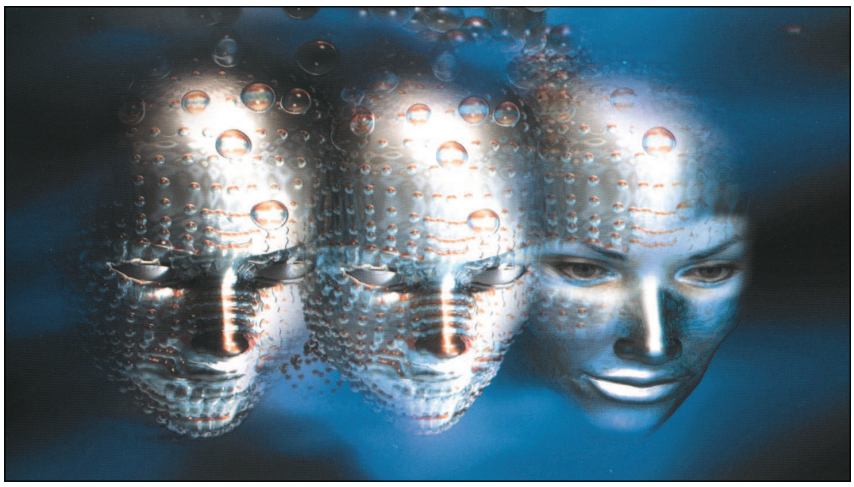
Változások



„Minden gyökeres változást megelőz egy passzív, egyet nem értő, de nem kihívó beállítottság az emberek részéről (…) elveszettnek kell érezniük magukat, hogy el legyenek szánva: szakítanak a múlttal és szerencsét próbálnak a jövővel.” (Saul Alinsky)

Kedves Mindenki! Lapunknál állandóan változik valami. E havi lapszámunkban például *Csontos Gyula*, aki idáig a lemezmellékleteket, a híreket és a benti munkaállomásokat állította össze, magára vállalta a lap szakmai szerkesztését. Különösképpen örülök, hiszen Gyula tapasztalt rendszergazda és programozó egy személyben. A munkakör változásának mellékhatásaként egy rövid ideig elcsendesedett a ➲ linuxvilag.hu, szerencsére azonban a honlap főnixmadár módjára újjáéledt, és lapunk megjelenésekor a hírezés ismét működik. Gyula a ➲ csontos.gyula@linuxvilag.hu címen érhető el.

Új szakmai szerkesztőnk évek óta Linux-párti és nagyon sokszor előfordult, hogy a hozzánk beérkezett próbagépeket szintén ő nyúzta meg. No igen, sajnos számos olyan gép létezik, amelyekkel nem bánhatunk kesztyűs kézzel, és ezen a ponton ismét a Linux és a többi rendszer felmutatta tulajdonságok különbségeivel találjuk szemben magunkat. Hogy mit is értek ez alatt? Hadd meséljek el ehhez kapcsolódóan egy röpke történetet! Talán két éve történt, hogy új munkaállomást (Windows) és új kiszolgálót (Linux) kellett belső rendszerünkbe beállítanunk. A kiszolgálóra főleg Samba-feladatokat osztottunk. Egy erős ASUS-os gépet választottunk kiszolgálónak, egy Aristo-alaplaposat pedig munkaállomásnak. A nagygéppel semmi baj nem volt, a kicsire azonban sehogy nem akart felmászni a Win98. Gondoltuk, nem baj, rakunk rá NT-t. Működött is a drága, két napig, utána lehalt, ahogy az a nagykönyvben meg van írva. Néztük balról, néztük jobbról, majd a harmadik telepítésnél gondoltunk egyet, és kicseréltük a két gépet. Egy fájlkiszolgálónak különben sem kell erős gép! No, futott is a Linux, de egy jó hónap múlva furcsaságokat tapasztaltunk. El-elszállt egy-egy folyamat, időnként „Seek error” magüzenetek jelentek meg. Hmm, majd kinövi – képzeltük. Eltelt további két hét és komoly lemezhibák jelentkeztek. Egy szó mint száz, a legvégén kiderült, hogy az alaplapon lévő IDE-vezérlő néha tévedett, mi pedig minden olyan anyagot elvesztettünk, amiről nem volt biztonsági mentésünk.

Szóval, nem mindig a Linux a ludas, ha nem akar futni egy gépen. A minap leveleztem *Lakos Imi*-vel is, aki kedvencünket élesztgette, de nem sikerült neki. Sejtelmes fagyások történtek, és már-már lemondóan írta, hogy csalódott a pingvinben, ennyit még a Windows sem fagy. A kérdéses gép egy Gigabyte/ALI alaplap S3 Savage 4 kártyával. Kártyacsere után örömmel írta, hogy a Voodooval repül a gép.

De kanyarodjunk el az alkatrészektől az elmélet felé. Nagyon örültem, amikor *Szaló István* cikkét olvastam, szerintem a Linux szempontjából legfontosabb kérdést boncolgatja: milyen helyzetű a számítástechnikai oktatás, tanítanak-e jelenleg Linuxot, és ha igen, akkor mennyire gyakran, milyen mélységben és ami még lényegesebb, hogyan? Mostani számunkban már a cikk második részét olvashatjuk, a 16. oldaltól kezdődően. Vitaindító, egyben gondolatébresztő is, és felmerül a kérdés: az API-k vajon tényleg a programozók elkorcsosulását okozzák? A napokban erről beszélgettem egy ismerősömmel, aki szerint ma már „nincsen szükség programozó polihisztorokra”, nem kell, hogy valaki a rendszerprogramozástól az SQL-en át a Perlig mindenhez értsen, elegendő ha kidolgozott elméleti alapra támaszkodik, és megfelelően képzi magát egy adott területen. Igaz, ha valahol programozóként helyezkedünk el, általában egy adott területen dolgozunk. Egy ideig – azután váltunk. Mi most például vért izzadunk a kereskedelmi részleg belső rendszerével, magunk fejlesztettük Delphi-InterBase környezetben. Ami színtiszta Delphi-kódolásnak indult, jelenleg PL/SQL- és PHP-programozással, Linux rendszergazdai feladatokkal, héjprogramok írásával bővült ki. És akkor nem is beszéltünk még az új elosztott objektumok kezeléséről, valamint azokról az újabb eszközökről, amelyek felé kacsingatunk (CORBA vagy SOAP, Zope stb.).

El ne feledkezzünk arról az apróságról, hogy egy programozó rendszerint vért izzad, mert lassú a programja, hetekig dolgozik, hogy a termék teljesítménye növekedjék (ez nagyon helyes nevelési szempontból, de nem mindig kifizetődő), ahelyett, hogy odaállna a főnöke elé, és azt mondaná: „Én ennyiért és ennyiért a kétszeresére gyorsítom a rendszert, de feleannyiért ötszörösére lehetne felgyorsítani a gépet!” És aki nem hiszi, hogy ez előfordulhat, annak csak annyit mondok, hogy láttam én már olyan hálózatot, ahol a központi adatkiszolgáló egy Pentium volt, a munkagépek pedig Pentium III-asok, mert „azokhoz kaptunk jó monitorokat”.

Így hát mindenkinek azt ajánlom, legyen nyitott, és ne csak a saját területére figyeljen, hiszen sokszor lényegesen hatékonyabb megoldásokra lelhetünk, ha nem csak az adott szinten keresgélünk utánuk! Remélem, ehhez kapcsolódóan egy-két hasznos megoldással újságunk is szolgálni tud.

Jó olvasást kívánok!

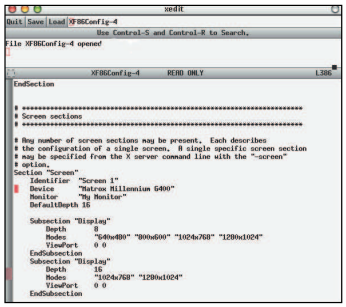
Szy György
a Linuxvilág főszerkesztője, a Kiskapu Kiadó vezetője. Mindenki véleményét és levelét örömmel várja az alábbi levélcímen:
Szy.Gyorgy@linuxvilag.hu

Nyárias az idő – mondjuk naponta homloktörölgetve, szerkesztőségünkben azonban már megjelentek az első levelek. Egy olvasónk több kérdéssel is fordult hozzánk, az alábbiakban neki válaszolunk.

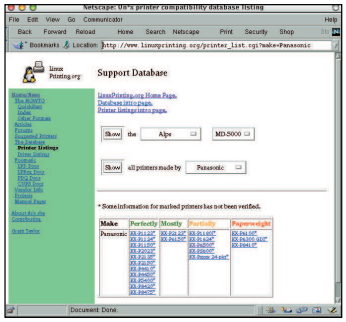
Hogyan tudnám a megszokott virtuális asztalt beállítani? A 4-es X teljesen figyelmen kívül hagyja ugyanis a Virtual 1024 768-as sort! Egy 14 hüvelykes monitorhoz lenne ehhez a beállításhoz szükségem, mivel így megnövelhetem a képernyőt, és nem kell új monitort vennem.
lakosimi@freemail.hu

A legegyszerűbb megoldás, ha a /etc/X11/XF86Config-4 fájlba egy az alábbihoz hasonló bejegyzést teszel:

```
Subsection "Display"
          Depth       16
          Modes       "1024x768" "1280x1024"
          ViewPort    0 0
     EndSubsection
```



A Modes sor adja meg a használandó felbontásokat, amit elsőként írsz, az lesz az alapértelmezett mód, tehát azzal indul el a grafikus felület. A második pedig már (abban az esetben, ha nagyobb, mint az első) a virtuális képernyőméretet adja meg – erre a CTRL+ALT és + billentyűkkel válthatsz át.



Panasonic gyártmányú nyomtatók támogatottsági listája

Valahol azt olvastam, hogy a nyomtató egyszerre kétféle üzemmódra is beállítható, tudniillik szöveges (300x300 dpi) és „képnyomtatás” (600x600 dpi) üzemmódra. A kérdés csak az, hogyan lehet ezt kihasználni? Nekem Panasonic KX-P6500-asom van, amivel minden gond nélkül tudtam nyomtatni (bármely Linuxszal), a HP laserjet, illetve a laserjet beállításokkal. Baj csak akkor van, ha mondjuk kép is található a kinyomtatott oldalon, mert a végeredmény igen-igen „szemcsés” lesz.

Ez a nyomtatótípus támogatott ugyan, de csak részben. Ez azt jelenti, hogy grafikus munkára valószínűleg nem tudod fogni. Az alábbi címen megnézheted, milyen Panasonic nyomtatókat hogyan támogat a Linux.
➲ http://www.linuxprinting.org/show_printer.cgi?recnum=65280
Érdemes szemezgetni még a ➲ http://www.cups.org oldalon vagy a ➲ http://www.linuxprinting.org-on is.

Négy számítástechnikai boltból háromban szinte nem is hallottak a Linuxról. Csupán egy volt (Mikland), ahol azt mondták: „nem gond…” – mármint a Linux beszerzése, mivel vásárolandó gépemre azt szerettem volna. A többi boltban pedig választani lehetett Windows és Windows közül, és hogy el ne felejtsem: ott volt még a Windows is.

Mindemellett még megjegyezték: „Á, a Linuxra nincsen alkalmazás, mert minden programot Windowsra írnak”. Ekkor én (tettetve a tudatlant) azt mondtam: „Egy barátomnál pedig láttam valamilyen Debian, vagy valami ilyesmi Linuxot, ami 3 darab CD-n volt, ez mintegy 1950 megabájtot tesz ki. Tessék mondani, ilyen nagy lenne az alaprendszer?” Itt is a tudatlanság győzedelmeskedett!

Olvasónknak sajnos teljesen igaza van abban, hogyha az ember Linux-ügyben bármilyen tájékoztatást szeretne kapni, netán vásárolna, akkor csak igen szűk körben tud válogatni. Manapság még mindig a Microsoft operációs rendszerei az elterjedtek, ezért a számítástechnikai boltok nem is igazán igyekeznek, hogy a Linuxról ismereteket szerezzenek. A másik tapasztalatunk, amelyről e számunk lapjain a Portocom-tesztnél („Az első találkozás: Kovács úr Linuxra vált”, 12. oldal) és a Hír-lelő rovat oldalain is beszámolunk, hogy előretelepített linuxos gépeket csak ritkán adnak el (szerencsére már több cég hirdet ilyen gépeket, ezek valószínűleg irodai célokra is megfelelőek lesznek), ugyanis aki kicsit is ért a rendszerhez, gépére saját kezűleg fog Linuxot telepíteni. A linuxos alkalmazásokról pedig csak annyit, hogy a Debian alaprendszer több mint 3500 csomagot tartalmaz, és ha elgondoljuk, hogy ezek a csomagok mind valamilyen feladatot látnak el, általában pedig nem ugyanazt, akkor nyugodtan elmondhatjuk, hogy bizony Linuxhoz is akad elegendő megfelelő mennyiségű és minőségű alkalmazás. Természetesen akadnak még hiányosságok, de a nyílt fejlesztésnek köszönhetően gombamód szaporodnak a különféle feladatra alkalmas programok.

(A leveleket – az újság főszerkesztőjének elvei szerint – esetenként átszerkesztjük.)

Európa első IP-alapú multimédiás ügyfélkapcsolati központja

Nemrégiben megszületett Európa első IP-alapú multimédiás ügyfélkapcsolati rendszere, a 3Com® Contact Advantage™, mely a 3Com NBX® telekommunikációs rendszerének és az Apropos Technology ügyfélkapcsolatokat és -tevékenységeket kezelő termékének az ötvözete. A rendszert a közelmúltban helyezték üzembe az Egyesült Királyságban működő új lakossági hitelintézménynél, a Halifax Cetelem Credit Ltd-nél. A 3Com Contact Advantage használatával a Halifax – az ígéretek szerint – teljes körű kapcsolattartását zökkenőmentesen bonyolíthatja le. A 3Com olyan megoldáscsomagot ajánl, amely biztosítja a multimédiás ügyfélkapcsolati központ működtetéséhez szükséges felületet, alkalmazást és tanácsadást, valamint a megvalósítást. A rendszert három szintre építették fel. Az első az IP Contact, mely egyesített hálózati telefonrendszert foglal magába – támogatja a beszéd, az e-mail és a fax átvitelét. Ezeket közös városorba rendezi és a kezelő egyetlen felületen látja őket. A második szint a Web Contact, melynek szolgáltatásai között a visszahívás, az internetes csevegés és a közös böngészés szerepel. A harmadik szint pedig az Ultimata Contact, ami az eCRM-en keresztül teszi lehetővé egy vállalati szintű adatbázis létrehozását, az ügyfelek hívási szokásainak és viselkedésének megismerését.

A Compaq támogatja a hazai felsőoktatást

Áprilisban a Compaq Computer Magyarország Kft. nagyteljesítményű Alpha ki-

szolgálókra hirdetett nyílt pályázatot a hazai felsőoktatási intézmények körében. A húsz nyertes intézmény az AlphaServer DS20-as kiszolgálók 3 évre szóló, térítésmentes használati jogát kapta meg. Az AlphaServer DS20 6/500-as számítógépek 64 bites Alpha processzorral rendelkeznek, legnagyobb méretük eléri a 4 GB-ot (6 PCI bővítő hellyel). Mindehhez egy Compaq Campus licence csomag is tartozik, Tru64 UNIX vagy Open VMS (a Compaq által fejlesztett) operációs rendszerrel.

Portocom MultiMagic – a karcsúság titka

2001. augusztus 1-jén a Portocom Rt. új multimédiás noteszgépet mutatott be Portocom MultiMagic néven. A hordozható gépeknél a könnyen kezelhetőség, a méret (35 mm vastag) és az akkumulátor teljesítménye (2,5 óra) a legfontosabb szempont, és ez ennél a típusnál kecsegtetően jó eredményeket mutat. Az újdonságok közé sorolható, hogy a CD még a gép becsukott állapotában is hallgatható; valamint rendkívül hasznosak a számítógép elején elhelyezett, öt azonnali parancsot végrehajtó ún. indításbillentyűk (shortcut keys), melyek gyors kapcsolatot jelentenek a levelezőrendszerhez, az Internethez, és két szabadon választott, a gépen futó alkalmazáshoz, egyikükkel pedig a hangerő szabályozható közvetlenül. A gép szíve a Pentium III-as mikroprocesszor, valamint a VIA PN133-alapú lapkakészlet. Memóriamérete akár 1,5 GB-ig, merevlemezéé pedig mintegy 30 gigabájtig bővíthető. A hajlékonylemez-meghajtón kívül az alapgép CD-meghajtót vagy DVD-meghajtót, illetve CD író-újraírót vagy kombinált DVD-olvasó/CD írót tartalmazhat. A legjobb képminőség elérése érdekében három képernyőméret közül választhatunk (13,3; 14,1 vagy 15,0 hüvelykes TFT-kijelző, 1024x768-as /XGA/ illetve 1400x1050 /SXGA/ képpontos – az elérhető legjobb – felbontással).

A MultiMagicet a legkorszerűbb eszközök csatlakoztatására is felkészítették: összeköthetjük hagyományos, vagy akár plazma-televízióval, videókivetítővel, videokamerával, digitális fényképezőgéppel, mikrofonnal, fejhallgatóval, de LCD-képernyővel is. És ami nekünk a legfontosabb: a Portocom ezt a hordozható gépét linuxos operációs rendszerrel is kínálja a számunkra.

PC Expo 2001 – New York City

A Hewlet-Packard a New York Cityben megtartott PC Expo 2001-en mutatta be legújabb fejlesztésű üzembehelyezés-támogató megoldását, amely az ügyfelek számára egyszerűbbé teszi a többféle operációs rendszert futtató hálózatokban működő számítógépek távoli üzembe helyezésének, irányításának és helyreállításának folyamatait. A Rembo Auto Deploy és Rembo Auto Backup kialakítású megoldás segítségével akár több ezer, egyedi beállítású személyi számítógépet is használhatunk. A tervek szerint a HP ezzel a megoldással az elkövetkezendő hat hónap során világszerte 70 ezer számítógépet helyez majd üzembe. Az üzembehelyezés-támogató megoldás a HP és más szállítók mind Linux-alapú, mind pedig Microsoft operációs rendszereken futó asztali és noteszgépeit támogatja.

Körösztös Anita (Korosztos.Anita@linuxvilag.hu) a Linuxvilág marketingfelelőse. Nem olyan rég csöppent bele a Linux világába, azóta lelkesen foglalkozik vele. A barátaival töltött időt tartja a legfontosabbnak és lelkesedik a vízi- és téli sportok iránt is.

Dell-gépek Linux nélkül

A kis keresletre való tekintettel a Dell számítógépeit nem szállítja többé előre telepített Linux-rendszerrel. Saját tapasztalatom azt mutatja, hogy az emberek jól megszokott rendszerükről nem mernek váltani, aki pedig egy kicsit is ért a Linuxhoz, rendszerét kedvenc Linux-változatából saját maga szereti felépíteni.
➲ http://www.linuxnews.com/stories.php?story=01/08/06/2217404

Ingyen programok Solarishoz

Aki a Sun cég Solaris nevű operációs rendszerét használja, biztosan hasznosnak fogja találni az alábbi oldalt. Szabadon letölthető és ingyenesen használható programok találhatók rajta – kategorizálva. Egyszerű menüpontokon lépkedve érdeklődésünk tárgyát könnyedén megtalálhatjuk.
➲ http://freeware4sun.com

RedHat 7.2 próbaváltozat

Letölthető és kipróbálható a RedHat 7.2 ROSWELL-próbaváltozata. A rendszer alkotóelemei: 2.4.6-os rendszermag naplózó fájlrendszer-támogatással, XFree 86 4.1.0, KDE 2.2 pre, Gnome 1.4.
Letölthető az ➲ ftp://ftp.redhat.com/pub/redhat/linux/beta/roswell/ és a
➲ http://www.redhat.com címekről.

Processzorhűtő energiafogyasztás nélkül?

Hazánkban először *Ordasi Gábornak* (a ➲ www.szamitogep.hu munkatársának) sikerült nagy hőelvezető képességű, néma és energiafogyasztás nélküli processzorhűtőt készítenie. Nincs mozgó alkatrésze, így nagyon kicsi a meghibásodás lehetősége. A hűtés alapelve: a processzor által termelt hő felmelegíti a ráerősített tartályban lévő folyadékot, ennek következtében az felforr és felfelé száll, ahol lehűl, majd visszafolyik a processzor tartályába. Ez a zárt rendszerű hűtőrendszer karbantartást nem igényel, élettartama pedig akár 50–100 év is lehet. További részletek a
➲ http://www.szamitogep.hu overclock rovatában.

Elektronikus Pearl Harbor

Nagyon érdekes írást olvashatunk az alábbi címen arról, hogy a Microsoft túlélne-e egy esetleges elektronikus Pearl Harbort. A szerző közérthető módon hasonlítja össze a Unix- és Windows-rendszereket. A cikk egy kis betekintést nyújt a Windows Registrybe, miért is omlik össze a rendszerünk, ha megsérül, és

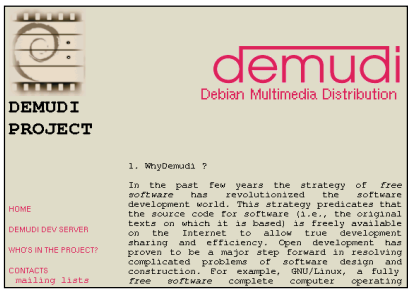
vajon mit csinálhatnak benne azok a részek, amelyek nem nyilvánosak?
➲ http://www.yowusa.com/Archive/May2001/CYBERW_1/cyberw_1.htm

Kétfejes Matrox G400/450

A ➲ http://www.debianplanet.org HOGYAN-jai között található egy nagyon hasznos leírás a kétfejes Matrox kártyák használói számára. Ennek segítségével tökéletesen beállíthatjuk a kártyáinkat a 4.1.0-s Xfree86 grafikus felülettel. Azok is ötletet meríthetnek belőle, akik két külön kártyát szeretnének használni.
➲ http://www.debianplanet.org
➲ http://www.debianplanet.org/debianplanet/article.php?sid=256&mode=thread&order=0
➲ http://www.matrox.com

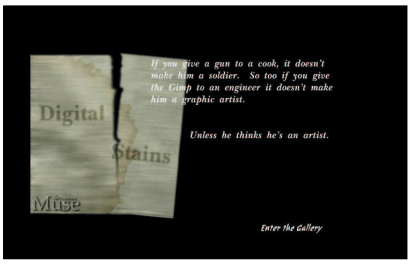
Debian

Egy újabb Debian-alapú Linux tűnt fel, a neve DeMuDi – ez a Debian Multimedia Distribution név rövidítése. A projekt fő célkitűzése, hogy könnyen beállítható és a multimédiára kihegyezett Linuxot adhassanak a felhasználók kezébe. Ez a Debian-változat minden olyan programot tartalmaz, amely e feladat megvalósításához szükséges: MP3-lejátszókat, videolejátszókat, valamint hang- és képszerkesztőket.
➲ http://www.demudi.org
➲ http://gige.xdv.org/pages/DeMuDi



Gimp

A Gimp (lásd még a 64. oldalon), ez a méltán népszerű képszerkesztő program szinte minden operációs rendszer alá elérhető. Legfrissebb kiadása a 1.2.2-es, amely számos hibajavítást tartalmaz, de a fejlesztők már gőzerővel dolgoznak az 1.4-es változaton. Ígéreteik szerint az újabb Gimp már CMYK színkezelésre is képes lesz, így az egyébként sem kis tudású programot akár nyomdai előkészítésre is használhatjuk majd. Nagyon sok internetes oldal foglalkozik a különböző Gimp-változatokkal, íme néhány a teljesség igénye nélkül:



➲ http://www.gimp.org
➲ http://www.macgimp.org
➲ http://www.gimp.hu
➲ http://gimp.rulez.org

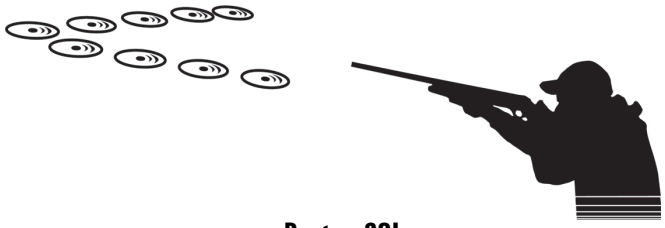
Kicsi, de mégis nagy

Szingapúrban egy újabb Linux-alapú tenyérgép született. A termék neve Terapin Mine és leginkább mobiltelefonra hasonlít. Képes MP3- és WAV-lejátszásra, emellett diktafonként is használható, ráadásul 10 GB-os tárolókapacitásának köszönhetően hordoz-

ható adattárolóként szintén megállja a helyét. PCMCIA csatolón keresztül hálózaton is elérhetővé lehet tenni. A Mine 18 cm hosszú, 9 cm széles és 2,6 cm vastag. Kezelése három gomb segítségével történik, érdekes azonban, hogy csak a Windows 98/Me és 2000 operációs rendszerekkel működik együtt. Kijelzője összesen négyszer húsz karakter megjelenítésére képes.
➲ http://www.mineterapin.com

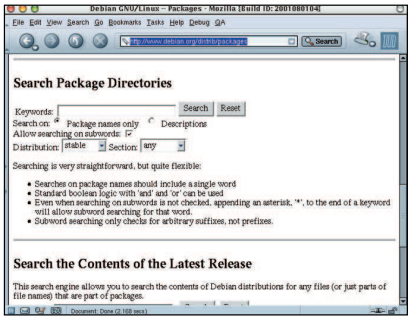
Csontos Gyula (Csontos.Gyula@linuxvilag.hu) a Linuxvilág szakmai, hír- és CD-szerkesztője. Szabadidejében szívesen mászik hegyet és kerékpározik.

Programvadászat



A 15. CD-melléklet Magazin könyvtárában a cikkekhez kapcsolódó programokat és forráskódokat, valamint a kimaradt listákat találhatják meg – természetesen ezt minden cikk szövegében az elérési úttal együtt jelöltük, ezzel is segítve a programok, forráskódok kipróbálását és megértését. Telepítési és beállítási ötleteket általában a cikkekben találhatunk. Előfordulhat, hogy azokon a rendszereken néhány, a telepítéshez szükséges csomag hiányzik, ilyenkor ezeket meg kell keresnünk és telepítenünk kell a folytatás előtt. Az rpm-alapú csomagokhoz a ➲ http://www.rpmfind.net címet ajánlanám,

ahol kereshető adatbázisban tudunk kutatni a fájlok után; és ha a rendszer a keresett csomagot megtalálta, azonnal le is tölthető. Deb-alapú rendszerekhez pedig a ➲ http://www.debian.org/distrib/packages oldalon kereshetünk. Előfordulhat az is,

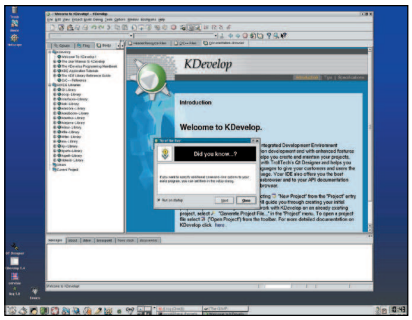


hogy Potato-rendszerünkben nincs meg az a friss változatú csomag, amire szükségünk lehet, ekkor két ösvényen haladhatunk tovább:

- Állítsuk be a /etc/apt/sources.list-ben a Woody-kiadás könyvtárait, majd az `apt-get update` és az `apt-get upgrade` (esetleg az `apt-get dist-upgrade`) parancsokkal frissítsük rendszerünket. Ezt a megoldást azonban csak azoknak ajánlom, akik nem életbevágóan fontos programokat futtatnak a gépükön, mivel a Woody egyelőre csak *Testing*-változat, azaz nem megbízható állapotú. (Saját tapasztalataim azt mutatják, hogy munkaállomásnak már kellőképpen megbízható.)
- A másik megoldás pedig az, ha a hiányzó csomag forráskódját letöltjük, lefordítjuk és telepítjük.

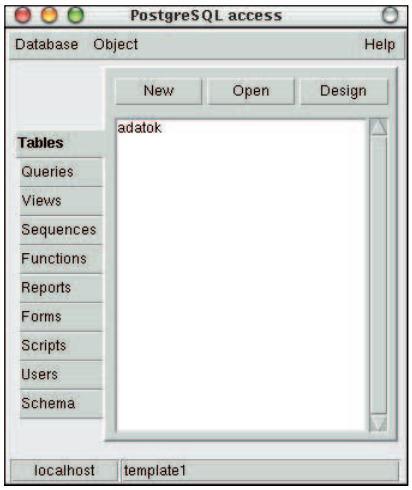
Postfix

A 66. oldalon kezdődő cikkhez tartozó források: postfix-20010228-pl04.tar.gz – a megbízható változat, snapshot-20010714.tar.gz – a legfrissebb fejlesztői változat.
Az első lista is itt kapott helyet.



KDevelop

Mustra rovatunkban a KDevelop 1.4-es fejlesztőkörnyezetről olvashatnak, ez ebben a könyvtárban telepíthető formában és forráskódként is megtalálható. A 76. oldalon kezdődő írásunk telepítési segítséget is nyújt. Mindenképpen érdemes a cikket mintegy előtanulmányként elolvasni, mivel sok buktatótól kímélhet meg mindenkit.
Jó fejlesztést!



PostgreSQL

A népszerű adatbázis-kezelő forráskódja szerepel ebben a könyvtárban, így mindenki saját igényeinek megfelelően testreszabott csomagot telepíthet a gépére.

Memóriahibák

A négyes, az ötös és a hatos lista kapott helyet ebben a könyvtárban.

MySQL

Gyors, hatékony adatbázis-kezelő rendszer, forrása és a cikkhez tartozó lista található itt.

Pov-Ray

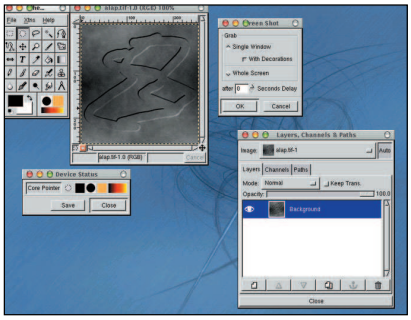
Folytatódik a megkezdett Pov-Ray cikksorozat, ehhez listák, a program forrása és képek találhatók a CD-n.

Gimp 1.2.2

A 64. oldalon található Gimp-sorozatunkhoz tartozó legújabb Gimp-változat is helyet kapott korongunkon: forráskód és rpm-csomag formájában. Telepítése az `rpm -i csomagnév` paranccsal, vagy a forrásból való fordítással lehetséges.

Frissítések

Szerencsére a nyár és a meleg nem szegte kedvét a fejlesztőknek (bár a világ néhány részén most éppen meglehetősen hideg van), ezért most is szép számmal jelennek meg a frissítések és a biztonsági javítások. Jelenlegi korongunkon a Mandrake Linux 7.2 és 8.0, a RedHat 7.1. és a Debian Potatóhoz



megjelent frissítések, illetve hibajavítások kaptak helyet. Mindenki számára ajánlott a rendszer frissítése, ily módon ugyanis számítógépünkön a biztonsági rés a lehető legkisebbre csökkenthető. Egy frissebb változatú programmal új szolgáltatásokat tudunk használni, ezzel is teljesebbé téve a minél kényelmesebb linuxozást.

A 16. CD-n találhatók nagyobb lélegzetvételű anyagaink.

XFree86 4.1.0

A Linuxnak és számos más operációs rendszernek az XFree86 az alapértelmezett grafikus felülete. Korongunkra a legújabb üzembiztos változat, a 4.1.0-s került fel, mindenki által telepíthető csomagokban és forrásban. Eddigi tapasztalataim a szerkesztőségben nagyon jók ezzel a kiadással kapcsolatban, és ami számomra a legfontosabb: ez a változat a Matrox G450-es kártyák kezelését alapértelmezetten tartalmazza, tehát a fájlokat nem kell a Matrox oldaláról letölteni és a telepítéssel bíbelődni. Természetesen számos változást és kiegészítést rejt magában az előző változathoz (4.0.3) képest, ezek többek között az alábbiak:

- Kiegészítették a grafikuskártya-meghajtókat,
- az fb-réteg használatához átírták a meghajtót,
- ATI Radeon meghajtó Alpha processzoros Linuxhoz is,
- VMware meghajtó,
- Matrox G450-, Trident CyberBladeXP- és CyberBladeXPm-, nVidia GeForce3-támogatás,
- 1400x1050-es beépített felbontás,
- Savage meghajtójavítás,
- GLINT meghajtófrissítés.

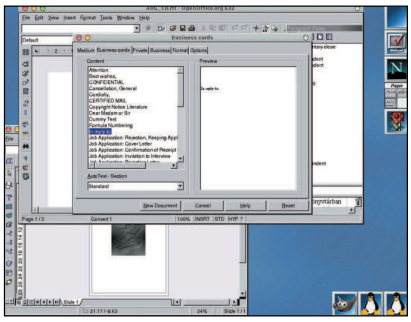
A csomag része:

- Mesa 3.4.2,
- FreeType 2.0.2,
- Támogat néhány „internet” billentyűzetet,
- Free86-VidMode bővítés.

Telepítése a rendszerünknek megfelelő könyvtárban lévő Xinstall.sh héjprogramocska elindításával egyszerűen elvégezhető. Számos kérdésre választ kell adnunk, ezt azonban rendszerünk a későbbiek során a jól sikerült beállításokkal meghálálja. Ha a telepítéssel elkészültünk és rendszerünket finomhangolni szeretnénk, esetleg valamit nem jól állítottunk be, akkor az xf86cfg grafikus vagy az xf86config szöveges beállítóprogramot használhatjuk.

OpenOffice 633

Mint már olvasóink megszokhatták, az OpenOffice állandó vendégünk a lemezmellékleten. Mivel fejlesztése nagy ütemben zajlik, és a linuxos társadalom számára nagy fontossággal bíró fejlesztésről van szó, korongunkon minden változatot közzétesszünk – mind lefordított és azonnal telepíthető, mind pedig forráskód-formában.



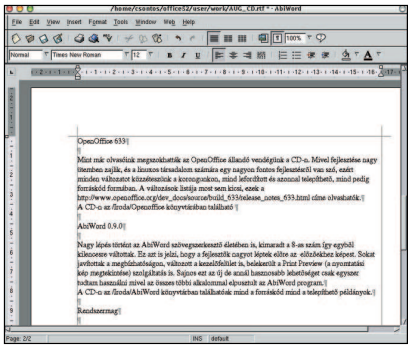
A változások listája most sem kicsi, ezek a ➲ http://www.openoffice.org/dev_docs/source/build_633/release_notes_633.html címen olvashatók.
A CD-n mindez az /Iroda/Openoffice könyvtárban található meg.

Rendszermag

A jelenlegi legfrissebb rendszermag a 2.4.7-es változat, megtalálható a CD /Rendszermag könyvtárában. A változások listája a ➲ http://www.kernel.org/pub/linux/kernel/v2.4/ChangeLog-2.4.7 címen lelhető fel.

AbiWord 0.9.0

Nagy lépés történt az AbiWord szövegszerkesztő életében is, kimaradt a nyolcas szám és egyből kilencesre váltottak. Ez azt is

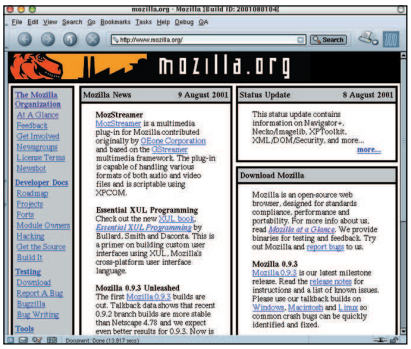


jelzi, hogy a fejlesztők az előzőekhez képest nagyot léptek előre. Sokat javítottak a megbízhatóságon, változott a kezelőfelület, belekerült a *Print Preview* (a Nyomtatási kép megtekintése) szolgáltatás is. Ezt az új, de annál hasznosabb lehetőséget sajnos csak egyszer tudtam használni, mivel az összes többi alkalommal az AbiWord program lefagyott.

A CD-n mind a forráskód, mind a telepíthető példányok az /Iroda/AbiWord könyvtárban érhetők el.

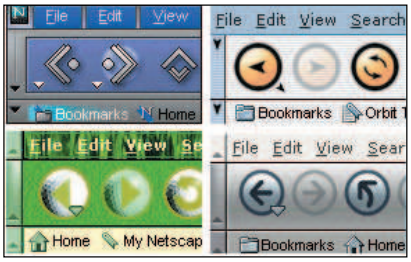
Mozilla 0.9.3

Nagyon jól sikerült ez a kiadás, amit szerencsére a változatszám is mutat, ugyanis egyre közelebb kerül a bűvös 1.0-hoz. Sokat javítottak a megbízhatóságán – a hivatalos Mozilla honlapon megjelent hír szerint sokkal használhatóbb, mint a Netscape 4.78. Ezt saját tapasztalataim is alátámasztják: ha elég memória áll rendelkezésünkre a gépben (legalább 64 MB), gyors és valóban megbízható böngészőt kapunk. A fejlesztők kiváló minőségű próbaváltozatot készítettek.



Netscape 6.1

Megjelent a Netscape-böngészők legújabb változata, a 6.1-es. A megbízhatóságon és a sebességen túl sokat javítottak önműködő telepítőjén is: nem fagy le olyan nagy összetevők telepítésekor, mint például a Java-támogatás. A témák továbbra is megmaradtak, sőt, böngészőjének felületét kész vagy maga készítette témával már mindenki kedve szerint testreszabhatja.
➲ http://www.netscape.com
➲ http://home.netscape.com/themes/index.html



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

Hiszek abban, hogy a valóságot társas módon építjük fel. Ezért azok a „tények", amelyek mindennapi életünket meghatározzák, túlnyomórészt szintén közösségileg felépített tények, így csak addig léteznek, amíg elegendő ember hiszi el, hogy igazak. A magántulajdonhoz való jog, a jog az önvédelemre és az élethez való jog – nos, ezek is társadalmilag létrehozott jogok, amelyek csak addig léteznek, amíg elegendően hiszünk az igazukban.
(Rusty Foster)

Linux az Agenda zsebtitkáron – és viszont

A jelek szerint Ázsiában szökőárhoz hasonló gyorsasággal terjednek el a Linux-alapú zsebtitkárok. Japánban a Sharp cég nemrég jelentette be, hogy egy olyan zsebtitkár piacra dobását tervezi, amely nem Palm vagy Microsoft operációs rendszert futtat, hanem Linuxot. A koreai G.Mate Yopy olyan gépecske, amely a Conversay – egy redmondi (Washington állam) cég – beszédfelismerő felületét használja. Az Ericsson Singapore és a szingapúri Centre for Wireless Communications a közelmúltban arról számolt be, hogy közös fejlesztésükben olyan DelphiPad névre keresztelt kézi számítógép született, amely tízhüvelykes érintőképernyővel rendelkezik. Értékesítését a 2001-es év utolsó negyedére tervezik, ezer dollár alatti áron. Létezik még a VTech Helio elnevezésű terméke, továbbá a Compaq iPAQ-jába, valamint az egyéb zsebtitkárokba is a beágyazott Linux számos változatát használhatjuk.

A leginkább figyelemre méltó zsebtitkár – legalábbis a Linux-közösség buzgárként feltörő lelkes támogatása után ítélve – pillanatnyilag mégis az Agenda Computing Agenda VR névre hallgató készüléke. Az Agenda tulajdonosa ugyan a hongkongi Kessell International, amely egyben a gyártást is felügyeli, jelek szerint azonban a cég vezetése a mégis a kaliforniai Irvine-beli központban van, ahol *Bradley La Ronde* elnök irányítja a vállalkozást.

Nemrég a Linux Show-n találkoztam Braddel, aki a hordozható Linux iránt legalább annyira elkötelezettnek tűnt, mint az operációs rendszer híres megalkotója. „Határozottan az a véleményem, hogy az Agenda annak a korszaknak az előfutára, amikor már nem a hatalmas vállalatok, hanem a kisebb fejlesztők fogják uralni a fogyasztói elektronikai cikkek piacát, akik kihasználják a szabadon hozzáférhető – a fejlesztői közösség által folyamatosan formált – lehetőségeket." Később rájöttem, hogy ez a sajátságos bemutató volt a legnagyobb érdeklődésre szert tevő program a Linux Show-n. Néhány nappal ezt követően az alábbi váratlan levelet kaptam:

„Múlt este, közel egy év után elmentem egy Linux-felhasználói találkozóra (➲ http://www.nblug.org/, North Bay Linux Users Group), ahol az Agenda Computing elnök-vezérigazgató fejlesztője tartott bemutatót, illetve megismertette velünk a VR3-at, azt a Linux-alapú zsebtitkárt, amelyet hamarosan meg is jelentetnek…Nem tudom, hogy korábban ismerted-e ezeket, de most még jobbak és használhatóbbak, mint amit valaha is remélhettünk."

Később a szerző hozzátette:
„Egy kicsit lassúak – a beszélgetés túlnyomó részében ennek a gondnak a megoldására vetettünk fel különféle megoldásokat. Nagyon-nagyon jó dolog volt részt venni egy igazi, a cég termékéről szóló módszertani vitában egy cég elnökével. Később beszélgettem vele és egyetértettünk abban, hogy a piaci megjelenés bizonyos területein még lehetnek gondjaik, ám a legjobban az őszinteségét értékeltem. Meglepően nyíltnak találtam."

És ez az ember csak egy volt a sokak közül. Az Agenda láthatóan ügyesen ápolja a kapcsolatait a Linux-közösséggel: már kezdetektől fogva az alacsonyabb árkategóriát (70 ezer forint körüli árral), a kisebb tudású PDA-k piacát célozta meg, készülékei pedig Linuxot futtattak, így arra ösztönözte a Linux-betyárokat, hogy különféle érdekes alkalmazásokat fejlesszenek a zsebtitkárra. A jelek szerint pontosan ez történik. Egyre több olyan független fejlesztői oldal létezik, amelyek – az Agendával együttműködve – gyorsan bővülő alkalmazásválasztékot kínálnak az Agenda VR-hoz, illetve gyaníthatóan a többi LinuxVR-alapú (➲http://linux-vr.org) eszközhöz is, így a Vadem, a Casio és az Everex termékeihez.

Ian LeWinter, az Agenda helyettes marketingvezetőjének szavaival élve az Agenda VR a Palm, a Handspring és az egyéb tenyérgépeket gyártó cégek versenytársa lesz, a Linux ama remeterákszerű tulajdonságának köszönhetően, hogy gyakorlatilag bármin, bármiben képes futni.

Maga a készülék egyébként egészében véve is jól néz ki. Mérete valóban tenyérnyi (11,43 x 7,62 x 2,03 cm), három színben kapható, a NEC 66 MHz órajelű, 32 bites MIPS processzorát használja, 8 MB RAM-mal és 16 MB flashmemóriával rendelkezik, a külső eszközökkel pedig IrDA-n vagy a saját csatlakozóján keresztül tartja a kapcsolatot. Képes hangokat is megszólaltatni. A független ➲ http://supermegamulti.com/agenda/ oldal szerint 93 Agenda VR3 program található az állandó elérésű programgyűjteményben. Ezek között 23 alkalmazás, 16 játék és 22 segédprogram rejlik. A cikk megjelenésének idejére ez a szám minden bizonnyal sokkal nagyobb lesz. Az Agenda VR-ről szóló teszt a Linuxvilág elkövetkező számainak valamelyikében fog megjelenni.

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

A két Nagy és a Kicsi

Samsung 210T

A Techdata Kft. jóvoltából kipróbálhattuk a mai monitorok csúcsának számító TFT-s kijelzők egyik kiemelkedő példányát, a Samsung 210T-t. Igazán meglepő tulajdonságokkal rendelkezik: első ránézésre már a mérete is meghökkentő, képzeljétek el, milyen lehet egy olyan monitorral szembekerülni, ami a mostani 21 hüvelykes monitorok teljes felületét (a kávával együtt) hasznos képfelületként adja, 1600x1200-as felbontásban! Analóg és digitális bemenettel egyaránt rendelkezik, így közvetlenül csatlakoztathatunk hozzá videó-, DVD- és műholdvevő készülékeket. A monitoron egyszerre két eszköz képét is megjeleníthetjük, megoszthatjuk például számítógépünk és egy videó között, akár képernyőfelezéssel, akár egy kisebb méretezhető képben nézhetjük a második eszköz képét. Kényelmünket tovább fokozandó, a készüléket távirányító segítségével a távolból is vezérelhetjük. A képernyő beállítását nyugodtan rábízhatjuk az automatikára, amennyiben ezt nem szeretnénk, az OSD menün keresztül kézzel is beállíthatjuk.

- A képernyő típusa: TFT LCD
- Mérete: 21,30 hüvelyk
- Képpontok távolsága: 0,270 mm
- Fényerő: 200 cd/m²
- Válaszidő: 50 ezredmásodperc
- Képfelbontás (legnagyobb): 1600x1200/60 Hz
- Állítási lehetőség: OSD menün keresztül
- Jelbemenet: RGB analóg/DVI-D
- Alacsony sugárzás: TCO 95 szabvány szerinti
- Energiafogyasztás (legnagyobb): 75 W
- Készenléti állapotban: kisebb 5 W-nál
- Mérete: 549x483x228 mm
- Tömeg: 11,2 kg (nettó)
- Garancia: 1+2 év

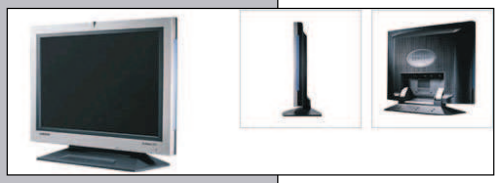
HP Business InkJet 2250 színes tintasugaras nyomtató

A másik „Nagy" szintén kiemelkedő tulajdonságokkal bír. A HP Business InkJet 2250 irodai felhasználásra szánt nagyteljesítményű színes tintasugaras nyomtató. Gyors beállítás után elérhetővé tettük a szerkesztőségben működő windowsos gépek számára, ehhez a Samba-csomagot használtuk. Az ügyfeleken is telepítettük a meghajtóprogramot és máris elkezdődhetett a próba. Minden felhasználó a saját sambás nevét és jelszavát használva érhette el és használhatta a nyomtatót. A próba során a szövegtől a legbonyolultabb színes grafikákig mindent nyomtattunk, mindegyik próbatételen kiválóan helytállt. Sebessége egyszerű szöveges nyomtatásnál a nyomtatás minőségének beállításától függően percenként 7 oldaltól 14–15 oldalig változott. Két papírtálcájának köszönhetően 500 lapot lehet behelyezni, így nem kellett attól félni, hogy naponta többször lesz szükség a papírtálcák feltöltésére. Nyomtathatunk papírra, fotópapírra, fóliára, kártyára és címkékre is. Mindent összevetve: a HP Business InkJet 2250 kiváló termék, kisebb irodáknak akár főnyomtatóként is ajánlható.

- Felbontás:1200x600 dpi
- Memória: 24 MB, mely 88 megabájtig bővíthető
- Csatlakozási felület: párhuzamos kapu (a 2250TN hálózati nyomtatókiszolgálóval is, 10/100 TX)
- Terhelhetőség: 10 000 oldal havonta
- Tápegység: beépített

Fujitsu-Siemens SCENIC T

A harmadik eszköz a címben a megtisztelő „Kicsi" nevet kapta, ez egy Fujitsu-Siemens SCENIC T asztali számítógép volt. A „Kicsi" jelző pedig nem véletlen: 64 MB RAM, 10 GB-os merevlemez és 700 MHz-es Celeron processzor, i810-es lapkakészlet jellemezte ezt a gépet. Eléggé szíven ütött és egy kicsit érthetetlen is számomra a CD-ROM hiánya. A gép előretelepített Windowszal érkezett, amit a meghajtóprogramokkal együtt CD-n is mellékeltek. A ház megbontása után beszereltünk egy CD-ROM-ot és megkezdtük a telepítést. Választásunk a Mandrake Linuxra esett – a telepítő minden alkatrészt hibátlanul felismert, így a nyomtatóval sem volt különösebb gondunk. A 64 MB memória sajnos elég kevésnek bizonyult a StarOffice 5.2 vagy akár az OpenOffice futtatásához, több memóriával (legalább 128 MB) azonban eszményi linuxos irodai munkatárs válhat belőle.



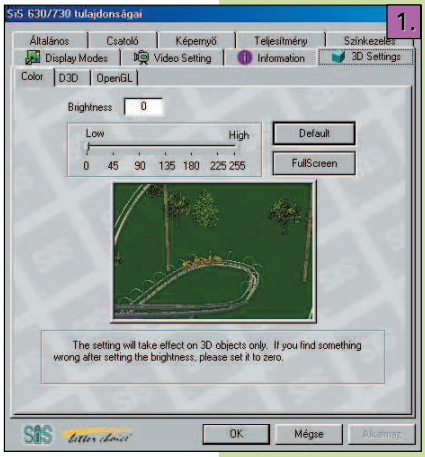
- Memória: 64 MB
- Merevlemez: 10 GB
- Processzor: Celeron 700 MHz

Adatok

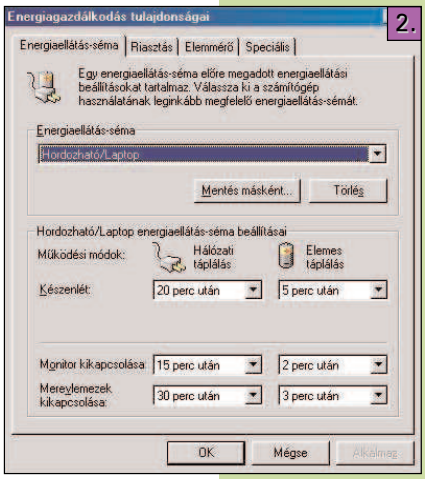
Tech Data Magyarország Kft.
telefon: 236-1900
e-mail: techdata@techdata.hu
➲ http://www.techdata.hu

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

Az első találkozás: Kovács úr Linuxra vált



1.



2.

Adatok

Portocom Rt.
Ár: 322–354 ezer forintig
Jótállás: 1–3 év garanciával
Telefon: 203-9269
E-mail: info@mail.portocom.hu
➲ http://www.portocom.hu

Nem is olyan régen történt az eset: egy japán felhasználó egy neves, világszerte ismert cégtől hordozható számítógépet vásárolt. A gépre már a gyárban Windows-rendszert telepítettek, melynek árát így eleve beleszámolták a számítógép árába. Vásárláskor természetesen ezt is meg kellett fizetni, csakhogy a történet főhőse – lelkes Linux-felhasználó – úgy döntött, hogy élni kíván a felhasználói szerződésben foglalt jogaival, így operációs rendszer és kellékeinek visszaszolgáltatása mellett visszakéri annak árát. A gyártó először megtagadta a program visszavásárlását, ám amikor az ügyfél nem tágított és a Világháló közönsége elé tárta az esetet, a cég visszakozott.

Ha nem a legáltalánosabban elterjedt operációs rendszerek egyikével szeretnénk hordozható számítógépet vásárolni, szerencsére nem kell hasonló megpróbáltatások elé néznünk. A Portocomhoz betérve és gépünkre Linuxot kérve ezen még csak csodálkozni sem fognak, hiszen a cég számítógépeit jó ideje Linuxszal is kínálja – bár honlapjukon sajnos semmilyen utalásra nem leltem ezzel kapcsolatban.

Mint *Koppány Imre* szerviztechnikustól megtudtam, szívesen telepítene sokkal több gépre Linuxot, de egyelőre meglehetősen csekély igény mutatkozik ez iránt. Egyik ismerősöm szerint vélhetően azért, mert aki hordozható gépén Linuxot is hajlandó használni, maga telepíti, aki pedig nem, az eleve nem kér ilyesmit. Mindenesetre az arra fogékonyak körében a hordozható gépeken is egyre több esetben bukkanhatunk Linuxra. Teljes joggal, hiszen ez szabad felhasználású, ingyenes operációs rendszer, és még ha némi barkácsolás árán is, képes a hordozható gépekben található összetevők megfelelő kezelésére. Ez szép teljesítmény, hiszen – főleg az alsó- és középkategóriás gépek esetében – sokszor találunk ismeretlen nevű modemet, hálózati kártyát, beépített alkatrészt, hogy a különféle egzotikus VGA-kártyákról és alaplapi lapkakészletekről ne is beszéljünk.

A tesztgép ezúttal egy egyszerűbb, de tetszetős modell, a Portocom Freestar volt. A masszív felépítésű gépbe 650 MHz órajelű processzor és 128 MB memória került, ez utóbbiból a CMOS-beállítások között tetszés szerint különíthetjük el az alaplapi lapkakészletbe épített VGA-vezérlő által használt memóriát (8, 16, 32 vagy 64 megabájtot). A vezérlő egy 13,3 hüvelykes LCD TFT panelt hajt meg, melyet 1024x768-as felbontással használtam. A gépbe a jelenleg alapfelszereltségnek minősülő összetevőket mind beépítették, így egy 56 kb/mp sebességű modemet és egy Fast ethernetkártyát is, további bővítési lehetőségei azonban viszonylag szűkösnek mondhatók, mivel csak egyetlen USB kapuval bír, és csak egy Type I/II/III-mas PCMCIA kártyát helyezhetünk az oldalsó foglalatba. Szükség szerint külső egységeket soros és párhuzamos kapun keresztül vagy infravörös átvitellel is csatlakoztathatunk hozzá. Mind a CD-, mind a hajlékonylemezes meghajtó a gépben kapott helyet, ezeket tehát nem kell külön magunkkal hurcolnunk. Apró, ám fontos kényelmi szempont: a Windows-gombokkal kiegészített billentyűzeten a magyar karaktereket is jelölték, a kurzorgombokat pedig a normál billentyűzeteken megszokott, fordított T betű alakban helyezték el.

A gyártók illesztőprogramot még mindig csak elvétve adnak vagy fejlesztenek, a támogatást pedig rendszerint nagyvonalúan a Linux-közösségre bízzák. A gép mindent (hálózati kártyát, VGA-vezérlőt, modemet) egyben tartalmaz, ezért lapkakészletének használata első pillantásra nehézkesnek tűnik, ám mint Koppány Imre elmondta, az újabb rendszermagok az összes alkatrészt kifogástalanul kezelik. Gondjai mindössze a VGA-vezérlővel akadnak: használatához apróbb kerülőutat kell tennie, de ez sem okoz nagyobb fennakadást.

Kovács úr időnként számítógépet használ. Hogy kicsoda Kovács úr? Nem számít, a lényeg az, hogy számítógépet használ, hiszen nem engedheti meg magának, hogy ne vegyen tudomást a világ fejlődéséről, ráadásul alapjában véve nyitott egyéniség. Kovács úr egyetlen olyan tulajdonsága, amely most megragadhat bennünket, az, hogy nemigen konyít a számítógépekhez.

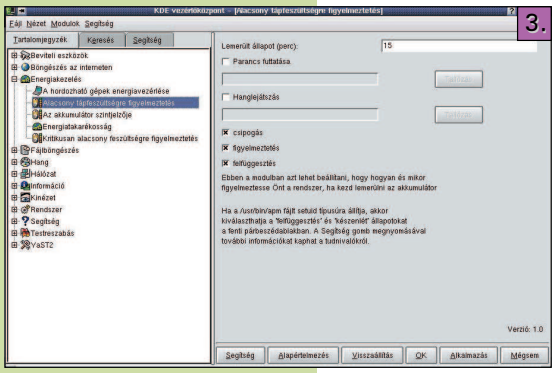
Kovács úrnak a minap apróbb gondja akadt otthoni gépével, ezért áthívta egy ismerősét, akitől segítséget remélt. Az ismerős – a gép rendberakása közben – valami Linuxról magyarázott, ami ingyenes, és rendszermagot, ablakokat, meg közösséget emlegetett. Kovács úr nem sokat értett az egészből, de nyitott személyiség lévén úgy gondolta, időszerűvé váló gépvásárlását ennek az „izének” a kipróbálásával köti össze.

Vajon miféle próbatételek várnak Kovács úrra? Le kell-e mondania valamiről, és ha igen, miről? Mit nyer, ha úgy dönt, hogy vált? Ezt próbáltam kideríteni, és ennek érdekében arra kértem Koppány Imrét, hogy két operációs rendszert is telepítsen a gépre: Windowst és Linuxot.

A Windows használatáról nincs értelme sokat értekezni, hiszen illesztőprogramok tekintetében – legalábbis a gyártók részéről – a Microsoft operációs rendszereihez a legjobb az ellátottság, így természetesnek vehetjük, hogy minden összetevő megfelelően működik. A gyártók vásárlóikat különféle segédprogramokkal halmozzák el, melyekkel például figyelhető a processzor hőmérséklete

– természetesen ezek is Windows alatt érhetők el.
Ez esetben sincs ez másképp, a géphez két CD jár (az egyiken angol nyelvű illesztő- és segédprogramokkal). Az alaplapi lapkakészlet VGA-vezérlőjének beállításait kiegészítő programon keresztül is elérhetjük, ennek ikonja a telepítés után a Tálca jobb alsó sarkában foglal helyet (lásd *1. kép*: SiS 630/730 tulajdonságai).
A másik CD több érdekességgel szolgál: magyar feliratokkal van ellátva és az előző korong teljes tartalma helyet kapott rajta (az Acrobat Reader 4.0 és a gép kézikönyve PDF formátumban). Ugyancsak ezen található a MonSafe nevű program bemutató változata, melynek segítségével dokumentumainkat egy olyan titkosított könyvtárba helyezhetjük el, ahova csak a megfelelő kulcs és jelszó birtokában léphet be bárki. Ha a gépet ellopják, legalább bizalmas jellegű adatainkat biztonságban tudhatjuk – hordozható gépen ez különösen hasznos apróságnak tűnik. A másik nagyágyú a lemezen: a StarOffice 5.2-es változatának windowsos és linuxos kiadása. Mivel a StarOffice az utóbbi időben ingyenesen érhető el, a gépre Windowst és StarOffice-t telepítve részben, operációs rendszerként Linuxot választva pedig mindenestül ingyenes, mégis teljes értékű irodai munkaeszközhöz jutunk.
A Windows képes azokat az energiamegtakarítási lehetőségeket kezelni, amelyeket az újabb gépek kínálnak (a képernyő és a merevlemez kikapcsolása), a Tálcán folyamatosan figyelemmel követhetjük az akkumulátor töltési szintjét, és megadhatjuk, hogy egyes feltételek (például az akkumulátor lemerülése) esetén milyen jellegű riasztás jelenjen meg, majd a gép milyen műveletet hajtson végre (például adjon hangjelzést és kapcsoljon ki – lásd *2. kép*: Energiagazdálkodás tulajdonságai).
Ha azonban Kovács úr Linuxot szeretne indítani és Windows is került a gépére, először meg kell küzdenie az indítómenüvel. Nem okozhat nagy gondot, hiszen amikor asztali gépén Windows 95-tel dolgozott, az is felkínálta a régi operációs rendszer indítását. A választást követően a gép azonnal nagy felbontásba vált, és így jeleníti meg az indítás menetével kapcsolatos tájékoztató szövegeket – Kovács úr ugyan nem fog sokat érteni belőlük, de legalább megjegyzi, hogy ha piros feliratot lát, szólnia kell ismerősének, mert alighanem valamilyen hiba történt.
Hősünknek még három sort szükséges megtanulnia: saját bejelentkezési nevét (ez a polgári neve), jelszavát és a `startx`-et. Kovács úr nem teljesen érti, hogy miért nem kap azonnal grafikus felületet, de elviseli a rövidke parancs begépelésének nyűgét (az xdm, a gdm vagy akár a kdm használatával ez is megoldódik – a szerk.), a Windows 3.1 korából úgysem teljesen ismeretlen a számára.
A KDE2 ablakkezelőt futtatva könnyedén megtalálhatjuk a *Vezérlőközpont*-ot, ahol a Windows által felkínált energiatakarékos lehetőségek némiképp más elrendezésben, de hiánytalanul megtalálhatók. Éppúgy megadhatjuk, hogy mi történjen, ha lemerül az akkumulátor, mint ahogy azt is, hogy mikor, milyen figyelmeztetést szeretnénk kapni – igazán nincs ok panaszra, a KDE fejlesztői gondoltak a hordozható gépek felhasználóira is (lásd *3. kép:* KDE vezérlőközpont).

Kovács úr hozzáértő ismerőse lelkesen magyarázta és mutogatta a Linux képességeit, de őt mindez nem nagyon érdekelte – udvariasan hümmögött, majd megkérdezte, hogy megszokott dolgait a jövőben merre találja. Szeretne ugyanis levelezni, néha internetezni is akar, legfőképpen azonban szövegszerkesztőre és táblázatkezelőre van szüksége. Ismerőse ekkor rámutatott egy StarOffice-ikonra – itt mindent egy helyen megtalál. A programok sajnos angol nyelvűek, ami Kovács urat érzékenyen érintette, megnyugtatta viszont, hogy az ablakkezelő rendszer magyarul szólt hozzá, valamint ismerőse szerint már gőzerővel – és állami támogatással – zajlik a programcsomag honosítása. Megmutatva, hogyan kezelheti ezután barátai, üzletfelei adatait, címlistáját, hogyan levelezhet és böngészhet az Interneten, hősünket nem különösebben vágta mellbe a változás, hiszen az új programok nagyon hasonlítottak azokra, amelyeket addig is használt.


3.

Kovács úr azóta otthonosan használja gépét. Eleinte egy kicsit küzdenie kellett az új felülettel, hiszen a gombok és ikonok a szokotthoz képest máshogy festettek, valamint a menük és a parancsok neve is gyakran eltért. Kovács urat azonban immár a megismerés öröme hajtja, és nem hajlandó új rendszeréről lemondani. Lehet, hogy hamarosan elkéri azt a „rút" vagy milyen jelszót is ismerősétől, mert látott valami érdekes programot az Interneten, amit telepíteni szeretne és szívesen kipróbálná.
A legfontosabb kérdés továbbra is: miről mondott le Kovács úr? Lemondott megszokott ablakairól, programjairól – kapott azonban helyettük újakat. A gép alkatrészei ugyanúgy működnek, mint eddig, ám megtakarított némi pénzt – hiszen mindezt *ingyen* kapta.

A gép adatai
Portocom FreeStar;
Processzor: Intel Pentium III 650 MHz;
Memória: 128 MB RAM;
Merevlemez: 10 GB;
CD-ROM-meghajtó: 24-szeres;
beépített hajlékonylemez-meghajtót, hangkártyát, modemet és hálózati kártyát tartalmaz.

Mellékelt programok CD-n
Illesztőprogramok Windows 98/ME/NT/2000 alá,
Monsafe-próbaváltozat,
StarOffice 5.2 Windows- és Linux-változat,
segédprogramok a merevlemez előkészítéséhez és a BIOS frissítéséhez.

Előnyök
- Szép külső
- Kényelmes billentyűzet

Hátrányok
- A tapipad sem Windows, sem Linux alatt nem állítható
- A gép néha nem indul el (bekapcsolás után nem csinál semmit)

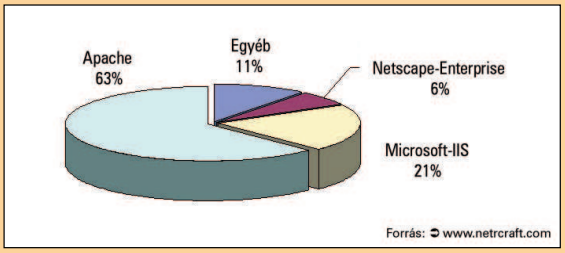
Linux-index

1. A gépek elérhetősége százalékban a ➲ Chek.com szerint: 99,928
2. A gépek elérhetősége százalékban egyes Microsoft-hirdetések szerint: 99,999
3. Ennyi milliárd egyedi C vagy C++ kódsort kell átültetni az Itanium 64-bites rendszerre: 100
4. A ➲ Kuro5hin.org olvasóinak ennyi százaléka néz naponta egy óránál kevesebbet (vagy egyáltalán nem néz) tévét: 65
5. Ennyiért adja a SuSE a középiskolás tanulóknak Linux-terjesztését: 0
6. A SuSE ennyi linuxos gépet támogat a középiskolákban az Egyesült Államokban: 2000
7. Ennyi milliárd dollárt fektettek be az újonnan induló cégekbe az elmúlt két év alatt: 160
8. Találatok száma naponta az ➲ apache.org weblapon 2 millió
9. Az ➲ apache.org sávszélesség-igénye csúcsidőben: 15 Mb/mp
10. Azon webkiszolgálók száma, amelyek a Netcraft felmérése szerint Apache-ot futtatnak (millió): 17 238
11. Jabber-kiszolgálók száma: 35 000
12. Vezeték nélküli készüléket használó vásárlók száma 2004-re (millió): 373
13. Hetvenből ennyi amerikai használja a vezeték nélküli Webet: 1
14. Háromból ennyi amerikai használja majd a vezeték nélküli Webet 2005-ben: 1
15. A Cingular húszmillió felhasználójának ennyi százaléka használja a Webet mobiltelefonról: 50
16. A vezeték nélküli Weben terjesztett hirdetésekre fordított összeg 2005-ben (milliárd dollár): 0,89-6,1

Forrás:
1.: Levél a ➲ Chek.com-tól
2.: Microsoft-hirdetés
3.: Aberdeen Group ➲ http://www.migratec.com/
4.: ➲ Kuro5hin.org
5–6.: SuSE
7.: Red Herring
8.: *Brian Behlendorf*-nak az áprilisi Apache Software Foundation Meetingen elhangzott beszéde alapján
9-10.: Netcraft ➲ http://www.netcraft.com/
11.: ➲ Jabber.org
12-16.: *Graeme Thickens* tudósította az Industry Standard Conference on Wireless-en elhangzottakról. *David Weinberger* hozzátette: „A találkozón két–háromszázmillió ember vett részt."

Az Apache tovább hódít

Lehet, hogy a régóta várt Apache 2 miatt alakult így a helyzet, de az is előfordulhat, hogy a gazdasági helyzet az oka. Akárhogy is nézzük, az Apache-ok – nyílt forráskódú webkiszolgálók – szolgáltatják a webes tartalom túlnyomó részét.



A Netcraft 2001 áprilisi felmérése szerint közel 18 millió kiszolgálón fut az Apache, vagyis a felmérés alapjául szolgáló 28 669 939 kiszolgáló 62,55 százalékán – ez 2,3 százalékos növekedést jelent. A Microsoft IIS kiszolgálók részesedése is emelkedett 0,89 százalékkal, így elérte a 20,64 százalékot. Még a Sun, illetve a Netscape iPlanet részesedése is nőtt, igaz csak 0,03 százalékkal, így ez megütötte a 6,27 százalékot. A többi kiszolgáló részesedése viszont csökkent.
Íme az Apache2beta néhány új tulajdonsága:

- hibrid (többfeladatos-többszálas) üzemmódban fut,
- új Apache Portable Runtime és többfeladat-kezelő modulok,
- szűrt kimeneti, illetve bemeneti modulok,
- Ipv6-támogatás.

Az Apache Software Foundation megtalálható a ➲ http://www.apache.org/ címen.
A Netcraft azt is közzétette, hogy a Compaq és az AltaVista követte az Amazont és linuxos kiszolgálókra váltott. Ezt megelőzően mindkét cég True64 (a korábbi Digital UNIX, amit a Compaq a Digital Equipment Corp. felvásárlásával szerzett meg) operációs rendszert használt. A Compaq 2001. januárban váltott Windowsra, majd később Linuxra.
A Netcraft a ➲ http://www.netcraft.com/ címen érhető el.

Doc Searls

Most már mindenki tudja, hol laksz

Elgondolkoztál már azon, vajon min törik a fejüket a srácok a Google-nál amellett, hogy azt kutatják, miként tud több mint 17 ezer linuxos kiszolgáló még gyorsabban még több mindent megtalálni? Próbáld csak megtalálni amerikai ismerősöd címét. A dolog egyszerű, talán túlságosan is egyszerű…
Próbálkozz a következő helyett barátod nevével: *John Doe* KY (azaz keresztnév, vezetéknév, az állam kétbetűs rövidítése).
Ha benne vagy valamelyik, a rendszer által feldolgozott telefonkönyvben, még az is előfordulhat, hogy megkapod az ismerősöd házának környékéről készült térképet a Yahoo! mapról.
A leiratkozáshoz:
➲ http://www.google.com/help/pbremoval.html
További leírás:
➲ http://www.google.com/help/features.html#wp

Doc Searls

A NASA JPL-je háborús szimulációt készít Linuxra

A pasadenai Jet Propulsion Laboratory (JPL) (Kalifornia) az űrprogram meghatározó szereplője. A JPL a California Institute of Technology fennhatósága alá tartozik és vezető szerepet tölt be az Egyesült Államokban a robotok által végzett űrkutatás területén. Önműködő űrjárművei a Plútó kivételével már a Naprendszer összes bolygóját kutatták. A NASA megbízásai mellett a JPL más állami hivataloknak is dolgozik. Az egyik ilyen megbízatásuk a Corps Battle Simulation (CBS) nevű program fejlesztése, amit nemrég ültettek át VAX-ról RedHat 7.0-ra – ennek következtében lényegesen nőtt a teljesítménye és csökkentek a kiadások.

A CBS-t katonatisztek harcászati kiképzéséhez használják már tizenöt éve. Korábban a program a legnagyobb teljesítményű, több mint százezer dollár értékű VAX 7800-as sorozat gépein futott, de a program növekvő bonyolultságának és egyre bővülő képességeinek „köszönhetően" a VAX-on korlátokba ütközött. Ez nehézkessé tette a további kutatásokat, és néhány éven belül elavulással fenyegette a harcászati szimulációs programot. Ezt látva az Egyesült Államok Hadseregének szimulációkkal és kiképzéssel foglalkozó parancsnoksága (a STRICOM) Orlandóban (Florida) felkérte a JPL-t, hogy a programot ültessék át Linuxra, ilyen módon javítva tovább annak képességeit a költségek mérséklése mellett.

Egy évet igényelt a CBS forráskódjának beállítása, lefordítása Linux alatt, valamint a próbafutások és a felbukkanó hibák kijavítása. A programozócsapat ezt követően lemérte a program teljesítményét Linuxon, és büszkén szemlélte az eredményeket. „Azáltal, hogy a CBS-t VAX-ról Linuxra ültettünk át, sokkal jobb eredményeket értünk el lényegesen kisebb anyagi ráfordítással, és számtalan új lehetőség nyílt meg előttünk" – mondta *Jay Braun*, a JPL egyik szimulációs programozást kutató mérnöke.

A Linux többlettulajdonságai megteremtik a program továbbfejlesztésének lehetőségét. A terep domborzatát például sokkal részletesebben lehet utánozni, mint korábban, amikor is az összetett látószöget figyelembe vevő számítások a végsőkig kihasználták a VAX lehetőségeit. Most nagyfelbontású térbeli térképeket használnak Linux alatt, így a szimulációk valóságosabbak, a csatahelyzetek pontosabbak.

A CBS egy 4000 dolláros PC-n fut, ami 1,2 GHz-es AMD Athlon processzort tartalmaz. Ezen a linuxos gépen a legnagyobb CBS-hadgyakorlat csaknem négyszer olyan sebesen fut, mint a leggyorsabb VAX-on, és ez mégsem megy a terephűség rovására! A VAX használata során ugyanis csökkenteni kellett a terep részletességét, hogy a szimuláció valós időben futhasson: azaz egy perc a programban is valóban egy percnek feleljen meg. Linux alatt viszont egy az egyhez időléptékkel gyakorlatozhatnak a programmal, méghozzá az elérhető legjobb terepfelbontás mellett.

A JPL-nek az is sikerült, hogy az állások mentése a legnagyobb pályák esetében húsz másodpercet igényeljen, kisebb terepeknél pedig hármat. Ez jócskán gyorsabbnak tekinthető, mint a régi VAX-rendszer. Linux alatt az alkalmazásnak minden szimulációhoz közel 3 GB-os címterület áll a rendelkezésére. „Ez aztán nagy kép!" – újságolta Braun. „Modellünk számos olyan tulajdonsággal bír, amely a Linux lehetőségeinek határait feszegeti."

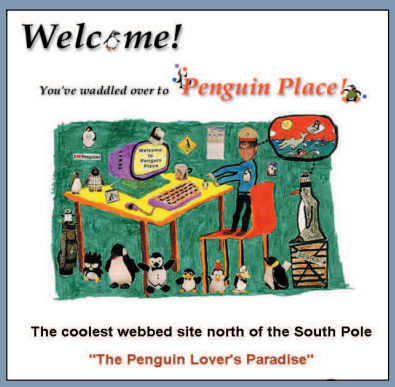
A JPL 2001 júniusában szállította az átültetett programot. Braun véleménye szerint a rendszert a közeljövőben úgy módosítják, hogy az a kétprocesszoros gépeket is kihasználhassa, így egyszerre több szimuláció futtatására nyílna mód. A JPL-t jelenleg 7.1-es RedHatre és 2.4-es rendszermagra ültetik át.

Drew Robb

Pingvinek a csapatmunkáról

A Penguin Post ➲ http://www.penguin-place.com/ oldalain találtunk egy érdekes beszámolót, mely szerint a pingvinek igencsak szeretik a sportot. Az antarktiszi Woods Hole Óceanográfiai Intézet környékén kószáló pingvinek különös érdeklődést mutatnak a sima hó- és jégmezőkön rendezett focimeccsek iránt.

Egy nap a tudósok ámulva vették észre, hogy a pingvinek átvették



a pályát. Nagyjából két csoportban álltak, majd elkezdtek ugrálva és totyogva lökdösődni, amíg el nem dőltek. Kis idő múlva felálltak és folytatták a játékot. Az valahogy elkerülhette a figyelmüket, hogy labda is kell a meccshez, de ez az apró hiányosság látszólag egyáltalán nem zavarta a frakkos sportembereket.

Az egész látvány annyira hihetetlen volt, hogy az egész személyzet a jégen fetrengett a nevetéstől.

Linux a közoktatásban (2. rész) – A tudatlanság útvesztői

A Microsoft hadjáratot folytat a szabad programok ellen, teszi ezt annak ellenére, hogy több szakember szerint a köztiszteletben álló világcég maga is jó néhány szabadon használható FreeBSD kódrészletet épített be a programjaiba.

A Linux az iskolákban

Sok iskolában a diákok csaknem kizárólag internet-elérésre használják a számítógépeket, és a jelenlegi helyzetet vizsgálva felmerülhet bennünk a kérdés, hogy mi értelme fizetős operációs rendszerre költeni a pénzt, amikor a Linux ezen a területen százszázalékos választást nyújt. Mivel a diák csak a Netscape böngészőt használja, fel sem merül benne a kérdés, hogy milyen operációs rendszer van a gépen. Ha az internetezéshez szabad programot használunk, akkor a felszabaduló pénzen újabb gépeket vásárolhatunk. Azokban az iskolákban, ahol képszerkesztést tanulnak a diákok, jól hasznosítható a Gimp, ami teljesítményében, kinézetében és logikájában igen hasonló az ismert Adobe Photoshophoz. Az előbbi szabad (és ingyenes) program, az utóbbi ára több százezer forint. A Gimp ismeretében a diák későbbi munkahelyén bármilyen más képszerkesztő használatára könnyedén áttérhet, hiszen azok kezelése hasonló lesz.

A Linux iskolai felhasználhatóságát vizsgálva a szövegszerkesztők terén nem találunk ennyire egyértelmű helyzetet, hiszen egyetlen általam ismert linuxos irodai csomagban sincs magyar nyelvű helyesírás-ellenőrzés, még nem magyarították a feliratokat, az üzeneteket és a Súgót. Némelyikben nincs olyan betűkészlet – és nem is telepíthető, amelyik meg tudná jeleníteni a magyar ő és ű betűket, helyettük csak kalapos õ és ũ gépelhető! Mindaddig, amíg a linuxos szövegszerkesztők nem képesek ezeknek az alapszolgáltatásoknak az ellátására, addig a Microsoft Office cseréje nem javasolható az iskolákban. Ezen az a tény sem változtat, hogy a SuSE pénzért elérhetővé teszi a magyar helyesírás-ellenőrzést a StarOffice 5.2-ben.

Hazánkban egyelőre a Microsoft termékei az egyeduralkodók, amit az is mutat, hogy azok az iskolák, amelyek Linuxra vagy Macintoshra állnának át, a továbbiakban nem tudnának oktatóprogramokat vásárolni, hiszen azokon rendszerint a következő felirat szerepel:

„A program Windows 95 alatt működik. A Windows elindítása után helyezze a lemezt a CD-ROM-olvasóba!"

A honi programgyártókat és CD-kiadókat a piac nem kényszeríti arra, hogy más operációs rendszereken is futtatható kiadványokat hozzanak létre. Ezek a kiváló szerkesztők talán soha nem hallottak más operációs rendszerekről, hiszen azokat nem említették nekik az iskolákban és a továbbképzéseken. Mivel nem ismerik ezeket a lehetőségeket, nem írnak rá oktatóprogramokat. A kör itt bezárul. Nincs, aki oktassa azokat, akiknek oktatniuk kellene.

Pedig a Linuxot is tanulni kell! A tanárnak és a diáknak egyaránt el kell sajátítani a kezelését. Mindig találhatunk olyan Linuxra írt programot, ami elvégzi a kívánt feladatot, de nem feltétlenül a Windowsban megszokott módon. Kell egy kis idő, amíg a Unix-logikához hozzászokik az ember, és ez a „kis idő" akár egy egész évnyi is lehet. De ha windowsos múltunkra visszagondolunk, emlékezhetünk rá, hogy azt a rendszert sem egy hét alatt ismertük ki. Vajon mire való az iskola, ha nem arra, hogy új dolgokat tanuljanak benne?

Amiben egyértelmű hátrányt szenvednének a Linuxra kapatott ifjak: a játék. Bár mostanra több ismert játékot átírtak Linuxra, még nem elég elterjedtek és magyarítva sincsenek. Azok az iskolák tehát, amelyek Linuxot telepítenek a gépeikre, meggátolják, hogy ifjaik olyan lélekemelő játékokkal játszhassanak, mint a Holtak Háza vagy a Quake3, amelyekben egységnyi idő alatt minél több hullát kell gyártani. Örömteli vagy éppenséggel szomorú hír, hogy az utóbbi játékot időközben átírták Linuxra.

Sérülnének még az iskolai rendszergazdák érdekei is, akiknek immár kettő vagy több operációs rendszert kellene karbantartaniuk. S mint tudjuk, az iskolai rendszergazdák is rosszul fizetettek és túlterheltek, emiatt gyakran nincsenek a helyzet magaslatán.

Az iskoláknak számítógép-vásárlásaik megtervezésekor meg kellene fontolniuk, hogy milyen célra vásárolják a gépeket; és át kellene gondolniuk, hogy például internetezésre megfelelő lenne-e számukra a szabadon és ingyenesen használható Linux-rendszer telepítése. Ha valaki iskolaidőben tanulás helyett játszani akar, annak célszerű a klasszikus Windowst telepítenie.

A szövegszerkesztéshez ma még szintén célszerűbb jogtiszta programokkal a Windowst használni, hiszen a Linux-fejlesztőknek még szükségük lehet néhány évre ahhoz, hogy minden tekintetben jól használható és magyarított irodai alkalmazásokat hozzanak létre. Véleményem szerint ma olyan iskolai számítógépparkra lenne szükség, amelyben vegyesen vannak a gépek, és helyenként egy-egy Macintosh is felbukkan.

Mi gátolja a Linux terjedését?

A szaktudás hiánya és az anyagi ellenérdekeltség. Elképzelek egy újabb Sulinet-programot, amelyben nem Windowst és Internet Explorert, hanem Linuxot és Netscape böngészőt szállítanának az iskoláknak, hogy azok rákapcsolódhassanak a Világhálóra. Az előbbi, windowsos megoldás tetemes összeget emésztene fel az adófizetők pénzéből, az utóbbi teljesen szabad és ingyenes lehetőség lenne, ugyanakkor mindkettő képes lenne ugyanazt a feladatot ellátni. Talán nehéz lenne olyan cégeket találni, amelyek meg tudnának birkózni a Linux telepítésével, hiszen amikor tavaly két linuxos gépet akartam beszerezni, az ismert szakboltok sorra elutasították kérésemet, mivel nem volt megfelelő szakemberük. Ezért csodálkoztam, amikor a közelmúltban hirtelen kiderült, hogy több világcég is képes márkás gépeit Linux operációs rendszerrel szállítani – nagy tételű állami megrendelés esetén. Vannak tehát, akik kellő anyagi ellenszolgáltatás esetén hajlanak a Linux felé, de többnyire mégis az érdektelenség vagy éppenséggel az ellenérdekeltség a jellemző a szakemberekre.

Az elmúlt évtizedben sokan szedtek magukra jelentős számítástechnikai szaktudást, és most ez a tudás került veszélybe a választható operációs rendszerek megjelenésével, valamint végső soron a számítástechnikai világ

változása következtében. A vállalkozásoknak át kellene képeztetni dolgozóikat, akiknek ki kellene dobni addigi, szélsőségesen a Windowsra kihegyezett szaktudásukat, és újat kellene tanulniuk. A tudás pénzbe kerül, amit most az átképzésre kellene fordítani. Utóvédharcok folynak, amelyekben nem mindenki használ tiszta eszközöket. Az átlagember is viszonylag jól eligazodik ebben az egysíkú világban, de neki is van vesztenivalója. Új ismereteket kellene szereznie, pedig még windowsos tudása sem tekinthető igazán megbízhatónak. Az új dolgok tanítására hivatott, alulfizetett pedagógusokban pedig felmerül a kérdés, hogy miért tanuljon és tanítson ő Linuxot, hiszen van neki éppen elég baja. Pedig ha körülnézünk a világban, akkor azt látjuk, hogy a számítástechnikát meghatározó országok felsőoktatásában a programozók tanítása valamelyik Unix-rendszeren alapul. Nem mintha ellenszenves lenne nekik a többi operációs rendszer, de programozókat és nem felhasználókat akarnak nevelni. Ha valamelyik hazai egyetemen vagy főiskolán Windowsra alapozzák az oktatást, az csak azt bizonyítja, hogy Magyarország fogyasztó és nem gyártó ország. Az a diák, aki Windows API-hoz van szoktatva, soha sem válik teljes értékű programozóvá, hiszen végső soron maga is csupán felhasználó marad. Tudása szükségképpen Microsoft függvényhívásokra és változók megadására korlátozódik. A Windows feketedoboz-módszere nem teszi lehetővé a számára, hogy megismerje, valójában miként működnek a dolgok a mélyebb rétegekben, hiszen azokat a Windows API-k eltakarják. Takaros felhasználói felületeket fog létrehozni, de képtelen lesz önálló programozói feladatok megoldására.

A Unix-rendszerek forráskódjai ezzel szemben szabadon tanulmányozhatók, említhetem a Linux, a FreeBSD, a Solaris vagy akár a Java-változatokat. A Java jelentősen eltér a többitől, számos dologban azonban követi a unixos hagyományokat. E rendszerek forráskódjának és programozói stílusainak tanulmányozása közben széles körű ismereteket szerezhet a programozó, és a legtöbb kérdésére választ vagy mintát kaphat. Ezeket a mintákat a GNU-programok esetében nemcsak szabadon tanulmányozhatja, de be is építheti saját programjaiba, így jelentős programozási időt takaríthat meg. Természetesen felmerülhet a kérdés, hogy programozók kellenek-e az országnak vagy a Microsoft API-t jól ismerő, egyetemet végzett technikusok?

A Linux a tapasztalatok szerint nehezen értékesíthető kereskedelmi forgalomban, ezért a kereskedő cégek is ellenérdekeltek. Ennek ellenére Budapesten két könyvkereskedés is szakosodott Linux-programok és szakirodalom forgalmazására. A vasat értékesítő boltokban ugyancsak felfedezték a Windows nélküli számítógépeket, hiszen mostanában már operációs rendszerek nélkül adják a PC-ket, amikre a későbbiekben vagy nem kerül operációs rendszer, vagy Windows-kalózmásolatokat telepítenek rájuk, esetleg Linuxot vagy valamilyen más szabad programot. Az olvasóra bízom, hogy eldöntse, mi történik majd a gépek megvásárlása után. Általános tapasztalat azonban, hogy bármilyen nem windowsos lehetőséggel szemben erős az ellenállás, hiszen a cégek abban érdekeltek, hogy mindenkinek a Windowsra írt, drága programokat árusítsák, köztük az iskoláknak is, hiszen hasznuk csak azon van. Nő a félelem a szabad programok miatt. A Microsoftnál dolgozó *Jim Allchin*, a Windows operációs rendszert fejlesztő részleg főnöke szerint a szabadon terjesztett program – mint például a vetélytárs Linux – megfojthatja az újítási szándékot és a törvényhozóknak szükségszerűen fel kell ismerniük ezt a veszélyt. „A nyílt forráskód elpusztítja a szellemi tulajdont, nem tudok ennél rosszabbat elképzelni a programgyártásra és a szellemi tulajdonra nézve” – mondta Allchin, majd folytatta: „Amerikai vagyok, hiszek az amerikai útban. Aggódom, hogy a kormányzat bátorítja a nyílt forráskódot, és szerintem nem tettünk eleget, hogy a politikacsinálók felfogják a benne rejlő veszélyt.”

Ha körülnézünk a világban, akkor azt látjuk, hogy a számítástechnikát meghatározó országok felsőoktatásában a programozók tanítása valamelyik Unix-rendszeren alapul.

A Microsoft hadjáratot folytat a szabad programok ellen, teszi ezt annak ellenére, hogy több szakember szerint a köztiszteletben álló világcég maga is jó néhány szabadon használható FreeBSD kódrészletet épített be programjaiba. Ez természetesen nem törvénytelen, hiszen a szabad program mindenki számára szabadon felhasználható. A Microsoft mégis cáfol, hiszen a fenti állítás bizonyíthatatlan, mivel senki sem fejtheti vissza a cég által írt forráskódot, ugyanis azt a törvény tiltja!

Allchin arról sem beszélt, hogy a kereskedelmi programok ára túlértékelt. A felhasználó nem tudja, de nem is akarja megfizetni azt a tetemes összeget, amit elkérnek értük, hiszen számára nem hajtanak semmilyen hasznot. Legtöbbünk telepíti a méregdrága programot, fél órát játszadozik vele, majd elfelejti azt is, hogy mikor és miért került fel a gépére. A Linux ilyen szempontból is eszményi, hiszen ezzel az operációs rendszerrel együtt gigabájtnyi olyan szabad programot telepíthetünk gépünkre, amit sohasem fogunk semmire sem használni. Amennyiben a piacgazdaság azt jelenti, hogy a fogyasztókat kifinomult módszerekkel fölösleges presztízsfogyasztásra vegyük rá, akkor a fenti aggodalmak minden bizonnyal jogosak. A vásárlók kétségtelenül védekeznek, és csak vonakodva hajlandók (pénzért) frissíteni régi, de az adott feladatra kitűnően megfelelő programjaikat. Egyes szakemberek is szkeptikusak. Egy számítástechnikai oktató magánvéleménye szerint: „A programok a 4.0-s változatig használhatóak, utána már nem biztos, hogy érdemes velük foglalkozni!”

Az operációs rendszerek háborúja mostanra nemcsak szakmai és gazdasági, hanem politikai, nevelési és törvényhozási gonddá is vált. Több kérdést kell az

elkövetkező évtizedben megválaszolnia az országnak. Végig kell gondolnunk, hogy megáll-e a világ vagy elhalad mellettünk, miközben mi becsukjuk a szemünket? Hihetünk-e rendületlenül a Microsoft-programok mindenhatóságában és örökkévalóságában? Felépíthetjük-e a nemzetgazdaságot egyetlen pillérre, ami rajtunk kívül álló okok miatt bármikor kidőlhet alólunk? Eljöhet-e Magyarországon az idő, amikor törvény fogja tiltani az otthoni programírást, amennyiben a fejlesztő azt ingyenesen mások rendelkezésére akarja bocsátani? Betiltható-e a szabad program?

A Linux a világban

Míg Európában a SuSE, Amerikában pedig a RedHat (Vörös Kalap) Linux a menő, addig Kínában a Red Flag (Vörös Zászló) Linuxot támogatják. Ezt a Vörös Zászlót olyankor szokták lobogtatni, amikor a Microsoft és a többi nagy programgyártó cég feszegetni kezdi, hogy rengeteg a kalózprogram arrafelé. A hírekben az áll, hogy külön hivatal foglalkozik a Red Flag Linux fejlesztésével, és mivel szabad program, valamint a forráskód átírható, kitűnően alakítható a helyi viszonyokhoz és a sajátos kínai íráshoz.

Kapcsolódó címek

- http://server.wesselenyi-bp.sulinet.hu/sulinux/
- http://lme.linux.hu/forditas/index.html
- http://sunserv.kfki.hu/pipermail/tanforum/
- http://www.lme.hu

Eljöhet-e Magyarországon az idő, amikor törvény fogja tiltani az otthoni programírást, amennyiben a fejlesztő azt ingyenesen mások rendelkezésére akarja bocsátani? Betiltható-e a szabad program?

A Microsoft Windows ezzel szemben érinthetetlen, kívülállók semmilyen formában sem változtathatják meg, még akkor sem, ha hibát javítanának benne vagy tökéletesítenék. Ezért börtön jár. A Red Flag Linuxot a kínaiak olyanra alakítják, amilyenre akarják.

Valószínűleg a szabad használat és a kedvező ár, azaz a program ingyenessége az oka annak, hogy egyes szakértők már Linux-földrészként kezdik emlegetni Afrikát, bár a program még csak most kezd terjedni.

A mintaország azonban Mexikó, amely nemcsak vezető Linux-fogyasztóvá, hanem vezető Linux-fejlesztővé is vált, hiszen a mexikói *Miguel de Icaza* guru volt a Gnome munkaasztal megteremtője és máig ő a vezető tervezője. S ezzel Mexikó legalább egy területen élre került a programfejlesztésben. Nem tudok róla, hogy bármely magyar céget hasonló súllyal jegyeznék a programok világában. Ahogy *Gary Chapman*, a 21st Century Project igazgatója rámutatott, a nyílt forráskódú program segítheti a fejlődő országokat abban, hogy elkerüljék azt a sorsot, ami „Microsoft-ügyfélállamokká" tenné őket. (A 21. Század Projekt nem nyereségközpontú szervezet a texasi Austinban, amely a harmadik világ tudományos és szakmai jövőjét vizsgálja.) A Sulinethez hasonló program keretében Mexikóban az iskolák Linux operációs rendszerrel ellátott gépeket kaptak, hogy azok rákapcsolódhassanak az Internetre. Hazai szakértők becslése szerint a magyar Sulinet program árát 15–20 százalékkal lehetett volna csökkenteni, ha annak idején linuxos gépeket kaptak volna az iskolák. Mexikónak más tervei is vannak a Linuxszal. Mexikóváros kormányzata bejelentette, hogy a város hivatalaiban lévő gépek operációs rendszerét fokozatosan Linuxra cserélik. Becslésük szerint az átmenet körülbelül két évet fog igénybe venni, mert nem akarnak zökkenőket a mindennapi ügymenetben. A „fölöslegesen programokra költött" dollármilliókat a város szegényeinek megsegítésére szánják.

Mielőtt valaki arról kezdene beszélni, hogy a Linux hazánkat a harmadik világ szintjére süllyesztené, hadd említsek francia példát is. A párizsi Linux Expón *Jean-Pierre Archambault*, a Centre National de Documentation Pedagogique (CNDP) hivatalnoka nem annyira az anyagiak fontosságát, hanem inkább a választás szabadságát hangsúlyozta. Szerinte a CNDP sohasem próbálta megdönteni a Microsoftot vagy bármilyen más egyeduralmat. De végiggondolta, hogy egy monopólium megléte önmagában is nevelési kérdés. A CNDP a monopóliumokkal szemben a kulturális sokszínűséget és a választás szabadságát próbálja előmozdítani. Ez az oka annak, hogy a CNDP erőfeszítései főként a következőkre irányultak: tájékoztassák a létező választási lehetőségekről a felhasználókat, és bemutassák azokat. Véleménye szerint elsőként az oktatási kormányzatot kell meggyőzni arról, hogy a Linux és a nyílt programok értékesek és figyelemre méltóak. Példaként hozza fel a grenoble-i körzetet, ahol 350 általános és középiskola használja a linuxos SLIS programot az Internetre való kijutáshoz és az iskolai elektronikus levelek kézbesítéséhez. Létrehoztak egy saját Linux-változatot, amit DemoLinuxnak neveztek el, és pályázatokkal támogatják azokat a tanárokat, akik Linuxra írnak oktatóprogramokat.

Sőt, már Norvégiában is vannak olyan állami alkalmazottak, akik Linuxot vezetnének be az iskolákban és a közhivatalokban.

Mennyire vagyunk gazdagok?

A magyar tanulókat megfosztottuk a választás szabadságától, hiszen jelenleg egyetlen operációs rendszert kapnak az iskolákban, így állampolgári jogaikban korlátozzuk őket. Ha a törvénytelen programhasználat felett szemet hunyunk, tovább rontjuk az amúgy is gyengülő közerkölcsöt. Ha életszerűen gondolkodunk, akkor ma hazánkban Windows-nélküli iskola elképzelhetetlen, hiszen az oktatási intézményeknek tükrözniük kell a jelen Magyarországát, amelyikben még a Microsoft uralkodik. De hibát követ el az az oktatási rendszer, amely nem tükrözi egyúttal a jövőt. A jövőben pedig változhat a helyzet. Még azt sem állítom, hogy a Linux lesz a jövő operációs rendszere, hiszen a Linux nem csodagyógyszer, hanem választási lehetőség. A Windows-korszakot feltehetően nem a Linux-korszak fogja követni, hanem az olcsóbb és nyitottabb programok, a nemzetgazdaságban pedig többfajta operációs rendszer egyidejű jelenléte

lesz a jellemző. A Linux és a többi választható operációs rendszer nem fogja rövid időn belül leváltani a Windowst, de csökkenteni fogja a súlyát. A jelen tanulóit pedig fel kell készíteni arra, hogy mire megjelennek a munkaerőpiacon, mások lesznek az operációs rendszerek és a programok, mint ma. Ezt viszont nem lehet egy monolit számítástechnikai kultúrára alapozni. Én személy szerint olyan iskolát képzelek el, ahol az operációs rendszerek terén is pluralizmus van, és a tanulóknak választási lehetőségük van. A csőlátás és a tudatlanság türelmetlenséget és végül diktatúrát szül, a gazdaságban pedig lemaradást.
Ahogy az olvasók is észrevehették, ebben a cikkben a Linux hangsúlyozottabban jelent meg, mint a Macintosh, s ennek egy oka van. A Linux szabadon használható termék, PC-re telepíthető, míg a Mac igencsak költséges választás. Én szívem szerint Apple-gépeket is szeretnék látni az iskolákban, és ha valaki azt mondja, hogy a Linux a harmadik világ operációs rendszere lesz, mi pedig nem a harmadik világhoz akarunk felzárkózni, akkor csak annyit kérdeznék tőlük, hogyha annyira gazdagok vagyunk, akkor miért nincsenek Macintosh-gépek (is) az iskolákban? Vagy annyira azért mégsem vagyunk gazdagok?

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. Azóta hátat fordított a feketedoboz módszereknek, és most szabadidejében a nyílt forráskódú Java és a GNU C vagy C++ programokat tanulmányozza. Több írása megjelent már a hazai számítástechnikai lapokban. Ha néha feláll számítógépe mellől, rendszerint művészettörténész feleségével és kisiskolás lányával „találja szemben" magát.

Olyan iskolát képzelek el, ahol az operációs rendszerek terén is pluralizmus van, és a tanulóknak választási lehetőségük van.

Szőke ciklus

Telefonos támogatás

Szőke nő: „A vévéve rendes vével vagy dupla vével írandó?"
Telefonos ügyfélszolgálat: „dupla…"
Szőke nő: „…és utána vessző van vagy pont van?"
Telefonos ügyfélszolgálat: „pont…"
Szőke nő: „…és utána mit írjak?"

Szépségápolás

– Mi a számítógépes szépségápolás?
– ???
– Átküldök neked egy kiló bájt!

Nőnemű

– Miért nem lehet a bit nőnemű???
– ???
– Mert a bitnek nincs „Nem tudom…" állapota.

Végtelen ciklus

– Mi a programozónő rémálma?
– A végtelen ciklus…

Hasonlóságok

Hét pont, amelyben a Windows és egy átlagos nő megegyezik:

1. Szinte mindenkinek van vagy volt már.
2. Logikusnak tűnő szerkezete ellenére teljesen kiszámíthatatlan.
3. Teljesen azonos eseményekre képes a legkülönfélébb, legmeghökkentőbb válaszokat produkálni.
4. A legtöbb tetszetős és célszerűnek látszó terméket rájuk tervezik.
5. A használat során bebizonyosodik, hogy a divatos külső nem minden.
6. A használati idő előrehaladtával kapacitásunknak egyre nagyobb százalékát veszi igénybe.
7. Az újabb változatok mindig a régebbi hibáinak kijavítását ígérik, ehelyett mindig újabb és újabb, addig elképzelhetetlennek tartott hibaforrásokkal találkozunk.

Hét pont, amelyben egy belevaló férfi hasonlít a Linuxra:

1. Egyszerre akár több felhasználó igényeit is képes maradéktalanul kielégíteni, akár úgy is, hogy egyik sem tud a másik létezéséről.
2. Nyílt és egyszerű forráskódja ellenére saját képükre formálni igazán csak a hozzáértők képesek.
3. Türelemmel és megfelelő gyakorlattal minden feladatot meg lehet vele oldani.
4. Feltételezi, hogy az őt használó tudja, mit is akar vele kezdeni.
5. A rendelkezésre álló erőforrások lehető legnagyobb részét a felhasználó(k)nak adja.
6. Működésében bonyodalom csakis durva felhasználói behatás esetén következik be.
7. Habár gyakorta és sokféle újabb változat jelenik meg, a felhasználók a régit igyekeznek megtartani, legfeljebb csak egyes részeit kívánják frissíteni.

Akciós áru

Amikor mezei lapkákat használsz mezei merevlemezekkel, ezeket pedig gyors ethernettel kötöd össze, akkor mezei operációs rendszerre van szükséged: például ilyen a Linux. (John K. Thompson)

Az Új Falu

Mivel a Web nem kézzelfogható, a távolság sem játszik benne szerepet. Vegytiszta társadalmi övezet; csak belőlünk áll és mindabból, amit írtunk – azt pedig másoknak írtuk. A Web találkahely boldog-boldogtalan számára, melyet szabad akaratunkból hoztunk létre. (David Weinberger)

Halandzsapárosítás: ki mit csinál?

Ki tudod-e találni, hogy az alábbiakban felsorolt cégek mi mindent halandzsáznak össze magukról? Próbáld meg összepárosítani a bal oldali cégneveket a jobb oldalon található állításokkal – utóbbiakat mindig az adott cég sajtónyilatkozatából vagy céges szóróanyagából másoltunk át. Ha még ennél is jobban akarsz szórakozni, hasonló halandzsaszövegeket alkothatsz a ➲ http://www.BuzzPhraser.com/ oldalon. Amennyiben nincs ínyedre a mondatokat létrehozó motor, forrását alakítsd át igényeid szerint – a program nyílt és szabad.

Doc Searls

(A Linux Journal vezető szakembere, halandzsafelelős megoldásszállító)

1. RedHat	a) Internetes és vállalati rendszerek építéséhez szállít Linux-alapú szoftvermegoldásokat.
2. Caldera	b) Piacvezető cég – CyberSecurity (KiberBiztonság)-termékeket, -szolgáltatásokat, valamint -oktatást biztosít.
3. Linuxcare	c) Piacvezető cég az összeköttetésben álló intelligens (smart) eszközök szoftvereinek és szolgáltatásainak területén.
4. VALinux	d) A Linux és a nyílt forráskódú webes megoldások nagy tapasztalattal rendelkező szállítója.
5. APC	e) A Unix egyesítése a Linuxszal a jobb üzletért – Linux-alapú üzleti megoldások piacvezető műszaki fejlesztője és terjesztője.
6. Chek	f) Piacvezető a nyílt forráskódú alapokra épülő megoldások fejlesztése, üzembe helyezése és kezelése területén.
7. Aberdeen	g) A távközlésben, a hálózati tárolóeszközökben, a képalkotásban, az orvosi felszerelésekben és a félvezetőgyártásban használható beágyazott rendszerek és berendezéspróba-alkalmazások világvezető szállítója.
8. Mainsoft	h) A méretezhető üzenettovábbító és levélrendszer-szoftverek piacvezető fejlesztője internetszolgáltatók, alkalmazásszolgáltatók és cégek számára.
9. Bay Mountain	i) A modern alkalmazásfejlesztési eljárások és üzleti megoldások piacvezető szállítója.
10. IBM	j) A legszakszerűbb linuxos és nyílt forráskódú fejlesztéssel kapcsolatos megoldások piacvezető szállítója.
11. Trustix	k) Piacvezető alkalmazásszolgáltató.
12. Zero G	l) A web- és alkalmazásszolgáltatás piacvezető szállítója.
13. SecureInfo Corp.	m) Az átfogó, mindenkori elérhetőséget biztosító megoldások piacvezető szállítója.
14. Motorola Computer	n) A nyílt forráskódú csoportmunka és e-businessprogramok piacvezető szállítója.
15. Wind River	o) A szolgáltatók vezető távközlési rendszerprogram-szállítója.
16. Rockliffe	p) Vezető piacelemző és helyzetfelmérő szolgáltatásokat nyújtó cég.
17. Magic Software	q) Az e-átültető vállalat.
18. TurboLinux	r) Piacvezető linuxos hálózatfelügyelő megoldásokat szállító független szoftverforgalmazó.
19. Zelerate	s) A teljes körű IT-megoldások piacvezető szállítója.

Megoldások: 1-f, 2-e, 3-j, 4-d, 5-m, 6-o, 7-p, 8-q, 9-l, 10-s, 11-r, 12-k, 13-b, 14-g, 15-c, 16-h, 17-i, 18-a, 19-n

Douglas Adams, annyira szerettünk!

„Tudod, ilyenkor, fogságban egy betelgeuse-i emberrel, egy vogon űrhajó zsilipkamrájában, arra várva, hogy megfulladjak az űr hidegében, ilyenkor igazán sajnálom, hogy nem figyeltem arra, amit anyám mondott, amikor fiatal voltam!
– Miért, mit mondott?
– Nem tudom, nem figyeltem rá!”

„Imádom a határidőket. Szeretem hallani süvítő hangjukat, amint elszállnak mellettem.”
(Douglas Adams)

„A tudomány elvesztett egy barátot, az irodalom elvesztett egy lángelmét, a hegyi gorillák és a fekete rinocéroszok elvesztettek egy lovagias védelmezőt (egyszer rinocérosz-jelmezben felmászott a Kilimandzsáróra, hogy pénzt gyűjtsön a rinocéroszszarv-kereskedők elleni harcra), az Apple Computer elvesztette ékesszóló hitvédőjét. Én elvesztettem helyettesíthetetlen szellemi társamat és az általam ismert egyik legkedvesebb és legmókásabb embert. Tegnap hivatalosan is a tudomásomra jutott egy örömteli hír, aminek Ő nagyon örült volna. Hetek óta tudtam már, de nem volt szabad elárulnom senkinek. Most, hogy már megtehetném, túl késő.”

(Richard Dawkins Douglas Adamsről)

Új termékek

Firewall-in-a-Box

Az EMAC a közelmúltban mutatta be Firewall-in-a-Box (FIB) nevű termékét. A FIB ventilátor-nélküli, kisméretű tűzfal, amely több munkaállomás és kiszolgáló biztonságos csatlakozását teszi lehetővé.

A FIB testreszabott Linux-változaton alapul, és az adatfolyamot analóg modemen, kábelmodemen és digitális bérelt vonalon keresztül tudja szabályozni. Képes IP-címeket osztani az ügyfeleknek (DHCP) és gyorstárazó DNS-t biztosít. A FIB alacsony feszültségű NS Geode GXLV-200 processzort és 50 wattos tápegységet használ. A menüvezérelt beállítóeszköz terminálon vagy Telneten keresztül érhető el.

Adatok: EMAC, Inc., 2390 EMAC Way, Carbondale, Illinois 62901
telefon: 618-529-4525
e-mail: info@emacinc.com
➲ http://www.emacinc.com/

A Pockey

A Pockey Drives új hordozható tárolóeszköze a Pockey nevű külső USB merevlemez, amely kicsi, gyors és nem igényel külső áramforrást. A tenyérben elférő Pockey tíz- és húszgigás méretben készül, ráadásul minden hordozható és asztali géphez csatlakoztatható. A Pockey USB kábel szállítja az adatokat és a tápfeszültséget. A Pockey újraindítás nélkül cserélhető a számítógépek között, ezért a fájlmegosztás egyszerű. Az eszköz méretei 12,5x7,5x2,5 cm, az átviteli sebesség legfeljebb 1,5 MB/mp.

Adatok: Pockey Drives, 21356 Nordhoff Street, Suite 109, Chatsworth, California 91311
telefon: 818-717-9556
e-mail: info@pockeydrives.com
➲ http://www.pockeydrives.com/

VelociGenX

A VelociGen webalapú alkalmazásfejlesztő és -futtató környezete a VelociGenX, amely XML-t használ a hordozhatósághoz. A vállalatok ezáltal újrafelhasználható webalkalmazásokat készíthetnek, amelyek tetszőleges forrásból vehetik az adatokat. A VelociGenX képes metaalkalmazásként burkolni, csatolni és futtatni különféle XML-összetevőket – akár hagyományos, akár dinamikus adatforrásokból. A metaalkalmazások futtathatók és az eredmények elektronikus levélben böngészőre, PDA-ra, személyhívóra vagy mobiltelefonra küldhetők el.

Adatok: VelociGen, Inc., 8380 Miramar Mall, Suite #105, San Diego, California 92121
telefon: 858-622-1164
e-mail: info@velocigen.com
➲ http://www.velocigen.com/

Heroix eQ Management Suite

A Heroix Corporation új terméke, a Heroix eQ Management Suite több felületet kezelő felügyelőprogram,

Heroix eQ Management Suite

amely néhány tíztől kezdve több ezer rendszer felügyeletére alkalmas. Az eQ-képesség a rendszer és a hálózat hibáinak észlelését, a figyelmeztetést és a megoldást foglalja magában. A rendszerfelügyeletet célirányos szabályrendszer-motor hajtja, amely az emberi tudást és döntéshozást utánozza. Az eQ-csomag tartozéka az önműködő alkalmazásfelismerés, az Express Wizard-felület, a vészjavító és az eseményjelentő. Az eQ számos Linux-, Unix- és Windows-felületen fut.

Adatok: Heroix Corporation, 120 Wells Avenue, Newton, Massachusetts 02459
telefon: 1-800-229-6500
➲ http://www.heroix.com/

PostgreSQL 7.1

A PostgreSQL Global Development Group elkészítette a PostgreSQL 7.1-es változatát, amely webhelyükről és a tükörkiszolgálókról tölthető le. A legfontosabb újdonságok a következők: írás előtti napló (WAL), amely növeli az adatbiztonságot és a feldolgozás sebességét, mert csak egyetlen módosított naplófájlt szükséges a lemezre írni; tetszőleges hosszúságú sorok a túlméretezett tulajdonságtároló módszeren keresztül (TOAST); az SQL92 külső csatlakozásainak támogatása; 64-bites C függvények támogatása, valamint a bonyolult lekérdezések gyorsabb és továbbfejlesztett támogatása.

PostgreSQL

Adatok: PostgreSQL Global Development Group, PO Box 1648, Wolfville, Nova Scotia, Canada B0P 1X0
telefon: 877-542-0713
e-mail: info@pgsql.com
➲ http://www.postgresql.org/

Max Server Pages

A PlugSys bejelentette, hogy elkészült a Max Server Pages (MSP), amely adatbázis-központú kiszolgálóoldali parancsnyelvrendszer webkiszolgálókhoz. Az MSP Xbase parancsokat és függvényeket használ, így a fejlesztők könnyen elérhetik DBF fájlokban vagy SQL-adatbázisokban tárolt adataikat. Az SQL-lekérdezések és -frissítések a PlugSys ODBC-csatolóján keresztül zajlanak. Az eredmények HTML-lel és JavaScripttel keverhetők. Az MSP egységes egészként települ; futásidejű kipróbálás mellett lehetőség nyílik a dinamikus fordításra és gyorstárazásra. A szabadon letölthető változat (MSP/FE) a cég weblapján érhető el.

MSP Max Server Pages

Adatok: PlugSys International LLC, 1636 Graff Avenue, San Leandro, California 94577
telefon: 510-352-2228
➲ http://www.plugsys.com/

DupLinux az Arcótól

2001 májusában a tokiói LinuxWorld Expón mutatta be az Arco Computer Products cég a DupLinuxot. Ez segédprogramul szolgál IDE RAID vezérlőkhöz, háttérben zajló újraépítéssel és menet közbeni cserelehetőséggel. A terméket az Arco DupliDisk II IDE RAID 1 vezérlőjéhez tervezték, segítségével a felhasználók beállíthatják, felügyelhetik és újraépíthetik RAID tömbjüket – parancssorból vagy az X Window Systemen keresztül. A DupLinux a merevlemezek menet közbeni cseréjét is lehetővé teszi.

DupliDisk II IDE RAID Controllers

ARCO

Adatok: Arco Computer Products, Inc., 3100 North 29th Court, Second Floor, Hollywood, Florida 33020
telefon: 1-800-458-1666
e-mail: sales@arcoide.com
➲ http://www.arcoide.com/

A hónap szakmai tanácsai

Miként írhatom én az FTP üdvözlő szövegét?

Szeretném módosítani mind az FTP-kapcsolat létrehozásakor, mind a bejelentkezéskor megjelenő szöveget. Tudom, hogy az ftpaccessben vannak valahol, de sehogy sem találom a .message és /welcome.msg fájlokat, amelyekre a beállítási fájlok hivatkoznak. Lehet, hogy ezek a fájlok nem is léteznek és a megjelenő üzenetek alapbeállítások?
Jon Dewey, jmdewey@clunet.edu

Ezek a fájlok az anonymous ftp területen találhatók. RedHat esetén ez általában a /home/ftp-t jelenti. Ha ide elhelyezel egy welcome.msg nevű fájlt, annak tartalma fog megjelenni, amikor valaki névtelenül belép a gépedre.
Christopher Wingert, cwingert@qualcomm.com

Hiba a megosztott könyvtárak betöltése közben

Nemrég telepítettem a Netscape 4.76-ot linuxos gépemre (a gépet a minap telepítettem újra, ezért a rendszermag és a programkönyvtárak frissek), és amikor futtatni próbáltam, a következő hibaüzenetet kaptam:

```
/usr/local/bin/netscape
/usr/local/bin/netscape:
      ➥error in loading
      ➥shared libraries:
libstdc++-libc6.1-1.so.2:
      ➥cannot open shared
      ➥object file:
      ➥No such file or directory
```

Ellenőriztem a programkönyvtárakat és valóban nem volt ilyen könyvtár, csak egy újabb:

```
# cd /usr/lib
# ls libstdc++
libstdc++-3-libc6.1-2-2.10.0.a
libstdc++.a.2.10.0
libstdc++-3-libc6.1-2-2.10.0.so
libstdc++-libc6.1.so
libstdc++-libc6.1-2.a.3
libstdc++-libc6.1-2.so.3
```

A gondot sikeresen megszüntettem azáltal, hogy közvetett hivatkozást helyeztem el az újabb programkönyvtárra:

```
# ln -s libstdc++-libc6.1-2.so.3
  libstdc++-libc6.1-1.so.2
```

A gond megszűnt és a Netscape szépen fut. Ennek ellenére az eset kapcsán megfogalmazódott bennem néhány kérdés:
Mit jelentenek a .2 és .3 végződések a fájlnévben? Megfelelő-e a megoldásom? Szerettem volna egy rövidebb hivatkozásnevet, például `libstdc++-libc6.1.so`, de ez nem működött. Jobb lett volna talán megkeresni a `libc6.1-1`-et, és telepíteni a meglévő mellé? Úgy emlékszem, a Netscape programba a programkönyvtárak változatszámai bele vannak drótozva (tudomásom szerint ennek a Netscape-változatnak a forráskódja nem érhető el).
Michael, micky@alum.mit.edu

A programkönyvtár fenntartója a változatszámot általában azért változtatja meg, mert jelentős változás történt a kódban vagy a felületen. A fenntartó többnyire úgy véli, nem lenne okos megoldás, ha egy régebbi változathoz dinamikusan csatolt program az újabb változattal is minden további nélkül működne.
A „megfelelőség” nézőpont kérdése. Lehetséges, hogy a dinamikusan csatolt program, amelyet „becsaptál”, olyan szerencsétlenül fagy le, hogy magát és másokat is megsemmisít. A biztonságosabb megoldás az, ha megkeresed a hozzá való programkönyvtárat.
A Netscape 4.x binárisok valóban tartalmaznak bedrótozott könyvtárneveket és változatszámokat.
Christopher Wingert, cwingert@qualcomm.com

Tar rsh-n keresztül

Van egy kis hálózatom, amin RedHat 7.0-t használok. Miután a következő parancsot adtam ki a Lucy nevű gépen, válaszul az alábbi üzenetet kaptam:

```
[root@lucy]# tar cvf
   ➥testbed:/home/someuser file.txt

Permission denied.
tar: testbed\:/home/someuser:
   ➥Cannot open: Input/output error
tar: Error is not recoverable:
   ➥exiting now
```

A következő sorok a `testbed`-rendszer /var/log/messages állományából származnak:

```
Mar 30 08:14:57 testbed
   ➥pam_rhosts_auth[853]:
   denied to root@lucy as root:
   ➥access not allowed
Mar 30 08:14:57 testbed
   ➥in.rshd[853]: rsh denied to
   root@lucy as root:
Permission denied.
Mar 30 08:14:57 testbed in.rshd[853]:
   rsh command was ´/etc/rmt´
```

Olvastam már a PAM-ról, de nem értem. Tudna valaki segíteni abban, hogy ez a parancs működjön? Nem bánom, ha ideiglenesen el is rontom a biztonsági beállításokat a /etc/pam.d-ben, de a legutóbbi kísérlet után már be sem tudtam lépni.
Les Hilliard, les.hilliard@home.com

Azon a gépen, amelyen a tar fájlt szeretnéd használni, hozd létre a .rhosts fájlt a saját könyvtáradban, 0400 jogosultsággal. A .rhosts fájlban csak egy sor legyen: „X.X.X.X felhasználó” – ahol az „X.X.X.X” annak a gépnek az IP-címe, ahol a `tar` parancs fut, a „felhasználó” pedig annak a személynek az azonosítója, aki a parancsot futtatja.

A másik gépen add ki a következő parancsot:
`tar -cvf root@mm:/root/aa.tar munka/`
Ebben az esetben az „mm” a távoli gép neve vagy IP-címe, és a helyi könyvtár a munkakönyvtár. Vigyázat, ez a megoldás egyúttal azt is lehetővé teszi, hogy jelszó megadása nélkül lépjünk be az rlogin programmal.
Usman S. Ansari, uansari@yahoo.com

Linux telepítése egy IBM A20p számítógépre

IBM A20p típusú hordozható számítógépet használok. Nem találtam a weben semmilyen leírást azzal kapcsolatosan, hogy milyen nehézségekre (alkatrész, beállítások stb.) számíthatok, ha a gépemre Linuxot telepítek.
Mikael Koh, dragulako@yahoo.com

Hasznos adatokat tartalmaz a telepítéssel kapcsolatosan az alábbi honlap:
➲ http://www.zhlive.ch/zhl_contents_linux.html
Marc Merlin, marc_bts@valinux.com

A LinuxCare szerint ez a hordozható számítógép jól működik a SuSE 6.4-es változatával. Részletes leírást a következő címeken találhatsz:
➲ http://www.linuxcare.com/labs/certs/pada20p-suse64-sys.epl ➲ http://www.zhlive.ch/zhl_contents_linux.html.
Paul Christensen, pchristensen@penguincomputing.com

Rendszermag-pánik indításkor

Toshiba noteszgépemre Windows 98 mellé RedHat 7.0-t telepítettem. Miután a gépet bekapcsolom, egyik operációs rendszer sem indult el, csak egy csomó szám jelent meg zárójelekben, valamint az alábbi üzenet:

```
Code:89 02 85 c0 74 03 89 50 04 b8
     ➥01 00 00 00 eb 03 90 31 c0 c7

Aiee, killing interrupt handler
Kernel panic: Attempted to kill the
idle task!
In interrupt handler - not syncing
```

Most már sem lemezről, sem CD-ről nem tudok rendszert indítani.
Neil O'Connor, bowstn@yahoo.com

Indítsd el gépedet a vészindító lemezzel, amelyet a telepítéskor készítettél, és futtasd a `/sbin/lilo` parancsot. Ez helyreállítja a LILO-t, ezután a gép ismét a merevlemezről indítva fog működni.
Usman S. Ansari, uansari@yahoo.com

SMP rendszermag

Létezik olyan rendszermag, amely több processzor használatát is támogatja? Két PII 300-as processzor van a gépemben, az operációs rendszer azonban csak az egyiket ismeri fel.
Crist Besore, cbesore@kuhncom.net

Ha több processzort szeretnél használni, akkor a rendszermagot ennek megfelelően kell beállítani.

Add ki a `cd /usr/src/linux ; make menuconfig` parancsot, válaszd ki a *Processor type and feature* menüpontot, majd a *Symmetric multi-processor support* lehetőséget. Miután végeztél a beállításokkal, fordítsd újra a rendszermagot, majd telepítsd.
Pierre Ficheux, pficheux@wanadoo.fr

Mandrake X nélkül

Jelenleg Mandrake 7.2-t használok kiszolgálóként. Nem szeretném az X-et futtatni. A Mandrake telepítésekor minden lehetséges beállítást kipróbáltam, még kézzel is bejelöltem, hogy nincs szükség az X-re, a telepítő mégis felrakja. Hogyan lehet megakadályozni, hogy az X és az X-összetevők települjenek?
Egy további gondom: szöveges módban nem tudom megakadályozni a képernyő energiatakarékos módba kerülését. Letiltottam az apm démont és az energiagazdálkodást a BIOS-ban, de a képernyő még mindig lekapcsol.
Gerard Nicol, gerard.nicol@tapems.com.au

Használd az `rpm -qa` parancsot és távolítsd el azokat az összetevőket, amelyekre nincs szükséged. Érdemes felírni, miket távolítasz el – így szükség esetén később újratelepítheted őket.
Az energiatakarékos módot szöveges módban a `setterm -blank 0` paranccsal tilthatod le.
Usman S. Ansari, uansari@yahoo.com

A cron használata értesítő levelek nélkül

Gépem tízpercenként lefuttat egy `crontab` parancsot, hogy a hálózati kapcsolatom ne bontson le.
Minden egyes sikeres lefutásról elektronikus levél formájában értesítést kapok. Ki lehet kapcsolni valahogy ezt a szolgáltatást? Használhatom-e továbbra is ezt a parancsot anélkül, hogy állandóan leveleket kapnék? Jelenleg ugyanis napi 144 levelet kapok.
Scott A. Morrison Markham, scottm@ca.ibm.com

Természetesen. A /etc/crontab fájlban az alábbi parancsot használd:
`ping -c 1 gép &>/dev/null`
Marc Merlin, marc_bts@valinux.com

A merevlemez MBR-jének kitörlése

Nem sikerült Linuxot telepítenem, mert gondok akadtak az MBR-rel. A következő hibaüzenetet kaptam:
`RAMDISK: Compressed file at block 0.`
Kipróbáltam a cfdisk programot, de nem segített. Próbálkoztam mindenféle rendszerindító-, illetve vészindító lemezzel, sajnálatomra azonban már két éve nem lelem a megoldást. Miként törölhetném le ezt a merevlemezt a RH Linux telepítése előtt?
Joseph Lalingo, joseph.lalingo@ablelink.org

A `lilo -u` paranccsal visszaállíthatod az MBR-t.
Christopher Wingert, cwingert@qualcomm.com

A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tükröoldalait, a gyakran feltett kérdéseket és 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 írhatok. A levél tárgyában a „BTS” kulcsszó szerepeljen.

Programfejlesztés

Olvass többet! Tudom, ez úgy hangzik, mintha egy mozgókönyvtár plakátjáról kölcsönöztem volna, de komolyan gondolom: olvass többet! Annyiféle eszköz áll a rendelkezésünkre programíráshoz, és oly sokféle helyen tehetjük alkotásainkat közzé, hogy az írás lassan veszélyessé válik. Ezáltal pedig az olvasás sem lesz könnyebb.

Az első tanács, amit minden programozónak célszerű megjegyeznie: ne írd meg azt, amit máshonnan fel tudsz használni! Ez rendben is lenne, csakhogy ahhoz, hogy meg tudjuk ítélni, mit lehet újrafelhasználni, értenünk kell az olvasáshoz. A hatékony kód-újrafelhasználás nem csak annyiból áll, hogy például: „SSL-t akarok használni, ezért rákeresek valamilyen +ssl+ könyvtárra.” Nagyszerű, megtaláltad az OpenSSL-t. Most hogyan tovább? Miként lesz a segítségedre? Mivel nem egy barlang mélyén laktál az elmúlt tíz évben, elolvasol valahol valamilyen leírást a témáról. Milyen más megoldást – esetleg valami kevésbé közismertet – lehetne használni az SSL-en kívül? Ha nem olvasol, sosem fogsz tanulni, és ismét csak mindent magadnak kell megírnod.

Brian Behlendorf egyszer ezt írta: „Kevesebb programot a világnak!” Ez nem azt jelenti, hogy ne írjunk programokat, amennyiben szükségesek, csupán azt, hogy megfelelő indok hiányában ne írjunk programot. Természetesen eleinte sokkal élvezetesebb lehet saját sorokat írni, mint mások kódját hosszasan böngészni és megérteni, ám egyszer annak a napja is felvirrad, amikor észreveszed, hogy ezt a lenyűgözőnek tartott „új” dolgot már időtlen idők óta írod, és a puszta gondolatától is rosszul leszel; arról nem is beszélve, hogy újra és újra foglalkoznod kell a karbantartásával. Tanuld meg inkább, hogyan lehet mások munkáját újrafelhasználni, ezáltal arra a tudásra is szert teszel, hogyan dolgozhatnak neked mások.

Egy levelezőlista-tag vetette fel egyszer a következő kérdést: „Egy olyan folthoz is szükséges a szerzők nevét feltüntetnem, amely csupán kódrészleteket távolít el?” A projekt egyébként egy GUI-tervezet volt, a szóban forgó folt pedig néhány alapértelmezett elrendezéssel egy párbeszédablaknál használt tetszetős, ám félkész állapotban hagyott programot módosított. Természetesen ebben az esetben is fel kell tüntetni a szerzők nevét. A folttal ugyanis nemcsak jobbá és gyorsabbá tetted a programot, de példát is mutattál másoknak.

A kód megtisztítása (a sorok eltávolítása) a kód alapos megértését követeli meg, ami számos esetben egyben még a hozzáadásnál is magasabb fokú hozzáértést jelent – így aki erre képes, annak neve joggal kap helyet a szerzők listájában.

Emlékszel az XCoffeera? Hatalmas ötlet volt: a név egy olyan különleges célú kiszolgálóalkalmazást takart, amely kizárólag kávéképeket szolgáltatott a Hálózaton keresztül, valamint egy hasonló ügyfélprogramot, amely csak kávéképeket jelenített meg. Egészen leleményesnek számított a maga idejében – de csak kávéképeket szolgáltatni, elegendő az? És csak egyetlen képet adni ügyfélprogramonként? Az eredeti XCoffee helyére egy weboldal került; nincs már különleges ügyfélprogram, elérhető X-nélküli ügyfelekről is, és az összes, nehézkes hálózatkezelési feladatot egy webkiszolgáló látja el. Ha tudomásod lenne egy olyan általános megoldásról, amely dédelgetett gyermekedet, a saját programodat, kisebbre varázsolhatná – felhasználnád?

Az olvasáshoz visszatérve: nézzük meg, mit is találunk ebben a számban! Ha ellenőrzést kívánsz gyakorolni a forráskód felett, esetleg olyan tervezetek fejlesztői kódját szeretnéd átböngészni, amelyek hivatalosan még nem jelentek meg, akkor szükséged lesz a CVS ismeretére. Az alapokat *Ralph Krause* ismerteti a 29. oldalon.

Számos gondot okozhatnak a memóriakezelési hibák: akár szakaszolási hibát (segmentation fault) találunk első C programunkban, akár olyan biztonsági hiányosságokat, amelyek teljes cégeket dönthetnek romba. Memóriakezeléssel kapcsolatos tudásodat gyarapíthatod, ha olvasol néhány szabadon elérhető, a GNU C könyvtárban található eszközről. *Petr Sorfa* a memóriakezelésről és a hibák elhárításáról szól a 32. oldalon.

Ha cserzett arcú, öreg Unix-motoros vagy, aki az első kiadás óta velünk tart, talán kicsit gunyoros arckifejezéssel szemléled, ha az összetett fejlesztői környezetekkel kapcsolatban ejtünk szót valamiről. Míg Te ezen élcelődsz, addig mi mesélünk egy kicsit az olvasóknak a KDevelopról és a Glade-ről. A 36. oldalon *Mitch Chapman* ismerteti a Glade-et, azt a gyorsan megkedvelhető eszközt, amely segít elkülöníteni a GUI tervezését a kódolástól. Csak mutatni kell és kattintani, és máris kész a GTK-felület – a kódot pedig külön is meg lehet írni.

A legjobb az egészben az, hogy Pythont használ a példákhoz, mely remekül illeszkedik abba az újfajta szemléletbe, amely szerint „használj új, objektumközpontú parancsfájlkészítő nyelvet, és nem kell többé a sebességgel foglalkoznod”.

A munkaasztalok harcából nem maradhat ki a KDE sem, *Petr Sorfa* cikke a KDevelopról az 76. oldalon olvastható, Mustra rovatunkban. Természetesen ahogy a KDE-alkalmazások is elsősorban a C++-ra és a Qt-re összepontosítanak, egy C++ programozó számára a Qt GUI kipróbálása egyet jelent annak megkedvelésével. A Qt esztétikai letisztultságát C++ programozó nehezen fejezheti ki szavakkal, és ez így is van rendjén. A KDevelop együttműködik a CVS-sel és a Qt Designerrel, amely – mint sejthető, felhasználói felülettervező eszköz a Qt-hez; a Qt-ről bővebben majd a következő szám egyik cikkében szólunk.

Sok örömet kívánok ezekhez a fejlesztői eszközökhöz, és ne feledd: nem árt néha abbahagyni a kódolást, és olvasgatni egy kicsit!

Kapcsolódó címek

Brian Behlendorf The World Needs Less Software (A világnak kevesebb programra van szüksége) ➲ http://discuss.userland.com/msgReader$16542#16601Subversion ➲ http://subversion.tigris.org/

The Trojan Room Coffee Pot ➲ http://www.cl.cam.ac.uk/coffee/qsf/coffee.html

The Art of Unix Programming (A Unix-programozás művészete) ➲ http://www.tuxedo.org/~esr/writings/taoup/index.html

Don Marti
a Linux Journal szakmai szerkesztője, olvasóink a dmarti@linuxjournal.com címen érhetik el.

Bemutatozik a CVS

Használjuk ki a változatkövetés előnyeit, dolgozzunk együtt a projekteken, és nézzük vissza a tegnapi munkánkat.

Amennyiben valaki már töltött le programokat az Internetről, különösképpen a SourceForge-ról, valószínűleg találkozott már a CVS rövidítéssel. A CVS a Concurrent Versions System rövidítése (magyarul: párhuzamos változatkezelő rendszer), és egy olyan eszközt jelent, amely lehetővé teszi a fejlesztőknek, hogy nyomon követhessék a projekteket, illetve hogy a fejlesztők együttműködhessenek a projektek megvalósításában.
Bár a CVS-t hálózaton megosztott, számos fejlesztőt foglalkoztató nagy projektekben is lehet használni, ez a cikk inkább a helyi rendszereken történő egyedi felhasználást próbálja a középpontba helyezni. A CVS használatának gondolata első ízben általában akkor merül fel, amikor változtatásokat végzünk valamilyen parancs- vagy beállítófájlban, majd valami mással kezdünk el foglalkozni. Némi idő elteltével rádöbbenünk, hogy a változtatások mégsem voltak helyesek, de nincs biztonsági mentésünk a fájlról, ráadásul arra sem emlékszünk már, pontosan miféle változtatásokat hajtottunk végre. A CVS úrrá lehet az ilyen gondokon, mivel nyomon követi a fájlok változtatásait, és lehetővé teszi, hogy visszatérjünk a működőképes változathoz.

A CVS-ről röviden

A CVS által kezelt fájlok egy különleges raktárnak (repository) nevezett könyvtárban találhatók, ahol minden fájlhoz egy CVS által nyilvántartott változatszám tartozik. A fájlok módosításához először másolatot kell készíteni a raktárban található példányról. Lekérhetjük akár a legfrissebb vagy bármely korábbi változatot is. Ha a módosításokkal elkészültünk, a fájlt visszamásoljuk a raktárba, ahol a következő változatszámot kapja. Valahányszor fájlműveletet hajtunk végre a raktárban, egy naplóbejegyzést is készíthetünk, amely segít majd a fájlok közötti eligazodásban.
A CVS abban különbözik más változatkezelő rendszerektől, hogy nem zárolja a fájlokat, valamint a különböző fejlesztők egy időben dolgozhatnak ugyanazon a fájlon. A CVS megbizonyosodik arról, hogy valamelyik fejlesztő változtatásai nem ütköznek-e a másikéival, amikor visszahelyezi azokat a raktárba. Ha a CVS ütközést észlel, jelzést helyez a második fejlesztő fájlmásolatába, így téve lehetővé számára, hogy feloldja az ellentmondást. Az ellentmondás feloldása után a fejlesztő a fájlt visszaküldheti a raktárba.

Ízelítő a CVS-könyvekből

Az O'Reilly kiadásában megjelentetett CVS Pocket Reference *Gregor N. Purdy* tollából velős írás a CVS telepítéséről és használatáról (2000. augusztus; ISBN: 0-596-00003-0). Elérhető a ➲ http://www.kiskapu.hu/kiskapu/search.phtml?detailed=120439701 címen.
Az írás részletezi a CVS által használt felügyeleti fájlokat és leírja a CVS-parancsokat és beállításokat. A CVS-hez használható Perl-modulokról, ügyfelekről és webfelületekről felsorolást is tartalmaz.

Karl Franz Fogel Open Source Development with CVS (Coriolis Kiadó, 1999. november; ISBN: 1-57610-490-7. ➲ http://www.kiskapu.hu/kiskapu/search.phtml?detailed=2762601/) című írása a CVS használatáról és felügyeletéről tartalmaz részletes útmutatót a helyi és a távoli felhasználók számára egyaránt. Betekintést nyújt a nyílt forráskódú fejlesztés elméletébe és gyakorlatába. Kitűnő rész a „CVS tanácsok hibakereséshez" (CVS hints for troubleshooting) című fejezet, amely a szerző által tapasztalt valós helyzeteket tárgyalja. A könyv legnagyobb része GPL-szabadalom alá tartozik és a kiadó weblapján is megtalálható.

A CVS telepítése

A CVS telepítéséhez semmiféle trükk nem szükséges: a forrást letölthetjük és lefordíthatjuk vagy telepíthetjük az RPM-csomagot – teljesen mindegy, melyik megoldást választjuk.
Amint a CVS telepítésével elkészültünk, el kell döntenünk, hol legyen a raktár. Amennyiben lehetőség nyílik rá, olyan lemezterületet válasszunk, amelyen megfelelő mennyiségű hely áll rendelkezésre, és természetesen írási jogunk is van rá. A raktár kiválasztása után létrehozzuk, majd benépesítjük CVS felügyeleti fájlokkal. Ezt a CVS `init` parancsával tehetjük meg. Ha például azt szeretnénk, hogy a raktár a /usr/local/cvsstuff könyvtárban legyen, az alábbi parancsot kell végrehajtani:

```
cvs -d /usr/local/cvsstuff init
```

A `CVSROOT` környezeti változó vagy a `-d` kapcsoló mutatja meg a CVS-parancsoknak, melyik raktárban kell dolgozniuk. A `CVSROOT` változó beállításához a következő sort kell valamelyik belépési fájlba, (például .bash_profile) helyezni:

```
export CVSROOT=/usr/local/cvsstuff
```

A raktár benépesítése

Már meglévő projekteket az `import` paranccsal lehet a raktárba helyezni. Tegyük fel, hogy az alább bemutatott könyvtárszerkezetünk létezik, és a későbbiekben még további ügyfélkönyvtárakat is ide szeretnék helyezni.

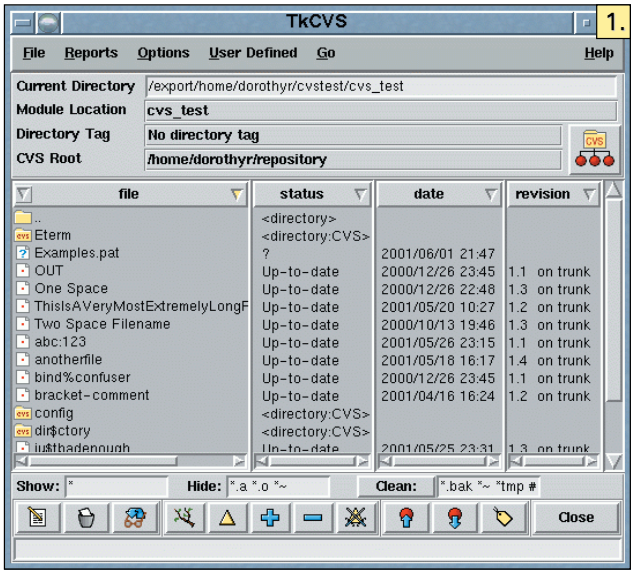
```
html_projects/
          client1/
             images/
          client2/
             images/
```

Ha a html_projects könyvtárat összes alkönyvtárával együtt a raktárba szeretnénk helyezni, a következő parancsot kellene kiadnunk:

```
cd html_projects
```

A változatkezelő következő változata

Hamarosan új versenyző tűnik fel az ez idáig kizárólag CVS uralta mezőnyben. Az új Subversion-projekt célja egy olyan változatkezelő készítése, amely kiválthatja a CVS-t. A Subversion csapat a következő területeken szeretné fejleszteni a CVS-t: könyvtárak változatkezelése, átnevezések és fájl-metaadatok; elemi műveletek (a változatszámok műveletenként adhatók és nem fájlváltozatonként); parancsok 18N-támogatása, felhasználói üzenetek és hibák; ismételt összeolvasztások könnyű kezelése; ügyféloldali diff-program kezelése, illetve hatékonyabb raktárfelület és átlátszó bináris fájlkezelés. A Subversiont már a kezdetektől úgy készítik, hogy a WebDAV segítségével kiszolgálóalapú környezetben is használható legyen.
A projekt jelenleg pre-alpha állapotban áll, de már bizonyította létjogosultságát. A szerzők remélik, hogy 2001 szeptemberére elkészülnek az alfaváltozattal. A Subversion weblap a ➲ http://subversion.tigris.org/ címen található.




```
cvs import -m "A html.projets raktárba helyezése"
➥html_projects vendor release
```

A `-m` kapcsoló segítségével a művelet hosszú nevét lehet megadni; ha nem használjuk, a CVS elindítja az alapértelmezett szövegszerkesztőt, hogy begépelhessük az üzenetet, majd a szövegszerkesztőből történő kilépés után befejezi a műveletet. A *vendor* és a *release* tagokat a CVS nem használja, ennek ellenére kötelezőek. A *vendor* tag általában a cégnév, a `start` pedig megfelel *release* tagnak. Ha a projekt bináris fájlokat is tartalmaz, például képeket, nézzünk a `-k` kapcsoló után, hogy biztosak legyünk abban, helyesen másolódnak-e a raktárba. Ha most belekukkantunk a raktárunkba, akkor a html_projects nevű könyvtárat találhatjuk benne, amely az eredeti html_projects könyvtár fájljainak másolatait tartalmazza.
Új projekt készítése CVS alatt mindössze annyiból áll, hogy elkészítjük az üres könyvtárszerkezetet, majd pedig az `import` paranccsal felvesszük a tárhelybe. Ha a projekthez új fájl kerül, a CVS `add` parancsával adhatjuk a raktárhoz.

A CVS használata

A CVS használatának alaplépései a következők: jelöljük meg a projektet a raktárban, készítsük el a kívánt változtatásokat és ellenőrizzük, hogy működnek-e, töltsük vissza a módosított fájlokat a raktárba és készítsünk megjegyzéseket a változásokhoz.
Mielőtt bármiféle szerkesztést végeznénk a fájlokon, előbb meg kell jelölnünk a CVS-raktárban. A CVS alapértelmezés szerint a projekt legutóbbi javított változatát jelöli meg, de kívánságra lehetőség nyílik a korábbi változatok lekérésére is. Amikor a projektet megjelöljük, a CVS az adott könyvtárba másolja a projekt fájljait, illetve amennyiben szükséges, alkönyvtárakat készít.
A projektfájlokat a könyvtárnév megadásával (*html_projects*) vagy egy adott projektfájl beírásával tudjuk megjelölni (*html_projects/client1/index.html*). Ha csak egy fájlt adunk meg, a projekt könyvtárszerkezete akkor is létrejön a munkakönyvtárban, de csak ez a fájl másolódik ki a raktárból.
A client1 fájljainak megjelöléséhez lépjünk be abba a könyvtárba, ahol dolgozni szeretnénk (például a saját könyvtárunkba), majd adjuk ki az alább bemutatott kijelentkező CVS-parancsot. Könnyen előfordulhat, hogy jó néhány projektváltozatunk keletkezik, ha a megjelölt parancsot nem mindig ugyanabból a könyvtárból futtatjuk:

```
cvs checkout html_projects/client1
```

Ezután váltsunk a client1 könyvtárba (`cd ~/html_projects/client1`) és kedvenc szövegszerkesztőnkkel végezzük el a változtatásokat. Mielőtt visszatöltenénk a raktárba, bizonyosodjunk meg róla, hogy változtatásaink működnek. A változati kezelő programok használata során gyakran felmerülő hiba, hogy a fájlokat túlságosan hamar tesszük vissza, ezáltal a raktárban számos fájlváltozat kerül nyilvántartásba, ráadásul a legtöbbjük nem is működik.
A megváltozott fájlt a következő paranccsal helyezhetjük a raktárba:

```
cvs commit -m "valamicskét változtattam" index.html
```

Ha a fájl sikeresen bekerült a raktárba, a CVS visszajelez, illetve megmutatja az új változat számát is.
A fájl korábbi változatait a `checkout` paranccsal, a változatszám vagy a dátum megadásával lehet elérni. Ha például az index.html jelenleg az 1.3-as változatnál tart és a tegnapi 1.2-s változatot szeretnénk elérni, gépeljük be a következő parancsok bármelyikét:

```
cvs checkout -r 1.2
       ➥html_projects/client1/index.html
```

vagy

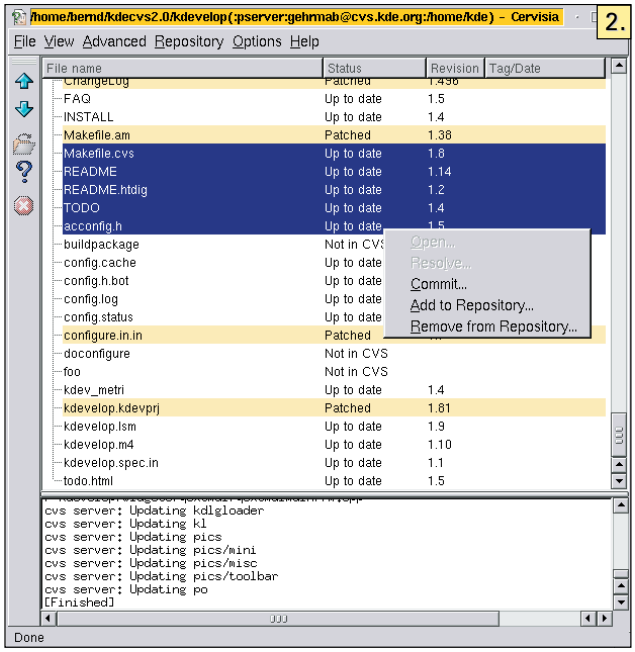
```
cvs checkout -D yesterday
       ➥html_projects/client1/index.html
```

A `-r` kapcsoló segítségével a változatszámot adhatjuk meg, míg a `-D` kapcsolóval a dátumot – utóbbi lehet szabványos ISO dátum (például 2000-03-23) vagy relatív dátum is (például *yesterday*, azaz tegnap).

Projektfájlok hozzáadása és törlése

Ha új fájlt szeretnénk a már létező projekthez hozzáadni, jelöljük meg a projektet, majd az új fájlt hozzuk létre a projekt munkakönyvtárában. Ezután a következő parancsokkal adhatjuk a raktárhoz:

```
cvs add newfile
cvs commit -m
➥"A newfile hozzáadása" newfile
```



A fájlok törlése nagyon hasonló a hozzáadáshoz. Először is jelöljük meg a projektet, majd töröljük az eltávolítandó fájlokat a munkakönyvtárból. Végül távolítsuk is el őket a raktárból:

```
cvs remove newfile
cvs commit -m
➥"Newfile törlése a raktárból" newfile
```

Projektálnevek

Az egyik lehetőség arra, hogy egy projektkönyvtárakkal teli raktárral megbirkózzunk, az, hogy a CVS-parancsokban a könyvtárnevek helyett álneveket (aliases) használunk. Az álnevek lehetővé teszik, hogy a hosszú könyvtárnevek helyett a projekteknek rövid, mégis sokatmondó neveket adjunk. Az álneveket még arra is használhatjuk, hogy különféle projekteket egyetlen név alá csoportosítsunk, így valamennyit egyetlen paranccsal megjelölhetjük. Végezetül az álnév néhány projektfájlt is takarhat, például a leírást vagy a fejállományokat (headers), ezáltal lehetővé válik, hogy egyszerre csak a projekt kis részeit jelöljük meg. Az álnevek használatához a raktár CVSROOT könyvtárában található modules fájlt szükséges szerkesztenünk. A CVS-leírás részletesen ismerteti ezt a folyamatot.

Projektfájlok felcímkézése

A CVS lehetővé teszi, hogy a `tag` (cimke) parancs segítségével jelképes vagy logikai neveket (például `release-1` vagy `beta`) rendeljünk a projekt összes fájljához. Mivel a projekt fájljainak mindegyike különböző változatszámú lehet, ez a címkézés teszi lehetővé, hogy pillanatképet készítsünk a projektről. Ezután a `tag` parancsot a `-r` kapcsolóval futtatva megjelölhetjük a projekt teljes pillanatképét (snapshot) anélkül, hogy a projekt összes fájljának pillanatnyi változatára emlékeznünk kellene. A címkékkel kapcsolatban csupán annyit szükséges megjegyezni, hogy szóközt vagy pontot nem tartalmazhatnak.

Projektágak

A CVS lehetővé teszi, hogy a projektben elágazásokat készítsünk, ahol az egyes ágak a projektkód különböző állapotait őrzik, például lehet külön águk a hibajavításhoz (bugfix) és az új képességek fejlesztéshez (new features) is. A különböző ágakon a többi ág megzavarása nélkül dolgozhatunk, majd a későbbiek folyamán a két ág változtatásait önműködően egyetlen egységgé fűzhetjük össze.
A példa kedvéért képzeljük el, hogy a *FaxMan* nevű projekten dolgozunk és kibocsátott változatunk az 1.0-s. A raktárban található forrásfájlokat kiadáskor felcímkéztük a rel-1-0 címkével, majd elkezdtünk dolgozni a 2.0-s változaton. Később kapunk egy panaszt, hogy az 1.0-s változatunk hibás és javítanunk kell. Ilyenkor a FaxMan projekt 1.0-s változatához a következők szerint készíthetünk új ágat:

```
cvs rtag -b -r rel-1-0 rel-1-0-bugfix FaxMan
```

Az `rtag` parancs új címkét (rel-1-0-bugfix) rendel a raktár kódjához. A `-b` kapcsoló azt jelenti, hogy a címke egyúttal új ágat is jelent, a `-r rel-1-0` pedig azt jelenti, hogy ez az ág a korábban `rel-1-0`-ként felcímkézett változat kódját tartalmazza.
Az 1.0-s változat megjelöléséhez a következő parancsot kell használnunk:

```
cvs checkout -r rel-1-0-bugfix FaxMan
```

A hibajavítást ezután bele kellene olvasztanunk a jelenlegi FaxMan-kódba is. Ehhez először megjelöljük a legfrissebb kódot, majd utasítjuk a CVS-t, hogy olvassza egybe a `rel-1-0-bugfix` kóddal. Ezt a következőképpen tehetjük meg:

```
cvs checkout FaxMan
cvs update -j rel-1-0-bugfix
```

CVS-ügyfelek

Számos CVS-ügyfélprogram létezik, így nem feltétlenül kell parancssorból vezérelnünk a CVS-t. A TkCVS (*1. kép*) jelenleg a 6.4-es változatnál tart, és Tcl/Tk 8.1 vagy újabb változat szükséges hozzá. A Pharmacy, ami jelenleg 0.2.1-es változatszámmal rendelkezik, a Gnome-projekt része, míg a Cervisia – *2. kép* (száma 1.0-stable) a Qt és a KDE-könyvtárakat használja. Egy másik, szintén Qt-eszköztárat használó program a 0.3 változatszámú LinCVS. A *Kapcsolódó címek* között további adatokat találhatunk ezekről a projektekről.

Összegzés

A CVS kiemelkedően hatékony eszköz, amely egymástól távoli helyeken élő fejlesztők Interneten keresztüli bármilyen projektben együttműködését teszi lehetővé, ugyanakkor helyi gépen is könnyedén beállítható és használható. Ha a fájljainkat gyakran frissítjük és változásvezetést szeretnénk, a CVS az összes végrehajtott változtatásról jegyzőkönyvet készít.

Ralph Krause
(rkrause@netperson.net) Michiganben alkotó író, programozó és webmester.

Kapcsolódó címek

CVS ➲ http://www.cvshome.org/
Cervisia ➲ http://cervisia.sourceforge.net/
LinCVS ➲ http://www.lincvs.org/
Pharmacy ➲ http://pharmacy.sourceforge.net/
TkCVS ➲ http://www.twobarleycorns.net/tkcvs.html

Memóriahiba-keresés Linux alatt

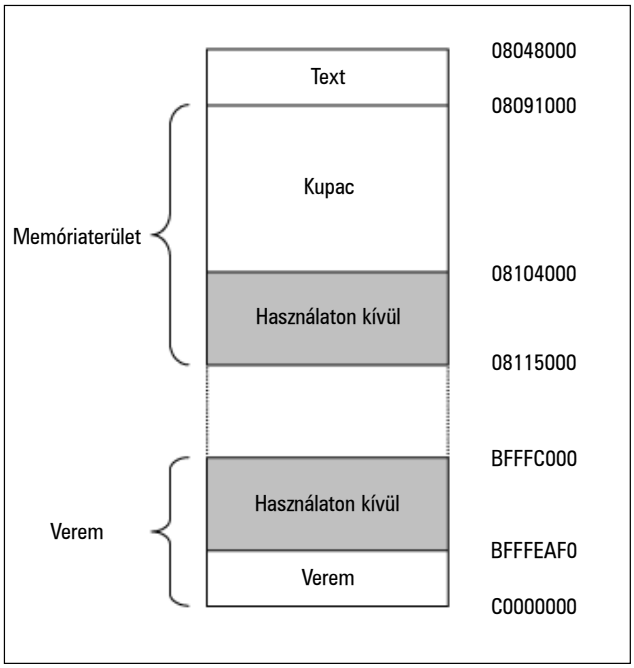
Petertől megtudhatjuk, miként kerülhetik el a programozók a kellemetlen memóriahibákat.

Minden program használ memóriát. Még azok is, amelyek semmit sem csinálnak. A memória helytelen használata pedig meglehetősen gyakori oka az olyan végzetes programhibáknak, mint például a programleállás vagy a váratlan viselkedés.
A memória az adatok tárolására szolgáló eszköz. A program memóriája általában egy bizonyos mennyiségű fizikai memóriához rendelhető, ha azonban éppen nincs használatban, lehet valamilyen háttértárolón is, például a merevlemezen.
A felhasználók szempontjából a memóriát két eszköz használja: maga a rendszermag és az éppen futó program, olyan memóriaszolgáltatásokon keresztül, mint a `malloc()` .

A magmemória

Az adott programhoz vagy annak példányaihoz (az operációs rendszer egy időben több példányt is képes futtatni) szükséges összes memóriát az operációs rendszer magja kezeli. Amikor a felhasználó végrehajt egy programot, a mag bizonyos memóriaterületet foglal le a program számára. Ezután a program a memóriarészt a következő területekre felosztva kezeli:

- Szöveg (Text) – Ez az a terület, ahol a program csak olvasható részei tárolódnak, tulajdonképpen ez a program utasításkódja. Ugyanazon program példányai ezt a memóriaterületet megosztják egymás között.
- Állandó adat (Static Data) – Az előre ismert memória számára lefoglalt terület. Általában a globális változók és statikus C++ osztályelemek kerülnek ide. Az operációs rendszer minden egyes programpéldányhoz külön lefoglal egy ilyen területet.
- Memóriaaréna (Memory Arena, avagy break space) – Ez az a terület, ahol a dinamikus futásidejű memória tárolódik. A memóriaaréna két részből áll: a kupacból és a felhasználatlan memóriából. A felhasználó által lefoglalt memória a kupacban helyezkedik el. A kupac az alacsonyabb memóriacímektől a magasabbak felé növekszik.
- Verem (Stack) – Valahányszor a program függvényhívásokat hajt végre, a pillanatnyi függvényállapotot a verembe kell menteni. A verem lefelé növekszik, a magasabb memóriacímektől az alacsonyabbak felé. Minden egyes programpéldány számára egyedi memóriaaréna és verem foglalódik.



1. ábra A programpéldányhoz rendelt memória

Felhasználói memória

A felhasználó által foglalható memória a memóriaaréna kupacrészében található. A memóriaaréna kezelését a `malloc()`, `realloc()`, `free()` és `calloc()` eljárások végzik. A felsorolt eljárások a libc részét képezik. Lehetséges azonban, hogy ezeket a függvényeket valamilyen más megvalósítással helyettesítsük, amelyek adott feladat esetén esetleg hatékonyabb megoldást biztosítanak. Az *1. ábrán* ilyen memóriafüggvény-választási lehetőségeket soroltunk fel.
A Linux-rendszereken a programok memóriaaréna-területe előre meghatározott egységekben növekszik, általában egy memórialap-egységenként, avagy a laphatárhoz igazodva. Ha a kupacnak több memóriát igényel, mint amennyi a memóriaarénában rendelkezésre állna, a memóriafüggvények meghívják a `brk()` rendszerhívást, amely további memóriaterületet igényel a magtól. Az igényelt terület méretét a `sbrk()` hívással lehet beállítani.
Bármely folyamat pillanatnyi memóriaaréna- és veremterületét megtekinthetjük, ha megvizsgáljuk az adott folyamathoz tartozó /proc/<pid>/maps fájl tartalmát, ahol a `pid (process id)` a folyamat azonosítója (lásd az *1. listát*).

Szerkezet

Valahányszor a `malloc()` függvénnyel új memóriaterületet foglalunk le, mindig egy kevéssel több memóriát kapunk, mint amennyit kértünk. A memóriaeljárások ezt a további területet használják fel karbantartásra. Ha a felhasználói kérésre lefoglalt memória valós méretére vagyunk kíváncsiak, használjuk a `malloc_usable_space()` függvényt. A valós memóriaszelet ennél általában nyolc bájttal nagyobb.
A memóriaszelet szerkezete a szelet végén található és a szelet korábbi méretét tartalmazza (*2. ábra*). A méretérték egy további bitet is magában foglal: ez mutatja meg, hogy a memóriakezelő rendszer fenntart-e egy másik memóriaszeletet közvetlenül a pillanatnyi előtt.
A GNU libc memóriaeljárásai rekeszeket használnak a hasonló méretű memóriaszeletek tárolására, ilyen módon segítve elő a növekvő teljesítményt, és egyúttal megakadályozzák a széttöredezett memóriaterületek kialakulását, ahol a memóriaaréna belsejében kihasználatlan memórialyukak maradnak. Ezek a memóriaeljárások természetesen szálbiztosak (threadsafe) is. Jóllehet ezek a függvények gyorsak és megbízhatók, van néhány terület, ahol fejleszthetők lennének, például a sebesség és a memórialefedettség terén.

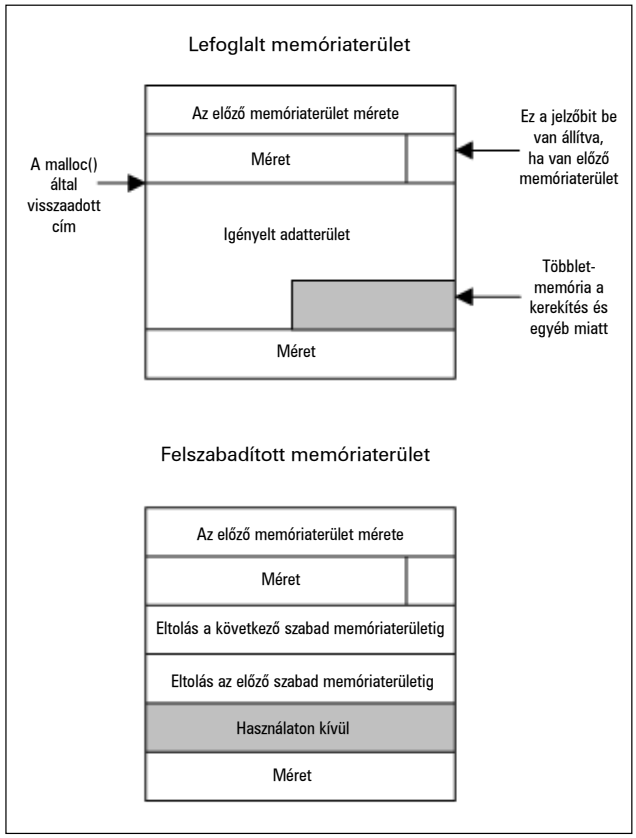
1. lista A /proc/<pid>/maps kimenete

```
<home>$ cat /proc/$$/maps
08048000-08091000 r-xp 00000000 03:03 77807
      ➥/bin/bash
08091000-08097000 rw-p 00048000 03:03 77807
      ➥/bin/bash
08097000-08115000 rwxp 00000000 00:00 0
40000000-40016000 r-xp 00000000 03:03 33122
      ➥/lib/ld-2.2.so
40016000-40017000 rw-p 00015000 03:03 33122
      ➥/lib/ld-2.2.so
40017000-40018000 rwxp 00000000 00:00 0
40018000-4001a000 rw-p 00000000 00:00 0
40023000-40026000 r-xp 00000000 03:03 31161
      ➥/lib/libtermcap.so.2.0.8
40026000-40027000 rw-p 00002000 03:03 31161
      ➥/lib/libtermcap.so.2.0.8
40027000-40148000 r-xp 00000000 03:03 33125
      ➥/lib/libc-2.2.so
40148000-4014e000 rw-p 00120000 03:03 33125
      ➥/lib/libc-2.2.so
4014e000-40152000 rw-p 00000000 00:00 0
40152000-4015c000 r-xp 00000000 03:03 33137
      ➥/lib/libnss_files-2.2.so
4015c000-4015d000 rw-p 00009000 03:03 33137
      ➥/lib/libnss_files-2.2.so
4015d000-40167000 r-xp 00000000 03:03 33140
      ➥/lib/libnss_nisplus-2.2.so
40167000-40169000 rw-p 00009000 03:03 33140
      ➥/lib/libnss_nisplus-2.2.so
40169000-4017c000 r-xp 00000000 03:03 33130
      ➥/lib/libnsl-2.2.so
4017c000-4017d000 rw-p 00012000 03:03 33130
      ➥/lib/libnsl-2.2.so
4017d000-40180000 rw-p 00000000 00:00 0
40180000-4018a000 r-xp 00000000 03:03 33139
      ➥/lib/libnss_nis-2.2.so
4018a000-4018b000 rw-p 00009000 03:03 33139
      ➥/lib/libnss_nis-2.2.so
bfffc000-c0000000 rwxp ffffd000 00:00 0
```

Nyomkövetés

A memóriakezelés hibákat és többnyire váratlan memóriaviselkedést okozhat. Ennek egyik oka az lehet, ha felszabadított memóriát használunk, azaz olyan memóriaszelettel dolgozunk, amelyet a program már felszabadított. Bár ez nem feltétlenül idéz elő azonnali hibát, valamilyen gond mégis biztosan felmerül, ha ugyanerre a területre új memóriafoglalás kerül bejegyzésre. Ennek eredményeképpen – mivel ugyanazt a memóriaterületet két különböző célra használjuk – váratlan értékeket fog eredményezni, sőt könnyen a program leállásához és *core dump*-hoz vezethet, ha memóriaterület-mutatókat vagy eltolási értékeket is tartalmaz.

A másik hibaforrás az lehet, ha a memóriaszelet előtagját írjuk felül. Amennyiben ugyanis a program felülírja a memóriaszelet bevezető részét, a memóriakezelő rendszer szinte bizonyosan hibázni fog vagy váratlan működést mutat, amikor a hibás memóriaszelettel találkozik. Néha a felülírás a szomszédos memóriaszeletben történik, és ez az ottani adatokat károsíthatja. Elképzelhető, hogy a felhasználó csak később veszi észre ezt a hibát, amikor a programvégrehajtás során furcsa értékeket vagy gyanús viselkedést tapasztal.



2. ábra A memóriaszelet szerkezete

Hasonlóképpen, ha a felszabadított memóriaszelet kezelőadatait összezavarjuk, felülírjuk vagy indokolatlanul használjuk, több mint valószínű, hogy a memóriakezelő rendszer maga is hibázni fog.

A memóriaaréna szabad (lefoglalatlan) részeinek használata ugyancsak okozhat hibát. Elképzelhető, hogy a kupacon kívül használunk fel memóriát, de még nem lépünk ki a memóriaarénából. Ez önmagában nem okoz gondot mindaddig, amíg frissen lefoglalt terület nem kerül ugyanerre a helyre. Az ilyesféle hibát igen nehéz felfedezni, mivel a rákövetkező memóriaműveletek (szabályosan) a kupacon belül találhatók.

A legnyilvánvalóbb és közvetlen hiba, amikor a program a memóriaarénán és a programterületen kívülre próbál írni. Ez azonnal SIGSEGV hibát (segmentation violation fault) okoz, következésképpen a program önműködő módon core dumpot készít.

A legkártékonyabb és legnehezebben nyomon követhető memóriahiba az, amikor a program verme károsodik. A program rengeteg helyi változót, értéket, az előző keretből (frame) származó regisztert, és ami a legfontosabb: visszatérési címet tárol a veremben. Így ha a verem megsérül, a programot hagyományos hibakeresővel szinte lehetetlen nyomon követni, mivel a veremhatárok maguk is használhatatlanná válnak. A veremmemória-hibák felderítésére mindössze néhány nyílt forráskódú (például libsafe) és kereskedelmi termékként kínált memóriahiba-kereső alkalmas, mivel a veremmemória-hibák felderítéséhez a programvégrehajtás menetét kell megváltoztatni vagy fejleszteni.

Létezik néhány módszer, amelyek segítségével meg lehet próbálni felderíteni a hibás memóriahasználatot, közülük néhány sajnos mellékhatásokat is okoz, például lassul a programvégrehajtás, vagy nagyobb lesz a memóriafelhasználás, következésképpen a nagy memóriaigényű programokban nemigen használhatók.

A következőkben bemutatandó hibakeresőkben használt hibás példa-

2. lista mytest00.c példaprogram

```
#include <stdlib.h>
#include <stdio.h>

int
main (int argc, char **argv)
{
        char    *msg = malloc (4);
         // Lefoglalunk 4 bájtot

        strcpy (msg, "hello Linux users");
         // Túlcsordultatjuk a lefoglalt
         // memóriát
        printf ("%s\n", msg);
        free (msg);
         // Felszabadítjuk a foglalt memóriát
        strcpy (msg, "hello again");
         // Ráírunk a felszabadított helyre
        printf ("%s\n", msg);
        free (msg);
         // Felszabadítjuk a felszabadított
         // memóriát
        realloc (msg, 2);
         // Újra lefoglaljuk a felszabadított
         // memóriát
        strcpy (msg, "hello there");
         // A hibás memóriába írunk
        printf ("%s\n", msg);

        return 0;
}
```

3. lista mytest01.c példaprogram

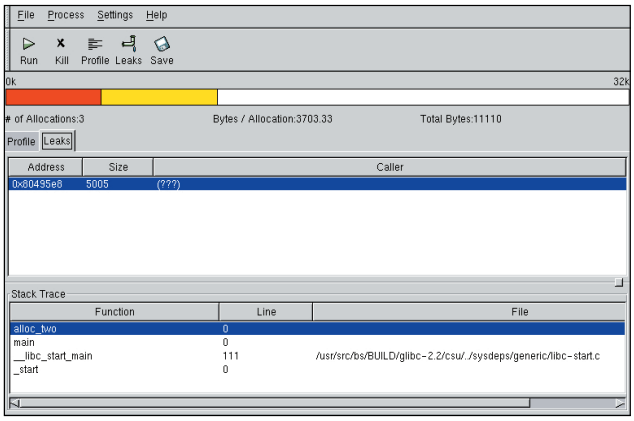
```
#include <stdlib.h>
#include <stdio.h>

int
main (int argc, char **argv)
{
      char    msg[4];
       // 4 bájtot foglalunk a veremben

      strcpy (msg, " hello Linux users 1234");
       // Túlcsordultatjuk a veremhelyet
      printf ("%s\n", msg);

      return 0;
}
```

programok forrása a *2.*, a *3.* és a 4. listában található. (A 4., 5., 6. listák a 15. CD Magazin/memoriahibak könyvtárában találhatók.) Alapértelmezés szerint a MALLOC_CHECK_ környezeti változó egyre állítva minimális hibakeresési célokra használható fel az alapértelmezett *malloc*-függvényhez. Ha a MALLOC_CHECK_-et egyre állítjuk, hibajelentéseket kapunk, ha pedig kettőre, a program bármilyen memóriahibánál azonnal leáll. A kimenet meglehetősen rejtélyes is lehet, mivel a hibakereső mód a hibás területeket olvasható szimbólumok helyett cím formájában jeleníti meg. Ezért nem árt, ha kéznél van egy nyomkövető, amivel megállapíthatjuk, hogy a programban hol is keletkeztek ezek a hibák. A következő példában alapértelmezett memóriahiba-keresést alkalmaztunk:



3. kép Memóriahiba-keresés memproffal

```
<home>$ MALLOC_CHECK_=1 ./mytest00
malloc: using debugging hooks
hello Linux users
free(): invalid pointer 0x80496d0
hello again
free(): invalid pointer 0x80496d0
realloc(): invalid pointer 0x80496d0
malloc: top chunk is corrupt
hello there
```

A kimenet azt mutatja, hogy a hiba a mytest00.c fájl 8. sorában (2. lista) található, ahol az `strcpy()` függvény túlcsordul és megsérti azt a memóriaszeletet, amelyre az `msg` mutat. A további nyomkövető hibaüzenetek ennek a károsodásnak a következményei.
Létezik néhány kitűnő nyílt forráskódú memóriaeszköz is. Az egyes megvalósítások memóriahiba-lefedettségben, kimenetben és interaktivitásban különböznek.
Az Electric Fence az egyik legkönnyebben használható eszköz. A könyvtár végrehajt néhány memóriaellenőrzést, majd amikor hibát észlel, megállítja a programot. Ez többnyire core dump formájában történik, amelyet aztán a felhasználó hibakereső programmal megvizsgálhat. Az Electric Fence a leghatékonyabban valamilyen nyomkövető programba ágyazva alkalmazható, például a GNU hibakeresőjében, a GDB-ben. Amikor az Electric Fence leállítja a programot, a GDB kapja meg a vezérlést, mégpedig pontosan azon a helyen, ahol a hiba bekövetkezett (lásd az 5. listát).
Ez a példakimenet az Electric Fence könyvtárral fordított ellenőrző-program végrehajtását mutatja GDB alatt. A legelső sérelem a mytest00.c 8. sorában SIGSEGV-et okoz. A GDB által felajánlott veremkövető segítségével a felhasználó azonosíthatja a hiba helyét.
A libsafe-rendszert többféle lehetséges veremhatár-átlépési sérelem ellenőrzésére is fel lehet használni, de csak néhány C könyvtárfüggvény esetében (`strcpy`, `strcat`, `getwd`, `gets`, `scanf`, `vscanf`, `fscanf`, `realpath`, `sprintf` és `vsprintf`) alkalmazható.
A libsafe példa kimenete igen tömör: amint a veremhiba bekövetkezik, a libsafe megjeleníti a hibaüzenetet, majd leállítja a programot. Emellett a libsafe az adott hiba részleteit több levélcímre is el tudja küldeni. Ez a hibajelentésnek kétségtelenül csavaros módja, de a libsafe-et a felhasználók elsődlegesen mégis inkább a puffertúlcsordulást kihasználó biztonsági töréskísérletek felderítésére használják. Egy kis szerkesztéssel a fejlesztő átalakíthatja a libsafe kódját, hogy

valamivel adatdúsabb üzeneteket küldjön. A másik lehetőség, hogy a programot GDB-ben hajtjuk végre és a `_libsafe_die()` címkére töréspontot állítunk be, ami a libsafe által felderített veremsértés esetén lép működésbe. A következő példában a libsafe veremfelülírást észlel, amit a mytest01.c 8. sorában található `strcpy()` hívás okoz (3. lista):

```
<home>$ LD_PRELOAD=/lib/libsafe.so.1.3  ./mytest01
Detected an attempt to write across stack
boundary.
Terminating mytest01.
Null message body; hope that´s ok

# Email is sent with the following subject
# header

libsafe violation for /tmp/mytest01, pid=27265;
overflow caused by strcpy()
```

A debauch kimenetét szimbólumok helyett kizárólag címek alkotják, emiatt csakis valamilyen hibakeresővel együtt érdemes alkalmazni. A debauch rendelkezik néhány, a felhasználó által előhívható különleges képességgel, amelyeket kifejezetten GDB alatti használatra terveztek. Ezek a képességek jobb memóriafoglalási és felszabadítási nyomkövetést tesznek lehetővé. A debauch meglehetősen alapos, és igen sokfajta memóriahibát képes észlelni, illetve azok megtörténte után helyreállítani (lásd a 6. listát).

A memprof legfőbb jellemzője a grafikus felület, amely könnyen érthetővé teszi, és ezen keresztül világosan látható lesz a memóriahiba kialakulásának helye. Meglehetősen hatékony képességekkel rendelkezik, ugyanis a bináris fájlleíró könyvtár (BFD library) által támogatott függvényeket használja – ugyanazokat, amelyeket a GDB is használ a folyamatok irányítására. *Képünkön* (lásd a 30. oldalon) bemutatjuk, amint a memprof megtalálja a lyukat a mytest02.c program `alloc_two()` függvényében.

A nyílt forráskódú memóriaeszközökön kívül néhány kereskedelmi termék is megvásárolható, ezek többnyire grafikus felülettel rendelkeznek, és még alaposabb ellenőrzést tesznek lehetővé, mint nyílt forráskódú megfelelőik (a kereskedelmi memóriaeszközök felsorolása a *Kapcsolódó címek* részben található.)

Végső megoldásként esetleg saját memóriakezelő-függvények megírása is szóba kerülhet. Ez sokat segíthet a memóriakezelés megértésében, illetve bizonyos egyedi esetekben teljesítménynövekedést eredményezhet (például nagy memóriaterületek gyors foglalása és felszabadítása esetén).

A memóriahibák felderítése nagyon fontos, nemcsak a megbízhatóság, hanem a biztonság szempontjából is. A Linuxhoz számos memóriahibakereső létezik, és mindegyik sajátos képességekkel bír, illetve egyedi működési feltételei vannak. Legjobban tesszük, ha a programot a fenti hibakeresők közül többel is kipróbáljuk valamilyen nyomkövető, például GDB alatt, hiszen így a hibák szélesebb tartományára derülhet fény.

Petr Sorfa
(petrs@sco.com) a Santa Cruz Operation's Development Systems Group tagja, ahol cscope és a Sar3D nyílt forráskódú projektek felelőseként tevékenykedik. Főiskolai végzettséget szerzett a Cape Town és a Rhodes University-n. Érdeklik a nyílt forráskódú fejlesztések, a számítógépes grafika, a fejlesztőrendszerek, továbbá a képregény-rajzolás (sequential art).

Kapcsolódó címek

Választható memóriafüggvények

Boehm Garbage Collector
➲ http://www.hpl.hp.com/personal/Hans_Boehm/gc/

CSRI ➲ ftp://ftp.cs.toronto.edu/pub/moraes/malloc.tar.gz

GNU Malloc
➲ ftp://ftp.cs.colorado.edu/pub/misc/mallocimplementations/

Hoard ➲ http://www.cs.utexas.edu/users/emery/hoard/

Ptmalloc (GNU C Library)
➲ http://www.malloc.de/en/index.html

QuickFit Malloc ➲ ftp://ftp.cs.colorado.edu/pub/misc/qf.c

vmalloc (az ast része)
➲ http://www.research.att.com/sw/download/

Nyílt forráskódú memóriaeszközök

ccmalloc
➲ http://www.inf.ethz.ch/personal/biere/projects/ccmalloc/

Checker
➲ http://www.gnu.org/software/checker/checker.html

dbmalloc ➲ http://dickey.his.com/dbmalloc/dbmalloc.html

DbMalloc ➲ http://www.cs.bris.ac.uk/~mm7323/DbMalloc/

debauch ➲ http://quorum.tamu.edu/jon/gnu/

dmalloc ➲ http://dmalloc.com/

Electric Fence ➲ http://www.perens.com/FreeSoftware/

fda ➲ http://packages.debian.org/unstable/devel/fda.html

leak ➲ http://sources.isc.org/devel/memleak/leak.txt

LeakTracer ➲ http://www.andreasen.org/LeakTracer/

libcw ➲ http://libcw.sourceforge.net/debugging/

libsafe ➲ http://www.bell-labs.com/org/11356/libsafe.html

MCheck
➲ http://www.cs.vu.nl/~rveldema/mcheck/mcheck.html

Memleak (Az X11R6.4 része)
➲ ftp://ftp.x.org/pub/R6.4/xc/util/memleak/

Memdebug
➲ http://www.bss.lu/Memdebug/Memdebug.html

MemProf ➲ http://people.redhat.com/otaylor/memprof/

Memwatch ➲ http://www.link-data.com/sourcecode.html

MM ➲ http://www.engelschall.com/sw/mm/

mpatrol ➲ http://www.cbmamiga.demon.co.uk/mpatrol/

mpr ➲ http://freshmeat.net/projects/mpr/

NJAMD ➲ http://fscked.org/proj/njamd.shtml

YaMa
➲ http://www.geocities.com/ipsgvm/libyama/index.html

YAMD ➲ http://www3.hmc.edu/~neldredge/yamd/

Kereskedelmi memóriaeszközök

Aprobe ➲ http://www.aprobe.com/

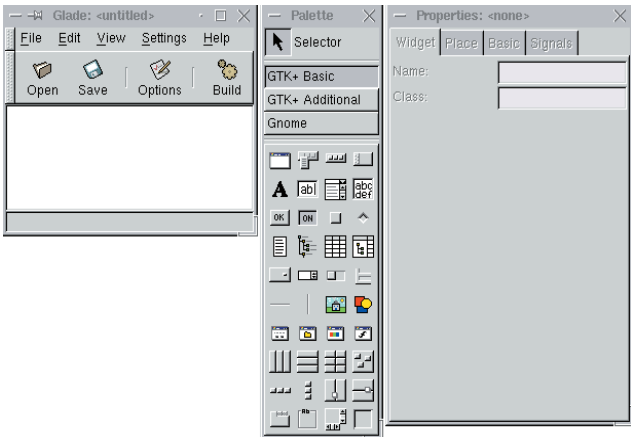
Insure++ ➲ http://www.parasoft.com/products/insure/

Purify ➲ http://www.rational.com/products/purify_unix/

Felhasználói felületek létrehozása Glade-del

Hogyan használjuk a gnome-python libglade-jét Python-alapú grafikus alkalmazások fejlesztéséhez?

A Glade a Gtk+ eszközkészlet GUI-készítője (GUI = Graphical User Interface – grafikus felhasználói felület). A Glade segítségével kényelmesen gyárthatunk felhasználói felületeket és hozzá forráskódot, valamint ezekhez a felületekhez a visszahívó (callback) függvények vázát is létre tudjuk hozni.
A *libglade* programkönyvtár lehetővé teszi a programok számára, hogy a Glade-projektfájlokban megadott elemszerkezetet könnyen megjelenítsék. Ennek része a projektfájlban megnevezett visszahívók csatolása a program által nyújtott visszahívó programrészekhez.
James Henstridge a fenntartója a libglade és a gnome-python csomagnak, mely a Pythont a Gtk+ eszközkészlettel és a Gnome felhasználói felülettel köti össze, valamint magával a libglade-del. Ha Python-alapú grafikus alkalmazásainkban libglade-et használunk, a fejlesztési és a karbantartási idő jelentősen rövidül.
A cikkben szereplő összes példaprogram Glade 0.5.11, gnome-python 1.0.53 és Python 2.1b1 segítségével készült, Mandrake Linux 7.2 rendszeren.



1. kép A Glade indítása

A Glade futtatása

Indítás után a Glade három ablakot jelenít meg (lásd *1. kép*). Az alkalmazás főablaka mutatja a pillanatnyi Glade-projektfájl tartalmát, azaz a projektfájlban meghatározott ablakok és párbeszédablakok listáját.
A palettaablak mutatja a Glade által támogatott Gtk+ és Gnome-elemeket. Amikor egy elemet szerkesztésre kijelölünk, a tulajdonságablak jeleníti meg az elem tulajdonságainak pillanatnyi értékeit.
A palettaablak három csoportra osztja a Glade által támogatott elemeket: a „GTK+ Basic” (Alap) a leggyakrabban használt Gtk+ elemeket, a „GTK+ Additional” (Kiegészítő) a ritkábban alkalmazott elemeket tartalmazza, például a vonalzót és a naptárat. A Gnome-elemek a GNOME UI programkönyvtárból származnak.
A tulajdonságablak négylapos füzetben ábrázolja az elem tulajdonságait. A *Widget* lap az elem nevét és az elemosztályra jellemző tulajdonságokat jeleníti meg. Amikor az elem egy kényszerfeltétel-alapú tároló (például GTKTable vagy GTKVBox) belsejében helyezkedik el, a *Place* lap azokat a tulajdonságokat mutatja meg, amelyek az elem elhelyezkedését befolyásolják a tárolóban – máskülönben ez a lap üres. A *Basic* oldal olyan alapvető tulajdonságokat tartalmaz, mint például a szélesség és a magasság, ezekkel minden elem rendelkezik. Végül a *Signals* oldal segítségével adható meg, hogy az elem milyen jelzéseket adjon ki, és ezekhez a jelzésekhez megadhatjuk a kezelő függvényeket.

Elemszerkezetek létrehozása

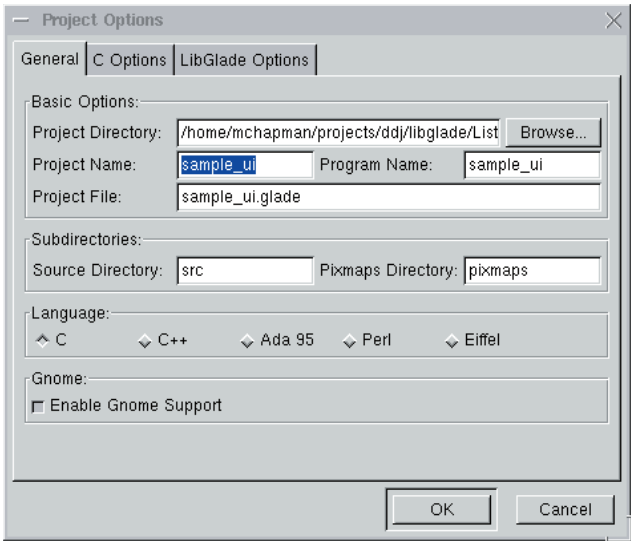
Az elemszerkezetek létrehozásának folyamata a Glade-ben hasonlóképpen zajlik, mint a Visual Basicben. A szerkezet kiindulópontja a felső szintű ablak vagy párbeszédablak. Az elemek a felső szintű ablakban helyezhetőek el úgy, hogy előbb a Glade palettaablakából kiválasztjuk őket, azután a tároló olyan részére kattintunk, ahol a célkereszt megjelenik.

Jelzéskezelők létrehozása

A Signals lap a Glade tulajdonságablakán lehetővé teszi, hogy az elemhez alkalmazásfüggő viselkedést rendeljünk. A lap felső részén láthatók a pillanatnyi elemhez tartozó jelzéskezelők. Az alsó részen található gombok segítségével az elem által kibocsátott jelzések közül lehet választani és a kezelője is létrehozható.
Új jelzéskezelő létrehozásához a *Signal entry* mező mellett található három pontra kell kattintani. Ekkor megjelenik a *Select Signal* párbeszédablak, amely felsorolja az elem által kibocsátható összes jelzést. A jelzések azok szerint a Gtk+ elemosztályok szerint vannak csoportosítva, amelyekben meghatározták őket.
A jelzés kijelölése után kattintsunk a Select Signal párbeszédablakban az *OK* gombra: a kijelölt jelzés neve megjelenik a Signals lapon a Signal entry mezőben. A Glade önműködően kitölti a *Handler* mezőt is, az `"on_<widget>_<signal>"` elnevezést használva. Ha igényeinknek ez nem felel meg, kézzel megváltoztatható.
A Signals lap alján további beviteli mezőket találunk, ahol alkalmazásfüggő adatok adhatók meg: a jelzést elfogó objektum stb. Ezeket mindig hagyjuk üresen, ugyanis szükségtelenek, amikor gnome-pythonnal dolgozunk.

Glade-projektfájlok

A Glade a projekt adatait a .glade kiterjesztésű XML fájlba menti. A Glade XML használata könnyűvé teszi különféle a projektfájlon dolgozó segédprogramok kifejlesztését, például új programnyelvekhez való kódlétrehozó írását.
Az új projekt első mentésekor a Glade felhozza a *Project Options* párbeszédablakot. A Project Options párbeszédablak legtöbb beállítása akkor és csak akkor számít, ha a Glade-et a projekt forráskódjának létrehozására használjuk. Néhány beállítás azonban – például a projekt könyvtára – akkor is fontos, ha a Glade-et csak az elrendezés megtervezésére használjuk.
Alapértelmezés szerint a Glade feltételezi, hogy az új projektet bejelentkezési könyvtárunk alá, a *Projects/project1* könyvtárba szeretnénk menteni. Ez nem feltétlenül egyezik meg az akaratunkkal. Én általában abba a könyvtárba szoktam menteni, ahonnan a Glade-et indítottam. Szerencsére könnyű megváltoztatni a projekt könyvtárát. Kattintsunk a *Browse…* gombra a *Project Directory* beviteli mező mellett, amely



3. kép A Project Options párbeszédablak

után a *Select the Project Directory* párbeszédablak jelenik meg. Ez a párbeszédablak a Glade aktuális munkakönyvtárát választja ki alapértelmezettként, ezért csak az OK gombot kell megnyomni. Ezután a Project Directory mező a Project Options párbeszédablakban a pillanatnyi munkakönyvtárat fogja mutatni, és a *Project Name* mező üres lesz. Írjuk be az új projektnevet, ennek hatására a Project Name és a *Project File* mezők frissülnek (lásd *3. kép*). Az OK lenyomása után a projekt a megadott projektfájlba íródik.

A libglade használata

Ha egyszer létrehoztunk egy Glade-projektfájlt, grafikus felületet (amelyet a projektfájl ír le) hozhatunk létre a gnome-python libglade modulját használva, és programból elérhetjük a szerkezet elemeit.

```
import libglade
loader = libglade.GladeXML
       ➥("helloworld.glade", "window1")
```

A libglade programkönyvtár megadja a GladeXML-osztályt, ez végzi a munka nagy részét. Egy elemszerkezet betöltéséhez a GladeXML-osztály egy példányát kell létrehozni, átadni neki a Glade-projektfájl nevét és a legfelső elem nevét, amelyet meg akarunk jeleníteni. Jegyezzük meg, hogy a szerkezet bármelyik elemét megadhatjuk itt, még akkor is, ha egy legfelső szintű ablakban mélyen el van ásva. Ez lehetővé teszi, hogy az összetett szerkezeteket – például egy bonyolult, füzetszerű felület lapjait – több Glade-projektfájlba osszuk el. Könnyen kezelhetők a dinamikus látványtartalmú projektek is, csak azt az összetevőt kell betölteni, amelyik éppen szükséges. Ha az elemszerkezet már be van töltve, egy adott elemet a GladeXML `get_widget` eljárásával kaphatunk meg. A `get_widget` a kért elemet adja vissza; ha pedig az elem nem található, akkor *None*-t.

```
window1 = loader.get_widget("window1")
if window1:
    window1.set_title("Szia Világ!")
```

Jelzéskezelők hozzákapcsolása

A GladeXML leghatékonyabb jellemzőinek egyike, hogy képes összekapcsolni a Python által hívható (Python callable) objektumait (eljárásokat, függvényeket stb.) a Glade-projektfájlban megnevezett jelzéskezelőkkel. Mindezt a `signal_autoconnect` eljárás teszi lehetővé.

A signal_autoconnect csak egy értéket vár: egy olyan szótárt, amely a jelzéskezelők nevét Python által hívhatókra képezi le. Minden egyes jelzéskezelőre, amelyet a Glade-projektfájlban megadtunk, a signal_autoconnect a meghatározott szótárból kikeresi a megfelelő Python által hívhatót, s ha egyező bejegyzést talál, akkor a jelzéshez kapcsolja – azaz a Python által hívható válik a jelzés kezelőjévé.

```
def button1_click_handler(*args):
    print "Ne nyomd meg a gombot!"

signal_handlers = {
    # Kilépés a főhurokból, ha a felhasználó
    # bezárja a főablakot.
    'on_window1_delete_event': gtk.mainquit,
    # A button1_click_handler hívása, ha a
    # felhasználó megnyomja a button1 gombot.
    'on_button1_clicked': button1_click_handler
    }

loader.signal_autoconnect(signal_handlers)
```

GladeBase

A libglade nagymértékben csökkenti a kézi kódolás szükségességét gnome-python alkalmazások esetében. Az elemszerkezeteket megtervezhetjük Glade-del, és két-három sor kód segítségével be is tölthetjük. Mindez több száz sort venne igénybe, ha közvetlen *pygtk* hívásokat használnánk. Ráadásul a viselkedés is egyszerűen hozzáadható a Python által hívhatók szótárának összeállításával és a `GladeXML.signal_autoconnect` átadásával, ahelyett hogy állandóan az elemcsatoló eljárásokat kellene hívogatni. A libglade sok munkát takarít meg, ennél azonban többre is képes, például a nagyméretű Python-alkalmazások gyakran egy kis főprogramból és valahol a Python elérési útjában elhelyezkedő kapcsolt csomagokból állnak. A karbantartási költség csökkenthető lenne, ha az alkalmazás Glade-projektfájljai a Python csomagjaival egy helyen lennének, és futás közben relatív útvonalneveken keresztül importálhatnánk azokat. Az is jó lenne, ha az elemeket közvetlenül valamilyen felhasználói felület objektumpéldányán keresztül érhetnénk el, és nem a `GladeXML.get_widget` segítségével kellene megkeresni őket. Végül hasznos volna, ha az objektum névterében található hívhatók szótárának összeállítását a gépre lehetne bízni, és az eredményt át lehetne adni a `signal_autoconnect`-nek. Ez lehetővé tenné az ügyfelek számára, hogy a jelzéskezelőket az objektum eljárásaiként adják meg, és a kezelőket ne közvetlenül kelljen bejegyezni.
A következő fejezet a GladeBase-modult írja le, amely rendelkezik ezekkel a tulajdonságokkal. A GladeBase a libglade szolgáltatásait tartalmazza, úgy módosítva, hogy azok az MVC (model view controller) tervezőmintához illeszkedjenek (lásd az 1. listát a 15. CD Magazin/Glade könyvtárban). A GladeBase két alapvető exporttal rendelkezik: az `UI` és a `Controller` osztállyal.

GladeBase.UI

A `GladeBase.UI` az MVC tervezőminta *View* összetevőjének felel meg. Ez felelős az elemszerkezetnek a Glade-projektfájl alapján való létrehozásáért, valamint az alkalmazás látható tartalmának frissítéséért egy hozzárendelt vezérlő irányítása alatt. A GladeBase.UI a libglade GladeXML-osztályának leszármazottja, így minden korábban tárgyalt eljárást örököl.
A GladeBase.UI létrehozója (konstruktora) három értéket vesz át: a Glade-projektfájl nevét, amelyből az elemszerkezetet betölti; a gyökérelemként szereplő elem nevét, és egy választható kulcsszót, a `gladeDir`-t. Ez annak a könyvtárnak a relatív útvonalnevét tartalmazza, ahol a Glade-projektfájlokat kell keresni.

2. lista PathFinder.py

```
#!/usr/bin/env python
"""PathFinder.py
Ez a modul feloldja a Python útvonalhoz képest
       relatív útvonalakat."""

import sys, os

class Error(Exception):  """Akkor jön elő, ha
       az útvonalnév nem feloldható"""

def find(pathname):
    """Egy hely útvonalnevét feloldja a
       Python útvonalon."""

    if os.path.isabs(pathname):
        return pathname
    for dirname in sys.path:
        candidate = os.path.join(dirname,
                    pathname)
        if os.path.isfile(candidate):
            return candidate
    raise Error("%s nem található a Python
                útvonalon."
                  % `pathname`)

def main():
    """A modul fősora (egyedülálló végrehajtás
                       esetén)"""
    def testFind(path, expectError=0):
        """Megpróbálja feloldani az útvonalat,
diagnosztikai üzenetekkel."""
        print "%s feloldása..." % path
        try:
            print find(path)
        except Error, info:
            if expectError:
                print "Ahogy vártuk:", info
            else:
                print "HIBA:", info

    # Próbáljunk valami ismert dolgot
    # megtalálni az útvonalon,
    # relatív és abszolút útvonalakat egyaránt
    # használva.
    testFind("socket.py")
    join = os.path.join
    socketPath = join(sys.prefix, join("lib",
        join("python", "socket.py")))
    testFind(socketPath)

    # Próbáljunk egy valószínűleg nem létező
    # dolgot
    # megtalálni.
    testFind("I_Dont_Exist.nonexistent_path", 1)

if __name__ == "__main__":
    main()
```

A gladeDir kulcsszó alapértelmezett értéke a pillanatnyi könyvtár. A fájlnévvel összekapcsolva a Glade-projektfájl relatív elérési útját kapjuk. Furcsának tűnhet a gladeDir és a fájlnév együttes használata, ahelyett hogy a Glade-projektfájl nevét egyszerűen relatív elérési útjával adnánk meg. Ez a szétválasztás azonban csökkentheti az olyan alkalmazások karbantartási költségeit, amelyek egyetlen alcsomagban tartják a Glade-projektfájljait. Az ilyen alkalmazások a GladeBase.UI olyan alosztályát adhatják meg, amelybe a gladeDir értéke be van drótozva.

```
import GladeBase

class UIBase(GladeBase.UI):
  def __init__(self, filename, rootname):
    GladeBase.UI.__init__(self, filename,
      ➥rootname, gladeDir="MyApp/GladeFiles")

class MainWinUI(UIBase):
  def __init__(self):
    UIBase.__init__(self, "main_win.glade",
      ➥"window1")
```

Ezután az alkalmazás minden UI-osztályát ebből az alosztályból származtathatja. Ilyen módon az alkalmazás egy helyen adhatja meg az összes Glade-projektfájl könyvtárának relatív útvonalát.
A PathFinder.py segédmodul lehetővé teszi a GladeBase.UI számára, hogy a Python-útvonalon keressen fájlokat. A `PathFinder.find` függvény egyetlen argumentuma az elérési út. Ha az útvonalnév abszolút, további feldolgozás nélkül visszatér. Ha relatív, akkor a `find` függvénysorban minden egyes Python útvonalbejegyzéssel összekapcsolja, így áll elő a megvizsgálandó útvonalnév. Ha ez létezik, a függvény visszatér. Ha egyik ilyen útvonalnév sem létezik, akkor `PathFinder.Error` kivétel lép fel (lásd a *2. listát*).
A `GladeBase.UI.__getattr__` eljárás teszi lehetővé az ügyfelek számára, hogy a GladeBase.UI-szerkezetben elhelyezkedő elemekhez úgy férjenek hozzá, mintha azok a példány tulajdonságai lennének. A `__getattr__` eljárás feltételezi, hogy a hívó által megadott tulajdonságnév az elem neve, és az elemet a GladeXML.get_widget segítségével kikeresi. Ha az elemet megtalálta, a gyorstárba új példányváltozóként kerül be, ezáltal a későbbiek folyamán gyorsabban lehet elérni. Ha a keresett elem nem található, `__getattr__` `AttributeError` hibajelzést ad vissza. Ha az elemszerkezet egynél több azonos nevű elemet tartalmaz, akkor nem lehet megmondani, hogy a GladeBase.UI melyiket adja vissza. A GladeBase.UI használata során jól tesszük, ha az elemeket úgy nevezzük el, ahogy a Python-példánytulajdonságokat: az objektumon belül minden név legyen egyedi, és érvényes Python-azonosítóval rendelkezzen.
Az alkalmazásfüggő UI-osztályok a GladeBase.UI-osztályt általában olyan eljárásokkal bővítik ki, amelyek összetett frissítéseket végeznek a felhasználói felületen.

GladeBase.Controller

A GladeBase.Controller az MVC *Controller* összetevőjének felel meg. A Controller úgy válaszol a felhasználói bemenet eseményeire, hogy azokat az alkalmazás belső adatmodelljének állapotváltozásaira fordítja le. Hasonlóképpen, az adatmodell állapotváltozásai esetén frissíti a felhasználói felületet.

A GladeBase.Controller nem segít válaszolni az alkalmazás adatmodelljében bekövetkezett változásokra, de a jelzéskezelő eljárásokat önműködően összeköti a Glade-projektfájlban megadott jelzéskezelőkkel. A GladeBase.Controller konstruktora egy értéket vesz át, ez a GladeBase.UI egy példánya és a vezérelendő felhasználói felület. A betöltés során az új GladeBase.Controller-példány átfut (traverses) az osztály szerkezetén, és a példány névterében található összes hívható objektumból felépíti a szótárt (az átfutás – traversal – a példány szótárával kezdődik, amennyiben példánytulajdonságként hívhatókat adtak meg).
Ezután a GladeBase.Controller ezt a szótárt a megadott GladeBase.UI-példány signal_autoconnect eljárásának adja át. Az alkalmazásfüggő vezérlőosztályok a GladeBase.Controller-osztályt terjesztik ki a jelzéskezelő eljárások megadásával.

```
class Controller(GladeBase.Controller):

    def __init__(self, ui):
        ...
        GladeBase.Controller.__init__(self, ui)

    def on_window1_delete_event(self, *args):
        gtk.mainquit()

    def on_button1_clicked(self, *args):
        print "1. gombot megnyomták."
```

Vezérlővázak létrehozása

A GladeBase önműködővé teszi a Gtk+ elemszerkezetek Python objektumszerkezetekké alakítását, és önműködően összekapcsolja a Python jelzéskezelőivel, de még mindig megkívánja, hogy azonosítsuk és megvalósítsuk az összes Glade-projektfájlban megadott jelzéskezelőt. A tiszta Gtk+ projekteknél ez nem gond, hiszen csak azok a jelzéskezelők léteznek, amelyeket kifejezetten megadtunk. Ellenben, ha a Glade-et Gnome-alkalmazás készítésére használjuk, sok jelzéskezelő önműködően megjelenik. Például egy új Gnome-alkalmazásablak szabványos menüsorral jön létre, és ezek a menüelemek előre megadott jelzéskezelőket tartalmaznak. Nagyon unalmas lenne egy Gnome-alapú projektet átböngészni, az előre megadott jelzéskezelőket kézzel megkeresni, és ezeket az alkalmazás vezérlőihez hozzáadni. Ahogy korábban megjegyeztük, a Glade-projektfájlok XML-formátumban vannak (pillanatnyilag nincs még DTD, amely leírná a projektfájl szerkezetét, de e nélkül is könnyű megérteni). A Python 2.0 tartalmaz egy XML-programkönyvtárat, amely *James Clark* Expat könyvtára fölé rétegződik. Ezért meglehetősen egyszerű olyan Python-alkalmazást írni, amely végigszalad a Glade-projektfájlon, azonosítja az adott elemszerkezetekben hivatkozott jelzéskezelőket, és a Controller-modul vázát ehhez a szerkezethez elkészíti.
A `GladeProjectSignals.py` (a 3. listát lásd a 15. CD Magazin/Glade könyvtárban) a Glade-projektfájlból gyűjti ki a jelzéskezelők adatait. A `WidgetTreeSignals` osztály felépíti az elemszerkezetet képviselő `XML DOM` (Document Object Model) -fát és minden, a jelzéskezelőkre vonatkozó hivatkozást feljegyez. A GladeProjectSignals-osztály betölti a Glade-projektfájlt, és a WidgetTreeSignal példányokból az összes legfelső szintű elemhez felépít egy szótárat.
A WidgetTreeSignals létrehozójának argumentuma egy DOM-csomópont. Feltételezi, hogy ez a csomópont egy elemet ír le, és elvárja, hogy az elem nevét megadó névcsomópontot tartalmazza. Ha ezek a feltételek teljesülnek, a WidgetTreeSignals feljegyzi a csomópont kezelőgyermekének értékét, amely nem más, mint a jelzéskezelő neve. Egyébként a WidgetTreeSignals feltételezi, hogy a csomópont gyermekcsomópontokat tartalmaz, és ezekkel folytatja az olvasást.
A `GladeProjectSignals` hasonlóképpen egyszerű. A Python a Glade-projektfájlt `xml.dom.minidom` csomagjának segítségével betölti a DOM-fába. Ezután a fában megkeresi a legfelső szintű elemeket (a Glade-tervezőfájl más legfelső szintű csomópontokat is tartalmaz, például a GTK-felület és projekt csomópontjait).
A GladeProjectSignals minden megtalált elemcsomóponthoz egy új WidgetTreeSignals-példányt hoz létre, amely az elem és leszármazottai által meghatározott jelzéskezelőket sorolja fel. Minden egyes WidgetTreeSignal-példány a legfelső szintű elem nevével megjelölt *self.widgets* nevű szótárba kerül.
Ha a `ControllerGenerator.py`-nek (lásd a 4. listát lásd a 15. CD Magazin/Glade könyvtárban) átadunk egy Glade-projektfájlnevet és a fájlban megadott egyik legfelső szintű elem nevét, akkor kinyomtatja az elemhez és gyermekeihez tartozó Controller vázát. A munka oroszlánrészét a `ControllerGenerator` osztály végzi. Ez az osztály tartalmazza a `generate` eljárást, amely a Glade-projektfájl nevét és a legfelső szintű elem nevét veszi át. A generate eljárás a szóban forgó elem jelzéskezelőihez a GladeProjectSignals egy példányán keresztül jut hozzá. Ezt követően elkészíti ezeknek a kezelőknek a vázát. Egy mintakarakterlánc és a Python karakterlánc-formázó operátorainak segítségével a generate létrehozza a Controller modul vázát alkotó karakterláncot, és visszaadja azt a hívónak.

Összegzés

A Glade, a libglade és a gnome-python nagymértékben képes csökkenteni a Python nyelven írt Gtk+ és a Gnome-alkalmazások fejlesztéséhez szükséges munkát. A cikkben bemutatott eszközök a Glade-elemszerkezetek Python objektumszerkezetekké történő alakításával, a vezérlőkben megadott jelzéskezelők összekapcsolásával és a vezérlővázak létrehozásával a karbantartás költségét tovább csökkentik.

Ajánlott irodalom

Design Patterns: Elements of Reusable Object-Oriented Software, írta Gamma, Helm, Johnson és Vlissides (Addison-Wesley, 1994). Leírja az Observer mintát, idézi a Smalltalk Model-View-Controller keretrendszerét mint a minta egy korai példáját.

Mitch Chapman (chapman@bioreason.com) vezető programmérnökként dolgozik a Bioreasonnél. Az új-mexikói Santa Fében él, ahol csaknem egyidejűleg hódolhat a Python-programozás, a hódeszkázás, a sziklamászás, valamint a Napba nézés és a repülés örömeinek.

Kapcsolódó címek

A Glade projekt honlapja
➲ http://glade.pn.org/
A PyGTK burkolókönyvtár honlapja
➲ http://www.daa.com.au/~james/pygtk
A libglade programozói leírása
➲ http://developer.gnome.org/doc/API/libglade/libglade.html
A Gtk+ projekt honlapja ➲ http://www.gtk.org/
Glade tananyag
➲ http://linuxfocus.org/English/July2000/article160.shtml

A DreamWorks és a Linux

A Linux kedvező tulajdonságai a számítógépes animációk készítésének folyamatában.

Kétszáznál is több linuxos asztali és négyszáz szintén linuxos kiszolgálógépével a DreamWorks SGI nemcsak a számítógépes mozifilm-animációk fő gyártója, de az egyik legnagyobb Linux-felhasználó is. A cég az animációk létrehozásánál három gyártócsatorna: a bristoli Aardman (Csibefutam), a Palo Altó-i PDI/DreamWorks (Shrek; Z, a hangya) és a kaliforniai Glendale-ben székelő eredeti DreamWorks (Irány Eldorádó, Egyiptom hercege) tapasztalatait egyesíti. Mindegyik gyártóegység önálló animációs módszerrel bír: az Aardmant az agyagtechnológia (claymation) jellemzi, a PDI/DreamWorks a számítógépes grafikáról, a glendale-i egység pedig a hagyományos gyártástechnológiáról ismerhető fel.

A glendale-i stúdióba látogattunk el, ahol jelenleg épp egy nagyszabású mozifilm, a *Spirit, Stallion of the Cimarron* munkálatai folynak. A történet – amelynek amerikai bemutatója 2002 közepére várható – egy musztáng vadnyugati kalandjait tárja majd a nézők szeme elé.

Sokan kételkednek abban, hogy a Linux az asztali számítógépeken háttérbe szoríthatja a Windowst, hiszen nem használhatjuk a legelterjedtebb alkalmazásokat, mint például a Microsoft Office (Word, Excel, Access). Ámde a mozifilm-grafikusok által használt alkalmazások túlnyomó részének linuxos változata mára már hozzáférhető, s a Linuxra átírt vagy közvetlenül a Linux alá fejlesztett programok száma is figyelemre méltó arányokat kezd ölteni.

A DreamWorks három úton közelített a Linux felé: új programok fejlesztésével, saját programok átültetésével és valamely külső cég programjának illesztésével. *Ed Leonard* technikai vezető így beszél erről: „Animátoraink Linuxra történő átállításának egyik legfontosabb motivációs tényezője a költségek jelentős mérséklése volt. Mégsem ez számított a leginkább, hanem munkájuk hatékonyságának javulása. Ami minket igazából érdekel, az nem a háttérben lévő gyártási módszer, hanem a létrehozott alkotás minősége.” A Linux használatával a grafikusok időt takaríthatnak meg, mivel egy linuxos PC annyival gyorsabb, mintha egy ötéves gépet cserélnénk újra, habár meg kell hagyni, azok az SGI IRIX munkaállomások bámulatra méltó gépek voltak. Leonard még hozzáteszi: „A Microsoft programjai továbbra is kulcsszerepet játszanak a vállalat üzleti életében, de a Linux rendkívül hatékonyan alkalmazható az animációkészítés folyamatában.”

Az animátorok számítógépei nagymértékben különböznek azoktól a PC-től, mint amelyeket egy tisztviselő vagy titkárnő asztalán találhatunk. Az animációk elkészítéséhez villámgyors munkaállomások szükségesek – nagy teljesítményű kétmonitoros grafikus rendszerrel kiegészítve, és kifejezetten mozifilmgyártásra kihegyezett programokkal felszerelve. Most vegyük szemügyre a DreamWorks gyártási folyamatát és nézzük, milyen szerepet játszik ebben a Linux (lásd *táblázat*)!

Egy animációkkal dolgozó film készítésének első lépése a történet kidolgozása, a teljes gyártási folyamat pedig két évet vesz igénybe. Az előkészítési szakasznak – amelyet vizuális tervezésnek (vizdev) neveznek – az a célja, hogy az alkotás lényegi formáját, látványát sikerüljön felvázolni. Ezt a legkülönbözőbb művészi ábrázolási módszerek (például olajfestészet) segítségével érik el. Sokszor ezen előzetes alkotások részletessége, valósághűsége messze meghaladja azt az általános színvonalat, ami majd a kész alkotásban megjelenik.

Elkészül egy kézzel megrajzolt történeti vázlat is, amely leginkább egy képregényhez hasonlatos, és az a szerepe, hogy a film legfontosabb jeleneteit láthatóvá, elképzelhetővé tegye, ennek alapján hozzák létre a művészek az animációt.

Ehhez az Alias/Wavefront cég 3D-s animációkat készítő programcsomagjának, a Mayának bizonyos bővítményeit használják fel. Bár ennek az animációnak a minősége elmarad a kész filmétől, jól láthatóvá teszi a jelenetek összefüggéseit és a kameranézeteket, segíti a szereplők megtervezését. Ezt alkalmazva a filmkészítők mozgásában tekinthetik át a filmet, mielőtt még a gyártás megkezdődne. Ennek az anyagnak egyetlen részlete sem kerül bele az elkészült filmbe, kizárólag arra használatos, hogy útmutatást adjon a későbbi munkához.

A látványtervezés e szakasza határozza meg a film szereplőit, a háttereket, illetve a használt hatásokat. Az animációkkal, a hátterekkel és a hatásokkal külön részleg foglalkozik. Ezek a különálló részek a gyártás egy későbbi szakaszában egy különleges program segítségével válnak egységes egésszé. A szereplők mozgásának tervezésére a ToonShooter programmal ütemvázlatot készítenek. „A ToonShooter olyan Linux-alkalmazás, amit mi fejlesztettünk. Azoknak a 640x480-as alacsony felbontású vonalvázlatoknak a rögzítésére készítettük, amelyeket az animátorok később a film időzítéséhez használnak fel” – magyarázza *Derek Chan* vezető alkalmazásfejlesztő. „A program elkészítése óta eltelt egy év alatt a programot három animációs műhelyben helyeztük üzembe. Égető szükség volt erre a Linux-alkalmazásra, s nekünk határidő előtt sikerült befejeznünk a munkát. A DreamWorks jelenleg 60 példányban használja a programot” – teszi hozzá Chan. A ToonSketch grafikusaink számára lehetővé teszi, hogy a gyors, másolás jellegű munkákat papír és lapolvasó nélkül végezzék el. Ahogy *Nhi Casey* programmérnök bemutatja: „A vázlat egy Wacom-táblán készül el, amely ugyanúgy használható, mint egy darab papír, de gyorsabb és korlátlan visszalépést tesz lehetővé.” *Jim Mainard*, a programrészleg vezetője hozzáfűzi: „Láthatóvá tehetők a megelőző és a pillanatnyit követő képek, így a munka sokkal könnyebbé válik az animátorok számára, mintha

papírkötegekkel kellene birkózniuk.” Ezzel a szolgáltatással nyomon követhetők a szomszédos képek, hatékony útmutatást nyújtva az animátorok számára.

Ahhoz, hogy a hagyományos animációs módszerekkel készülő filmek látványát megőrizzék, az előtérben lévő szereplők rajzai kézzel készülnek el a DreamWorks műhelyében. A digitalizáló részleg nagyméretű önműködő lapolvasókat használ a kézzel rajzolt képek nagy felbontású (2 k pixel formátumú) rögzítésére. Az Egyiptom hercege című film készítése során több mint egymillió papírlap kezelésére volt szükség.

A képek saját tömörített, raszteres formátumba való mentéséről a ScanLevel elnevezésű IRIX-alapú program gondoskodik. Mainard szerint a program Linuxra való átültetését még ebben az évben végrehajtják, mihelyt elkészül a lapolvasóhoz a megfelelő meghajtóprogram.

A ProcessLevels nevű linuxos eszközgyűjtemény segítségével a háttérrajzot sötétítve vagy világosítva finomítják a kép kontrasztját, eltüntetve a papír színét. A ferdeségeket és az optikai torzításokat a rendszer önműködően eltávolítja. Az InkAndPainttel válnak színessé a digitalizált vonalrajzok. *Tina Staples*, a cég kitűnő festő-grafikusa mutatja be, mennyire jól tudja használni ezt a saját Linux-alkalmazást.

A program hasonlít a Gimpre és a Photoshopra, azzal a különbséggel, hogy működése a kitöltő szolgáltatásokra van kihegyezve. Leonard azzal viccelődik, hogy olyan az egész, mintha atomokat szeretnénk egyenként kifesteni. Tina olyan gyorsan fest ki egy képkockát, hogy arra kellett kérnünk, lassítson kicsit, így legalább követni tudjuk, mit is csinál. Tina nagyon lelkes: „Sokkal gyorsabbak lettünk.

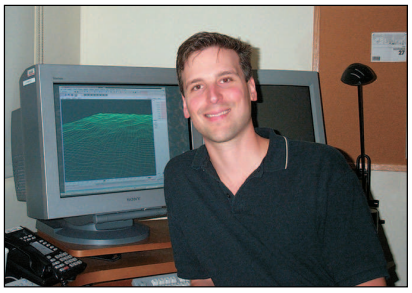
A Linux használatával a munkám sebessége kétszeresére nőtt, így minden héten sikerül elvégezni a tervezett mennyiséget.”

Az InkAndPaint képes automatikusan kifesteni tartományokat, továbbá utómunka-eszközökkel és a közbülső kép önműködő létrehozásának lehetőségével rendelkezik. „Az automatizmus a jelenet előrehaladtával egyre kevésbé működik” – mondja Mainard. „A szakszerű emberi beavatkozás nélkülözhetetlen azoknak a területeknek a kiszínezésénél, ahol a vonalak metszik egymást. Amennyiben a feladatot nem megfelelően hajtják végre, az „bizsergést” eredményez a képen – annak következtében, hogy a vonalak találkozásánál a színek képről-képre változnak.

A ToonShooter képernyőképe

Nhi Casey vezető programmérnök

Tina Staples, a villámkezű festő-grafikus



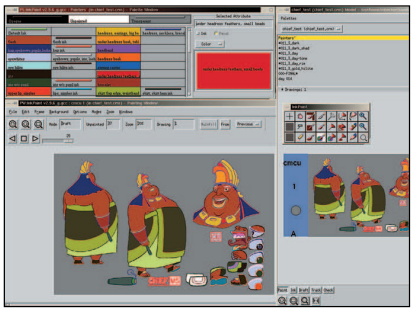
Galen Growicz, az Advanced R&D Future Films vezető programmérnöke

A vonalak találkozásánál nehéz önműködővé tenni a színkitöltést, de talán sikerül továbbfejlesztenünk a szolgáltatást" – bizakodik Mainard. „Bár Tina annyira gyors az InkAndPainttel, hogy ennek szinte nincs is jelentősége."
Az InkAndPaint a kétmonitoros Octane munkaállomásokat linuxos PC-kel váltotta fel.
A legtöbb háttérgrafikát kétdimenziós formában, kézzel készítik el a grafikusok – vagy valamilyen hagyományos eszközt (például olajfestés), vagy számítógépes programot (például Photoshop) véve ehhez igénybe.

Az SGI renderfarm

A Linux leképezőtelep

A háttérrajzok igen nagy méretűek is lehetnek. Mainard elmondása szerint az Egyiptom hercege némely hátterének szélessége a 2–3 métert is elérte. A ScanBackground IRIX-alapú alkalmazás eljárásai teszik lehetővé, hogy a 36 hüvelykes Tangent lapolvasóval beolvasott képrészletekből az eredeti hátteret újra összeillesszék.
A változó látószögű hátterek három dimenzióban készülnek el a Maya segítségével. A színek mintázatvetítéssel vagy festéssel születnek meg. A 3D-festéshez korábban az Alias/Wavefront Studio Paintjét használták, de éppen most folyik az átállás a wpaint nevű saját fejlesztésű Linux-alkalmazásra.
A különleges hatásokkal foglalkozó részleg készíti el az olyan egységes hatásokat, mint a por, tűz, folyadékok és füst.
Az Advanced R&D Future Films vezető



Az InkAndPaint képernyőképe

programmérnöke, *Galen Gornowicz* egy éve dolgozik egy még be nem jelentett film szakmai hátterén. „Olyan tengeri jeleneten dolgozom, amelynek képszámítása valós időben hajtódik végre. Vannak olyan zárt rendszerek, amelyek tudják ezt, de mi a Mayához készítünk kiegészítést" – mondja Gorowicz. „A régi rendszernél ebédelni mehettünk, amíg a gép ehhez hasonló dolgot számolt. Ráadásul a vízmintákat parancssorból kellett futtatni, nem így interaktívan, mint most. A hatások készítéséhez a Mayát használjuk egy sor olyan kiegészítéssel, mint amilyen például a Linuxra írt Calypso water." A DreamWorks úgy találta, hogy a Calypso által létrehozott víz *túl* valószerű, az eredmény túlságosan hasonlít a Viharzóna-belire. Gorowicz most azon dolgozik, hogy a víz-hatást a film többi képének megjelenésébe beillessze. Egy Maya-bővítmény fejlesztése során az egyszerű hatásokhoz MEL-héjprogramot (a Java parancsfájlhoz hasonlatos nyelv) kell írni, a bonyolultabbak vagy számításigényesebbek, vagy pedig C/C++ kódot igényelnek. Amikor már lefordított bővítményt futtatunk, egy futásidőben értelmezett MEL-parancsfájl végzi el a menü létrehozását és Mayához való csatolását. A MEL-parancsfájllal bármilyen függvény meghívható. A bővítmény nem rendelkezik `main()` függvénnyel.
A leképezés, vagyis egy számítógép által előállított objektum képének megalkotása folyhat a felhasználó saját munkaállomásán, vagy a DreamWorks leképezőtelepén (renderfarm). A leképezőtelep – három hűtőkamion méretű, 8–16 processzoros SGI Origin 2000 kiszolgálócsoport – a feladatokat sorban vagy elsőségi elv alapján is végezheti. A PDI/DreamWorksnél a Shrekhez használt leképezőtelep egy több mint ezerprocesszoros – 80 százalék Linuxot, húsz százalék IRIX-ot futtató – rendszer volt. A glendale-i telepen teljes egészében Linuxot használnak. Még megvannak az IRIX-ot futtató Origin 2000 kiszolgálók, de folyik az átállás a sokkal olcsóbb pentiumos rendszerre. A glendale-i leképezőtelep feleakkora, mint a PDI/DreamWorksé, mivel a hagyományos animációs módszer kevésbé számításigényes, mint a számító-

Animációtípusok és eljárások

Feladat	*Leírás*
Elhelyezkedési vázlat	A helyzettervező grafikus felvázolja a jelenetek szereplőinek elhelyezkedését, jelzi az elrendezést, a kameraállásokat és a kamera mozgását. A teljes jelenethez egy animáció készül.
Animációvázlat	Az animátorok által készített durva vázlat, amely a jelenet szereplőinek mozgását mutatja.
CG animáció	Számítógép által létrehozott objektumok modellezése és mozgatása.
Durva jelenetterv	A jelenet digitális megvalósítása a szükséges objektumokkal.
CU elrendezés	A helyzettervezők végső formára hozzák az elhelyezkedési vázlatot, jelzik a megvilágítást és felvázolják a hátteret, melynek végső, színes formátumát a háttérrészleg készíti el.
Hátterek	Elkészülnek a hagyományos és a digitálisan létrehozott képek.
Az animáció letisztázása	A szereplők elhelyezése, az utolsó simítások elvégzése, az összes részlet és beiktatott rajz beépítése.
CU átvizsgálás	A letisztázott animáció átvizsgálása.
Digitális ellenőrzés	A szereplői szintek digitális sötétítése, kiemelése, a kifestésre történő előkészítés.
Kijelölés	Jelek rajzolása az InkAndPaint számára, tekintettel a használandó színmodellre, a vonalak és tartományok megfelelő kezelésére.
Színezés	A szintek digitális kifestése a meghatározott színmodell és a határoló jelek alapján.
Megvilágítás/leképezés	A CG-animáció végső szín- és megvilágításszámítása nagy felbontásban (legalább 2k) leképezve.
Végső ellenőrzés	A jelenet részeinek – kamera, színek, kompozíció – végső ellenőrzése.
Filmre írás	A képkockák végső formájának filmre rögzítése.

gépes grafika. A VA Linux biztosította az első glendale-i linuxos leképező gépet, amely bizonyította az Intel PC-k képességeit. A DreamWorks most egy új leképezőtorony építésébe fogott, amely két 1 GHz-es Pentium III processzort és 2 GB RAM-ot tartalmazó számítógépből fog állni. A hűtőszekrény nagyságú gép fogja helyettesíteni a korábbi, 10–15 méter hosszú állványzatot elfoglaló rendszert.

A kompozíciós részleg feladata, hogy az előbb felvázolt folyamatok során előálló részeket, a szereplőket, a háttereket, a hatásokat egységgé olvassza össze. Itt történik az előtérben álló objektumok elhomályosítása, amikor például a szereplő egy szikla mögül látszik. „A DreamWorks az eredetileg a Cambridge Animation által írt Director nagymértékben módosított változatát használja. A program a Shake-hez, a népszerű kompozíciós programcsomaghoz hasonlít, de a változtatások után képes kezelni az X-lapokat" – magyarázza Mainard. Az X-lapok az animátorok hagyományosan kedvelt munkaeszközei, a film egyes kockáihoz tartozó adatokat tároló listák. A DreamWorks mindegyik munkaeszközét úgy módosították, hogy támogassák ezek használatát. A Director is rendelkezik egy egyedi szolgáltatással: az InkAndPaint-szintek megfelelő felbontású és élsimított leképezésével.

A rajzolók legtöbbször negyedfelbontású 1 k széles képekkel dolgoznak. A végső leképezés 2 k, illetve az IMAX-kiadás 3 k és 4 k formátumú. A végső leképezés eredménye egy TIFF fájl, amelyet a kimeneti kiszolgáló az egyes sorszámú Cineon Lightening filmrögzítőnek továbbít. A lézeres filmrögzítő körülbelül 3 kép/mp sebességgel, 16 bit csatornánkénti felbontással 2 k-s képkockákat rögzít. „Amikor a sorrend elkészül, az adatokat Perl parancsfájlgyűjtemény segítségével továbbítjuk a filmrögzítőnek. Így egy színes mozgófilm körülbelül napi gyakorisággal készül. (…) A Linux remekül működik a játékok fejlesztése során, de az SGI gazdag grafikus API-gyűjteményének átültetése meglehetősen időigényes. A HP, az IBM és az SGI kiváló linuxos megoldásokat szállított számunkra" – tájékoztat Leonard. Szerinte a Dell is kezdi „venni a lapot" és a jövőben szélesebbre tárja az ablakot a Linux felé. A kérdésre, hogy miért nem találkozunk AMD processzorokkal, Leonard így felel: „Az Intel-Linux felület erős és kiegyensúlyozott megoldást kínál a forgalmazók számára a felső kategóriájú munkaállomások terén. Emiatt nem erőltetjük a Linux+Alpha, a FreeBSD vagy az egyéb saját unixos megoldásokat." Majd hozzáteszi: „A HP nagyszerű Linux-támogatással rendelkezik. Előfordult, hogy napi két hibajavítást is kaptunk tőle." A korábbi szerződések ápolgatása helyett a DreamWorks most a költségkímélőbb PC-vásárlás mellett döntött. Leonard szerint ez az „eldobható" megoldás lehetővé teszi, hogy az asztali gépeket és a leképezőtelepet kétévenként cseréljék le – az eddigi öt év helyett.

A DreamWorks egyaránt használ belső fejlesztésű, az adott feladat igényeihez igazított programokat, valamint a piacon beszerezhető animációs programokat. Mára mintegy hárommillió programsornyi saját kóddal rendelkeznek, amelynek nagy része az SGI IRIX operációs rendszerre készült. Mivel a Linux sokkal jobban hasonlít az IRIX-ra, mint a Windows, e nagy mennyiségű kód átültetése is egyszerűbb így. A Linuxon futó Maya (Linuxvilág, 2001. június-július, 108–111. oldal) az egyik olyan kereskedelmi programcsomag, ami igen nagy szerepet játszik a cég alkotásaiban.

Mindhárom DreamWorks gyártócsatorna (Aardman, PDI/DreamWorks, DreamWorks traditional) megfelelő képességekkel rendelkezik ahhoz, hogy 18–24 hónaponként egy mozifilmet hozzon létre. A Linux gazdaságosságának köszönhetően most egy negyedik csatorna is születőben van Glendale-ben. A termelés jövő év eleji indulásával egy teljesen Linuxra épülő, Intel IA-64 és Pentium IV-es processzorokkal felszerelt műhely kezdi meg munkáját. Kíváncsian várjuk, hogy a Shrek és a Spirit után, a Linux lehetőségeit kihasználva milyen még izgalmasabb alkotások hagyják el a DreamWorks SKG műhelyeit.

(A Shrek és a munkatársak összes képét a DreamWorks biztosította, minden jog fenntartva, 2001.)

Robin Rowe a MovieEditor.com internetes és televíziós videoalkalmazásokat készítő cég egyik partnere. Írásai a Dr. Dobb's Journalban, a C++ Reportban, a C/C++ Users Journalban, a Data Based Advisorban jelentek meg és számos tanácskozás anyagában megtalálhatók. A Robin által készített programok sorában található többek közt az a kiszolgálóalapú videoszerkesztő rendszer, amit a Manhattan 24 órás televíziós hírcsatorna, a Time Warner New York One, illetve a kapcsolódó honlap ➲ http://www.ny1.com/ is használ. Elérhető a robin.rowe@ movieeditor.com címen.

A PostgreSQL és a teljesítménynövelés

Pörgessük fel a gépet, hogy a legtöbbet hozhassuk ki e nyílt forráskódú programból.

A PostgreSQL objektumközpontú adatbázis-kezelő program, amelyet a Föld minden csücskéből származó fejlesztők készítenek az Interneten keresztül, és tulajdonképpen az Oracle, illetve az Informix kereskedelmi alkalmazások nyílt forrású megfelelője.
A PostgreSQL-t eredetileg a Berkeley-n (University of California) készítették. Az egyetem csapata 1996-ban kezdte meg az Interneten keresztüli fejlesztést: az ötletek cseréjére elektronikus leveleket, a kódfájlok átviteléhez pedig fájlkiszolgálókat használtak. Mára a PostgreSQL képességei teljesítmény és megbízhatóság tekintetében felveszik a versenyt a kereskedelmi alkalmazásokkal. Rendelkezik tranzakciókezeléssel, nézetekkel (views), tárolt függvényekkel és hivatkozási épségkezeléssel (referential integrity constraints) egyaránt. Nagyszámú programozási felületet támogat, többek között az ODBC-t, a Javát (JDBC), a Tcl/Tk-t, a PHP-t, a Perlt és a Pythont. Hála a tehetséges internetes fejlesztőgárdának, a PostgreSQL folyamatosan és hihetetlen ütemben fejlődik.

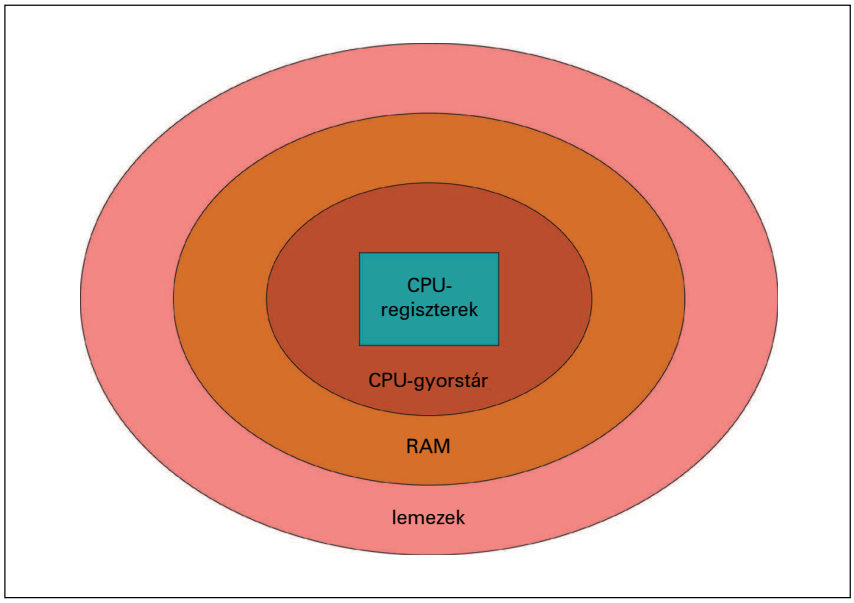
Teljesítményalapfogalmak

Az adatbázis-teljesítmény növelése kétféle módon történhet: az egyik út, hogy a processzor-, a memória- és a merevlemez-terhelést növeljük. A másik lehetőség az adatbázisnak küldött lekérdezések egyszerűsítése. Írásukban a teljesítménynövelés vas felőli oldalát mutatjuk be. A lekérdezések hatékonyabbá tételének eszközei: a `CREATE INDEX`, `VACUUM`, `VACUUM ANALYZE`, `CLUSTER`, `EXPLAIN` és hasonló SQL-parancsok. Ezek bővebb taglalásával könyvemben foglalkozom (PostgreSQL: Introduction and Concepts, elérhető a ➲ http://www.postgresql.org/docs/awbook.html címen is).
A vas teljesítményére vonatkozó dolgok megértéséhez fontos, hogy pontosan lássuk, mi is történik a gép belsejében. Az egyszerűség kedvéért a számítógépre most úgy gondoljunk, mint tárolók körülvette központi egységre (CPU). A processzorral egy lapkán található néhány CPU-regiszter, ezekben számításközi adatok, mutatók és számlálók tárolhatók. Mellettük helyezkedik el a processzor-gyorstár, amely a legutóbb elért adatokat tartalmazza. A gyorstáron kívül még itt található a nagy mennyiségű, véletlen elérésű főmemória (RAM) is, amely a végrehajtandó programot és az adatokat tartalmazza. E főmemória mögött helyezkednek el a merevlemezek, amelyek még nagyobb adatmennyiség tárolására képesek. Egyedül ezek a merevlemezek tekinthetők állandó tárolóhelynek, így mindent, amit meg kell tartani, ide kell helyezni, mielőtt a gépet kikapcsoljuk (lásd a *táblázatot*).
A központi egységet körülvevő tárolóeszközöket az alábbi *ábrán* mutatjuk be.

Hogyan tartsuk a processzor mellett az adatokat?

Az adatok áramlása a különböző tárolóeszközök között önműködően zajlik. A fordítóprogramok meghatározzák, mely adatokat szükséges regiszterekben tárolni. A processzor logikája a gyakran használt adatokat a CPU gyorstárban tartja. Az operációs rendszer pedig meghatározza, melyik adatot kell a RAM-ban tartani és cserélgeti a merevlemezen tároltakkal.
A CPU regisztereket és a processzor-

Jelenleg a PostgreSQL képességei teljesítmény és megbízhatóság tekintetében felveszik a versenyt a kereskedelmi alkalmazásokkal.



Tárolóterületek

Láthatjuk, hogy a processzortól távolodva a tárolóméretek egyre növekednek.
Az eszményi az lenne, ha hatalmas mennyiségű állandó memóriát helyeznénk közvetlenül a központi egység mellé, csakhogy ez igencsak költséges megoldásnak bizonyulna. A gyakorlatban a processzorhoz közel a leggyakrabban használt adatokat tároljuk, a kevéssé használtakat pedig távolabb, és akkor hozzuk közelebb, ha szükség van rájuk.

A tárterületek típusai

Tárterület	*Egység*
CPU regiszterek	bájt
CPU gyorstár	kilobájt
RAM	megabájt
lemezmeghajtók	gigabájt

gyorstárat adatbázis-felügyelőként nemigen lehet hatékonyan egyszerűsíteni. A hatékony adatbázis-gyorsítás a minél nagyobb mennyiségű hasznos adat RAM-ba töltését jelenti, mivel így, ahol csak lehet, megelőzzük a lemezelérést.

Azt hihetnénk, hogy ez könnyű feladat, pedig nem az. A számítógép memóriája számos dolgot tartalmaz, beleértve a végrehajtható programot, a programadatokat és a vermet, valamint a PostgreSQL megosztott és a rendszermag gyorstárát. A helyes egyszerűsítés azt jelenti, hogy annyi adatot tartunk a memóriában, amennyit csak lehetséges, de csak addig, amíg ez nem befolyásolja hátrányosan az operációs rendszer más részeit.

A PostgreSQL megosztott gyorstára

A PostgreSQL az adatokat közvetlenül nem változtatja meg a merevlemezen, hanem megosztott gyorstárába olvastatja be. A PostgreSQL-motor ezután ide írja, illetve innen olvassa a blokkokat, amelyek végül visszaíródnak a merevlemezre. Az a motor, amelynek valamelyik táblát el kell érnie, a szükséges blokkot először ebben a gyorstárban keresi, és ha már úgyis itt van, azon nyomban folytathatja a feldolgozást. Amennyiben a blokkot nem találja, operációs rendszerhívást hajt végre, hogy betöltődjön. A blokkok vagy a rendszermag gyorstárából, vagy a merevlemezről töltődnek be. Ez azonban időigényes művelet lehet.

Az alapértelmezett PostgreSQL-beállítás 64 megosztott gyorstárat foglal le, melyek egyenként nyolc kilobájt méretűek. A gyorstárak számának növelésével egyre valószínűbbé válik, hogy a motor a keresett adatot megtalálja a gyorstárban, így takarítva meg az időigényes rendszerhívást.

Mekkora számít túl nagynak?

Úgy vélhetnénk, jó elgondolás az összes memóriát a PostgreSQL megosztott gyorstárának adni. Csakhogy ha ezt tesszük, nem marad elég hely a rendszermag vagy a többi program számára. A PostgreSQL megosztott gyorstárának megfelelő mérete az a legnagyobb hasznos méret, amely az egyéb tevékenységeket még nem befolyásolja hátrányosan.

Ha meg akarjuk érteni a káros folyamatokat, előbb azt kell megértenünk, miképpen kezeli a Unix operációs rendszer a memóriát. Ha elég memória áll rendelkezésre ahhoz, hogy az összes program és adat a memóriában maradhasson, nincs gond. Ha azonban a memóriában már nem fér el minden, a rendszermag a memórialapokat egy lemezterületre kezdi el átirányítani, ezt csereterületnek (swap) nevezzük. A végén használt lapokat kirakjuk (kilapozzuk) a csereterületre. Ezt a műveletet *lapkiolvasásnak* (swap pageout) nevezzük. A lapkidobások nem okoznak bonyodalmat, mivel inaktív időszakban hajtódnak végre. Az viszont nehézséget okoz, ha ezeket a lapokat valamikor vissza kell hozni a csereterületről, azaz a régi lapot, amit már egyszer kimozgattunk a csereterületre, vissza kell másolni a memóriába. Ezt lapbeolvasásnak (swap pagein) nevezzük. Ez kedvezőtlen folyamat, mert amíg a lapbeolvasás a csereterületről véget nem ér, addig a programfutás szünetel.

A lapbeolvasást a vmstat, a sar vagy hasonló eszközökkel figyelhetjük meg, amelyek jelzik, ha a hatékony működéshez már nincs elegendő memória. A cseretár-lapbeolvasásokat ne tévesszük össze a hagyományos lapbeolvasásokkal, amelyekbe a fájlrendszerből olvasó normál lemezműveletek is beletartozhatnak. Amennyiben a csereterület-lapbeolvasásokat nem találjuk, úgy a gyakori lapozások jól mutatják, hogy egyúttal lapbeolvasások is zajlanak.

A gyorstár méretének hatásai

Valószínűleg sokan elcsodálkoznak azon, miért ilyen központi kérdés a gyorstár mérete. Először is képzeljük el, hogy a PostgreSQL megosztott gyorstára elég nagy ahhoz, hogy a teljes táblát tárolni tudja. A táblára irányuló ismételt szekvenciális keresések egyáltalán nem igényelnek lemezelérést, mivel minden adat eleve a gyorstárban helyezkedik el. Most gondoljunk arra, hogy a gyorstár egy blokkal kisebb, mint a tábla. A tábla szekvenciális keresése esetén minden táblablokk a gyorstárba fog töltődni, kivéve az utolsót. Amikor erre a blokkra lesz szükség, a legrégebben betöltött blokk eltávolítódik, mely jelen esetben a tábla legelső blokkja. Amikor a következő szekvenciális keresésre kerül a sor, az első blokk már nincs a tárban, és a betöltéshez ismét a legrégebbi blokkot kell eltávolítani, ez pedig jelen esetben már a tábla második blokkja. És így tovább: az éppen beolvasott blokk mindig kiüti a következő szükségeset, egészen a tábla végéig. Ez természetesen nagyon végletes példa, de jól látható, hogy egyetlen blokk hiánya a gyorstár hatékonyságát száz százalékról nullára csökkentette. Ez azt mutatja, hogy a helyes gyorstárméret megválasztása nagymértékben hat a teljesítményre.

A megosztott gyorstár méretének helyes beállítása

Eszményi esetben a PostgreSQL megosztott gyorstára elég nagy ahhoz, hogy a legtöbbet használt táblákat tárolni tudja, és elég kicsi ahhoz, hogy még ne használjon csereterületet. Ne feledjük, hogy a *Postmaster* (a Postgres főprogram) már indításkor lefoglalja az összes megosztott memóriát. Ez a terület mindig ekkora marad, még akkor is, ha az adatbázist senki sem használja. Néhány operációs rendszer dobja azokat a megosztott memórialapokat, amelyekre semmi sem hivatkozik, míg mások a RAM-ban hagyják. A PostgreSQL 7.2 Administrator's Guide-ban fontos adatokra bukkanhatunk a különféle operációs rendszerekhez tartozó rendszermag-beállításokról: ➲ http://www.postgresql.org/devel-corner/docs/admin/kernel-resources.html.

A rendező memória-köteg (batch) mérete

Egy további szabályozható érték a rendező kötegfájlokhoz használt memória mennyisége. Amikor a PostgreSQL nagyméretű táblákat vagy eredményeket rendez, a műveletet darabokban hajtja végre, a köztes eredményeket ideiglenes fájlokban tárolva. Ezeket a fájlokat később egybeolvassa és újrarendezi, amíg minden sor rendezett nem lesz. A köteg méretének növelése kevesebb ideiglenes fájlt eredményez, és gyakran gyorsabb rendezést tesz lehetővé. Ezzel szemben viszont, ha az adag túl nagy, lapolvasásokat okozhat, mivel a rendező köteg egy része a rendezés során kilapolódik. Ebben az esetben sokkal célszerűbb lenne kisebb adagokat és sok ideiglenes fájlt használni. Tehát még egyszer: a csereterület lapolvasásai határozzák meg, mikor foglaltunk le túl sok memóriát. Ne feledjük azt sem, hogy minden háttérművelet ezt az értéket használja az összes rendezés során, legyen az `ORDER BY`, `CREATE INDEX` vagy `merge join`. Több egyidejű rendezés ennek a memóriának természetesen a többszörösét foglalja le.

Gyorstárméret és rendezőméret

A gyorstárméret és a rendezőméret a memóriahasználatra egyaránt kihat, így egyiket sem növelhetjük anélkül, hogy ne lennénk befolyással a másikra. Ne feledjük, hogy a gyorstárméretet a Postmaster induláskor foglalja le, míg a rendezőterület mérete az éppen futó rendezések számától függően változhat. A gyorstár mérete általában jobban számít, mint a rendezőméret. Emellett bizonyos `ORDER BY`-t, `CREATE INDEX`-et vagy `merge join`-t használó lekérdezések sebessége nőhet, ha nagyobb rendező kötegfájlméretet adunk meg.

Ezenkívül számos operációs rendszer megköti, hogy mennyi megosztott memóriát használhatunk fel. E határ növelése viszont már az operációs rendszer belső ismeretét igényli, hiszen újra be kell állítani és újra kell fordítani a magot. További adatok a PostgreSQL 7.1 Administrator's Guide-ban ➲ http://www.postgresql.org/docs/admin/kernel-resources.html címen olvashatók.

Lemezkezelés

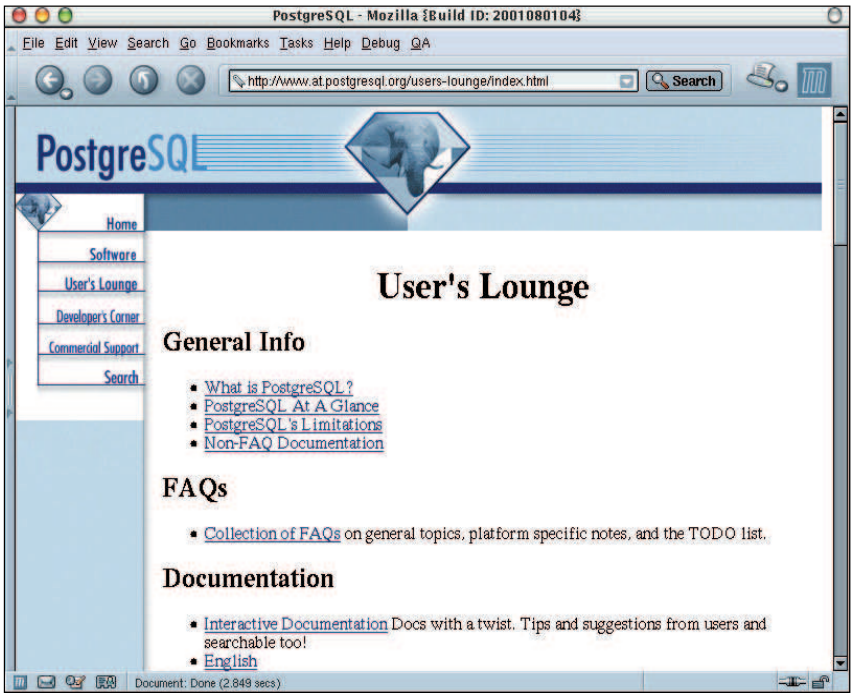
A lemezmeghajtók fizikai felépítése ezeket az eszközöket a fentebb említett egyéb tárolóeszközöktől a teljesítmény szempontjából különbözteti meg. A többi tárolóeszköz bármely bájtot ugyanolyan sebességgel képes elérni. A lemezmeghajtó – pörgő lemezei és mozgó feje miatt – a fej pillanatnyi helyzetéhez közeli adatokat sokkal gyorsabban éri el, mint a távolabb lévőket. A fej átmozgatása a lemez másik cilinderére elég sok időt vesz igénybe és ezt a Unix-rendszermag fejlesztői természetesen tudják. Amikor nagyméretű fájlokat tárolunk a lemezen, a fájldarabkákat egymáshoz minél közelebb próbálják meg elhelyezni. Tegyük fel, hogy a fájl tíz blokkot foglal el a lemezen. Az operációs rendszer az 1–5. blokkokat az egyik cilinderen, a 6–10. blokkokat a másik cilinderen helyezi el. Ha a fájlt elejétől a végéig beolvassuk, mindössze két fejmozgás szükséges – az első, hogy elérjük a 1–5. blokkokat tároló cilindert, valamint még egy, hogy átlépjünk a 6–10. blokkok cilinderére. Ha viszont a fájladatokat nem sorban olvassuk be, hanem például 1., 6., 2., 7., 3., 8., 4., 9., 5., 10. sorrendben; máris tíz fejmozgás szükséges. Amint láthatjuk, lemezek esetében a szekvenciális elérés sokkal gyorsabb, mint a véletlen elérés, ezért ha a tábla jelentős részét be kell olvasni, a PostgreSQL az indexekre a szekvenciális keresést részesíti előnyben. Az előbbiek fényében már a gyorstár értékét is láthatjuk.

Többszörös lemezhasználat

A lemezfej jó néhányszor körbefordul az adatbázis-tevékenység alatt. Ha túl sok olvasási, illetve írási kérelem érkezik, a meghajtó telítődhet, ez pedig gyenge teljesítményt eredményez (a vmstat és a sar az egyes lemezek használatának mértékéről ugyancsak szolgáltat adatokat).
A lemeztúlterheltség elkerülésére az egyik megoldás az, ha a PostgreSQL néhány adatfájlját egy másik meghajtóra helyezzük. Ne feledjük, itt a fájlok azonos meghajtónak másik fájlrendszerére mozgatása jelent segítséget, ugyanis a meghajtó összes fájlrendszere ugyanazokat a fejeket használja. Az adatbázis-elérés a lemezek között többféleképpen is szétoszthető:

- *Adatbázisok mozgatása* – a initlocation segítségével más lemezeken is készíthetünk adatbázisokat.
- *Táblák mozgatása* – közvetett hivatkozások segítségével a táblákat vagy indexeket másik lemezre is áthelyezhetjük. Ezt a mozgatást csak akkor szabad elvégezni, ha a PostgreSQL le van állítva. Továbbá a PostgreSQL semmit sem tud a közvetett hivatkozásokról (symbolic link), így ha letöröljük, majd újra létrehozzuk a táblákat, akkor azok az adatbázishoz rendelt alapértelmezett helyen fognak létrejönni. A 7.1 változat alatt a pg_database.oid és a pg_class.relfilenode rendeli az adatbázis-, a tábla- és az indexneveket a fájlnevekhez.
- *Indexek átmozgatása* – a közvetett hivatkozások lehetővé teszik, hogy az indexeket adatszerkezet-táblájukból más meghajtókra helyezzük át. Ezáltal megvalósítható, hogy míg az egyik lemezen indexkeresést hajthatunk végre, addig a másik adatszerkezet-keresést (heap lookups) végezhet.
- *Joinok mozgatása* – közvetett hivatkozások használatával az összekapcsolt táblákat is külön lemezre helyezhetjük. Ha az A és a B táblát összekapcsoljuk, az A tábla keresései az egyik meghajtón, a B tábla keresései a másik meghajtón hajtódhatnak végre.
- *Naplók átmozgatása* – közvetett hivatkozásokkal a thepg_xlog könyvtárat is átmozgathatjuk másik lemezre. (A Pg_xlog csak a PostgreSQL 7.1-es vagy későbbi kiadásokban létezik.) Más írásszolgáltatásokkal ellentétben a PostgreSQL-naplózást mindig a tranzakció befejezése előtt kell a lemezre kiírni. A gyorstárat ezekhez az írásokhoz nem használhatjuk. Ha a naplózáshoz külön lemez áll rendelkezésre, akkor a lemezfej folyamatosan a napló pillanatnyi cilinderében maradhat, és nem kell kivárni a fejmozgatási időt. Használhatjuk a Postgres `-F` kapcsolót is, ami a naplóírások azonnali ürítését megakadályozza, de egy rendszerösszeomlás esetén mindent mentésekből kell helyreállítani.

További lehetőség a RAID-képességek használata, amikor egyetlen fájlrendszert osztunk szét számos meghajtó közt.



➲ http://www.at.postgresql.org/users-lounge/index.html

Összegzés

Szerencsére a PostgreSQL-nek nincs szüksége túl sok gyorsításra. A legtöbb érték a hatékony teljesítmény eléréséhez önműködően beállítódik. A felügyelő befolyásolhatja a gyorstár méretét és a rendezőméretet a rendelkezésre álló memória minél jobb kihasználása céljából. A lemezműveleteket több lemez közt is megoszthatjuk. További értékeket állíthatunk be a share/etc/postgresq/postgresql.conf fájlban. Amennyiben a PostgreSQL-t különlegesebb értékekkel is ki szeretnénk próbálni, akkor ezt a fájlt másoljuk a data/postgresql.conf helyre.

Bruce Momjian a PostgreSQL Global Development Team társalapítója és a Great Bridge adatbázis-fejlesztés LLC (➲ http://www.greatbridge.com/) alelnökeként tevékenykedik. Ő a szerzője az Addison-Wesley kiadásában megjelent PostgreSQL: Introduction and Concepts című könyvnek.

Hálózaton keresztüli MySQL-lekérdezés Python segítségével

Hogyan készítsünk kifinomult keresőfelületet Python CGI parancsfájlok alkalmazásával?

Mostanában ezernél is több Go-játékállás került a birtokomba (a Go ma is kedvelt táblás játék az ősi Keletről), amelyek profi játékosok vagy éppen lelkes műkedvelők lépéseit őrzik. Ezek Smart Game Format-ban (SGF) találhatók – ami az egy-két játékoson alapuló táblás játékok állásainak tárolására tervezett szöveges formátum. Honlapomon keresztül ezeket természetesen más játékosok számára is elérhetővé tettem.
Az adatállományban való keresést megkönnyítendő először egy HTML-űrlapot használtam, amely egyetlen szöveges bemeneti mezőt tartalmazott. A felhasználó begépelhetett egy karaktersorozatot (például a játékos nevét), ami azután egy Python CGI parancsfájlhoz került, és ez a jó öreg *grep*-et használta arra, hogy megtalálja a megfelelő fájlt (*1. kép*). Csakhogy meglehetősen nyers formája ez a keresésnek. Például az „egy adott játékos által játszott összes játékot” könnyen kikereshetjük ily módon, de az olyan összetettebb dolgokkal már nem tudunk mit kezdeni, mint „az összes játék, amelyet egy bizonyos játékos fekete kövekkel játszott” vagy „az összes, adott játékos által 1995-ben megnyert játék”.
Ahhoz, hogy jobb keresőmotort készíthessek, más megközelítést kellett választanom. Először is az adatbázisnak le kell írnia a játékfájlok gyűjteményét, majd többszörös bemeneti mezőt szükséges készíteni, hogy a felhasználónak egy időben több különféle adat szerinti keresésre is lehetősége nyíljon. Végezetül valami módot kell találnunk arra, hogy a böngésző és az adatbázis között kapcsolat épülhessen fel, valamint, hogy a keresés eredményét HTML-formátumban a felhasználóhoz tudjuk juttatni. A keresés teljes menetét az 1. kép szemlélteti.

Az eszközök

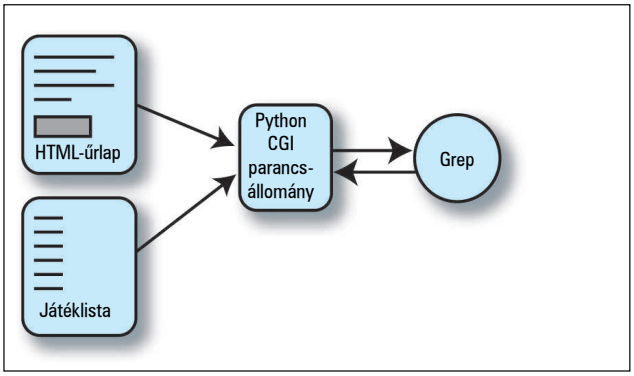
Adatbázisként a MySQL 3.22.32-változatát, CGI parancsfájl készítéshez pedig a Python 1.5.2 rendszert választottam. A Perlen is gondolkoztam, de a Python stílusa inkább kedvemre való volt. Ezeknek a programoknak a telepítése már számos cikkben megjelent, így erről most külön nem szólunk. Az olvasó további adatokat lelhet a telepítéssel kapcsolatban a cikk végén megadott címeken. A MySQL és a Python közötti kapcsolattartást kiegészítő modul segítségével lehet megvalósítani, amire rövidesen kitérek.

A MySQLmodule

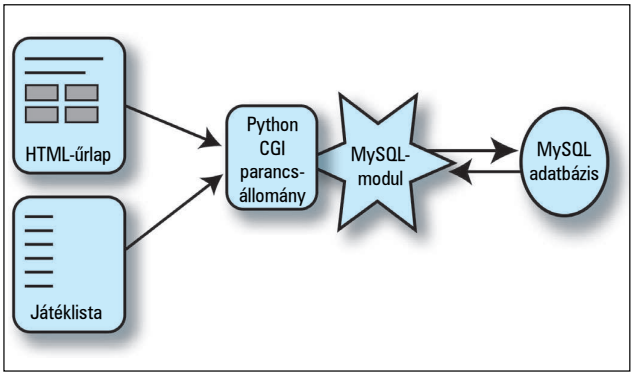
A Python minden MySQL-lekérdezést a *Joerg Senekowitsch* által tervezett egyedi modulon keresztül végez. Természetesen jó néhány egyéb modul is található a hálózaton, de a MySQLmodule1.4 könnyen telepíthető és megérthető. Nálam igen jól bevált. Néhány rendszeren (mint például a FreeBSD) ezt a modult a Pythonnal egy időben fel lehet telepíteni. Az én Slackware 7.1 Linuxom alatt dinamikusan betölthető modulként kellett lefordítanom. Ez háromlépéses művelet: ki kell bontani a MySQLmodule tárolóállományát, le kell fordítani a megosztott modult, majd a készet valahová a Python könyvtárszerkezetébe célszerű telepíteni.
Az első lépés meglehetősen egyszerű. Rendszergazdaként a következőt gépelhetjük be:

```
myhost:~# tar xvzf MySQLmodule-1.4.tar.gz
```

Ezáltal néhány állományt tartalmazó új könyvtár jön létre MySQLmodule-1.4/ néven. Az itt található fájlok közül a legfontosabbak: a MySQLmodule.c (a lefordítandó modul forrásfájlja), illetve a README (ami a telepítéssel kapcsolatos hasznos adatokat tartalmazza).



1. ábra A régi, kimunkálatlan keresési felület



2. ábra Kifinomultabb felület

A második lépésben már egy-két trükk is megbújik. Pontosan tudni kell például a MySQL és a Python könyvtárainak és fájlállományainak a helyét. Az én rendszerem a MySQL 3.22.32 a /usr/lib/mysql könyvtárba helyezi a mysqlclient könyvtárat, a mysql.h include fájl pedig a /usr/include/mysql helyen lelhető fel. A Python-könyvtárak a /usr/lib/python1.5/config helyen találhatók, míg az include fájlállományok a /usr/include/python1.5-ben. A MySQLmodule fordítását végző parancs a következő:

```
myhost:~# gcc -shared -I/usr/include/mysql -I/
 ➥usr/include/python1.5 MySQLmodule.c -L/
 ➥/usr/lib/mysql -lmysqlclient -L/usr/lib/
 ➥python1.5/config -lpython1.5 -o MySQLmodule.so
```

Megszívlelendő tanács: a fenti parancsban megadott elemek sorrendje kötött, tehát nem szabad megváltoztatni! Hidd el nekem, ez drágán szerzett tapasztalat.

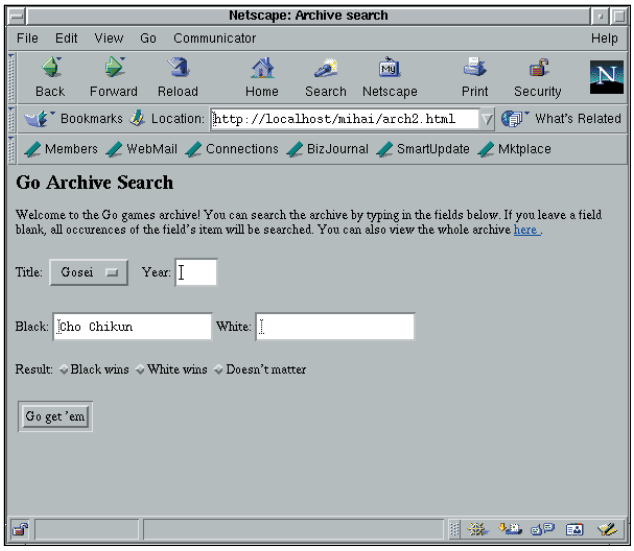
A harmadik lépés a MySQLmodule.so átmásolása egy olyan könyvtárba, ahol a Python futtatható állomány megtalálja. A Python 1.5 esetében ez a /usr/lib/python1.5/lib-dynload, ahol az egyéb megosztott objektumfájlok is találhatók. A Python 2.0 esetén (ezt szintén kipróbáltam) a /usr/lib/python2.0/site-packages/ könyvtár használatát javaslom. Amint a modul telepítve lett, azonnal el is érhető a Pythonból. Célszerű rögvest ki is próbálni egy egyszerű `import` utasítással:

```
myhost:~$ python
Python 1.5.2 (#1, May 28 2000, 18:04:10)
Copyright 1991-1995 Stichting Matematisch
Centrum,
Amsterdam
>> import MySQL
>>
```

Ha a Python nem „panaszkodott”, valószínűleg sikerült helyesen telepíteni a MySQLmodule-t és jól működik.

A HTML-űrlap

A felhasználó több szempont szerint is kereshet az adatbázisban, mint például „a viadal neve”, „a fekete és a fehér játékosok neve”, „a játék kelte” (legalább az év), illetve „a játék győztese”. Mindezen adatok elérhetők az SGF fájlokban, a pillanatnyi játékmezőben. Írtam egy HTML-dokumentumot, amely az 1. listában (lásd a 15. CD Magazin/MySQL könyvtárában) látható. Szeretek kézzel írni HTML-kódot, ráadásul jelen esetben semmilyen bonyolult dologra nincs szükség. Természetesen a *myhost* szócskát a webkiszolgáló pillanatnyi nevével helyettesítjük. Az olvasó azt is észreveheti, hogy a HTML-űrlap – amint a *submit* (elküld) gombot leüti – a search.py nevű CGI parancsfájlt hívja meg. A dokumentum Netscape alatti megjelenése a 1. képen látható.



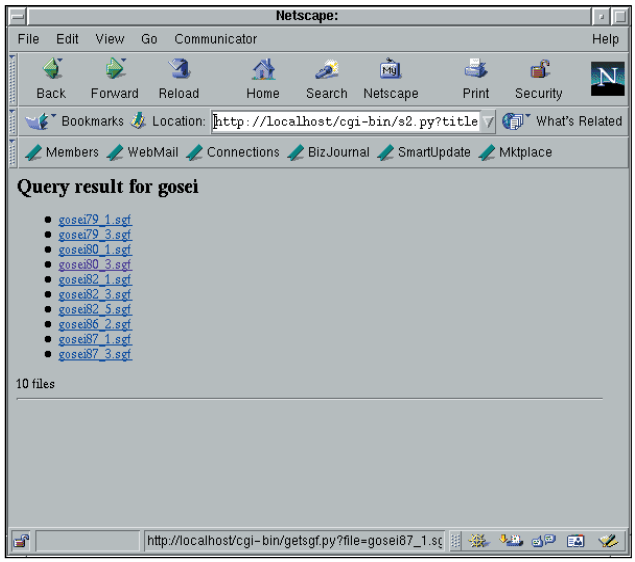
1. kép A dokumentum Netscape alatt

Az Igo Adatbázis

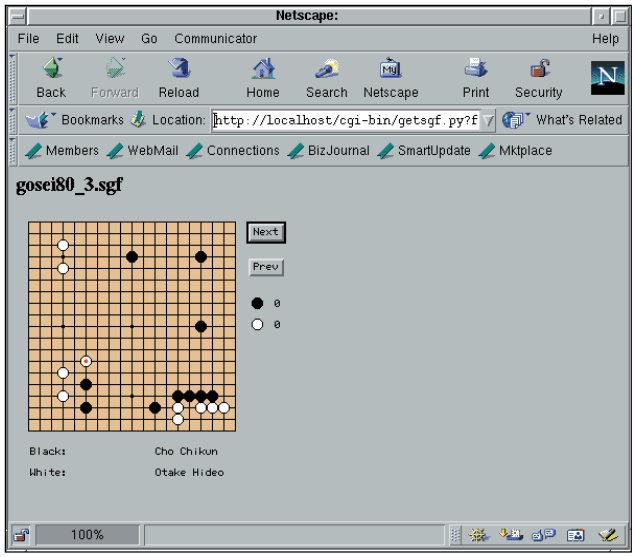
A HTML-űrlapon megjelenő összes elemet le kell írni az adatbázisban. Ezért készítettem egy új adatbázist, amely a táblákat tárolja. Ezt rendszergazdaként lehet végrehajtani, a következők begépelésével:

```
myhost:~# mysqladmin create igo
```

ahol az igo az új adatbázis neve. Ez természetesen még nem elég,



2. kép A search.py eredményei



3. kép gose:80_3.sgf.applet

mivel így egyedül a rendszergazdának lesz hozzáférése az új adatbázishoz. Ahhoz, hogy minden felhasználónak megadhassuk a SELECT jogosultságot (és csak azt) az új adatbázishoz, a rendszergazdának a következőket kell begépelnie:

```
myhost:~# mysql mysql
mysql> insert into db values (
       ➥´%´,´igo´,´´,´Y´,´N´, ´N´, ´N´,
       ➥´N´,´N´,´N´, ´N´, ´N´, ´N´);
```

A másik megoldás az lehet, ha `GRANT SELECT ON ´igo.*´ TO ´%´` parancsát használjuk, ebben az esetben a flush parancsra sincs szükség – *MonK*, a fordító.
Ez a MySQL belső adatbázisában megváltoztatja a táblaadatbázist, és új bejegyzést készít az igo adatbázis számára. Ezután a MySQL démont újra kell indítani vagy gépeljük be a `flush privileges;` parancsot, hogy a MySQL démon a jogosultságváltozásokat figyelembe vegye.

(Javaslatom esetén semmi ilyesmit nem kellene tenni, mindez szépen, önműködően megtörténne – MonK.)
Így már minden felhasználó elérheti az igo adatbázis adatait, de csak a rendszergazda képes megváltoztatni azokat.

A MySQL-táblázatok

A táblák készítése (minden viadalhoz egy-egy) meglehetősen könnyű volt. Például a Gosei viadalhoz tartozó *gosei* nevű tábla a következő parancsokkal készült:

```
myhost:~# mysql igo
mysql> create table gosei (
black varchar(30),
white varchar(30),
dt date,
rez varchar(10),
fname varchar(30)
);
Query OK, 0 rows affected, (0.00 sec)
mysql>
```

A táblának öt oszlopa van: a fekete játékos neve, a fehér játékos neve, a játék dátuma, az eredmény és végül a megfelelő SGF fájl neve.
Az adatok betöltése a táblába azonban már más kérdés. Az egyik módszer akár ez is lehetne:

```
mysql> insert into gosei values (
Cho Chikun´, ´Kato Masao´, ´1987-07-03´,
´B+3.5´, ´gosei87_1.sgf´ );
```

de inkább megtanulok egy új programozási nyelvet, mintsem ezt ezerszer begépeljem. Szerencsére akad más módja is annak, hogy az adatokat a MySQL táblába betöltsük – mégpedig szövegfájlokon keresztül. A fájlban minden sor egy táblasornak felel meg a táblában, a mezőket pedig szóközkarakterek választják el, ahogy ez az első táblában látható. Tételezzük fel, hogy ezt a fájlt is goseinek nevezzük. Ahhoz, hogy ezeket az adatokat a MySQL táblába tölthessük, a következőt szükséges begépelni:

```
mysql> load data infile "gosei" into table gosei;
```

Az olvasó joggal kérdezhetné: „Rendben, de a hatalmas szöveges állományok elkészítése talán nem éppolyan nagy munka?" A válasz: nem, mert ez feladat meglehetősen könnyen elvégezhető egy másik egyszerű Python-parancsfájl segítségével (ezt most nem fogom bemutatni, mivel nem tartozik közvetlenül a tárgyhoz).
Amint az igo adatbázis elkészült és a táblákat feltöltöttük adatokkal, már csak egyetlen dolog maradt hátra: megírni azt a Python CGI parancsfájlt, amely a felhasználóhoz juttatja el HTML-formátumban az űrlapot, lekérdezi az adatbázist és listázza a megtalált játékfájlokat.

A Python CGI parancsfájl

Ez a search.py nevű program a 2. listán látható (elérhető az ➲ ftp://ftp.ssc.com/pub/lj/listings/issue85 címen). Két nagyszerű modult használ, amelyeket a harmadik, illetve a negyedik sorban importálok. A CGI modul csaknem olyan, mint egy varázslat: a HTML-űrlap által továbbított adatokat a Python könyvtárként kapja meg. A programozónak nem is kell olyan részletekkel foglalkoznia, hogy vajon milyen eljárás (GET vagy POST) alkalmazásával érkezett az adat a CGI parancsfájlhoz. Ugye, milyen csodálatos a Python?
A MySQL-module-t szintén könnyű használni: a kívánt adatbázissal négy egyszerű utasítással létrehozza a kapcsolatot, elküldi a megfelelő lekérdezést, megkapja az eredményt, és sorlistákban (rowlist, olyan Python-lista, amelynek minden eleme egyetlen sort tartalmaz) tárolja azokat.
A parancsfájl szerkezete a következő: a HTML-adat lekérdezése, az adatbázis-lekérdező karaktersorozat elkészítése az űrlap adatainak megfelelően, a tábla lekérdezése, majd a megfelelő sorok kiíratása. Mivel minden Go-viadal külön táblában helyezkedik el, a harmadik lépést egy ciklusban ismételgetjük, ameddig csak szükséges.
A Python-kód és a csatolt megjegyzések önmagukért beszélnek, ezért csak a következő sorokra térek ki bővebben:

```
print ´<li><a href="http://myhost´ + \
´/cgi-bin/getsgf.py?file=´ + e[0] + ´">´
print e[0] + ´</a>´
```

Itt az e[0] az SGF fájl neve. A fájlnév egyszerű kiíratása helyett (ez csupán korlátozott segítséget jelentene a felhasználók számára), a `print` utasítás egy HTML-hivatkozást (anchor) készít, amely a fájlnevet a getsgf.py CGI parancsfájlhoz továbbítja. Ez utóbbi parancsfájl (az egyszerűség kedvéért most nem mutatjuk be) kikeresi az SGF fájl jelenlegi elérési útját, és elindít egy Java appletet, hogy a fájl tartalmát szép grafikus felületen jeleníthesse meg.
Természetesen mind a search.py, mind pedig a getsgf.py fájlokat futtatható állománnyá kell tenni, és át kell helyezni a cgi-bin könyvtárba. Végül az utolsó elkerülendő csapda: minden SGF játékfájlt valahová a DocumentRoot könyvtárba helyezzünk (annak megfelelően, ahogy azt a httpd beállítófájljában meghatároztuk), máskülönben a CGI parancsfájl nem találja meg őket.
A search.py futtatásának eredményét a *2. képen* figyelhetjük meg, ahogyan azt a felhasználó látni fogja. Bármely fájlra kattintva az *3. képen* látható applet indul el, amely a játék egyes lépéseit a megfelelő sorrendben mutatja be.

Összegzés

A Python csodálatos nyelv: a MySQLmodule segítségével egyszerűvé válik az olyan apró programok írása, amelyek a MySQL-adatbázisból nyernek adatot. A Python CGI parancsfájlok írására is kiválóan alkalmazható. Egy adatbázis webes elérhetőségének megteremtése mindössze pár soros kód megalkotását jelenti.
Az itt bemutatott alkalmazás meglehetősen korlátozott: a felhasználó mindössze öt változó szerint képes választani, és ezt is csak kötött, a HTML-kódba előre beégetett módon teheti meg. Ugyanakkor az is elképzelhető, hogy a felhasználó egy textarea bemeneti mezőbe saját adatbázis-lekérdezését írja be, majd a lekérdezés eredményét közvetlenül megszemlélheti. A lehetőségeknek tulajdonképpen csak a programozói képzelet szab határt.

Mihai Bisca
A Romanian Go bajnokság előző nyertese (AKA 5dan). Megszámlálhatatlan órát tölt Slackware Linuxán játékkal, és arról álmodozik, hogy otthonról fog dolgozni, az Interneten keresztül. Másik – nappali – életében szemorvosként tevékenykedik.

Kapcsolódó címek

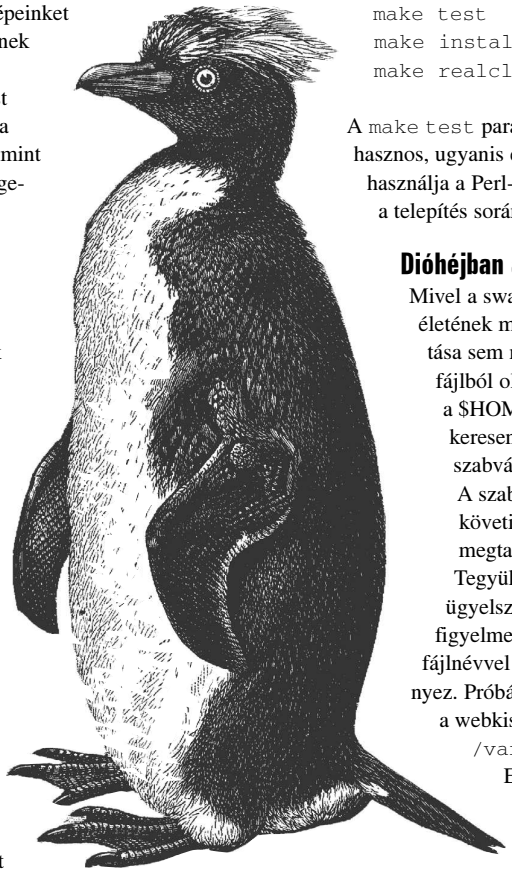
Go ➲ http://www.cwi.nl/people/jansteen/go/index.html
MySQL ➲ http://www.mysql.com/
MySQLmodule ➲ http://mysql.he.net/Downloads/Contrib/
Python ➲ http://www.python.org/
SGF ➲ http://www.red-bean.com/sgf/
A szerző honlapja ➲ http://igo.profis.ro/

Swatch: önműködő naplófigyelés

A napló tanulmányozásának javasolt módja éber, de lusta emberek számára – így több idejük marad feladataik végrehajtására.

Múltkor a Szaktekintély rovat bemutatott néhány programot, amelyek megvédhetik gépeinket a gonosz szándékú betörőktől. Ezeknek a segédeszközöknek van egy közös feladatuk: a naplózás. Legalább olyan fontos lehet tudomást szerezni a betörési kísérletről, mint távol tartani a betörőket. De kinek akad ideje és türelme – nap mint nap – az általa ellenőrzött összes rendszeren rengeteg nagyméretű és többnyire érdektelen adatot tartalmazó naplófájlt végigolvasni?

A swatchnak (Simple WATCHer) van. A száz százalékig Perlben fejlesztett swatch már az írás pillanatától figyeli a naplófájlokat és rögvest a tettek mezejére lép, amint olyasmit talál, aminek a keresésére megkértük. Ez az egyszerű, rugalmas és hasznos eszköz minden egészséges mértékben óvatos rendszergazda számára kötelezően telepítendő program.



A swatch telepítése

A swatch két módon telepíthető. Az első természetesen az általad kedvelt Linux-terjesztéshez adott bináris csomag, ha van ilyen. A Mandrake jelenlegi változata magában foglalja a swatch RPM-csomagot, de más népszerű terjesztések (például RedHat, SuSE, Slackware vagy Debian) nem tartalmazzák. Ez nem is baj, mert a swatch telepítésének második módja elég érdekes. A swatch forrása, amely a

➲ http://www.stanford.edu/~atkins/swatch

címről tölthető le, a Makefile.PL nevű bonyolult parancsfájlt tartalmazza. Ez a parancsfájl önműködően ellenőrzi a szükséges Perl-modulok meglétét, a Perl 5 CPAN lehetőségének segítségével letölti és telepíti a hiányzó modulokat, majd előállítja a Makefile-t, amellyel a swatch lefordítható.

Miután a szükséges modulokat telepítetted, akár a swatch Makefile.PL parancsfájljával, akár kézzel (és utána futtatod a Makefile.PL-t), a Makefile.PL a következő eredményt adja vissza:

```
[root@barrelofun swatch-3.0.1]# perl Makefile.PL
Checking for Time::HiRes 1.12 ... ok
Checking for Date::Calc ... ok
Checking for Date::Format ... ok
Checking for File::Tail ... ok
Checking if your kit is complete...
Looks good
Writing Makefile for swatch
[root@barrelofun swatch-3.0.1]#
```

Miután a Makefile.PL a swatch Makefile-ját elkészítette, a következő parancsokkal fordítható le és telepíthető:

```
make
make test
make install
make realclean
```

A `make test` parancsot nem kötelező kiadni, de hasznos, ugyanis ez biztosítja, hogy a swatch helyesen használja a Perl-modulokat, így megelőzhetjük a telepítés során esetleg fellépő hibákat.

Dióhéjban a swatch beállításáról

Mivel a swatch programot a rendszergazdák életének megkönnyítésére hozták létre, beállítása sem nehéz. Beállításait a swatch egyetlen fájlból olvassa: alapértelmezés szerint ez a $HOME/.swatchrc. A fájl a swatch által keresendő szövegmintákat tartalmazza szabványos kifejezések formájában.

A szabványos kifejezést azok a műveletek követik, amelyeket a swatchnak a szöveg megtalálásakor végre kell hajtania.

Tegyük fel, hogy egy webkiszolgálót felügyelsz és azt szeretnéd, hogy a rendszer figyelmeztessen, ha valaki egy nagyon hosszú fájlnévvel tártúlcsordulásos támadást kezdeményez. Próbáld ki saját magad, mit ír ilyenkor a webkiszolgáló a naplóba: add ki a `tail /var/apache/error.log` parancsot! Ebből kiderül, hogy a naplóbejegyzés tartalmazza a *File name too long* karakterláncot. Tegyük fel továbbá azt is, hogy levélben szeretnél értesülni az eseményről. Ekkor ezt kell beírnod a .swatchrc fájlodba:

```
watchfor /File name too long/
    mail addresses=mick\@visi.com,
    subject=BufferOverflow_attempt
```

Látható, hogy a bejegyzés a *watchfor* kifejezéssel kezdődik, amelyet egy szabványos kifejezés követ. Ha nem magas szinten ismered a szabványos kifejezéseket (de ugye tervezed a megtanulásukat?), ne aggódj, a legegyszerűbb esetben elegendő – két perjel között – a keresett szöveg egy darabját megadni.

A swatch tetszőleges számú műveletet képes végrehajtani, ha szabványos kifejezésedhez illeszkedést talált. Ebben a példában arra utasítottuk a programot, hogy küldjön levelet a mick@visi.com címre *BufferOverflow_attempt* tárggyal. Vegyük észre a fordított perjelet a @ jel előtt, ha ez nem lenne, a Perl a @-ot különleges karakterként értelmezné. Ha a tárgyban szóközöket szeretnél használni, azokat is meg kell védeni a fordított perjellel például: *subject=Buffer\ Overflow\ attempt*. A levélküldés mellett más műveleteket is használhatunk (lásd még a *táblázatunkban*).

A beállításokról és a műveletekről további részleteket a swatch(1) súgóoldal tartalmaz.
Bővítsük tovább az előbbi példánkat! Tegyük fel, hogy a tártúlcsordulási kísérletek kapcsán küldendő levélen kívül azt is tudni szeretnéd, hogy mikor olvas valaki egy adott weboldalt, de csak akkor, ha éppen be vagy jelentkezve a konzolra. Ugyanebbe a .swatchrc fájlba valami ilyesmit írj be:

```
watchfor /wuzza.html/
       echo=red bell 2
```

Az esemény bekövetkeztekor a gép sípol és a konzolra ír.
Fontos, hogy ezeket az üzeneteket csak akkor fogod látni (és a sípszót hallani), ha ugyanabba a héj-munkafolyamatba jelenkeztél be a konzolra, amelyből a swatch programot is indítottad. Ha kilépsz, elmégy ebédelni és visszatértedkor ismét bejelentkezel, a régi munkafolyamatban elindított swatchfolyamatok üzeneteit nem fogod látni, annak ellenére, hogy ezek a folyamatok még ekkor is futnak.
Ha nem vagy benne biztos, add hozzá a `mail` vagy más, konzoltól független műveletet, például az `exec`-et, amely egy figyelmeztető parancsfájlt indít el. Természetesen csak akkor tedd ezt, ha a kérdéses esemény tényleg annyira fontos.
A figyelmes olvasó már minden bizonnyal észrevette, hogy az előző példa csak olyan Apache webkiszolgálóval működik, amely a hibaüzeneteket és az elérési naplót ugyanabba a fájlba írja. A különböző megfigyelt fájlokhoz nem társítottunk különböző műveleteket, nem is tehettük volna. Mi történik akkor, ha egynél több naplófájlt szeretnél a swatch programmal megfigyeltetni?
Semmi gond: bár minden egyes .swatchrc fájl csak egy megfigyelendő fájlt ír le, nincs akadálya annak, hogy a swatchot több példányban futtasd – mindegyiket a saját .swatchrc fájljával. Másként fogalmazva: a swatch beállításainál a .swatchrc az alapértelmezett neve, ez azonban más is lehet.
Az előző két példát úgy mentheted két fájlba, hogy az előző egyszerű .swatchrc fájl tartalmát például a .swatchrc.hterror fájlba teszed, az előző *watchfor* bejegyzés sorait pedig mondjuk a .swatchrc.htaccess fájlba mented.

A swatch beállításai haladók számára

Eddig olyan műveleteket tárgyaltunk, amelyeket minden egyes alkalommal végre kell hajtani, amikor az esemény bekövetkezik. A swatch viselkedését sokféleképpen, finomabban is szabályozhatjuk.
Az első és legkézenfekvőbb dolog, hogy kihasználjuk a keresési minták szabványos kifejezés voltát. A szabványos kifejezések, amelyek valójában saját szövegformázó nyelvet alkotnak, hihetetlenül hatékonyak, és nagy részben ezek felelősek a Perl, a sed, a vi és sok más unixos segédeszköz varázslatos működéséért.
Illik legalább egy pár *regex*-trükkel tisztában lenni, ezért ismertetek közülük néhányat. Az egyes számú trükk a vagylagosság, ez a szabványos kifejezéshez a „|” jel formájában a logikai *vagy* műveletet adja. Vegyük a következő szabványos kifejezést:

```
/reject|failed/
```

Ez a kifejezés minden olyan sorra illik, amely a `reject` vagy a `failed` szavakat tartalmazza. A vagylagosságot akkor használd, ha azt szeretnéd, hogy a swatch egynél több kifejezésre vonatkozóan hajtsa végre ugyanazt a műveletet.
A második számú trükk a Perlre jellemző szabváynos kifejezésmódosító: a *kis- és nagybetű egyforma*, amely *per-i* néven is ismert, ugyanis mindig a szabváynos kifejezés záró perjele után következik. A `/reject/i` szabváynos kifejezés minden sorra illeszkedik, amely a *reject* szót tartalmazza, legyen az `Reject`, `REJECT`, `rEjEcT` stb. formában írva. Ez legalább olyan hasznos, mint a vagylagosság, a teljesség kedvéért azonban elárulom, hogy a *per-i* az egyik legszámításigényesebb Perl-módosító. Ha a naplónézegetés és az önmegtámadás stb. ellenére sem vagy száz százalékig biztos az aggodalomra okot adó támadás naplófájlbeli kinézetében, a *per-i* segít a találgatásban.

Néhány művelet, amelyet a swatch elvégezhet

Művelet (kulcsszó)	Leírás
echo=normal, underscore, blue, inverse stb.	Az egyező sort kiírja a konzolra – a beállított különleges szövegmódban (az alapértelmezett mód a „normal”).
bell N	A sort kiírja a konzolra és N-szer sípol (alapértelmezés szerint N=1).
exec command	Végrehajt egy parancsot vagy parancsfájlt.
pipe command	A command parancs bemenetére irányítja a kimenetet.
throttle HH:MM:SS	HH:MM:SS ideig vár az egyezés után, és csak ez idő lejárta után hajtja végre a műveletet egy ugyanolyan típusú egyezésre. Így elkerülhető, hogy rövid időn belül bekövetkező nagy mennyiségű swatch-eseménnyel a rendszert akasszák meg.

Amennyiben a szabványos kifejezéseket a lehető legmagasabb szinten szeretnéd kezelni, javaslom *Jeffrey E. F. Friedl* Mastering Regular Expressions című könyvének tanulmányozását.
A swatch szabályozásának másik módja, hogy megadjuk, a nap melyik időpontjában hajtson végre egy adott műveletet. Ezt a művelet után a `when=` kulcsszó beírásával teheted meg. Vegyünk például egy közepesen fontos eseményre vonatkozó .swatchrc bejegyzést, amelyről hétköznap konzolüzenetek formájában szeretnék értesülni, hétvégén azonban levelet óhajtok kapni róla. Ehhez a `when` értékét így kell beállítani:

```
/file system full/
       echo=red
       mail addresses=mick\@visi.com,
       subject=Volume_Full,when=7-1:1-24
```

A `when=` beállítás írásmódja:
`when=napok_tartománya:órák_tartománya`
Láthatjuk, hogy bármikor, amikor *file system full* kerül a naplóba, a swatch a naplóbejegyzést pirossal kiírja a konzolra. Levelet is küld, de csak szombaton („7”) és vasárnap („1”).

A swatch futtatása

A swatch elvárja, hogy aki elindítja, annak a könyvtárában a .swatchrc létezzen. A swatch program ugyanis alapértelmezés szerint ideiglenes fájljait itt is tartja (minden indításkor létrehoz és futtat egy „watcher process” nevű parancsfájlt, amelynek neve egy pontra, valamint a létrehozó swatch-folyamat PID-jére végződik).

Megengedjük-e a Perlnek, hogy a saját moduljait letöltse és telepítse?

A Comprehensive Perl Archive Network (CPAN) Perl programgyűjteményeket összefogó világméretű hálózat. A Perl 5.6.x változata olyan modulokat (többek között a CPAN és a CPAN::FirstTime) tartalmaz, amelyek segítségével Perl-modulokat tölthetünk le, ellenőrizhetünk, sőt akár le is fordíthatunk gcc-vel az Interneten elhelyezkedő CPAN-helyekről. A CPAN és a Perl CPAN lehetőségeinek részletes tárgyalása meghaladja e cikk kereteit, de röviden bemutatom, továbbá egy fontos dologra szintén felhívom a figyelmedet.
Először lássuk a használat módját. Az Example::Module modult (ez nem létező Perl-modul) így telepítheted:

```
perl -MCPAN -e ´install Example::Module"
```

Ha a `-MCPAN` kapcsolót ez alkalommal használtad először, a CPAN::FirstTime modul indul el, és néhány kérdést kell megválaszolnod a modulok letöltésével és telepítésével kapcsolatban. Ezek jól megfogalmazott kérdések és az alapértelmezett válaszok is értelmesek. Mégis fontos odafigyelni a parancs kimenetére, mert a telepítendő modul más moduloktól függhet, esetleg vissza kell menni és végrehajtani például a

```
perl -MCPAN -e ´install Example::PreRequisite"
```

parancsot, mielőtt az első modul telepítését másodszorra is megkísérelnéd.
És most jöjjön a figyelmeztetés: a CPAN biztonsági szempontból se nem sokkal jobb, se nem sokkal rosszabb az egyéb internetes programforrásokhoz képest. Véleményem szerint a CPAN-segédprogramok viszonylag biztonságosak, mivel telepítés előtt minden letöltött modulra kiszámítják az ellenőrzőösszeget, amely erős tikosításnak számító MD5 kódot tartalmaz.
Még ha feltételezzük is, hogy egy adott csomag ellenőrzőösszegét nem cserélik le, amikor a csomagot megpiszkálják (erős feltevés), ez akkor is csak a program engedély nélküli módosítása ellen véd, miután a szerzője feltöltötte a CPAN-ra. Egy gonosz CPAN-fejlesztőt (bárki bejegyeztetheti magát) semmi sem akadályozhat meg abban, hogy kártékony kódot töltsön fel érvényes ellenőrzőösszeggel. Természetesen ugyanez a helyzet a *SourceForge* vagy a *Freshmeat* esetében is.
Remélve, hogy ezt a részt nem hangsúlyozom túl, ha mégis óvakodni szeretnél, akkor a legbiztonságosabban így telepíthetsz egy adott Perl-modult:

1. Keresd meg a modult a ➲ http://search.cpan.org/ címen.
2. Kövesd a hivatkozást a modul CPAN oldalára.
3. Ne a CPAN-ról töltsd le a modult, hanem fejlesztőjének hivatalos weboldaláról, amelyet az *Author Information* alatt adott meg, a 2. pontban már említett weboldalon.
4. Töltsd le a fejlesztő által biztosított ellenőrzőösszeget az imént letöltött tarcsomaghoz (természetesen csak akkor, ha sikerül rábukkanni).
5. Használd a gpg, MD5 stb. programokat a tarcsomag ellenőrzésére (erről akár önálló cikket is lehetne írni a Szaktekintély rovatba). Rengeteg különböző sértetlenség-ellenőrzőt használnak a programterjesztők az MD5-ön kívül, de egyiket sem használja túl sok végfelhasználó.
6. Bontsd ki a tarcsomagot, például: `tar -xzvf groovyperlmod.tar.gz`.
7. Ha az igaz utat követő üldözési mániás kung-fu mester vagy, esetleg azzá szeretnél válni, nézd át a forráskódot hibák után kutatva, jelentsd az eredményt a fejlesztőnek, illetve kürtöld világgá, és gyűjtsd be a nyílt forrás közösségének nagyrabecsülését és hálá-ját. (A nyílt forráskód csak akkor igazán nyílt, ha akad valaki, aki veszi a fáradságot és elolvassa!)
8. Kövesd a modul fordítási és telepítési utasításait, ezeket általában az INSTALL fájl tartalmazza. A teendő többnyire a következő:

```
perl ./Makefile.PL
make
make test
make install
```

Ha a szükséges modulokat a swatch Makefile.PL parancsfájlja hozta a tudomásodra, és az üldözési mániás telepítési módszert szeretnéd használni, akkor le kell írnod a szükséges modulok nevét és „meg kell ölnöd" a parancsfájlt (a jó öreg Ctrl+C-vel) – mielőtt telepíted a modulokat és újra futtatod a swatch Makefile.PL-jét.

swatch-modulok

Perl Module	*RedHat 7 RPM*	*Debian „deb" csomag*
Date::Calc	perl-Date-Calc	libdate-calc-perl
Time::HiRes	perl-Time-HiRes	libtime-hires-perl
Date::Format	perl-TimeDate	libtimedate-perl
File::Tail	perl-File-Tail	libfile-tail-perl

Akad egy harmadik módja is a hiányzó Perl-modulok telepítésének: Linux-változatod FTP-kiszolgálójáról vagy CD-ROM-járól is végrehajthatod. Bár egyik sem éri el a CPAN által nyújtott választékot, a legtöbb Linux-változat tartalmazza a legnépszerűbb Perl-modulok csomagolt változatait. Az alábbi táblázat megadja a swatch-hoz szükséges modulok csomagneveit a RedHat 7 és a Debian 2.2 rendszerekre. A fentiek nem tűnnek a swatch programhoz kapcsolódó ismeretek-nek, nem is azok, de fontosak – egyre több hasznos segédprogram jelenik meg Perl-modulként vagy olyan Perl-parancsfájlként, amely Perl-moduloktól függ, ezért valószínűleg nem a swatch lesz az utolsó alkalmazás, amely a Makefile.PL segítségével telepíthető. Hidd el nekem, megérte ezt a liternyi „tintát" elhasználni arra, hogy a modulok lelkivilágát megismertessem.

A `-c beállítások_útvonala` és a `--script-dir=útvonal` kapcsolókkal a swatch beállításait tartalmazó fájlnak és a parancsfájloknak egyéb helyek is megadhatók. Egyiket se tartsd olyan könyvtárban, amely mindenki által írható! A leghelyesebb, ha ezeket csak a fájlok tulajdonosa tudja olvasni. Ha például azt szeretném, hogy a swatch egyéni beállításaimat a /var/log könyvtárból vegye, és ezt a könyvtárat használja a parancsfájl tárolására is, akkor a következő parancsot kell kiadnom:

```
swatch -c /var/log/.swatchrc.access
➥--script-dir=/var/log &
```

Azt is meg kell mondanom a swatchnak, hogy melyik fájlt figyelje, ehhez pedig a `-t fájlnév` kapcsolót használom. Ha a fenti parancsot a /var/log/apache/access_log figyelésére akarnám használni, az alábbi a parancsot szükséges kiadnom:

```
swatch -c /var/log/.swatchrc.access \
        --script-dir=/var/log \
        -t /var/log/apache/access_log &
```

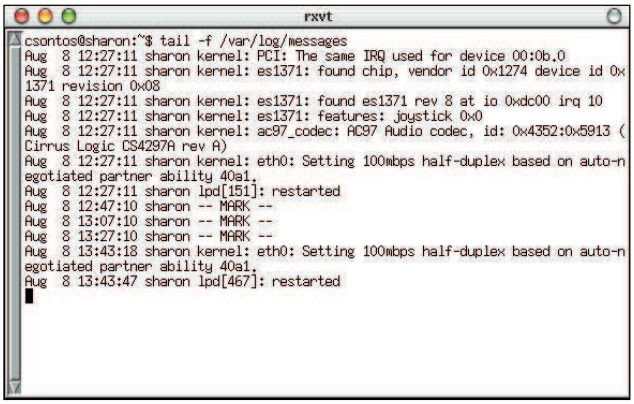
A swatch maga után általában nem takarít ki valami jól, szokása ugyanis, hogy maga mögött hagyja a watcher-process parancsfájlokat. Figyelj erre, és időről időre töröld ezeket a fájlokat a könyvtáradból vagy a `--script-dir` kapcsolóval megadott könyvtárból.
Ha egy időben több fájlt szeretnél megfigyelni, akkor több példányban kell futtatnod a swatchot; legalább a célfájl (a `-t` kapcsoló után) legyen különböző, de ezekre valószínűleg különböző beállításokat is fogsz használni.

A swatch finomhangolása (mindkét irányban)

Miután a swatchot beállítottuk és a program már fut, figyelmünket célszerű az arany középút szabályának betartására fordítanunk. Nem akarjuk, hogy a swatch túl gyakran jelezzen (mindennapos vagy egyszerű műveletekre is riasszon), de azt sem szeretnénk, ha soha semmire nem figyelmeztetne. Vajon mi lehet a helyes megoldás? Annyi különböző válasz létezik, ahányféle dologra a Unixot fel lehet használni.
Akárhogy is, nem kell megmondanom, hogy mi a túlzott részletességű jelentés: meg fogod ismerni, ha találkozol vele. Előfordulhat, hogy egyszer-kétszer megijedsz, amikor olyan események váltanak ki riasztást, amelyek később ártalmatlannak bizonyulnak. Olvasd szorgalmasan a kézikönyveket, állítgasd a .swatchrc-t és haladj tovább!
A másik eshetőséget – amikor túl kevés dolgot figyelsz meg – sokkal nehezebb észrevenni, különösen igaz ez kezdő rendszergazdák esetében. A rendkívüli események, ahogy a nevük is mutatja, ritkán fordulnak elő. Hogyan is vehetnéd észre, miként jelennek meg a naplóban? Első tanácsom, hogy szokd meg a rendszernapló gyakori olvasását, így benyomásod lesz arról, hogyan működik rendszered a mindennapok során.
Még jobb, ha a naplót valós időben olvasod. Ha kiadod a

```
tail -f /var/log/messages
```

parancsot, a rendszernapló utolsó tíz sora kiíródik, és minden ezután következő sor, létrejöttének idejében, egészen addig, amíg a Ctrl+C billentyűkombinációval le nem állítod a `tail` parancs végrehajtását. Ez minden fájl esetén működik, még a gyorsan változó naplófájloknál is.
A másik hasznos dolog a rendszer megtámadása az egyik virtuális konzolon vagy xterm-ben, míg egy másikban a naplófájlokat figyeled. A múlt hónapban és a két hónappal ezelőtt bemutatott Nessus- és Nmap-programok erre a célra tökéletesen megfelelnek. Most azt gondolhatod: „Azt hittem, azért telepítettem a swatchot, hogy ne kelljen a naplófájlokban kotorásznom!” Ez nincs így.
A swatch mérsékli ugyan, de ki nem küszöböli a naplófájlok olvasásának szükségességét.
Miután megkaptad életed első zsebszámológépét, vajon el is felejthetted a négy alapműveletet? Használhatod-e a számológépet, ha nem tudod, hogyan kell összeadni és szorozni? Természetesen nem. Ugyanez vonatkozik a naplófájlok olvasására: nem mondhatod



A tail -f /var/log/messages kimenete

meg a swatchnak, hogy keresse meg azt, amit magad sem tudnál azonosítani, mint ahogyan nem kérdezheted meg azt sem, hogy merre keresd a várost, amelynek elfelejtetted a nevét.

Miért rossz, ha beállítottad a swatchot és elfeledkezel róla?

A fentiek szellemében: ne légy elégedett, ha a swatch csendben marad. Ha a swatch jelzései ritkán jelennek meg, az egyrészt jelentheti azt, hogy a rendszered nem gyakorta van kitéve támadásoknak, másrészt azonban legalább ilyen valószínű, hogy a swatch nem elég nagy területre veti ki a hálóját. Időről-időre pásztázd végig kézzel a naplófájlokat, hogy biztosan ne maradjon ki semmi, és szükség esetén módosítsd a .swatchrc-t.
Végül ne feledd rendszeresen újragondolni a naplóbejegyzéseket létrehozó démonok naplózási beállításait! A swatch nem tud olyan eseményt elkapni, amely nem is kerül be a naplóba. Olvasgasd mind a syslogd(8) súgóoldalt, amely a syslog-démon kezelésének általános leírását tartalmazza, mind a syslogba író programok súgóoldalait, melyekben az ezekre jellemző naplózási beállításokról lelhetsz hasznos tudnivalókat.

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

Kapcsolódó címek

A swatch honlapja
➲ http://www.stanford.edu/~atkins/swatch/
Stephen Hansen és *Todd Atkins* a swatch megalkotói, valamint ők a szerzői a Centralized System Monitoring with Swatch című írásnak, mely a
➲ http://www.stanford.edu/~atkins/swatch/lisa93.html
címen olvasható.
Ajánlott irodalom továbbá *Lance Spitzner* Watching Your Log című írása.
Rövid bevezetés a swatch használatáról, eléhető a
➲ http://www.enteract.com/~lspitz/swatch.html címen.
A cikkben ajánlott könyv: *Jeffrey E. F. Friedl* Mastering Regular Expressions (O'Reilly & Associates, 1998).

Önműködő tűzfal naplókövetéssel

Néhány módszert és parancsfájlt mutatunk be a szűrők által készített naplófájlok önműködő nyomon követéséhez.

A tűzfalak olyan számítógépek, amelyek legfőbb feladata két hálózat közti hálózati forgalom egyes elemeinek szűrése. Általában azért alkalmazzák őket, hogy a belső hálózatot (LAN) megóvják az Internet egyéb részeitől. A belső háló összes gépének egyedi védelme sokkal költségesebb és időigényesebb lenne, mint egyetlen tűzfalat telepíteni, karbantartani és figyelemmel kísérni. A tűzfal különösen azoknak az intézményeknek nélkülözhetetlen, amelyek folyamatosan csatlakoznak az Internethez. A hálózati beállítástól függően az útválasztót (router) is be lehet állítani csomagszűrési feladatra, általában azonban sokkal kényelmesebb egy külön gépet működtetni tűzfalként. Mivel hihetetlen biztonságossá tehetők és az áruk is igen kedvező, Linux-alapú gépekből nagyon hatékony tűzfalak építhetők.
A 2.2.x változatú Linux-rendszermag esetében a tűzfal telepítése az IP Chains segítségével történik, míg az új, 2.4.x rendszermagban ezt a feladatot az IP Tables tölti be. Egy valódi tűzfal összeállításának leírása meghaladná cikkünk kereteit; az érdeklődő olvasók további adatokat az ipchains HOWTO-ban (2.2.x magokra vonatkozóan) és a Linuxvilág honlapján (➲ http://www.linuxvilag.hu/static.phtml?file=hogyanok/ipchains/ipchains-hogyan-00), illetve *Paul „Rusty" Russell* Packet-Filtering HOWTO-jában (2.4.x magokhoz) találhatják meg. Mindkettő – bármely kereső segítségével – fellelhető az Interneten.
A tűzfal felépítése azonban még kevés, ha igazán biztonságos rendszert szeretnénk, a tűzfalat felügyelni is kell. Ebben a cikkben azt mutatjuk be, miképpen lehet felépíteni és használni az inside-control elnevezésű webalapú IP Chains figyelőrendszert.
A tűzfalfigyelő rendszerek két fő feladattal bírnak: ellenőrizniük kell, hogy rossz szándékú betörő nem próbál-e bajt keverni a belső hálón, valamint hogy a LAN-on belüli felhasználók nem élnek-e vissza valamelyik internetszolgáltatással.

Tűzfalbeállítási példa

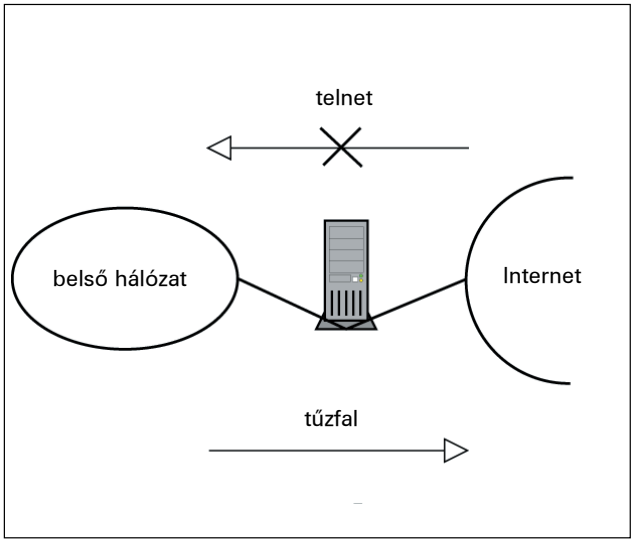
Lássunk egy nagyon egyszerű tűzfalbeállítást, amelyre ebben a cikkben a későbbiekben példaként fogunk hivatkozni.
Tegyük fel, hogy van egy 10.0.1.0/255.255.255.0 címtartományú belső hálózatunk; a linuxos átjáró, illetve tűzfal a 10.0.1.1 című csatolón keresztül éri el a belső hálót, és a 10.200.200.1 címen csatlakozik az Internethez (valójában egyik IP-cím sem nyilvános, így ez természetesen kitaláció). A tűzfal elindításának első lépése a két hálózati csatoló közötti átjáró felépítése:

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

A következőkben a naplózó tűzfalat építhetjük fel az IP Chains segítségével. Előbb kiürítjük az összes korábbi szabályt, majd engedélyezzük a hurokeszköz (loopback interface) csomagjait és az összes ICMP-csomagot:

```
ipchains -F
ipchains -A input -i lo -j ACCEPT
ipchains -A input -p ICMP -j ACCEPT
```

Ezt követően megakadályozzuk, illetve naplózzuk az Internetről a belső hálózat felé irányuló Telnet-protokollkérelmeket:



A példatűzfal beállítása

```
ipchains -A input -p TCP -s 0.0.0.0/0
       ➥-d 10.0.1.0/24 23 -l -j DENY
```

Ugyanakkor engedélyezzük és naplózzuk a HTTP-protokollt, amely a belső hálózatról az Internetre irányul:

```
ipchains -A input -p TCP -s 10.0.1.0/24
       ➥-d 0.0.0.0/0 80 -l -j ACCEPT
```

Befejezésül beállítjuk a megengedő szabályokat (permissive policies):

```
ipchains -P input ACCEPT
```

Tűzfalunk megakadályoz és naplóz minden bejövő Telnet-kapcsolatot, engedélyez és naplóz minden kimenő HTTP-kapcsolatot, illetve engedélyez minden mást (lásd a *fenti ábrát*). Egy ilyen beállítás természetesen túl engedékeny ahhoz, hogy komoly védelmet nyújtson, de jól szemlélteti, hogy mire képes az önműködő naplóvizsgáló parancsfájl.
A tűzfal a kimenetét általában a /var/log/syslog vagy a /var/log/messages fájlok valamelyikébe küldi. A következő paranccsal kideríthetjük, hogy a kettő közül melyik az igazi:

```
grep -q "Packet log" /var/log/syslog && echo yes
```

Ha a kimenet *yes*, akkor a /var/log/syslog a keresett fájl, ha a kimenet üres, akkor valószínűleg a /var/log/messages lesz az. Ezt is ellenőrizhetjük a

```
grep -q "Packet log" /var/log/messages && echo yes
```

paranccsal.

1. lista 2.4.x IP Tables parancssorozat

```
iptables -F
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p icmp -j ACCEPT
iptables -A INPUT -p tcp -d 10.0.1.0/24 \
         --dport 23 -j LOG \
         --log-prefix "Packet log: " \
         --log-level info
iptables -A INPUT -p tcp -d 10.0.1.0/24 \
         --dport 23 -j DROP
iptables -A INPUT -p tcp -s 10.0.1.0/24 \
         --dport 80 -j LOG \
         --log-prefix "Packet log: " \
         --log-level info
iptables -A INPUT -p tcp -s 10.0.1.0/24 \
         --dport 80 -j ACCEPT
iptables -P INPUT ACCEPT
```

Ha mindkét parancs üres kimenetet ad, akkor a tűzfal nem működik, vagy még semmilyen naplózott forgalom (a példánkban Telnet vagy HTTP) sem haladt át a tűzfalon.

2.4.x magok és az IP Tables

2.4.x-es rendszermag és IP Tables esetén a dolog egy árnyalatnyit bonyolultabb. Először is: nem szabad elfelejtenünk a magba az összes csomagszűrő szolgáltatást belefordítani, ideértve a naplózás (LOG) részt is. Másodszor cseréljük az IP Chainst IP Tablesre, majd a láncok neveit nagybetűsre (pl.: az inputból INPUT lesz). Ezt követően változtassuk meg a célpontok neveit (DENY DROP-ra). Végül a kapuszámokat is másféleképpen kell megadni. Az *1. lista* a fentebb bemutatott 2.2.x parancsok 2.4.x-es változatát mutatja be.

IP Chains naplóformátum

Most vizsgáljuk meg tűzfalunk /var/log/syslog bejegyzéseit:

```
Jun 12 16:15:54 myfirewall kernel:
       ➥Packet log: input DENY eth1 PROTO=6
       ➥212.65.214.2:34251 10.0.1.2:23
       ➥L=52 S=0x10 I=24016 F=0x4000 T=53 SYN (#38)
```

Ezek szerint június 12-én, délután negyed ötkor a tűzfal (mely a nem túl ötletes myfirewall nevet viseli) elutasított és naplózott egy az eth1 hálózati csatolóra érkező (azaz az Internetről jövő) TCP-protokollú csomagot, mely a 212.65.214.2 címről (és a 34251 kapuról) indult, a 10.0.1.2 címre (a 23-as, azaz a Telnet-kapura) irányult, és 52 bájt hosszúságú volt. A többi részletet nyugodtan átugorhatjuk, kivéve egyet: a SYN azt jelenti, hogy ez a csomag volt a kapcsolat első csomagja. A gyakorlatban ez az adat nagyon hasznos lehet, mivel ennek alapján lehet megkülönböztetni a már létező kapcsolatok csomagjait (amiket esetleg a belső hálózat kezdeményezett), és azokat a csomagokat, amelyek új kapcsolatot próbálnak létrehozni az Internetről a belső háló felé. Általában engedélyezzük a *reply* (válasz) csomagokat (azaz azokat amelyeknek nincsen SYN-jelük), viszont tiltjuk a SYN-csomagokat, mert ezek azt jelentik, hogy valaki odakint valamelyik belső hálózati géppel kapcsolatot szeretne felépíteni. Természetesen a tűzfal állapotát a naplófájlok megfelelő bejegyzéseinek végigböngészésével is meg lehet vizsgálni, de ez csak akkor célszerű, ha valaki csak néhány gyanús kinézetű csomagot naplóz. Például némelyik általam beállított tűzfalnál úgy döntöttem, hogy naplózni fogom az összes olyan csomagot, amely az Internetről a 31337-es számú kapu irányába megy a belső hálózat számítógépein, mert ez a BackOrifice alapértelmezés szerint használt kapuja. Ha viszont valaki a tűzfalról kimutatást is szeretne készíteni, valószínű, hogy a naplófájl mérete meghaladja a napi öt megabájtot. Ilyen esetekben a naplófájl kézi vizsgálata már nem járható út. Ekkor nyújthat segítséget az önműködő naplófigyelés.
Ha 2.4.x mag tűzfalnaplókat vizsgálunk, a formátum más lesz:

```
Jun 12 16:15:54 myfirewall kernel
   ➥Packet log: IN=eth1 OUT=
   ➥MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00
   ➥SRC=212.65.214.2 DST=10.0.1.2 LEN=52 TOS=0x10
   ➥PREC=0x00 TTL=64 ID=0 DF PROTO=TCP SPT=34251
   ➥DPT=23 WINDOW=11592 RES=0x00 SYN URGP=0*
```

A minket érdeklő mezők: az SRC (forrás IP-cím), a DST (cél IP-cím), a SPT (forráskapu), a DPT (célkapu), illetve a SYN-kapcsoló megléte vagy hiánya.

Az inside-control szerkezete

A naplókövető felépítéséhez Perlt fogok használni. Természetesen nem ez az egyetlen lehetséges választás, sőt, ha valaki a legnagyobb sebességre törekszik, akkor tulajdonképpen jobban teszi, ha valamilyen fordítóval működő (compiled) nyelvet választ. Amikor ugyanezt a parancsfájlt C++ nyelven újraírtam, körülbelül százszázalékos sebességnövekedést tapasztaltam. Az inside-control parancsállomány egy főértelmező ciklusból és egy HTML-adatmegjelenítő ciklusból áll. Mivel ez CGI parancsfájl, olyan webkiszolgálón kell elhelyezni, amelyet felkészítettek CGI programok futtatására.
A kód, amint majd alább kiderül, az érthetőség érdekében feláldoz valamennyit a használhatóságból, továbbá kihagy pár hasznos részletet, például a hibaellenőrzést is. Többek között azt sem ellenőrzi, hogy a fájlmegnyitás sikeres volt-e, mielőtt olvasna abból.
Azt is érdemes megemlíteni, hogy a lenti kód 2.2.x-es magok csomagnaplózási formátumához lett igazítva. A 2.4.x változathoz illesztés a korábbiakban már megismert csomagnapló-példa alapján magától értetődik.

A fő értelmező ciklus

Először is megnyitjuk a naplófájlt és alapértéket adunk néhány változónak (akik RedHatet használnak, azok a /var/log/messages fájlt írják a /var/log/syslog helyére):

```
#!/usr/bin/perl
open(LOGFILE, "/var/log/syslog");
$firstdate = "";
$date = "";
$total_traffic = 0;
```

Most végiglépkedünk a naplófájl minden során:

```
while ( <LOGFILE> ) {
```

Kihagyjuk az összes bejegyzést, ami nem a tűzfalhoz tartozik:

```
next unless /Packet log/;
```

Értelmezzük a sort:

```
chomp;
@log = split;
($month,$day,$time,$policy,$proto,$ipsource,$ipdest,
     ➥$tot_len) = @log[0,1,2,8,10,11,12,13];
```

Ezután kiszámoljuk a dátumot és mentjük a napló első időpontját. Ahogy továbbhaladunk, a pillanatnyi keltezést mindig a legutolsó dátumként tároljuk, így az utolsó lépés után a *lastdate* változó a naplóban található utolsó dátumot tartalmazza majd:

```
$date = $day . " " . $month . " " . $time;
if (length($firstdate) == 0) {
  $firstdate = $date;
}
$lastdate = $date;
```

Beolvassuk a protokolltípust, a forrás IP-címet, a forráskaput, a cél IP-címet, a célkaput és a csomag hosszát:

```
$proto = substr($proto, -1);
($ips, $ports) = split ":", $ipsource;
($ipd, $portd) = split ":", $ipdest;
($flush, $packetlen) = split "=", $tot_len;
```

Tároljuk a cél IP-címet egy karakterláncban, és vessük össze a forrás IP-címmel, így az adatmegjelenítő ciklusban majd végig tudjuk nézni a forrás IP-címeket és kikereshetjük, hogy mely címekhez kapcsolódtak:

```
unless ( $sourcedest{$ips} =~ /$ipd/ ) {
  $sourcedest{$ips} =
            ➥$sourcedest{$ips} . $ipd . " ";
}
```

Megszámoljuk a forrás IP-címbejegyzéseket:

```
++$source{$ips};
```

és összegezzük a teljes forgalmat:

```
$total_traffic += $packetlen;
```

Utoljára összegezzük a kiszolgálónkénti forgalmat is:

```
$traffichost{$ips} += $packetlen;
}
```

Figyeljük meg, hogy egyáltalán nem használtuk fel az összes begyűjtött adatot (például szó sem esett a kapukról), tehát rengeteg hely akad még a bővítgetésre.

Az adatmegjelenítő ciklus

Elsőként egy mutatós honlap-fejlécet rakunk ki, amint az a *2. listában* látható. Végiglépkedünk a rendezett forrás IP-címeken, majd kiírjuk a forrás IP-címet, az arról a címről érkezett csomagok számát, illetve az arról az IP-címről származó forgalmat (bájtban mérve):

```
for (sort keys %source) {
  print "<TR><TD>$_</TD> ";
  print "<TD>$source{$_} </TD>\n";
  print "<TD>$traffichost{$_} bytes</TD>\n";
```

Mivel korábban tároltuk, ki tudjuk írni azokat a cél IP-címeket, amelyekre a pillanatnyi forrás-IP-ről csomagok továbbítódtak:

```
$tmp1 = $sourcedest{$_};
if (length($tmp1) gt 0) {
  print "<TD>\n";
  @lt1 = split " ", $tmp1;
```

2. lista Weblap-fejléc

```
print "Content-type: text/html\n\n";
print "<HTML>\n";
print "<HEAD><TITLE>ipchains Log
       ➥Scanner</TITLE></HEAD>\n";
print "<BODY bgcolor=\"#000080\"
       ➥text=\"#FFFFFF\" link=\"#BA3E8F\"
       ➥vlink=\"#BD66DE\"
       ➥alink=\"#FF0000\">\n";
print "<H1><i>inside-control</i></H1>\n";
print "<p>Log searched for network access from ";
print "<font color=\"#FFAAAA\">
       ➥$firstdate</font> to ";
print "<font color=\
       ➥"#FFAAAA\">$lastdate</font>";
print ".\n    </p>\n";
print "<H2>Total Traffic Volume:
       ➥<font color=yellow>";
print "$total_traffic bytes</font></H2>\n";
print "<hr>\n";
print "<center>\n";
print "<TABLE border=2>\n";
print "<TR><TH>Source IP</TH><TH>Packets</TH>";
print "<TH>Traffic from this host</TH>";
print "<TH>Hosts contacted</TH>";
print "</TR>\n";
```

```
  for(sort @lt1) {
    printf "$_ <br>\n";
  }
  print " </TD>\n";
}
print " </TR>\n";
}
```

Végül kiírjuk a HTML végét:

```
print "</TABLE>\n";
print "</center>\n";
print "</BODY></HTML>\n";
```

A letölthető inside-control

Az általam tényegesen megvalósított inside-control parancsállomány az itt bemutatottnál sokkal gazdagabb lehetőségekkel rendelkezik. A program a ➲ http://www.iris-tech.net/hdsl-fw címről tölthető le. Legfontosabb kiegészítő tulajdonsága: tetszőleges nevek megjelenítésére képes IP-címek helyett a *Source IP* (forrás-IP) oszlopban. Ezt egy nagyon egyszerű szöveges adatbázis segítségével tudja megvalósítani, amely az IP-számokat nevekhez rendeli. A fájl formátuma azonos a /etc/hosts fájl formátumával, így akár ezt a fájlt is használhatjuk, amennyiben az a belső hálónak megfelelően lett feltöltve. Az *IP nevek* adatbázis pontos helyét a megfelelő változó ($useripdb) átírásával adhatjuk meg a parancsfájl elején.

Egy keresőrendszert is beépítettek, amely lehetővé teszi, hogy egy adott forrás IP-címre (vagy az *IP név* adatbázisban megtalálható megfelelő névre) rákeressünk a naplóban. A kereső űrlap akkor jelenik meg, ha a CGI-t értékek nélkül hívjuk meg a böngészőből. Az értékátadás a `GET` metódussal történik.

A főciklus tartalmaz továbbá néhány adatérvényesítési eljárást (a mag nem mindig képes helyesen naplózni, különösen akkor, ha kis memória- vagy processzorteljesítmény mellett fut), és néhány kapufüggő adatot is tárol. Végezetül a parancsfájl webes csatolófelület nélkül is meghívható. Egyszerűen csak adjunk át valamilyen értéket az inside-controlnak, és a HTML-kimenet helyett hagyományos kimenetet kapunk. A keresendő forrás-IP karaktersorozatát (vagy az *IP név* adatbázisban található megfelelő nevet) a programnak a `-t` kapcsolóval adhatjuk át.

Megjegyzések és figyelmeztetések

E cikk írójának az volt a célkitűzése, hogy néhány tervezési alapelvet mutasson be és tanácsokkal szolgáljon – semmiképpen nem az, hogy a naplófigyelési gondokra előrecsomagolt megoldást szolgáltasson. Számos dolog akad, amiben az inside-control parancsállomány fejlesztésre szorul, például a teljesítmény és a biztonság területén.

A következőkben néhány szót ejtenék az inside-control jellegzetességeiről, különös tekintettel a biztonságot érintő kérdésekre.

Ahhoz, hogy a CGI a számítógép naplófájljait (/var/log/syslog vagy /var/log/messages) olvashassa, mindenki számára olvashatóvá kell tenni. Ez a `chmod +r /var/log/syslog` paranccsal érhető el. Csakhogy ez nem túl biztonságos megoldás, hiszen mindenkinek jogot ad a naplófájlok olvasgatására. Lényegesen jobb lenne rávenni a webkiszolgálót, hogy az inside-controlt egy adott csoportjogosultsággal futtassa, és ezután a naplófájlokat ehhez a csoporthoz rendelje.

A cikk elolvasása után bizonyára sokan jutnak arra a következtetésre, hogy a tűzfalnak egy webkiszolgálót is szükséges futtatnia, hiszen az inside-controlnak a tűzfal naplófájljait olvasni kell tudnia. Webkiszolgálót rakni egy tűzfalra valójában óriási biztonsági rés: eszményi esetben a tűzfal semmiféle démonszolgáltatást nem futtat és a teljes karbantartást konzolon keresztül végzik. Ha távoli felügyelet szükséges, az egyetlen szóba jöhető szolgáltatás, amely egy tűzfalra feltehető, az SSH, azaz a biztonságos héjprogram (secure shell) program.

Az inside-control futtatása még ilyenkor is megoldható a belső hálózaton egy külön webkiszolgáló felállításával, amely egyúttal a tűzfal syslog kiszolgálójaként is működik.

A tűzfal naplója rövid idő alatt könnyedén feltöltheti a teljes lemezterületet. Hogy elkerüljük tűzfalunk merevlemezének lebénulását (ami az internetkapcsolatok leállásához vezethet), a naplózandó forgalom mennyiségétől függően megfelelően nagy naplófájlhelyet kell biztosítanunk. A nagy adatmennyiségű szolgáltatások (azaz általában a HTTP-, FTP-, SMTP-, NetBIOS-, LPD- és az adatbázisszolgáltatások) esetében egy második, legalább 20 GB méretű merevlemez használatát javaslom, amelyen mindössze egyetlen, a /var/log-ra csatolt lemezterület létezzen.

Befejezésképpen: még rengeteg lehetőség nyílik fejlesztésre az egész parancsfájlban, különösen a főciklusban. Sokkal több adatot lehetne felhasználni az egyes sorokból, mint ahogy ebben a rövid példában tettük. Ennek ellenére az sem baj, ha nem mutatunk meg túl sok részletet; máskülönben az egész önműködő naplófigyelés értelmetlenné válik. Ha minden elérhető részletet megjelenítenénk, a gyanús bejegyzéseket kezelhetetlenül nagy mennyiségű forgalmi naplóban kellene megkeresnünk.

Leo Liberti
az olaszországi IrisTech cég műszaki igazgatója, amely vásárlóit webalapú alkalmazásokkal és sokféle egyéb elektronikai szolgáltatással látja el. Szabadidejét annak szenteli, hogy annyi étteremben étkezzék, amennyiben csak tud.

Utazás a Postfix körül (2. rész)

Az előző részben megismertük a Postfix szerkezetét, a levélfogadást és -továbbítást. Az elmélet után hasznos, ha tudásunkat a gyakorlatban is kipróbáljuk.

Első lépésként – mint mindig – telepítjük az alkalmazást. Erre két lehetőségünk is nyílik: vagy egy csomagkezelőt használunk, vagy pedig forráscsomagból rakjuk fel az alkalmazást. A csomagkezelővel telepített változatok előnyei közé sorolhatjuk, hogy sok beépített foltot (patch) tartalmaznak vagy az alapfordítástól eltérő kapcsolókkal fordították (szerepel benne például a `MySQL`- vagy `OpenLDAP`-támogatás). Ekkor ugyanis nem kell feltenni az adott program `header` és `include` fájljait, amelyek a terjesztésekben általában a *`csomagnév`*`-dev` csomagban szerepelnek, például: `openldap-dev`. Másik előnye, hogy így szükségtelen fordítóprogramokat telepíteni gépünkre, ami külön hasznos, ha csak egy kicsit is gondolunk a biztonságra. Elvégre minden bit segít! A forráscsomagokból telepített változatok kedvező tulajdonsága, hogy az alkalmazások közül mindig a legfrissebbet használhatjuk, valamint a C fordító segítségével finomhangolni tudjuk a binárist, illetve csak olyan támogatás kerül a binárisba harmadik fél programjához, amilyet szeretnénk. Így a `postfix` telepítésekor nem kell felraknunk például az `openldap` csomagot, mert nem igényeljük, hogy az LDAP-alapú címtárszolgáltatásokat használja. Megoldás lehet, ha mi magunk készítünk deb vagy rpm csomagot a forrásból, esetleg egy másik gépen fordítjuk le a forrást, és a célgépre a binárisokat csupán átmásoljuk. Mi azonban most a legegyszerűbb megoldásokhoz folyamodunk: először megnézzük a csomagkezelővel történő telepítést a Debian- és RedHat-rendszeren.

Debian (vagy deb-alapú):

```
# apt-get install postfix
```

vagy

```
# dpkg -i postfix_0.0.19991231pl11-1.deb
```

RedHat (vagy rpm-alapú):

```
# rpm -Uvh postfix-20010202-4.i386.rpm
```

A Debian Potatóban láthatóan még a régebbi változat szerepel, és a RedHat sem tud lépést tartani *Wietse Venemával*. A `postfix` legújabb változata a `snapshot-20010714`. A forrásból ezt fogjuk a lehető legegyszerűbb módon telepíteni. Először is látogassunk el a `postfix` honlapjára, a ➲ http://www.postfix.org címre, keressük meg a *Downloads* pontot és válasszunk kiszolgálót. Két lehetőség áll előttünk: vagy az utolsó hivatalos – jelenleg a `Postfix Release 20010228` névre hallgató –, vagy a pillanatfelvétel-változatot (snapshot) – a cikkírás időpontjában `Postfix Snapshot 20010714` – töltjük le. Ha ezt megtettük, csomagoljuk ki a forrást a következő módon (feltételezzük, hogy a fájl neve `snapshot-20010714.tar.gz`):

```
# tar xvfz snapshot-20010714.tar.gz -C
      ➥/usr/local/src
```

Ez az /usr/local/src könyvtár alatt egy snapshot-20010714 nevű könyvtárat hoz létre. Amikor ez is megtörtént, akkor lépjünk be a könyvtárba és olvassuk el a számunkra érdekes README fájlokat. Mivel jelenleg az alaprendszert telepítjük, ezt a lépést kihagyhatjuk. A következő lépés a forrás lefordítása bináris állománnyá. A műveletet a szokásostól eltérően nem a `configure` paranccsal kezdjük, hanem egyenesen a `make` parancshoz fordulunk – mindenfajta kapcsoló és beállítás nélkül:

```
# make
```

Most megvárjuk, amíg az összes forrásfájl lefordítódik, majd a `postfix` futtatásához szükséges felhasználókat adjuk hozzá a rendszerhez:

```
# useradd -d /var/spool/postfix -s /bin/false
      ➥postfix
```

Ezzel hozzáadtuk a rendszerhez a `postfix` nevű felhasználót, aki nem tud bejelentkezni, saját könyvtárnak pedig a program által használt könyvtárat adtuk meg, bár megadhattunk volna nem létező könyvtárat és héjat is (a /bin/false is csak akkor létezik, ha szerepel a /etc/shells fájlban). Mivel nem mindenki által írható (world-writeable) sort fogunk használni a nem SMTP-kapcsolaton keresztül érkező levelek fogadására, egy `maildrop` csoportot is létre kell hoznunk:

```
# groupadd maildrop
```

Ezután kiadjuk a

```
# make install
```

parancsot, ez elkezdi a telepítést. Ezzel egyenértékű a következő parancs:

```
# sh INSTALL.sh
```

ugyanis a make install is ezt a parancsállományt hívja meg. Ekkor kérdésekkel kerülünk szembe, ezek nagy részére alkalmazhatjuk a felkínált lehetőségeket. Nézzük sorban:

install_root: [/] – Arra kíváncsi, hogy mi lesz a gyökérkönyvtár. A Unix-alapú rendszereken ez a / (perjel), azaz a gyökérkönyvtár.

tempdir: [/usr/local/src/snapshot-20010714] – Ez adja meg, hogy a telepítés közben szükséges átmeneti tárolóhely hol legyen.

config_directory: [/etc/postfix] – A beállítófájlok helye.

daemon_directory: [/usr/lib/postfix] – A Postfixet alkotó démonok helye.

command_directory: [/usr/sbin] – Hol lesznek azok a parancsok, amelyek a Postfixet vezérlik.

queue_directory: [/var/spool/postfix] – A Postfix sorai, ahol a levelek feldolgozás közben tárolódnak.

sendmail_path: [/usr/lib/sendmail] – A Postfix Sendmail programja.

newaliases_path: [/usr/bin/newaliases] – A Postfix Newaliases programja (az utóbbi kettő használatáról még később szólunk).

mailq_path: [/usr/bin/mailq] – A mailq program helye a rendszerben.

mail_owner: [postfix] – A Postfix sorainak (queue) tulajdonosa. Ha az előbb nem Postfix-felhasználót adtuk hozzá, akkor annak a nevét kell ide beírni, akit héj és saját könyvtár nélkül létrehoztunk.

setgid: [no] – Mindenki által írható (world-writeable) könyvtár, amelybe a helyileg – a Postfix Sendmail binárisán keresztül – küldött levelek kerülnek vagy pedig a könyvtár által megadott csoporthoz tartozik. Ha az adott csoport tudja csak írni a könyvtárat, akkor úgynevezett csoportjoggal (setgid) rendelkező program alkalmazására nyílik lehetőségünk. Ehhez a Sendmail program a Postdrop programot használja a levelek beillesztésére. Ajánlott a szigorított postázás, ezért adtuk hozzá a rendszerhez a telepítés elején a `maildrop`-csoportot. Tehát itt adjuk meg a maildrop-csoportot, a képernyőn pedig a következőket kell látnunk:

```
setgid:[no] maildrop
manpages:[/usr/local/man]
```

Ha mindezekkel végeztünk, a telepítés befejeződik és a

2. lista Az /etc/shells tartalma

```
# /etc/shells: valid login shells
/bin/bash
/bin/csh
/bin/sh
/usr/bin/ksh
/usr/bin/tcsh
/bin/sash
/bin/zsh
/bin/false
```

Az /etc/adduser.conf vonatkozó része

```
# /etc/adduser.conf: `adduser' configuration.
# See adduser(8) and adduser.conf(5) for full
# documentation.
# The DSHELL variable specifies the default
# login shell on your
# system.
DSHELL=/bin/false
```

```
# postfix start
```

paranccsal tudjuk indítani az alkalmazást. Ne ijedjünk meg! Amikor legelőször indítjuk el a rendszert, a `postfix` az addig nem létező alkönyvtárakat a neki fenntartott helyen hozza létre – ez alapértelmezésben a /var/spool/postfix –, ezek alkotják majd a sorokat. Ezt a folyamatot mutatja be az *1. lista* (a 15. CD Magazin/Postfix könyvtárában található).

A `postfix` minden indulásnál ellenőrzi, hogy léteznek-e a számára szükséges könyvtárak. Amennyiben nem, akkor létrehozza őket. Ha elindult a rendszer és egyetlen naplófájlban sem jelent meg hiba – főleg a /var/log/messages és /var/log/mail.log fájlokat nézzük át –, akkor állítsuk le a rendszert addig, amíg be nem állítjuk. Nézzük meg, hogy mi lesz a célunk:

- A felhasználókat felvesszük a rendszerre, de héj nélkül.
- Beállítjuk a kiszolgáló tulajdonságait: a nevét, illetve hogy melyik névtartománynak (domain) nyújt szolgáltatást.

Az első feladatot könnyű végrehajtani, Debian-rendszeren meg kell keresni a */etc/shells* fájlt és egy sorba írjuk bele a következőt: `/bin/false` – ezzel felvettük az érvényes héjak közé. Elértük, hogy bár a felhasználó szerepel a rendszeren, tehát levelet is tud fogadni, nem tud a rendszerre belépni. A második lépés, hogy átszerkesztjük a */etc/adduser.conf* fájl tartalmát. Keressük meg a `DSHELL=/bin/bash` sort és cseréljük ki azt az alábbira: `DSHELL=/bin/false`

Ha ezzel végeztünk, elkezdhetjük beállítani a rendszert. A démonok viselkedését a master.cf fájl tartalma határozza meg, mely a sorozatunk előző részében ismertetett `master` démont irányítja. A kiszolgáló tulajdonságait alapértelmezésben a /etc/postfix/main.cf-ben találjuk meg, most ezzel foglalkozunk.

Első dolgunk, hogy beállítsuk, hol is találhatók meg a levéltároló sorok, a programok és a démonok, valamint a jogosultságok. Ha szabályszerűen telepítettünk, akkor ezekhez a sorokhoz nem is kell nyúlnunk:

```
# a sorok helye
queue_directory = /var/spool/postfix
# a vezérlő programok helye
command_directory = /usr/sbin
# a postfix démonjainak helye
daemon_directory = /usr/lib/postfix
# ki a sorok tulajdonosa, azaz ki kezeli a sorokat
mail_owner = postfix
```

A következő sornál érdemes elidőzni. Lehet, hogy felhasználóink egyéni fájlokat hozhatnak létre saját könyvtárukban. Ha ebben egy *.forward* fájlt helyeznek el, akkor a bejövő levelet az itt megadott címre tudják továbbítani vagy egy programnak továbbadni a rendszeren. Minket ez utóbbi érdekel. A következő értéknél beállított felhasználó jogosultságaival fog végrehajtódni a *.forward* fájlon keresztül meghívott program. Ezt biztonsági megfontolásokból állítsuk minél alacsonyabb szintre. A *nobody* felhasználó megfelelő erre a szerepre, elvégre nem szeretnénk, hogy egy olyan parancsfájlnak adja át a felhasználó a levelet, ami például a levél tartalmát hajtja végre – abban pedig az `rm -rf /` szerepel és mindezt a rendszergazda jogaival teszi –, igaz? A `postfix`-et vagy más kiemelt jogosultságú felhasználót, illetve felhasználónevet ne állítsunk be!

```
default_privs = nobody
```

Ekkor minden héjprogram a nobody jogosultságaival fog futni, ami a *.forward* fájlból hívódik meg.

A következő sor adja meg a gép teljes nevét, például:

```
myhostname= mailserver.linuxvilag.hu
```

A mydomain érték pedig a tartományt adja meg:

```
mydomain= linuxvilag.hu
```

Ha helyileg (rendszeren belül) keletkezik egy levél, programból vagy felhasználó által, akkor a myorigin értékét veszi alapul. Itt már látható, hogy a Postfix az ismert értékeket más paramétereknél változóként képes kezelni. Itt a myorigin értéke a linuxvilag.hu lesz:

```
myorigin = $mydomain
```

Az `inet_interfaces = all` egyelőre maradhat a helyén, ugyanis ez határozza meg, hogy mely IP-címekre fogadunk el levelet. Az all mindegyiken elfogadja az SMTP-kapcsolatot. A következő számban már néhány trükköt is elsajátíthatunk.

A következő érték adja meg, hogy mely gépnevekhez tartozó leveleket dolgozzon fel a rendszer. Ne soroljuk fel a virtuális tartományokat – azok majd egy későbbiekben tárgyalandó érték feladatait képezik, amelynek használatával a következő számban ismerkedünk meg.

```
mydestination =   $myhostname,
                  ➥localhost.$mydomain
```

A tökéletes beállítás az, ha gépünknek csak ez az egy neve van.

```
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

A fenti két adattábla közül az elsőt csak a Postfix, a másodikat más programok is alkalmazhatják. Itt éppen közös adatbázist használnak. Ha az alias fájlon bármit változtatunk, akkor ki kell adni a `newaliases` parancsot.

Az utolsó fontos érték, ami ahhoz szükséges, hogy egyszerű, de mégis biztonságos és gyors levelezőkiszolgálót kapjunk, a mynetworks. A mynetworks értékbe kerülnek azok a hálózatok, ahonnan a küldés engedélyezett.

```
mynetworks = 127.0.0.0/8 192.168.1.0/24
```

Ez abban az esetben igaz, ha a helyi hálózat – amely előtt a kiszolgáló található – a 192.168.0.0-s címeket használja. Fontos megjegyezni, hogy a 127.0.0.0/8 címtartományt, tehát a saját gépet se hagyjuk ki!

A változtatások megtétele és a Postfix elindítása után a rendszer máris használható. A következő részben egy kisebb, nem állandó kapcsolattal rendelkező iroda szükségleteinek kielégítésével és virtuális tartományok kezelésével ismerkedhetünk meg.

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.

Könnyű álom (8. rész)

Hálózati forgalom vizsgálata.

Amikor a rendszer nem úgy viselkedik, ahogy elvárnánk, vagy egyszerűen nem tudjuk, hogy mi történik a hálózaton, hasznos segédeszköz lehet a `tcpdump` program. A gépünket érintő vagy a hálózati területen átmenő forgalom lehallgatásával és megjelenítésével nélkülözhetetlen segédeszközzé válik a hálózati hibák felderítésében. Írásunkban a program használatának fortélyait ismertetjük, valamint a cikksorozatunk korábbi részében [1.] már bemutatott TCP/IP protokoll működését követjük nyomon.

Hogyan működik a tcpdump?

A tcpdump működéséhez elengedhetetlenül szükséges, hogy a program a rendszer által vett vagy a hálózati csatoló által érzékelhető összes keretet megkapja. A keretek vételét a rendszermag biztosítja számunkra. Ethernethálózat esetén a kártyák általában csak a nekik szóló, valamint a csoportcímzett (multicast) és az üzenetszórt (broadcast) csomagokat veszik. A kártyák az összes keret vételére is lehetőséget adnak, ezt hívjuk *lehallgató* (promiscuous) módnak. Fontos azonban megjegyezni, hogy a lehallgató mód egy terhelt hálózaton észrevehető további terhelést jelenthet a számítógépnek. A tcpdump több Unix-alapú rendszeren is működőképes, a rendszermag-támogatás viszont eltérő. A System V-alapú rendszereknél a keretek beolvasására a DLPI (Data Link Provider Interface) használatos, míg a BSD-alapú rendszerek a BPF-et (BSD Packet Filter) támogatják. A Linux egyik szabványnak sem felel meg, e célra saját alrendszert vesz igénybe. Hordozhatóság céljából a programból a szorosan a rendszermaghoz kapcsolódó keretbeolvasó szolgáltatásokat leválasztották és a `libpcap` csomagban helyezték el. Mint az előzőekben említettük, a lehallgató mód komoly terhelést jelenthet a számítógépnek. A helyzetet súlyosbítja, hogy a vett kereteknek a rendszermagból a felhasználói programba is át kell jutniuk. Ha csupán jól meghatározott csomagokat szeretnénk látni, akkor a szűrést a felhasználói programban végezni erőforrás-pazarlás. Emiatt

A tcpdump szűrő alapelemei

Alapelem	*Jelentés*
dst host <gép>	Igaz, ha az IP-csomag célja a megadott gép.
src host <gép>	Igaz, ha az IP-csomag forrása a megadott gép.
host <gép>	Igaz, ha az IP-csomag forrása vagy célja a megadott gép.
ether dst <cím>	Igaz, ha a csomag ethernet-célcíme egyezik.
ether src <cím>	Igaz, ha a csomag ethernet-forráscíme egyezik.
ether host <cím>	Igaz, ha a csomag ethernet-forrása vagy célja a megadott cím.
gateway <gép>	Igaz, ha az IP-csomag a gépet mint átjárót használja. Megegyezik az „ether host <cím> and not host <gép>” kifejezéssel, ahol is a cím a gép ethernet címe míg a gép az IP-címe.
dst net <háló>	Igaz, ha az IP-csomag célja a megadott hálózaton helyezkedik el.
src net <háló>	Igaz, ha az IP-csomag forrása a megadott hálózaton helyezkedik el.
net <háló>	Igaz, ha az IP-csomag forrása vagy célja a megadott hálózaton helyezkedik el.
net <háló> mask <maszk>	Igaz, ha az IP-csomag forrása vagy célja a megadott net <háló>/<hossz> hálózaton helyezkedik el. Itt a hálózat maszkját kifejtve adjuk meg.
dst port <kapu>	Igaz, ha az IP-csomag célkapuja a megadott.
src port <kapu>	Igaz, ha az IP-csomag forráskapuja a megadott.
port <kapu>	Igaz, ha az IP-csomag forrás vagy célkapujával megegyezik.
less <hossz>	Igaz, ha a csomag hossza kisebb vagy egyenlő a megadott értéknél.
greater <hossz>	Igaz, ha a csomag hossza nagyobb vagy egyenlő a megadott értéknél.
ip proto <proto>	Igaz, ha a keret egy proto-protokoll azonosítójú IP-csomagot hordoz.
ether broadcast	Igaz, ha a keret ethernet-címe üzenetszórt.
ip broadcast	Igaz, ha a csomag IP-címe valamilyen üzenetszórt cím.
ether multicast	Igaz, ha a keret ethernet-címe csoportcímzett.
ip multicast	Igaz, ha a csomag IP-címe valamilyen csoportcímzett cím.
ether proto <proto>	Igaz, ha az ethernet-keret proto-protokollt hordoz.
ip, arp, rarp	Igaz, ha az ethernet-keret protokollja a megadott. Rövidítések az „ether proto <proto>” formulára.
tcp, udp, icmp	Igaz, ha az IP-protokoll azonosítója a megadott. Rövidítések az „ip proto <proto>” formulára.

a) ICMP-példa

```
(1) 16:13:26.681407 P dolphin > cheetah: icmp: dolphin tcp port 1234
      unreachable [tos 0xc0]  (ttl 255, id 25596)
(2) 16:13:29.191709 P cheetah > dolphin: icmp: echo request (ttl 64, id 115)
(3) 16:13:29.191709 P dolphin > cheetah: icmp: echo reply (ttl 255, id 25595)
```

b) ARP-példa

```
(1) 15:53:50.474273 B 0:c0:c:3:9:4d Broadcast arp 60: arp who-has dolphin tell cheetah
(2) 15:53:50.474273 P 0:4f:4c:3:35:ea 0:c0:c:3:9:4d arp 60: arp reply dolphin is-at 0:4f:4c:3:35:ea
      (0:c0:c:3:9:4d)
```

c) UDP-példa

```
( 1) 16:10:31.076878 B 0.0.0.0.bootpc > 255.255.255.255.bootps: xid:0xa1bcf3fc vend-rfc1048
       DHCP:DISCOVER
( 2) 16:10:31.076878 < dolphin.1313 > hawk.domain: 23418+ A? cheetah.home. (30)
( 3) 16:10:31.076878 > hawk.domain > dolphin.1313: 23418* 1/1/1 A cheetah (81)
( 4) 16:10:31.086877 < dolphin.1313 > hawk.domain: 23419+ A? hawk.home. (27)
( 5) 16:10:31.086877 > hawk.domain > dolphin.1313: 23419* 1/1/1 A hawk (73)
( 6) 16:10:31.086877 P dolphin.bootps > cheetah.bootpc: xid:0xa1bcf3fc Y:cheetah S:dolphin ether
       0:c0:c:3:9:4d vend-rfc1048 DHCP:OFFER SID:dolphin LT:3232235520 RN:1616117760 RB:2828206080
       SM:255.255.255.0 DG:hawk NS:hawk HN:"cheetah" DN:"home" BR:10.1.2.255 [tos 0x10]
( 7) 16:10:31.136873 B 0.0.0.0.bootpc > 255.255.255.255.bootps: xid:0xa2bcf3fc vend-rfc1048
       MSZ:9218 PR:SM+DG+NS+DN+BR+HN+LOG+LPR+NTP+XFS+XDM HN:"cheetah.andrews^@" LT:3232235520
       DHCP:DISCOVER
( 8) 16:10:31.136873 P dolphin.bootps > cheetah.bootpc: xid:0xa2bcf3fc
       Y:cheetah S:dolphin ether 0:c0:c:3:9:4d vend-rfc1048 DHCP:OFFER SID:dolphin LT:3232235520
       SM:255.255.255.0 DG:hawk NS:hawk DN:"home^@" BR:10.1.2.255 HN:"cheetah" [tos 0x10]
( 9) 16:10:31.186868 B 0.0.0.0.bootpc > 255.255.255.255.bootps: xid:0xa2bcf3fc vend-rfc1048
       MSZ:9218 PR:SM+DG+NS+DN+BR+HN+LOG+LPR+NTP+XFS+XDM HN:"cheetah.andrews^@" LT:3232235520
       DHCP:REQUEST SID:dolphin RQ:cheetah
(10) 16:10:31.186868 P dolphin.bootps > cheetah.bootpc: xid:0xa2bcf3fc
       Y:cheetah S:dolphin ether 0:c0:c:3:9:4d vend-rfc1048 DHCP:ACK SID:dolphin LT:3232235520
       SM:255.255.255.0 DG:hawk NS:hawk DN:"home^@" BR:10.1.2.255 HN:"cheetah" [tos 0x10]
```

a BPF-megfelelő alrendszereket egy egyszerű szűrőrésszel is ellátták. A szűrő használata esetén a felhasználói programba már csak a kért csomagok jutnak el. A nem BPF-megfelelő rendszereknél a szűrés a pcap eljáráskönyvtáron belül valósul meg. Ahhoz, hogy a program használható legyen, a rendszermagot a `CONFIG_PACKET` lehetőséggel kell fordítani, és a támogatás modulba helyezésekor az *af_packet.o* állományba fog kerülni. Ha azonban a rendszernaplóban az alábbi üzenetet látjuk, akkor a libpcap számára szükséges támogatás hiányzik a rendszermagból:

```
modprobe: can't locate module net-pf-17 socket:
Address family not supported by protocol
```

A 2.2 rendszermag-sorozat óta a Linux is tartalmaz a BPF-nek megfelelő szűrőalrendszert, amit Linux Socket Filternek neveznek. Ezt a `CONFIG_FILTER` lehetőség kiválasztásával fordíthatjuk a rendszermagba.

A tcpdump használata

Először is nézzünk egy egyszerű példát a tcpdump használatára: kapcsolódjunk a `telnet` paranccsal egy számítógépre (a *2. c listán* a *cheetah* gépről kapcsolódunk a *dolphin* SSH-kapujára), de előtte indítsuk el a tcpdump programot (ha a program a hálózatról vesz csomagokat, akkor elindításához rendszergazdai jogosultságok szükségesek). A tcpdump minden sorba egy vett keretet ír (ez azonban a képernyő szélessége miatt törik). A program kimenete függ a vizsgált protokolltól. Írásunkban TCP-, UDP- és ICMP-protokollokat vizsgálunk ethernethálózaton. A legelső oszlopban található az időbélyeg, utána a csomag forráscíme és -kapuja, majd célcíme és -kapuja szerepel. A címet a kaputól a pont (.) karakter választja el. A program a címeket és kaput megpróbálja feloldani. Siker esetén a cím vagy kapu helyett a neve fog szerepelni. Néhány esetben ez a feloldás lassú lehet vagy olyan forgalmat hozhat létre, ami a vizsgálatot zavarja. Ez esetben a tcpdump programnak a `-n` kapcsolót megadva a címfeloldás elmarad, a `-nn` hatására pedig kapufeloldás sem történik. Ebben az esetben mind a címek, mind a kapuk eredeti formában láthatók. Ha az adott keret egy IP-csomag darabja (erről már az IP-darabolás bemutatásakor az [1.] és [5.] részben bővebben szót ejtettünk), akkor a sor végén a *(frag azonosító:méret@kezdőpozíció)* jelölés található. Az azonosító az IP-fejléc azonosító mezője, a méret az adott darab mérete (bájtokban számítva az IP-fejléc hosszát), míg a kezdőpozíció a darab kezdete az eredeti IP-csomagban. Ezek után még a pluszjel (+) is előfordulhat, amely további darabokra utal. Ez nem más, mint az IP-csomag MF (More Fragments) bitje. Amennyiben a csomag nem darabolható, a sor végén egy (DF)-jelzés található. Ez a csomag fejlécében rejlő DF (Don't Fragment) bitre utal. Ha a programnak a `-v` vagy a `-vv` kapcsolót is megadjuk, további adatok birtokába jutunk. Ez esetben a sor végén mind az IP-fejléc azonosító

a) UDP-kapu zárva

```
(1) 18:27:26.019610 P cheetah.1024 > dolphin.1233: udp 6
(2) 18:27:26.019610 P dolphin > cheetah: icmp: dolphin udp port 1233
      unreachable [tos 0xc0]
```

b) UDP-kapu tiltva

```
(1) 18:27:49.667538 P cheetah.1024 > dolphin.1234: udp 6
(2) 18:27:49.667538 P dolphin > cheetah: icmp: dolphin udp port 1234
      unreachable [tos 0xc0]
```

c) TCP-példa

```
(1) 15:59:36.933715 P cheetah.1031 > dolphin.ssh: S 4011370143:4011370143(0)
      win 32120 <mss 1460,sackOK,timestamp 1205813 0,nop,wscale 0> (DF)
(2) 15:59:36.933715 P dolphin.ssh > cheetah.1031: S 3980097340:3980097340(0)
      ack 4011370144 win 32120 <mss 1460,sackOK,timestamp 1237732
      1205813,nop,wscale 0> (DF)
(3) 15:59:36.933715 P cheetah.1031 > dolphin.ssh: . 1:1(0) ack 1 win 32120
      <nop,nop,timestamp 1205813 1237732> (DF)

(4) 15:59:36.933715 P dolphin.ssh > cheetah.1031: P 1:26(25) ack 1 win 32120
      <nop,nop,timestamp 1237733 1205813> (DF)
(5) 15:59:36.933715 P cheetah.1031 > dolphin.ssh: . 1:1(0) ack 26 win 32120
      <nop,nop,timestamp 1205813 1237733> (DF)

(6) 15:59:47.122817 P cheetah.1031 > dolphin.ssh: F 1:1(0) ack 26 win 32120
      <nop,nop,timestamp 1206832 1237733> (DF)
(7) 15:59:47.122817 P dolphin.ssh > cheetah.1031: . 26:26(0) ack 2 win 32120
      <nop,nop,timestamp 1238752 1206832> (DF)
(8) 15:59:47.122817 P dolphin.ssh > cheetah.1031: F 26:26(0) ack 2 win 32120
      <nop,nop,timestamp 1238752 1206832> (DF)
(9) 15:59:47.122817 P cheetah.1031 > dolphin.ssh: . 2:2(0) ack 27 win 32120
      <nop,nop,timestamp 1206832 1238752> (DF)
```

mezőjét, mind a csomag élettartam-számlálóját kiíratjuk. A tcpdump az adatkapcsolati réteg adatait alapesetben nem mutatja. Ha ezekre is kíváncsiak vagyunk, használjuk a `-e` kapcsolót. Ethernethálózat esetén a keret ethernetszintű forrás- és célcíme, a hordozott protokoll, valamint a keret hossza is megjelenítésre kerülnek. Alapesetben a tcpdump csak a gép által vett vagy küldött forgalmat mutatja. Amikor a `-p` kapcsolót megadjuk, akkor a kártyát nem kapcsolja lehallgató módba. A `-p` kapcsoló értelmezése sajnos nem teljesen egyértelmű. Az eredeti forrásban és a Debian-változatban a fenti értelmezés az elfogadott, de a RedHat-csomagban a `-p` kapcsoló értelmezése ennek épp az ellenkezője. Érdemes a kapcsolók értelmezésének rendszerünk tcpdump súgóoldalán utánanézni. Abban az esetben, ha a gép több hálózati csatolót is tartalmaz, a csomagok beolvasását kényelmesebb csak az egyikre szűkíteni. Amennyiben a lehallgató módot választjuk, ezt kötelező megtenni. Ezt a `-i` kapcsolóval ejthetjük meg, a kapcsoló után pedig a csatolókártya nevét kell beírnunk. A terhelés csökkentése céljából a tcpdump nem olvassa fel a teljes keretet, pusztán csak az első 68 bájtot. Ha a teljes keret érdekel (például a csomag adatrészére vagyunk kíváncsiak), a `-s` kapcsolóval megmondhatjuk, milyen hosszan tárolja. A kapcsoló után a legnagyobb beolvasandó hosszt jelöli – célszerű a csatolókártya MTU-értékét megadni, ez ethernet esetén általában 1500 bájt. Ha a csomag tartalmát is meg szeretnénk jeleníteni, akkor a `-a` kapcsolót (ASCII) vagy a `-x` kapcsolót (hexadecimális) is megadhatjuk. Sok esetben hasznos lehet, ha a beolvasott adatokat a későbbiekben is meg tudjuk vizsgálni, ezért a tcpdump lehetőséget nyújt arra, hogy a beolvasott kereteket állományba menthessük. Természetesen az ilyen állományok beolvasására is képes. Ez sokkal hasznosabb lehet, mint ha a program kimenetét mentenénk el, hiszen ilyenkor lehetőség nyílik a későbbi szűrésekre vagy más programokkal történő elemzésre.

Szűrés

Hasznos segédeszköz a tcpdumpba épített szűrési lehetőség, amivel a hálózatról felolvasott keretek körét befolyásolhatjuk. A szűrő logikai kifejezés, csak azokat a kereteket olvassa fel, amelyekben a kifejezés értéke igaz. Ha nem adunk meg szűrőkifejezést, az összes keretet felolvassa. A kifejezés alapelemek (primitives) összekapcsolásából áll.
A lehetséges alapelemeket az 56. oldalon található *táblázat* mutatja. A szűrőfeltételben – a protokoll nevét tömbként használva – a csomag tartalma is vizsgálható. A formai követelmény az alábbi: *`protokoll [ eltolás : méret ]`* ahol a protokoll az `ether`, `fddi`, `ip`, `arp`, `rarp`, `tcp`, `udp` vagy `icmp` valamelyike. Az eltolás a protokollréteg kezdetétől számított táv (offset), a méret az adat hossza. Mindkét értéket bájtokban adjuk meg. A méret lehetséges értékei 1, 2 vagy 4, az alapértelmezett az 1. A `len` a tcpdump beépített változója, amely a keret hosszát tartalmazza. Ezek között az adatok között a C nyelvben megszokott aritmetikai és relációs műveletjeleket is használhatjuk. A lehetséges aritmetikai műveletjelek: `+`, `-`, `*`, `/`, `&`, `|`, míg a relációs műveletjelek: `>`, `<`, `>=`, `<=`, `=`, `!=`
Az adatvizsgálat relációs műveletjelek segítségével kapcsolható a logikai kifejezésbe. A logikai kifejezés tagjai a már megszokott módon kapcsolhatók össze. Lehetőség nyílik *ellentétképzésre* (negálásra: `not` vagy `!`), az *és* (`and` vagy `&&`), valamint a *vagy* (`or` vagy `||`) műveletre is. Az ellentétképzés sorrendisége a legmagasabb, az *és,* valamint a *vagy* műveletek sorrendisége megegyezik, és balról jobbra értékelődnek ki. A műveletek sorrendje zárójelek segítségével módosítható.

Az ICMP-csomagok

Az ICMP-csomagokat a forrás- és célcímek után következő `icmp:` rész jelöli. Ezután következik maga az üzenet, például a *port unreacheable (dest unreacheable/port unreacheable* – célkapu nem érhető el) vagy a `ping` parancs által küldött *echo request*, valamint a válaszul kapott *echo reply* üzenetek. Egy *port unreacheable* üzenet látható az *1. a lista* (1.) sorában és a `ping` parancs eredménye a (2–3.) sorokban.

a) TCP-kapu zárva

```
(1) 18:04:16.998792 P cheetah.1036 > dolphin.1233: S 3321364585:3321364585(0)
      win 32120 <mss 1460,sackOK,timestamp 1953835 0,nop,wscale 0> (DF)
(2) 18:04:16.998792 P dolphin.1233 > cheetah.1036: R 0:0(0) ack 3321364586 win 0
```

b) TCP-kapu tiltva

```
(1) 18:04:33.597399 P cheetah.1037 > dolphin.1234: S 3328752345:3328752345(0)
      win 32120 <mss 1460,sackOK,timestamp 1955496 0,nop,wscale 0> (DF)
(2) 18:04:33.607398 P dolphin > cheetah: icmp: dolphin tcp port 1234 unreachable [tos 0xc0]
```

c) tcpdump-példák

```
(1) tcpdump -i eth0 'host cheetah'
(2) tcpdump -i eth0 'src host cheetah'
(3) tcpdump -i eth0 'host cheetah and tcp and port ssh'
(4) tcpdump -i eth0 'not ( host cheetah and tcp and port ssh )'
(5) tcpdump -i eth0 'not ( tcp and ( src host cheetah and dst port ssh ) or
      ( dst host cheetah and src port ssh ) )'
(6) tcpdump -i eth0 'tcp[13] & 7 != 0'
(7) tcpdump -i eth0 'tcp[13] & 5 != 0 or tcp[13] & 18 = 2'
```


Az ARP-csomagok

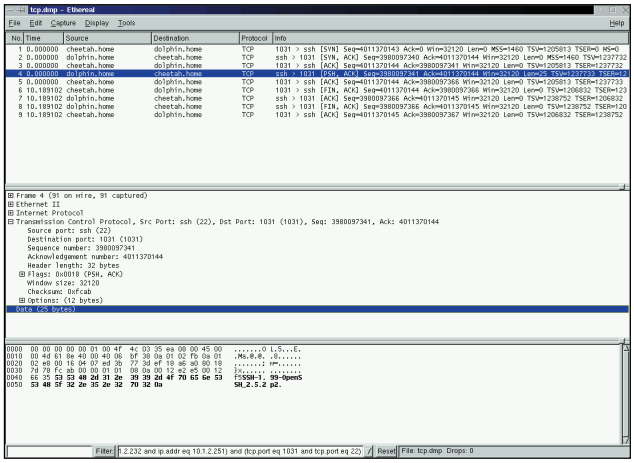
Az ARP (Address Resolution Protocol) feladatait cikksorozatunk 5. részében [2.] már bemutattuk. Most nézzük meg, hogyan is működik ez a valóságban. Ahogyan az *1. b listán* is látható, a gép ARP-kérést küld ki a hálózatra (1.). A címzett (vagy az erre feljogosított egyéb) gép válaszol (2.), az ARP-kérés üzenetszórt módon kerül továbbításra (miként a példában is látható), míg a válasz közvetlen címzéssel érkezik. Ahhoz, hogy az ethernetcímeket is láthassuk, a `-e` kapcsolót is megadtuk a programnak.

Az UDP-csomagok

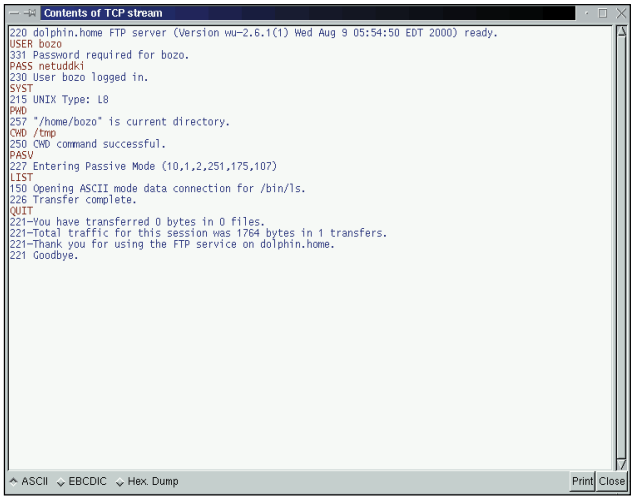
Az UDP szemléltetéséhez egy gép indulását vizsgáltuk az *1. c listában*: itt a BOOTP- illetve a DHCP-, valamint a DOMAIN-protokollok használatára is láthatunk példát. A tcpdump mindkét protokollt ismeri, így ezek értelmezésére is képes. A *cheetah* induláskor üzenetszórt csomagot küld ki, amellyel DHCP-kiszolgálót keres 1. sor. A kiszolgáló-bejegyzésben címek helyett nevek szerepelnek, így a DHCP-kiszolgálónak ezeket először fel kell oldania. A két tartománykérés a 2. és 4. sorban, a DNS válasza pedig a 3. és 5. sorokban látható. A két névfeloldás egymás után következik be. A DHCP a választ a csomagban (6. sor) küldi el az ügyfélnek. Az ügyfél újabb adatokat kér (7. sor), amelyeket a kiszolgáló vissza is küld (8.). A beérkezett adatok alapján a gép DHCP-kiszolgálót választ (9.), amelyet a kiszolgáló nyugtáz (10.). Mi történik azonban, ha a kiszolgáló adott kapuja zárva marad, miközben megkísérelünk csomagot küldeni neki? Ezt szemlélteti a *2. a lista*. A csomagot a *cheetah* (1.) küldi a *dolphin* gép 1233-as kapujára. Az adott kaput azonban zárva találja, így a *dolphin ICMP port unreachable* üzenettettel válaszol (2.). Ugyanígy ICMP-üzenetet bocsát ki a kiszolgáló akkor is, ha az adott kaput csomagszűrővel védjük. Az üzenet típusa *destination unreachable*, a kódja viszont változhat. A Linux 2.2.x magsorozat ilyenkor *port unreachable* kóddal válaszol, mintha a kapu zárva lenne. Ezt mutatja meg a *2. b lista*.

A TCP-csomagok

A *2. c listán* a TCP-kapcsolat felépítésére, az adattovábbításra és a lebontásra láthatunk példát. Mint már a protokollok ismertetésénél [1.] említettük, a kapcsolat felépítése három lépésből áll. A példán a *cheetah* gép kapcsolódik a *dolphin* SSH-kapujára. A kezdeményező gép egy SYN-es csomagot küld (1.), amelyben az általa választott ISN (Initial Sequence Number – kezdő sorszám) rejtőzik. A SYN bit meglétét a címek utáni S betű jelzi. Ha a címzett gép adott kapuja fogadja a kérést, akkor SYN+ACK csomagot küld vissza (2.). A SYN bit itt szintén látható, de később az `ack` kulcsszó is szerepel, mögötte a *cheetah* által választott kezdősorszámnál eggyel nagyobb értékkel. Figyeljük meg, hogy a távoli gép maga is előállít egy kezdősorszámot. A *cheetah* gép a csomag vétele után egy ACK-csomagot küld (3.), ezzel a kapcsolat felépült – a csomagon nincsenek jelzőbitek, ezért itt most pont (.) szerepel. A két szám a kezdő és a végső sorszám, mögötte zárójelben a csomagban levő adatbájtok száma található. Látható, hogy a csomag csak nyugta, adatot nem hordoz. A sorszámok viszont furcsa módon megváltoztak. Ezt a változtatást a tcpdump követte el, így az eredmény könnyebben olvasható lett. Ha a csomag nem tartalmaz SYN bitet, a kezdősorszámot mind a pillanatnyi, mind a nyugta sorszámából kivonja. A `-S` kapcsoló megadásával az eredeti abszolút sorszámokat látjuk, amelyeket a csomagok is tartalmaznak, kapcsoló nélkül pedig a relatív TCP-sorszámokat fogja megjeleníteni. Aki kíváncsi rá, próbálja ki – pillanatokon belül azt is észreveszi, hogy miért tesz így a program. A (4.) csomagban a *dolphin* elküldi az SSH-protokoll azonosító üzenetét az ügyfélgépnek. Láthatjuk, hogy ez a 1–26. relatív sorszámokat foglalja el, így az üzenet 25 bájt hosszú. A csomagon a P (PUSH) jelzőbit szerepel, amely az ügyfél TCP-, illetve IP-alrendszerét utasítja, hogy a vett adatokat továbbítsa az alkalmazásnak. Az (5.) csomagban a *cheetah* nyugtázza mind a 26 bájtot, mely az `ack` után látható. Az ügyfél nem küld adatokat. A kapcsolat lebontását a (6–9.) csomagok végzik. Mint már említettük, mindkét adatirányt függetlenül kell lebontani. A bontást a *cheetah* kezdeményezi (6.), a csomag egy F (FIN) jelzőbitet tartalmaz; a *dolphin* nyugtázza a bontási kérést (7.). A FIN bit egy sorszámot foglal el, ettől lesz a nyugtasorszám 2. A FIN vétele után a *dolphin* is kezdeményezi a kiszolgáló–>ügyfél adatirány-bontását (8.), a *cheetah* ezt nyugtázza (9.), majd sikeresen lebontja a kapcsolatot. Mi történik, ha a kiszolgáló adott kapuja zárva marad, miközben megkísérlünk rákapcsolódni? Ezt szemlélteti a *3. a lista.* Az 1. csomagban a *cheetah* kapcsolatfelépítési kérést küld a *dolphin* gép 1233-as kapujára. Az adott kaput viszont zárva találja, így a *dolphin* egy RST-szakaszt küld válaszul (2.) – mint láthatjuk, az RST jelzőbit meglétét az R betű jelzi. Ha azonban a kaput csomagszűrővel védjük, a



1. kép Az ethereal fő képernyője



2. kép Az ethereal TCP kapcsolatkövető képernyője

visszautasítás módja megváltozik. Ilyenkor a rendszermag egy ICMP-üzenetet küld vissza. Az üzenet típusa *destination unreachable*, a kódja viszont változhat. A Linux 2.2.x magsorozat ilyenkor *port unreachable* kóddal küldi vissza (ezt szemlélteti a *3. b lista*).

tcpdump-példák

Az *3. c lista* hasznos példákat mutat a tcpdump használatára. Az 1. példában látható, hogy az `eth0` hálózati kártyán a *cheetah* gépről induló vagy oda érkező IP- és ARP-csomagokra figyelünk. A 2. példában csak a *cheetah* gépről kiinduló IP-csomagokat nézzük. A 3. példa már összetettebb szűrőt használ: csak azon TCP-csomagokat mutatja, amelyekben a *cheetah* gép, valamint az `ssh port` (22/tcp) érintett (a *cheetah* SSH-kapujára érkező és induló vagy a *cheetah* gép és tetszőleges más gép 22-es TCP-kapuja közötti csomagok). Sokszor előfordul, hogy SSH-protokollon jelentkezünk be egy gépre, és úgy futtatjuk a tcpdumpot. Ilyenkor hasznos lehet a 4. példa. Itt a *cheetah* gépet és az SSH-kaput érintő forgalom figyelmen kívül marad (ez a feltétel ugyan más csomagokat is eldob, de a célnak sokszor így is megfelel). A kifogástalan feltétel az 5. példában látszik. Ilyen bonyolultabb feltételek esetén lehet hasznos a `-F` kapcsoló, aminek hatására a szűrőkifejezést a megadott állományból olvassa; ilyenkor a parancssorban megadott szűrőfeltételt figyelmen kívül hagyja.
Ha csak a SYN, a FIN és a RST jelzőbitekkel rendelkező csomagok érdekelnek minket, akkor a 6. példa mutatja a megoldást. Mint cikksorozatunk korábbi részeiben említettük [1., 5., 6.], ezek a jelzőbitek a TCP-fejléc 14. bájtján helyezkednek el. Ha pedig azt szeretnénk, hogy a SYN+ACK bitkombinációval rendelkező csomagok ne látsszanak, akkor a (7.) példa szerinti szűrőfeltételt kell használni.

ethereal

Az *ethereal* hálózati elemzőcsomag Unixokra. Könnyen kezelhető, számos kényelmi lehetőséggel: képes a csomagok közvetlen beolvasására, de akár a tcpdump segítségével létrehozott állományok is beolvashatók vele. A csomag két programot tartalmaz: a *tethereal* karakteres felülettel rendelkezik, míg az *ethereal* GTK+ toolkit segítségével grafikus felületen fut. Az ethereal-csomag szinte minden Linux-terjesztéshez elérhető, de forrásban is letölthető [4.]. Az ethereal-csomag több keretformátumot és protokollt ismer, mint a tcpdump. Roppant hasznos segédeszköz, ha valaki hálózaton kapcsolatot tartó programot készít vagy valamely protokoll működésére kíváncsi. Az ethereal főképernyőjét mutatja a *1. kép*. Itt jól látható, hogy miként jeleníthető meg a beolvasott csomagok összes jellemzője. A program nagyon hasznos további lehetősége, hogy a TCP-kapcsolatok végigkövethetők. Ebben az esetben hexadecimálisan vagy szöveges formában láthatjuk a kapcsolatban átáramló adatokat. A két gép által küldött karaktereket eltérő színnel jeleníti meg.
A *2. képen* FTP-példa látható: a kiszolgáló által küldött üzenetek (kék színnel) jól elkülönülnek az ügyfél által küldött üzenetektől (piros). Az ethereal használatát rendkívül könnyű elsajátítani. Szűrőfeltétel felépítésére itt is lehetőség nyílik, bár a szűrő formai követelményei eltérnek a tcpdumpnál megismertektől. A pontos formátumot az eszköz kézikönyvében jól leírták.

Biztonsági kockázatok

Talán kissé furcsán hangzik, de a hálózati elemzőknek is komoly biztonsági kockázatai akadnak. Ennek oka, hogy a program magas jogosultsággal fut és bemenete a támadó által befolyásolható. Egy rendszergazdai jogosultsággal futó tcpdump-programra ugyanúgy kell tekinteni, mintha hálózati démon lenne. Bármilyen biztonsági hiba (amire már volt is példa a tcpdump esetén is) a figyelő állomáson kód-végrehajtást tehet lehetővé.

Irodalomjegyzék

[1.] Könnyű álmok: Az Interneten használt fontosabb protokollok – Linuxvilág, 2001. január
[2.] Könnyű álmok: Behálózva – Linuxvilág, 2001. április
[3.] A `tcpdump` honlapja ➲ http://www.tcpdump.org
[4.] Az `ethereal` honlapja ➲ http://www.ethereal.com
[5.] W. Richard Stevens TCP/IP Illustrated, Volume 1; ISBN 0201633469
[6.] RFC793: Transmission Control Protocol

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 (2. rész)

Ebben a részben az egyszerű testek meghatározását, a testek közötti halmazműveleteket és a különféle transzformációkat tekintjük át.

Bemutatjuk az egyszerű testek (sík, gömb, téglatest, kúp, csonka kúp, henger és tórusz) meghatározását, a testek közötti halmazműveleteket (CSG – Constructive Solid Geometry), valamint a különféle átalakításokat (tranformációkat). A jelenet megadásához a PoV-Ray néhány alapvető testet bocsát a rendelkezésünkre, amelyekből összetett testek képzésére nyílik lehetőségünk – így például forgástesteket, Bezier-felületeket is létrehozhatunk.

Amint sorozatunk első részében már láthattuk, egy *gömb* meghatározása egy koordinátával és a gömb sugarának megadásával történhet, a következő formában:

```
sphere {
        VEKTOR
        SUGAR
}
```

A megadott `VEKTOR` a gömb középpontját jelöli ki, a valós `SUGAR` pedig a gömb sugarát határozza meg.

Az előző részben ugyancsak használtunk *sík* felületet. Ezt a következő módon hozhatjuk létre:

```
plane {
        VEKTOR, VALÓS
}
```

Itt az első érték egy vektor, ami merőleges a síkra (a sík normálvektora), az ezt követő valós szám pedig azt határozza meg, hogy a síkot hány egységgel kell eltolni a normálvektor által kijelölt tengelyen.

Egyszerű téglatestet a `'box'` kulcsszóval hozhatunk létre, mely a következő értékekkel bírhat:

```
box {
        VEKTOR1
        VEKTOR2
}
```

Az első vektor itt a nullponthoz közelebb eső bal alsó csúcspont helyvektorát adja meg, míg a második vektor az előbbi csúcsból induló testátló másik végén lévő csúcspont helyvektora.

Egy testátló két végének megadásával a téglatestet egyértelműen meghatároztuk.

Kúp és *csonkakúp* megalkotása a következő sorokkal lehetséges:

```
cone {
        VEKTOR1, SUGAR1
        VEKTOR2, SUGAR2
}
```

A két vektorral meghatározzuk a kúp tengelyének végeit, míg a vektorok után megadott két valós szám a csonka kúp végeinek sugarát szabja meg. Szabályos kúpot úgy hozhatunk létre, hogy az egyik sugarat 0-ként adjuk meg.

```
cone {
        <0,0,0>, 2
        <0,4,0>, 0
}
```

Ha azt szeretnénk elérni, hogy a csonka vagy a szabályos kúp mindkét vége nyitott legyen, akkor a fenti két sor után az `'open'` kulcsszót is használnunk kell. *Hengert* a következő módon adhatunk meg.

```
cylinder {
        VEKTOR1
        VEKTOR2
        SUGAR
}
```

A két vektor szerepe megegyezik a fentebb tárgyalt kúpnál szereplő vektorokéval, vagyis ezekkel adjuk meg a henger két végét. Tekintettel arra, hogy a henger minden keresztmetszete azonos sugarú, itt csak egyetlen sugármegadás szükséges, mely a henger tengelye mentén állandó lesz. Itt is használható az `'open'` kulcsszó, hatására a henger mindkét vége nyitott marad.

A *tórusz* talán nem ismeretlen fogalom azok körében, akik már használtak valamilyen 3D-modellező programot. A többiek pedig könnyedén képet alkothatnak maguknak erről a testről, ha egy lyukas közepű fánkra vagy hullahopp-karikára gondolnak.

A PoV-Rayben tóruszt a következő utasításokkal hozhatunk létre:

```
torus {
        NAGYOBB_SUGAR, KISEBB_SUGAR
}
```

A fenti listában a NAGYOBB_SUGAR valós érték határozza meg a tárgy tényleges sugarát, míg a KISEBB_SUGAR a cső sugarát adja meg.

Ezzel végére is értünk az egyszerű testek sorának.

Következzenek a különféle *átalakítások* (transzformációk) formai követelményei! Három alapvető átalakításról beszélhetünk, ezek minden 3D-modellezéssel kapcsolatos programban megtalálhatók. Az első ilyen az eltolás, amely nem csupán tárgyakra értelmezhető, hanem természetesen kamerára és fényforrásokra is. Formája a következő:

```
translate {
        VEKTOR
}
```

A VEKTOR az eltolás vektora.

Szintén egy-egy vektorral adható meg a következő két átalakítás: a méretezés (scale) és a forgatás is.

```
scale {
        VEKTOR
}

rotate {
        VEKTOR
}
```

A fentiekre álljon itt példaképpen ez a rövid lista, amiben mindhárom átalakítás fellelhető.

```
box {
< 0, 0, 0 >, < 1, 1, 1 >
translate { < 2, 2, 2 > }
rotate { 20 * x }
scale { < 1, 1.2, 2 > }
}
```

Ha nincs szükségünk az átalakítási vektorok minden elemére, használhatjuk a PoV-Raybe beépített egységvektorokat is, amint azt a forgatásnál láthattuk.

Az alábbiakban erre a három műveletre alapozva létrehozunk egy *kereket*. A kerék abroncsát egy tórusz fogja képezni, a küllőket pedig

hengerek alkotják. A kerékagyat és a tengely csatlakozási helyét két torzított gömbből alakítottam ki. Az elkészült képet a CD-mellékleten kerek.tga néven találhatjuk meg.

```
// Makróként megadjuk a küllőt, hogy a
// későbbiekben csak a forgatással és eltolással
// kelljen foglalkozni.
#declare kullo=cylinder {
        <0, 0, 0>
        <0, 0.8, 0>
        0.1
        pigment { color rgb <0.2, 0.2, 0.5> }
}

// Szórt fény
global_settings {  ambient_light 0.9 }

// Kamera
camera
{
  angle 40
  location  <-2.0 , 2.0 ,-2.0>
  look_at   <0.0 , 0.0 , 0.0>
}

// Direkt fényforrás
light_source
{
  0*x
  color rgb <1,1,1>
  spotlight
  translate <20, 40, -20>
  point_at <0, 0, 0>
  radius 10
  falloff 20
}

// Alaplap
plane {
       y, 0
       pigment { color rgb <0.6, 0.6, 0.6> }
}

// Kerékabroncs
torus
{
  0.8,
  0.15
  pigment { color rgb <0.2, 0.5, 0> }
  translate 0.25*y  // <dX dY dZ>
}

// 6 küllő a kerékhez
object { kullo
       rotate <90, 0, 0>
       translate y*0.25
}
object { kullo
       rotate <90, 60, 0>
       translate y*0.25
}
object { kullo
       rotate <90, 120, 0>
       translate y*0.25
}
object { kullo
       rotate <90, 180, 0>
       translate y*0.25
}
object { kullo
       rotate <90, 240, 0>
       translate y*0.25
}
object { kullo
       rotate <90, 300, 0>
       translate y*0.25
}

// Kerékagy
sphere {
       <0, 0, 0>
       0.4
       pigment { color rgb <0.8, 0.3, 0.2> }
       scale y*0.3
       translate y*0.25
}
sphere {
       <0, 0, 0>
       0.2
       pigment { color rgb <0.4, 0.15, 0.1> }
       scale y*1.2
       translate y*0.25
}
```

Az alapvető átalakításokat áttekintve, ebben a hónapban nem maradt más hátra, mint a *CSG-műveletek* megértése és használatuk elsajátítása. A CSG – azaz a már említett Constructive Solid Geometry – a legkönnyebben úgy érthető meg, hogyha az egyes testeket térbeli pontok halmazának tekintjük. Ebben az esetben a műveletek egyszerű halmazműveletek, nevük is ennek megfelelő: unió (union), különbség (difference) és metszet (intersection). A PoV-Rayben létezik egy negyedik művelet is, ezt összekapcsolásnak (merge) nevezhetjük, és feladatát tekintve nagyon hasonlít az unió-művelethez.
A CSG segítségével rendkívül összetett alakzatokat is létrehozhatunk, de talán felmerül a kérdés, hogy miért nem elégséges egyszerűen csak egymás mellé helyezni azokat a tárgyakat, amelyekből az összetett objektum áll? Ha az objektumokat csupán egymás mellé helyezzük, akkor annak minden egyes apró elmozdulásakor minden tárgyat pontosan ugyanannyival kell arrébb tennünk, amennyivel maga is elmozdult – és ugyanez lesz igaz a méretezésre és a forgatásra is! Ezenkívül ha tárgyunk minden része azonos anyagból készült, felesleges munka lenne ezt az anyagot minden rész számára megadni, egyszerűbb a részeket egymáshoz kapcsolva kezelni. A CSG-vel létrehozott tárgyak szerkezetét faszerkezetként ábrázolhatjuk, ahol a levelek az egyes részobjektumok, míg a fa belső csomópontjai az egyes műveletek. Elsőként az egyes műveletek formai követelményeit vizsgáljuk meg, majd létrehozunk egy összetett objektumot, amelynek szerkezete a következő ábrán látható. Objektumunkat nevezzük – a nagyfokú hasonlóság miatt – mondjuk *hamutartónak*.
Az első művelet legyen az unió. Ennek formai követelménye az alábbi:

```
union {
       OBJEKTUM_1
       OBJEKTUM_2
       ...
       ...
       OBJEKTUM_N
}
```

2. lista

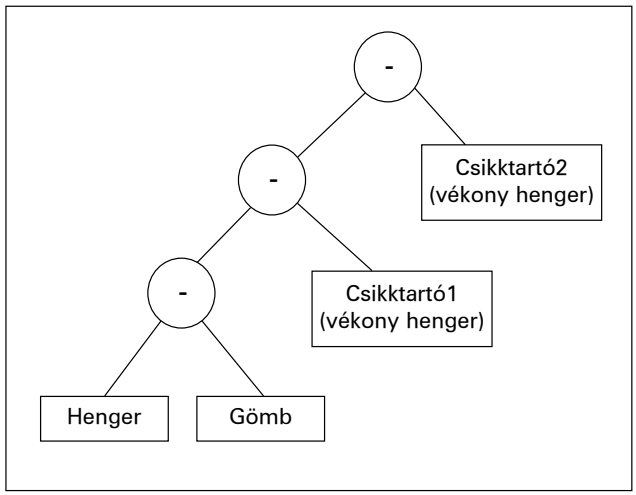
```
**
// A hengerből kivonjuk a gömböt
#declare Hamu_alja=difference {
                    cylinder {
                            <0,0,0>
                            <0,0.5,0>
                            0.9
                    }
                    sphere {
                            <0,0.82,0>
                            0.8
                    }
            }

// Az előbbi művelet eredményéből kivonjuk
// az egyik kis fekvő hengert
#declare Alja_egylyukkal=
        difference {
                object { Hamu_alja }
                cylinder {
                        <0,0.3,0>
                        <0,2.8,0>
                        0.1
                        rotate <90, 90, 0>
                        translate y*0.5
                }
        }

// Egylyukú hamutartó - másik kis fekvő henger =
// = készen van a hamutartó.
difference {
        // Hamutartó alja - egyik csikktartó
        object { Alja_egylyukkal }
        cylinder {
                <0,0.3,0>
                <0,2.8,0>
                0.1
                rotate <90, 270, 0>
                translate y*0.5
        }
        // Adjunk neki kék színezetet
        pigment { color rgb <0.3, 0.3, 1> }
}
*
```

A fentiekből kitűnik, hogy ez a művelet nemcsak két objektumon értelmezhető, hanem egyidejűleg több objektumot is eggyé olvaszthatunk. Az egyes műveletek értékei összetett objektumok is lehetnek, ahogyan azt a későbbi forráslistákon láthatjuk. Tehát a különbségképzés és a metszet a következőképpen néz ki:

```
difference {
        OBJEKTUM_1
        OBJEKTUM_2
        ...
        ...
        OBJEKTUM_N
}
intersection {
        OBJEKTUM_1
        OBJEKTUM_2
        ...
        ...
        OBJEKTUM_N
}
```



Hamutartó szerkezete CSG-fa ábrázolásban

Mivel a különbségképzés eredménye számos objektum esetében nehezen képzelhető el előre, ezt a műveletet célszerű egyszerre csak két tárgyon elvégezni és a kapott eredményt használni a továbbiak során. Utolsó műveletként az összekapcsolást (merge) tárgyaljuk, amely az uniótól annyiban tér el, hogy a tárgyak belső felületei eltűnnek. Ez főként átlátszó tárgyak létrehozásakor lehet fontos, hiszen például egy domború lencsében nem szép, ha úgy tűnik, mintha a végső forma két félből lenne összeragasztva. Formája az előbbiek alapján könnyen érthető:

```
merge {
        OBJEKTUM_1
        OBJEKTUM_2
        ...
        ...
        OBJEKTUM_N
}
```

A következő listákban kétféle módon határozzuk meg a hamutartó formáját: elsőként egyszerre végezzük el az összes műveletet (1. lista, mely a 15. CD Magazin/PoV-Ray könyvtárában található), majd a második megoldás során az egyes részeredményeket különkülön objektumként kezeljük (*2. lista*). Így végezetül ugyanahhoz az eredményhez jutunk.

Ezzel lezárjuk ezt a részt, a következőkben pedig olvasóinkat az objektumok létrehozásának bonyolultabb módszereivel ismertetjük meg, ezt követően térünk rá a különböző anyagokkal és mintázatokkal való ismerkedésre.

Addig is remélem, hogy mindenkinek kellemes perceket szerez a PoV-Ray és az emberi alkotóerő találkozása.

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.

Gimp a gyakorlatban (4. rész)

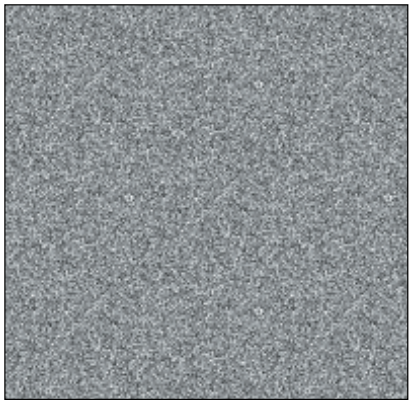
Munkánkat az előző részben elkészített kitöltőmintákkal folytatjuk, írásunk második felében pedig a maszkolás lehetőségeit tekintjük át.

Készítsük el az *Enlightenment* ablakkezelőben használt *Brushed metal* hátteret. Első lépésként hozzunk létre egy 200x200 pixel méretű képet 50 százalékos szürke háttérrel. Ez lesz kitöltő mintánk alapja (*1. kép*). Ezután alkalmazzunk rajta *Zajosítást* (*Szűrők/Zaj/Zajosítás, Filters/Noise/Noisify*), amitől a kép szemcséssé válik. Válasszuk ki a nekünk tetsző mértéket. Készülő mintázatunk típusa, mélysége nagymértékben a mintázattól függ majd: ha nagyobb mélységű mintázatot szeretnénk, a zajosítást többször is alkalmazhatjuk ugyanazon a képen. A végeredmény a tévében látható adásszünethez fog hasonlítani. A legjobb, ha minél kisebb az ismétlődés, mert a képpontokból könnyen interferencia alakulhat ki, ami minta esetében nagyon zavaró lehet (*2. kép*).
Utolsó lépésként alkalmazzunk képünkre *Gauss-elmosást (IIR)* (*Szűrők/Elmosás/Gauss-elmosás (IIR), Filters/Blur/Gaussian Blur (IIR)*). A kívánt eredmény eléréséhez csak vízszintes irányban kell elmosáshoz folyamodnunk. Az eredményt a *3. kép* mutatja. Mint látható, az elmosás eredményeképpen nem sikerült jól ismételhető képet készíteni. Ezen úgy segítettem, hogy levágtam a kép szélét. A mintát brush.pat néven a ~/.gimp-1.2/patterns/ könyvárba mentsük el, és a művelet során nevezzük át *Brush metal*-ra. Mentés után a *Pattern Selections* párbeszédablak segítségével a mintát ki is választhatjuk. Ne feledkezzünk meg a frissítésről sem, hogy elmentett mintánkat használni tudjuk! Az eredményként kapott minta remekül alkalmas hátterek előállítására, én például ennek segítségével készítettem az újabb weboldaltervet (*4. kép*).

Kígyóbőr készítése

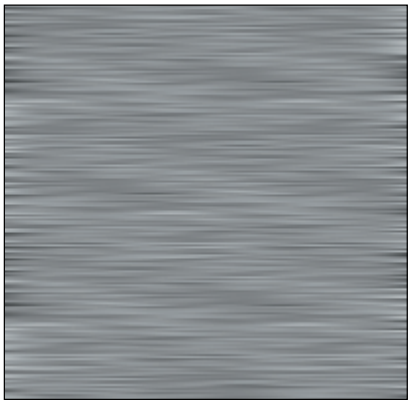
A kígyóbőrt utánzó kitöltő mintához hozzunk létre egy új képet (700x700 képpont, RGB), majd alkalmazzunk rajta plazmaszűrőt (*Szűrők/Megjelenítés/Plazma, Filters/Render/Clouds/Plasma – 5. kép)*. Ne ijedjünk meg, nem rontottunk el semmit! Alkalmazzunk rajta egy *Desaturate* telítetlen szűrőt (*Kép/Colors/Desaturate, Image/Colors/Desaturate*), ennek eredményeképpen alkotásunk fekete-fehér lesz (*6. kép*). Színábrázolása továbbra is RGB marad, ez tehát nem ugyanazt jelenti, mintha a színábrázolást fekete-fehérre változtattuk volna. (Ezzel a módszerrel kétszínű képeket tudunk készíteni.) Adjuk vissza a színeket képünknek! Használjuk ehhez a *Színegyensúly* menüpontot (*Kép/Colors/Színegyensúly, Image/Colors/Color Balance*). A kígyóbőrhöz zöld és sárga színekre van szükségünk – bár használhatjuk a kék és lila párosítást is, de ilyen színű

1. kép Kitöltőmintánk alapja



2. kép Zajos kép

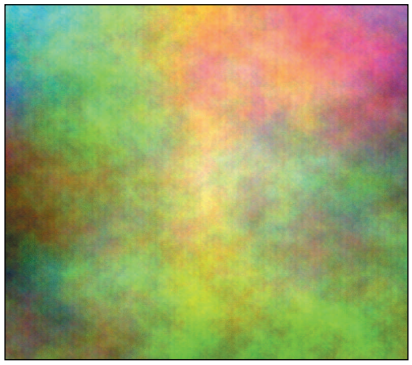
kígyóval ritkán találkozunk. Én a természetadta színeknél maradtam (*7. kép*), a végeredményt a *8. kép* mutatja be.
A kígyóbőr-mintázat elkészítéséhez a *Mozaikszűrő Szűrők/Megjelenítés/Minták/Mozaik, Filters/Render/Pattern/Mosaic* a legalkalmasabb eszköz (*9. kép*).
Nézzük meg az általam beállított értékeket! A kép nagy mérete miatt a csempe méretét növeltem, magassága és távolsága azonban kicsi. A fény irányát és a színek variációját alapértéken hagytam. A csempézettség csinossága (*Tile Neatness*) értékével a csempék elrendezettségét szabályozhatjuk: minél nagyobb az érték, annál szabályosabb a kitöltőminta. Ezt esetünkben 65 százalékra állítottam.
Nézzük a végeredményt a *10. képen*! Mondhatnánk azt is, hogy készen vagyunk, de finomítsunk rajta még egy kicsit! Tegyük még izgalmasabbá a mintát! Hozzunk létre egy új réteget és alkalmazzunk rajta ismét plazmaszűrőt. Ezzel az új réteggel fogjuk a bőrt rücskössé tenni. A rétegen ezt követően alkalmazzunk egy *Desaturate*-t, ezzel fekete-fehér felületet kaptunk. Ezzel a réteggel alkalmazzunk az első rétegen egy *Bump map*-et (*Szűrők/Leképezés/Bump-Map, Filters/Map/Bump Map*), így válik a bőr rücskössé (*11. kép*)!
A képet még sokáig csiszolgathatjuk, adhatunk hozzá például sötétítést és egyéb kiegészítéseket. Ennek a háttérnek sajnos van egy rossz tulajdonsága: nem ismételhető tökéletesen, ugyanis az ismétlések a csempézet illesztésekor nem tökéletesek, mivel



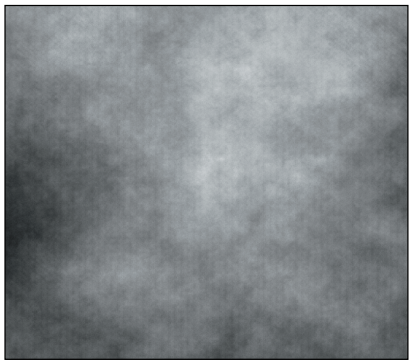
3. kép Az elkészült kitöltőminta

4. kép Krómfeliratú kitöltőminta

a mintázat túl részletgazdag. Természetesen különböző trükkökkel ezt a nehézséget is áthidalhatjuk. Ha megnézzük a Gimp telepítése után elérhető kitöltőmintákat, azok illesztése sem tökéletes (lásd az előző



5. kép Plazmaszűrő alkalmazása



6. kép Telítetlen színek alkalmazása

részben a Green Linux feliratot). Használatuk során ügyeljünk arra, hogy nagy, összefüggő területeket ne töltsünk ki velük, lehetőleg csak apróbbakra alkalmazzuk (például szöveg kitöltésére). Én ugyan vállalkoztam a merészebb feladatra, a háttérként való használatra, a megoldást azonban az olvasóra bízom (*12. kép*).

Mi a maszkolás? Miben különbözik a kijelöléstől?

A Gimp rendkívül hatékony eszközeinek egyike a maszkolás, ami nélkül számos esetben tehetetlenek lennénk, vagy csak nagyon körülményesen tudnánk elkészíteni a grafikát. Mondhatná a kedves olvasó: „Minek a maszkolás, amikor ott a kijelölés? Azt már ismerem, bőven elég az is!" Ez igaz! Vannak azonban olyan feladatok, amelyeket a maszkolás nélkül nem tudnánk megoldani. Egy létező képből csak egy részletet jelenítsünk meg kijelöléssel, vagy halványítsuk el függőlegesen átlátszóba. Az ilyen típusú feladatok maszkolással egy lépésben megoldhatók, kijelöléssel azonban hosszadalmas munka lenne. A maszkolás kiegészíti, kiterjeszti a kijelölés lehetőségeit: a 11. képen látható kígyóbőr mintát is maszkolással jelenítettem meg.

A Gimp két különböző módon kezeli a maszkolást: másként a rétegek és másként a csatornák esetében. A csatornákban való maszkolást tulajdonképpen kijelöléseink mentéseként is értelmezhetjük. Az összetett, több részletből álló kijelöléseket tárolhatjuk csatornában, melyeket egyszerűen az ismert rajzoló és törlő eszközökkel készíthetünk el. Így a kijelölések megalkotása rendkívül kényelmessé válik, és rétegek esetében a kép részleteit ki tudjuk takarni, „maszkolni". Így könnyen készíthetünk montázsokat, és a képeken később is egyszerű lesz változtatni. Nézzük át részletesen ezeket a szolgáltatásokat!

Csatornák használata a maszkoláshoz

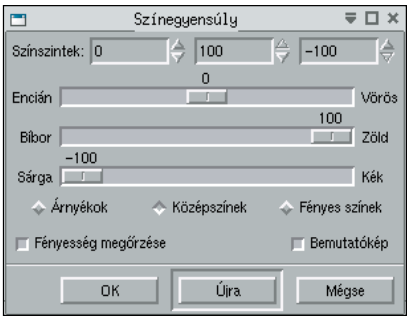
Csatornák esetében a maszkolásnak két fő feladata van:

a) kijelölések tárolása,

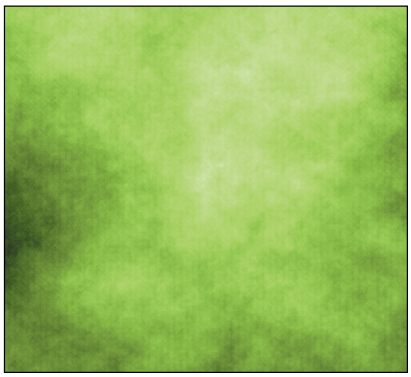
b) kitakarások beállítása.

A csatornákban nyolcbites (256 szürke árnyalatú) képekkel dolgozhatunk.

A láthatóság nagyon egyszerű: a csatornában levő képen a fehér átlátszó, a fekete nem (*13. kép*).



7. kép A Színegyensúly menüpont



8. kép A kígyó bőrének színei

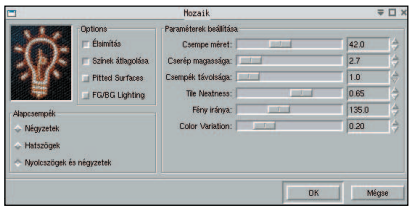
Miért hasznos kijelöléseinket tárolni? Többféle magyarázat létezik rá: hasznos, ha alkotásunk során végzett kijelöléseinket tároljuk, különös tekintettel azokra az összetett kijelölésekre, amiket nehezen tudnánk újra kijelölni, ugyanis részei a képnek. Bizonyos szűrők, szolgáltatások után azonban megszűnik a kijelölés.

A szűrők eredményét a kijelölések megszüntetésével és visszaállításával ellenőrizni lehet, így a legösszetettebb kijelölések között is gyorsan tudunk váltani.

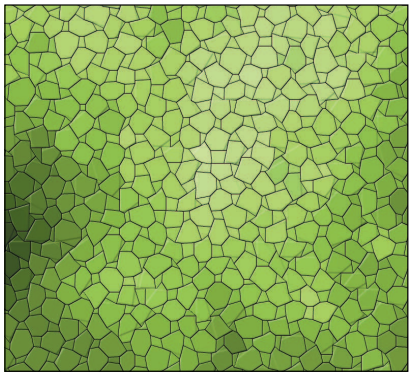
Remélem, olvasóinkat sikerült meggyőznöm arról, hogy milyen sok lehetőséget tudnak használni a Gimppel. Nem csak a *Fájl* menü létezik a programban…

Kijelölés-mentés csatornába

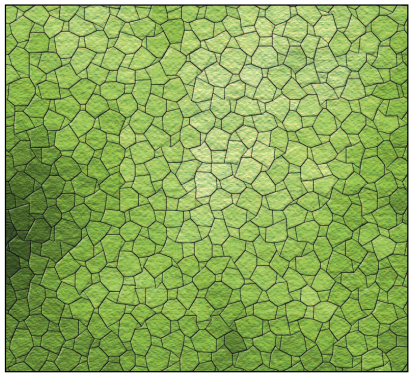
Képünk kijelölése után a legegyszerűbb megoldás a menüben található Kijelölés mentése csatornába *Kijelölés/Mentés*



9. kép A Mozaikszűrő beállított értékei



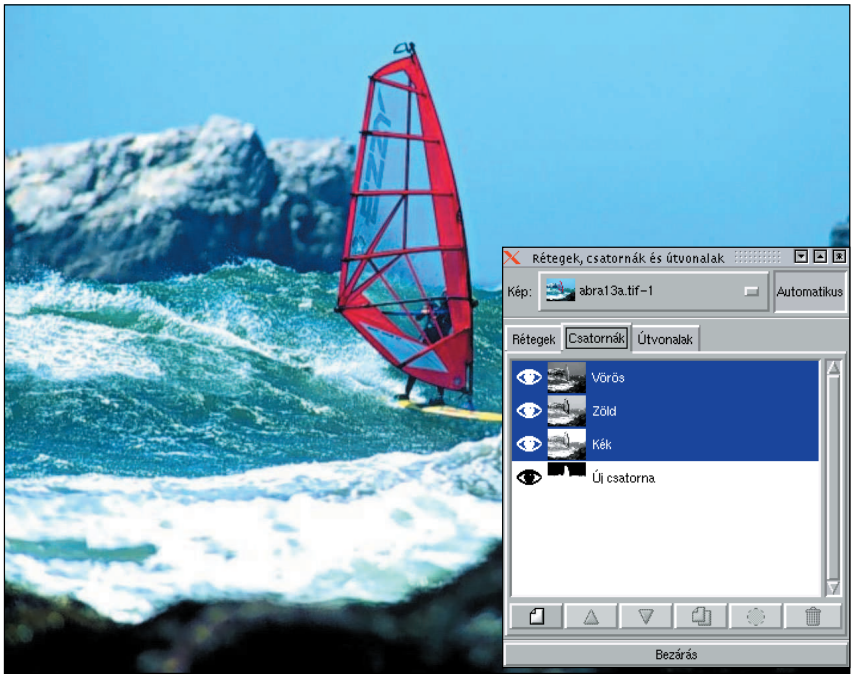
10. kép A kígyó bőre



11. kép Az elkészült bőrminta

12. kép A kígyóbőr háttérként való alkalmazása

csatornába (*Select/Save to Channel*) szolgáltatást használni. A párbeszédablakban az új csatornát meg is nézhetjük. A Gimp nagy előnye, hogy az elmentett kijelöléseket nagyon könnyen tudjuk szerkeszteni, módosítani: elég, ha kiválasztjuk azt a csatornát, amin változtatni szeretnénk és a rajzoló eszközzel módosítjuk a kijelölést!



13. kép Az eredeti kép és a maszkok megjelenítése párbeszédablak

14. kép Kitakarás rétegmaszk segítségével

Kijelölés módosítása

A 13. képen látható fotóból ki szeretnénk törölni a hátteret. A maszkolás használatával rendkívül egyszerű a dolgunk. Először is jelöljük a ki kívánt területet a kézi kijelölővel! A kijelölés még valószínűleg korántsem tökéletes. Mentsük csatornába: *Kijelölés/Mentés csatornába* (*Select/Save to Channel*). Ezután nincs más hátra, mint a kijelölést az apró részletekig finomítani. Jelöljük ki a csatornát, majd a megfelelő rajzolóeszközzel kényelmesen módosítsuk. Módosítás közben a Gimp fekete színnel jelöli a kijelölt területet. Ha más részletet is ki szeretnénk jelölni vagy takarni, hozzunk létre új csatornát, és azon dolgozzunk. Képeinkhez korlátlan számú kijelölést avagy csatornát készíthetünk.

Maszkok összevonása

Különálló, független kijelölések összekapcsolására, összevonására is lehetőség van a Gimpben. Csatornák egyesítésére többféle megoldás is kínálkozik.

Az egyik lehetőség:

1. Válasszuk ki az egyik csatornát: *Csatorna hozzáadása kijelöléshez* (*Channel to Selection*).
2. Lépjünk át a másik csatornára, majd azt is adjuk hozzá a kijelöléshez.
3. Mentsük el a kijelölést csatornába a korábban ismertetett módon.

Maszkolás rétegek esetében

A másik lehetőséget a kitakarásra a rétegek maszkolása jelenti. Ezt elsősorban kisebb részletek eltüntetésére használhatjuk. Ha a mű igényli, a kétféle maszkolás együttesen is alkalmazható. Rétegmaszk esetében a láthatóságot a fehér jelenti, a láthatatlan részt pedig a fekete, így ennek módosítása is rendkívül egyszerű. Bármelyik szerkesztő-eszközzel végezhetjük.

Nézzük meg, hogyan lehet rétegeket maszkolni! Milyen eszközöket használhatunk a maszkokon? Ecsetet, tollat, ceruzát, festékszórót, kijelölést, elmosást, színátmenetet – vagyis az összes eszköz a rendelkezésünkre áll. Ne feledkezzünk meg arról, hogy a maszkokban 256 szürkeszínárnyalatot használhatunk.

Képem kitakarásában például a szörfös mögötti hátteret szeretném törölni. Első lépésként elkészítettem a részletes kijelölést az előbb ismertetett módszerrel, így magának a kijelölésnek a létrehozása rendkívül egyszerű: kiválasztom a megfelelő csatornát, majd a *Csatorna kijelöléssé alakítása* (*Channel to Selection*) gombra kattintok. Ekkor kijelöli a kitörlendő területet. Ezután kiválasztom képemből azt a réteget, amit módosítani szeretnék (a szörföst ábrázoló réteget). A *Réteg* (*Layer*) menüből válasszuk ki a *Rétegmaszk hozzáadása* (*New Layer*) menüpontot, teljes láthatósággal (tehát fehér színnel). Ezt követően nincs más hátra, mint a meglevő kijelölés alapján a rétegmaszkot feketével feltölteni – és máris eltűnt a háttér. Ha viszont a rétegmaszkot töröljük, akkor a kép maszkolt része is láthatóvá válik, tehát ennek a használatával képünk részleteit valójában nem töröljük, csak kitakarjuk. A kitakart hátteret a *14. kép* szemlélteti.

Következő írásunkban a folytatjuk maszkolást, majd érdekes montázsokat készítünk. Addig is jó gimpelést kívánunk mindenkinek!

Süveg Gábor
(gsuveg@sgmobil2000.hu)
Régóta használ Linuxot és BSD-t. Hobbija a búvárkodás, vitorlázás és a számítógépes grafika.

Testreszabott JSP-eljárások

Hogyan egyszerűsítsünk összetett Java-kódot?

A Linuxvilág előző számaiban a kiszolgálóoldali Java-alkalmazásokkal ismerkedtünk. A vizsgálódást a servletekkel, ezekkel a különleges Java-osztályokkal kezdtük, melyek a servlettárolóból hajtódnak végre. A programozókat a servletek használata nyilván nem rettenti el, a grafikusok véleménye viszont e tekintetben megoszlik.

Ezt a gondot oldja meg a JavaServer Page-módszer (JSP-k) használata, amely a Javát a HTML-lel egyesíti, miközben a Microsoft Active Server Pages-hez (ASP) vagy a mod_perl-hez tartozó, nyílt forráskódú HTML::Mason-rendszer formai követelményeihez hasonló nyelvet használnak. Minden JSP valójában egy álcázott servlet; a lapot a JSP-motor servletté, majd pedig Java .class állománnyá alakítja.

A JSP-k Java-kódot is tartalmazhatnak, ami egyszerűbbé teszi az összetett műveletek elvégzését. Egy bizonyos ponton túl azonban a kód elnyomja a HTML-t, ezáltal a JSP fenntarthatatlanná válik.

A nem programozók sem kedvelik, ha nagy mennyiségű kódot találnak a JSP-ben, így viszont a JSP előnye erősen csökken az egyszerű servletekkel szemben.

Előző írásunkban az olyan módszerek egyikét vizsgáltuk meg, amivel elkerülhetjük a JSP-n belüli kódhasználatot, nevezetesen a JavaBeans alkalmazását. Egyszerű XML-alapú tagok használatával akár egy nem programozó is képes összerakni egy összetett viselkedésű JSP-t anélkül, hogy egyetlen sor kódot is írnia kellene. A JavaBeans valódi ereje nem magukban a babokban (JavaBeans magyarul „Java Babok”-at jelent) rejlik, hanem azokban a különleges tagokban, amelyek segítségével oly könnyen előhívhatjuk azokat.

Most azt vizsgáljuk meg, hogyan írjunk saját – ahogy mondani szokás – „testreszabott eljárást”, azaz olyan XML-alapú tagokat, amelyek lehetővé teszik, hogy Java-osztályokkal és eljárásokkal dolgozzunk anélkül, hogy magával a Javával egyáltalán találkoznánk. Példáinkat a nyílt forráskódú Jakarta-Tomcat servlet és JSP-megvalósításhoz terveztük. Mindazonáltal valószínűleg bármely olyan JSP-megvalósítással működnek, amelyek a testreszabott eljárásokat támogatják.

Több oka is lehet, amiért testreszabott eljárásokat érdemes használni. Az első, hogy csökkentik a JSP-be illesztendő Java-kód mennyiségét, ezáltal azok könnyebben olvashatóvá, érthetővé és karbantarthatóvá válnak. A testreszabott tagok ráadásul sokkal kevésbé bonyolultak, mint egy Java-kód, így azokat szélesebb felhasználói réteg használhatja. Végezetül minden testreszabott tagkönyvtár egyetlen központilag írt és fenntartott Java-osztályra mutat. Testreszabott eljárások használatával a honlap pontosan olyan tagok könyvtárát állítja elő, amelyek éppen szükségesek. Egyetlen Java-programozó számos tervezőgrafikus és JSP-felhasználó számára készíthet és adhat ki tagkönyvtárakat. Amint azt látni fogjuk, a testreszabott tag ugyan nem csodaszer, de nagyon hasznos; a magam részéről ezt tartom az egyik legfontosabb érvnek, és emiatt érdemesebb JSP-t használni a versenytárs módszerekkel szemben.

Melyek is azok a testreszabott eljárások?

A testreszabott eljárások JSP-ből rövidebb utat biztosítanak a Java-kódhoz. Bármit, amit testreszabott eljárással meg lehet tenni, az megoldható `<% %>` tagok közti Java-kóddal is. Hiszen – mint tudjuk – a JSP servletté alakul, mielőtt a végfelhasználó számára lefordítódik és végrehajtódik.

Ahogy azt a múlt hónapban a JavaBean-tagok esetében láthattuk, a testreszabott eljárásokat HTML helyett egyszerű XML-tagokkal határozhatjuk meg. Ez elsőre némileg zavaró és elkeserítő lehet, különösen azoknak, akik hibás HTML-írási szokásokat vettek fel. A következő sor például helyesnek tűnhet:

```
<P><jsp:getProperty name="simple"
➥property="userID"></P>
```

Valójában azonban a fenti sor nem fog működni és kivételt, illetve veremkövetést (stack trace) vált ki a JSP-ben. Ez azért van így, mert XML-ben minden tagot le kell zárnunk valahogyan. Ha a `<tag>`-nak nincsen lezáró `</tag>` párja, akkor jelölni kell, hogy saját magát zárja le: `<tag/>`. A fenti sort tehát így kell helyesen írni:

```
<P><jsp:getProperty name="simple"
➥property="userID"/></P>
```

A testreszabott eljárás tulajdonképpen csak formaikövetelmény-máz a Java-tagfüggvényeken. Minden egyes tagkönyvtár egy-egy eljáráskészletet határoz meg, a jsp tagkönyvtár például három eljárást ad meg: `getProperty`, `setProperty` és `useBean`. Minden eljárást egy külön Java-osztály határoz meg, amit tagkezelőnek (tag handler) nevezünk.

A tagkönyvtár meghatározásához először egy XML fájlt készítünk, amelyet tagkönyvtár-leírónak (tag library descriptor, azaz TLD) neveznek. A TLD az eljárásokat a megfelelő tagkezelő osztályhoz rendeli, felsorolva a választható és kötelező értékeket, illetve a taghoz kötődő egyéb adatokat.

Ha JSP-ből szeretnénk használni a testreszabott eljárást, TLD-nk betöltésére különleges vezérlőjelet használunk. Ez segít a JSP-motornak, hogy a JSP-nkben talált testreszabott tagokat ellenőrizze, illetve a hozzájuk tartozó kezelőosztályt megtalálja.

Egyszerű testreszabott eljárások

Itt az ideje, hogy egy olyan egyszerű eljárást határozzunk meg, ami alapján érthetővé válik a tagkezelő osztályok, a TLD-k és a JSP-k működése mögött rejlő szerkezet. Testreszabott eljárásunk legyen a *hello* tag, amely kezeli az elhagyható *firstname* értéket. Ha a *firstname* létezik, a tag egyszerű üdvözlő üzenetet küld a megnevezett felhasználónak; ha pedig az érték hiányzik, a tag egy általános üzenetet készít.

Az első lépés egy egyszerű tagkezelő írása, amely ezt a feladatot megvalósítja. Ilyen tagkezelőt mutat be az *1. lista* a *HelloTag* osztály meghatározásával. A HelloTag.java forrásfájlt az összes JSP és servlettel kapcsolatos osztállyal együtt a `$TOMCAT_HOME/classes` könyvtárba helyeztem. Mivel a HelloTag.java az il.co.lerner csomagban található, és miután a `$TOMCAT_HOME` az én gépemen a /usr/java/jakarta-tomcat-3.2.1 könyvtár, a Java-forrás nálam a következő helyen található:

```
/usr/java/jakarta-tomcat-
3.2.1/classes/il/co/lerner/HelloTag.java
```

1. lista HelloTag tagkezelő

```
package il.co.lerner;

import java.io.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class HelloTag extends TagSupport {

  private String firstname = null;
  // null alapértékű mező
  // ----------------------------------------
  // Bean-stílusú "set" tagfüggvény

  public void setFirstname (String newFirstname)
                           {

     // Csak akkor állítjuk be a firstname
     // mezőt, ha nem üres.
     if (! newFirstname.equals(""))
         {
              firstname = newFirstname;
         }
  }

  // ----------------------------------------

  public int doEndTag() throws JspException {

     // Szúrjunk be némi szöveget a JSP-be,
     // vagy lépjünk ki!
     try {

          // Ha nem volt név megadva, általános
          // üzenetet adunk
          if (firstname == null)
               {
                  pageContext.getOut().println
                    ("Üdvözlöm!");
               }
          else
               {
                  pageContext.getOut().println
                    ("Üdv, " + firstname +
                     "!");
               }

     } catch (IOException e) {
       throw new JspException(e.getMessage());
     }

        // Haladjunk tovább, és értelmezzük
        // a JSP további részeit!
        return EVAL_PAGE;
  }

  // ----------------------------------------

  public void release()
  {
       firstname = null;
       super.release();
  }

}
```

A HelloTag.java HelloTag.class osztállyá történő fordítása után ezt a tagkezelőt egy vagy több más tagkönyvtárral akár egybe is lehet olvasztani. Minden tagkezelő osztálynak tartalmaznia kell a két különböző alapcsatolófelület (*Tag* vagy *BodyTag*) egyikét. (Utóbbi azoknál a testreszabott eljárásoknál használatos, amelyeknek nyitó és záró tagjuk között „testük” van, ellentétben azokkal, amelyeknek egyáltalán nincs testük, és ezekkel e hónapban ismerkedünk meg.)
A gyakorlatban ezeket a csatolófelületeket nemigen van értelme megvalósítani. Sokkal egyszerűbb és célszerűbb a *TagSupport* és *BodyTagSupport* osztályoktól örökölni őket, amelyek a megfelelő alapértelmezett csatolófelületeket tartalmazzák. A *TagSupport* alosztály-lehetőséget kihasználva megtakaríthatunk némi munkát, és csak azokat a tagfüggvényeket írjuk felül, amelyeket az alapértelmezettől eltérő módon szeretnénk felhasználni. Végezetül HelloTag-megoldásunk mindössze három tagfüggvényt tartalmaz: *setFirstname doEndTag* és *release*.
Az első tagfüggvény, a *setFirstname*, úgy néz ki és úgy is működik, mint egy JavaBean tulajdonságbeállító tagfüggvény, egyetlen értéket fogad el és *void*-ot ad vissza. A setFirstname önműködően hívódik meg, amikor a JSP-motor a *firstname* értéket észleli saját JSP-eljárásunkban. Az érték a tagban átadott adattal töltődik fel, és akárcsak a JavaBeansben, a *firstname*-et beállító tagfüggvény nevének *setFirstname*-nek kell lennie, ahol a nagy *F* betű kötelező.
Második tagfüggvényként a *doEndTag* akkor hívódik meg, amikor a JSP-motor eléri testreszabott eljárásunk bezáró tagját. A *doEndTag* tagfüggvény nem vár értéket és egy egész számot ad vissza. Mi azonban egész szám visszaadása helyett a számunkra biztosított egyik állandót adjuk vissza. Általános esetben `EVAL_PAGE`-et adunk vissza, amely értesíti a JSP-motort, hogy folytathatja annak a JSP-nek az értelmezését, amelyből testreszabott eljárásunk meghívódott. Ha a JSP-motor fájlértelmezését le szeretnénk állítani, akár azért, mert hibát találtunk, akár mert a felhasználót egy másik címre szeretnénk irányítani, egyszerűen `SKIP_PAGE`-et adjunk vissza.
A *doEndTag* belsejében bármilyen Java-kódot elhelyezhetünk. Az általunk készített változókon kívül magáról a JSP-ről szóló adatokhoz is hozzáférhetünk, ideértve a kapott HTTP-kérelmet és -választ.
A felhasználó böngészőjére úgy írhatunk adatokat, hogy a testreszabható tagot HTML-, XML- vagy egyszerű szöveggel helyettesítjük. (A testreszabott eljárások többnyire egyszerű szöveget adnak vissza és a szöveg pontos formázását a JSP szerzőjére bízzák.)
A *TagSupport* szuperosztályban bevezetett *PageContext* objektum használatával megkaphatjuk a kimeneti folyamot és adatokat is küldhetünk rá:

```
pageContext.getOut().println("Hi there!");
```

Végül meghatározzuk a *release* tagfüggvényt, amely semmilyen értéket nem fogad és *void*-ot ad vissza. A `release()` a testreszabott eljárás végrehajtásának végén hívódik meg, ezáltal lehetőséget ad a tagkezelő osztálynak, hogy kitakarítson maga után. Ez általában annyit jelent, hogy az összes belső változót *null*-ra állítja, de adatbáziskapcsolat lezárása vagy valamilyen adat hibanaplóba küldése is állhatna itt. A HelloTag.java-ban egyszerűen csak *null*-ra állítjuk a *firstname*-et,

majd arra kérjük a szuperosztályt, hogy saját változóit nullázza.
Most hogy megértettük, hogyan működnek az egyes HelloTag tagfüggvények, már csak az a kérdés: miképpen működnek együtt? Ha a JSP (az alább ismertetett TLD-n keresztül) osztályunkhoz rendelt testreszabott eljárást tartalmaz, az eljárás összes értéke meghívja osztályunk megfelelő *set* tagfüggvényét. Ha valaki például a `firstname="foo"` értéket adja át, akkor tulajdonképpen a *setFirstname("foo")* tagfüggvényt hívja meg.
Mivel azt szeretnénk, hogy a *firstname* érték elhagyható legyen, amikor először elkészítjük, `null` alapértéket adunk neki. Amikor a JSP-motor a testreszabott eljárás értelmezését befejezi, meghívja a *doEndTag*-et és megvizsgálja a firstname értékét. Amennyiben a *firstname* `null`, akkor az általános *(Üdvözlet)* üzenetet küldi a végfelhasználónak. Ha viszont nem `null`, akkor az értéket a *doEndTag* egy valamivel személyesebb üzenethez használja fel. Amikor a testreszabott eljárás végrehajtása befejeződik, a JSP-motor meghívja a `release()`-t és alaphelyzetbe állítja a *firstname* változót, valamint néhány egyéb objektumot.

TLD írása

Ha az osztály megírásával elkészültünk, TLD-t hozhatunk létre, amely leírja azt a JSP-motornak. Néhány esetben kedvezőbb a fordított irány, ha a TLD-t használjuk leírásnak, amiből a JSP-szerzők és a tagkezelőírók párhuzamosan tudnak dolgozni. Én jobban szeretem előbb megírni a testreszabott eljárást és menetközben módosítgatni a TLD-t, annak ellenére, hogy ez nyilvánvalóan nem éppen a legbiztonságosabb és nem is a legelegánsabb munkamódszer.
A TLD, amint az a 2. listában látható (15. CD, Magazin/JSP könyvtár), egy viszonylag rövid XML fájl, amely az eljárásneveket a megvalósítást végző osztályokhoz rendeli. A TLD egyetlen eljárást egyetlen osztályhoz vagy akár ezernyi eljárást ezernyi különféle osztályhoz rendelhet. Mivel minden osztály elkülönítve létezik, még az is lehetséges (bár nem tűnik túl jó ötletnek), hogy az osztályt egyszerre több TLD-ben is felhasználjuk.
A TLD az első hivatkozás után servlettárolónkba töltődik. Ez azt is jelenti, hogy ha egy már betöltődött, testreszabott eljáráshoz tartozó TLD-t változtatunk meg, akkor sajnos újra kell indítanunk a Tomcatet (és az Apache-t, ha az Apache mod_jk-t használjuk a Tomcat kiszolgálóhoz). A JSP-motor ilyenkor tudja meg, hogy tagkönyvtárunk melyik változatot és szabványt támogatja. Ezáltal lehetővé válik, hogy a JSP-motor kitalálja, szükséges-e az adott könyvtárat frissíteni vagy sem, hogy a jelenlegi szabvánnyal összeegyeztethető legyen.
A TLD egy legfelsőbb szintű `<taglib>` tagból áll, amely legalább négy alrészt tartalmaz:

- a `<tlibversion>` tartalmazza, hogy ez a könyvtár mely változatú tagkönyvtár-szabványnak felel meg;
- a `<jspversion>` azon JSP-előírások változatszámát adja meg, amely szerint ez a könyvtár készült;
- a `<shortname>` nevet ad az adott tagkönyvtárnak, amelyet néhány JSP-motor ki is használ; végül minden, a tagkönyvtárban tárolni kívánt tagkezelőnél egyszer szerepel a `<tag>` tag. Minden tag saját nevet kap: a meghívandó eljárás nevét. Így ha egy tagkönyvtárat az *abc*-előtaggal importálunk, a *hello* nevű tag *abc:hello*-ként hívódik majd meg. A `<tagclass>` alrész a tagnevet az eljárást végző tagkezelő osztályhoz rendeli; ennek az osztálynak nyilvánvaló módon a kiszolgáló `CLASSPATH`-jában kell lennie. Az `<info>` alrész módot ad arra, hogy néhány, a taggal kapcsolatos alapadatot és sorközi leírást adjunk közre.

Végül az összes paramétert elnevezzük, amit testreszabott eljárásunk elfogad. Minden értéknek saját `<name>` tagja és egy jele van, ami megmutatja, elhagyható-e az adott érték.

Testreszabott eljárások használata JSP-kben

Most, hogy elkészítettük a TLD-t és a tagkezelőt, ezeket bármely JSP-nkben felhasználhatjuk. A tagkönyvtárat a `taglib` nevű egyedi JSP-utasítással importálhatjuk:

```
<%@ taglib uri="/WEB-INF/hello.tld"
➥prefix="hello" %>
```

Figyeljük meg, hogy a `taglib` utasítás két értéket vár, egy *uri* és egy *prefix* nevűt. Az *uri*-rész határozza meg a pillanatnyilag készített TLD fájl nevét. Ha a TLD-ket a `WEB-INF` könyvtárban szeretnénk elhelyezni, akkor a fenti formai követelmény teljes mértékben helyes. A *prefix* érték tulajdonképpen névtér- (namespace) meghatározás, ami tudatja a JSP-motorral, hogy a tagkönyvtárból importált eljárásoknál milyen előtagot fogunk használni. Azáltal, hogy a JSP-nek adtuk meg ezt a megkülönböztető előtagot (ahelyett, hogy a tagkönyvtárba fordítottuk volna), lehetővé válik, hogy egyszerre több tagkönyvtárat is importáljunk anélkül, hogy a névütközések miatt aggódnunk kellene.
Mivel TLD-nk mindössze egyetlen *hello*-tagot tartalmaz, és mivel a tagkönyvtárat a *hello*-előtaggal importáltuk, *HelloTag* tagfüggvényünket a következő formátumban hívhatjuk meg: `<hello:hello/>`. A *3. lista* a tag használatát bemutató teljes JSP-t tartalmazza (test-tag.jsp).
Ne feledjük el a testreszabott eljárások meghívásánál a lezáró perjelet kitenni! Ha erről megfeledkezünk, a Tomcat JSP-motor (más néven Jasper) a következő hibaüzenetet fogja visszaadni:

```
Unterminated user-defined tag:
ending tag &lt;/hello:hello&gt; not found or
incorrectly nested
```

A TLD szerint a *firstname* érték elhagyható. Amennyiben nem adjuk át a *firstname* értéket, a következő kimenetet kapjuk a böngészőn:

```
Testreszabott eljárásteszt.
Üdvözlöm!
```

A *firstname* értéket azonban át is adhatjuk:

```
<hello:hello firstname="Reuven"/>
```

Ha a fenti sort a JSP-be helyezzük, a böngészőre a következő kimenet kerül:

```
Testreszabott eljárásteszt.
Üdv, Reuven!
```

Összetettebb testreszabott eljárások

A fenti példa meglehetősen egyszerű testreszabott eljárás volt. A testreszabott eljárás-tagok sokkal többre is képesek, mint egyszerűen neveket írni a képernyőre. Például az objektumok adatbázisokkal léphetnek kapcsolatba, ott adatokat kérhetnek le (vagy tárolhatnak) anélkül, hogy közvetlen Java-hívásokat helyeznénk a JSP-be. Sőt, a testreszabott eljárásokat ismétlőelemekként vagy feltételes végrehajtásra is használhatjuk.
Hogy ezeket az összetettebb eljárásokat megvalósíthassuk, kihasználjuk, hogy a tagkezelő osztály a testreszabott eljárás testében is tud keresni; azaz abban a szövegben, ami az eljárás nyitó- és zárótagja közt található. Ezzel a szöveggel bármit megtehetünk: többszöröz-hetjük, feltételes elágazást készíthetünk vagy akár meg is kérhetjük a JSP-motort, hogy értelmezze a tartalmat, mielőtt átadná a tagkezelőnek. Még a tagok egymásba ágyazása is lehetséges, ahol az egyik eljárás hatékonyan adhat át értékeket a másiknak.

3. lista test-tag.jsp

```
<%@ taglib uri="/WEB-INF/hello.tld"
prefix="hello" %>

<HTML>
<Head>
    <Title>Testreszabott eljárásteszt</Title>
</Head>

<Body>
    <P>Testreszabott eljárásteszt.</P>

    <P><hello:hello/></P>

</Body>

</HTML>
```

Számos nyílt forrású tagkönyvtár létezik, ideértve azt is, amelyet a Jakarta Project támogat, és amely ügyesen használja ezeket a lehetőségeket, számos taggal növelve a lehetőségek tárházát.

Mire jók a testreszabott eljárások?

A testreszabott eljárások félelmetesen hatékony eszközök. A JSP-be helyezett közvetlen Java-kóddal szemben előnyök széles skáláját biztosítják: összetett feladatokat zárnak könnyen megjegyezhető tagokba, ami a nem programozók számára is lehetővé teszi, hogy adatbázisokkal vagy hasonló nem egyértelmű rendszerekkel dolgozzanak.

A testreszabott eljárásoknak azonban árnyoldaluk is van, melynek nyitja pontosan a „testreszabott” szóban rejlik. Az, hogy a JSP-k belsejében saját tagokat adhatunk meg, ügyes és kifinomult eszköz, és a honlap minden fejlesztője számára előnyökkel jár. Csakhogy a Web egyik legnagyszerűbb tulajdonsága éppen az, hogy meglehetősen szabványosított. A testreszabott eljárások segítségével ráadásul akár egy teljes, önálló nyelvet is készíthetünk Javában és tagkezelő osztályoknak feleltethetjük meg. *Hans Bergsten*, akinek JavaServer Pages című könyve kitűnő forrásmunka a JSP-k utasításai terén, ezt az alapötletet egészen a végletekig követi – tulajdonképpen teljes egészében feleslegessé téve a Java-használatot a JSP-kben. Engem viszont aggodalommal tölt el, ha egy ilyen viszonylag megbízható és jól ismert nyelvet, mint a Java, egy új, sokkal kevésbé ismert és sokkal kevésbé harcedzett nyelvvel (saját testreszabott könyvtárral) helyettesítenek.

Ha egy nagy cégnél dolgoznék, amely a Java, a servletek és a JSP felhasználását komolyabban tervezi, a testreszabott eljárások használatát valószínűleg nagyon kényelmesnek találnám. Egy ilyen cég megteheti, hogy elkészíti a saját tagkönyvtárát, amelyet aztán a honlap teljes élettartama alatt felhasználhat, valamint hogy saját szabványt alkot dolgai működtetésére.

Azok számára azonban, akik nem nagy cégeknél dolgoznak, vagy akik számos különböző ügyféllel állnak kapcsolatban, az átalakíthatóság a legfontosabb szempont. Ha összes ügyfelem saját honlapjához más és más testreszabott eljáráskészletet szeretne meghatározni, igencsak bajban lennék, hisz emlékeznem kellene, hogy hol melyik tagot és értéket kell használnom a ciklusokhoz, az adatbázisokhoz vagy az elágazásokhoz.

Továbbá, mint azt már korábban említettem, a nem programozók miatt is aggódom, akik már eleve sokat küszködtek, mire a HTML-lapokba ágyazott Javát megtanulták használni – és most, hogy két különféle ciklust is meg kell tanulniuk (egy javásat, és egy másikat, amit a testreszabott eljárás biztosít), szinte biztosan összezavarodnak.

Viszonylagos megoldást nyújthat egy nagy, szabványos testreszabott eljáráskészlet alkalmazása, amely a JSP-szabvány része lenne, valahogy úgy, ahogyan az a JavaBean-tagokkal is történt. A Bergsten-féle JavaServer Pages című könyvben szereplő könyvtár jó kiinduló alap lehetne, de ez csak egy a számos elérhető könyvtár közül. Jó volna látni, hogy a JSP-közösség ebben a témában összefog, mielőtt tucatnyi hasonló, de egymásnak még véletlenül sem megfelelő könyvtárral találkoznánk, melyek közül jó néhány kétségtelenül jogvédett lesz.

Összegzés

A JSP-kkel való munka hatékony és gyors módszer a kiszolgálóoldali Java-alkalmazások készítésében, különösen az olyan nem programozók számára, akik semmilyen nyelvet nem szeretnének megtanulni. A testreszabott eljárások – különösen JavaBean-elemekkel együttműködve – lehetővé teszik, hogy összetett feladatokat kevés kóddal oldjunk meg. Egy kis megfontoltsággal a honlap akár úgy is elkészülhet, hogy a JSP-kbe egyetlen Java-sor sem kerül, mert azok teljes egészükben testreszabott eljárásokon és tagkönyvtárakon alapulnak. A webhelyek (és a testreszabott eljárásokat használó tanácsadók) azonban nem árt, ha figyelmet fordítanak rá, hogy a tagkönyvtárak kényelmét és hatékonyságát élvezve esetleg egy új programozási nyelvet alkotnak. Ha óvatlanok vagyunk, a testreszabott tagok megoszthatják az ügyféloldali Java-közösséget, különböző összeférhetetlen könyvtárakat használó alközösségekre tördelve azt szét.

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 Apache Programalapítvány Jakarta Projectje, mely számos Java-vonatkozású fejlesztésnek ad otthont, a ➲ http://jakarta.apache.org/ címen található.

A Tomcat – a Jakarta Project része – a Sun servlet és JSP-előírásnak megfelelő nyílt forráskódú megoldáson dolgozik. A Jakarta-honlap mindig a Tomcat legújabb forrás-, bináris és fejlesztői állományait nyújtja az érdeklődőknek.

JSP-témában kitűnő forrás az O'Reilly kiadásában nemrégiben megjelent JavaServer Pages című mű *Hans Bergstentől*. Véleményem szerint a könyv túlságosan is a testreszabott eljárások alkalmazása felé hajlik, de ha ilyenfajta megoldást szeretnénk alkalmazni, vagy egyszerűen csak arra vagyunk kíváncsiak, hogyan lehet JSP-ket írni és használni, kezdetnek mindenképpen kitűnő.

Számos, a testreszabott tagokról és a JSP-kről szóló hálózati (online) útmutató is létezik. A Jakarta-Tomcat honlap szintén rendelkezik egy ilyen ismertetővel, ➲ http://jakarta.apache.org/taglibs/tutorial.html. Ebben a témában egy másik tájékoztató egyenesen a Suntól is letölthető a ➲ http://java.sun.com/products/jsp/tutorial/TagLibrariesTOC.html címről.

Vége a szótlan programozásnak!

Beszédfelismerő programmal egy lépéssel közelebb HAL-hoz.

Ugye, François, ez csodálatos? Kisgyermekkorom óta – még mielőtt ennek az étteremnek a megnyitására gondolhattam volna – egyfolytában valami ehhez foghatóról álmodoztam. Ahogy visszaemlékszem, a 2001: Űrodüsszeiát néztem, a HAL 9000-es számítógép hangját hallottam, és arra gondoltam: „Ez az, amit szeretnék! Egy beszélő számítógép!” Évek múlva rá kellett ébrednem, hogy a számítógépem még mindig nem képes beszélni. Elérkezett azonban a nap, amikor mindez megváltozhat. Hogyan? Már itt is vannak a vendégeink? Isten hozott benneteket Chez Marcelnél! Nagyon örülök az érkezéseteknek! Ma az igényes programozó csodálatos lehetőségek közül választhat az étlapon. Foglaljatok helyet, François rögtön hoz valamilyen finom bort. François, hozz fel a pincéből egy palack 1996-os évjáratú Ausztráliából származó Hill of Grace-t! Kedvelni fogjátok ezt a pompás ízt. Köszönöm, François. Igen, tölts kérlek a vendégeinknek!

Épp az imént mondtam Françoisnak, hogy manapság már mindenfelé beszélő számítógépeket kellene használnunk, de a linuxos gépem mégiscsak szótlanul dolgozik. Számomra érthetetlen, miért nem képes a beszédre egyetlenegy rendszeremre telepített program sem! Nos, mai receptjeinkhez a számítógépedbe szerelt és megfelelően beállított hangkártyára, valamint egy mikrofonra lesz szükséged.

A skóciai Edinburghi Egyetem Beszédtechnológiai Kutató Központja (CSTR) pontosan azt kínálja, amivel nekivághatunk a vágyott útnak – amelynek megtétele után már a saját számítógépünk is beszélni lesz képes! Aki ellátogat a ➲ http://www.cstr.ed.ac.uk/projects/festival címre, rábukkanhat a Festival elnevezésű izgalmas vállalkozásra.

A Festival többnyelvű beszédkészítő rendszer, amely számos emberi nyelven képes a begépelt szöveget beszéddé alakítani. A program API-felületével több további programba és alkalmazásba illeszthető be. Amint ezt a csomagot kibontjuk, rögvest megláthatod, mit is értek ez alatt.

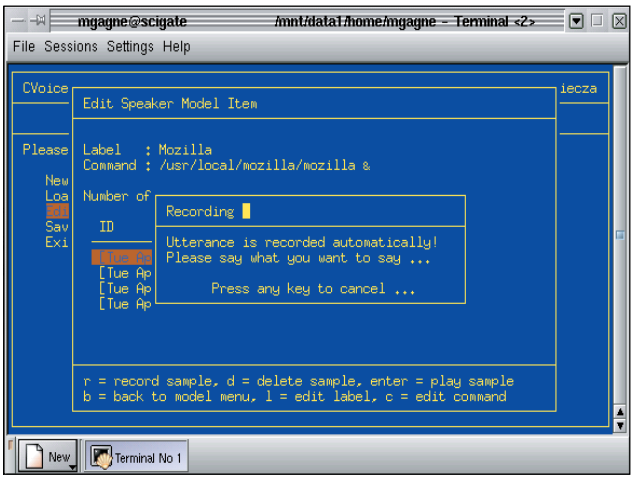
A Festivalhoz könnyű hozzájutni, csak meg kell keresned a fent említett helyet a Világhón, majd kattints a letöltés gombra – és máris a programcsomag legfrissebb változatát kapod kézhez. Ha már úgyis ott jársz, a speech_tools eszközkészletet is vedd magadhoz, mivel a Festival feltételezi ennek meglétét. Az első program, amelyet összeépítünk, következésképpen az alábbi kell legyen:

```
tar -xzvf speech_tools-1.2.1.tar.gz
cd speech_tools
cp config/config-dist config/config
chmod +w config/config
```

Ezen a ponton eldöntheted, szeretnél-e megosztott könyvtárakat használni, az alapértelmezés ugyanis ezt nem teszi lehetővé. Amennyiben igényled, a # jel törlésével a config/config állományban meg kell szüntetni a megjegyzést.

```
#SHARED=1.
```

Ez javasolt választás, és akár élsz ezzel a lehetőséggel, akár nem, az összeépítést tovább folytathatod.



Parancs hozzáadása a model_editorban

```
make info
make
make install
```

Elérkezett a lazítás ideje: kóstoljuk meg a libamájat (ez csakugyan finom, nem igaz?) és kortyintsunk egyet a borunkból! Amint a speech_tools összeépítése befejeződött, belekezdhetünk a Festival-rendszer létrehozásába. Bontsuk ki a rendszer forráskódját egy tetszőlegesen kiválasztott könyvtárba a `tar -xzvf festival-1.4.1.tar.gz` paranccsal.

E műveletet követően a benne szereplő összes állomány a */festival* könyvtárba kerül. Mielőtt bármi mást tennénk, végezzük el a nyelvi lexikon és a beszédadatbázis kibontását is. A munkát az alábbi állományokkal kezdtem el:

```
festlex_CMU.tar.gz
festlex_POSLEX.tar.gz
festvox_kallpc16k.tar.gz
```

A CMU elnevezésű könyvtár az angol beszédhangok, a POSLEX könyvtár pedig az általánosan használt angol nyelvi elemek tárolására szolgál. Végül helyére kerülhet a 16 k-s mintavételezéssel készült amerikai angol kiejtésminta-adatbázis is. A világ más-más részén élő olvasók kétségkívül eltérő igényekkel lépnek majd fel a beszélők hangját, nemét vagy nemzetiségét (angol vagy francia) illetően.

A szükséges feltételek részleteiről a Világháló fent említett helyén, a *Distribution* könyvtárban található README állományból tudhatunk meg a legtöbbet.

Ha ezeket az állományokat kibontod, a művelet befejezése után ugyanebbe a */festival* könyvtárba fognak kerülni. Innentől kezdve a folyamat megegyezik a speech_toolsszal végzett műveletsorozattal, a *config-dist* könyvtárból való config állomány másolásával bezárólag, amit természetesen ezúttal is a */festival* könyvtárba másolunk. Miután mindezzel végeztünk, a telepítő könyvtárból adjuk ki a `bin/festival` parancsot. Ekkor várhatóan a következő sorok fognak megjelenni a képernyőn:

```
bin/festival
Festival Speech Synthesis System 1.4.1:
release November 1999
Copyright (C) University of Edinburgh, 1996-1999.
All rights reserved.
For details type `(festival_warranty)´
festival>
```

Felkészültél arra, hogy most valóban *hallani* fogod, hogyan beszél a számítógéped? Ha a kérdésre igen a válasz, akkor a parancssorjelnél (>) gépeld be a következőket (ügyelj a zárójelekre és idézőjelekre, mert ezek is fontosak!):

```
(SayText "François. Vite. More wine.")
```

Feltéve, hogy minden megfelelően zajlott, a hangszórókból a következő szavakat hallod: François. Vite. More wine. Úgy találod, hogy meglehetősen parancsoló ez a hangnem? Részemről szólva szeretek ezzel a sorral játszani… hűséges pincéremet kissé kihozza a sodrából. Angol kiejtést és adatbázist használok, bizonyára ennek köszönhető, hogy a hanglejtés és a hangsúlyozás érdekesen hat. A CONTROL+D billentyűk együttes lenyomásával lehet kilépni a Festival parancsmódú végrehajtásából, de a *quit* is használható erre. Szeretném még egyszer a zárójelek fontosságára felhívni a figyelmet. Most próbáljunk ki valami érdekeset! A `--tts` kapcsoló segítségével megadhatjuk egy szövegállomány elérési útvonalát és rábírhatjuk a Festivalt, hogy a kiválasztott szöveget felolvassa nekünk. Vegyük például a cron ütemezőt, amely a fortune program lefuttatásával minden éjjel frissíti a nap üzenetét. Az üzenetet az alábbi módon olvastathatjuk fel:

```
bin/festival --tts /etc/motd
```

Ha a parancs begépelésekor nem adunk meg állománynevet, máris elkezdhetjük a beírást, a szöveg végére érve pedig nyomjuk le a CONTROL+D billentyűkombinációt – ezt követően a Festival program végrehajtása befejeződik. Akad itt még valami, amit érdemes kipróbálni: vezessük csővezetékbe a Festival program kimenetét! Kíváncsi vagy esetleg a dátum valamilyen egyéni hangú értelmezésére? Ha igen a válaszod, próbáld ki: `date | bin/festival --tts`.
A Fortune programot akkor is érdemes futtatni, ha ennél azért nagyobb bölcsesség birtokába szeretnél jutni:

```
/usr/games/fortune | bin/festival --tts
```

A Festival a `--server` kapcsoló megadásával kiszolgálóként is működtethető, így további program(ok) számára képes szöveges adatokat közvetíteni. Készíthető például olyan program, amely kimenetét a Festival csatolójára küldi: ez alapértelmezés szerint az 1314-es kapu. A *lista* egy apró Perl-programot tartalmaz, amelyet pontosan e célból írtam. Nem különösebben érdekes, de tekintheted alkalmazásaid kiindulópontjának. Ne feledkezz meg arról, hogy első alkalommal szükséges lehet a Perl végrehajtható modulok elérési útvonalának megváltoztatása.
Ezt az ötletet bámulatra méltó módon valósítja meg a Darxus által írt *speechd* nevű program. Ez olyan készülékleíró állományt telepít – a /dev/speech névvel hivatkozhatunk rá –, ami tetszőleges szöveg fogadására képes. Irányítsuk át a programkimenetet erre a készülékre, ezt a kiszolgálómódban futó Festival program kapja meg, majd hangosan felolvassa. Ez megint csak egy Perl-héjprogram, amit a SpeechIO szervezet honlapjáról tölthetsz le, címe
➲ http://www.SpeechIO.org/.
A tömörített állományokat csomagoljuk ki egy ideiglenes könyvtárba, és ott végezzük el a program összeépítését, az alábbi módon:

Írás a Festival program csatolójára

```
#!/usr/bin/perl
#
# Írás a Festival program csatolójára
# Marcel Gagne - 2001
#
$remote_port="1314";
$festival_host="localhost";
$my_message="Francois, come here.  I need you.";

use IO::Socket;
$socket = IO::Socket::INET->new
                  (PeerAddr => $festival_host,
                  PeerPort  => $remote_port,
                  Proto     => "tcp",
                  Type      => SOCK_STREAM)
   or die "Nincs élő kapcsolat".
$remote_host:$remote_port : $@\n";

# Írás a Festival kiszolgáló kapura
if ($socket) {    # Csak ellenőrzés
   print($socket ´(SayText "´,$my_message,´")´)
or
         die "Fut a Festival kiszolgáló?\n";
         }

# Csatlakozás lezárása, ha még nem volt
# lezárva.
close($socket);
```

```
tar -xzvf speechd-0.54.tar.gz
cd speechd
make
make install
```

A program démonként való futtatásához a parancssorba a következőt gépeljük be:

```
/usr/local/bin/speechd.
```

Mielőtt folytatnám, el kell mondanom, hogy itt gondjaim támadtak. RedHat-rendszerem /etc/hosts állományában az alábbi bejegyzés szerepelt:

```
127.0.0.1   localhost.localdomain  localhost
```

Elképzelhető, hogy a te rendszerednél is hasonló a helyzet. A helyes működéshez a két meghatározást a következő módon fel kellett cserélnem:

```
127.0.0.1   localhost localhost.localdomain
```

Előfordulhat, hogy a Festival kiszolgálóprogram éppen nem működik, ekkor ezt a speechd megpróbálja majd elindítani. Abban az esetben, ha a Festivalt forráskódból építetted fel, sajnos szükséges lehet a speechd héjprogramjának olyan módosítására, hogy a végrehajtható programmodul teljes elérési útvonalát használja. A másik lehetőség a Festival bináris moduljainak a /usr/local/bin könyvtárba történő másolása. De mit lehet ezzel kezdeni? Visszatérve a Fortune-progra-

mos példához, a program kimenetét egyszerűen átirányíthatom a /dev/speech állományba:

```
/usr/games/fortune > /dev/speech
```

Ezt már roppant könnyű lesz beépíteni a héjprogramjaidba. Álljon itt egy további példa is! Lehetőség nyílik rá, hogy egy héjprogramon – amely néhány percenként lefut és ellenőrzi a bejövő postát – megszámolja az üzeneteket, továbbá hogy ezt a beszédkészüléken keresztül hangosan fel is olvassa nekünk. Fontos megjegyezni, hogy az `frm` parancs tulajdonképpen fordított percjelek.

```
echo "You have `frm | wc -l` messages in your
     ➥mailbox" > /dev/speech
```

Miután Linux-rendszerünket sikerült szóra bírni, úgy tűnik, már csak egyvalami hiányzik. Meg kell tanítanunk a rendszert, hogy „megértse” az emberi szót és végrehajtsa, amit mondunk neki, vagyis *beszédfelismerő* programra van szükségünk. A szóban forgó testreszabás megvalósításához a Világhálón felkerestem *Daniel Kiecza* honlapját a beszédalapú irányítóprogram, a *cvoicecontrol* forráskódjáért, ezt a csomagot ugyanis a GPL-szabályok szerint terjesztik.
A cvoicecontrol segítségével belefoghatunk álmaink rendszerének létrehozásába. Természetesen ezentúl már igencsak óvatosnak kell lennünk, hogy mit mondunk a számítógép közelében!

```
tar -xzvf cvoicecontrol-0.9alpha.tar.gz
cd cvoicecontrol-0.9alpha
./configure
make
make install
```

A parancsok végrehajtása nyomán keletkező állományok a /usr/local/bin könyvtárban jelennek meg. Három tényezőt célszerű figyelembe venni: az első mindjárt maga a cvoicecontrol program. Használat előtt állítsuk be a mikrofont és hozzunk létre hangminta-állományokat! E feladatokat a microphone_config, illetve a model_editor programokkal leszünk képesek megoldani. Kezdjük a beállítást a microphone_config program segítségével, a párbeszédek végigkövetésével. Az alapértelmezett audiokészülékeket és keverőberendezéseket a rendszer önműködően érzékeli (elvileg már a megfelelő helyen ki vannak töltve). Az én rendszeremben ezek */dev/mixer*-ként és */dev/dsp*-ként jelentek meg. A következő lépés a keverőberendezés szintjeinek beállítása, a hangfelvételek zajszintjének meghatározása és a beállítóállomány létrehozása. Ebben a szakaszban valószínűleg az a legnehezebb feladat, hogy eléggé hangosan legyünk képesek beszélni mindaddig, amíg a program beállítja a működési jellemzőket. Már most elárulhatom, hogy ez sokkal nehezebb feladatnak fog bizonyulni, mint amilyennek hangzik. Ha az állomány elhelyezése felől külön nem rendelkezünk, akkor az állománynak az elérési útvonala a $HOME/.cvoicecontrol/config lesz. Ezután kezdődjék a móka: indítsd el a model_editort programot! A képernyőn megjelenik a menü, amelyen már létező hangmintákat tudsz betölteni vagy újakat alkothatsz, szerkeszthetsz és menthetsz el. Vess egy pillantást *képünkre*, amelyen működés közben láthatod a programot. Rajta, barátaim, most a ti hangotokat fogjuk mintaként használni! Minthogy egyelőre semmilyen beszédmintánk sincs, válasszuk a *New Speaker* (Új beszélő) lehetőséget, ezt követően pedig az *Edit*-et (Szerkesztést). Az újabb menüben szavakat rögzíthetünk és megadhatjuk, hogy mi történjen ezek kiejtésekor. Mindezt egy billentyű, az A (az angol *add* szóból elvonva) lenyomásával érhetjük el, amellyel új eseményt tudunk megadni. Ez először olyan általános elemként fog megjelenni a listában, amelyhez még nem tartozik parancs. Az ENTER gomb lenyomását követően beállíthatók az esemény jellemzői és beszédes címke adható, amely a megvalósított szolgáltatást jelzi. Egy ilyen indítóelemet hoztam létre például a Mozilla indításához és elneveztem *Start Mozilla*-nak, ezt követően pedig begépeltem a `/usr/local/mozilla/mozilla &` A `Mozilla indítása` parancsot. Lényeges, hogy a parancs végénél „és” (&) karaktert írjunk, mert így a parancs végrehajtása a háttérben zajlik, valamint a program elindulása után további beszédfeldolgozó parancsokat tudunk majd kiadni. Ha eddig eljutottál, most létre kell hoznod a hangmintáidat, mégpedig legalább négyet. Ejtsd tisztán a parancsok nevét, tarts némi szünetet, majd vedd fel a hangmintát a listába! Amint mind a négy megvan, üsd le a B (az angol *back* szóból rövidítve) billentyűt a kilépéshez és mentsd lemezre a hangminta-készletet. Az állomány nevére semmilyen kötöttség nem vonatkozik, csak az a fontos, hogy ne felejtsük el a pontos helyét. Ezzel befejeztük a mikrofon beállítását, és egy olyan számítógépes paranccsal rendelkezünk, amelyhez szóbeli utasítás kapcsolódik. A Mozilla elindításához egyébként akár a *browser* (böngésző) szót is használhatjuk, de a programokat többnyire mégiscsak a nevük szerint szokásos és érdemes kiválasztani. Végül indítsuk el a cvoicecontrol programot!

```
cvoicecontrol speakermodel.cvc
```

Hangmintakészletem mentésekor a nevemet követően (ezt nem szükséges kitölteni) a .svc kiterjesztést adtam meg, és a beszédfelismerő programot a `cvoicecontrol chefmarcel.svc` paranccsal tudtam üzembe helyezni. Ezután csak annyit kell mondanom: *Mozilla* – és a böngésző program máris munkához lát. Ilyen parancsállományokat készítettem kedvenc szövegszerkesztőmhöz, a vi-hoz és természetesen néhány játékhoz is.
Pingvin, kérlek, tárd ki az étterem ajtaját! A mai menünkön szereplő eszközökkel elindulhatsz azon az úton, amelyen járva már ma a tiéd lehet a jövő számítógépe. Hiszen Linuxot használsz – szinte máris ott vagy! Kedves barátaim! Pingvin pajtás azt mondja, később jár és sajnálatunkra rövidesen itt a záróra. De túlságosan korán nincs értelme bezárni. François, nem töltenél még egy pohárral a vendégeinknek? Hogyne tudnám, François, hogy ember vagy és nem gép! Ne nézz már olyan sértődötten! Csak vicceltem. Barátaim, még egyszer köszönöm, hogy eljöttetek. Legközelebb is találkozzunk itt, Chez Marcelnél. Kedves egészségetekre!

Marcel Gagné (mggagne@salmar.com) Mississaugaban (Ontario, Kanada) él, a Salmar Consulting Inc. cég elnöke. A cég rendszerépítéssel és hálózati tanácsadással foglalkozik. Marcel pilóta és író egy személyben (tudományos-fantasztikus regényeket ír), társszerzője egy scifi-, fantázia- és horrorantológiának, a TransVersionsnak. Kedveli a Linuxot és a Unix minden változatát. Mostanában a Linux System Administration: A User's Guide című művén dolgozik. A világhálón elérhető honlapján sok hasznos dolgot találhatunk. ➲ http://www.salmar.com/marcel/

Kapcsolódó címek

Festival honlap a Világhálón
➲http://www.cstr.ed.ac.uk/projects/festival/
SpeechIO (a speechd honlapja a Világhálón)
➲http://www.SpeechIO.org/
A borok főhadiszállása ➲http://www.winehq.com/

A KDE felzárkózása

Erőteljesen javul a Linux irodai felhasználhatósága.

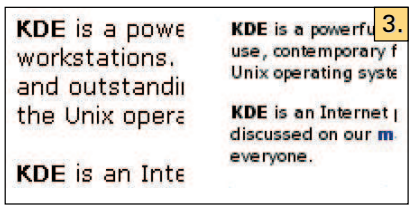
A K Desktop Environment nemrég piacra került 2.1.1-es változatával a KDE fejlesztői egy lépéssel közelebb kerülhetnek az irodai gépek piacának „meghódításához” (➲ http://www.kde.org/). A fejlesztés az 1.0-s sorozat óta gyorsult, új sajátosságokat és a megbízhatóság tekintetében javulást hozott a felhasználók számára. E cikk megírása óta már kiadták a KDE 2.2 első alfaváltozatát kipróbálásra. Mind a végfelhasználók, mind a fejlesztők részesülni fognak a legújabb rendszer áldásaiból. A KDE jelenleg mintegy 34 nyelvet támogat, és fejlesztői készek megválaszolni azokat a felmerülő kérdéseket, amelyek a Linux irodai felhasználásának életképességét firtatják.

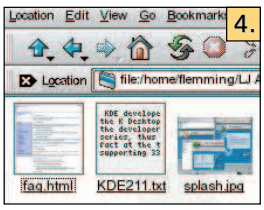
Az üzembiztonság további fejlesztésén kívül a legutóbbi változat nagyszámú „kozmetikai” fejlesztést is tartalmaz – ezek sokkal egységesebb és ragyogóbb kezelőfelületet teremtenek. A Kicker – amire a Kpanelt lecserélték, rengeteg új sajátosságra tett szert, valamint néhány régi is visszatért. Mindazon WindowMaker-felhasználóknak, akik a kedvenc kisalkalmazásaiktól való megválást nehezen viselték, nem kell tovább sóvárogniuk. A Kicker már képes a kisalkalmazásokat az új alkalmazásdokkoló oszlopba beágyazni (lásd *1. kép*). A gyermekpaneleket is támogatja, és a 2.0 változatból észrevehetően hiányzó külső tálca is visszatért, a témavezérlővel együtt. A témavezérlő visszakerülése ellenére az asztal különböző témaképes elemei közötti együttműködés még mindig hiányzik. A vezérlőelemek stílusát, ikonokat, színeket, háttereket és a Kwin-dekorációkat csak saját, önálló *Control Center*-ből lehet vezérelni. Ennek ellenére a KDE fejlesztőcsapatának munkáját nem szabad lebecsülni. Rengeteg új ikont adtak hozzá és fejlesztettek tovább, valamint az új indítóképernyők készen állnak, hogy egységesítsék az asztali alkalmazásokat (lásd *2. kép*).

A legutóbbi változatban a talán legkiemelkedőbb fejlesztés az élmosott betűtípusok támogatása. Az élmosott betűtípusok engedélyezéséhez a KDE-nek az XFree86 Xft-kiterjesztéssel kombinált Qt 2.3-ra kell épülnie. A Linux meglehetősen bosszantóan viselkedik a TrueType betűtípusok, valamint az élmosás tekintetében; és miközben ezek az előnyök nagy javulást eredményeznek, az élmosott betűtípusok beállítása meglehetősen nagy kihívást jelent. Az ehhez kapcsolódó útbaigazítások a ➲ http://trolls.troll.no/~lars/fonts/qt-fonts-HOWTO.html címen találhatók. Ha a KDE csomagokból lett telepítve, előfordulhat, hogy nem a megfelelő könyvtárakat használja. Ebben az esetben az újabb csomagokat ellenőrizni kell, mielőtt megpróbálnánk beállítani az élmosott betűtípusokat. A *3. képen* egy weblap látható (összehasonlítás céljából felnagyítva): így jeleníti meg a Konqueror elmosott betűtípusok támogatásával, valamint a Netscape elmosott betűtípusok támogatása nélkül.

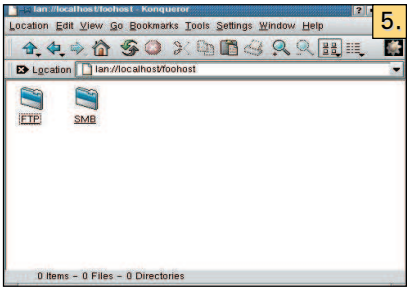
A projekt ékköve kétségkívül a Konqueror, a KDE következő nemzedékbeli webböngészője, fájlkezelője és dokumentummegjelenítője. A Konqueror moduláris szerkezete megengedi számunkra, hogy könnyen kiegészítsük olyan jelenlegi és jövőbeli internetes módszerekkel, mint a HTML 4.0, a Java, a Javascript, az XML, a Cascading Style Sheet (CSS-1 és CSS-2) és az SSL. A beépített elemeken kívül a Konqueror képes a Netscape készen beépülő moduljait felhasználni, Flash, RealAudio, illetve -Video és más multimédiás alkalmazások támogatása céljából. A Nautilusból átvéve a Konqueror a szöveges dokumentumok előnézetét is

1. kép Az új alkalmazásdokkoló oszlop
2. kép Az új indítóképernyők
3. kép Betűk élmosott betűkészlet-támogatással és anélkül
4. kép HTML-fájl előnézete
5. kép A LISa szolgáltatások keresése közben





meg tudja jeleníteni, oly módon, hogy az ikonon belül a szöveg első pár sorát mutatja meg. A kísérleti áttetszőség-támogatás (alfa-blending) segítségével képes a szöveg alatt félig átlátszó MIME-típusú ikonok ábrázolására. A szöveg és a kép előnézetén kívül a Konqueror HTML-dokumentumok előképének készítésére is alkalmas (lásd *4. kép*). A HTML-fájlok beolvasása és előnézetének megjelenítése a könyvtárszerkezet betöltődése után a háttérben zajlik, így a böngészés sokkal zökkenőmentesebbé válik. Azokat az eljárásokat, amelyek által a Konqueror inkább alkalmazáskeretté válik, semmint fájlkezelővé vagy webböngészővé, I/O szolgáknak hívják. Az I/O szolgák a KIO könyvtárra épülő apró kódok, amelyek tudják, hogyan küldjenek és fogadjanak adatokat egy adott protokollt használva. A KDE kezdetektől fogva a hálózat-áttetszőségre (network transparent) törekedett, és a KIO-szerkezet által lehetőség nyílik a különleges célú bővítmények (Plug-In) széles körű variációinak elkészítésére, amelyek kiszélesítik a Konqueror képességeit. Teljes körű támogatás létezik a Linux-fájlrendszer, a megosztott NFS és MS Windows, a weboldalak, valamint az FTP- és a LDAP-könyvtárak tallózására és böngészésére. A fejlesztők minden KDE-alkalmazáshoz csekély mennyiségű kóddal fejleszthetnek új protokollokat és csővezetékeket. Más I/O szolgákat digitális kamerák Gphoto2 programon keresztüli támogatása céljából fejlesztettek – ezek lehetőséget nyújtanak a kamerában tárolt képek „fogd és vidd” módszerrel való elérésére. Meglévő audiocd-modulokat továbbfejlesztve szintén a „fogd és vidd” módszerrel lehetséges a CD-lemezek rippelése és az MP3-ak, illetve az Ogg Vorbis kódolása (lásd még: Linuxvilág 2001. január, 34. oldal). Számos igen hasznos Konqueror I/O szolga létezik már minden különösebb csinnadratta nélkül. Például a Konqueror parancssorában kiadva a `man:/df"` vagy `"#df` utasítást, man-oldal jelenik meg a `df` parancsról (más parancsot is kiválaszthatunk). Ha a GNU Info dokumentációs böngészőben akarjuk ugyanezt megtenni, akkor próbálkozzunk az `info:/df` utasítással. A kismeghajtó gyors eléréséhez a `floppy:/` parancsot kell alkalmazni. A Konqueror kevéssé ismert kiegészítése az új `shell` parancsszolgáltatás. Eszerint miközben a könyvtárakban tallózol, a CONTROL+E billentyűt lenyomva olyan héj-parancsokat írhatsz be, mint a `du`. A parancs kimenete a pillanatnyi könyvtárban megnyíló párbeszédablakban lesz látható. A már telepített I/O szolgák listája a Control Center tájékoztató szakaszán keresztül érhető el. Nemrég mutatták be az egyik különleges I/O szolgát, amely LAN-böngészésre szolgál. Leginkább a Windowsban használt *Helyi hálózat*-hoz lehetne hasonlítani. Ennek az új szolgáltatásnak a motorja a LISa, a LAN Information server. A Windows Helyi hálózattól eltérően a LISa csak a TCP/IP-vermen alapul, más protokollokat, mint az SMB vagy a NetBIOS, nem használ. Összegezve: ha egyszer a helyi hálózatodnak megfelelően állítod be a LISát, megpróbálja megtalálni a hálózaton lévő eszközök általánosan elérhető szolgáltatásait (FTP, SMB, NFS és HTTP) (lásd az *5. képet*). Ha a Konquerorban kiadjuk a `lan:/` parancsot, az összes felderített kiszolgálót és azok szolgáltatásait kilistázza. Miközben úgy tűnhet, hogy a LISa túl nagy hálózati forgal-



5.

mat hoz létre feleslegesen, nem ez a helyzet. Valójában minél több LISát futtató ügyfél található a hálózaton, annál hatékonyabbá válik. A LISa tulajdonképpen egy démon, mely az ügyfélgépeken fut. Gépindításakor ez a démon üzenetszórt adatcsomagokat (broadcast) küld ki, amelyekkel megkísérli felderíteni a hálózaton létező LISa-kiszolgálókat. Abban az esetben, ha talál egyet, a hálózati kiszolgálók listája átmásolódik az új ügyfélre, további felesleges próbálkozások nélkül. A LISának rendszergazdai jogosultsággal kell futnia és a Control Center Networ/LAN modulján keresztül lehet beállítani – ez a kdenetwork csomag összetevője. A fejlesztők számára a KDE gazdag programfejlesztő eszközkészletet kínál, ezek között található a Desktop Communication Protocol (DCOP), a Component Object Model (az úgynevezett KParts), egy XML-alapú GUI-osztály és az előbb említett I/O könyvtárak (KIO). Ezeket a különböző fejlesztőeszközöket összefűzve alkották meg a KDevelop 1.4-et. A multimédiás összetevőket egy hálózat-áttetsző, analóg valós idejű szintetizátoron (aRts) alapuló szerkezeten keresztül kezeli.
A DCOP az a sokat említett ügyfél-ügyfél kapcsolattartó protokoll, amellyel a CORBA-t váltották ki a KDE 2.0 fejlesztésének korai szakaszában. A DCOP a szabványos X11 ICE könyvtárra épül, gyorsabb és könnyebben kezelhető felületet nyújt, mint az előző, CORBA-val fejlesztett változat. A KPartsszal az alkalmazások megoszthatják egymással elemeiket és beágyazhatják más alkalmazásokba. Ennek széles körű alkalmazása a Koffice-ban és a Konquerorban látható.
Az XML-t GUI-elemek dinamikus létrehozásának módszereként felhasználva a fejlesztők sokkalta testreszabhatóbb és szabványosítottabb kezelőfelületet hozhatnak létre. Az asztal állandóságának, szabványosságának elősegítésére KDE-szabványos kódok és GUI stílusútmutató létrehozásán dolgozik. Mióta a GUI-elemek dinamikusan készülnek, a stílusok frissítése azonnal visszatükröződik – anélkül, hogy módosítani vagy újrafordítani kellene őket. Az aRts ki tudja aknázni a CORBA-szerű hálózatokat úgy, hogy a távoli alkalmazások a hangot a helyi munkaállomáson szólaltassák meg. Az Xfree86 és a KIO által nyújtott hálózat-áttetszőségi tulajdonságnak köszönhetően multimédiás alkalmazások felhasználására nyílik lehetőség. A KDevelopról vagy a KDE-alkalmazásokról bővebb leírás a 76. oldalon szereplő KDevelop 1.4 áttekintésben található. A fejlesztőknek szánt bőséges anyag (beleértve az oktatói kézikönyveket, GYK-eket), valamint a szabványos útmutatók a ➲ http://developer.kde.org/ oldalon érhetők el. Ámulatba ejtő, milyen nagy lépésekkel halad a Linux az irodai operációs rendszerré válás útján. A felbukkanó új cégek, valamint a régebbi, szilárd alapon állók az irodai Linux fejlesztésére és egyre több alkalmazással való ellátására összpontosítanak. Nincs már távol az igazi elismerés! A Kompany (➲ http://www.thekompany.com/) megdöbbentő számú fontos Linux-alkalmazást gyűjtött össze. A KDE fejlesztése minden eddiginél nagyobb ütemben folyik és minden újabb kiadás a Linuxot az adatközponttól az irodai felhasználáshoz hozza közelebb.

Előnyök

- Konqueror (elég csak megemlíteni)
- Élmosostt (Anti-aliased) betűkészlet támogatása

Hátrányok

- A témaösszetevőket külön-külön kell vezérelni
- A rendszerindulás ideje a lassabb gépeken megnyúlik

➲ kde-user-request@lists.netcentral.net, (feliratkozás a felhasználók levelezőlistájára)

Robert Flemming
(flemming@valinux.com)
VA Linux System
rendszergazdája.

KDevelop 1.4

Utat a feltörekvőknek! – a KDevelop összetett fejlesztőkörnyezet programkészítésre, hibakövetésre és karbantartásra.

Egyetlen fejlesztőeszközben szenved hiányt a Nyílt Forráskód Közössége, egy profi IDE-ben (Integrated Development Environment – összetett fejlesztőkörnyezet). Szerencsére a KDevelop rendelkezésünkre bocsátott egy ilyen eszközt, amely egyesíti a már létező nyílt forráskódú termékeket és a hozzájárulók munkáit. Csakhogy vajon versenyre kelhet-e a KDevelop egy (többnyire valamilyen nem Unix-alapú) kereskedelmi IDE-vel?

Adatok

Gyártó: a nyílt forráskódot terjesztők
URL: ➲ http://www.kdevelop.org/
Ár: szabad (ingyenes) letöltés

Mi is az IDE?

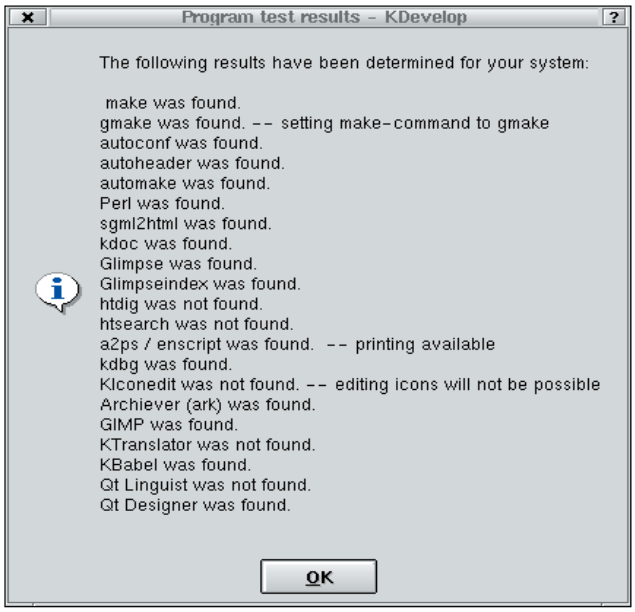
Az IDE olyan (lehetőleg grafikus) környezet, amely programkészítésre, hibakövetésre és karbantartásra szolgál. Ennek a környezetnek három alapösszetevője: a programozók szövegszerkesztője, amely szerkezetérzékeny a programozási nyelvre; a GUI (grafikus felhasználói felület)-szerkesztő, ezt az alkalmazás grafikus felületének tervezéséhez használhatjuk fel; és végül a hibakereső-nyomkövető program, ami a kódban rejlő hibákat deríti fel.
Ezek az IDE-vel szemben támasztott alapvető követelmények. Ahhoz azonban, hogy igazán előnyös eszközként használhassuk, további összetevőkkel kell kiegészíteni.

Telepítés

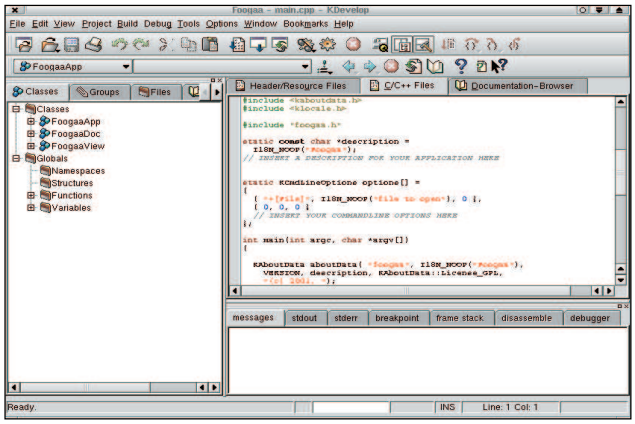
Mivel a nyílt forráskódú programok többnyire a feladat elvégzésére törekednek, és kevesebb figyelmet fordítanak arra, hogy felhasználóbarátok legyenek, a telepítés néha meglehetősen nehézkes és elkedvetlenítő lehet; különösen, ha figyelembe vesszük a számtalan Linux-változatot, valamint a folyamatosan változó könyvtárakat és eszközöket.
A KDevelop RPM binárisok letölthetők a KDevelop honlapról vagy felkutathatók valamilyen keresőlap segítségével, például a ➲ http://www.rpmfind.net/ címen.
Ehhez az ismertető megírásához teljesen új Linuxot telepítettem és megbizonyosodtam róla, hogy mindazt a csomagot és képességet tartalmazza, amit csak a terjesztés megenged. Még így is belefutottam azonban pár telepítési akadályba: néhány könyvtárnál olyan függőségekkel találkoztam, amelyek az én Linux-telepítésemen nem is léteztek. Egy kis internetes kitérő – a hiányzó könyvtárak letöltése – után viszont a gond hamar megszűnt.
A teljes telepítési idő (gyors Internet-eléréssel és némi szaktudással felvértezve) körülbelül félórát vett igénybe. Ez a telepítési módszer eszményi a kicsit is rendszergazdai vénával megáldott felhasználók számára. Néha a forrásból fordítás hasznos lehet a nem Linux/Unix operációs rendszert használó programozók vagy a leendő KDevelop-fejlesztők számára a testreszabott Linux-változatokhoz. De csak a tapasztalt vagy nagyon határozott fejlesztők próbálják meg forrásból lefordítani a KDevelopot!
Elsőként telepíteni kell az összes szükséges könyvtár fejlesztői változatát. Mivel nem áll a rendelkezésünkre egyszerű mód ezeknek a függőségeknek a megállapítására, a forrás fordítása valószínűleg próbálkozások és hibák sorozata lesz.
A KDevelop szolgáltatása, hogy számos létező nyílt forráskódú eszközt fel tud használni. Az összes ilyen eszköz megléte nem kötelező, de ha azt szeretnénk, hogy a KDevelop olyan hatékony legyen, mint ahogyan azt elvárjuk, mindenképpen ajánlott. Amikor a KDevelop első ízben indul el, megadja a vonatkozó eszközök listáját, jelenlévőként vagy hiányzóként bejelölve az egyes elemeket (lásd az *1. képet*). A lista nem végleges, ugyanis a hiányzó eszközök később még telepíthetők.
A KDevelop által felhasznált segédeszközök: g++2.7.2, g++2.8.1 vagy egcs 1.1 (én a g++ 2.9.2-t ajánlom); make; perl 5.004; autoconf 2.12; automake 1.2; flex 2.5.4; gettext; Qt 2.2.X (magába foglalja a Qt designert és a uic-t) és a KDE 2.X.
A választható eszközök: enscript, Ghostview vagy KGhostview, Glimpse 4.0, htdig, sgmltools 1.0, KDE-SDK (KDE software development kit), KTranslator, KDbg, KIconedit és Qt Linguist. Bár választhatók, a legjobb, ha ezek az eszközök is mind elérhetők.

Képességek

A KDevelop teljesíti az IDE három alapvető követelményét (tartalmaz szövegszerkesztőt, GUI-szerkesztőt és -hibakeresőt – lásd a *2. képet*, és más olyan képességekkel is fel van vértezve, amelyek hatékony és megbízható, akár üzleti feladatokra is alkalmazható eszközzé teszik.
Egy összetett program mind a kezdők, mind a profik számára rémisztő lehet, ezért nagyon fontos a jó programleírás. A KDevelophoz készült leírás kimelkedően jó, közvetlenül elérhető súgót szolgáltat – sajnálatunkra azonban a pillanatképek és egyéb grafikai elemek nem túl sűrűn fordulnak elő benne. A tartalomérzékeny súgó az eszközsorból és a *What's this?* kurzormódból egyaránt elérhető.
A KDevelop hivatkozik a KDE Lib és a Qt leírásaira is. Lehetőség nyílik továbbá könyvjelzők elhelyezésére, ami egyszerűvé teszi a visszatérést a tárgyhoz tartozó leíráshoz. A KDevelop honlapján további tanítófájlok és leírások is elérhetők.
A KDevelop beépített HTML-böngészővel rendelkezik, így a dokumentáció elérése gördülékeny és külső böngésző használata sem szükséges.
Az alapfelület elemei a következők:
Tree View, amely a class, a groups, a file, a books és a watch nézeteket tartalmazza;
Output View (Kimeneti nézet), ez a következő kimenetek üzeneteit jeleníti meg: stdout, stderr, debugger breakpoints, debugger frame stack, debugger disassembly és debugger messages;
Szövegszerkesztő és *Dokumentáció*, melybe beleértendő a fejléc- és erőforrás-szerkesztő, C/C++ fájlszerkesztő és a dokumentációböngésző;
Eszköztár azaz a főmenüparancsok ikonos megfelelője.

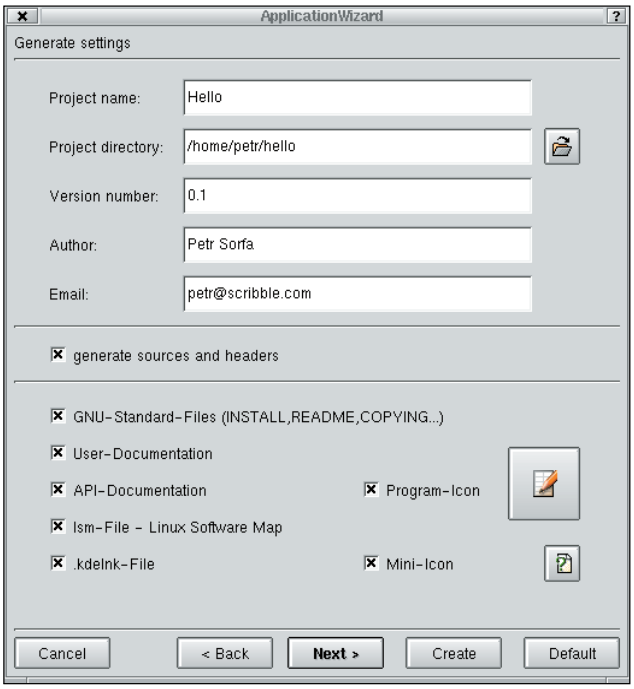


1. kép A KDevelop első indulásakor észleli a telepített elemeket

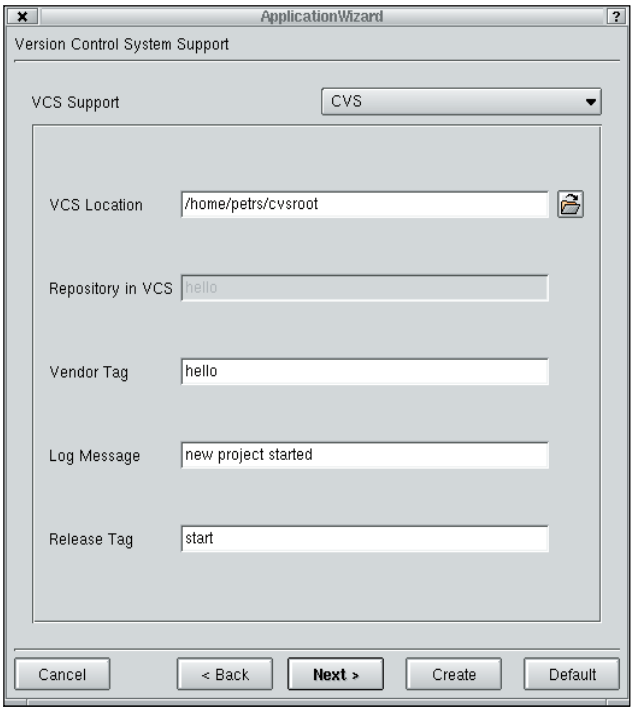


2. kép A KDevelop 1.4 működés közben

3. kép Az alkalmazásvarázsló



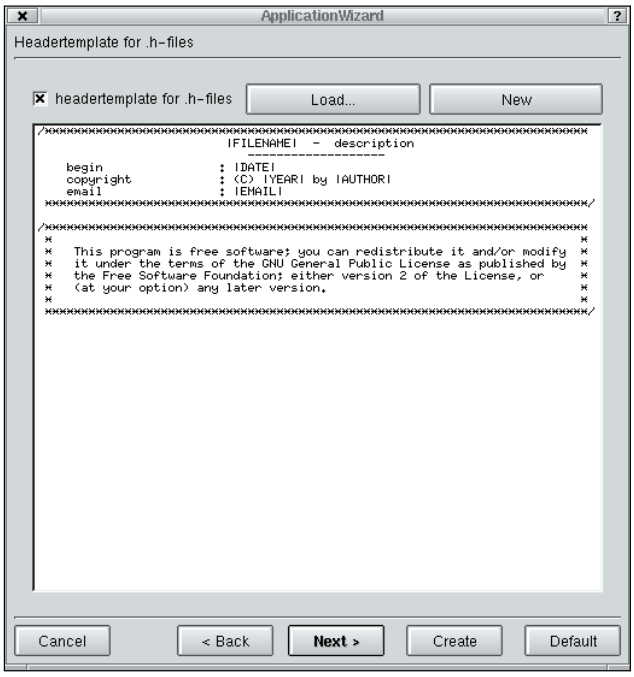
4. kép Általános projektbeállítások



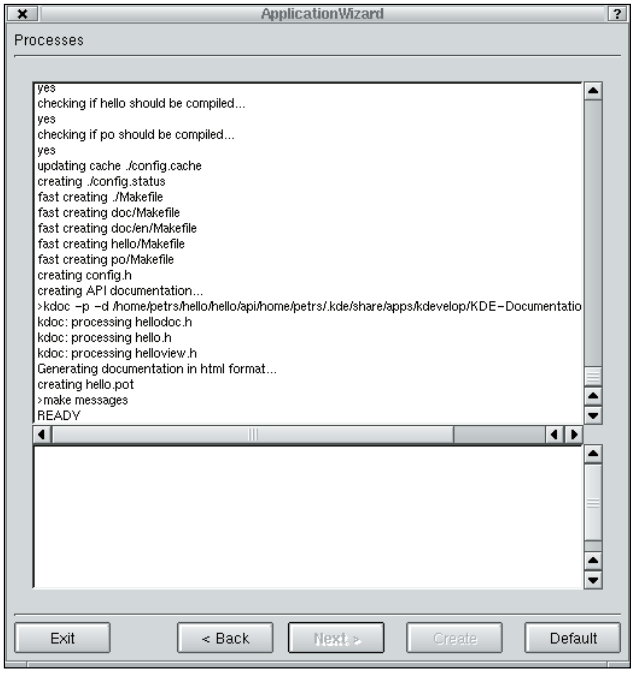
5. kép A változatkezelő rendszer kiválasztása

A KDevelop alatt a legegyszerűbb dolgok egyike a projektkészítés, melyet az *Application Wizard* segítségével, három lépcsőben tehetünk meg:

1. *Application Type* (lásd a *3. képet*)
   E lépés során a felhasználó különféle sablonok közül választhat: KDE 2 Mini; KDE 2 Normal; KDE 2 MDI GNOME (Normal); Qt (Normal, Qt 2.2 SDI, Qt 2.2 MDI, QextMDI); Terminal, például szöveges (C, C++) és egyéb (saját készítésű).

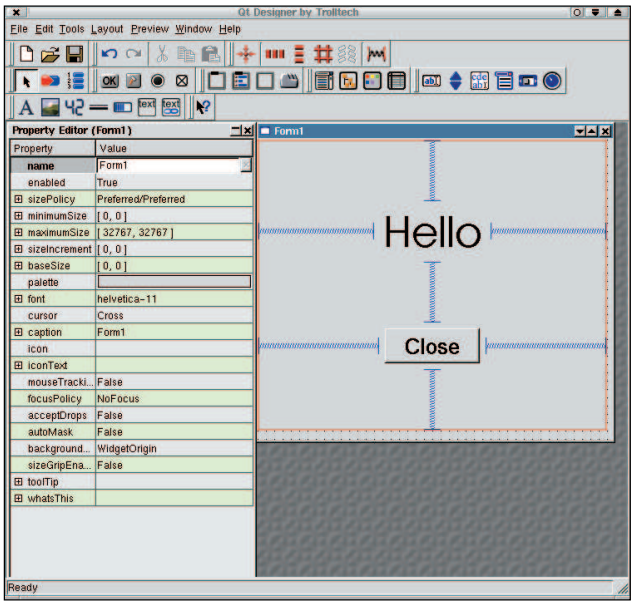


6. kép Fejlécsablonok

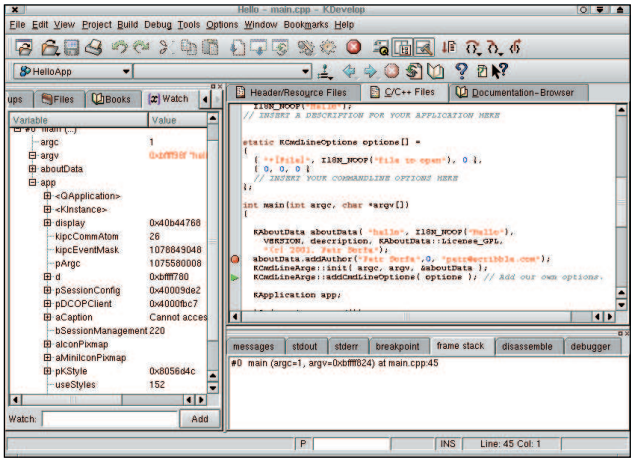


7. kép A projektkészítés kezdete

2. *Általános beállítások* (lásd a *4. képet*)
   Itt lehet megadni a projekt nevét, elérési útját, kezdeti változatszámát, alkotójának nevét és levélcímét. Lehetőség nyílik a projekthez kötődő fájlok létrehozására is, például forrásfájlok, fejlécek, GNU-alapfájlok, ikonok; továbbá a projekthez tartozó leírás szintén e helyütt készíthető el.
3. *Változatkezelő rendszer* (lásd az *5. képet*)
   A Változatkezelő rendszer beállításait módosíthatjuk. Ez a rész Linux-változatfüggő, de általában a CVS eszköz található itt.
4. *Fejléc és kódfájlokhoz használható fejlécsablonok* (lásd a *6. képet*)
   A fejlesztők itt dönthetik el, hogy milyen sablonokat használ a rendszer a program-fájlokhoz, illetve fejlécfájlokhoz. Ezek a fejlécek tagkibontásos módszerrel testreszabhatók, néhány adat (szerző, fájlnév, dátum) pedig önműködően kitöltésre kerül.
5. *Projektkészítés* (lásd a *7. képet*)
   A projektkészítés végső lépéseként, az `automake` és `configure` eszközök segítségével elkészülnek a megfelelő projektfájlok és könyvtárak. Figyelem! Ha a szükséges eszközök hiányoznak a Linux-változatból, a létrehozás folyamata megszakadhat. Amennyiben ilyesmi történik, a leghelyesebb feltelepíteni a hiányzó összetevőket, és újra felépíteni a projektet, ugyanis egy projektleállási hibát hihetetlenül nehéz helyreállítani.



8. kép A KDevelop 1.4 GUI-szerkesztője: a Trolltech Qt Designere



9. kép Hibakeresés a projektben

Amint a projekt létrejött, megkezdődhet a fejlesztés. Ezen a ponton nagyon hasznos, ha projektet megpróbáljuk felépíteni és végrehajtani a, hogy ezáltal felderíthessük az esetleges összeállítási hibákat.

Qt Designer

A KDevelop 1.4-es változata a Trolltech Qt Designerét használja. A Qt Designer profi felülettel rendelkezik, ahol a legtöbb Qt vizuális elem (widget) használata grafikus felületen keresztül oldható meg, és nagyon hasznos eszköz lehet a GUI-elemgyűjtemény és -összetevők

10. kép Lebegő hibakereső eszköztár

Előnyök

- Kitűnő leírás
- Nagyszerű támogatottság
- Az IDE legfontosabb szolgáltatásait támogatja
- Üzembiztos és használható valós többfejlesztős projektekben
- Testreszabható

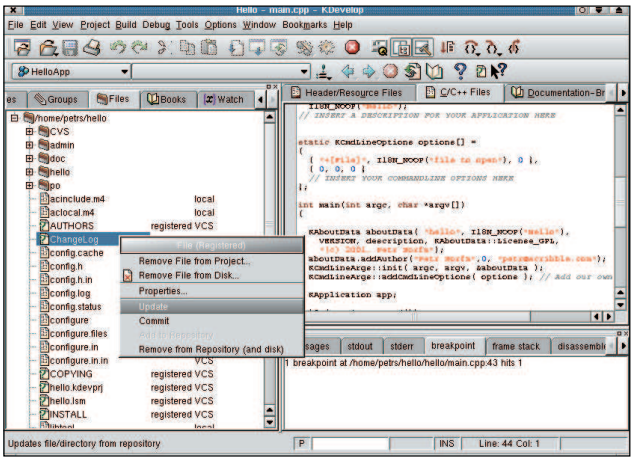
Hátrányok

- Vizuáliselem-támogatottság (widget support)
- A programozási nyelvek és fordítók támogatása
- Létező projektek importálása
- Nincs RAD-eszköz (egyelőre)

egymás közötti kapcsolatainak kialakításában (akár vizuális programozásnak is nevezhetnénk).
Foglaljuk össze röviden a KDevelop alatti Qt Designer segítségével végzett grafikus felület készítés lépéseit (lásd a *8. képet*)!
A Qt Designer komolyabb termék, képességeinek és használatának teljes leírása önmagában is megtöltene egy cikket, ezért most csak a KDeveloppal való kapcsolatára térünk ki.
A Qt Designer lehetővé teszi a terveszközök (layout tools) használatát és valamennyi vizuáliselem-tulajdonsághoz hozzáfér.
Képes az egyes vizuális elemek közötti függőségi viszonyok kiépítésére is, például: egy gomb megnyomásakor záródjon be az adott ablak.
A Qt Designer csak egy a párbeszédablakot leíró köztes XML .ui fájlt készít. A tényleges forráskódot egy másik eszköz, az uic készíti el a .ui fájlból. A KDevelop 1.4 támogatja ugyan az .ui fájlokat, de azokat a felhasználónak kell a projekthez adnia. Amikor a felhasználó `make`-et vagy `rebuild`-et indít, a KDevelop a szükséges forráskód elkészítéséhez önműködően meghívja az uic-t.
Sajnálatos módon az uic az összes elkészült forrásfájlt újraírja, valahányszor a felhasználó a Qt Designerben az .ui állományt megváltoztatja. Emiatt a felhasználó ezeket a forrásfájlokat tulajdonképpen nem szerkesztheti. Az uic eszköz által készített kód kihasználásához a létrehozott kódosztályokat örökölni kell, mielőtt saját képességeket vihetnénk a programba.

Hibakeresés

Hibakereső szolgáltatások nyújtásához a KDevelop a GDB-t használja fel (lásd a *9. képet*). A szerkesztőablak bal oldali oszlopára kattintva töréspontokat állíthatunk be az elkészült kódban. A töréspontok akkor is beállíthatók, ha a program nem fut vagy lefordíthatatlan állapotban van. Ezek az úgynevezett kötetlen (lazy) töréspontok.
A KDevelop a kötetlen töréspontokat kékkel, míg a működőket vörös színnel jelöli.
A megfelelő forrássor utáni kis zöld nyíl a végrehajtás pillanatnyi pontját mutatja.
A KDevelop a legtöbb alapvető hibakereső szolgáltatást támogatja. Képes az egyszerű végrehajtásra, a következő sorra ugrásra és a programmegszakításra. A felhasználónak lehetősége nyílik a hibakeresőt lebegő eszköztár formájában előhívni, ami megkönnyíti a hibakereső parancsok elérését (lásd a *10. képet*). Fanézetben a változófa a jelenleg elérhető változókat mutatja be.
A hibakeresőhöz tartozó adatok a hibakereső (debugger), assembly, verem (frame stack) és töréspont kimeneti ablakokban jelennek meg.
Az alapértelmezett hibakereső hátránya, hogy hiába lenne szükség esetleg finomhangolt hibakeresésre, a GDB-t nem lehet közvetlenül



11. kép A forrásvezérlő rendszer használata

elérni. A másik gond, hogy a felhasználó a változók értékeit a *Watch input* sorban nem magától értetődő módon változtathatja meg.
A KDevelop úgy is beállítható, hogy külső hibakeresőt használjon, legyen az a népszerű DDD, kdbg vagy xxgdb.

Külső alkalmazások

A KDevelop KDE-alkalmazások futtatását teszi lehetővé saját keretrendszerén belül. Néhány alkalmazás, például a Gimp, az Ark és a KBabel alapértelmezés szerint be is van állítva, további elemeket pedig az *Options tools* menü alatt adhatunk a rendszerhez.

Fordítás, felépítés és terjesztés

A projekt fordítása és felépítése több menüszolgáltatáson keresztül is elérhető. Ezek a *Make*, *Clean*, *Rebuild*, *Clean for distribution* és *Auto configuration* menüpontok. A KDevelop, ha szükséges, még a futtatás előtt rákérdez, hogy újra akarjuk-e fordítani a programot.

Project Make Distribution (Terjesztéskészítő)

A *Source.tgz* menüpont forrásváltozatok készítését teszi lehetővé. Úgy tűnik, a hasznosabb csomagolást nyújtó RPM vagy RPM spec fájlok önműködő készítésére sajnos nincs lehetőség.

Configuration Management (Beállításkezelő)
Source Control (Forrásvezérlő)

Ha projektkészítéskor kiválasztottuk a változatkezelő rendszert, a kijelölt fájlokat a forrásvezérlő (source control) rendszerhez adhatjuk (lásd a *11. képet*). Ezt úgy tehetjük meg, hogy *Group* vagy *File* nézetben a fájl *Add to Repository* helyi menüt (Pop Out) választjuk. A változásokat a *Commit* pont kiválasztásával érvényesíthetjük, más fejlesztők változtatásait pedig az *Update* pont választásával hozhatjuk vissza.
Mivel a CVS távoli tárlóhelyeket is támogat, a KDevelop alatt egyszerre több fejlesztői projektünk is lehet. A KDevelop azonban nem támogatja a CVS által nyújtott valamennyi szolgáltatást, többek közt a fájlmegtekintési és -szerkesztési jogokat sem.

Felhasználói leírás

A KDevelop a kdoc és a doxygen rendszeren keresztül program API-leírások készítésére is képes. Ezalatt a felhasználó folyamatosan láthatja a felhasználói API-leírást. Különösen a nagy, többfejlesztős projektek esetén igen hasznos ez a szolgáltatás.
Ha projektkészítéskor a felhasználói leírást kiválasztottuk, akkor a felhasználói kézikönyv HTML-űrlapja önműködően elkészül. Innentől már csak a felhasználótól függ, kitölti-e az adatokat valamilyen HTML-szerkesztővel.

Támogatottság

A nyílt forráskódú projektek egyik leggyakoribb hátránya az alacsony támogatottság. Néha a projekt meg is szakad – így tulajdonképpen lehetetlenné válik a kapcsolatteremtés, ha gondok merülnek fel vagy hibákat találunk. A KDevelop azonban szerencsére pezsgő levelezőlistával rendelkezik, melyet néhány fejlesztője folyamatosan felügyel. A közvetlen leírások széles választékával a KDevelop olyan szintű támogatottságot tudhat magáénak, amellyel még a legtöbb kereskedelmi termék sem versenyezhet.

Hiányzó kívánságok listája

Annak ellenére, hogy a KDevelop hatékony és hasznos eszköz, néhány felhasználási terület még hiányos vagy fejlesztésre szorul:

- Kívánatos volna egy okos szövegszerkesztő, amely önműködően befejezné a kódot, például kitöltené az adott függvény értékeit.
- A KDevelop 1.4 nyelv támogatása mindössze a C++ és C alkalmazásokra, valamint gcc/g++ fordítóra korlátozódik.
- Más GUI-szerkesztők (például a Gnome GUI-szerkesztő, a Glade) támogatottsága lehetne jobb.
- Egy létező projekt KDevelopba illesztése nem éppen egyszerű feladat.
- Nincsenek Rapid Application Development (RAD) -elemek, amelyek adatbázis-kapcsolatot szolgáltathatnának, és alapját képezhetnék a vállalati szintű fejlesztéseknek.

Mivel a KDevelop nyílt forráskódú program, ezek a hiányzó lehetőségek hamarosan talán nem jelentenek gondot.

Kapcsolódó címek

KDevelop honlap ➲ http://www.kdevelop.org/
Linux RPM-csomagok jegyzéke ➲ http://www.rpmfind.net/
Nyílt forráskódú programok tájékoztató jegyzéke ➲ http://www.freshmeat.net/
A Qt, Qt Designer és a Qt Linguist készítői ➲ http://www.trolltech.com/
KDE honlap ➲ http://www.kde.org/

Összefoglalás

A KDevelop egy középszintű kereskedelmi IDE képességeivel rendelkezik. Igen jól illeszkedik a linuxos felülethez, számos nyílt forráskódú eszközt kihasznál, és olyan szintű támogatottsággal bír, amit nehéz túlszárnyalni. Ennek ellenére van még hely a további fejlesztések számára. A KDevelop képességei megfelelően elégítik ki a kis- vagy közepes méretű projektek és fejlesztőcsapatok fejlesztőkörnyezet-igényeit.

Petr Sorfa
(petrs@sco.com) a Santa Cruz Operation's Development Systems Group tagja, ahol cscope és a Sar3D nyílt forráskódú projektek felelőseként tevékenykedik. Főiskolai végzettséget szerzett a Cape Town és a Rhodes University-n. Érdeklik a nyílt forráskódú fejlesztések, a számítógépes grafika, a fejlesztőrendszerek, továbbá a képregény-rajzolás (sequential art).