljára töltse tele a poharaitokat és visszatérjünk a való világba, amelyre a mai alkalom után talán már nem is tudtok ugyanúgy tekinteni. A következő alkalomig au revoir, mes amis. A votré santé! Bon apétit!

Marcel Gagné (mggagne@salmar.com) Mississaugaban (Ontario, Kanada) él, a Salmar Consulting Inc. rendszerépítéssel és hálózati tanácsadással foglalkozó cég elnöke. Pilóta és sci-fi író egy személyben. A világhálón elérhető honlapján sok hasznos dolgot találhatunk. ➲ http://www.salmar.com/marcel/

Kapcsolódó címek

3dfile Web Site ➲ web.ss.pub.ro/~mms/3dfile

fsv (3-D Filesystem Visualizer) ➲ fsv.sourceforge.net

tdfsb (3-D Filesystem Browser) ➲ www.hgb-leipzig.de/~leander/

TDFSB The WINE Headquarters ➲ www.winehq.com

XCruise ➲ www.unixuser.org/~euske/pub