Közösség kovácsolódik végre idehaza is?

Gondolatok a hazai közösség kialakulásáról.

Névvel ezelőtt lelkesen figyeltük a külföldi eseményeket, és közben szomorúan sopánkodtunk: „Haj, mikor lesznek ilyen megmozdulások Magyarországon is…!”. Valahogy az volt az érzésem, hogy a linuxos szakembereknek mindig csillog a szemük, ha a Nyugat nagy történéseiről beszélnek, de mihelyt szóba jött kis hazánk, ezt a csillogást felváltotta valami tompa beletörődés. Sokszor rákérdeztem, és mindig ehhez hasonló válaszokat kaptam: „Á, a magyar helyzet miatt nem lehet ilyen nagy lépéseket tenni”, vagy „Láttál te már olyat, hogy magyarok valamit ingyen csináljanak?”. Azóta szerencsére sokat változtunk. Megértettük, hogy nagyon kicsi pontok vagyunk ugyan az óriáscégekhez képest, de együtt és hosszú távon bizony igen komoly eredményekre vagyunk képesek. A nemzetközi fejlesztésekbe rendkívül gyorsan bekapcsolódtunk, számos olyan szabadon használható programcsomag van, amelyben több magyar szakember is ellenszolgáltatás nélkül segített, sőt, hazai fejlesztések is kibújtak a földből. Hogy tökéletes volna minden, azt nem állítom, mivel ez még a legnagyobb jóindulattal is pimasz huncutságnak nevezhető. De nagyon jó irányba tartunk! Hiszen teljes és működőképes munkafelületek állnak rendelkezésre, vagy ott van például a legutóbbi vízválasztó megmozdulás, az OpenOffice.org programcsomag magyarítása (erről részletesen lapunk 80. oldalán olvashattok).

Mégis nézzük a dolog árnyékos oldalát! Aki ismer, tudja, hogy mindig a gyenge pontokat keresem először. A hazai helyzettel kapcsolatban például a következőkhöz hasonló kérdések motoszkálnak a fejemben: Hol voltak eddig a mostanában oly sokat tevő emberek? Ezelőtt is voltak jó kezdeményezések, azok miért nem élték túl, miért haltak hamvukba? Hol van az egykor lelkes csapat hiányzó fele? A külföldi eredményességhez képest idehaza miért tudunk ilyen döbbenetesen keveset felmutatni?

Ne értsetek félre, nem az a célom, hogy most mindenkiről elmondhassam, mennyire rosszul dolgozott, áldozta fel a szabadidejét vagy éppen egyéb lehetőségeit. De igenis szükség van az összehasonlításra, az elemzésre. Ráadásul az elemzés most időszerű, hiszen a linuxos világ legismertebb egyesülete, az LME éppen a lap megjelenésekor tartja rendes éves közgyűlését, ahol évet zárnak, illetve új évet alapoznak meg tisztségviselők választásával, a fő irányvonalak elfogadásával. És mint a legismertebb szervezet nagyon sokat tehet a közösségért. Egy lépéssel továbbmegyek: nagyon sokat is kell tennie.

Nem tudom, nem tudhatom, hogy az új vezetőség milyen irányvonalat követ, néhány kérdés viszont annyira fontos, hogy nem csak a vezetőségen múlik. Milyen kérdésekre gondolok? A következőkre: folytasson-e az LME pénzügyi tevékenységet? Vállalhat-e véleményt szakmai felelősséggel nem közvetlenül egyesületen belüli ügyekben? Kell-e Budapesten kívül is tevékenykednie? Lehet-e, szabad-e változtatni a szervezet felépítésén?

Sok vita folyt az elmúlt évben, hogy folytathat-e az LME pénzügyi tevékenységet. Gondolok itt reklámtevékenységre, szakmai szolgáltatásokra, akár a bevétel érdekében szervezett rendezvényekre, oktatásra vagy éppen kereskedelemre. Gyakran hallottam, hogy az Egyesület nem azért van, hogy pénzt gyártson, hogy ahol pénz van, ott előbb-utóbb korrupció is van, hogy kereskedelmi tevékenység nem fér össze a szabad programok elvével.

Amit viszont figyelembe kell venni, az, hogy az Egyesület céljainak megvalósításához anyagi alap kell. Vitathatatlanul nagyobb hatékonysággal működne az Egyesület, ha komoly háttérrel (mind embererővel, mind anyagi és tárgyi javakkal) rendelkezne. Pénzügyi keret nélkül egy „kisebb” rendezvény megszervezése is (mint amilyen például a CEU-ban tartott tanácskozás volt) döbbenetes munka, nem is beszélve arról, hogy az Egyesület, ha nem rendelkezik biztos háttérrel, bizony pillanatok alatt eljuthat oda, hogy az elveit néhány százezer forintért „eladja”. Ez pedig semmiféleképpen nem engedhető meg. Fontos azonban, hogy ne essünk át a ló túloldalára, és az elveket kövessük, ne a pénztárcánkat. Ez a legfontosabb elvárás az új vezetőséggel szemben.

Szintén sokszor hallottam, hogy az LME nincs olyan helyzetben, hogy valamiről szakmai véleményt adjon. Nem rendelkezik megfelelő szakemberháttérrel, nem birtokol szükséges gépeket és jogokat, azok nélkül pedig „minek pattogjon a bolha”. Bizonyára emlékeztek, több olyan eset volt az elmúlt egy évben is, amikor egy erős lobbi, egy erős érdekképviselet igenis ki kellett volna álljon megvédendő a közösség érdekeit és érdemeit. Úgy gondolom, egészen addig, amíg az Egyesület „saját magát” (tagjait, szervezeti egységeit, feladatait) nem tekinti komolynak, addig hiába várjuk el ezt a világtól is.

Még egy záró gondolat. Sokan mondják, hogy „odakint könnyű”. Állítom, hogy idehaza sem megoldhatatlan a dolog, csak nem pont ugyanúgy. De a legfontosabb, hogy lássuk és érezzük, hogy egy olyan közösségről van szó, amelyik tényleg összetart – nem azért, mert fizetik, nem azért, mert rövid távon személyes haszna van belőle, hanem mert érti, hogy hosszú távon mindenkinek (és természetesen nekünk is) haszna származik belőle. Ha értjük, hogy a munkánk nem értelmetlen, akkor hegyeket tudunk megmozgatni. Szerintem Shakespeare egy magyarra gondolt, amikor ezt mondatta a lelkes Zubollyal: „Ide nekem az oroszlánt is!”

Utóirat: A cikk írása közben a téma kapcsán még számos gondolat felvetődött bennem. A cikk folytatását a Linuxvilág honlapján a ➲ http://www.linuxvilag.hu/szy címen olvashatjátok.

Szy György

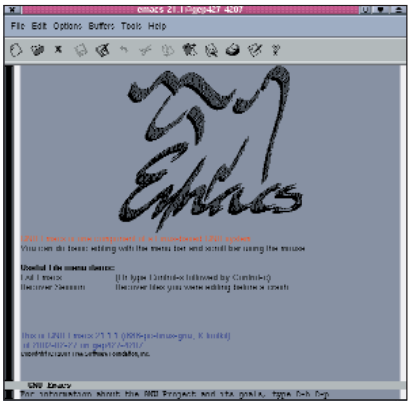
Programvadászat

AbiWord-0.99.2

Mellékelt korongunk megszokott vendége az AbiWord, mely ezúttal már a 0.99.2-es változat. A fejlesztők tervei szerint ez az utolsó, főként hibajavításokat tartalmazó változat az üzembiztos 1.0-s megjelenése előtt. Az előre elkészített rpm-csomagokból telepíthető, korongunk továbbá a forráscsomagot is tartalmazza, amelynek segítségével bárki testreszabottan lefordíthatja a saját rendszerére.

Emacs

A program eredetileg szerkesztőprogram, de ha szeretnénk, a megfelelő bővítményekkel akár hírcsoportolvasó, levelezőprogram, FTP-ügyfél és webböngésző is lehet belőle. A 2.1-es változat újdonságai közé tartozik a beépített MIME-támogatás, a képmegjelenítési lehetőség és a wav-fájlok lejátszása. A kezelőfelület is megváltozott, ezentúl



eszköztárakkal és gyorstippekkel is találkozhatunk benne. A betűkezelése szintén javult, már nem csak az állandó szélességű betűket támogatja. Forráskódból telepíthető, amely a CD-mellékleten is megtalálható.

Böngészők

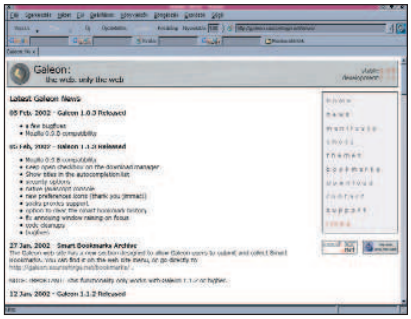
Több közel azonos erőforrásigényű, de különböző tudású böngésző is felkerült a korongunkra, így mindenki találhat kedvére való. Fontos megemlítenem, hogy ezek a programok még nem üzembiztosak. Kipróbálásukat mindenképpen csak a kísérletező kedvű felhasználóknak ajánlom! Mindegyik a Netscape forráskódja alapján készült, és a Mozilla Gekko-motorját használja.

Mozilla 0.9.9

CD-mellékletünkre felkerült a Mozilla 0.9.9 Milestone. Az új kiadásban csiszoltak az Address Book használhatóságán, és a nyomtatási kép beállítási lehetőségét is módosították. A fájlletöltés hibáját szintén kijavították, az előzőkben ugyanis a mentendő fájl nevének az eredeti név helyett a gyorstárban tárolt szövegrészt ajánlotta fel. A 0.9.7 változattal ellentétben ismét adott a lehetőség a dinamikus témaváltásra.

Gaelon 1.0.3

A böngészők sorából fejlesztése jelenlegi szakaszában a Gaelon 1.0.3 a legígéretesebb, amelyet rengeteg kényelmi szolgáltatással láttak el. A ***Bookmarkletek*** segítségével beállíthatjuk az oldalak



önműködő görgetését és ennek sebességét, a HTML-oldalak színén is változtathatuk, a lapot kétszerezhetjük, valamint a weboldalon belül könyvjelzőket helyezhetünk el. A könyvjelzőkezelés újdonsága a parancsra történő, tetszőleges számú oldalakhoz tartozó ***favicon.ico*** letöltése. Kedvencünket a CD-mellékletünkön szereplő rpm- és tar.gz-csomagok segítségével telepíthetjük, és a ***Themes*** könyvtárban található eszköztártémákkal új felülettel láthatjuk el.

Beonex 0.7

A Beonex 0.7 is Mozilla-alapú böngésző. Tartalmaz egy webböngészőt, egy levél- és hírcsoportolvasót és egy weboldalszerkesztőt. Kezeli a HTML 4-, a DOM level1-, a CSS1-, az XML- és a JavaScript 1.5-ös szabványokat. Ezenkívül a levelezőprogram a POP3- és az IMAP-típusú kiszolgálókat is képes elérni. Előre csomagolt tar.bz2 csomagok segítségével telepíthető.

A SkipStone 0.79

SkipStone 0.79 egy újabb Mozilla-alapú böngésző, mely próbaállapota ellenére meglepően üzembiztos. Mind a felülete, mind a kezelése egyszerű. Működésének alapfeltétele a Mozilla jelenléte.

Megjelent az Opera 6.0 első próbaváltozata Linuxra

Ennek a rendkívül népszerű böngészőnek az 5-ös, linuxos változatát eddig egymillió alkalommal töltötték le. Az Opera tovább gyorsult, a sebesség növelése érdekében a tudását több, a Linux követelményeinek megfelelő tulajdonsággal bővítették. A program bétaállapota ellenére meglepően megbízható. A bővítmények támogatása sokat javult, így a legtöbb Netscape-bővítmény is működik, például a Macromedia Flash, a Plugger, a Real Player, az Acrobat Reader, a TCL 2.0, a Codeweaver Crossover (Apple Quicktime) és a Java. Ez a változat már támogatja a unicode-os ázsiai és kelet-európai karaktereket is.
Megújult a ***Personal Bar***, melyben könyvjelzőinket ezentúl nemcsak szerkeszthetjük, hanem kereshetünk is közöttük.
Másik újdonsága a ***Hotclick*** szolgáltatás, amelynek segítségével – ha egy szövegrészre duplán kattintunk – bármely szó rögtön kijelölhető, három kattintás esetén a mondat, négynél pedig az egész bekezdés. A kijelölt szövegrészlettel egy időben felbukkanó úszómenüből a ***Szöveg máso-***

lása, a *Szöveg keresése* kereső (például Google, AltaVista), illetve a *Fordítás* közül választhatunk.
A programot a CD-mellékleten szereplő előre csomagolt rpm, deb és tar.bz2 csomagok segítségével telepíthetjük.

PHP4.1.2

A PHP önállóan is használható programozási nyelv, amely nagyméretű adatbázisok működtetésére is képes. A hivatalosan Hypertext Preprocessor névre keresztelt nyelv a kiszolgálóoldali programozás nyelve lett. A kiszolgáló és az ügyfél közti parancsok többnyire a HTML-kódba illesztve helyben használják, a kiszolgálón pedig a PHP-értelmező segítségével kerülnek feldolgozásra – eközben a HTML-elemek változatlanok maradnak.
A PHP teljesen felületfüggetlen, azaz Linuxon, Unix-rendszereken, Windows operációs rendszeren és Macintosh-gépeken egyaránt fut. Kiemelkedő tulajdonsága, hogy a legtöbb adatbázis a PHP-hez közvetlenül csatlakoztatható. Számos Linux-változatban megtalálható például rpm formátumban. Ilyenkor nincs más teendőnk, mint a megfelelő csomagok telepítése akár terjesztésünk saját csomagkezelőjével.
Ha más feladatok ellátására is szükségünk van, például kiszolgálóként szeretnénk működtetni vagy más adatbázist kívánunk támogatni, akkor a PHP-t, illetve a kiszolgálót magunknak kell fordítani.
A CD-mellékleten található ***php-4.1.1.tar.gz*** csomagot a
`tar -xvzf php-4.1.1.tar.gz`
paranccsal csomagoljuk ki a megfelelő könyvtárba.
Ezután a következő utasítás segítségével lépjünk be a ***php*** könyvtárba:
`cd ./php-4.1.1`
Ebben a könyvtárban található a configure program, melynek a megfelelő kapcsolókkal megadhatjuk, hogy a PHP-ba milyen szolgáltatások épüljenek be.
A `./configure --help` paranccsal a megfelelő kapcsolókat magunk is lekérhetjük, vagy a jobb áttekinthetőség kedvéért a `./configure --help > opciok.txt` segítségével fájlba irányíthatjuk át.

Configure kapcsolók

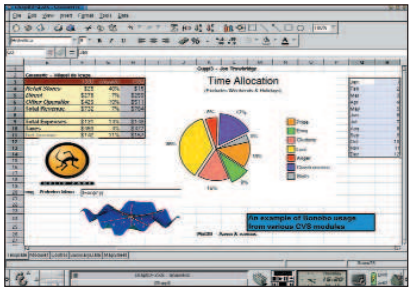
A legfontosabbak a következők:
adatbázisokhoz

- `--with-adabas` az Adabas-támogatás beépítésére szolgál
- `--enable-filepro` a FilePro-támogatás beépítésére szolgál, ez azonban csak olvasásra képes
- `--with-ldap` az LDAP-támogatás
- `--with-msql` a MySQL-támogatás
- `--with-oracle` az Oracle-támogatás
- `--with-pgsql` a PostgreSQL-támogatás beépítésére szolgál.

Ha az adatbázisrendszer nem az alapbeállítású könyvtárban található, a pontos elérési út megadható:
`-- with -adabas=/elØrØsi/œt/`
Amennyiben a `configure` rendben lefutott, a `make` programot azonnal indíthatjuk. Ehhez azonban rendszerünknek tartalmaznia kell egy C fordítót. Legvégül a `make install` parancsot kell kiadnunk, ennek végeztével a telepítést befejeztük.

Gnumeric 1.0.4

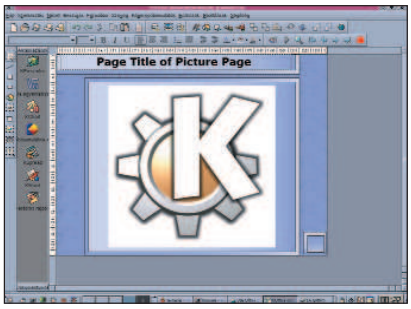
A Gnumeric könnyen használható, nem túl nagy méretű táblázatkezelő program. Nem igényel erőművet, a korosabb gépeken is jó sebességgel futtatható.



Az alkalmazás a Gnome Projekt része. Az Excel, Applix, Sylk, XBase és az Oleo állományokat tudja beolvasni. Érdekessége, hogy az Excelben megismert függvények mellé külső függvényeket mi magunk is megadhatunk a Gimpben megismert bővítményes rendszer segítségével, három különböző nyelven: Pythonban, Guile-ben és Perlben. Az Excel, az XML, a HTML, a LaTeX, a Troff, a PDF, az EPS, a DVI és a CSV fájlformátumaiban képes menteni. Mellékletünkön előre csomagolt rpm és tar.gz formában található meg.

KOffice 1.1.1

A KOffice összetett, minden igényt kielégítő irodai csomag szövegszerkesztővel, egyenletszerkesztővel, bemutatókészítővel, táblázatkezelővel, folyamatábra-rajzolóval, valamint vektoros rajzolóprogrammal. A Linux-rendszereken futó irodai csomag minden összetevője nyílt forrású és már 29 nyelvre lefordították.
A menük teljes mértékben magyarítottak, tartalmazza az ispell helyesírás-ellenőrző rendszert, de a magyar nyelvű ispell-csomagokat nekünk kell telepítenünk. Korongunkra a magyar nyelvű ispell helyesírás-ellenőrző rendszerrel együtt került fel rpm és tar.gz formában.



SuSE 7.3 frissítés

A 31. CD Frissítések könyvtárában található a SuSE 7.3 kiadásához eddig megjelent összes frissítés első része. A biztonsági hibajavításokat és a programok új változatra történő frissítését is tartalmazza.

Rendszermag

Korongunkon ezúttal a rendszermagok több sorozatának tagjait adjuk közre.
A 2.0 sorozat legújabb tagja a 2.0.39-es változata. A 2.2.-es üzembiztos sorozat fejlesztésének és karbantartásának eredményeként jelent meg a 2.2.20-as rendszermag. Kis hibával megjelent a 2.4 sorozat legújabb tagja, a 2.4.18-as. A hiba az x86-os kiépítést használókat nem érinti, a más rendszeren dolgozóknak ajánlott az utólagos hibajavító csomag telepítése. A 2.5-ös fejlesztői sorozat immár a 2.5.5-ös változatnál tart. Használatát csak a hozzáértőknek ajánlom.

MPlayer 0.6.0

A program Linuxon működő magyar fejlesztésű videolejátszó, amely sok más Unixon és akár nem x86-os processzorokon is fut. Használható parancssorból és grafikus felülettel is. A legtöbb ismert fájltípus lejátszására képes. Az MPEG, a VOB, az AVI, a VIVO, az ASF/WMV, a QT/MOV, a FLI, a NuppelVideo, a yuv4mpeg, a FILM, a RoQ, és némely RealMedia-fájlt egyaránt ismeri. Futtatásához nem árt, ha rendelkezünk 2.4-es rendszermaggal és a legfrissebb X-felülettel. CD-mellékletünkön a program forrás formájában található meg, így mindenkinek magának kell fordítania. Még a telepítés előtt ajánlatos elolvasni a leírást, amely a sikeres fordításhoz szükséges tudnivalókat tartalmazza.

Dankaházi István
(danka@linuxvilag.hu) A Linuxvilág munkatársa. Szabadidejében szívesen úszik és kerékpározik.

Ultraszélessávú hálózatok

Hamarosan elérkezhet az az idő, amikor a jelenlegieknél sokkal nagyobb átviteli sebességet biztosító vezeték nélküli hálózatokat használhatunk. Az eredetileg katonai célokra fejlesztett UWB (Ultra Wide Band) ultraszélessávú megoldás már csak arra vár, hogy az amerikai Szövetségi Távközlési Bizottság áldását adja a polgári alkalmazásokra.

Az UWB érdekessége, hogy – mint neve is mutatja – rendkívül széles, a néhány Hz-től a GHz-ig terjedő frekvenciatartományban üzemel. Radaroknál és más katonai alkalmazásoknál már elterjedt, azonban két rendkívül előnyös jellemzőjének köszönhetően a polgári életben is ígéretes jövő előtt áll.

Az UWB egyrészt rendkívül kicsi, az elektronikus készülékek által kibocsátott zajhoz hasonlóan kis energiájú jelekkel dolgozik. Ennek köszönhetően működtetéséhez is csekély energiára van szükség, sokkal takarékosabb, mint a Bluetooth vagy az IEEE 802.11b hálózatok. Másrészt a széles frekvenciatartománynak hála az UWB a több száz Mb/s sebességet is elérheti.

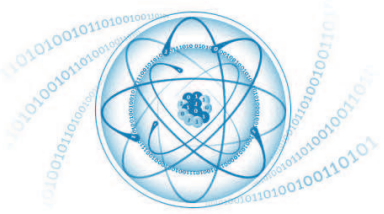
Amennyiben a bizottság kedvezően dönt, valószínűleg több gyártó – például az Intel – azonnal megkezdi a fejlesztéseket, hogy minél hamarabb piacra dobhassa új áramköreit és megoldásait.

➲ http://www.uwb.org

Digitális azonosítót mindenbe!

A Sun Microsystems is csatlakozott ahhoz a kezdeményezéshez, amelynek nyomán minden termékbe apró digitális azonosítót (Auto-ID) építenének. Az elképzelések szerint minden egyes árucikkre egyedi azonosító kerülne, amelyet rádiófrekvenciás jellel lehetne leolvasni. A vásárlás ezzel lényegesen egyszerűbb lenne, hiszen elég volna telepakolni a kocsit, majd eltolni a kimenő kapuk mellett – a számítógépek érzékelnék a kocsi tartalmát, és azonnal ráterhelnék a vételárat a vevő bankszámlájára.

Természetesen sokkal könnyebb volna a raktárkészlet fenntartása is, hiszen a termékek fogyatkozását figyelemmel követve az utórendeléseket is azonnal intézni lehetne. A kimutatások szerint a vásárlók által keresett árucikkeknek hétköznapokon 8, hétvégéken 11 százaléka nincs ott a polcokon. A dolog csak abban sántít, hogy ezt a jelenlegi számítógépes rendszerekkel is meg lehetne tenni, hiszen a pénztárak forgalma a vonalkódos számlázás segítségével napjainkban is valós időben figyelhető. Sokat elmond a nagyra törő tervről, hogy támogatója többek közt a Procter & Gamble, a Gillette, a Wal-Mart, az Unilever és a Tesco – csupa olyan nagy cég, amelyek vásárlók millióit szolgálják ki nap mint nap. Ezek a cégek a vásárlási szokásokból felépített adatbázisokból rengeteg értékes adatot nyerhetnének ki, amelyek segítségével például hatékonyabbá tehetnék hirdetéseiket, vagy nyomon követhetnék a vásárlók szokásait. Nem véletlen, hogy az adatvédelemmel foglalkozó szervezetek máris felemelték a hangjukat.



Sokan megfogalmazták már azt a rémítő jövőképet, hogy hamarosan minden eszköz – a mikrohullámú sütőkig bezárólag – internetkapcsolattal fog rendelkezni. Az eszközök által előállított hatalmas mennyiségű adatot valahogyan kezelni, tárolni kell – ebben lát üzleti lehetőséget a Sun is. Szerencsére a digitális azonosítók elterjedésére még várni kell, hiszen az előállítási költségek egyelőre túl magasak, de a Wal-Mart már elindította az első próbarendszert egy amerikai városban.

➲ http://www.autoidcenter.org

Bemutatták az AMD Hammer processzort

Az AMD egy teljes számítógépet – tehát nem egy elvont próbarendszert – mutatott be, amely új Hammer processzorával futott. Az AMD új lapkája x86-64 megoldást használ, így egyaránt képes az újabb 64-bites, és a megszokott 32-bites operációs rendszerek és programok futtatására. A bemutatón a 32-bites oldalt a Windows képviselte, a 64-bitest viszont Linux.

Az új processzorok 0,13 mikronos SOI módszerrel készülnek. A memóriavezérlőt beépítve tartalmazzák, a rájuk épülő rendszerek pedig támogatják majd a HyperTransport-megoldást, amely a számítógép egyes összetevői között folyó adatáramlás gyorsítását célozza.

➲ http://www.amd.com
➲ http://www.x86-64.org/

A Matrox új VGA kártyákat jelentett be

Újabb tagokkal bővül a Matrox népszerű MMS, Multi-Monitor Series sorozata. A Matrox G450 X2 MMS és a G450 X4 MMS kártyák érdekessége, hogy nem egy, hanem kettő és négy grafikus lapka található rajtuk, így egy-egy kimenethez a lapkák teljes számítási teljesítménye rendelkezésre áll. Mint nevük is utal rá, az X2-es típuson kettő, az X4-es típuson pedig négy lapkának szorítottak helyet, így két és négy monitor csatlakoztatható hozzájuk.

Analóg képernyő használata esetén a legnagyobb elérhető felbontás 2048×1536, míg digitális panelnél 1280×1024 képpont. A kártyákon 64 MB memória található, a megfelelő frissítésről 360 MHz-es RAMDAC gondoskodik. Természetesen a monitorok helyére tévékészülékek is kerülhetnek. A kártyák PCI-foglalatba helyezhetők, így operációs rendszertől függően akár 16 képernyő is köthető egyetlen számítógépre.

Érdekes szolgáltatás a Matrox-telepítőkészlet oldaláról, hogy windowsos végrehajtható fájlok formájában telepítési készletek hozhatók létre, amelyek már a Matrox PowerDesk beállításait is tartalmazzák, így a telepítés felügyelet nélkül is könnyedén elvégezhető.

Az új kártyák a második negyedévben lesznek elérhetők.

➲ http://www.matrox.com

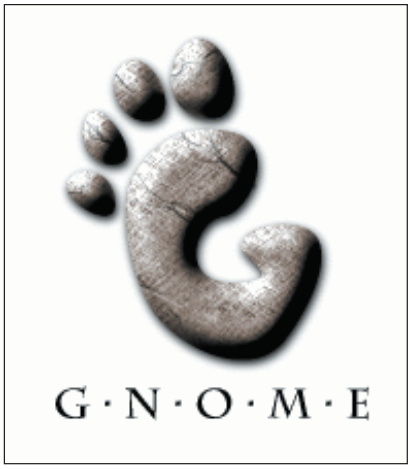
SuSE Linux tanároknak

A Széchenyi-terv pályázatainak keretében támogatással több mint tízezer pedagógus jut számítógéphez. A géphez operációs rendszer is jár, a SuSE Linux AG magyarországi irodája minden érintettnek ingyenesen ajánlja fel Linux-terjesztését. Az oktatónak szánt programcsomagban KDE ablakkezelő és OpenOffice irodai csomag található – természetesen mindkettő magyar nyelven. Az ingyenes rendszer segítségével az ECDL vizsgára is gond nélkül fel lehet készülni.

➲ http://www.suselinux.hu

Megjelent a Gnome 2.0 próbaváltozata

A legfrissebb ütemterv szerint május 1-jén meg kell jelennie a Gnome ablakkezelő 2.0-s változatának. Már sorban követik egymást a próbaváltozatok, a honosításra váró szövegek, a felhasz-



nálói felület véglegesítése pedig már megtörtént. A próbaváltozatok egyelőre hangsúlyozottan nem valók terjesztésekbe vagy üzemi használatra, ám a kísérletező kedvű rajongók máris elkezdhetnek ismerkedni vele.
➲ http://www.gnome.org

India szuperhálózatot épít

Az indiai kormány 10 éves fejlesztési tervének részeként olyan nemzeti hálózat kiépítéséről határozott, amely lehetővé tenné az országban található nagyteljesítményű számítógépek nagysebességű összekötését, így egyesítve feldolgozási teljesítményüket. A „grid" ötlete nem új, más nemzetek is élnek vele, ám India sem szeretne kiesni azoknak az országoknak a köréből, amelyek a legnagyobb számítási teljesítmények felett rendelkeznek a világon. Az így létrejövő hálózat képességeit rengeteg kutatási és szimulációs célra lehet majd felhasználni, a turbinák tervezésétől kezdve egészen az olajkutatásokig.
➲ http://news.com.com

Vízhűtéses hordozható számítógép a Hitachitól

A Hitachi olyan hordozható számítógépet készített, amely vízhűtéssel működik. A korábban csak a tuningőrültek által és különféle nagyteljesítményű rendszerekben alkalmazott hűtési módszer ezáltal teljes polgárjogot nyerhet a tömegtermékek piacán is. A cég hamarosan piacra szeretné dobni új típusát, amelyben egy Pentium 4 processzor által termelt hőt kell majd vízzel elvezetni. A korábbi vízhűtéses rendszerek súlyos gondja volt a hűtőközeg párolgása, valamint a hűtőrendszer korrodálódása. A Hitachi mérnökei állítólag megoldották ezeket a gondokat. Az új gépben vékony rozsdamentes csőben kering a víz, mozgatásáról a gép testében elhelyezett pumpa gondoskodik. A felmelegedett víz a képernyő oldalára kerül, és ott hűl le. A nem teljesen újfajta hűtés a hagyományosnál csendesebb, másfélszer hosszabb élettartamú, ugyanakkor a végtermék árát nem növeli lényegesen.
➲ http://www.pcworld.com
➲ http://www.hitachi.com

Elkészült a honosított OpenOffice

Mindössze egy hosszú hétvégére volt szüksége egy csapatnyi lelkes fiatalnak, hogy teljesen honosított irodai programcsomagot varázsoljon a régebben StarOffice, újabban a Sun

neve alatt kereskedelmi változatban StarOffice, valamint forráskódja kiadása óta OpenOffice név alatt – ennek megfelelően több változatban – is terjedő együttesből.
A magyar változathoz helyesírás-ellenőrző is társul, amelynek további javításához bárki hozzájárulhat. A windowsos változat fordítását sajnos egyelőre nem tudják megoldani, a műveletet elvégző hozzáértő személynek a tervezet támogatói 40 000 forint jutalmat ajánlottak fel.
➲ http://www.openoffice.hu
➲ http://www.szofi.hu/gnu/magyarispell/index.html

Bejelentették a CrossOver Plugin 1.1-es változatát

A CrossOver Plugin segítségével széles körben használt windowsos állományok – Microsoft Office-dokumentumok és eFAx-fájlok – és beépülő modulok érhetők el linuxos alkalmazások alól. A program 1.0-s változata 2001 augusztusában jelent meg, az újabb változat fontos újdonságokkal dicsekedhet: támogatja a sokfelhasználós környezeteket, ismeri az Apple QuickTime, a Macromedia Shockwave Director formátumokat és a Windows Media adatfolyamokat, képes a TrueType betűkészletek megjelenítésére, és újabb beépülő modulok támogatására.
Az új program ára internetről letöltve 25 dollár, CD-n megrendelve pedig 35 dollár.
➲ http://www.codeweavers.com

Linuxos forgalomirányítók

A forgalomirányítóban is van processzor, memória, így miért ne futhatna rajta Linux? Az Ayr Networks embereiben a kérdésfeltevést tett követte, és

kedvenc rendszerüket elsőként a Cisco 7600-as sorozatú forgalomirányítóira ültették át. A munka keretében át kellett szabni a rendszermagot, a merevlemez nélkül működő készülékekhez megfelelő környezetet kellett fejleszteni, a különféle nyílt forrású forgalomirányító protokollokat a hálózati szolgáltatók igényeihez kellett igazítani, valamint teljesen új modulokat és démonokat kellett fejleszteni, amelyek képesek a különleges csomagkezelő áramkörök vezérlésére.
A rendszer részét képezi a meglévő Cisco operációs rendszerrel egyenértékű felhasználói héj, valamint a rendszer részeinek vagy egészének frissítésére megfelelő alkalmazás.
Az AYR Linux egyelőre próbaváltozatban érhető el. Akik megfelelő Cisco forgalomirányítóval rendelkeznek, és szeretnék kipróbálni a rendszert, a cég honlapján jelentkezhetnek tesztelőnek.
➲ http://www.ayrnetworks.com

Újabb területre merészkedik az AMD

Az AMD bejelentette, hogy felvásárolja az Alchemy Semiconductort. Az Alchemy eddig magánkézben volt, és nagy

teljesítményű, MIPS-felépítésű mikroprocesszorok tervezésével, fejlesztésével és forgalmazásával foglalkozik. Termékei elsősorban mobileszközökhöz, webtáblákhoz, zsebtitkárokhoz készültek.
Az AMD a személyi számítógépek piaca mellett az internetes és mobileszközök piacán lát lényeges fejlődési lehetőséget. Az Alchemy alapítóinak munkáját dicsérik többek között az Alpha- és a StrongARM-processzorok. A két cég erőit – és termékeit – egyesítve minden bizonnyal sikeres portékák dobhatók piacra.
➲ http://www.amd.com
➲ http://www.alchemysemi.com

Megjelent a PostgreSQL 7.2 végleges változata

Közel egy év munka gyümölcseként elérhető a PostgreSQL 7.2-es végleges változata. Több területen is fejlesztettek az adatbázis-kezelőn, így javultak a honosítási lehetőségek, és a rendszernek négymilliárdnál nagyobb számú tranzakció kezelése sem okoz gondot, a program a jelszavak biztonságos kezelését és tárolását pedig MD5-algoritmussal szavatolja.

➲ http://www.hu.postgresql.org
➲ http://apachetoday.com/

A Sigma bejelentette az első PC-s MPEG-4 dekóderkártyát

A Sigma REALmagic Xcard kártyája az első olyan PC-s bővítőkártya, amely – számos további formátum mellett – az MPEG-4, MPEG-2 és MPEG-1 filmek dekódolására egyaránt képes. A kártya támogatja a legújabb megoldásokat, így többek közt a HDTV-ket, ismeri az

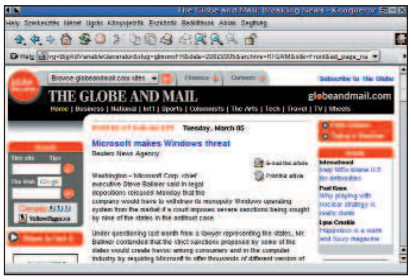
NTSC- és PAL-szabványokat, valamint – valószínűleg kevesek örömére – a Macrovision másolásvédelmi megoldást és a DVD-régiókat.

A kártya március 28-tól lesz elérhető, de az amerikai vásárlók már most is feladhatják előrendeléseiket a 99 dolláros kiegészítőre.

➲ http://www.sigmadesigns.com/

Mi lesz veled, Windows?

Régóta, különféle szereplőkkel és epizódokkal húzódik az a per, amely a Microsoft egyeduralmát vizsgálja a személyi számítógépek piacán, illetve a Microsoft Internet Explorer beépítését, pontosabban a Windows operációs rendszerekbe való beépítés tisztességes voltát firtatja. A Windowsok forráskódjának felfedése már korábban is szóba került, de nyilván a Microsoft sem fogja önszántából mutogatni legféltettebb kincsét. A cég ugyanakkor azt, hogy az Internet Explorer továbbra is a Windowsok része maradt, azzal indokolja, hogy mélyen beépült magába a rendszerbe, és az eltávolítása nem oldható meg annak tönkretétele nélkül. A felperesek most azt kifogásolják, hogy a Microsoft a forráskód bemutatása nélkül érvel, így állítását nem tudja érdemben



bizonyítani. Ezért a bíróság arra akarja kötelezni a céget, hogy független szakértőknek tegye lehetővé a Windows forráskódjának tanulmányozását. Mivel a felperes amerikai államok mögött a Microsoft riválisai állnak, a cég egyrészt megfelelő garanciákat kér a forráskód biztonságára, illetéktelenektől való megóvására, másrészt amennyiben operációs rendszereinek átszabására kényszerítik, a Windows a rendszerek piacáról való kivonásával fenyeget. Az „ötlet” több mint megdöbbentő, hiszen nemcsak a Microsoftra, de a teljes PC-iparra beláthatatlan következményeket gyakorolna.

➲ http://www.globeandmail.com

Felgyorsulhat a beszédfelismerésre épülő alkalmazások fejlesztése

A beszédfelismerésre épülő alkalmazások fejlesztését egyelőre gátolja az a sajnálatos tény, hogy a legtöbb megoldás egy-egy cég nevéhez fűződik, így a felhasználók választhatnak: vagy legyökereznek egyetlen gyártó mellett, vagy többféle alapra is elkészíttetik a szükséges alkalmazásokat.

Jelenleg a World Wide Web Consortium VoiceXML-szabványa az irányadó, amelynek segítségével telefonon keresztül lehet különféle alkalmazásokat elérni – a szerencsések lekérdezhetik például a bankszámlájuk egyenlegét, vagy az éppen érvényes részvényárfolyamokat.

Ezen a helyzeten változtathat a SALT (Speech Application Language Tags) szabvány erre a negyedévre tervezett megjelenése. A támogatók között olyan nagy cégeket találunk, mint a Cisco, az Intel, a Microsoft vagy a Philips. A fejlesztés célja olyan szabványos felület létrehozása, amely szóban feltett kérdésekre szöveges válaszok adását teszi lehetővé. A Microsoft már áprilisban be szeretne mutatni egy új fejlesztői készletet, valamint még ebben az évben megjelenik interaktív webes alkalmazások telepítésére alkalmas rendszerének próbaváltozata. Az első SALT-alkalmazások felbukkanására egy év múlva számíthatunk.

➲ http://www.w3.org/Voice/
➲ http://www.saltforum.org/

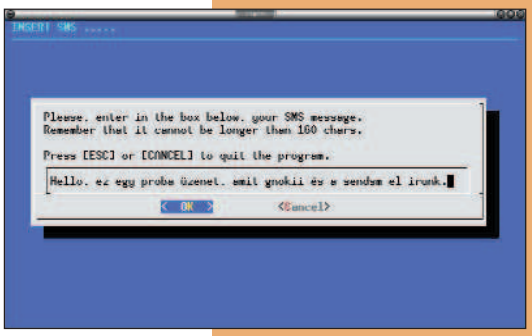
Először kerül a bíróságokra a GNU GPL

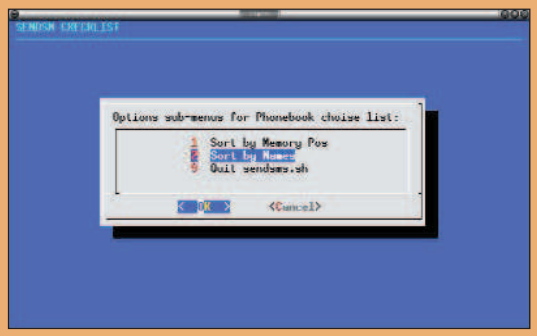
A MySQL AB, a MySQL adatbázis-kezelő fejlesztője bíróság elé citálta a Progress Software Corporationt, mivel ez utóbbi megsértette a MySQL-re is vonatkozó GNU GPL szerződést. A cég saját fejlesztésű Gemini programösszetevőjét statikusan összekapcsolta a MySQL-rendszerrel, és egyetlen futtatható programot készített az ily módon saját és GPL alá eső részeket egyaránt tartalmazó NuSphere termékből. A cégnek a forráskódot is mellékelnie kellett volna programjához, ám ezt nem tette meg. Természetesen az ügy nem ennyire egyszerű, és nem is most kezdődött, a két társaság közötti vita hasonló okok miatt már több mint egy éve húzódik.

Medgyesi Zoltán (mzx@axelero.hu)
A BMGE 24 éves informatika szakos hallgatója. Szabadidejét legszívesebben a barátnőjével tölti. Szeret autózni és bográcsban főzni. A Linuxot hat éve ismeri, de még nem volt lelkiereje, hogy áttérjen rá. A Linuxvilág magazin hírszerkesztője.

Nokia-mobiltelefonok Linux alatt

Tulajdonképpen mi lehet az értelme annak, hogy mobiltelefonunkat Linuxra kössük? Semmi és nagyon is sok haszna lehet, az öreg linuxosokban (itt elsősorban azokra gondolok, akik legalább 4–5 éve használnak Linuxot) már kialakult egy olyan szemlélet, hogy amikor valamilyen új





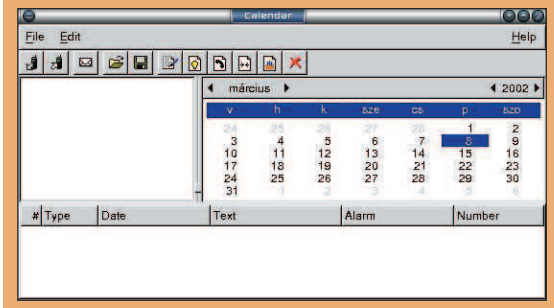
számítógép-alkatrészt vásárolnak, elsődleges szempont, hogy működjön Linux alatt is. Nos, így van ez egy mobillal is. Bármikor szinte bárhonnan betárcsázhatunk vele, és a Linux (például egy notebook) segítségével bármilyen adatot lekérhetünk, gépeket felügyelhetünk stb. Vagy éppen asztali SMS-kiszolgálót alakíthatunk ki belőle, hiszen milyen hasznos, ha az elfelejtett találkozókról munkahelyi számítógépünket akár vásárlás közben is lekérdezhetjük, vagy éppen a tárgyalás előtt félórával értesít bennünket. Mindent összegezve nem árt, ha a mobiljainkkal is tudunk varázsolni Linux alatt. Akik járatosak Linux és a mobil rejtelmeiben, kitalálhatták, hogy a Gnokiiról és elsősorban az infrakapuval rendelkező Nokia-telefonokról lesz szó.

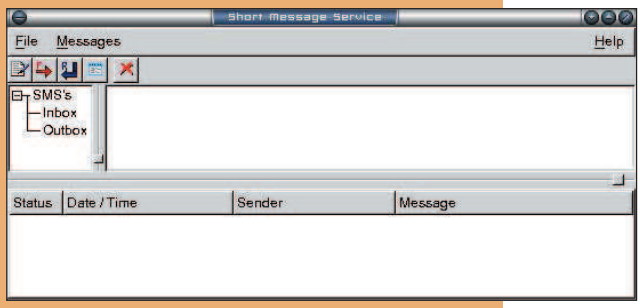
Nézzünk pár példát, miért is érdemes ezt a cikket továbbolvasni:

1. Jó pár évvel ezelőtt barátomban és bennem megfogalmazódott az igény egy olyan szolgáltatásra, amely a következőképpen működik: amikor az utcán járva az ember pénzszűkébe kerül, ha van miből, illetve mivel pénzt kivennie egy bankautomatából, komoly gondot okoz, hogy hol van a közelben olyan automata, amelyből a pénzfelvételi költség az én bankkártyámmal a legmegfelelőbb (vagyis a saját bankom automatája). Így hát terveztünk egy egyszerű rendszert, amely úgy működött, hogyha az ember a város, illetve a kerület néhány koordinátáját SMS-ben elküldte egy Linuxra kötött Nokia 5110-re, akkor a Linux egy internetes adatbázisból kikereste a hozzá legközelebb eső automatákat és SMS-ben (ha kellett, több részletben) értesített, hogy hol találom a kívánt bankot, illetve automatát. Ez a rendszer – bár csak ketten használtuk – 1998-ban egyedülálló volt, ám sajnos egy merevlemez-összeomlás és a mentés hiánya miatt az örök enyészeté lett.
2. Linux alatt sok különböző eszköz adott arra, hogy a gépeinket figyeljük, például: BigBrother, mon, netwatch, netsaint. Szinte mindegyik képes e-mailt küldeni egy megadott címre, ami akár Westel, illetve Pannon SMS-sé is tud alakulni, ha a szolgáltatónál megfelelő előfizetésünk van, de sokszor előfordul, hogy a levélben feladott SMS csak jó pár órával a feladás után érkezik meg. Ez természetesen egy behatolás érzékelésénél nem elfogadható sebesség. Ha viszont egy mobilt aggatunk a figyelő gépre, garantált lesz, hogy szinte azonnal értesít minket. Ehhez szükség van némi hozzáértésre, és például egy Nokia 5110-re, összekötő kábelre és egy PrePaid előfizetésre.
3. Távoli bejelentkezés a leállt (holt) kiszolgálóra: kiszolgálónk egy co-location központban áll. Szinte minden lehetőségünk megvan, hogy hozzáférjünk a kiszolgálónkhoz, ha azonban az ott dolgozók válaszideje nem túl jó, vagy például hétvégén nincs ügyelet, célszerű lehet, ha a gépünkre telefonon is be tudunk jelentkezni. Ezt megtehetjük egy modem és vonalas telefon segítségével, csak nem jellemző, hogy a szolgáltató erre lehetőséget adna. Itt is megoldás lehet egy mobil és a `megtty` csomag (a későbbiekben még részletesen ismertetjük).

Nézzük akkor, mire van szükségünk a fentiek megvalósításához! Ha csak SMS-ek küldésére akarjuk használni, elegendő a 10–20 ezer forintból beszerezhető Nokia 5110, egy soros kábel és egy például PrePaid előfizetés. Szükségünk van továbbá a Gnokii programra. Töltsük le a terjesztésünknek megfelelő csomagot vagy a forrást, és rakjuk fel. Ha soroskapu-kábellel rendelkezünk, egyszerű a dolgunk, mert a telepítés után a saját könyvtárunkban lesz egy *.gnokiirc* állomány, ami a beállításokat tartalmazza.

```
[global]
# Ide kell a kaput  rni, ahova a
# telefonunkat k t tt k.
/dev/ircomm0,/dev/ttyUSB0 stb.
port = /dev/ttyS0

# Ide ker l a telefonunk t pusa:
# 5110, 6110, 6210 stb.
model = 6210

# A kapcsol dÆs m dja: serial, irda
connection = serial

# Hol talÆlhat k a Gnokii-binÆrisok
bindir = /usr/local/sbin/

# HasznÆljon-e eszk z lockot, ez
# fontos lehet, mert ha egy idıben
# ketten pr bÆlnak az eszk zh z
```






```
# fØrni, a kapcsolat megszakadhat
use_locking = yes

# A soros kapcsol dÆs sebessØge
serial_baudrate = 19200
```

Ezekután soros vagy infrakapun keresztül máris használni tudjuk a készüléket. A főpróbát a Gnokii `--monitor` kapcsolóval kezdjük. Ha működik, akkor valami ilyesmit kell látnunk:

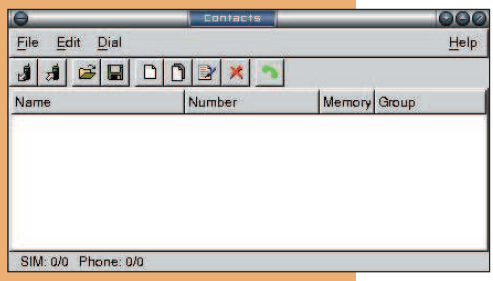
```
Network: Westel 900 (Hungary),
➥LAC: 006e, CellID: 2b65
RFLevel: 40
Battery: 40
SIM: Used 140, Free 60
Phone: Used 140, Free 360
DC: Used 0, Free 256
EN: Used 0, Free 1
FD: Used 0, Free 12544
MC: Used 0, Free 512
ON: Used 0, Free 15
RC: Used 2, Free 766
SMS Messages: Unread 0, Number 27
```

Amennyiben idáig eljutottunk, nagyon nagy az esélye annak, hogy ugyanez grafikus felületen is menni fog, tehát próbálkozzunk meg az Xgnokii elindításával.

A Gnokii minden szolgáltatását kényelmesen elérhetjük grafikus alkalmazásból, de akár egyszerű héjprogramból is. Azoknak, akik az SMS-küldést önműködővé szeretnék tenni, a konzolos elérés lesz a megfelelő – de ha csak egyszerűen üzenni akarunk a barátunknak, sokkal kézenfekvőbb a grafikus felületen kattintgatni. Pár szolgáltatás, amelyet elérhetünk:

- telefonkönyvek szerkesztése, illetve tárolása,
- SMS-küldés és -fogadás,
- operátorlogó szerkesztése,
- bekapcsoláskor megjelenő logó szerkesztése,
- netmonitor működővé tétele, illetve lekérdezés,
- idő-, dátum-, illetve ébresztésállítás,
- csengőhangfeltöltés.

Innentől már a képzeletre van bízva, mire is használjuk a Linuxszal egybekötött telefont. Abban az esetben, ha a Gnokiit konzolon akarjuk használni, mert nincs lehetőségünk X-re, vagy csak úgy érezzük, hogy gyorsabbak vagyunk konzolon, egy nagyon hasznos, a Gnokiihoz írt programot tudunk letölteni. Ez a *sendsms.sh* névre hallgat. Egyszerű héjprogramról van szó, amely párbeszédszerű felépítéssel rendelkezik. Nagyon ügyes, mert nem kézzel kell rögzítenünk a kapcsolókat, valamint a telefonszámokat az első futtatás alkalmával letölti a telefonról, így lehetőségünk nyílik rá, hogy a későbbiek során egy listából válasszuk ki a címzett nevét és telefonszámát. Most már csak programozói kedv és elérhető adtabázisok kérdése, hogy milyen szolgáltatásokat teszünk elérhetővé saját magunk számára: ha a banki példát szeretnénk megvalósítani, az adatbázist keressük a ➲ http://www.bankkartya.hu oldalon. Továbbá bátran írhatunk egy egyszerű pár soros Perl-programot, ami a már meglévő kalendáriumunkból (például az ical) kiválogatja a találkozók időpontját, és előttük SMS-ben értesít bennünket.



Remélhetőleg, mindenkinek sikerült kedvet csinálnunk ahhoz, hogy már meglévő telefonját összekösse a Linuxával. A következő számban részletesen foglalkozunk a kiszolgálóoldali megoldásokkal, pontosabban azzal, hogyan is tudja egy a kiszolgáló soros kapujára kötött mobiltelefon megkönnyíteni az életünket (betárcsázás, távfelügyelet stb.).

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

Kapcsolódó címek

Gnoki ➲ http://www.gnokii.org
SendSMS ➲ http://pserver.samba.org/cgi-bin/cvsweb/checkout/gnokii/utils/sendsms
Bankkártya-adatbázisok:
➲ http://www.bankkartya.hu
A mobiltelefonhoz szükséges kábeleket beszerezhetjük például a
➲ http://www.mobiltelefonok.hu oldalon.

Ingyenes CAS bármely operációs rendszerre: a HartMath

Egy Javában íródott, GPL szerződés alá eső Számítógépes Algebra Rendszert szeretnék most bemutatni. Okos és gyors – még a mai grafikus számológépek mellett is tartalmaz meglepetéseket.

Ha még nem hallottál a Számítógépes Algebra Rendszerről (CAS), röviden elég annyit tudnod, hogy olyan számológép, amely számok mellett képleteket is kezel. Vannak változói, képes törteket gyökteleníteni, és még rengeteg feladatot megoldani, hogy a matematikai műveleteknél véletlenül se számoljuk el magunkat. Programozható is, lehet benne ciklusokat, elágazásokat készíteni. A ➲ http://freshmeat.net-en keresgélve két ígéretes fejlesztést is találtam. Az egyik a Yacas (Yet Another Computer Algebra System), amely a ➲ http://www.xs4all.nl/~apinkus/yacas.html címen érhető el, a másik pedig a HartMath ➲ http://www.hartmath.org. Ez utóbbival szerzett tapasztalataimat szeretném most megosztani veletek.

A telepítés

A most következő lépések az előfordított Java *.jar* állomány telepítését írják le. A program forráskódja természetesen nyílt, bárki lefordíthatja, én azonban maradtam a kész *.jar* fájlnál. Mindenekelőtt ha még nem rendelkeznél Java Virtuális Géppel (JVM), telepíts egyet. Javaslom a Sun-féle J2RE (Java 2 Runtime Environment) megvalósítást. Ez jelenleg az 1.3.1-es változatnál tart, eléggé megbízható, és viszonylag gyors is. A telepítőt a ➲ http://java.sun.com címről ingyenesen letölthetted. Ez egy *.bin* kiterjesztésű állomány, és ha kiadod a `bash j2re-verzio.bin` parancsot, a felhasználási szerződés elfogadása után szó nélkül kicsomagolja magát a pillanatnyi könyvtárba. Én ezt követően átmásoltam a */opt*-ba, és */usr/bin/java* néven készítettem egy közvetett hivatkozást a */opt/jre1.3.1_02/bin/java* állományra (amely szintén csak egy hivatkozás, de ez ebből a szempontból lényegtelen). Mivel a */usr/bin* benne van a `PATH` környezeti változóban, a Java már futtatható. Ellenőrzésképpen beírhatod a következőt:

```
$ java -version
```

Amire én ezt kaptam:

```
java version  1.3.1_02
Java(TM) 2 Runtime Environment,
➥Standard Edition (build 1.3.1_02-b02)
Java HotSpot(TM) Client VM (build
➥1.3.1_02-b02, mixed mode)
```

Ezekután itt az idő, hogy beszerezd a HartMathot. A fentebb említett címről letölthetsz egy Javában írt telepítőt. Ha megvan a *hartmathverzioinstall.jar* fájl, a java `-jar fajlnev.jar` paranccsal indítsd el egy terminálról. Ez azonban grafikus felületet igényel, tehát a legjobb, ha az X-et rendszergazdaként indítod (és utána gyorsan ki is lépsz). A telepítés innentől már nagyon egyszerű. Én a */opt/HartMath07pre19* könyvtárba telepítettem, és egyúttal ide tettem a rendszer pdf formátumú kézikönyvét. Ez nincs benne a telepítőcsomagban, ugyancsak a fenti címről külön tölthető le.

Ismerkedés a rendszerrel

Javáról lévén szó nem lepődtem meg azon, hogy az ablak két kattintás után nem pattan fel azonnal a képernyőre. Egy kicsit bosszantott, hogy az `xosview`-t figyelve a program az indítás során 17 MB fizikai memóriát evett meg, és ekkor még egy gombot sem nyomtam le. A kezelőfelület szépnek nem nevezhető, de könnyen átlátható és tanulható. A felső menüsorban a már megszokott *File* mellett, az *Edit* alatt érheted el a vágólappal kapcsolatos szolgáltatásokat: másolás, kivágás és beillesztés. A további két menüben a gyakran használt függvények és példák kaptak helyet.

A menüsor alatt található beviteli mezőben van lehetőség a képletek beírására és szerkesztésére. Ez alatt számológéphez hasonló elrendezésű gombok kaptak helyet. A számok, zárójelek, logikai és egyéb műveleti jelek mellett itt találhatod meg a 16-os (0x), a 8-as (0o), és a 2-es (0b) számrendszerhez tartozó előtagokat (prefix). Ezek mellett jobb oldalt található egy legördülő menü, amelyben a kimeneti formátumot adhatod meg. A számos lehetőség közül most csak a LaTeX- és az XML-lehetőségeket emelném ki. A legalsó vezérlőelem első fülén láthatod a program kimenetét. A második fül egy periódusos rendszert foglal magában, a harmadikon háromdimenziós függvényeket rajzolhatsz, míg a negyediken paraméteres egyenletek jeleníthetők meg grafikonon.

A nyelv alapjai

A rendszer programozható. Saját nyelve van, amely nagyon könnyen elsajátítható. Sőt, ha valaki ismeri egy kicsit a C-t, annak a HartMath ebből a szempontból semmi újat nem fog mutatni.

Mindenekelőtt meg kell értened a numerikus és a szimbolikus kiértékelés közti különbséget. Ha egy kifejezést numerikusan értékelünk ki, csak a számeredményt kapjuk meg, még akkor is, ha az esetleg nem pontos (akár a számológép). Ellenben a szimbolikus kiértékelésnél a végtelen szakaszos tizedes törtek megmaradnak két szám hányadosaként, az irracionális számok nem kerülnek „kiszámolásra" stb. A legjobb példa a gyök kettő, amelyről bizonyított, hogy irracionális. Numerikus kiértékelés esetén az `Sqrt(2)` parancsra az 1,4142135... eredményt kapjuk, szimbolikus esetén viszont gyök kettőt.

Az alapműveletek a C-hez hasonlóan a +, -, *, / jelekkel végezhetők el. A hatványozás jele a ^, a faktoriálisé a !. Zárójelek is használhatók, például a (2^3)/(2-4+5*3) kifejezést szimbolikusan kiértékelve a 8/13 eredményt kapjuk. Lehetőség nyílik változók használatára. Így az x=2; y=3; x*y kifejezés értéke 6. Összevont műveleti jelekkel is dolgozhatunk: x/=2 (ami egyenértékű az x = x / 2 kifejezéssel). Használhatunk függvényeket is, többek között a gyökvonást, a szinuszt és koszinuszt, a vektorok értelmezése pedig csak ezek segítségével oldható meg. Egy változó törlése is egyedül függvénnyel lehetséges: `Clear(x)`.

Jelentősebb függvények

- `Print(arg)`
  A megadott argumentumot kiírja a képernyőre, és Null értékkel tér vissza.
- `Print( Szia vilÆg! )`
- `Div(arg1,arg2)`
  Az `arg1` és `arg2` maradékos osztásából származó egész rész.
- `Div(13,8) -> 1`
- `Mod(arg1,arg2)`
  Az `arg1` és `arg2` maradékos osztásából származó maradék.
- `Mod(13,8) -> 5`
- `Log(arg)`
  Az `arg` természetes alapú logaritmusa.
- `Log(E^3) -> 3`
- `Sqrt(arg)`
  Az `arg` négyzetgyöke, például `Sqrt(25) -> 5`.
- `PieChart(lista)`
  „Tortadiagramot” rajzol egy újabb fülre a listában megadott értékek alapján. A lista elemei { és } kapcsos zárójelek között helyezkednek el, és vesszővel vannak elválasztva, például `PieChart({1,2,3})`.
- `IsProbablePrime(arg)`
  A JVM `BigInteger.isProbablePrime` megvalósítás alapján nagy valószínűséggel meg tudja állapítani, hogy prím-e, például `IsProbablePrime(63) -> False`.
- `Fibonacci(n)`
  Meghatározza a Fibonacci-sorozat n-edik elemét, például a `Fibonacci(3333) ->` kifejezésre egy egész képernyőt betöltő számot kaptam, méghozzá meglepően gyorsan.
- `SolveP(polynom,valt)`
  A legfeljebb másodfokú polinomot megoldja a változóra, például a `SolveP(x^2,x)` kifejezés az x^2=0 egyenlet gyökeit keresi. Eredményül a {0,0} listát kapjuk, ebből látszik, hogy egyetlen megoldása van.

Programozni jó

A HartMath saját nyelve is tartalmazza az összes eset+szétválasztásos módszert és ciklusszervező eljárást, én mégis csak kettőt mutatnék be. A nyelv primitívekből építkezik, így egy elágazás nem több, mint egy függvény a megfelelő értékekkel ellátva.

```
If (allitas, igaz_ag, hamis_ag,
➥egyeb_ag)
```

Ha az `allitas` igaz, az `igaz_ag` értékelődik ki, ha hamis, akkor a `hamis_ag`, és ha logikailag nem értelmezhető az `allitas`, akkor pedig az `egyeb_ag`.

```
If(22,Print( Igaz ),Print( Hamis ),
➥Print( nem logikai kifejezØs! ))
➥  nem logikai kifejezØs!
If(True,Print( Igaz ),Print( Hamis ),
➥Print( nem logikai kifefejezØs! ))
➥-> Igaz
For (elott,feltetel,utan,kifejezes)
```

Az `elott` a ciklusba történő belépés előtt fut le. A `feltetel` minden iteráció elején, és ha hamis lesz, a ciklus nem fut tovább. Az `utan` minden iteráció végén értékelődik ki. A `kifejezes` a ciklusmag.

```
For (i=0,i<10,i++,Print(i))
➥-> 0 1 2 3 4 5 6 7 8 9
```

Ezek alapján készítsünk egy olyan programot, ami 2-től 100-ig írja ki a prímszámokat. Nyilván szükség lesz egy számlálós ciklusra. Ezen belül ellenőrizzük, hogy a ciklusváltozó prím-e, és ha igen, írjuk ki a képernyőre.

```
For (i=2,i<100,i++,If(IsProbablePrime
➥(i),Print(i),Null,Null))
```

A `Null` értékekre azért van szükség az `If()`-ben, mert a mezőket nem lehet üresen hagyni, még akkor sem, ha az adott ágba semmit sem akarsz semmit.

Függvények, fájlok használata

A HartMathban saját függvények írására is lehetőség nyílik. Egy jól megírt függvény nem baj, ha megmarad a merevlemezen, és később vissza lehet tölteni. Ezt a lehetőséget egy példán keresztül szeretném bemutatni.

```
Clear(x); kobre(x_)=x^3
```

Ez a kifejezés létrehoz egy `kobre()` függvényt, amely az átadott értéket köbre emeli. Ki is próbálhatod:

```
kobre(2) -> 8
```

Fontos, hogy a függvény változói a függvény meghatározása előtt más kifejezésekben nem használhatók. Ezért szerepel egy `Clear()` utasítás még az elején. Láthatod azt is, hogy a függvény változói az értéklistában egy aláhúzásjellel bővülnek.

Van már egy jól működő függvényünk, de ha most kilépsz a programból, elveszik. Ha későbbre is meg szeretnéd őrizni, a mentésre használd a `WriteObject()` függvényt. Ezzel nemcsak függvényeket, hanem bármilyen kifejezést, például egy előző számítás végeredményét tartalmazó változót is menteni lehet. Az így mentett állományok a *<saját könyvtár>/.hartmath/hmscripts* könyvtárba kerülnek.

```
WriteObject( kobos ,kobre(x_)=x^3)
```

Ezzel a fenti könyvtárban létrehozol egy *kobos.hm* különleges formátumú fájlt. Ezt a `ReadObject()`-tel lehet visszatölteni:

```
ReadObject( kobos )
```

Végezetül

Mindent egybevéve a HartMath nagyon jó rendszer. Látszik rajta, hogy nem érte még el az 1.0-s változatot, mert a kézikönyvben vannak olyan függvények, amelyek a programban nem működnek, viszont üzembiztos, így a mindennapi használatra is nyugodtan ajánlhatom.

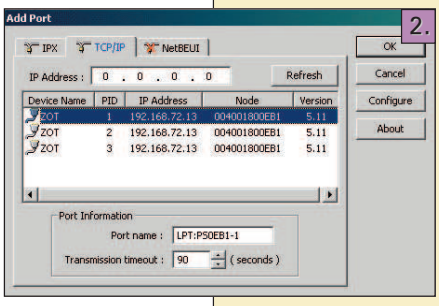
Fülöp Balázs (xut@freemail.hu) 17 éves, imádja a Túró Rudit, a Debian Linuxot és a teheneket. Az ELTE Radnóti Miklós Gyakorlóiskola tanulója immár ötödik éve. Kedvenc írója Slawomir Mro□ek. Leginkább a számítógépes hálózatok biztonsága érdekli.

Nyomtatni jó

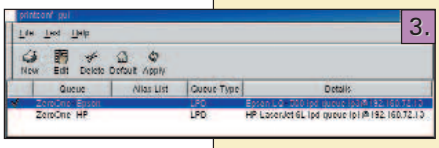
Ez első hallásra elég bugyuta megállapításnak tűnhet, ám bizonyos kimutatások szerint annak ellenére, hogy az informatika korába léptünk, és gyakorlatilag minden olvasnivaló elérhető és megtekinthető valamilyen képernyőn, mi azért mégiscsak szeretünk nyomtatni, méghozzá egyre többet. Jobban szeretjük ugyanis a jelentéseket mutatós könyvtárakba helyezni, sokan elektronikus leveleiket is szívesebben olvassák nyomtatva – így nyomtatási vágyainknak semmi sem szab gátat.


1.

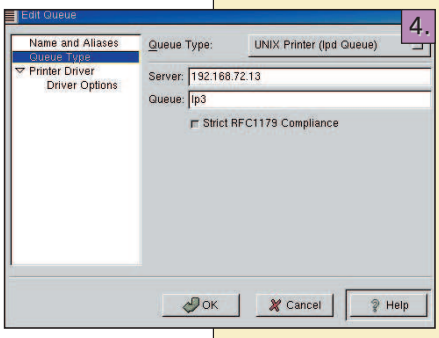
Ha már féktelenül nyomtatunk, általában megpróbáljuk olcsón tenni. Egy asztali tintasugaras nyomtatóba kéthetente patront venni ugyanis anyagi szempontból elég megrázó, nagyobb teljesítményű, olcsón üzemeltethető lézernyomtatót viszont nem érdemes minden asztalra venni – meg kell osztani a munkatársak között. A megosztást rengetegféle módon lehet intézni. Ha a munkacsoport rendelkezik valamilyen – akár Novell NetWare-t, Windowst vagy Linuxot futtató – központi kiszolgálóval, akkor érdemes ezt befogni a célra, ha viszont nem, gondok lehetnek, hiszen a kiszolgálókkal ellentétben a munkaállomások nem biztos, hogy folyamatosan elérhetők, így nem célszerű rájuk közös nyomtatót kötni.



2.



3.



4.

Sokszor kellemetlen az is, hogy a jelenleg még főleg párhuzamos kapura csatlakoztatott nyomtatók használata sok erőforrást foglal le – a múltból ránk ragadt ősi párhuzamos kapu kezelésekor a gép teljesítménye feltűnően romlik. Jobb tehát lerázni ezt a terhet: itt lépnek a képbe a nyomtatókiszolgálók, mégpedig a szónak abban az értelmében, amely egy apró dobozkát jelent. A dobozka sokféle lehet, de mindenképpen jellemző rá, hogy önálló tápellátással és hálózati kapcsolattal rendelkezik, többféle hálózati protokollt támogat, így vegyes hálózatokban is használható, valamint egy vagy több nyomtató kezelésére képes. Az egyszerűbb típusok csak 10 Mb/s sebességű ethernetcsatolót kapnak, és általában csak egyetlen párhuzamos kapuval rendelkeznek, míg a nagyobbak 10/100 Mb/s sebességű csatolóval és három kapuval is bírnak. Én ez esetben a nagyobb teljesítményű változatokból szereztem be négy típust.

Mivel azonos célra készültek, nem meglepő módon a képességeik nagyon hasonlóak; ezen kár is volna meglepődni, autóból is mindenki négykerekűt gyárt, a lényeg pedig ezúttal is a részletekben rejlik: mennyire kezelhető jól a készülék, esetleg mennyi és milyen jótállást vagy támogatást kapunk hozzá.

Zero One Technologies P300

Mivel a téma iránti érdeklődésemet a Humansoft AHT hírlevelében meglátott Zero One Technologies – a továbbiakban ZOT – termék keltette fel, vele kezdeném a termékek ismertetését. Rajta keresztül igyekszem bemutatni mindazt, amit a készülékek rendkívül hasonló tudásából fel tudtam mérni – sajnos mindenre kiterjedő vizsgálatot nem sikerült végeznem, például NetWare kiszolgáló nem volt kéznél (*1. kép*).

Miután nyomtatókiszolgálónkhoz hozzájutottunk, az első, amit szemügyre kell vennünk, a csomagolás. A ZOT termékét gondos szivacságy óvja a Tajvan – Magyarország viszonylatban megtett utazás viszontagságaitól, ez esetben viszont meg kell szabadulnunk tőle. A tápegység apró papírdobozban rejlik, a mellékelt programot két hajlékonylemezen kapjuk, és a dobozban még egy angol nyelvű könyvecskét is találunk, amely kellő részletességgel taglalja a telepítést Windows, Novell NetWare, Sco Unix és Solaris operációs rendszer alatt.

Első lépésként adjunk áramot a készüléknek, és nem árt egy hálózati kapcsolat sem. A tápellátás meglétéről egy erősen pirosló LED fénye értesít, a hálózati kapcsolat létrehozását pedig zöld társa nyugtázza. Magát a hálózati forgalmat a készülék tetején elhelyezett zöld lámpácska villogása jelzi – a kezelő –, és visszajelző szervek felsorolása ennyiben ki is merült. A készülék hátulján kapott helyet a három párhuzamos kapu, amelyeket sajnos elfelejtettek megszámozni, így a szomszéd szobában tanyázó munkatárstól szerzett filctollal hamarosan kiélhetjük művészi és díszítési hajlamainkat. Én ezt a fontos lépést kihagytam, tekintettel a készülék itt tartózkodásának ideiglenes jellegére.

Második lépésünk célszerűen a felügyeleti program telepítése, ami ebben az esetben egy kisméretű, könnyen kezelhető és jól áttekinthető, PSAdmin névre hallgató – windowsos – program telepítését jelenti. Indításkor a PSAdmin szórásos csomagokkal megkeresi a nyomtatókiszolgálót, aminek azonnal megkezdhetjük a beállítását. Először is adhatunk neki jelszót, amivel a többi felhasználót megakadályozhatjuk a beállítások módosításában. Adhatunk neki nevet is, ez a könnyebb azonosítást szolgálja akkor, ha több készüléket is csatlakoztatunk a hálózatra.

A készüléknek adhatunk IP-címet, illetve úgy is dönthetünk, hogy ezt DHCP-kiszolgálótól szerezze be. Amennyiben NetWare-környezetben dolgozunk, többek közt megadhatjuk, hogy a készülék melyik NDS-fába tartozzon, milyen néven legyen elérhető, illetve milyen jelszóval jelentkezzen be. Ha a hálózaton AppleTalk protokollt is használunk, meg kell határoznunk a zóna nevét, és ki kell választanunk az egyes kapukra csatlakoztatott nyomtató

típusát, valamint az általa használt adatformátumot. Mivel SNMP segítségével távolról is megfigyelhetjük a nyomtatókiszolgálót, megadhatjuk a kapcsolattartó nevét, helyét, a közösségi neveket, valamint beállíthatjuk az SNMP-csapdák működését.

Ha a kiszolgáló beállításával megküzdöttünk, még korántsem végeztünk, hiszen a megfelelő kapukat az ügyfeleken is be kell állítani, valamint telepíteni kell a nyomtatók illesztőprogramjait.

Windows alatt változattól függő, hogy a nyomtatót először fel kell-e telepítenünk a helyi párhuzamos kapura csatlakozó eszközként, majd később módosíthatjuk az általa használt kaput, vagy azt a telepítés közben is létrehozhatjuk. A lényeg mindkét esetben ugyanaz: a létrehozható és a rendszerhez hozzáadható kaputípusok között – hála a készülék illesztőprogramjának – megjelenik egy új, Zero One Network Port nevű. Ezt kell kiválasztanunk, majd a felbukkanó ablakban megadni, hogy a kiszolgáló melyik kapuját szeretnénk használni. Adhatunk neki valamilyen könnyen megjegyezhető nevet is, majd miután a rendszert az új kapuval kibővítettük, ennek használatára kell beállítani az új nyomtatót, és végeztünk is (*2. kép*).

Természetesen nem kötelező az illesztőprogram szolgáltatásait kihasználni, Windows NT vagy 2000 alatt szabványos TCP/IP-kaput is telepíthetünk, és házi nyomdánkat lpr nyomtatóként használhatjuk.

Linux alatt – Red Hat 7.1 terjesztéssel próbálkoztam – sem bonyolult a dolog, a printtool indítása után várakozási sorként válasszunk UNIX lpd-t, adjuk meg a kiszolgáló IP-címét, a kiszolgálón pedig az lp1/lp2/lp3 várakozási sort válasszuk attól függően, hogy melyik kaput szeretnénk használni. Természetesen a nyomtató típusát is ki kell választani, valamint néhány további, általános beállítást is megadhatunk, például a papírméretet (*3–4 kép*).

Elsőre én is így tettem, csakhogy a HP 6L nyomtató a próbanyomtatáskor szemetet kezdett el gyártani. A gondon a „Strict RFC1179 Compliance” jelölőnégyzet bekapcsolása segített, ettől kezdve mind a HP, mind a kiszolgáló másik kapujára kötött tűs Epson LQ-550 nyomtató szépen nyomtatott.

Természetesen nem kell feltétlenül windowsos gépet fenntartanunk arra a kizárólagos célra, hogy a nyomtatókiszolgálót beállítsuk. Lehetőség nyílik arra is, hogy `telnet`-en keresztül lépjünk be rá, majd egy egyszerű menü pontjaiban lépkedve minden beállítást módosíthatunk, valamint állapotfigyelést is végezhetünk.

```
Telnet Service on the PrintServer

Password:
** Main Menu on Printer Server **

  1. Look at status in Print Server
  2. Setting value in Print Server
  3. Load Default
  4. Reset Print Server
  0. Exit Setup

Enter your choice ->
```

Ha a telnetes felület sem nyeri el a tetszésünket, kerítsünk egy webböngészőt, és a címsorba írjuk be a nyomtatókiszolgáló IP-címét! Mivel egyszerű webkiszolgáló is helyet kapott benne, pillanatok alatt megjelenik a készülék állapotáról tudósító oldal, és a beállítások módosítására is lehetőség nyílik (*5. kép*).

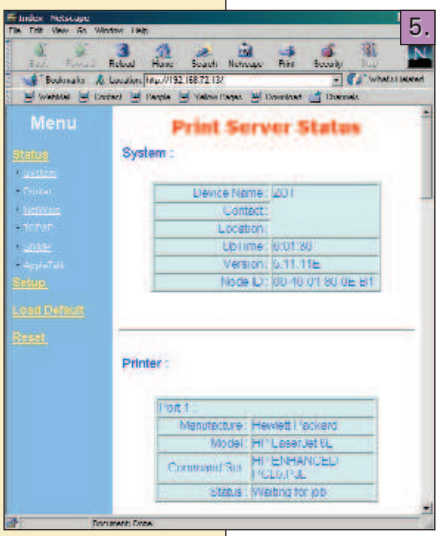
Lehetőség van arra is, hogy a nyomtatókiszolgáló IP-címét DHCP-kiszolgálótól szerezze be. Ezt a lehetőséget nem próbáltam ki, de nem is nagyon értem: ha a kiszolgáló dinamikusan változó IP-címet kap, az ügyfeleken viszont IP-cím alapján telepítettük a kapukat, akkor honnan fogják tudni, hogy a kiszolgáló éppen melyik IP-címet használja? Az sem kellemes dolog, hogy ha nincs kéznél a PSAdmin futtatására alkalmas windowsos gép, akkor telepítéskor a kiszolgáló fizikai címe alapján – szerencsére ez rá van nyomtatva a doboz aljára – kell kinyomoznunk annak alapállapotban felvett IP-címét. Ezeken az apróságokon viszonylag könnyen túl lehet lépni, de azért nem árt számolni velük.

A gyártó honlapja szégyenlősen bújik meg a ➲ http://www.zot.com.tw címen, és sajnos csak az ottani – valamilyen távol-keleti – nyelven érhető el. A szerencsénk az, hogy a HTML-fájlok nevét ők is kénytelenek voltak angolul írni, így a hivatkozásokat figyelve nagyjából tájékozódhatunk az oldalon. Például frissítéseket tölthetünk le, hiszen a készülék belső programja flashmemóriában található, és szükség szerint frissíthető.

D-Link DP-300

Miután a ZOT termékét nagyjából kiveséztem, a D-Link hasonló készüléke következett. Érdekessége, hogy a felhasználók között két párhuzamos

5.



6.

Adatok

Végfelhasználói árak, jótállás:
Edimax: kb. 40 000 Ft + áfa, 3 év;
Linksys: kb. 46 000 Ft + áfa, 1 év;
D-Link: kb. 45 000 Ft + áfa, 5 év;
ZOT: 47 800 Ft + áfa, 2 év, 1 hét pénzvisszafizetési garancia.

Forgalmazók

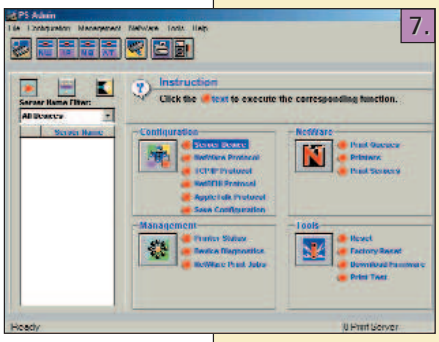
Edimax: Kelly-Tech Kft.
➲ http://www.kellytech.hu
telefon: 350-1246

D-Link: CHS Hungary Kft.
➲ http://www.chs.hu/
telefon: 451-3500

Linksys: Alphasonic Kft.
➲ http://www.alphasonic.hu
telefon: 350-6822

ZOT: Humansoft AHT Kft.
➲ http://www.humansoftaht.hu
telefon: 414-4048

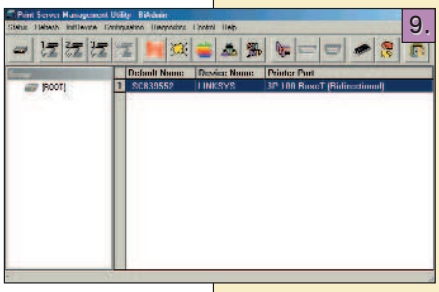
és egy soros kaput oszt meg. Jómagam ugyan még sosem láttam soros kapura csatlakozó nyomtatót, de ahol ilyet használnak, ott jó szolgálatot tehet. Érdekessége még, hogy nem gumitalpakat kapott, hanem kisebb vágatokat a hátoldalára, így a dobozban mellékelt kiegészítők segítségével falra is szerelhető. A jelzőfények az előlapon kaptak helyet, külön lámpa jut a tápnak, a hálózati kapcsolatnak és minden egyes kapunak (*6. kép*).


7.

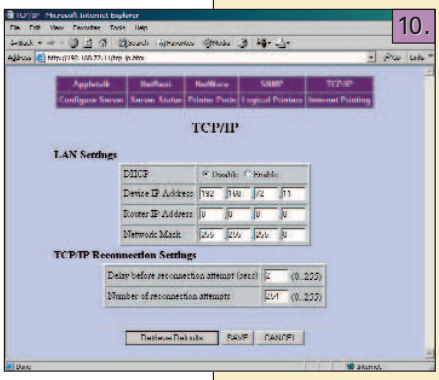
A D-Link ugyancsak gondos csomagolással látta el termékét, a vaskos szivacspárnák közt biztosan nem esik bántódása a fekete színű doboznak. A csomag tartalmát egy papírlapon sorolják fel, a telepítés legfontosabb lépéseiről kis, többnyelvű füzet tájékoztat.


8.

A CD-lemezen kapott felügyeleti programot a fejlesztők – alighanem tomboló fantáziájuk nyomásának engedve – PS Admin névre keresztelték. A programot futtatni felér egy időutazással, mindazonáltal még Windows XP alatt is remekül működött. Sajnos csak akkor volt hajlandó megtalálni a kiszolgálót, ha IPX/SPX protokollt is telepítettem, ami – figyelembe véve, hogy lassan kikopik a használatból – nem biztos, hogy túl szerencsés (*7. kép*).


9.

A PS Admin beállítási lehetőségei minden igényt kielégítők, a TCP/IP, IPX/SPX, AppleTalk és NetBEUI protokollok támogatását külön-kibekapcsolhatjuk, a kapuk sebességét és egyéb tulajdonságaikat egyenként adhatjuk meg – ez a lehetőség a ZOT esetében valamiért elérhetetlen volt –, a készülék az IP-címét RARP, BOOTP és DHCP segítségével egyaránt beszerezheti.


10.

Kihasználva a CD által biztosított bőséges helyet, a D-Link a korongon pdf formátumban mindenre kiterjedő kézikönyvet helyezett el. A leírásban – természetesen angolul – részletesen taglalják a PS Admin és a használatához szükséges IPX/SPX protokoll telepítésének és a készülék beállításának menetét. A kézikönyvre is ráférne némi frissítés, hiszen hiába ad értékes tudnivalókat a Windows NT 3.51 vagy a Novell NetWare 3.x alatti telepítéssel kapcsolatban, ha az újabb Windows és Novell operációs rendszereket meg sem említi.

Újdonság, hogy a készülék elvileg támogatja a NetBEUI protokollt, és a Microsoft Networköt tallózva ezáltal megosztott nyomtatóként látszik. Pontosabban látszana, mert érthetetlen okból – bizonyára az én ügyetlenségem miatt – nem látszik. Ettől még lehet használni, ugyanolyan TCP/IP-alapú eszközként kell telepíteni, mint a ZOT termékét. Ehhez nemcsak Windowsokhoz, hanem BSD-, Sco Unix- és Solaris-rendszerekhez is részletes telepítési leírást kapunk. Mivel a Windows 9x/ME-rendszer alapállapotban nem támogatja a TCP/IP-alapú nyomtatást, egy megfelelő illesztőprogramot is mellékeltek a CD-n, a telepítésével úrrá lehetünk ezen az apró gondon.

Érdekesség még, hogy a D-Link DP-300-ra FTP- és TFTP-protokoll segítségével is nyomtathatunk, ha egyszerű szöveges állományokról van szó. A gyártó üzemi nyomtatásra nem javasolja ezt a lehetőséget, tesztelni azonban kiválóan lehet vele.

Itt érdemes megemlíteni, hogy FTP-alapú nyomtatáskor – ha több gépről is nyomtatunk egyszerre – a különféle nyomtatási feladatok összekeveredhetnek. Más körülmények között ilyesmit nem tapasztaltam. Amikor mindkét nyomtatóm egyszerre küzdött egy-egy nagyméretű pdf fájllal, érzésem szerint mintha kicsit lelassult volna a feldolgozás, de a két nyomtató egymástól függetlenül, párhuzamosan dolgozott.

```
ftp 192.168.72.12
Connected to 192.168.72.12.
220 FTP server ready.
User (192.168.72.12:(none)):
231 User name accepted.
ftp> put teszt.txt DLINK-P1
200 OK.
150 Opening data connection.
226 Closing.
ftp: 1563 bytes sent in 0,02Seconds
78,15Kbytes/sec.
ftp> quit
221 Goodbye.
```

Linksys EPSX3

A Linksys jellegzetes kék-naracs színű dobozában a nyomtatókiszolgálót tojástartó jellegű védőborítás vette körül. Tervezői a készüléket némi lekerekítéssel tették áramvonalassá, amely azért így sem kapna formatervezési nagydíjat – igaz, nem is ez a cél. Meglepetést okozott, hogy amikor még nem kapott tápot, de az egyik nyomtató kábelét már csatlakoztattam rá, varázslatos módon kigyulladt rajta a piros hibajelző lámpa. Emellett egyébként még egy állapotjelző található, amely a hálózati forgalomról is értesít, valamint a fizikai hálózati kapcsolat meglétét egy további jelzőfény nyugtázza. A hálózat sebességét a készülék oldalán elhelyezett DIP-kapcsolókkal állíthatjuk

önműködő választásúra, 10 és 100 Mb/s-ra, valamint fél- és teljes kétirányú módba is válthatunk. Ugyancsak itt kapott helyet egy RESET gomb is (*8. kép*).
Az EPSX3 alapszolgáltatásairól az eddigiek fényében kár lenne külön szólni, az illesztőprogram telepítése után a rendszerhez hozzáadható kapuk listájában újabb típus jelenik meg, és a kicsit korosnak tűnő BiAdmin nevű felügyeleti program sem vonultat fel különösebb újdonságokat. Dicséretes ugyanakkor, hogy a mellékelt CD-n a legkülönfélébb operációs rendszerekhez nagy mennyiségű leírást és mindenféle illesztőprogramokat találunk, sőt még DOS alá is kapunk beállító segédprogramot (*9. kép*). Sokkal érdekesebb ennél, hogy a Linksys mintha egy kicsit többet tudott volna kihozni a témából. A mellékelt programok segítségével például internetes nyomtatót telepíthetünk. Ekkor a nyomtatni kívánt anyagot a nyomtató felé elektronikus levélben fogjuk továbbítani, a leveleket az illesztőprogramnak kell a túloldalon a levelezőkiszolgálóról letöltenie, majd gondoskodni a nyomtatásukról.

```
Welcome to Print Server

PS>monitor

(P1)STATE: Idle
TYPE: Parallel
PRINTER STATUS: On-Line

(P2)STATE: Idle
TYPE: Parallel
PRINTER STATUS: Offline

(P3)STATE: Idle
TYPE: Parallel
PRINTER STATUS: On-Line

PS>
PS>exit
```

A Linksys készüléke `telnet` alapon nem állítható be, ezzel a módszerrel csupán megfigyelést végezhetünk. A készülék beépített webkiszolgálója viszont igényes felületet biztosít, a segítségével minden beállítást elérhetünk (*10. kép*).

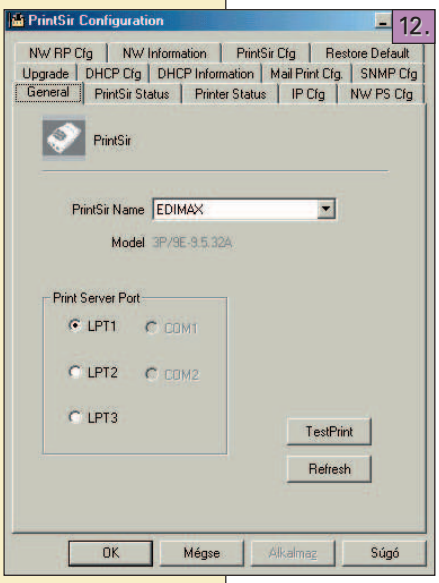
Edimax PS-3101P

Az Edimax készüléke nem a méreteivel hívja fel magára a figyelmet, hanem azzal a vörös színű műanyag betéttel, amelyet a szürke színű doboz tetejére illesztettek. Semmi unalmas szögletesség: ha ügyesen helyezzük el az irodában, akár jól is nézhet ki. Ő kapta a legtöbb lámpát, külön fény jelzi a tápellátást, a hálózat 10 vagy 100 Mb/s sebességű voltát, valamint az egyes kapuk használatát. Csomagolása tömör, célratörő, szakképzettség nélkül egyszersmind megismételhetetlen (*11. kép*). A mellékelt CD-t a meghajtóba helyezve Windows alatt hamarosan egy apró program bukkan fel, amely felajánlja az ügyfélként és a rendszergazdaként való telepítést. Előbbi esetben csupán a kiszolgáló használatához szükséges illesztőprogramok kerülnek a gépre, ha pedig az utóbbit választjuk, mellékjük egy felügyeleti program, valamint egy Internet Explorer és Netscape Navigator alatt használható beépülő modul is társul, amellyel elvileg böngészőn keresztül is lehetségessé válna a készülék felügyelete – ez sajnos nekem nem sikerült.
Különösebben nem is hiányzott, hiszen a böngészőbe egyszerűen a kiszolgáló IP-címét beírva a beépített webkiszolgálót használhatjuk, másrészt a PrintSir nevű felügyeleti programmal minden beállítás elérhető. A programozók ezúttal nem akartak túl nagyot és ötleteset alkotni, az egyes tulajdonságokat egyszerűen lapokra csoportosították. Ezekből alapállapotban is meglehetősen sok van, de ha IPX/SPX-protokollt is telepítünk, akkor tovább osztódnak (*12.kép*).
A kapuk hozzáadásával ez esetben nem nekünk kell bajlódnunk. Első telepítéskor megadhatjuk, hogy mely kapuk – és milyen névvel – legyenek elérhetők, később a telepítő magától hozzáadja őket a nyomtatók által használhatók listájához, és nekünk csak válogatnunk kell közülük.
A PrintSir is felkínálja az elektronikus levélben érkezett anyagok nyomtatásának lehetőségét. Leírások tekintetében ugyanakkor kicsit mostohán bánnak velünk. A gyors telepítési útmutatót, valamint a teljes kézikönyvet ízlés szerint angol és hagyományos kínai nyelven olvashatjuk. Maguk a leírások részletesek, kivitelezésük igényes, ám kizárólag Windows és NetWare operációs rendszereket említenek meg, Unix-rendszerekkel kapcsolatos szolgáltatások szóba sem kerülnek. Az eddigiek alapján azonban nem szoríthatnak sarokba bennünket, hiszen az IP-címet felhasználva a korábbiakhoz hasonló lpr nyomtatóként eszközeinket most is telepíthetjük.

Kapcsolódó címek

- http://www.edimax.com/
- http://www.dlink.com
- http://www.linksys.com
- http://www.zot.com.tw


11.



12.

Medgyesi Zoltán (mzx@axelero.hu)
A BMGE 24 éves informatika szakos hallgatója. Szabadidejét legszívesebben a barátnőjével tölti. Szeret autózni és bográcsban főzni. A Linuxot hat éve ismeri, de még nem volt lelkiereje, hogy áttérjen rá. A Linuxvilág hírszerkesztője.

Linux-index

1. A 65 év felettiek aránya Németországban várhatóan ekkorra éri el a lakosság ötven százalékát: 2030
2. A 65 év feletti népesség növekedési arányának ennyiszeresével fog csökkenni a 35 év alatti német lakosság, ha a születési arány nem változik: 2
3. Ennyi millió bevándorlót kell majd Németországnak évente befogadnia, hogy a jelenlegi munkaerőt eltartsa: 1
4. Ennyiszer nyerte meg *Tove Torvalds* a finn karatebajnokságot: 6
5. Ennyibe kerül az a Windows hálózati megoldás, amit a CRN tesztközpontban kipróbáltak: 4,688 dollár
6. Ennyibe kerül a vele egyenértékű, Linux-alapú hálózati megoldás ugyanabban a CRN tesztközpontban: 317 dollár
7. Ennyi százalékos ármegtakarítást jelent a Linux a Windowszal szemben a CRN tesztközpontban: 93
8. Az emberiség ennyi százaléka él napi két dollárnál kevesebb pénzből: 50
9. Ennyi milliárd ember él napi egy dollárnál kevesebb pénzből: 1
10. Percenként ennyi nő hal bele a szülésbe: 1
11. Ennyi napi aktatologatást jelent egy pékség hivatalos megnyitása Kairóban: 500
12. Ennyi százalékkal nőtt a webes hibafelügyelet (1×1 képpontos ellenőrző képek) alkalmazása három év alatt, 2001 augusztusáig: 500
13. A száz leglátogatottabb weboldal közül ennyi használ valamilyenfajta weboldalszármazékot (kéretlen ablakok nyitása) (százalék): 30
14. A száz legnépszerűbb európai tartomány közül ennyi alkalmaz származékot (százalék): 20
15. Az Interneten a webhelyek ennyi százaléka alkalmaz „egérfogót”, amellyel megakadályozza a weboldal bezárását és a Vissza gomb használatát: 5,7
16. A szexuális partnerek átlagos száma a 16–55 évesek körében az Egyesült Államokban: 14,3
17. A nemi érintkezés átlagos, évi gyakorisága ugyanebben a körben az Egyesült Államoban: 124
18. A fenti két eredmény helyezése a kutatásba bevont 27 ország körében: 1

Források

1–3.: Economist
4.: Open Source Initiative
5–7.: Computer Reseller News
8–11.: *Bill Clinton*
12.: CNET, a Cyveillance adatai alapján
13–15.: Cyveillance

Linux Torvalds megkapta a Világtechnológia díjat

Linus Torvalds nyerte a 2001. évi Világtechnológia díjat a kereskedelmi szolgáltatások osztályában. Ezeket a díjakat a World Technology Network (Világtechnológia Hálózat) adományozza „elismerésül azoknak az egyéni vezetőknek vagy esetenként csapatoknak, akik a legnagyobb mértékben segítették elő az üzleti élet és a társadalom javát szolgáló különféle technológiák kifejlesztését”. Azokat az újítókat tüntetik ki, akiknek a közelmúltban végzett munkája hosszú távon várhatóan igen nagy jelentőségűnek és hatásúnak bizonyul majd, és akik valószínűleg a következő évek technológiai életének kulcsfigurái lesznek vagy maradnak.

Ezenkívül a díjakat „olyan személyiségek kaphatják meg, akik jelenlegi munkája a szervezet szerint a legösztönözőbb hatással lesz a jövőre… akár előre látható, akár váratlan területeken.”

„Linus Torvaldsot választottuk a Linuxszal kapcsolatos tevékenységéért és a nyílt forrású programmodell megteremtéséért” – olvasható a díjak weboldalán.

„Linus Torvalds írta a Linux rendszermagját, és ő alkotta meg a nyílt forrású programmodellt, amely forradalmi újításnak tekinthető a programkészítés terén. Nemcsak az egyik legjelentősebb program megtervezése fűződik a nevéhez, hanem egy új, átfogó programtervezési modell létrehozása is.”

Tovább idézve a weboldalt:

„A Linux az egyik legfontosabb operációs rendszer, vetekszik a Unixszal és az MSDOS-szal. Különösen nagy fontossággal bír a hordozható kommunikációs eszközök, a webkiszolgálók és az Internet fejlesztése terén, valamint számos más számítási, hálózati és IT-területen. Linus Torvalds nemcsak kimagasló tehetségű programtervező, de a nyílt forrás programközösség vezető személyisége is.”

A 2001. évi díjak nyerteseinek nevét 2002 elején hozták nyilvánosságra. Huszonhárom tárgykörben osztottak díjat, s minden osztályban öt esélyes jelölt közül kellett választani. Néhány további tárgykör győztesei:

- *Lawrence Lessig*, a Stanford University jogi professzora és szerző – a Jogi tevékenység csoportban.
- *Robert Metcalfe*, az ethernet feltalálója és a 3Com alapítója – a Kommunikációs technológia osztályban.
- *Gordon Moore*, az Intel társalapítója és nyugalmazott elnöke – az IT Számítógép-alkatrészek tárgykörben.
- *Shawn Fanning*, a Napster szerzője – két osztályban is nyert: a Szórakoztatás és a Vállalkozás osztályában.

A díjazottak teljes listája a Nature honlapján tekinthető meg: ➲ http://www.nature.com/nature/wta

Doc Searls

Linux-alapú Google-szerkezet

Emlékszem az esetre, amikor néhány évvel ezelőtt a Linux egyik szakértője a Google-ról beszélt nekem. Azt mondta, hogy az új keresőmotor – amely akkor még csak a nyilvános bétaváltozatban létezett – kiemelkedően jó lesz, mert Linux-kiszolgálókra építették fel. Nem hittem neki. Abban az időben a HotBot volt a kedvenc keresőmotorom, amelynek teljesítménye az engem legjobban érdeklő területen mindig felülmúlta az összes többi keresőmotor teljesítményét: a dokumentumoknak olyan karakterláncok alapján történő megtalálásánál, amelyek egy része az oldal szövegében mélyen el van temetve. A HotBot nem sokkal azt megelőzően szorította ki az AltaVistát, amelyet a keresőmotorok közül az első helyre tettem. Az AltaVista előtt az InfoSeeket kedveltem (azon kevesek közé tartoztam, akik az InfoSeek szolgáltatásainak tényleges előfizetői voltak). Még korábban pedig a Lycos, a Carnegie-Mellon egyik tudományos fejlesztése állt a legközelebb hozzám. Végül a HotBot alulmaradt a FAST-tal, e BSD-alapú motorral szemben, amely a meglehetősen szerencsétlen alltheweb.com címre hallgatott. Ellenállásom azonban hiábavalónak bizonyult, a Google végül győzött.

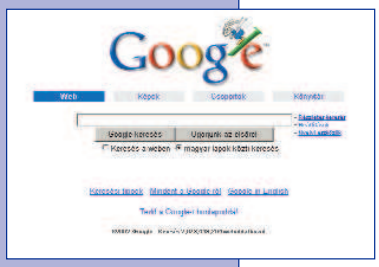
Eleinte ugyanis nem szerettem ezt a keresőt, mert túlságosan egyszerűnek tartottam, és mert mindenáron tudni akarta, mi az, amit keresek. Ezt utáltam benne – és ma is utálom. Mégis megkedveltem, ugyanis nagy ámulatomra mégiscsak tudja, mit akarok – persze nem mindig, de eléggé gyakran. A legtöbbünkhöz hasonlóan most már szinte csak a Google-t használom.

Ma már a többi motor esélytelen vele szemben: a Google a jelek szerint az új szolgáltatások minden egyes lépcsőfokán (képek és hírcsoportok keresése, fájltípusú keresések, más nyelvek) egyre jobban maga mögött hagyja versenytársait.

Eltelt már egy kis idő, amikor utoljára beszéltem a Google munkatársaival, ezért úgy gondoltam, bejelentkezem és megtudok tőlük néhány részletet – például a választ a legfontosabb létkérdésre: vajon még mindig eredményesek-e? Ezért elmentem régi barátomhoz, *Cindy McCaffrey*-hez, a Google marketingalelnökéhez, aki a következőket mondta nekem:

„Nyereségesek vagyunk. A hirdetések jelentősen hozzájárulnak a nyereségünkhöz. Mindkét hirdetési programunk (Premium Sponsorships, AdWords) rohamosan bővül. Sok ezer hirdetőnk van, és hirdetési kampányunkat éppen most kezdtük el kiterjeszteni más országokra is, továbbá néhány kisebb hirdetésértékesítő irodát nyitottunk meg az Egyesült Királyságban, Japánban és Németországban."

Ezt rendkívül érdekesnek találtam, ugyanis a Google-oldalakon megjelenő hirdetések az újságokban közölt apróhirdetésekhez hasonlítanak, és a reklámozás egyetlen olyan formáját képviselik, amelyekre nagy az olvasói igény. A Google reklámjai az apróhirdetésekhez hasonlóan nem tolakodóak és nem tartalmaznak grafikákat. Amikor az egyik hirdetőtől megkérdeztem, eredményesek-e ezek a hirdetések, ezt felelte: „Nagyon is azok. Mi csak a Google-ban hirdetünk." A helyzet az, hogy olyannyira eredményesek, hogy annak ellenére is ott hirdetnek, hogy helytelenítik a Google azon politikáját, miszerint a technológiájukat szabadalmaztatják. Cindy hozzátette: „A kulcsszavakra irányuló megközelítési mód nagyon jól működik. A látogatók több mint két százaléka rákattint a reklámokra, és ez az arány körülbelül ötször magasabb a hagyományos szalagfejléces hirdetések esetében megszokottnál. Ezenkívül keresési szolgáltatásokat nyújtunk más cégeknek, például a Yahoo!-nak, a Ciscónak, a Sonynak stb. – mintegy 130 ügyfélnek hozzávetőlegesen 30 országban. A két forrásból származó bevételeink aránya nagyjából 50–50 százalék".



Megkérdeztem, hogy változott-e bármiben is a társaság küldetése. Gyanítottam, hogy nem, mivel soha nem hajoltak meg az olyan zavaró tényezők előtt, amelyek a legyőzött versenytársakat a jelentéktelenség vidékére száműzték: részvényárak, sporteredmények, közös promóciók a szórakoztatóipar webhelyeivel stb. Azt felelte, nem, missziójuk ma is változatlan: „A világ információinak elrendezése olyan módon, hogy egyetemesen elérhetőek és felhasználhatóak legyenek."

Talán nem túlzás azt állítani, hogy a Google sokunk számára a világháló háttérrendszerének, keresési illesztőegységének a részévé vált. Annak érzékeltetésére, hogy milyen messzire is nyúlnak el ezek az illesztőegységek, megkértem Cindyt, küldjön el nekem néhány számadatot. Az alábbi adatokat kaptam tőle:

- adatközpontok: 4
- Linux-számítógépek: >10,000
- keresések száma naponta: >150 millió
- weboldalak indexe: >1,6 milliárd
- képanyag: >330 millió
- Usenet-üzenetek: >650 millió
- hírcsoportok: >5 000
- nyelvi alcsoportok az indexben: 28
- nemzetközi tartományoldalak: 23
- PDF-ek: >22 millió

Ezek közül sokan „elsősorban a világhálón vannak" – tette hozzá szerényen. Viszont nem volt hajlandó megerősíteni annak a cikk elején említett szakértőnek a feltételezését, aki először hívta fel a figyelmemet a Google-ra: hogy mindennek a Linux az oka. Ezt a következtetést saját magunknak kellett levonnunk.

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

Új termékek

BRU-Pro 2.0 és BRU 17.0

A Tolis Group inc. megjelentette a BRU-Pro és a BRU Workstation legújabb változatait. A BRU-Pro 2.0 Linux-kiszolgáló az ügyfél és kiszolgáló közötti kapcsolat biztonsága érdekében titkosítja a hálózati adatforgalmat, valamint a hálózati forgalom tömörítése útján a sávszélességgel is takarékoskodik. A BRU Workstation 17.0 kis- és közepes méretű vállalkozásokat támogat, a BRU Desktop 17.0 otthoni, illetve kisirodai rendszereket támogat – helyileg csatlakoztatott archiválóeszközökkel. A BRU Personal Edition 17.0 pedig a nem üzleti célú felhasználók számára nyújt biztonságos adatkezelést.
A BRU-Pro 2.0 és a BRU 17.0 felújított grafikus felhasználói felülettel rendelkezik, és támogatja a 64-bites fájlrendszereket.
Adatok: The Tolis Group, Inc., 10225 East Via Linda, Suite 300, Scottsdale, Arizona 85258
telefon: 480-346-2008
e-mail: sales@tolisgroup.com
➲ http://www.tolisgroup.com

Volution Manager 1.1

A Caldera Inc. kiadta a Volution Manager 1.1-et, amely webalapú rendszerkezelő megoldás több rendszer böngészőprogramon keresztüli biztonságos távoli felügyeletére és frissítésére. Az 1.1-es változat új jellemzői: többféle rendszer támogatása egységes kezelőfelületen keresztül, egyszerűsített telepítés, javított állapotjelentés és diagnosztika.
A Volution Manager 1.1 minden nagyobb Linux-terjesztés legfrissebb változatát és a Caldera Unix-termékeit is támogatja.
Adatok: Caldera, Inc., 240 West Center Street, Orem, Utah 84057
➲ http://www.caldera.com

cPCIS-2103 Chassis

A cPCIS-2103 Chassis az ADLINK Technology 3U CompactPCI termékcsaládjába illeszkedő legújabb terméke. A 19 hüvelykes állványba szerelhető vagy asztali kivitelű számítógépház a PICMG 2.0 szabványnak teljes mértékben megfelel.
A gépház az elsődleges és a másodlagos oldalon is hat bővítőhellyel rendelkezik, amelyekbe a felhasználók által meghatározott 32-bites bővítőkártyák kerülhetnek. A gépházban a rendszerprocesszornak, három működés közben cserélhető tápegységnek és a további bővítésekhez egy PCI–PCI-hídnak van helye.
A cPCIS-2103 önmagában vagy egy rendszer beépített részeként vásárolható meg.
Adatok: ADLINK Technology 15279 Alton Parkway, Suite 400, Irvine, California 92618
telefon: 1-866-423-5465
➲ http://www.adlinktechnology.com

X4 NAS

A NetEngine Inc. megjelentette az X4 NAS-t, amely kis- és közepes méretű vállalatok, munkacsoportok, fiókirodák és szolgáltatók hálózati tárolóeszközeként használható. Az X4 NAS támogatja a többfelhasználós üzemmódot, alkalmazható fájlok megosztására és biztonsági mentések tárolására. A tárméret 160 GB és 480 GB közé eshet, szabványos 1U-méretű, amely állványba szerelhető. Az X4 NAS a hibatűrés fokozására két 10/100TX ethernetkártyát tartalmaz. Az operációs rendszer meghibásodását tükrözéssel küszöbölik ki. További jellemzői: beépített RAID, önműködő adatellenőrzés és önműködő újraépítés.
Adatok: NetEngine, Inc., 4116 Clipper Court, Fremont, California 94538
telefon: 510-668-2112
e-mail: solutions@netengine1.com
➲ http://www.netengine1.com

PowerUpdate 2.0

A PowerUpdate 2.0 több felületen futó, Java-alapú programfrissítő és -terjesztő megoldás. A webböngésző, az adatbázis, a jelentéskészítő modulok és az egyedi kezelőprogram összessége alkotja a PowerUpdate-et, amellyel bármilyen program tetszőleges ügyfél- vagy kiszolgálófajtára frissíthető. A fejlesztők kézben tarthatják a frissítési folyamatot, meghatározhatják, hogy mi frissüljön, mikor és milyen módon. A 2.0-s változat újdonsága a fájlösszehangolás, az MSI támogatása, a Mac OS X támogatása, és az archívfájlok kicsomagolásának és futtatásának képessége. A PowerUpdate fut Linux, Solaris, HP-UX és AIX operációs rendszereken.
Adatok: Zero G Software, 514 Bryant Street, San Francisco, California 94107
e-mail: info@ZeroG.com
➲ http://www.ZeroG.com

GFS 5.0

Már elérhető a Sistina Global File System (GFS), amely lehetővé teszi, hogy a megosztott adattárat SAN-ban elhelyezett több kiszolgáló is írhassa és olvashassa. A GFS 5.0 újdonsága a továbbfejlesztett telepítő- és fürtbeállító eszközök, a többutasság dinamikus támogatása az adattár kötetkezelőjében az egyutas hibák kiküszöbölése érdekében, megosztott gyökérfájlrendszer, további zároláskezelők és a gyártóktól származó pillanatfelvételi lehetőségek továbbfejlesztett támogatása.
Adatok: Sistina Software, 1313 Fifth Street Southeast, Suite 111, Minneapolis, Minnesota 55414
telefon: 612-638-0500
➲ http://www.sistina.com

Optimizeit Suite

Az Optimizeit Suite eszköztárának segítségével a fejlesztők tetszőleges Java-program fejlesztése során könnyen felfedezhetik a sebességet és a megbízhatóságot befolyásoló gondokat. Az Optimizeit Suite teljesen beépülhet az alkalmazáskiszolgálóba, távoli folyamatok kapcsolódhatnak hozzá, jól méretezhető, és az adatok minden J2EE alkalmazásméretnél pontosan szabályozottak.
A programcsomag három részből áll. A *Profiler* segít megtalálni a hibás programkódot vagy algoritmust, és javítja a memóriaszivárgásokat. A *Thread Debugger* a szálak állapotát valós időben képes megjeleníteni. Végül a *Code Coverage* megmutatja, hogy a program az egyes eljárásokat és kódsorokat milyen gyakorisággal hajtja végre.
Adatok: VMGEAR, 1479 Saratoga Avenue, Suite 200, San Jose, California 95129,
telefon: 1-888-655-0055,
➲ http://www.vmgear.com

A hónap szakmai tanácsai

A régi rendszermaggal működő modem nem megy az újjal

Miután Red Hat 7.1-ről 7.2-re frissítettem a rendszeremet, a modem nem működik, ha a gépet az SMP-rendszermaggal indítom. Ha nem SMP-rendszermagot használok, nincs gond. Letöltöttem és lefordítottam a legújabb megbízható rendszermagot, de a hiba nem szűnt meg. A 7.1 alatt a modem SMP rendszermag alatt is működött. US Robotics 56 K-s FaxModemem (3CP5610A típus), Tyan alaplapom és két Intel Pentium 133-as processzorom van.
Nathan Myers, myersn@voyager.net

Ellenőrizd, hogy a rendszermag felismerte-e a modemet! Ezt a `grep ttyS /var/log/messages*` parancs kiadásával nézheted meg, amely kiírja a soros eszközöket, bár ezek némelyike az alaplapra szerelt.
Christopher Wingert, cwingert@qualcomm.com

Köszi, hogy megadtad a modem típusszámát, ez a leghasznosabb adat. A Google keresőrobot segítségével megtudtam, hogy ez egy PCI-modem, amely kissé másképp működik, mint a régimódi ISA-modemek. A jó hír az, hogy igazi modemről van szó, nem Winmodemről. A legjobb találat a következő:
➲ http://www.idir.net/~gromitkc/3cp5610.txt. Az USR szintén írt egy példa parancsfájlt a Red Hat számára: a ➲ http://www.usr.com/support/drivers-template.asp?prod=s-modem címen érhető el.
Marc Merlin, marc_bts@valinux.com

Az NFS a múltban él

Az egyik gépemen (Red Hat 7.2), amely a felhasználók saját könyvtárait tartalmazza, ext3 fájlrendszer található, ehhez több más gép (mind Red Hat 6.2) is csatlakozik. Észrevettem, hogy a csatlakozó gépeken a frissített fájlok nem frissülnek a kiszolgálón, és a többi csatlakoztatott gépen sem érvényesülnek a változások. A kiszolgáló a következő beállításokat exportálja: `rw`, `no_root_squash`. Az ügyfelek beállítása pedig `defaults`, `nodev`, `rw`. Úgy tűnik, mintha az ügyfél gyorstárazna, de soha sem ürítené a tárat (van olyan fájl, amely több, mint egy napja megváltozott, azonban a kiszolgálón még mindig változatlan).
R. K. Owen, rk@owen.sj.ca.us

A NFS semmilyen beállításnál nem gyorstáraz egy fájlt ilyen hosszú ideig. A legkézenfekvőbb dolgokat ellenőrizném, azaz: az ügyfelek tényleg az NFS-en keresztül beillesztett könyvtárba írnak-e, és elérik-e az NFS-kiszolgálót. Ellenőrizheted azt is, hogy az ügyfelek a kiszolgálón megváltoztatott fájlt újnak látják-e.
Marc Merlin, marc_bts@valinux.com

A fel nem ismert PCI-modem

Nemrég vettem egy US Robotics 56 K-s PCI modemkártyát, mely nem Winmodem, ezért is döntöttem mellette. Terveim szerint Linuxot és Windows felváltva futtató gépen fogom használni. A Windows 98 COM5-re csatlakozó eszközként ismerte fel a szokásos COM2 helyett.
A Linuxban létrehoztam a */dev/ttyS4*-et, és a `setserial` paranccsal beállítottam a kaput és a megszakítást. Létrehoztam az erre mutató */dev/modem* hivatkozást. Ha a minicom alkalmazást futtatom, nem mondja, hogy nem találja az eszközt, de nem csinál semmit. Hogyan ismertethetném fel a modemet a Linuxszal? Próbáltam az `echo ATH1>/dev/ttyS4` parancsot is, de a vonalhang nem szólt a modem hangszórójából, ezért meglehetősen biztos vagyok benne, hogy a parancs a modemig el sem jutott.

Tony Preston, apreston@k2nesoft.com

Ez úgy hangzik, mintha az IRQ nem lenne megfelelően beállítva. Ellenőrizd az IRQ értékét az `lspci -vv` paranccsal. Keresd meg a modemet a listán, és a *setserial* programmal ezt az IRQ-értéket állítsd be.
Christopher Wingert, cwingert@qualcomm.com

Nézd meg a
➲ http://www.idir.net/~gromitkc/3cp5610.txt oldalt, ahol leírják, hogy kell az IRQ-t a *setserial* programmal beállítani.
Marc Merlin marc_bts@valinux.com

Tud-e a StarOffice EPS-t importálni?

Red Hat 7.1-et használok 933 MHz Petium III-as gépen. A gondom az, hogy mind a StarOffice 5.2, mind a 6.0-beta szövegként és nem grafikaként olvassa be a beágyazott PostScript-állományokat (*.ps* vagy *.eps* fájlokat). Ez történik mindegyik összetevőben (például szöveges dokumentum, bemutató, rajz vagy ábra). A `gv` a grafikát helyesen jeleníti meg. A kérdésem az, hogy képes-e a StarOffice PostScript formátumú grafikát beolvasni és megjeleníteni? Ha igen, hogyan?
John C. Burgess, burgess@wiliki.eng.hawaii.edu

Az *Insert/Graphics/From File* (Beszúrás/Grafika/Fájlból) menü kezeli az *.eps* fájlokat.
Scott Maxwell maxwell@ScottMaxwell.org

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

A terjeszési szerződésekkel kapcsolatos fejlemények

Néhány hónapja a Microsoft közösségi (shared-source) szerződésének veszélyeiről írtam (Linuxvilág, 2001. december, 20. oldal), trójai falónak nevezve azt. Ugyanis pusztán azáltal, hogy megtekintjük a Microsoft kódját, „megfertőzhetjük” a saját programunkat, kitéve magunkat a Microsoft szerzői jogi perének.

Egy olvasói levél szerint kettős mércét alkalmazok, mivel pontosan ugyanez a gond a GPL-engedéllyel is. „A hagyományos, szerzői jogdíjas programokat fejlesztő cégeket – írja – a GPL-kód puszta megtekintése ugyanabba a helyzetbe hozhatja, mintha a Microsoft felhasználási szerződésével terjesztett kódot nézték volna meg”. Ami még roszszabb, hogy mint állítja, a GPL „fertőző” záradékai megkövetelik, hogy az egész kereskedelmi terméket a GPL alatt adják ki, vagy a GPL-részeket távolítsák el. Az olvasónak a GPL-ről adott elemzése viszont csak részben helytálló, ugyanis három különböző lehetőséget kever össze.

1. Tegyük fel, hogy egy kereskedelmi programokat fejlesztő cég GPL-engedéllyel terjesztett kódot vesz át és épít be egy programjába. A cég elfogadta a GPL-szerződés feltételeit, és követnie kell azokat. A bíróság jóvátételként kötelezheti a céget, hogy a szerződés feltételeit tartsa be és a kereskedelmi programot – a forráskódot is beleértve – a GPL-szerződés alatt adja ki; jellemzően ezt nevezik „GPL-fertőzés”-nek, ám ebben az esetben a cég tudatosan vállalta a kockázatot.
2. Tegyük fel, hogy a kereskedelmi programokat fejlesztő cég nem fogadja el a GPL feltételeit, ugyanakkor a GPL-kódot felhasználja a saját kereskedelmi programjában. A szerzői jog értelmében a céget kártérítés fizetésére kötelezhetik és arra, hogy hagyjon fel a kód használatával, de a GPL feltételeinek betartására (például a forráskód közzétételére) valószínűleg nem lenne kötelezhető. Ez nem GPL-fertőzés, hanem egyszerűen a szerzői jogok megsértése.
3. Tegyük fel, hogy a kereskedelmi programokat fejlesztő cég egyik alkalmazottja cége tudta és engedélye nélkül, szándékosan vagy más módon GPL-kódot épít be a kereskedelmi programba (amennyiben a cselekedet szándékos, jogi szempontból az alkalmazott „saját felelősségére cselekedett”). Ebben az esetben a cég valószínűleg nem vonható felelősségre a szerzői jog szándékos megsértéséért, noha a jogsértő program használatát be kell szüntetnie. Ilyenkor szó sincs fertőzésről, pusztán a szerzői jog megsértéséről.

A jogdíjas programok fejlesztőinek valóban körültekintően kell eljárniuk. Valaki más szerzői jog által védett programjának felhasználása a saját termékükben (még akkor is, ha nem szándékosan történik), nem kívánt következményekkel járhat – például költséges szerzői jogi perekhez, nagy összegű kártérítésekhez, illetve a jogsértő program terjesztését és árusítását tiltó végzéshez vezethet.

Minden programkészítő cégnek ki kell alakítania a biztonságos fejlesztés gyakorlatát. Ez azt is magában foglalja, hogy meggyőződünk róla, minden fejlesztő tisztában van-e vele, mennyire fontos, hogy ne másolja le valaki más programját, mielőtt egy megfelelően képzett jogász segítségével meg ne vizsgálnák azokat a feltételeket, amelyek mellett a programot beszerezték. Ha egy cég meg akarja őrizni programjainak kereskedelmi jellegét, vigyáznia kell, hogy más kereskedelmi programok, valamint a szabad és nyílt forráskódú programok terjesztési engedélyének hatásait elkerülje. A megfelelő biztonsági intézkedések megtétele, beleértve a vezetőknek az alkalmazottak képzésére és a munkakörnyezet biztosítására fordított idejét csakúgy, mint az engedélyek megvizsgálására fordított ügyvédi munkaidőt is, része az üzleti költségeknek, amelyeket a program árába bele kell építeni.

Ezek a megfontolások érvényesek azokra is, akik szabad és nyílt forráskódú programokat fejlesztenek. Pusztán attól, hogy egy programot ingyen terjesztenek, még nem zárható ki, hogy egy szerzői jogi per egyik pillanatról a másikra véget vet a fejlesztésnek.

Íme néhány biztonsági intézkedés, amelyet az ügyfeleimnek javasolni szoktam: szerezzünk be aláírt szerzői jogi nyilatkozatot vagy valódi terjesztési engedélyt minden harmadik féltől, aki hozzájárul a fejlesztéshez. A nyilatkozat megfogalmazása legyen ehhez hasonló: „Alulírott szerző(k) kijelenti(k), hogy a programkód eredeti és a saját alkotás”.

Ha a programkódot egy másik cég alkalmazottja szolgáltatta, a cég írásos engedélyének beszerzése javasolt.

A Szabad Szoftver Alapítvány egy Alkalmazotti Jogi Nyilatkozat használatát javasolja, amely az alkalmazottat felhatalmazza, hogy a kódot „a szabad programok eljárása szerinti terjesztésre és megosztásra” ajánlja fel.

Ha alkalmazottaink megismerhettek harmadik fél birtokában lévő programot, amely kereskedelmi terjesztésű, és amelynek forráskódja nem másolható, lehet, hogy ésszerűbb az alkalmazottakat egy másik fejlesztéshez beosztani, ahelyett, hogy szerzői jogi (vagy üzleti titkok eltulajdonítása elleni) pert kockáztatnánk.

Jogászként az a kötelességem, hogy körültekintő legyek és felhívjam a figyelmet a kockázatokra. A szabad és nyílt forráskódú programok elkötelezettjeként azonban tartózkodni szeretnék attól, hogy felesleges félelmet, bizonytalanságot és kétségeket ébresszek.

A nyílt forráskódú fejlesztés célja az, hogy megkönnyítse a kód elterjedését és megszüntesse a titkolózást. A kereskedelmi programok fejlesztőinek ugyan vigyázniuk kell arra, hogy ne nézzenek bele más cégek kódjába, de a szabad és nyílt forráskódú programok fejlesztői nyugodtan másolhatják más szabad és nyílt forráskódú programok részeit – a szerző terjesztési engedélye által megszabott kereteken belül. Az eredmény: mindenki számára jobb programkód.

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


Linuxvilág
www.linuxvilag.hu

A webalkalmazások új nemzedékének lelke: az XSLT

Cameron bemutatja az Extensible Stylesheet Language for Tranformations (XSLT) nyelvet, és elmondja, miért is olyan fontos téma ez manapság.

Az Extensible Stylesheet Language for Transformations (XSLT – bővíthető stíluslapnyelv átalakításokhoz) olyan számítási nyelv, amelyet kifejezetten két XML-dokumentum közötti átalakításra fejlesztettek ki. Az XSLT ismertetése nem kis feladat. A legfőbb nehézséget a változatosság okozza: az XSLT-nek számos alkalmazási köre van, rengeteg XSLT-motor létezik és egyetlen XSLT-alkalmazás több együttműködő módszert is alkalmaz. Ezért meglehetősen fontos, hogy csak a valóban lényeges részekre összpontosítsunk.

A mindenre használható XML

Az első lényeges tudnivaló az XSLT-vel kapcsolatban, hogy Extensible Markup Language- (XML) alapú (lásd még az ***XSLT-fogalmak gyűjteménye*** című széljegyzetet). Az XML olyan egyetemes adatformátum, amelyet úgy terveztek, hogy bármilyen típusú adatot képes legyen tárolni: eljárásadatokat, programokat és dokumentumokat – kezdve a vásárlási szabályoktól a bibliafordításokig, bármilyen emberi nyelven, bármilyen számítógépes rendszeren és operációs rendszeren. Az XML úgy néz ki, mint a HTML, csak éppen némileg összetettebb. Valójában az XML egyik tervezési célja éppen a HTML általánosítása volt oly módon, hogy eközben a HTML-szakértők kényelemérzete megmaradjon. Létezik egy XHTML-nek nevezett külön XML-változat is, amely közvetlenül HTML-ként értelmezhető. A Linuxvilág gyakorta jelentet meg olyan cikkeket, amelyek valamilyen szinten kapcsolatosak a XML nyelvvel.

Egy tökéletesen XML-esített világ sok szempontból egyszerűbb lenne. A tartozás-nyilvántartás működésének vizsgálatánál például nem kellene tudnunk, hogy ki kihez tartozik, ki ment el háromhetes vakációra, és a többi zavaros emberi részletet. Ha fel tudunk rajzolni egy diagrammot, amely a beérkezett számlákat és kimenő kifizetéseket ábrázolja – a folyamat során valószínűleg egyre szaporodó hitelesítő bejegyzésekkel megtűzdelve –, akkor sikerült a lényegi adatot levezetnünk. Mámorító látomás. Azt sugallja, hogy az a rendszer, amely az egyik XML-dokumentumot (például számla) képes egy vagy több másik XML-dokumentummá átalakítani (fizetési csekk, hitelesítő bejegyzés), az önműködően megszervezi, sőt akár meg is oldja az összes lényeges szervezési feladatot. Ez az, amiért manapság az XSLT olyannyira érdekes.

Azok az olvasók, akik rendelkeznek XML-világbeli tapasztalatokkal, projekttapasztalataikat általánosíthatják, és máris fogalmat alkothatnak az XSLT valódi értékéről. Bárki, aki a gyakorlatban is foglalkozott az XML-lel, tudja, hogy valójában a megoldás kezdetét jelenti, nem pedig valamiféle csodaszert, mint amilyennek a reklámszövegek beállítják. Az XSLT-vel pontosan ugyanez a helyzet: hasznos és elég hatékony módja a termelésben használt alkalmazások összeállításának megszervezésében. A XML-dokumentumok átalakításának alapötlete igen fontos dolog – hogy belássuk, valóban működő ötletről van szó, közelebbi pillantást kell vetnünk a technikai részletekre.

example1.xsl

```
<xsl:stylesheet
     ➥xmlns:xsl =
➥"http://www.w3.org/1999/XSL/Transform"
     ➥version = "1.0">
  <xsl:output method="html"/>

  <xsl:template match = "/">
     <html>
          <body>
             <xsl:apply-templates select =
             ➥"datum"/>
          </body>
     </html>
  </xsl:template>

  <xsl:template match = "datum">
     <h1>
          <xsl:value-of select = "."/>
     </h1>
  </xsl:template>
</xsl:stylesheet>
```

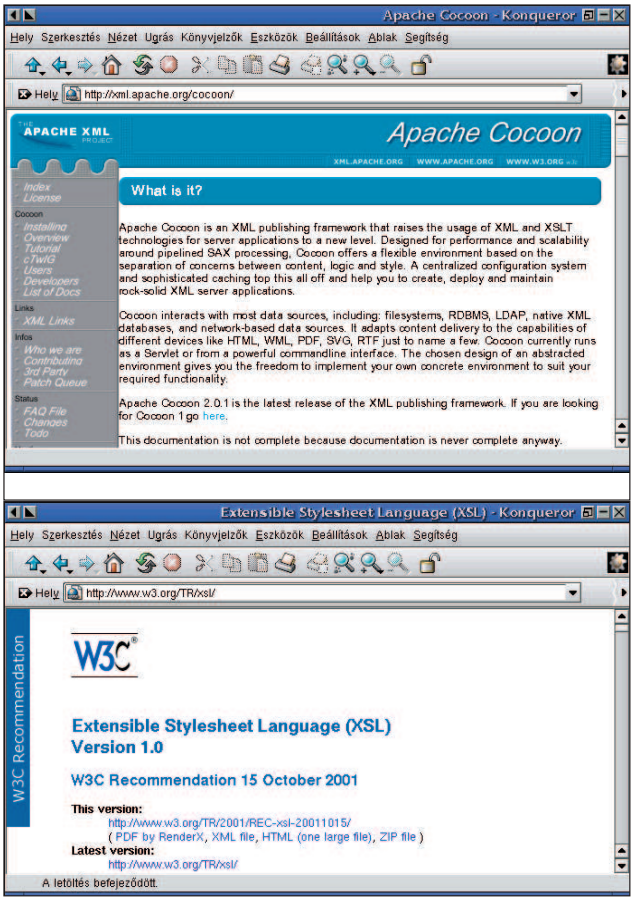
Saját motorunk

Ahhoz, hogy XSLT-utazásunkon elindulhassunk, és könnyebben ráérezhessünk a nyelv apró finomságaira, egy saját motorra, azaz egy nyelvfeldolgozóra (language processor) lesz szükségünk. A legszélesebb körben elterjedt eszköz Java-alapú, illetve üzleti. Ezeket gyakran nagyobb kiszolgálótermékekbe építve találjuk: adatbáziskiszolgálókban, alkalmazáskiszolgálókban és így tovább.

A példákat a fentiek helyett írásunkban a tDOM-motor meghatározásai alapján mutatjuk be. A tDOM több előnnyel is rendelkezik, amelyek között a legfontosabbak: a szabad, nyílt forrás felhasználási szerződés hatálya alá tartozik, kivételesen memóriatakarékos, és méréseink szerint kétszer olyan gyors, mint a versenytárs XSLT-motorok. Telepítése gyors és tömör, ezenkívül parancsfájlosítható (scriptable) parancsmóddal rendelkezik, amelyen keresztül kényelmesen utasíthatjuk tDOM-ot. Továbbá kitűnően illeszkedik az alább leírt kétszintű programozási stílushoz, ugyanakkor elég üzembiztos ahhoz, hogy több igényes honlap is alkalmazza a termelésben.

Saját tDOM-példányunk elkészítéséhez olvassuk el a ***Hogyan kezdjünk XSLT-t programozni*** széljegyzetet. Ez a cikk ugyanis a XSLT használatának első példájával zárul, amelyet a következőképpen hívhatunk meg:

```
tclsh8.3 xslt.tcl example1.xml example1.xsl
➥example1.html
```



Ez a parancssor a következőkre utasít: „használd a 8.3-as változatú Tcl-értelmezőt az ***xslt.tcl*** program indításához. Az ***xslt.tcl***-eszköz alkalmazza az ***example1.xsl*** stíluslapot az ***example1.xml*** dokumentumra, és kimenetként előállítja az ***example1.html***-t".
Figyeljük meg, ahogy a gép bemenetként átveszi az ***example1.xml***-t, majd előállítja a mindössze néhány soros ***example1.html***-t:

```
<?xml version="1.0"?>
<datum>elsı  zenet </datum>
```

Gondoljunk az ***example1.html***-re mint a fenti sor helyesen formált HTML-kiterjesztésére:

```
<html><body><h1>elsı  zenet </h1></body></html>
```

Az XML mint adat és kód

Ha mindössze az ***example.html***-hez hasonló egyszerű HTML-dokumentum előállítása a célunk, az XSLT elsajátítása helyett közvetlenül valamilyen egyszerű makrónyelvet is megírhatunk vagy használhatunk. Az XSLT igazi értéke akkor nyilvánul meg, amikor összetettebb feladatokat oldunk meg.
Az XSLT-átalakítást beállíthatjuk, hogy az ***example1.html*** kimenetet az adott stílusban készítse el, esetleg az adott betűkészlettel vagy szabványos honlaphivatkozásokat és -nyilatkozatokat alkalmazva.
Az XSLT ezen átalakítások végrehajtásához a stíluslapok nyelvét alkalmazza. Bár a stíluslapok már az XSLT bevezetése előtt is használatban voltak, ebben a cikkben minden más felhasználási területet figyelmen kívül hagyva, az XSLT-stíluslapokat következetesen stíluslapoknak fogjuk nevezni.
Bizonyos szempontból a stíluslap program. Ahogy az

```
int main()
{
    puts("Hello.");
}
```

egy C program forráskódja, a stíluslap az XSLT program forrása. A stíluslapok különlegessége, hogy maguk is XML-dokumentumok. A hagyományos számítógépprogram-látvány helyett (mondjuk ahogy a C, a Java és a `ksh` kinéz), az XSLT-forrás egyfajta kulcsszószöveg (markup text, lásd az *1. listát*). Az XML-re olyannyira jellemző bőbeszédűséggel ez nagyjából a következőkre utasít: „működj úgy, mint egy program, amely beolvassa a `<datum>`-elemeket, majd a tartalmukat egy helyesen megformált HTML-dokumentum `<h1>` fejlécei közé helyezi." Így készül az ***example1.html***.
Az XSLT-feldolgozást megvalósító alkalmazás maga egy Tcl

Hogyan kezdjünk XSLT-ben programozni

A tDOM a Tcl teljes telepítését igényli. A 8.3.4-es vagy újabb változat használatát javaslom. A legbiztonságosabb a forrásából telepíteni. A telepítés általában a következő lépésekből áll:

- A forrás telepítőkészlet letöltése, például a következő helyről: ➲ http://prdownloads.sourceforge.net/tcl/tcl8.3.4.tar.gz.
- A források kicsomagolása.
- Végül a hagyományos `autoconf`-alapú összeépítés lépései:

```
cd tcl8.3.4/unix
./configure
make
make install
```

A tDOM helyes telepítése nagyjából ugyanezt a sémát követi:

- Letöltés: ➲ http://phaseit.net/binaries/tDOM-0.63.tar.gz.
- Kicsomagolás.
- Előállítás:

```
cd tDOM-0.63/unix
./configure
make
make install
```

Az e cikkre vonatkozó források a ➲ http://phaseit.net/examples/xslt.zip címen találhatók. Töltsük le és telepítsük őket a munkakönyvtárunkba. Ettől kezdve a következő parancsot már meg tudjuk hívni:

```
tclsh8.3 xslt.tcl example1.xml example1.xsl
➥example.html
```

Figyeljük meg, hogy a fenti parancs kimenete az *example.html* lesz. Bár linuxos végrehajtható állományok is letölthetők, általában elmondható, hogy a legtöbb Unix Tcl programozó a forrásból szeret dolgozni. Ennek aztán az a következménye, hogy a forrásterjesztések általában egy kicsit kifinomultabbak. Különösen igaz ez a tDOM esetében.
A Windows esetében viszont pont fordítva áll a helyzet. A tDOM kitűnően működik Windows alatt is, megeshet azonban, hogy szükségünk is lesz rá, de ennél az operációs rendszernél a bináris terjesztések telepítése az általános. A legfrissebb adatokért keresd a ➲ http://mini.net/tcl/tDOM címet.

XSLT-fogalmak gyűjteménye

API – Application Programming Interface, azaz alkalmazásprogramozási felület.

Cocoon – Java- és XSLT-központú terjesztő keretrendszer az Apache-hoz.

CSS – Cascading Stylesheet, vagyis kaszkádolt összerendelt stíluslap. A jelkulcsok (markup elements), például hivatkozások összerendelése tulajdonságokkal, főleg megjelenítési tulajdonságokkal (mint például „rajzold kékkel"). A legtöbb böngészővel dolgozó ember találkozott már CSS-ekkel.

DOM – Document Object Model, azaz dokumentum objektummodell. A DOM egy dokumentumokhoz szánt programozási API, amely például az XML osztályrendszerű faszerkezetét emeli ki.

DTD – Document Type Definition, azaz dokumentum-típusmeghatározás. A DTD olyan metanyelv, amely az XML dokumentumszótárát határozza meg. Az XML Schemata általánosabb keretrendszerben szintén ugyanezt teszi.

FO – Objektumok formázása. A FO-k az XML és a megjelenítés közötti átalakítást írják le. Tervrajzadatokat tartalmaznak.

FOP – objektumokat pdf-be (és általánosabb esetben más megjelenítési formátumokba) formázó átalakító.

Funkcionális programozás – a Lisp, Haskell és Erlang nyelveket gyakran nevezik funkcionális programozási nyelvnek, mivel megváltoztathatatlan változókat alkalmaznak, mellékhatás nélküli műveletek, rekurzió és provability jellemzik őket. Az XSLT funkcionális és nem eljárásokból építkező programozási nyelv, ellenben a C, Java, vagy Visual Basic nyelvekkel.

HTML – HyperText Markup Language. A Világhálózat általános nyelve.

Névtér (namespace) – programnyelvi elv a többértelműség kizárására. A Baseball és Biology névterek megkülönböztetésével egyértelműsíthetjük, mikor gondolunk `Baseball::bat`-re és mikor a `Biology::bat`-re (angolul a „bat" szó denevért és ütőt is jelent). Az XML-fejlesztők számára ennek ott van jelentősége, hogy alkalmazásaik fejlesztése közben nem kell félniük attól, hogy változóneveik és más nevek véletlenül ütik egymást.

SAX – Simple API for XML, azaz egyszerű XML API. A DOM kiegészítésére szánt API, eseményközpontú, és az XML-t mint karakterek folyamát szemléli.

Schema – metanyelv-előírás. Az XML Schema például olyan szabályokat ad meg, amelyek az XML-dokumentum tartalmát szabályozzák.

Scripted document – olyan fájl, amely adatot és az adaton műveleteket végző kódot egyaránt tartalmaz.

Stíluslap (stylesheet) – a dokumentum értelmezésének vagy átalakításának leírása. Az XSLT-stíluslapok érdekessége, hogy maguk is XML-dokumentumok.

Vocabulary XML – az XML-szótár tulajdonképpen metanyelv, amelyben lehetőség nyílik adott témakörökhöz tartozó megvalósítások vagy szótárak leírására. Léteznek testreszabott XML-szótárak matematikához, autóeladáshoz, a Gnome GUI-felülethez és sok más dologhoz.

W3C – az XML-szabványokat sok más tevékenység mellett a World Wide Web Consortium adja ki.

Xalan – A Xalan egy Java-központú XSLT-motor, az Apache Project része.

XML– Extensible Markup Language. A HTML nyelvhez hasonló kinézetű nyelvre kell gondolni, amely azonban elvben bármilyen digitális adatot képes kódolni.

XPath – olyan nyelv, amely az XML-dokumentum egy részét azonosítja vagy címzi meg. Gondolhatunk rá lekérdező-nyelvként, az XSLT kiegészítéseként (az XSLT a változásokat írja le, az XPath azt mutatja meg, hol történjenek ezek a változások).

XSL – Extensible Stylesheet Language, vagyis bővíthető stíluslapnyelv.

XSLT – Extensible Stylesheet Language for Transformations, azaz átalakításokhoz szánt bővíthető stíluslapnyelv. Olyan nyelv, amelyet XML-források más XML-forrásokká alakításához fejlesztettek ki.

XSP – eXtensible Server Pages a Cocoon egyik képessége, melynek segítségével dinamikusan készíthetünk XML-alapú honlapokat.

program. Ezért itt az ideje, hogy pár alapvető dolgot a Tcl-ről is megtudjunk. A tDOM XSLT-motorját Tcl-csatolások valósítják meg, és a ***xslt.tcl*** parancsfájl a XML-dokumentumok fájlneveit egyszerűen parancssori változókként kéri be, majd továbbadja a motornak.
Vizsgáljuk meg újra a példahívásunkat!

```
tclsh8.3 xslt.tcl example1.xml example1.xsl
➥example1.html
```

A ***tclsh8.3*** az elindított futtatható program neve, és a ***xslt.tcl*** az a legkisebb Tcl-parancsfájl, amely a tDOM XSLT-motorját előhozza. Ha egy kicsit tovább szeretnénk fejleszteni az eszköz hibakezelő képességét, a legkézenfekvőbb kezdés a ***xslt.tcl*** újratervezése.
A ***xslt.tcl*** futtatása elindítja a XSLT-feldolgozót, amely három fájlnevet kap meg. Az ***example1.xml*** fájl a felhasznált példa XML-dokumentumforrás. A stíluslapot ezekkel a fájlnevekkel alkalmazzuk az ***example1.xml***-re. A folyamat az eredménydokumentumot az ***example1.html*** fájlba írja. Válasszunk más logikai tartalmat, azaz másik XML-forrást az ***example1.html***-hez. Legyen például az ***example2.xml***. A kimenet stílusának megváltoztatásához viszont az ***example1.xsl***-t kell újraírnunk.

Sikeresen lefuttatunk egy XSLT-programot. Már csak az maradt hátra, hogy megismerjük az XSLT nyelvet, illetve megtudjuk, hogyan is alkalmazhatjuk valós feladatok megoldásában. Mielőtt komolyabban belemerülnénk az XSLT szintaxisába és szemantikájába, vessünk egy rövid pillantást a felhasználási területekre!

Egy nyelv – sok alkalmazás

Képzeljük el, hogy egy több tízezer lapot tartalmazó webhelyért felelünk. Ezeket a lapokat szervezetfüggő XML-szótár szerint tároljuk, amely kiszedegeti a formázási adatokat és a HTML-szemetet – dokumentumaink kizárólag az adott laphoz tartozó logikai adatokat tartalmazzák. A látogatóknak természetesen HTML-re van szükségük, de ezeket dinamikusan állítjuk elő, szabványos fejléccel, keretekkel, vezérlőelemekkel, lábjegyzetekkel és minden egyéb a Weben elvárható díszítőelemmel. Az XSLT lehetővé teszi, hogy az összes lap stílusát egyszerre változtassuk meg. Továbbá szépen megosztja a munkát az XML-tartalmú fájlok és a XSLT-stíluslapok közt, így a különböző szakemberek hatékonyan tudnak együttműködni. Ez a döntéshozói szintű leírás eléggé sokat elrejt a megvalósítás változatosságából. Hol és mikor történik az XSLT-átalakítás? Lehet egy XML-dokumentumokból álló hátterünk, amit aztán időnként parancssoros XSLT-feldolgozó programmal hagyomá-

nyos webkiszolgálónak szánt statikus HTML-dokumentumokká alakítunk. Tarthatjuk az XML-forrásokat adatbázisban is, ahonnan vagy XML-ként kapjuk vissza és HTML-é alakítjuk át őket, vagy rögtön teljes értékű HTTP-oldalként kérjük le. Ezeket a felületeket különféle alkalmazáskiszolgálók, tartalomkezelők és XML-adatbázisok teszik elérhetővé. Még egy változat: csak a forrásokat tartjuk a kiszolgálón, majd HTML-kiterjesztések és böngésző megfelelő együttesével magát a böngészőt utasítjuk, hogy értelmezze a megkapott XSLT-t. Valamennyi lépést tetszés szerinti mértékben tehetjük dinamikussá, egészíthetjük ki gyorstárazással a sebességnövelés érdekében, böngésző vagy olvasó szerinti testreszabhatósággal és így tovább. Az alkalmazások ilyen sokszínűsége következtében a kiadók műveit olvasni igazi kihívás. Valamennyien különböző stílusú Java-programozást alkalmazunk attól függően, hogy éppen appleteken, serveleteken vagy babokon dolgozunk-e, annak ellenére, hogy ezek közül bármelyikre ráragaszthatnánk a javás webprogram címkét. Hasonlóképpen azt is fontos megérteni, milyen más XSLT-feldolgozási lehetőségeket nyújtanak a különféle termékek.

Összetett honlapfejlesztés

Neil Madden, a University of Nottingham hallgatója egy különlegesen gyors telepítésre és karbantartásra kiélezett XSLT-rendszerrel rendelkezik. Az elképzelése több részből álló webhelyen alapul, amelyet rendszergazdák csapata, szerkesztők és felhasználók alkalmaznak. A TclKitet, ezt az újszerű, nyílt forrású eszközt használja, amely egyetlen különösen pehelysúlyú, kis munkaigényű csomagban egyesíti az adatbázis- és a HTML-szolgáltatásokat. A TclKit Tcl-programokat is képes értelmezni, így a szabványos sablonokkal kiegészített tDOM-ot programozható modulokba csomagolhatja. Ezekkel látott hozzá a webhely fejlesztésének:

1. XML-dokumentum tervezése, amely képes tárolni a honlap adatait.
2. XSL-stíluslapok elkészítése, hogy az átalakított adatok minden ügyfél igényeit kielégítsék.
3. Az első két lépés ismétlése minden olyan részre, amely különleges igényeket támaszt.
4. Felhasználók, részek és lapok felvitele.

Ezeket a különböző adattípuscsomagokat (lapszerkezet, XML-források, stíluslapok) a parancsfájlosított dokumentumok foglalják magukban, és teszik egyszerűvé egy működő kiszolgáló új kiszolgálóra, lemezrészre telepítését vagy frissítését. Madden tervei szerint a tisztán webalapú szerkesztést kiváltására egy gazdagabb, gyorsabb GUI-felület is készül. A Tcl egységessége és parancsfájlosíthatósága ezt a kettős portolást akár webkiszolgálón, akár helyi GUI-felületen keresztül is egyszerűvé teszi.

A jól meghatározott modulhatárok a rendszer szempontjából létfontosságúak. A tervezők a stíluslapokkal foglalkoznak, a rendszergazdák a jogosultságokat kezelik, a szerkesztők az egyes részeket rakják össze ütközés nélkül. Azáltal, hogy ezeket a szolgáltatásokat a megbízható összetevőket egymáshoz ragasztó apró parancsfájlokként valósítjuk meg, elég könnyen adhatunk a rendszerhez új alkalmazásokat. Madden középtávú céljai közt a Wiki együttműködési hírcsoport, illetve a magas szintű bemutató (presentation) kimenet készítésére alkalmas XSP- és FOP-modulok elkészítése szerepel. Madden büszkén hasonlítja rendszerét a Cocoonhoz, a közismert Apache- és Java-alapú XML-terjesztési keretrendszerhez. Rendszere több szempontból is túlszárnyalja a Cocoon teljesítményét, a forráskódja viszont a töredéke annak.

Egy másik termelési példa a tDOM XSLT használatára ***George J. Schlitz*** MediaOne programja. Pénzügyi dokumentumokat készít XSLT-vel igen fontos (mission-critical) webkörnyezetben. Bár a terjesztést eredetileg a Xalannal kezdte, a teljesítménykövetelmények végül rákényszerítették, hogy a tDOM-ra álljon át. Mindezen felhasználásoknál alapvető fontosságú az XML-kódolt vagy XML-kódolható adatok felkutatása. Csevegőszoba-naplófájlok, hitelesítő tanúsítványok, nyomtató folyamatok, újságfotók, képernyőbeállítások, családfafeljegyzések, játékállások, alkalmazástervek, földrészleteladások, orvosi fájlok és még sok-sok egyéb adat lehet jelölt az XML-elesítésre. Ha már egyszer ebben a formátumban vannak, az XSLT-átalakítás általában a legmegbízhatóbb és -méretezhetőbb módszer az adatok egyedi felhasználáshoz történő tálalására.

Az XSLT elsajátítása

Van még mit tanulnunk az XSLT-ről mint szakmáról. Amilyen ütemben a felhasználási területe növekszik, sokkal kevesebb XSLT-ben jártas hozzáértő programozó akad, mint mondjuk object pascalos.

Az XSLT gyors terjedésének másik akadálya a kényelmetlen XML-alapú írásmód és zavaró telepítés mellett szolgáltatott vagy alkalmazott szemantikája. A legtöbb számítási nyelv, ami a Linuxvilág oldalain megjelenik, többé-kevésbé eljáráselvű (procedurális): a Java és a C programok a processzort valamilyen művelet elvégzésére utasítják, majd újabb és újabb utasításokat adnak. Az eljáráselvűség a számítás miként jében is megmutatkozik.

XSLT-tanulmány

Bár a Világhálón bőséges találhatunk anyagokat, az XSLT-ről a nyomtatott könyvek értékesebbek azok számára, akik az XSLT-t mélyebben is meg szeretnék ismerni. *Steve Holzner* Inside XSLT című műve gazdag receptgyűjteményt tartalmaz, miközben szigorú pontossággal ismerteti az XSLT kulcselgondolásait. Egyéb haszna mellett az Inside XSLT utolsó fejezetei az objektumformázásról szólnak. Az utóbbi pár hónapban igen nagy érdeklődést tapasztaltam a pdf és a hozzátartozó megjelenítési kimenetek XSLT segítségével történő előállítása kapcsán.

Ugyancsak hasznos *Doug Tidwell* XSLT könyve, *Michael H. Kay* XSLT Programmer's Reference, illetve *Khun Yee Fung* XSLT: a Working with XML and HTML című alkotása. Az XSLT majdnem teljes hivatkozási függelékkel rendelkezik, és kifejezetten jól írja le a kiterjesztés módszer (extension mechanism) működését. Az XSLT: Working with XML and HTML nagy népszerűségre tett szert azok körében, akik komolyabb grafikai vagy webes háttérrel rendelkeztek, és akik nem igazán tartják magukat programozóknak, de XSLT-projektjüket gyorsan el szeretnék indítani. Az XSLT Programmer's Reference a másik véglet: meglehetősen szabatos az elvonatkoztatások terén, ugyanakkor laza előadásmód jellemzi, emellett részletes és összefogott. Bár van egy olyan érzésem, hogy igen sok XSLT-programozó manapság a Programmer's Reference-szel kezdi XSLT tanulmányait, nem árt tudni, hogy ez kézikönyv (reference), és nem oktatókönyv (tutorial).

Az XSLT megértéséhez az utolsó ajánlatom az ActiveState Tools Corporation, Komodo IDE-je. A Komodo 0-tól körülbelül 300 dollárig terjedő felhasználási díjért cserébe nagyon hasznos XSLT hibakereső berendezést kínál, amelyhez hasonlót még egyetlen termékben sem láttam.

Az XSLT szolgáltatásait tekintve a Lisppel mutat rokonságot. A jó XSLT-programok a kívánt eredmény „lényegét” mutatják be. Egy időbeli folyamatra való összpontosítás helyett az XSLT a kívánt eredmény elérése érdekében a teljes XML-dokumentumon vagy annak jól meghatározott részein dolgozik. Ezt funkcionális megoldásnak nevezzük (a matematikai modellt idézve fel), ahol a függvények a bemenetet – mellékhatások vagy időbeliség nélkül – kimenetté alakítják. Továbbá a matematikai függvényeket különböző kombinációkban is összerakhatjuk. Az általános XSLT-szemantika jó néhány különböző átalakítást vezet be, időbeliségük megadása nélkül. A stíluslapok egyszerre alkalmazandók.
Az XSLT rendelkezik ugyan változókkal, de azok megváltoztathatatlanok (inkább feltételesen meghatározott konstansok, nem pedig változók – a ford.). Csakis egyetlen értéket vehetnek fel, és nem lehet őket például ciklusba szervezni, mint ahogyan a

```
for (i = 0; i < 10; i++);
```

utasításban tesszük.
A paraméteres vagy ismétléses műveletek közvetlenül megadott rekurziókon és ismétlődéseken (iteration) keresztül hajtódnak végre. Az XSLT változóhasználati írásmódja meglehetősen csúnya, mivel meg kell felelnie az XML kötöttségeinek. C-ben vagy Javában azt írnánk, hogy

```
if (level > 20)
     code = 3;
else
     code = 5;
```

(Egyszerűbb írásmódja a `code = (level > 20) ? 3 : 5` – a ford.)

Ugyanehhez XSLT-ben körülbelül a következőket gépelnénk be:

```
<xsl:variable name = "code">
      <xsl:choose>
           <xsl:when test = "$level &gt; 20">
                <xsl:text>3</xsl:text>
           </xsl:when>
           <xsl:otherwise>
                <xsl:text>5</xsl:text>
           </xsl:otherwise>
      </xsl:choose
</xsl:variable>
```

Mint a fenti példa is bizonyítja, bár az XSLT önmagában is elég hatékony az általános feladatok megoldásában, a legjobban mégis kettős programozási módban használhatjuk. Az XSLT igazi ereje inkább a sablonok feldolgozásában, a mintaazonosításban (pattern matching), illetve az XML-elemek rendezésében és csoportosításában rejlik. ***Steve Ball***, a Zveno Pty. Ltd. vezető tanácsadója csak annyit dolgozik XSLT-ben, amennyit még célszerűnek érez, majd az eredményt egy másik nyelvet használó alkalmazásba illeszti be, hogy a külső rendszerek felületeit kezelhesse – ideértve a fájlrendszert és a felhasználói nézetet. Azok között a fejlesztők között, akikkel az utóbbi fél évben találkoztam, a Java és a Tcl volt a legnépszerűbb, de a Python, a C, a Perl és más partnernyelvek többé-kevésbé ugyancsak megfelelően támogatják az XSLT-motorokat. Továbbá az XSLT olyan kiterjesztést lehetővé tévő módszereket is meghatároz, amelyek segítségével a fejlesztők új szemantikát vihetnek az XSLT-be, ezek: ***extension elements*** (kiterjesztett elemek), ***extension functions*** (kiterjesztett függvények) és ***fallback processing*** (tartalék feldolgozás).
Az XSLT végső sorsa még nem teljesen egyértelmű. Ebben a tekintetben hasonló cipőben jár, mint a Java. Öt évvel ezelőtt úgy tűnt, hogy a Java egyetlen célja ügyes kis vizuális alkalmazások készítése. Azóta kiderült, hogy a nehézsúlyú üzleti kiszolgálók jobb otthont adnak a Java-programozásnak. Egyelőre még korai lenne meghatározni, hol tudjuk az XSLT-t használni. A ReportLab Inc. például olyan üzleti szolgáltató, amely igen komoly minőségkövetelményű leíráskészítéssel járó termékeket és szolgáltatásokat ad el a világ legnagyobb szervezeteinek, például a Fidelity Investmentsnek és az American Insurance Groupnak. A ReportLab alapítója, ***Andy Robinson*** elmesélte nekem csapatának XML-átalakító projektek készítése közben szerzett tapasztalatait. A cégüknél minden befejezett projekt XSLT helyett valamilyen pehelysúlyú parancsnyelvben íródott. Igaz ugyan, hogy az XSLT kifejezetten XML-átalakításra szakosodott, de a tanácsadó csapatok egyszerűbbnek látták az általánosabb célú, de hatékony Python nyelvet használni.

Köszönetnyilvánítás

Különös köszönettel tartozom ***Rolf Ade***-nek, aki bevezetett a tDOM program világába, és segített megérteni.

Cameron Laird
a Phaseit Inc. alelnöke és főállású fejlesztője. Gyakran ír programozással kapcsolatos cikkeket, és az elmúlt évben számos XSLT-vel kapcsolatos cikket adott közre. Jelenleg egy a nyelvet oktató tanfolyam előkészítésén dolgozik.

Kapcsolódó címek

Az XSL, XSLT és Xpath megismerését kezdjük a W3C ajánlásával a ➲ http://www.w3.org/TR/xsl, ➲ http://www.w3.org/TR/xslt és a ➲ http://www.w3.org/TR/xpath címen.

A FO-ról (azaz az objektumformázásról) is találunk itt részleteket ➲ http://www.w3.org/TR/xsl/slice6.html#fo-section

Lars Marius Garshol összegyűjtötte a leginkább elérhető XSLT-motorokat. Listája megtekinthető a ➲ http://www.garshol.priv.no/download/xmltools/cat_ix.html#SC_XSLT címen.

A tDOM-mal kapcsolatos legfrissebb adatokat a ➲ http://mini.net/tcl/tDOM címen találjuk.

A SAX-szal a ➲ http://www.megginson.com/SAX helyen ismerkedhetünk.

A Cocoon adatait az ➲ http://xml.apache.org/cocoon lapon találjuk.

A héjjal ellátott dokumentumokról (scripted documents) a ➲ http://mini.net/tcl/ScriptedDocument címen olvashatunk.

Hiperhivatkozásokat és más, a saját XSLT-projektjeimhez tartozó adatokat, véleményeket találhatnak az alábbi címen: ➲ http://starbase.neosoft.com/~claird/comp.text.xml/XSLT.html

Ügyféloldali parancsfájlok készítése

Marco megmutatja, hogy miként lehetséges böngészés közben az oldalaknak csak azt a részét letölteni, amely valóban érdekel bennünket.

Linuxhoz rengetegféle böngészésre és FTP-re alkalmas eszköz létezik, amelyek szolgáltatásban gazdagok és a felhasználóik minden igényét képesek kielégíteni: kezdve a parancssor megszállottaitól a 3D többmonitoros képernyőfüggőkig. De az ilyen eszközöknek létezik egy nagy hibája: elvárják a felhasználótól, hogy ott üljenek a billentyűzet előtt. Természetesen akadnak olyan eszközök, amelyek egész webhelyet képesek tükrözni, amíg te alszol, ilyen például a `wget` is, azonban ezeknek az eszközöknek először meg kell találnod a megfelelő URL-t, ha pedig az anyagot letöltötték, azt bitről bitre végig kell olvasnod. Kicsi, statikus oldalaknál ez nem gond, de mi történik olyankor, ha egy oldalt minden nap le kell töltened egy véletlenszerű címről? Vagy ha le akarsz tölteni egy 100 K-s dokumentumot, és csak néhány címszó érdekel belőle? Ismerkedj meg az ügyféloldali parancsfájlok készítésével, és minden olyan módszerrel, amellyel lehetővé válik, hogy csak olyan oldalakat vagy oldalrészeket tölts le, amelyek valóban érdekelnek, és a számítógép már előkereste őket. Ilyen parancsfájlokkal csak azokat a közlekedési és időjárási adatokat kell elolvasnod, amelyek a te környezeteddel kapcsolatosak, nem kell feleslegesen érdektelen képeket nézegetned, és még a továbbhaladáshoz szükséges hivatkozásokat is kézhez kapod.

Lényeges adatok a szerzői jogról és a sávszélességről

Amellett, hogy az ügyféloldali parancsfájlok használatával időt takaríthatsz meg, még sok egyebet is meg tanulhatsz, többek között önfegyelemre nevel. Ha az itt leírt módszereket válogatás nélkül alkalmazzuk, az esetleg szerzői jogsértésnek minősülhet, vagy a teljes sávszélességedet felemészthetik, olyannyira, hogy végül akár az internethozzáférésedet is elveszted. Másrészről ez a szabadság csak addig áll fenn, míg a weboldalak magántulajdonnak nem minősülő nyelveken készülnek (HTML, illetve XML), és szabadon hozzáférhető ASCII-ban írják őket. Rengeteg jó weboldal marad életben mindenféle további költség nélkül, ha megfelelő mennyiségű hirdetést töltenek le tőlük, tehát a leírtakat csak meggondoltan szabad alkalmazni.

A lehetőségek

Mielőtt nekikezdenénk megalkotni programunkat, szokás szerint alaposan nézzünk körül, hátha valaki már elkészítette azt, és esetleg az ő munkáját is felhasználhatjuk. Ha a Freshmeat.net keresőjébe beírjuk a „news ticker” szavakat, azonnal 18 találatot kapunk, olyanokat, mint a Kticker, a K.R.S.S – egészen a GkrellM Newstickerig. Ezek mindegyike nagyon jól használható eszköz, de csak híreket töltenek le, és nem működnek anélkül, hogy bizonyos dolgokat meg ne változtatnál bennük. Ezen túlmenően a felsorolt projektek mindegyike valamilyen grafikus eszköz, tehát `cron`-feladatként nem képesek futni, és az sem biztos, hogy a kimenetüket egy másik programnak továbbítani tudod.
Ezen a területen, ha a saját elképzeléseidet szeretnéd megvalósítani, csaknem mindig magadnak kell megalkotnod azt.
Ez okból most mi sem kísérletezünk semmilyen hiánytalan megoldás közreadásával, helyette az általános megközelítést tanulmányozzuk.

Az alapadatok összegyűjtése

```
#! /usr/bin/perl
#20011210

use strict;

use LWP::UserAgent;
use LWP::Simple;
use HTML::Parse;
use HTML::Element;
use URI::URL;
use Image::Grab;

my $HTML_FILE = get($ARGV[0]);
my @HEADER    = head($ARGV[0]);

my @ALL_URLS;
my $PARSED_FILE =
➥HTML::Parse::parse_html($HTML_FILE);
for (@{$PARSED_FILE->extract_links()}) {
        my $LINK = $_->[0];
        my $URL  = new URI::URL $LINK;
        my $FULL_URL = $URL->abs($ARGV[0]);
        push @ALL_URLS, $FULL_URL;
}
```

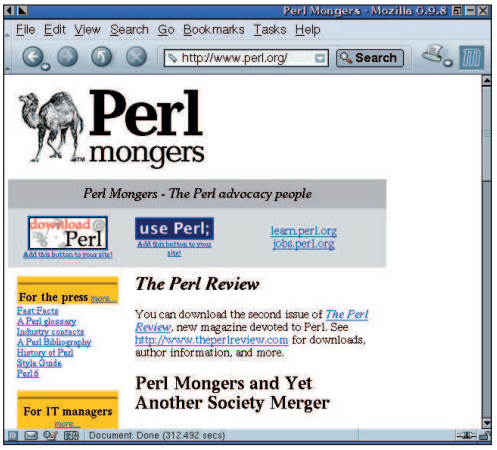
Mire van szükség?

Hogy az e cikkben leírtakból előnyt kovácsolhass, mindössze néhány eszközre lesz szükséged, valamint valamelyest ismerned kell a Perlt, össze kell tudnod állítani egy-két szabályos kifejezést, illetve a következő Perl-modulok használata is elkél: `LWP::UserAgent`, `LWP::Simple`, `HTML::Parse`, `HTML::Element`, `URI::URL` és `Image::Grab`.
A CPAN-ról mindegyiküket ➲ http://www.cpan.org letölthető. Ahhoz, hogy ezeket a modulokat feltelepíthesd, még akkor is, ha nem rendelkezel rendszergazdai jogosultsággal (ez az irodai gépek esetében általános), nem kell mást tenned, mint bemásolni őket egy általad kiválasztott könyvtárba – ahogyan az a Perl leírásában és a kapcsolódó README fájlokban le van írva.
A cikkben leírtakat egy Red Hat Linux 7.2-n próbáltam ki, és amennyiben a kódban szereplő abszolút útvonalakat megváltoztatod, bármilyen Unixon működniük kell, amelyeken a Perl és a szükséges alkalmazások elérhetők.

A szükséges adatok összegyűjtése

A lejjebb leírt feladatok mindegyikéhez és általánosságban véve az ügyféloldali parancsfájlok készítéséhez is szükséges,

hogy kezdésképpen képes legyél néhány weboldalt letölteni és tárolni, amelyeket később elemezhetsz: mikor módosították utoljára, milyen URL-ekre található hivatkozás bennük vagy ezek kombinációja.
Ezek az adatok a webügyfelek elején elhelyezkedő néhány soros programocskákkal mind összegyűjthetők, mint ahogyan az *listában* láthatjuk.
A Perl-parancsfájlok a kötelező `use strict` meghatározással kezdődnek, majd betöltik a szükséges modulokat. Amint ez megtörtént, a webhely teljes tartalmát a `get()` tagfüggvényen keresztül a `$HTML_FILE` változóba mentjük.



A következő műveletekkel a HTTP-fejléc minden sorát egyenként a `@HEADER` tömbbe mentjük. Végül pedig egy tömböt `(@ALL_URLS)` hozunk létre, és egy `for` ciklussal az összes hivatkozást mentjük az oldalról – egyúttal a relatív hivatkozásokat az `abs()` tagfüggvénnyel abszolúttá alakítjuk. Így a ciklus végén az `@ALL_URLS` tömb az összes hivatkozást, amelyet az oldalon talált, tartalmazni fogja.
Egyebek mellett e tagfüggvények teljes leírását megtalálod a ***Web Client Programming*** című könyvben (lásd a Hivatkozások részt).
Miután ezt az anyagot összeszedtük, elkezdhetjük használni. Ha a weboldal tartalmát menteni szeretnéd, az eredeti kódhoz egy `print` utasítást kell hozzáadnod:

```
print $HTML_FILE;
```

Majd futtasd le a héjprogramból:

```
./webscript.pl http://www.fsf.org > fsf.html
```

Ez a parancssor letölti neked a ➲ http://www.fsf.org kezdőoldalát és menti az ***fsf.html*** fájlba. Ne felejtsd el, ha csak ennyire van szükséged, erre a célra a `wget` sokkal jobb eszköz (Lásd a Hivatkozások részt: ***Letöltés böngésző nélkül***).

Képek mentése weboldalról

Ha az abszolút hivatkozások már az `@ALL_URLS` tömbben vannak, a következő `for` ciklussal az összes képet is letölthetjük:

```
foreach my $GRAPHIC_URL (grep
➥/(gif|jpg|png)$/, @ALL_URLS) {
  $GRAPHIC_URL =~ m/([^\/]+)$/;
  my $BASENAME = $1;
  print STDERR "$GRAPHIC_URL elmentØse ide:
➥$BASENAME....\n";
  my $IMG = get ($GRAPHIC_URL);
  open (IMG_FILE, "> $BASENAME") || die
➥"Nem tudtam megnyitni: $BASENAME\n";
  print IMG_FILE $IMG;
  close IMG;
}
```

A ciklus a dokumentumból minden olyan hivatkozást ment, melynek ***.gif***, ***.jpg*** vagy ***.png*** a kiterjesztése (ezeket az eredeti tömb `grep`-elésével kapja meg). A szabályos kifejezés először megtalálja a valódi fájlnevet, melyet az utolsó törtvonal és a hivatkozás vége közül vág ki – ezt a kifejezést lehetne általánosítani úgy, hogy olyan rendszereken is működjön, ahol a könyvtárelválasztó egy fordított perjel.
Az illesztés eredménye a `$BASENAME` változóba kerül, magát a képet pedig a már ismert `get()` tagfüggvénnyel mentjük az `$IMG` változóba. Ezután a fájlt helyileg megnyitjuk, és a változó tartalmát egészében beleírjuk.
Természetesen a legtöbb esetben nincs minden képre szükséged, hiszen a többségük hirdetés, vagy éppen a pillanatnyi oldal logója. Ilyen esetekben ha vetsz egy pillantást az oldal forrására, szinte azonnal meg tudod mondani, hogy melyik képet választod. Tegyük fel, hogy egy mindig változó nevű képre van szükséged, amely azonban három hivatkozásként szerepel. Ebben az esetben a kódot a következőképpen módosítsuk:

```
my $IMG_COUNT  = 0;
my $WANTED_IMG = 3;
foreach my $GRAPHIC_URL (grep
➥/(gif|jpg|png)$/, @ALL_URLS) {
        $IMG_COUNT++;
        next unless ($IMG_COUNT ==
        ➥$WANTED_IMG);
        # a ciklus tovÆbbi rØsze vÆltozatlan
        last if ($IMG_COUNT == $WANTED_IMG);
}
print "A mai napon nincs ilyen fÆjl\n" if
➥($IMG_COUNT != $WANTED_IMG);*
```

Az első utasítás a ciklusban mindig növeli a képszámláló változónkat, míg következő ***next*** utasítás újraindítja a ciklust, amennyiben még nem értünk a megfelelő képhez. A ***last*** utasítással megelőzhetők a felesleges ciklusok, ha a keresett képet már megtaláltuk. A legutolsó sor azt ellenőrzi, hogy találtunk-e megfelelő számú képet – amennyiben nem, hibát jelez.
Ha a kép neve nem teljesen véletlenszerű, még egyszerűbb a dolgunk, mivel a nevére külön is kereshetünk:

```
foreach my $GRAPHIC_URL
➥(grep /(^daily(\d+).jpg)$/, @ALL_URLS) {
```

Ez a ciklus csak olyan képeket keres, melyek a „daily” szóval kezdődnek, tetszőleges számú szám követi őket, és ***.jpg*** a kiterjesztésük.
A két módszer kombinálható, és természetesen egyéb módszerek is elképzelhetők. Ha tudod, hogy a kép neve megegyezik az oldal címével, majd ezt követi a dátum ÉÉÉÉHHNN formátumban, akkor először keressük ki az oldal címét:

```
$HTML_FILE =~ m/<TITLE>([^<]+)<\/TITLE>/;
my $TITLE = $1;
```

Majd számoljuk ki a dátumot:

```
my ($sec, $min, $hour, $day, $month, $year,
➥@dummy) = localtime(time);
$month++;       # a h napok nullÆnÆl kezdıdnek
$year += 1900; # felkØsz lt nk aY2K-ra ;-)))
$TODAY = $year.$month.$day;
```

Végül pedig ez alapján szűrjünk:

```
foreach my $GRAPHIC_URL (grep
➥/(^$TITLE$TODAY.jpg)$/, ALL_URLS) {
```

Jelenítsük meg a szöveg bizonyos részét!

Most kezd csak igazán érdekes lenni a dolog. A legtöbb erőfeszítést és időt igénylő művelet az, ha azt szeretnénk elérni, hogy az oldalnak csak bizonyos része jelenjen meg, mivel ebben az esetben az összes oldal teljes oldalfelépítését elemeznünk kell, és ha valamelyikük szerkezete megváltozik, annak az oldalnak az elemzését elölről kezdhetjük.

Ha lassú az internetkapcsolatod, vagy éppen gyors, de nem akarod lelassítani az MP3-ak és játékok letöltését, bizonyosan megtérül a parancsfájlok írására szánt időd. Mindamellett ha a hozzáférésedet perc alapján számlázzák (mint nekem), még pénzt is megtakaríthatsz!

Az elemezni kívánt HTML-fájlt meg kell nyitnod, és ki kell találnod, hogy milyen szabványos kifejezéssel ollózhatod ki belőle a szükséges elemeket. A Perl ***LWP*** könyvtára alaphelyzetben függvényeket kínál, amelyek segítségével egy HTML-fájlból akár a teljes szövegrészt kiemelheted. Ha a dokumentumnak csak az ASCII-változatára van szükséged, akkor máris elindulhatsz.

Ilyen esetekben rendkívül csábító az ***LWP*** használata, mivel az összes szöveget kivágja a dokumentumból, és ezzel már könnyen dolgozhatsz. Ez a szolgáltatás akkor is jól jöhet, ha az oldalból csupán néhány sorra van szükséged, mivel sokkal egyszerűbb a kivágott szövegen dolgozni, mint a teljes HTML-fájlon. Ez a módszer azonban sok esetben mégis jóval bonyolultabb feldolgozást eredményez. Természetesen a tiszta ASCII-szöveg sokkal könnyebben olvasható, ám a dokumentumból a HTML-jelölések elvesznek, pedig ezekkel könnyebben meghatározható lenne, hol kezdődik az érdekes rész.

Kezdjük mindjárt a legegyszerűbb példával: tegyük fel, hogy csak a hírekre van szükségünk, melyek `<H1>` és `</H1>` tagok között helyezkednek el. Ezeket a tagokat egy szabályos kifejezéssel könnyedén megtalálhatod, nélkülük viszont elég nehezen lehetne rávenni a programot, hogy felismerje a híreket tartalmazó részt.

Hogy valós helyzetben mutassuk be a példánkat, próbáljuk meg az FSF híroldalán ➲ http://www.fsf.org/news/news.html található címeket közvetlenül a saját terminálunkon kinyomtatni. Ha a programunkat elküldjük erre a címre, az oldal teljes tartalmát menteni fogja a `$HTML_FILE` változóba. Most pedig alkalmazzuk a következő szabványos kifejezéseket (javaslom, hogy előtte nézd meg a kérdéses oldalt és a forráskódját, hogy megérthesd, miről is van szó):

```
$HTML_FILE =~ s/.*>Press Releases<//gsmi;
$HTML_FILE =~ s/.*<DL>//gsmi;
$HTML_FILE =~ s/<\/DL>.*$//gsmi;

$HTML_FILE =~ s/<dt>([^<]*)<\/dt>/->
➥$1: /gi;

$HTML_FILE =~ s/<dd><a href=[^>]*>([^<]*)<\/a>/
➥$1 /gsmi;
$HTML_FILE =~
➥s/\.\s+\([^\)]*\.\)<\/dd>/<DD>/gsmi;

$HTML_FILE =~ s/\s+/ /gsmi;
$HTML_FILE =~ s/<DD>/\n/gsmi;
```

Az első három sor minden lényegtelen dolgot levág, ami nem tartozik a hírek közé. A negyedik sor megkeresi a dátumot, és levágja róla a HTML-címkéket. A következő két sor ugyanazt teszi hírek címével. Az utolsó két sor eltávolítja a felesleges szóközöket, és a szükséges helyeken a szöveget új sorokra tördeli. Ezen a 2001. december 14-i napon a héjamban a következő látható (a kimeneten az olvashatóság végett kicsit változtattam):

```
-> 3 December 2001: Stallman Receives
    ➥Prestigious...
-> 22 October 2001: FSF Announces Version 21
   ➥of the...
-> 12 October 2001: Free Software Foundation
   ➥Announces...
-> 24 September 2001: Richard Stallman and
   ➥Eben Moglen...
-> 18 September 2001: FSF and FSMLabs come
   ➥to agreement...
```

A fenti kifejezéslista nem teljes, például a hírek frissítését nem kezeli. A kifejezéseket a kisebb HTML-módosításoktól (például a színek cseréjétől, betűtípusok méretétől stb.) lehetőség szerint függetleníteni kellene. A következő szabályos kifejezés minden betűtípusokkal kapcsolatos jelölést eltávolít:

```
$HTML_FILES =~ s/<font face="Verdana"
➥size="3">([^<]+)<\/font>/$1/g;
```

Ez ugyanazt teszi, de bármilyen fajtájú és (pozitív) méretű betűtípus esetén működik:

```
$HTML_FILES =~ s/<font face="[^"]+"
➥size="\d+">([^<]+)<\/font>/$1/g;
```

Az itt bemutatott példák szemléltetik a módszer alapelveit, és mint már szó volt róla, egyszeri befektetéssel a későbbiekben sok-sok időt nyerhetünk.

Hírek megjelenítése a saját képernyődön

Ha már sikerült az értékes szöveget valamilyen oldalról kinyerned, természetesen nem vagy arra korlátozva, hogy csak a saját konzolodon, egyénileg használd fel. Amennyiben valami mást is szeretnél tenni, például mindig értesülni arról, ha mondjuk ***Stallman***-tól jelenik meg valami, csak három lépés szükséges hozzá. Első lépésben vedd fel a parancsfájlt a `cron`-bejegyzéseid közé (ezzel kapcsolatban a `man cron` parancs mindent elmond), ezután pedig a programodhoz add a következő ellenőrzést:

```
if ($HTML_FILE =~ m/Stallman/) {
        #  RTES˝T S K LD SE
}
```

Így a parancsfájl a teendőit csak akkor végzi el teljesen, ha a megadott feltétel teljesül, és a hírekben szerepel Stallman neve (vagy természetesen bárki másé, akiét beállítjuk). A következő lépésben ezt a pár sort írjuk be a kapcsos zárójelek közé:

```
open (XMSG, "|/usr/bin/X11/xmessage
➥-title \"NEWS!\" -file -") || die;
print XMSG $HTML_FILE;
close XMSG;
```

Ez a néhány sor megnyit az X alatt egy ablakot a `-title` után

megadott címmel, és a `-file` tulajdonság után megadott fájl tartalmával. Esetünkben a `-tulajdonsÆg` azt jelenti, hogy a program a szöveget az állandó bemenetről olvassa be. Ezekután a Perl-parancsfájl addig vár, amíg az `xmessage` ablakot be nem zárod. Talán pont erre van szükséged. Nem szabad azonban arról sem megfeledkezni, hogy a `cron`-ból futunk, tehát jobb megoldás, ha az `xmessage`-t a háttérben futtatjuk egy ideiglenes fájlon, majd pedig kilépünk:

```
open (XMSG, "> /tmp/gee") || die;
print XMSG $HTML_FILE;
close XMSG;
exec "/usr/bin/X11/xmessage
➥-title \"NEWS!\" -file /tmp/gee&";
```

Ellenőrizzük, hogy bizonyos idő elteltével megváltozott-e az oldal!

Amennyiben az oldalt csak akkor szeretnéd feldolgozni, ha az utóbbi látogatásod óta vagy az elmúlt két órában megváltozott a tartalma, a ***Last-Modified*** HTTP-fejlécre lesz szükséged. Ez már `@HEADER` tömbünk 3. elemeként rendelkezésre áll. A hozzá tartozó érték 1970. január 1-je óta a másodperceket számolja. Ezért ha csak olyan oldalakra van szükséged, amelyek az utóbbi két órában módosultak, számold ki az időt, amit ez a számláló két órája mutatott (mindig az „eltelt másodpercek” egységben):

```
$NOW = time;
$TWO_HOURS_AGO = $NOW - (3600 2);
```

Majd ezt az időt hasonlítsd össze a weblap módosítási idejével:

```
if ($HEADER[2] > $TWO_HOURS_AGO) {
        # a sz ksØges feladat elvØgzØse
}
```

Dinamikus könyvjelzők hozzáadása az ablakkezelő menüjéhez

Ez a művelet azon kevés esetek egyike, amikor a csináld magad szabály alól kivételt kell tennünk: töltsd le a WMHeadLines nevű programot (lásd a Hivatkozások részt), majd telepítsd fel, és az ízlésednek megfelelően állítsd be. A program 120 különböző webhely híreihez nyújt hozzáférést, amelyeket aztán elhelyez a BlackBox, a WindowMaker, az Englightenment és a Gnome menüiben, oly módon, hogyha kattintasz rajtuk, a böngészőt elindítva a kért oldalakra jutsz.

A böngésző vezérlése parancsfájlból

A Netscape-nek például többféle parancsot kiadhatunk a héjból vagy akár egy parancsfájlból. Egy ilyen paranccsal a Netscape arra utasítható, hogyha eddig még nem futott, induljon el és jelenítse meg a kért oldalt, illetve amennyiben már futott, az oldal a pillanatnyi vagy pedig új ablakban jelenjen meg. Hogy milyen parancsot kell használnunk, attól is függ, hogy a böngésző éppen fut-e. Vess a WMHeadLines csomagban egy pillantást az ***nslaunch.pl*** parancsfájlra, és meglátod, miként kérdezhető le, hogy a Netscape fut-e már.
A Netscape parancsfájljaidból egyéb feladatok elvégzésére is utasítható, például miután behozta a választott oldalt, a böngésző segítségével nyomtathatunk is:

```
exec($NETSCAPE, ·-noraise·, ·-remote·,
➥"openURL($URL,new-window)");
```

Az oldal PostScriptbe mentése:

```
exec($NETSCAPE, ·-noraise·, ·-remote·,
➥"saveAs(/tmp/netscape.ps, PostScript)");
```

Végül pedig nyomtatása:

```
exec("mpage -PYOURPRINTER -1 /tmp/netscape.ps");
```

Akár a könyvjelzők közé is felvehetjük:

```
exec($NETSCAPE, ·-noraise·, ·-remote·,
➥"addBookmark($SOME_URL, $ITS_TITLE)");
```

A Konquerort, a KDE webböngészőjét egyszerűen a következő módon indíthatjuk el:

```
system("/usr/bin/konqueror $URL");
```

A Konqueror nagyon jól kezelhető parancsfájlokból áll, és nemcsak a Webre jellemző feladatokat végezhetünk vele, hanem akár fájlokat is másoltathatunk, vagy eszközöket fűzhetünk be. Működésének megismeréséhez írd be a következőt:

```
kfmclient -commands
```

A Galeon is ugyanígy indítható:

```
system("/usr/bin/galeon $URL");
```

Mint ahogyan az „A User’s Guide to Galeon” cikkben látható (Hivatkozások rész), még azt is eldöntheted, hogy az új hely egy új fül alatt nyíljon-e meg:

```
system("/usr/bin/galeon -n $URL");
```

Esetleg egy teljesen új ablakban:

```
system("/usr/bin/galeon -w $URL");
```

Vagy akár egy ideiglenes könyvjelzőt is készíthetünk:

```
system("/usr/bin/galeon -t $URL");
```

Okos böngészés

Egy ellentétes megközelítés, mint például az általános tükrözés, vagy a képek letöltése elvégezhető a böngészőből is, például a Konquerorból vagy a KMailből. Ha a jobb egérgombbal kattintasz egy hivatkozáson, a megjelenő menüből kattints a „Megnyitás ezzel…” menüpontra, és ha beírod a programod elérési útját, a következőkben már önmagától fel fogja ajánlani. Ez annyit jelent, hogy a `fetch_images` parancsfájllal néhány kattintással elkészítheted az oldalakról a saját másolatodat, amennyiben követed az utasításokat és a programodat a háttérben futtatod.

Okos tükrözés és FTP

Az `@ALL_URLS` tömbben található URL-ekkel FTP-oldalakról is készíthetsz tükrözést. Ez a Perlből is teljességgel elvégezhető, a sok-sok FTP-ző és tükröző modulok valamelyikének felhasználásával, vagy pedig olyan módon, hogy a szükséges címeket összegyűjtjük, és a dolog lényegi részét a `wget`-re vagy a `curl`-re bízzuk, mint az ***A. J. Chung*** „Downloading without a browser” című cikkében látható (Hivatkozások rész).
Ha a kedvenc portálod napról napra változtatja a kinézetét, és te mégis le szeretnéd tölteni magadnak, csak mentsd az oldal

címét, ahogyan képek esetén tennéd, majd a programodban add ki a következő parancsot:

```
exec "wget -m -L -t 5 $COMPLETE_URL";
```

Az URL-eket a Perl programnak tulajdonságként továbbadva az összes szükséges parancs végrehajtható, amely a párhuzamos FTP-zéshez, vagy a tükrözéshez szükséges.

Építsd fel a saját portálodat!

Sokunknak több kedvelt oldala is akad, és az összes kedvencünket egyetlen ablakban szeretnénk látni. Általános megoldás, ha a HTML-törzset minden oldalból a következő módon vágod ki:

```
$HTML_FILE = s/^.*<body[^>]*>//i;    # levÆg
➥mindent a HTML t rzs elıtt
$HTML_FILE = s/<\/body[^>]*>.*$//i; #  Øs utÆn
```

Ekkor kinyomtat egy HTML-táblát, amelynek minden egyes négyzetében az oldalak egyenként találhatók meg:

```
print<<END_TABLE;
 {{Ide j n minden HTML HEAD Øs BODY dolog}}
<TABLE>
<TR><TD>$HTML_FILE_1</TD></TR>
<TR><TD>$HTML_FILE_2</TD></TR>
.........
</TABLE></BODY></HTML>
END_TABLE
```

A programot ***myportal.html*** néven mentsük a saját könyvtárunkba, a böngésződben erre az oldalra állítsd be a kezdőoldalt, és gyönyörködj benne! A teljes programnak szüksége lehet arra, hogy az oldalak CSS- és karakterkészlet-beállításain csiszoljon, de mostanra már te is képes vagy erre, tudod?

Összegzés

Éppen csak érintettük az ügyféloldali héjprogramozás felszínét. Ezeken kívül még sok egyéb bonyolult megoldás is elképzelhető, például a sütik és a jelszóval védett oldalak kezelése, önműködő űrlapkitöltés, keresés a weben bármilyen jellemzőkkel, weboldalak elemzése, és a tíz legtöbbet kiválasztott hivatkozás megjelenítése, vagy akár a webes levélellenőrzés. Mindehhez csak egy kis türelem szükségeltetik, egy csipetnyi Perl-gyakorlat, és a rendelkezésre álló modulok ismerete.
Jó böngészést kívánok!

Marco Fioretti
alkatrészrendszerek fejlesztésével foglalkozik hálózati kábel nélküli ATM-eszközökhöz az Ericsson Labnél Olaszországban. Érdekli még a természetjárás, a hegymászás, a Perl, a webdesign, az etika és az új módszerek közötti kapcsolat, továbbá a Linux-asztal hatékonnyá varázsolása. Marco Rómában él családjával, és a linuxdesk@inwind.it címen érhető el.

Miképp növelhetjük PHP-programjaink sebességét?

Bruno rámutat néhány problémára, amelyektől PHP-programjaink lelassulhatnak, és egyúttal megoldást is közöl az elkerülésükre.

Számos oka lehet PHP-programjaink lelassulásának . Dugót okozhat az adatbázis-hozzáférés sebessége, a weboldalak elérése, vagy akár egy rosszul megtervezett, lassú algoritmus. Ha a sebesség csökken, a felhasználó feszültté válik, míg a kérésre várakozik. Kevés felhasználó kisebb hasznot hoz, és a webhelyed elveszíti népszerűségét. Legfőképpen a rossz tervezés és fejlesztés okolható a lassúságért. Gyakran hoznak létre és indítanak el honlapokat mindenféle teljesítménypróbák nélkül. Az adatbázisok sokszor kevesebb adat kiszolgálására képesek, mint amennyit várnak tőlük. Az algoritmusok pedig számtalanszor rosszul tervezettek, és sebesség tekintetében sem hatékonyak.
Ha a webhely felgyorsításához nem áll módodban újratervezni az egészet, a teljesítmény fenntartása érdekében statikus oldalak kiszolgálására lesz szükséged, így a PHP-kódot nem kell minden egyes letöltéskor újrafordítani. Nézzük meg, milyen lehetőségeink vannak ennek a kivitelezésére!

A hagyományos megközelítés

Az első dolog, amit tehetsz, hogy megvizsgálod, mely programokra vagy programrészekre kell a legtöbbet várakozni. Ezt a legegyszerűbben a PHP-héj (shell) segítségével oldhatod meg. Tegyük fel, hogy létezik egy ***index.php*** nevű parancsfájlod, és ennek a kimenetét szeretnéd statikussá tenni. Feltéve, hogy a PHP-héj neve ***phpsh***, a parancssor a következőképpen néz ki:

```
phpsh --q /egy/k nyvtÆr/index.php >
➥/egy/k nyvtÆr/index.html
```

Az ***index.html*** fájl most egy teljesen egyszerű HTML-fájl, amely semmiféle PHP-feldolgozást nem igényel, és az ügyfél változatlan formában töltheti le. Vajon mi történik olyankor, amikor a PHP-program kimenete időről-időre változik? Minden egyes alkalommal, amikor a kimenet megváltozik, újra el kell készítenünk egy új, statikus változatot.

Rendszeres újrafeldolgozás

Linux alatt az időnkénti futtatás legegyszerűbb módja a `crontab`. A következő `crontab`-bejegyzés a megadott parancsot minden negyedórában lefuttatja:

```
*/15 * * * * root phpsh  q
➥/egy/konyvtar/index.php >
➥/egy/konyvtar/index.html
```

Előfordulhat, hogy az adat frissentartásához ez az időzítés nem megfelelő. Léteznek olyan parancsfájlok, amelyeket nagyon ritkán kérnek le, azonban akadnak olyanok is, amelyeket folyamatosan próbálnak elérni – ebben az esetben ez a módszer értelmetlen, helyette más parancsfájlalapú lehetőség szükséges.

Előfeldolgozás csak szükség esetén

Ez a módszer a parancsfájlokat bizonyos időközönként újra feldolgozza, de csak akkor, ha valamilyen kérés érkezik rá. Ez a működés nagyon hasonló egy webproxy működéséhez. Két fogást mutatok be e módszer megvalósítására: a kimenet átmeneti tárazását és az időn túli feldolgozást.
A kimenet átmeneti tárazása esetén a parancsfájl ellenőrzi a tárolt fájl dátumát és idejét, és csak szükség esetén dolgozza fel őket. A feldolgozás úgy működik, hogy mielőtt elküldené az ügyfélnek, a parancsfájl kimenetét egy gyorstárfájlban tárolja. PHP-ben ezt az `ob_start()` függvény segítségével lehet elvégezni. Ez a függvény bekapcsolja a kimenet átmeneti tárazását, és elküldi egy visszahívó függvénynek. Létezik egy másik függvény is, mellyel a kimenetet elküldhetjük a böngészőnek: `ob_end_flush()`. Vessünk egy pillantást a példánkra:

```
<?php
// a fejlØcfÆjl csatolÆsa
include("header.php");
?>
<?php
// 10 mÆsodperc pihenı
sleep(10);
?>
Test:
<?php
// A lÆblØc csatolÆsa
include("footer.php");
?>
```

A ***header.php*** fájlban található minden gyorstárazással kapcsolatos szolgáltatás. Ez először ellenőrzi, hogy szükség van-e új gyorstárfájl-változat tárolására a `needscache()` függvény segítségével. Ez az ellenőrzés történhet idő- vagy fájlmódosítás alapján, vagy ahogyan szeretnéd. Írásunkban idő szerinti példát veszünk alapul.
Ha az előző (gyorstárazott) példány elavult, a parancsfájl elindít egy `ob` ciklust, és a kimenetét kiírja a fájlba. Ugyanakkor ha a fájl még aktuális, annak tartalmát egyszerűen elküldi a böngészőnek (1. lista, 30. CD. Magazin/PHP).
A ***footer.php*** fájl az `ob`-feldolgozást egyszerűen csak lezárja:

```
<?php
ob_end_flush();
?>
```

Működését kipróbálhatod, ha a parancsfájlt többször meghívod, míg a gyorstárban tárolt fájl el nem évül. Tapasztalni fogod, hogy az első futás alkalmával a prancsfájl tíz másodpercet vár (mivel a parancsfájl egy `sleep()` hívás következtében a jobb szemléletesség céljából várakozik), míg a többi futás alkalmával a parancsfájl azonnal visszatér. Mindenesetre ha a tárolt fájl elévül, mindenképpen várnod kell, hogy a parancsfájl újat hozzon létre.
Lássuk, hogyan tudod ezt megkerülni, és azt az látszatot kelteni, mintha a parancsfájlod mindig gyors lenne!

Időn túli feldolgozás esetén a parancsfájl ellenőrzi a fájl dátumát és idejét, de a feldolgozás mindig csak azután történik meg, hogy a böngészőt már kiszolgálta – megoldva ezzel a fájl elévülésével járó feldolgozási nehézséget. Ebben az esetben a gyorstárazott fájl csak a parancsfájl lefutása után lép működésbe. PHP-ben ez úgy oldható meg, hogy „a parancsfájl befejezése” eseményhez a `register_shutdown_function()` híváson keresztül egy függvényt rendelünk hozzá. Ebben az esetben csak a ***header.php*** fájlon szükséges módosítani (*2. lista*). Egyszerűen csak hozzáadtam a `doaftercache()` függvényt, amely akkor hívódik meg, ha a parancsfájl futása befejeződött. Ez a függvény meghívja a parancsfájlt – ahogyan a böngésző is tenné –, és a tartalmát tárolja. Csak egyetlen esetben várható a böngészőből, mégpedig ha a parancsfájl korábban még sosem futott. Próbáld ki, és az lesz az érzésed, hogy a program nagyon gyors!

Összegzés

Bemutattuk, hogyan működik a PHP gyorstárazási módszere, és a működését egy „csináld magad” példával szemléltettük. Ha a példákat kipróbáltad és tetszettek is, nyugodtan használd fel őket a saját programjaidban. Léteznek egyéb módok is a teljesítmény javítására, mint például a függvény-gyorstárazás vagy a PHP-előfordítás. E feladatok megvalósítására a PHP további eszközöket kínál. Keresd meg a legjobb megoldást, és próbáld ki, hogyan állja a viszontagságokat!

Bruno Pedro
rendszerfejlesztő, tízévnyi tapasztalattal rendelkezik az adatbázis-alkalmazások terén, ezenkívül társalapítója és vezetője az ethernet lda projektnek.

2. lista Tetszőleges eljárás azonosítása a parancsfájl kilépési eseményével

```
<?php
// a gyorstÆr elØrØsi œtvonala
$CACHE_PATH="/tmp";

// a gyorstÆr idıtœllØpØse mÆsodpercekben
$CACHE_TIMEOUT=10;

// a parancsfÆjl elØrØse
$SCRIPT_URL="http://".$HTTP_HOST.$PHP_SELF;

// gyorstÆrfÆjl lØtrehozÆsa a parancsfÆjl
// nevØvel Øs a gyorstÆr œtvonalÆval
function cachefilename() {
global $PHP_SELF,$CACHE_PATH;

    return($CACHE_PATH."/".md5($PHP_SELF).
➥".cache");
}

// megvizsgÆljuk, hogy a parancsfÆjlt
// sz ksØges-e gyorstÆrazni
function needscache($timeout) {
    clearstatcache();
    if (time()-
    ➥filemtime(cachefilename())>$timeout)
        return(true);
    else
        return(false);
}

// beolvassuk a gyorstÆrat Øs kik ldj k
// a b ngØszınek
function outputcache() {
    readfile(cachefilename());
}

// gyorstÆrazzuk a parancsfÆjlt
function docache($buffer) {

    // ki rjuk az eredmØnyt a
    // gyorstÆrba
    $fp=fopen(cachefilename(),"w");
    if ($fp)
        fputs($fp,$buffer);

    // Ætadjuk a kimenetet
    // a b ngØszınek
    return($buffer);
}

// gyorstÆrazzuk az eredmØnyt
function doaftercache() {
global $SCRIPT_URL;

    // beolvassuk a parancsfÆjl kimenetØt
    $fp=fopen($SCRIPT_URL,"r");
    while (!feof($fp))
        $buffer.=fgets($fp,4096);

    // ki rjuk a parancsfÆjl kimenetØt
    // a gyorstÆrba
    $fp=fopen(cachefilename(),"w");
    if ($fp)
        fputs($fp,$buffer);
    fclose($fp);
}

if (needscache($CACHE_TIMEOUT)) {
    // a parancsfÆjlnak sz ksØges
    // a gyorstÆrazÆs
    if (file_exists(cachefilename())) {

register_shutdown_function("doaftercache");
        outputcache();
        exit();
    } else
        ob_start("docache");
} else {
    // a parancsfÆjl gyorstÆrban van,
    // olvassuk ki
    outputcache();
    exit();
}
?>
```

Böngészőprogramok mérkőzése

A Linux-rendszerben manapság sokféle böngészőprogramhoz hozzá lehet jutni, és meglehetősen sok program fejlesztése még csak az 1.0-s változathoz közelít. Mostani szemlénkből megtudhatjuk, mire képesek és milyen feladatokat képtelenek ezek a programok elvégezni. Amióta néhány éve a Mozilla nyílt forrású vállalkozás útjára indult, azóta több, a Linux-rendszerekben használható pehelysúlyú és a szabványoknak megfelelő böngészőprogram ígérete tűnt fel a láthatáron.

Jelen írásunk hét különböző fejlesztés alatt álló böngészőprogramot vizsgál meg, és hasonlít össze abból a szempontból, hogy az egyes böngészési feladatokat mennyire sikeresen oldották meg. Valamennyi böngésző hibátlanul dolgozott és a rendszer biztonságát sem veszélyeztette, mégis olyan kiábrándító vizsgálati eredmények is születtek, mint amilyen például a többoldalas dokumentumból az oldaltartomány nyomtatásának meghiúsulása.

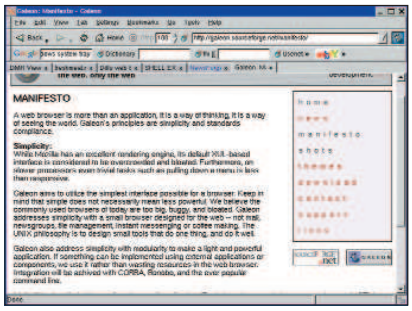
A vizsgálatnak az alábbi névvel és változatszámmal ellátott böngészőprogramokat vetettük alá: Beonex Communicator 0.7-dev-2, BrowseX-1.5.0, Galeon-1.0.1, Konqueror-2.2.1, Mozilla-0.9.6, Opera-5.0-static és a Skipstone-0.7.7. A Skipstone és a Konqueror programokat forráskódból fordítottam, a többit pedig rpm-csomagokból telepítettem. A programfejlesztés nagy sebessége és a termelési folyamatok havi ütemezéshez való kapcsolódása miatt meglehet, hogy amikorra olvasóink a Linuxvilág mostani számát kézbe veszik, a böngészőprogramok frissebb változatai is megjelentek már.

A programokat Red Hat 6.2 rendszer alatt próbáltam ki egy 133 MHz-es Pentium processzorral és 80 MB memóriával ellátott gépen. A GNOME 1.4-et és a KDE 2.2.1-et, továbbá a CUPS-1.1.5-öt még az összehasonlítás előtt telepítettem a számítógépre.

A vizsgálatok felépítése

A böngészőket olyan feladatok elé állítottuk, amelyek kimutatták, hogy egy-egy program milyen jól hajtotta végre a böngészési, állományletöltési és nyomtatási feladatokat. A vizsgálatokat és eredményüket a *táblázatban* foglaltam össze, az egyes vizsgálatok mibenlétét pedig az alábbiakban fogom részletezni.

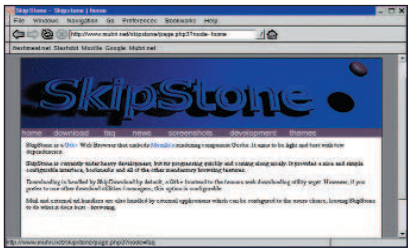
- Webbank: ez a részvizsgálat azt ellenőrizte, hogy a böngészőprogram képes volt-e belépni a felhasználó bankjába, és ott megmutatni a számlaadatokat.



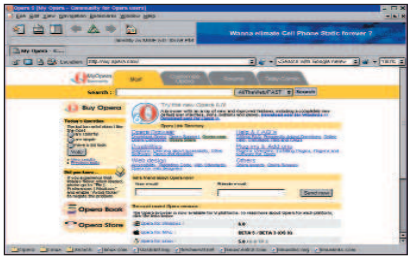
Galeon képernyőmentés

- PayPal: minden egyes böngészőprogramnak be kellett jelentkeznie a PayPal ➲ http://www.paypal.com címre, ott ellenőrizni a számlaegyenlegeket és a kívánt összeget átutalni a PayPal számláról.
- Titkosítás: a ➲ http://fortify.net címen SSL-ellenőrzés zajlott le, amely meghatározta a böngészőprogramok titkosításának mértékét.
- My eBay: a vizsgálat e része a ***Bejelentkezés*** (Sign In) lehetőség kiválasztása révén az eBay ➲ http://www.ebay.com címre történő bejelentkezést és több weboldal megjelenítését ellenőrizte anélkül, hogy az eBaynél a felhasználónak ismételten be kellene magát jegyeztetnie.
- Árverés az eBaynél: minden böngészőnek képesnek kellett lennie árverés létrehozására, amely folyamat a következő lépésekből állt:
  1. kattintás a ***Sell*** (Eladás) gombon,
  2. kategória kiválasztása,
  3. az eladásra kínált árucikk űrlapjának felhasználó általi kitöltése és
  4. a ***Continue*** (Folytatás) gombon való kattintást követően az űrlap feldolgozásának megtörténte.

  A korai Mozilla-alapú böngészőprogramok JavaScript-kezelési hibák miatt ezen a vizsgán megbuktak (Mozilla programhiba 911018), de úgy tűnik, mostanra már kinőtték ezt a gyermekbetegségüket.
- iPrint: a ➲ http://www.iprint.com címen működő URL lehetővé teszi a papíralapú üzleti levelezés dokumentumainak böngészőprogrammal való előállítását. Akkor mondható, hogy egy böngészőprogram sikeresen vette az akadályt, ha képes volt választani az üzleti csekkek közül és a szerkesztési feladatokkal is megbirkózott.



Skipstone képernyőmentés

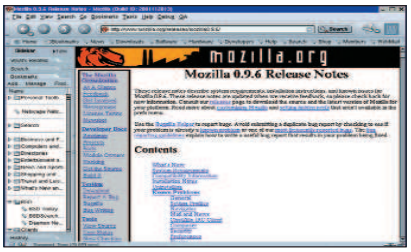


Opera képernyőmentés

- Nyomtatás: a böngészők nyomtatási képességeinek megítélésénél ellenőriztem a dokumentum oldaltartományainak nyomtatását, a dokumentum elejétől kezdődő, illetve a fordított sorrendben történő nyomtatást, a színes és a szürkeárnyalatos, az álló vagy fekvő laptájolást, valamint az állományba történő nyomtatást.
- Állományok letöltése: a böngészőprogramok letöltési képességeit a letöltési hivatkozásokon (link) való kattintással és az FTP-helyekre való belépéssel engedélyeztem. Külön megemlítem, ha a böngészőprogram külső letöltéskezelő program használatát is megengedte.
- Használhatósági jellemzők: ebben a csoportban olyan tulajdonságok szerepelnek, amelyek megkönnyítik a böngésző használatát. Ilyen szol-

gáltatások például a mozgóképek lejátszásának kikapcsolása (vagyis a *.GIF* kiterjesztésű állományok lejátszásának felfüggesztése), a húzd és ejtsd lehetőség használata (URL-eket más alkalmazásokból be lehet oda vinni, vagy onnan más alkalmazásokba át lehet vinni), az ID felhasználói azonosítót meg lehet változtatni, az egér görgetőkerekét a böngészőprogram megfelelően tudta használni, a helymegadási mezőt egy egérkattintással törölni lehetett (a böngésző gondoskodott egy mód-

Beonex Communicator képernyőmentés



Mozilla képernyőmentés

szerről, amellyel a helymegadási mező tartalmát egy egérkattintással ki lehetett üríteni), végül pedig ellenőriztem a beépített nagyítás (zoom) használatát, amellyel a képernyőn megjelenő szöveget nagyítani vagy éppen kicsinyíteni lehetett.

- Levelezés: ez a vizsgálat azt ellenőrizte, hogy a böngészők hogyan kezelték a címzett hivatkozásokat. Továbbá azt is jelzi, hogy a böngésző a felhasználó által megadott levelezőprogramok indítási lehetőségét is felkínálja.
- Java: ebben az erőpróbában a böngészők megmutathatták, hogy képesek-e Java-kisalkalmazások futtatására, mint amilyenek például a Sun honlapján található bemutatóprogramok. A vizsgálathoz felhasznált Java2 programcsomag a Netscape webhelyéről származik.
- Bővítmények: annak meghatározása, hogy a böngésző felismeri-e a Netscape-bővítményeket, pontosabban szólva, a Macromedia Flash-bővítményt, és helyesen jelenít-e meg olyan weboldalakat, amelyeknek ilyen programra van szükségük. Ez a vizsgálat egyik fontos célja volt.
- Átlátszó PNG-kép: a böngészőnek a vizsgálaton való sikeres szerepléshez a Gimppel készített áttetsző hátterű képet tartalmazó weboldalt kifogástalanul kellett a képernyőn megjelenítenie. Amennyiben a böngészőben a kép háttere fekete maradt, a böngésző ezen a vizsgán bizony megbukott.

A böngészőprogramok bizonyítványa

A Beonex Communicator mind működését, mind küllemét tekintve gyakorlatilag megegyezik a Mozillával, a különbség annyi, hogy a Beonex Communicator biztonsági és keresési fejlesztésekkel is dicsekedhet.

Konqueror képernyőmentés

BrowseX képernyőmentés

Alapértelmezett keresőgombja, amely az oldalsó eszköztáron foglal helyet, a Mozillánál több keresőgép használatát megengedi, sőt a keresések egyidejűleg több keresőgépen való futtatását és a találati eredmények együttes megjelenítését is lehetővé teszi. A Beonex Communicator beépített keresőgépei továbbiakkal egészíthetők ki, ha felkeressük a Mozilla Sherlock weboldalát a ➲ http://sherlock.mozdev.org címen.
A Beonex egyik vezetője szerint céljuk az, hogy erőfeszítéseik nyomán a Beonex a felhasználók számára vonzóvá váljon. Ezt a böngészőprogramot könnyű telepíteni, és az olyan programokat, mint a Java, a Beonex webhelyéről mindössze néhány egérkattintással telepíthetjünk. Ezenkívül a böngésző „gyári beállításait" is megváltoztatták: az egy-

egy idény során kikényszerített sütielfogadást és a HTTP-hivatkozók címeit egyaránt a nagyobb biztonságnak megfelelően állították be. A cég programhasználó ügyfeleit a hibákról és a felfedezett biztonsági résekről a webhelyén keresztül értesíti.
A BrowseX-1.5.0 a megvizsgált böngészők közül egyediségével tűnik ki: a program elkészítéséhez a cl és C nyelveket használták fel, amely emiatt mérsékelt erőforrásigényű lett. Saját levelező- és beszélgető-, illetve csevegő- (talk/chat) programmal rendelkezik, támogatja a HTML 3.2-t, képes a grafikák, az SSL-felületű biztonságos webhelyek és a JavaScriptek (héjprogramok) kezelésére. Eme böngészőprogramnak további sajátos jellemzője, hogy a TML-állomány-kiterjesztést használja, valamint a Tcl, Perl és Python nyelvű programok számára beágyazott felületet ad. A BrowseX a ***File/Util*** menü használata révén képes a már létező Netscape-könyvjelzőket átvenni: ***Import Netscape Bookmarks***.
A böngészőnek az eBayjel való nyomtatással meggyűlt a baja. A nyomtatóra történő nyomtatás meghiúsult, viszont az állományba való nyomtatás sikeres volt. A böngésző az eBay-oldalak megjelenítése során felesleges karaktereket is a képernyőre küldött, emellett az árverés létrehozásakor nem engedte meg tárgykör kiválasztását. A BrowseX legördülő listái nem nyílnak ki teljesen, ehelyett a lista alján a további választási lehetőségekre utaló ***More*** felirat látható, amely további elemeket tartalmazó ugyanilyen listákat rejt magában, a Netscape-hez nagyon hasonló módon. Emiatt a rendkívül hosszú listákban szinte lehetetlen tájékozódni.
A Galeon jelmondata úgy szól, hogy „a Web, csakis a Web", amely arra utal, hogy a Galeon kizárólag webböngészői feladatokat kíván ellátni, és nem áll szándékában „mindent egy helyen" eszközzé válni. A Galeon a Mozilla-kódon alapul, és a működéséhez szükség van a Mozilla előzetes telepítésére.
A Mozilla-alapú böngészőprogramok közül az elsők között jelent meg, a program kiforrottságát az általa nyújtott szolgáltatások is bizonyítják.
A böngészőprogram egyik elismert szolgáltatása a ***Smart Bookmarks*** (Okos könyvjelzők), amelyhez saját eszköztár tartozik, és a felhasználónak akár a Freshmeat II, akár a Google kereső vagy a Google hírlevéltárában módjában áll keresgélnie. További kellemes szolgáltatás a ***Beállítások***-nak a felső menüsorban történő elhelyezése, amely gyors

A böngészők próbaeredményei

	Beonex	BrowseX	Galeon	Konqueror	Mozilla	Opera	Skipstone
Változatszám	0.7-dev-2	1.5.0	1.0.1	2.2.1	0.9.6	5.0 static	0.7.7
Web bank	IGEN	IGEN	IGEN	IGEN	IGEN	IGEN	IGEN
Fizetés PayPallal	IGEN	IGEN	IGEN	IGEN	IGEN	IGEN	IGEN
Kódolás	RC4 cipher 128-bits	Triple DES 168-bits1	RC4 cipher 128-bits	Triple DES 168-bits1	RC4 cipher 128-bits	RC4 cipher 128-bits	RC4 cipher 128-bits
My eBay	IGEN	NEM[2]	IGEN	NEM[3]	IGEN	IGEN	IGEN
eBay-árverés létrehozása	NEM[4]	NEM[2]	IGEN5	HIBA	IGEN	IGEN	NEM
iPrint	NEM[3]	NEM[6]	IGEN	IGEN	IGEN	IGEN	IGEN
Köztes tartomány nyomtatása	IGEN	N/A	NEM[7]	IGEN	IGEN	IGEN	NEM7
Nyomtatási sorrend	NEM	N/A	NEM	IGEN	NEM	NEM	NEM
Színes nyomtatás	IGEN	IGEN8	N/A	IGEN	IGEN	HIBA[9]	IGEN
Papír tájolása	N/A	IGEN8	N/A	IGEN	IGEN	NEM	N/A
Fájlba nyomtatás	IGEN	IGEN	IGEN	IGEN	IGEN	IGEN	IGEN
Webhely mentése lemezre	IGEN	IGEN[10]	IGEN	IGEN	IGEN	IGEN[11]	IGEN
Letöltés	IGEN	IGEN[12]	IGEN[13]	IGEN	IGEN	IGEN	IGEN[13]
Használhatóság	animáció kikapcsolása, húzd és dobd, egérgörgő kezelése, nagyítás	animáció kikapcsolása	animáció kikapcsolása, húzd és dobd, egér kerék, nagyítás	húzd és dobd, törlés egy kattintással, egérgörgő kezelése, nagyítás	animáció kikapcsolása, húzd és dobd, egérgörgő kezelése, nagyítás	animáció kikapcsolása, húzd és dobd, törlés egy kattintással, egérgörgő kezelése, nagyítás	egérgörgő
Több oldal megnézése	több ablak	több ablak	több ablak/fülek	több ablak	több ablak	fülek	több ablak/fülek
Levelezés	beépített	beépített	a felhasználó határozza meg	a Kmailt indítja	beépített	a felhasználó határozza meg	a felhasználó határozza meg
Java	IGEN	N/A	IGEN	IGEN	IGEN	N/A	IGEN
Bővítmények (Plug-in)	IGEN	N/A	IGEN	IGEN	IGEN	N/A	IGEN
Átlátszó PNG	IGEN	N/A	IGEN	IGEN	IGEN	NEM	IGEN

Magyarázat
IGEN = sikeresen átment, HIBA = átment ugyan, de néhány nehézség adódott, NEM = nem sikerült, N/A = nincs adat

Megjegyzések:
1. Nem figyelmeztet a lejárt SSL-azonosításra.
2. Az oldal letöltődött, de nem jelent meg rendesen.
3. Letöltött egy DLL fájlt az eBayről, azután megjelenítette az oldal forrását.
4. Az űrlapok Tovább gombja nem működött, amikor új árverést hoztunk létre, JavaScript-hiba miatt. (Mozilla bug 91018).
5. Rossz oldalt jelenített meg a kategóriában (Régiségeket a Könyvek helyett), a hivatkozás viszont jó volt.
6. „Page Not Found" (Az oldal nem található) hibával tért vissza.
7. Az összes oldalt kinyomtatta a kijelölt rész helyett.
8. A nyomtatás fájlba volt az összes lehetőség, ami működött.
9. Akkor is színesben nyomtatott, ha a szürkeskála volt kijelölve.
10. Oldal mentése lemezre, de a mentett oldalt nem jelenítette meg rendesen.
11. Oldalmentés képpel vagy kép nélküli.
12. Jobb gombbal kattintás a hivatkozáson és mentés.
13. Külső letöltőprogram engedélyezése.

hozzáférést enged a proxy- és JavaScript-beállításokhoz és a mozgóképlejátszáshoz.
A Galeon némi bosszúságot tartogat azok számára, akik a beépített letöltéskezelőjét szeretnék használni. Ha a program úgy lett beállítva, hogy a felhasználó kiválaszthatja a letöltésre váró állomány helyét, akkor a letöltendő állomány neve a könyvtár kijelölésekor törlődik. Ezért az állomány nevét a tárolási hely kijelölését követően, még a mentés megkezdése előtt vissza kell írni.
A Konqueror a KDE-felület webes és állományrendszer böngészőprogramja, amely nem a Mozilla-forráskódján alapul. A böngésző valóban egyedi szolgáltatásokat kínál, azaz a böngésző

Kapcsolódó címek

Böngészők

Beonex Communicator
➲ http://www.beonex.com/communicator

BrowseX
➲ http://www.browsex.com

Galeon
➲ http://galeon.sourceforge.net

Konqueror ➲ http://www.kde.org

Mozilla ➲ http://www.mozilla.org

Opera ➲ http://www.opera.com

SkipStone
➲ http://muhri.net/skipstone

A próbához használt oldalak

eBay ➲ http://www.ebay.com

iPrint ➲ http://www.iprint.com

PayPal ➲ http://www.paypal.com

Internet bank
➲ http://www.michigannational.com

Egyéb

Fortify SSL-ellenőrzés
➲ http://www.fortify.net/sslcheck.html

Mozilla Sherlock bővítmény-kereső-motor ➲ http://sherlock.mozdev.org

Sun Java-bemutatók
➲ http://java.sun.com/j2se/1.3/docs/relnotes/demos.html

ablakai ablaktáblákra oszthatók, amelyekben más-más weboldal vagy épp adott helyi könyvtár tartalma jeleníthető meg. A Konqueror Java és JavaScript futtatását is lehetővé teszi, és emellett a böngésző azonosításának egyes webhelyenkénti irányítását is.
Végül az eszköztáron található egy nyomógomb – fekete alapon fehér X –, amellyel a böngésző keresőmezőjének tartalma könnyűszerrel törölhető.
A Konquerornak viszont néhányszor meggyűlt már a baja az eBay kezelésével. Az első helyen kell megemlítenem, hogy a ***My eBay*** hivatkozáson való kattintás a hatására a Konqueror ***.dll*** állomány letöltését jelezte, azonban honlap helyett a képernyőn csak annak HTML-kódját jelenítette meg.
Ezenkívül ha valaki az árverés létrehozásakor a ***Könyvek*** tárgykört választotta ki, a böngészőben tévesen a ***Régiségek*** tárgykör jelent meg. A többi csoportot viszont hibátlanul ki lehetett választani.
A ➲ http://www.konqueror.org címen található egy olyan Konqueror + Java útmutató, amelyből megtudhatjuk, hogy a Konquerorban hogyan kell Javát használni. A KDE forráskódjának fordítása során az OpenSSL-környezet összeépítésénél használjuk a `config shared` lehetőséget, hogy a Konqueror által igényelt megosztott könyvtárak létrejöjjenek.
A Mozilla böngészőprogramot a Netscape még 1998-ban nyílt forrásúvá tette. A pehelysúly nem szerepel a böngésző tervezési jellemzői között, és valóban: lomhasága nagyon is szembetűnő. A böngészőprogram honlapja szerint a kód működési sebességének növelése érdekében a fejlesztés nagy erőkkel zajlik, és minden változat ténylegesen gyorsabb a megelőzőnél.
Támogatja az olyan jelentős szolgáltatások használatát, mint a könyvjelzők, a keresési eredményeket tartalmazó oldalsó eszköztár és a beállítási témák váltogatásának lehetősége. Sőt, azon néhány böngészőprogram közé tartozik, amelyeknek egy-egy oldaltartomány kinyomtatása sem okozott gondot, viszont a lapok fordított sorrendben való nyomtatása már túlságosan nehéz feladatnak bizonyult.
A Mozilla 0.9.6-os változatában a nyomtatási előképnél néhány furcsasággal találkozhatunk. A böngésző főablakában először a nyomtatási előkép jelenik meg, és úgy tűnik, a weblap képéhez a frissítés hiánya miatt semmilyen módon nem lehet visszatérni. További hiányosság, hogy a nyomtatási előképben láthatók olyan fejlécek, mint a weboldal címe vagy az egyesített erőforráskereső (URL), amelyek a kinyomtatott lapokon sajnos már nem jelennek meg. A Mozilla is büszkélkedhet azzal a képességgel, hogy egy ablakban több weboldalt képes megjeleníteni, amelyek között a TAB billentyűvel lehet váltani – új ablak megnyitása nélkül.
Amennyiben a Mozillát rpm-csomagokból telepítjük, a ***Personal Security Manager*** (PSM) csomagot is telepíteni kell, hogy a Mozilla vagy a rokon böngészőprogramok – például a Galeon és a Skipstone – a titkosított honlapokat képesek legyenek kezelni.
A Windows-felület számára már régóta hozzáférhető Opera a leggyorsabb böngészőprogramként hirdeti magát. Maga a program gyorsan elindul és készséges, de egyáltalán nem úgy tűnik, hogy a weblapokat a többi programnál gyorsabban tudná a képernyőre varázsolni.
A Linux-felület számára készült Operát Qt-eszközzel fejlesztették, és a program statikus és dinamikus változatai a program honlapjáról egyaránt letölthetők.
Az Opera kiforrottságot mutat mind felületében, mind szolgáltatásaiban, de nála is akad kifogásolnivaló. Nagyon durva, bár csak esetenként jelentkező hiba, hogy a böngésző a régi weboldal tartalmát nem képes az újjal felváltani, annak ellenére, hogy határozottan állítja: már befejezte annak letöltését. Ez a Freshmeat honlapjával gyakran megtörtént, és az egyetlen hibaelhárítási lehetőség a lap újbóli betöltése volt. Nehézségei akadtak a nyomtatással is: a fekvő laptájolás kiválasztásakor a nyomtatás elmaradt, és a nyomtató annak ellenére ragaszkodott a színes nyomtatáshoz, hogy szürkeárnyalatos nyomtatás volt beállítva. Az Opera 5.0-s változata sem a Javát, sem a bedolgozó modult nem támogatja, de az Opera 6 előzetes műszaki leírásában már vannak jelei annak, hogy ezek a lehetőségek az Opera 6-os változatában használhatók lesznek.
A Skipstone szintén Mozilla-alapú, a Galeonnál fiatalabb böngésző, amely karcsúbb működési felületet biztosít. Ez a böngésző olyan meglepetésekkel szolgál, mint például a semmilyen módon meg nem tekinthető helyi könyvtárak, még a ***File/Open*** (Fájl/Megnyitás) menün keresztül sem. Sőt, az előre-hátra nyilakkal az eseménytörténeteket sem lehet lejátszani.
Miközben a Mozilla-alapú böngészőnek nem jelentett gondot az eBaynél árverést létrehozni, addig a Skipstone-ban a folytatás gombon való kattintás a program összeomlásához vezetett.

Összefoglalás

A Linux-felület számára már jó néhány böngészőprogram elérhető, és úgy tűnik, hogy a verseny valamennyi programban hasznos szolgáltatások megjelenéséhez vezet(ett). Az esetek túlnyomó többségében valamennyi böngésző alkalmas a mindennapi használatra, de a felhasználók számára a korlátot az általuk felkeresett honlapok jelentik. Több most megvizsgált program még csak valamilyen 1.0 előtti változatnál tart, így remélhető, hogy a hibák és a furcsaságok a fejlesztés során eltűnnek.

Ralph Krause
a michigani alsó félszigeten lakik. Egy személyben író, webtervező és programozó. Három évnél is régebb óta használ Linuxot, elérhető a rkrause@netperson.net címen.

Mit parancsolsz? – avagy Linux-idomítás dióhéjban

A kezdő, illetve a leendő Linux-felhasználóknak szánt sorozatunk első két részében csupán elméleti síkon beszéltünk a Linux világáról. Mostantól azonban lépésről lépésre a gyakorlati elemeket is átvesszük. Kezdjük rögtön a legalapvetőbb parancsok ismertetésével!

A rendszer telepítésének kérdéseivel sorozatunkban azért nem foglalkozunk, mert lapunk korábbi számaiban szinte az összes elterjedt változat telepítését a lehető legrészletesebben kivesézték. Ezenkívül egyes Linux-változatok (például SuSE, Mandrake) telepítését annyira leegyszerűsítették, hogy felpakolásuk semmiféle unixos előképzettséget nem igényel.
Először azokat az alapvető Unix-felhasználói ismereteket mutatjuk be, amelyek elsősorban azok számára lesznek hasznosak, akik ez idáig semmilyen egyéb Unix-alapú rendszerrel nem kerültek közelebbi kapcsolatba. Ebben a részben az alapparancsokat tárgyaljuk, természetesen csupán két oldalra korlátozva – hiszen valahol meg kell húznunk a határt –, így a legtöbb utasításról csak pár szót fogunk ejteni. Mindenesetre bárki részletes ismertetőt kaphat, ha akármelyik parancsról a `man` *`parancs`* utasítást adja ki.
Linuxunk betöltése után rögtön a barátságtalan „login:” sor fogad minket. Ha telepítettünk valamilyen grafikus bejelentkező programot (például: kdm, gdm, xdm), akkor a „fogadtatás” valamivel melegebb hangú lesz. A grafikus rendszer beállításával, felépítésével, kezelésével csak a következő részekben foglalkozunk, mivel először a „fapados” szöveges (konzolos) környezettel kell megbarátkoznunk. A feladatok többségét a szöveges felület segítségével gyorsabban és egyszerűbben végrehajthatjuk, ezért hasznos lehet, ha már az elején a konzolhoz szoktatjuk magunkat.
Ha telepítettünk valamilyen grafikus bejelentkező programot, szöveges környezetre a CTRL-ALT-F1 billentyűkombináció segítségével térhetünk át.
A Linuxban eredetileg hat úgynevezett virtuális terminált határoztak meg. Ezt úgy kell elképzelnünk, mintha a gépünkhöz hat monitor és billentyűzet lenne csatolva, ahol a programokat egymástól függetlenül futtathatnánk. E virtuális terminálok között az ALT-F1 – ALT-F6 kombinációkkal tudunk váltani. Ezenkívül egy grafikus terminál is létezik, amit az ALT-F7-tel érhetünk el. Ez természetesen csak a grafikus rendszer elindítása után érhető el.
Először is jelentkezzünk be! Ha rendszergazdaként (root) lépünk be a rendszerbe, promptként mindig egy #-t kapunk. Ez azonban halálos figyelmeztetés: most a rendszer korlátlan urai vagyunk, azaz bármiféle megkötés nélkül bármit megtehetünk. Tehát csak óvatosan!
A Linuxban (és a többi Unix-alapú rendszerben) a felhasználókat alapvetően két csoportba oszthatjuk: a „mindenhatókra” és az egyszerű „halandókra”.
A mindenhatók alatt természetesen a rendszergazdai jogosultságú felhasználókat értjük, akik a `root` felhasználóval megegyező jogosultsággal bírnak. Rajtuk kívül vannak még „általános” felhasználók, akik csak a saját állományaikhoz és az általuk futtatott alkalmazásokhoz férnek hozzá, máséhoz nem. A rendszergazdára efféle megkötés természetesen nem vonatkozik, ő bármelyik felhasználó „cuccához” hozzáfér.
A felhasználók nyilvántartása a rendszerben nem az azonosítójuk alapján történik, hanem egy szám segítségével, amelyet UID-nek (User ID) hívunk. Minden felhasználónak egyedi UID-dal kell rendelkeznie. A 0-s UID-ú felhasználó birtokolja a rendszergazdai jogosultságokat (ez rendszerint a `root`). Egyes Unix-rendszerek lehetővé teszik, hogy a rendszerben több rendszergazdai jogosultsággal bíró felhasználó is létezzen. Ebben az esetben az ilyen felhasználókhoz UID-ként a 2-nek valamely nagyobb hatványát kell rendelnünk.
A felhasználókat úgynevezett csoportokba is szervezhetjük. Ennek gyümölcsét egy otthoni gépen aligha élvezhetjük, ugyanis nem sok előnye van akkor, ha maximum 3-4 felhasználó létezik, de egy több száz felhasználós rendszerben jelentősen megkönnyítheti a rendszerfelügyeletet, mivel az egy csoportra meghatározott jog az összes az adott csoportot alkotó felhasználóra vonatkozik. A csoportok is rendelkeznek azonosítóval, de a felhasználókhoz hasonlóan az azonosítás itt is számokkal történik. Ezt a számot GID-nek (Group ID) hívjuk. Minden felhasználónak kötelezően egy csoporthoz kell tartoznia, amelyet a felhasználó alapcsoportjának hívunk. Az alapcsoporton kívül természetesen számtalan más csoportba is tartozhat.
A felhasználókra jellemző az úgynevezett ***Saját könyvtár*** is, ilyennel mindenki rendelkezik. Ez tulajdonképpen a felhasználó saját könyvtára, ahová csak ő írhat be, itt tárolhatja a „cuccait”, illetve a saját beállításait. A felhasználói könyvtárak a ***/home*** alatt találhatók. A rendszergazda Saját könyvtára a ***/root***. A felhasználó saját könyvtárának útvonalát a ~ tilde jellel is helyettesítheti. A pillanatnyi könyvtár jele a `.` (pont).
Az utolsó felhasználókra vonatkozó fontos jellemző az úgynevezett héj (shell). A héj fogalmával gyakran találkozhatunk a Unix világában. Olyan programról van szó, amely a felhasználó és az operációs rendszer között biztosítja a kapcsolatot. Minden operációs rendszernek létezik héja, például a DOS-nál a command.com számít annak, a Windowsoknál pedig a ProgramManager. A Linuxban többféle héjprogram közül válogathatunk, sőt minden felhasználóhoz külön héjat rendelhetünk. A felhasználókhoz meghatározott héjak rendszerint parancsértelmezők, mint például a `bash`, `sh`, `csh` vagy a `ksh`. A héjnak választott alkalmazás a bejelentkezés után önműködően elindul.
A rendszer a felhasználók adatait a ***/etc/passwd*** állományban tárolja, amelyet természetesen csak a rendszergazda módosíthat. Ez is – mint csaknem minden beállítófájl – szöveges állomány, tehát egyszerű szövegszerkesztővel is módosíthatjuk, de sokkal kényelmesebb, ha a felhasználók felügyeletére az

`adduser`, `userdel`, `usermod` parancsokat használjuk.
A felhasználó saját jelszavát a `passwd` utasítás segítségével természetesen saját maga is megváltoztathatja. Ilyenkor először a régi, majd kétszer az új jelszót kell megadni. A rendszergazda bárkinek a jelszavát megváltoztathatja. Ehhez is a `passwd` parancsot kell használni, csak értékként annak a felhasználónak az azonosítóját kell megadni, akinek új jelszót kívánunk adni.
A felhasználók jelszavai egyébként titkosított formában tárolódnak, és majdhogynem lehetetlen visszafejteni őket. Ez azzal jár, hogy a rendszergazda sem tudja megállapítani, kinek mi a jelszava. A régebbi Unix-rendszerekben ezeket a titkosított jelszavakat is a */etc/passwd* állományban tárolták, ez azonban a rendszer biztonságára nézve komoly veszélyeket hordozott. A ***passwd*** állomány tartalma ugyanis bárki számára hozzáférhető, tehát senki és semmi nem állíthatta meg a sötétebb lelkű betörőket, hogy a jelszópróbálgatós módszerrel (a nyers erő módszerével) fényt derítsenek a gyenge jelszavakra (a mai számítógépek másodpercenként akár több ezer jelszót is kipróbálhatnak). Ezt a biztonsági hiányosságot felismerve a titkosított jelszavakat ma már külön állományban tárolják (*/etc/**shadow***), amit csak a rendszergazda olvashat, illetve írhat.
Ezekután ismerkedjünk meg néhány egyszerűbb fájlkezelő utasítással. A Linuxban természetesen használhatunk hosszú fájlneveket, a legnagyobb méret 255 karakter. Figyelem, a Unix a kis- és nagybetűk között különbséget tesz! Mivel rendszergazdaként bármit letörölhetünk, biztonságosabb, ha a saját felhasználói nevünkkel jelentkezünk be, és úgy gyakorlunk. Most már a # helyett egy $ fogad minket.
A Linux parancsértelmezői nagyon „okosak”, a DOS parancssoránál összehasonlíthatatlanul fejlettebbek. Például itt is lehetőségünk nyílik batchprogramírásra (Unix alatt ezeket héjparancsoknak hívjuk), de sokkalta több lehetőség rejlik bennük. Olyan ez, mintha egy teljesen különálló, magas szintű programozási nyelven programoznánk (a héjparancsok írását sorozatunk későbbi részében taglaljuk bővebben).
A parancssorba egyszerre több utasítást is begépelhetünk, amit a parancsértelmező egymás után fog végrehajtani. Ebben az esetben az utasításokat a „;” jellel kell elválasztanunk egymástól.
Arra is lehetőség van, hogy egy parancs kimenetét egy másik parancs számára átadjuk, ami majd az előbbit feldolgozza. Erre az egyik legjobb példa a `grep` parancs, ami a szövegben egy adott részletre keres rá: `cat /etc/passwd | grep root`. Az utasítás végrehajtását követően a */etc/**passwd*** állományból csak azok a sorok íródnak ki a képernyőre, amelyekben a `root` karaktersorozat szerepel. A `cat` parancs egy állomány tartalmának megjelenítésére szolgál, feladata a DOS type utasításával egyezik meg.
Egy parancs kimenetét nemcsak a képernyőre, hanem más eszközre is kiküldhetjük, például egy állományba: `cat /etc/passwd >xxx`. Ekkor létrejön egy „xxx” nevű állomány, melynek tartalma megegyezik a */etc/**passwd***-évél. Ha egy `>` helyett kettőt használunk, és a kimenetként megadott állomány már létezik, akkor nem felülírja azt, hanem új tartalomként a végéhez fűzi.
Egy alkalmazás háttérben való futtatását a parancs után írt `&`-tel érhetjük el.
A DOS-ból már jól ismert `cd` paranccsal tudunk könyvtárat váltani, egy könyvtár tartalmának listázására az `ls` utasítást használhatjuk. A rejtett fájlok (a `.` ponttal kezdődő állományok) listázására a `-f` kapcsoló alkalmas. Az `ls -l`-lel az adott könyvtár állományainak egyéb adatait is megtekinthetjük.
Az első oszlop első karaktere az állomány típusa: `-`, ha általános állomány, `d`, ha könyvtár és `l`, ha hivatkozás (link). Gyakorlati haszna az, hogy a segítségével ugyanazt az állomány több néven is elérhetjük. Hivatkozást az `ln` parancscsal hozhatunk létre: `ln ÆllomÆny hivatkozÆsnØv`. Az így keletkezett hivatkozás a közvetlen hivatkozás (hard link). Ennek van egy nagy hátránya, mégpedig hogy csak egyetlen fájlrendszeren belül használhatjuk, továbbá könyvtárakra nem alkalmazható. Ha más fájlrendszerbeli állományokra, illetve könyvtárakra is szeretnénk hivatkozni, ehhez a csatolás másik fajtáját, az úgynevezett közvetett hivatkozást (szimbolikus link) kell segítségül hívnunk. Közvetett hivatkozás létrehozására is az `ln` utasítást használhatjuk, csak ebben az esetben még egy `-s` kapcsolót is meg kell adnunk. Nézzünk erre is egy példát: `ln -s file link`.
Az ezt követő kilenc karakter az állományra vonatkozó jogokat jelzi. A Unix-rendszerekben minden állományhoz egy tulajdonos és egy csoport tartozik. Az állományokra vonatkozó jogosultságokat csak a tulajdonos, illetve a rendszergazda változtathatja meg (a tulajdonos és a csoport nevét a harmadik és a negyedik mezőben láthatjuk). Az első három karakter a tulajdonosra, a következő három a csoportra, és az utolsó három pedig a mindenki másra vonatkozó jogosultságokat írja le. Az `r` az olvasási jogot, a `w` az írási jogot, az `x` pedig a végrehajtási jogot jelenti.
A végrehajtási jog természetesen csak futtatható állományoknál érdekes.
A futtatható állományok esetében arra is lehetőségünk nyílik, hogy a program ne az őt futtató felhasználó, hanem a tulajdonos jogosultságaival fusson. Ebben az esetben egy úgynevezett Set UID joggal (ha a csoportja jogosultságaival akarjuk futtatni, akkor Set GID-del) kell felruháznunk. Ha egy fájl ilyen jogosultsággal bír, akkor a futtathatóságot jelző `x` helyett `s`-t láthatunk.
Ez miért lehet hasznos a számunkra? Azért, mert így lehetőség nyílik rá, hogy bármely felhasználó olyan értékeket is meg tudjon változtatni, amelyekre az érvényben lévő korlátozások miatt általában nem lenne lehetősége. Erre a legjobb példa a ***passwd*** program, amelyet jelszava megváltoztatásához bármelyik felhasználó futtathat. A ***passwd***-nek mindenképp Set UID-esnek kell lennie (tehát rendszergazdai jogosultságokkal kell futnia), mivel a jelszavakat tároló állományt csak a rendszergazda írhatja és olvashatja. Ha ez nem így lenne, a rendszergazdán kívül senki más nem tudna jelszót váltani a rendszerben.
A tulajdonost a `chown`, a csoportot pedig a `chgrp` utasítással változtathatjuk meg. Az állományra (vagy könyvtárra) vonatkozó jogok kiosztására a `chmod` parancs alkalmas (az idevonatkozó lista megtalálható a 30. CD Magazin/Dobbanto könyvárban). Ha például azt szeretnénk, hogy egy állomány mindenki számára futtatható legyen, a `chmod a+x file` parancs bepötyögésével érhetjük el.
Sorozatunk következő részében tovább folytatjuk a Linux-alapparancsokkal való ismerkedést.

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

Az Emacs (4. rész)

Hogyan kezeljük az Emacsban az ablakokat és a kereteket?

Elsőként a munkaterületet függőlegesen osszuk fel két részre a *C-x 2* paranccsal, úgy, hogy az egyik ablak a másik fölött legyen, majd a *C-x 1* paranccsal állítsuk vissza az eredeti állapotot! Most üssük be a *C-x 3* billentyűkombinációt, amivel vízszintesen osztjuk ketté a munkaterületet úgy, hogy az ablakok egymás mellé kerüljenek! Folytassuk a felosztásokat tetszőleges sorrendben, amíg jó sok vízszintesen és függőlegesen felosztott ablakunk nem lesz, majd a *C-x 0* paranccsal zárjuk be őket egyenként! Figyeljük meg, hogy a *C-x 1* a pillanatnyi kivételével minden ablakot bezár! A sok kinyitott ablak közt egérkattintással vagy a *C-x o* billentyűkkel válthatunk. A kettéosztott ablakokban a pillanatnyi átmeneti tároló tartalma fog megjelenni, ha új tartalommal akarjuk megtölteni, akkor új fájlt kell megnyitnunk, vagy a meglévő átmeneti tárolók közül kell választanunk.

Az ablakok az egérrel átméretezhetőek, de a következő billentyűkombinációk is használhatók erre a célra:

M-x shrink-window – úgy csökkenti az ablak méretét, hogy az alsó keretet felfelé viszi,
C-x ^ – úgy növeli az ablak méretét, hogy az alsó keretet lefelé viszi,
C-x { – az ablakot keskenyíti,
C-x } – az ablakot szélesebbé teszi.

A *C-x +* parancs minden ablakot azonos magasságúvá tesz, a *C-x -* pedig csökkenti az olyan ablak magasságát, amelyikben sok üres sor van alul. A felszabaduló helyet a többi ablak kapja. Ha két ablakot nyitunk meg egymás mellett, gyakran előfordulhat, hogy az egyikbe gépelünk, a másikból pedig olvasunk valamit. Ilyenkor nagyon jól jön, hogy a *C-M-v* paranccsal a másik ablakban előre és lapozhatunk.

Talán lehet valami logikát találni az ablakokhoz és a keretekhez kapcsolódó teljes billentyűk számozásában.

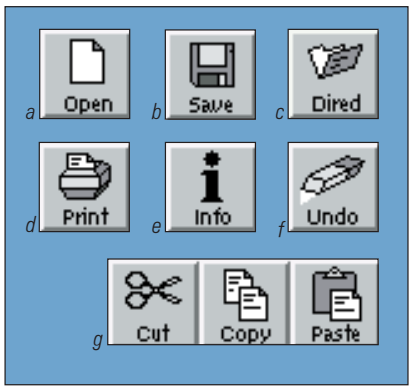
Tekintsük csak át őket még egyszer!

C-x 0 – törli a pillanatnyi ablakot.
C-x 1 – a pillananyi ablak kivételével minden ablakot töröl.
C-x 2 – az ablakot függőlegesen osztja ketté.
C-x 3 – az ablakot vízszintesen osztja ketté.
C-x 4 0 – törli az ablakot és a hozzá tartozó átmeneti tárat is.
C-x 4 b – ugrás a másik ablakban lévő átmeneti tárhoz.
C-x 4 d – egy másik ablakban nyitja meg a ***Könyvtár Szerkesztőt***.
C-x 4 f – új ablakban nyit meg egy fájlt.
C-x 4 r – új ablakban nyit meg egy fájlt – csak olvasásra.
C-x 4 C-o – egy másik ablakban mutatja meg az átmeneti tárat.
C-x 4 m – egy új ablakban levelet írhatunk.
C-x 4 . – egy címkét egy másik ablakban keres meg.
C-x 5 0 – bezárja a pillanatnyi keretet.
C-x 5 2 – ugyanazt az átmeneti tárat új keretben nyitja meg.
C-x 5 b – az általunk választott átmeneti tárat új keretben nyitja meg.
C-x 5 f – a fájlt új keretbe hívja be.
C-x 5 r – új keretben csak olvasásra nyit meg egy fájlt.
C-x 5 d – új keretben nyitja meg a ***Könyvtár Szerkesztőt***.
C-x 5 o – kiválasztja a következő keretet.
C-x 5 . – másik keretben keres meg egy címkét.
C-x 5 m – egy új keretben levelet írhatunk.
C-x m – levelet írunk a pillanatnyi ablakban.

A *C-z* az X nélküli parancssoron felfüggeszti az Emacs működését, az XEmacsot pedig ikonméretűvé csukja össze. Ha a főalkalmazás mellett más keretek is nyitva vannak, a *C-z* csak a működő keretet zsugorítja össze.

Kurzormozgatás

Ha a betöltött fájlunk elég hosszú, örömmel tapasztalhatjuk, hogy az ismerős PAGE UP és PAGE DOWN billentyűkkel lapozhatunk a szövegben, és a le, fel, jobbra és balra mozgató kurzorbillentyűk a megszokott módon működnek. A bal egérgombbal is lapozhatunk, ha többször rákattintunk az XEmacs-ablakokban található pörgetőrudakra, de ha a középső egérgombbal kattintunk



1. kép

2. kép

ugyanoda, a szöveg azonnal a kattintás által kijelölt szöveghelyre ugrik. A HOME a sor elejére, az END a sor végére ugratja a kurzort, míg a CTRL+HOME a fájl elejére, a CTRL+END pedig a fájl végére ugrik. A DELETE és az INSERT is a várakozásoknak megfelelően viselkedik: a DELETE egy karaktert töröl, az INSERT pedig a felülírási módot váltogatja a beszúrási móddal. Az Emacs természetéből fakad, hogy ugyanezeket a feladatokat billentyűkombinációkkal is meg lehet ismételni. Például a *C-v* előre, az *M-v* visszafelé lapoz egy képernyőnyit, a *C-x-<* balra, a *C-x->* pedig jobbra görget. Érdekes hatása van a *C-l* billentyűkombinációnak. Vigyük a kurzort valamelyik, a képernyő szélén lévő sorba, jegyezzük meg az ott lévő szavakat, majd nyomjuk meg a *C-l* teljes billentyűt. A parancs hatására minden újrarajzolódik, de a kurzort tartalmazó sor a képernyő közepére fog kerülni. Ha ezzel végeztünk, próbáljuk ki a le, fel, jobbra, balra nyilakat úgy, hogy közben a CTRL vagy az ALT billentyűket is nyomva tartjuk.

Ahogy korábban említettem, az ESC helyettesítheti az *Meta* billentyűt, próbáljuk ki tehát az *Esc-v* kombinációt is! Nálam működött. A le, fel, jobbra és balra mozgatás elérhető a *C-n* (next line azaz következő sor), a *C-p* (previous

line, azaz előző sor), ***C-f*** (forward, azaz előre) és a ***C-b*** (backward, azaz visszafelé) lenyomásával. Az előre és a hátra a ***Meta*** billentyűvel is kombinálható (***M-f*** és ***M-b***), ilyenkor a kurzor nem egy karaktert mozdul el, hanem egy egész szót ugrik előre vagy hátra. Figyeljük meg, hogy hátrafelé ugráskor a szó első betűjére ugrik, előreugráskor viszont a szó utáni első üres karakter helyre! ***C-a*** a sor elejére, ***C-e*** a sor végére ugrik. Hasonló viselkedést nyerünk, ha az ***M-a*** és ***M-e*** kombinációt alkalmazzuk, de most a kurzor a mondat elejére vagy végére visz bennünket. Programozás közben az Emacs felismeri, hogy egy kifejezésnek vagy függvénynek hol van az eleje vagy a vége. A ***C-M-a*** és ***C-M-e*** kombinációk ilyenkor egy függvény elejére vagy végére ugranak. A ***C-M-b*** egy kifejezés elejére, a ***C-M-f*** pedig a végére visz. ***M-<*** a fájl elejére, ***M->*** a fájl végére ugrik. Megjegyzem, a <> zárójelek nemcsak a magyar billentyűzeten nehezen hozzáférhetők, ahol az ALT GR-t kell megnyomni, de az angolon is csak a SHIFT megnyomásával hozhatók elő. Ez valójában az ALT és az ALT GR vagy az ALT és a SHIFT egyidejű megnyomását jelenti. Nálam az ***Alt-Alt Gr <***-változat ritkán hozta a várt eredményt, helyette inkább az ***Esc-Alt Gr <*** kombinációt használom, ha használom.
Láthatjuk, hogy a táblázat közepén a pont szó ül. Ez ismét olyan Emacs-sajátosságra utal, ami még a TECO nyelv idejére megy vissza, ahol a "." eredetileg egy parancs volt, ami megadta a pont helyén lévő értéket. A pont mindig a kurzor alatti karakter elé mutat (*2. kép*).

1. táblázat

	Görgetés balra		***Görgetés jobbra***
Előző képernyő		M-v;	
	C-x-<	C-l	C-x->
Következő képernyő		C-v	

Némi lazasággal a pont szó helyett a kurzor szót is használhatjuk.

A számkapcsoló

Több parancsnak számkapcsolót adhatunk át, ami megmondja neki, hogy hányszor ismételje meg ugyanazt. Például a ***C-u 3 C-a*** a harmadik sor elejére ugrik, a ***C-u 3 C-e*** ugyanennyit, de a sor végére. Üssük be még a ***C-u 6 C-n*** és a ***C-u 6 C-p*** parancsokat is, figyelve arra, hogy ugyanúgy viselkedik-e a kurzor, mint az előző példában, azaz a pont elé vagy a pont utánra ugrik-e! A ***C-u 8 C-v*** nem nyolc képernyőnyit, hanem csak nyolc sornyit mozgat visszafelé! A ***C-u 8 M-v*** szintén csak nyolc sort megy előre. A ***C-u*** előtag ***u*** betűje az ***universal argument*** szavakra utal, és ez az előtag adja át a számkapcsolókat az utána következő parancsoknak. Ha nem adunk meg számot a ***C-u*** után, az alapértelmezett érték mindig négy lesz, például a ***C-u C-n*** néggyel fogja lefelé mozgatni a kurzort. Próbáljuk ki a ***C-u C-l*** parancsot különböző számkapcsolókkal, majd a ***C-u -*** billentyűkombinációt is!

Keresés és csere

Az ***Edit > Search...*** vagy a ***C-s*** indítja a keresést. Ahogy a keresendő szót a mini átmeneti tárolóba gépeljük, az Emacs rögtön ráugrik a már beírt karakterlánc első előfordulására. Hosszabb szavak esetén előfordulhat, hogy a teljes szót be sem kell írni ahhoz, hogy rögtön célhoz érjünk. Ha a megtalált szó újabb előfordulásaira is kíváncsiak vagyunk, akkor ismételten meg kell nyomnunk a ***C-s*** kombinációt. Az ENTER leütése jelzi a keresés befejeztét, de ***C-s C-s*** még ezután is visszahívhatja a legutóbb keresett szót. A ***C-g*** kiléptet bennünket a keresésből, és a kurzort visszaviszi a kiindulási helyre, az ESC szintén abbahagyatja a keresést, de a kurzort a megtalált szó helyén hagyja. Gépeléskor a DELETE megnyomása letörli a mini átmeneti tárolóba beírt karaktereket, és sorban visszaugrál a kiindulási helyre. Ha a keresett karakterlánc nem található a vizsgált átmeneti tárban, akkor figyelmeztető hangjelzést kapunk, és a mini átmeneti tárolóban megjelenik a ***Failed I-search*** üzenet. Az ***Edit > Search backward...*** vagy a ***C-r*** visszafelé keres, különben ugyanúgy viselkedik, mint a ***C-s***.
Szavak sorozatát a ***C-s RET C-w szavak RET*** vagy visszafelé a ***C-r RET C-w szavak RET*** parancsokkal tudjuk keresni. A keresett szavak közé egyetlen szóközt teszünk. Az ilyen kereséskor nem számít, hogy a szövegben lévő szavakat hány szóköz választja el egymástól, hogy van-e közöttük új sor vagy bármilyen már elválasztó karakter a szövegben.
Lehetőség van reguláris kifejezések előre- és visszafelé történő keresésére is: a ***C-M-s*** és ***C-M-r*** billentyűkkel. A keresést és a helyettesítést az ***M-x replace-string RET karakterlánc RET új karakterlánc RET*** paranccsal végezhetjük, ami ***új karakterláncra*** cseréli ***karakterlánc*** minden előfordulását. Reguláris kifejezések esetében az ***M-x replace-regexp RET regexp RET új karakterlánc RET*** formát használjuk. Ezek a helyettesítőparancsok kérdezés nélkül cserélik le a karakterláncokat. Ha azt akarjuk, hogy a program egyenként kérdezzen rá a csere szükségességére, akkor az ***M-x query-replace RET karakterlánc<vége> RET új karakterlánc<vége> RET*** alakot, reguláris kifejezések használatakor pedig az ***M-x query-replace-regexp RET >regexp<vége> RET új karakterlánc<vége> RET*** parancsot írjuk be. Ha az átmeneti tárban lévő összes előfordulást le akarjuk cserélni, előrefelé keresés esetén a szöveg elejére kell mennünk, mivel a keresés és a csere mindig onnan kezdődik, ahol a kurzor áll. Ha visszafelé haladunk, ilyenkor a szöveg végére kell ugranunk.

2. táblázat

	függvény	***mondat***	***sor***	***Kifejezés***	***szó***	***karakter***		***karakter***	***szó***	***kifjezés***	***sor***	***mondat***	***függvény***
átmeneti tár							M-<						
oldal							C-x-[						
bekezdés							M-{						
sor							C-p						
	C-M-a	M-a	C-a	C-M-b	M-b	C-b	pont	C-f	M-f	C-M-f	C-e	M-e	C-M-e
sor							C-n						
bekezdés							M-}						
oldal							C-x-]						
átmeneti tár							M->						

3. táblázat

	mondat	kifejezés	sor	szó	karakter	pont	karakter	szó	sor	kifejezés	mondat
Emacs	C-x Del	M-- C-M-k	M-0 C-k	M-Del	Del?		C-d	M-d	C-k	C-M-k	M-k
XEmacs	C-x Del?	u.a.	u.a.	M-backspace	backspace		u.a.	u.a.	u.a.	u.a.	u.a.

Lapozgatás közben az állapotsoron láthatjuk, hogy a dokumentum hány százalékánál tartunk, de az ***M-x line-number-mode RET*** paranccsal ugyanott kiírattathatjuk, hogy a kurzor hányadik sorban jár. A parancs ismételt kiadásáig a sorszám mindig olvasható lesz. Az ***M-x goto-line RET*** a megadott sorra ugrik, a rövid alakja ***M-g***.

Kijelölés

Ha az Emacs átmeneti tárában lévő szövegre a jobb egérgombbal rákattintunk, majd a gombot nyomva tartva elmozgatjuk az egeret, kijelölhetünk egy területet (region). A terület mindaddig kijelölve marad, amíg új kijelölést nem kezdeményezünk, vagy bele nem kattintunk a szövegbe. Kijelölhetünk egy területet úgy is, hogy a ***C-szóköz*** vagy a ***C-@*** kombinációval egy jelet (mark) teszünk a kurzor helyére, majd a kurzort elmozgatva kijelöljük a kiválasztandó területet. Nemcsak a kurzormozgató billentyűket használhatjuk, hanem a fent megismert kurzormozgató billentyűkombinációkat is, mint például ***M-a***. Gondolom, nem lesz meglepő, hogy a ***C-g*** megnyomásával a kijelölést megszüntethetjük. Bizonyos műveletek csak a kijelölt területre hatnak. Tegyünk egy jelet a kijelölés elejére a ***C-szóközzel***, majd menjünk a kijelölés végére. Miután végeztünk a kívánt terület kijelölésével, a ***C-x C-x*** megnyomásával visszaugorhatunk a terület elejére, majd a ***C-x C-u*** parancs kiadásával az egész szöveget egy lépésben nagybetűssé alakíthatjuk. Most csináljunk egy újabb kijelölést, és a ***C-x C-l*** segítségével a szöveget alakítsuk vissza kisbetűssé! Hasznos lehet a ***C-x C-x*** akkor is, amikor elfelejtjük, hogy mi volt a kijelölés eleje vagy vége.

Ha egy kijelölésjelet a pillanatnyi kurzorállásra helyezünk, majd megnyomjuk az ***M-<*** vagy ***M->*** billentyűket, az átmeneti tároló elejéig vagy végéig egy lépésben kijelölhetjük a szöveget. A ***C-x h*** az egész átmeneti tárat, a ***C-x C-p*** egy oldalt, az ***M-h*** egy bekezdést, ***C-M-@*** egy kifejezést, ***C-M-h*** pedig egy függvényt jelöl ki egy lépésben.

Ha nem az Emacson belüli átmeneti tárolók közt akarunk szöveget cserélni, a bal egérgombbal ki kell jelölnünk a kívánt területet, ami rögtön X-kiválasztássá válik. Ez már minden további nélkül beszúrható lesz más X-szerkesztőkbe: mint például a mostani cikk megírásához használt StarOffice 5.2-be. Ebben a szövegszerkesztőben a ***Ctrl-v*** billentyűkombinációt, az ***Edit > Paste*** menüt, vagy a beszúrás ikont kell a beillesztéshez használni, de a KWrite vagy az Xcoral erre a célra a középső egérgomb megnyomását is elfogadja.

Ha programozás közben C módban egy függvény törzsében kiadjuk az ***M-C-h*** parancsot, akkor egy lépésben kijelölhetjük az egész függvény, mivel a parancs megkeresi a függvény elejét és a végét, majd aktívvá teszi.

Kivágás, másolás és beszúrás

A kivágást, a másolást és a beszúrást az ismert módon végezhetjük el: az Edit menüből vagy az idevágó ikonok használatával (*1 g kép*).

De kifizetődő lesz elsajátítani az Emacs rövidítéseit is:

C-w – kivágja a kijelölt területet (wipe).

M-w – másolatot készít a kijelölt területről (wipe).

C-y – beilleszti a korábban kivágott vagy átmásolt területet (yank).

A *w* betű itt a wipe (töröl) szó helyett áll. Az Emacsban kétféle törlés van. Az egyik az ***irtás*** (kill), ami egy szövegrészt töröl, de egyúttal menti is az ***Irtó gyűrűbe*** (Kill Ring), ahonnan később előránthatjuk (yank), és beszúrhatjuk a kívánt helyre. Innen az ***y*** rövidítés. Egyetlen ***Irtó gyűrű*** létezik, és minden kiirtott szövegrész ide kerül, hogy később bármelyik puffer hozzáférhessen. A másik törlési mód megsemmisíti (***delete***) a szöveget, ami még egy ideig a visszaállító (***Undo***) paranccsal visszahozható, de utána véglegesen elveszik. Ilyen megsemmisítő parancsot adunk ki például, amikor megnyomjuk a DELETE vagy a BACKSPACE billentyűket. Szintén mindent megsemmisítően töröl a kurzortól visszafelé a sor elejéig az ***M-0 C-k***, a ***C-k*** pedig a sor végéig. Ha a sor végén lévő új sor (newline) karaktert is törölni akarjuk, akkor a ***C-u 1 C-k*** parancsot kell kiadnunk, mert a ***C-k*** nem teszi meg. Ha beütjük a ***C-u 1 C-k*** parancsot, láthatjuk, hogy az alul lévő sorok eggyel feljebb lépnek.

A *3. táblázat* a megsemmisítő billentyűkombinációkat tartalmazza. Úgy vettem észre, hogy a törléseknél az Emacs és az XEmacs viselkedése nem mindenben feleltethető meg egymásnak. A leírásokkal ellentétben a ***Del*** például nálam nem töröl visszafelé, hanem ugyanúgy viselkedik, mint a ***C-d***. Az Emacsban ***C-x Del*** valóban törli a mondatokat visszafelé, az XEmacsban viszont ugyanazt csinálja, mint az ***M-k***. A táblázatban látható kérdőjelek erre utalnak.

Programozás közben hasznos lehet, ha sorokat szúrhatunk be az adott sor mögé (***C-o***), vagy ha egyszerre távolíthatjuk el a szerkesztett sor mögötti, fölösleges, csak szóközöket tartalmazó, üres sorokat (***C-x C-o***). Ez utóbbi szintén megsemmisítő törlés.

Az ***M-y*** a legutóbbi beszúrást lecseréli az előző irtással. Ennek a parancsnak megint csak az ***Irtó gyűrű*** működésének ismeretében érthetjük meg az értelmét. Tudjuk, hogy az ***Irtó gyűrű*** eltárolja a legutóbbi harminc irtást, amit később az ***M-y*** segítségével sorban elővehetünk. Hogy értsük, miről van szó, kezdjünk el dolgozni az XEmacsban, közben a ***C-w*** vagy ***M-w*** parancsokkal töröljünk vagy másoljunk le kijelölt területeket! Ugorjunk az átmeneti tár elejére, és a ***C-y*** paranccsal szúrjuk be a legutóbbi irtást! Ha most többször kiadjuk az ***M-y*** parancsot, az ***Irtó gyűrűből*** egyenként „előránthatjuk” az oda mentett területeket mindaddig, amíg meg nem találjuk a nekünk tetszőt. Láthatjuk tehát, hogy az ***M-y*** használata is egyfajta visszavonás (***undo***).

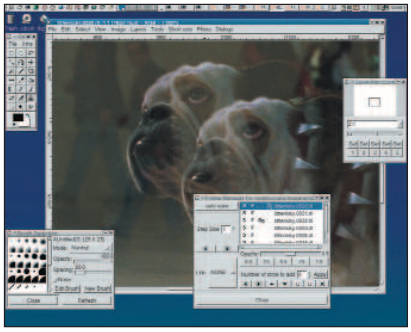
Sorozatunk következő részében a regiszterekkel, betűátalakítókkal és a könyvjelzőkkel ismerkedhetünk meg.

Szaló István (ratiosoft@freemail.hu) tanár, immár több mint másfél évtizede foglalkozik programozással, de csak a Java és a Linux megismerése után tudta meg, hogy mi is az igazi programozás. Több írása megjelent már a hazai számítástechnikai lapokban.

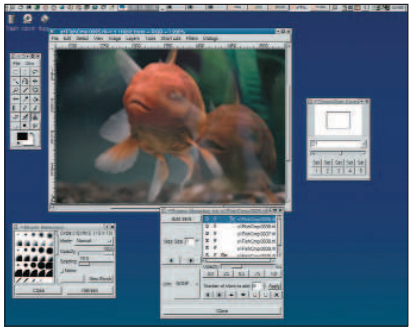
A FilmGimp alkalmazása a Rhythm & Huesnál

Robin a FilmGimpet, a nagy filmgyárak által használt linuxos, nyílt forrású eszközt mutatja be.

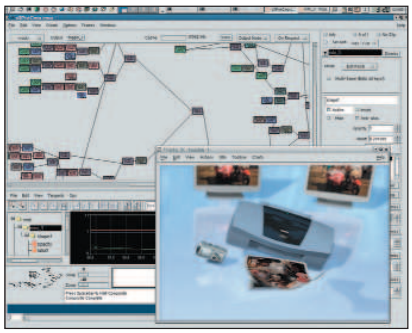
A Gimp valószínűleg vezeti a Linux csúcsalkalmazásainak listáját. Ez a Photoshop-szerű grafikai csomag rendkívül népszerű állóképszerkesztő alkalmazás. Filmes unokatestvérét, a képsorozatokon való munkát támogatni hivatott FilmGimpet már kevesebben ismerik.
A DreamWorks, a Pixar, az ILM és egyéb nagy stúdiók filmgyártó munkafolyamataikat épp napjainkban ültetik át Linuxra. A legtöbb ilyen linuxos törekvés néhány kereskedelmi alkalmazást is magával hoz, ilyen például a népszerű 3D animációs csomag, a Maya (lásd a Maya 3 már Linuxon is! című cikket a Linuxvilág 2001. június-júliusi számában), vagy az olyan ismeretlen eszközök, amelyek a stúdiók által saját felhasználásra gyártott kódsorok millióiban öltenek formát (lásd a Linuxvilág 2001. augusztusi számában a DreamWorks és a Linux kapcsolatáról írottakat). Jelen pillanatban a nagy filmgyárak csak egy jelentős nyílt forrású alkalmazást használnak. Vizsgáljuk meg egy kicsit közelebbről a FilmGimpet, és hogy hogyan használják a Los Angeles reklámfilmgyár utómunkálatokat végző részlegében, a Rhythm & Huesnál.
Caroline Dahllöf, az R&H programozója a FilmGimp vezetőfejlesztője és képviselője. „Minden olyan munkánknál a FilmGimpet használjuk, amelyben beszélő állatok szerepelnek" – tájékoztat Dahllöf. „A FilmGimpet több stúdió alkalmazza a gyártás során, de talán az R&H az, amelyik a legnagyobb mértékben. A többi stúdió is nagyszerű elgondolásnak tartja a Gimpet, de úgy tűnik, mi vagyunk az egyetlenek, aki pillanatnyilag fejlesztik és támogatják is." Dallhöf szívesen venné, ha mások is részt vennének a fejlesztésben. Az R&H a FilmGimpet használta a Harry Potterben, továbbá a Kutyák és macskák (Cats & Dogs), a Dr. Dolittle 2, a Little Nicky, a Grincs (How the Grinch Stole Christmas), a The 6th Day; a Stuart Little, a kisegér és a Majmok bolygója (Planet of the Apes) című filmekben. Az R&H reklámfilmeket is készít, alkotásai között szerepel a jól ismert mackós Coca-Cola reklám. Dahlöff még hozzáteszi: „A legfontosabb munkánkat a beszélő állatok jelentik. Háromdimenziós modellekkel dolgozunk, mert a hatások így sokkal jobbak, mint a kétdimenziós átmeneteknél. Elkészítjük a háromdimenziós modellt, majd leképezzük a síkra – az élő állat mozgását az állat fejének számítógépes grafikai modelljével illesztve összse. Ezután a megvilágító részleg a képet rávetíti a 3D-s modellre, a 2D-részleg pedig a hiányzó háttérrészleteket hozza rendbe, például az állat beszédmódját. A nyújtásokat felületi mintázatokkal kell kijavítanunk. A száj belseje tisztán számítógépes grafikából áll. Néhány projekt, mint például a kólareklám, szintén kizárólag számítógépes grafikára épül. Ez a módszer különbözik attól, amit a beszélő állatoknál használunk.
Ahogy az a filmstúdióknál megszokott, az R&H nemcsak egy eszközt, hanem egész eszközsort használ a háromdimenziós animációk és a különleges élőhatások előállításához."
Mielőtt elkezdenénk a FilmGimppel foglalkozni, vizsgáljunk meg néhányat az eszközláncot alkotó saját eszközök közül, amelyekkel a FilmGimpnek szorosan együtt kell működnie.



1. kép Film Gimp képernyőmentés – a Little Nicky készítése közben



2. kép Film Gimp képernyőmentés – Visa-hirdetés



3. kép ICY képernyőmentés – Cannon-hirdetés

3D-modellezés az And segítségével

A 3D-modellezésre az R&H egy belső modellezőeszközt használ, amelyet And and Mayának hívnak. ***Yeen-shi Chen*** modellező-TD beszél erről: „Épp egy tévéreklám számára készítek modellt egy macskáról, amelyik vizes öltönyt visel. A macska modelljét a modelltárunkból kerestük elő, és úgy módosítottuk, hogy illeszkedjen a reklámban szereplő macskára. A vizes öltözéket és a búvárfelszerelést utólag adjuk hozzá. Az egész modellt az And segítségével hoztuk létre. A modellt természetes testtartásban alkottuk meg, mert így az üzembehelyezők a csontvázat könnyebben bele tudják illeszteni."

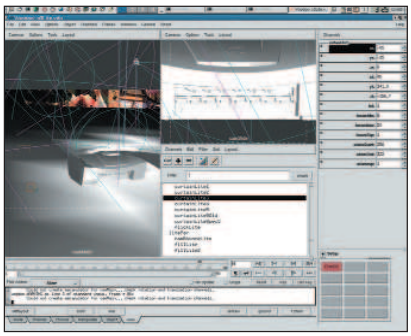
Szerkesztés az ICY segítségével

Jeff McLean, a műszaki igazgató elmagyarázza, hogyan is használja az R&H az IC-t (Interactive Compositing), a saját fejlesztésű szerkesztőjüket: „Jellemző feladat például a Scooby Doo egyik jelenetében: egy gördeszkázó karjait el kell távolítani, és egy csőben lévő Scoobyval kell helyettesíteni" (a Scooby Doo bemutatója 2002 júniusára várható). Amikor McLean élő színészt helyettesít egy Scoobyhoz hasonló animált figurával, a feladat nemcsak

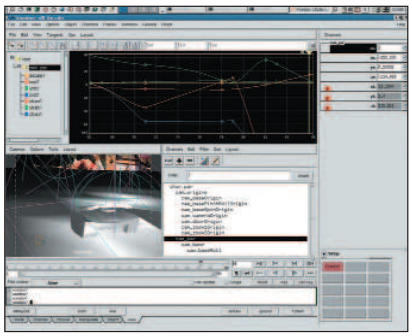
annyi, hogy lefedje a színészt, hanem azokat a helyzeteket is kezelnie kell, amelyekben a szereplő mozgása túlnyúlik a figura elfedte területen. „Helyre kell állítanom a háttér hiányzó részeit, ha egy jelenetből objektumot távolítok el” – mondja McLean.

Megvilágítás a Voodooval

Greg Yepes, a világítás vezetője egy vállalaton belül létrehozott eszközt, a Voodoot használja olyan projekteken, mint a Harry Potter vagy a Canon-hirdetések. „Amikor a jelenet 3D-objektumainak megvilágítását állítom be, a virtuális fényeket kezdetben olyan szélső helyzetre állítom be, hogy jól lássam, miről is van szó” – tájékoztat Yepes. „Később a Wrenbe ez már csökkentett értékekkel kerül.” A RenderManhoz hasonló belső leképezőegységet, a Wrent egy évvel ezelőtt ültették át Linuxra. „Minden átültetett programunk gyorsabb lett, mint amire számítottunk. DEC Alpha-gépeken kezdtem a Kutyák és macskákkal, majd jöttek az Intel PC-k és megmentettek minket” – meséli Yepes.



4. kép Voodoo képernyőmentés – Cannon-hirdetés



5. kép Voodoo képernyőmentés – Cannon-hirdetés

A FilmGimp használata

A FilmGimp egy vállalaton belül fejlesztett festőalkalmazást, az Incet váltotta fel. „A FilmGimpet a szennyeződések eltávolítására szeretjük használni” – árulja el Dahllöf. „Mindig lehet valami por, esetleg hajszál a lemezen – vagy a leolvasásból, vagy a negatívról kerül oda. A piszkot rendszerint az előző képről vagy a pillanatnyiról másolással tüntetjük el.” A FilmGimpet használják az öltöztetéseknél és a vezetékek eltávolítására, erre szolgáltat jó példát a Grincs című film. A világítók viszont a FilmGimpet a ***fur*** vezérlőállományok szerkesztéséhez alkalmazhatják. „Ezek a vezérlőállományokat az általunk fejlesztett furprogram, a Fur használja. A világítók is használják a Gimpet textúrák előállítására és képjavításra” – mondja Dahllöf.

„Ahogy a stúdió átáll Linuxra, a FilmGimpet egyre több ember a Linuxon fogja használni” – véli Dahllöf. Mivel ő azonban épp a Matador és az Illusion SGI-alapú kereskedelmi alkalmazásokat próbálja ki, a FilmGimpet általában SGI-n és nem Linuxon futtatja. „Azt vizsgálom, milyen szolgáltatásokat kell még a FilmGimphez adni, hogy egyenrangú eszközzé váljék. Megfelelő festőalkalmazásra van szükségünk Linux alá, amelyet a 2D-részlegünk majd használni tud, ugyanis nem találtunk kielégítő kereskedelmi programot” – teszi hozzá. „A Gimp semmilyen képsorműveletet nem támogatott, pedig egy 2D-alkotó számára ez igen fontos. Munkájuk tetemes részét a képkockák következőbe való átmásolása teszi ki, úgyhogy a Gimpet képkezelővel egészítettük ki” – meséli Dahllöf. Az R&H saját belső rll formátumát használja. „A FilmGimp fő jellemzője a csatornánkénti 16-bites színkezelés, ami illeszkedik az állományformátumunkhoz. A FilmGimp használatával semmilyen színveszteség nem jelentkezik, ami más 16-bites festőkészlettel esetleg előfordulhatna.”

A FilmGimp letöltése és fordítása

A FilmGimp letöltéséhez az anonim CVS-t kell meglátogatnunk. Nincs tarcsomag, rpm vagy deb formátumú változat, a programot a forrásból kell előállítani. A FilmGimp-ág neve HOLLYWOOD:

```
cvs -z3 -d:pserver:
➥anonymous@gap:/cvs/gnome
➥checkout -r HOLLYWOOD gimp
```

Úgy állítottuk be ***gap*** nevű tűzfal-számítógépünket, hogy a 2401-es kapu a ➲ http://anoncvs.gimp.org kiszolgálójára mutasson. Ha nem lettünk volna tűzfal mögött, az adatot a proxy helyett közvetlenül a `cvs checkout` parancsban adtuk volna meg.

Amikor a CVS-ről letöltöttük a 18 MB-os csomagot, a fordítás érdekében végre kellett hajtanunk néhány kisebb hibajavítást. A ***gimp/plug-ins/Makefile.am*** állományban a `SUBDIRS` változóban foglalt `rll`, `pts`, `fm_pts` és `parsley` bejegyzéseket törölnünk kell a könyvtárlistából. Ezek a kiegészítők nem épülnek be a FilmGimpbe, és megakadályoznák a fordítást.

```
cd gimp
libtoolize --force
aclocal
automake
autoconf
./configure -
➥prefix=/usr/local
make
./app/gimp
```

Áttérés Linuxra az R&H-nál

A témáról ***Mark Brown***, műszaki alelnök tájékoztat bennünket: „Jelenleg ötven linuxos asztali gépünk működik, és ez a szám 2002 végére 250-re fog nőni. Továbbá egy kétszáz csomópontos linuxos leképezőteleppel rendelkezünk, ezt igény szerint bővítjük vagy csökkentjük.” Az R&H elkészítette saját testreszabott állományrendszerét, a PTS-t a gyártási folyamat nyomkövető rendszerének támogatásához, mely az R&H az ext2 állományrendszert használja.

„Úgy döntöttünk, hogy nem saját kezűleg építjük a gépeket, hanem megrendelünk száz darab két 1,5 GHz-es AMD processzorral szerelt gépet az Angstrom Microsystemstől” – meséli Brown. „Jelen pillanatban ez tűnt a leggazdaságosabb megoldásnak. Jelenlegi asztali gépeink mindegyike kétprocesszoros Pentium III-as gép. Támogatni fogjuk a teljesen heterogén környezetet, hiszen tudjuk, hogy már a processzorok és videokártyák terén sem lehetünk teljesen következetesek.”

A FilmGimp fejlesztésének története

A FilmGimp létrejöttét az R&H és a programfejlesztő Silicon Grail támogatásának köszönheti. Mindkét cég egy évig OSS Gimp-programozót alkalmazott: az R&H ***Calvin Williamson*** GEGL-tervezőt (GIMP E Graphical Library), a Silicon Grail pedig ***Manis Singh***-t, a Gimp karbantartóját. „Túl voltunk már néhány egyéb nyílt forrású projekten, mint például meghajtóprogramok írása

Védjegy-adatok

Az And, ICY, Voodoo és a Wren elnevezések nem más, már létező védjegyekre vonatkoznak, a Rythm & Hues Studios által kizárólag belső használatra bejegyzett nevek. A cikkben szereplő Voodoo elnevezés semmilyen formában nem utal a következő védjegyekre: VOODOOtm (UNI SOFTWARE PLUS GmbH), LinuxVoodooTM (LinuxVoodoo, Inc.), VoodooTM (3dfx Interactive).

különböző filmfelvevőkhöz. Láttuk a Gimp előretörését és lehetőséget arra, hogy a Nyílt Forráskód Közösségével létrehozzunk valamit" – magyarázza ***Ray Feeney***, a Silicon Grail alapítója. ***Craig Zerouni*** a Silicon Grail RAYZ termékmenedzsere hozzáteszi: „Eltöltöttünk némi időt azzal, hogy a Gimpet mint kiegészítőt beillesszük a szerkesztőprogramunkba, a Chalice-ba, de végül rájöttünk, hogy egy Chalice-hoz hasonló eljárásalapú alkalmazásba nem igazán illik egy nem eljárásalapú rajzolóprogram." A Silicon Grail a Gimp Script-Fu nyelvét használta egy olyan FilmGimp parancssor létrehozásához, amelyet a Chalice-ban menteni lehetett. Ez a munka félbeszakadt, amikor a Silicon Grail fejlesztői egy új szerkesztőprogram, a RAYZ fejlesztésébe kezdtek. Zerouni úgy érzi, hogy egy valódi eljárásalapú nyelvre lenne szükség a rajzolóprogramban, valami olyasmire, mint a RenderMan nyelve. A Silicon Grail mostanában vásárolta meg a Kodaktól a Cineon forráskódját a Retoucher-alkalmazással együtt. „A FilmGimpen dolgozni nagyon hasznos volt számunkra, rengeteget tanultunk belőle" – mondja Zerouni.

A GEGL és a FilmGimp jövője

A GEGL, a GIMP E Graphical Library egy a GObjectsen alapuló képfeldolgozó könyvtár. A FilmGimp kifejlesztésében eredetileg az R&H-nál dolgozó ***Calvin Williamson*** segített a Silicon Grailnél alkalmazásban álló ***Ray Lehtiniemi***-nek. A Gimp következő változata az 1.4-es lesz, de a FilmGimp fejlesztése a 1.0.4. GEGL-változat ágán folytatódik, amely a 16-bites csatornákat támogatja és várhatóan – talán két éven belül – beleolvad a Gimp 2.0-s változatába. A Gimp 2.0 a FilmGimp profi jellemzőit előreláthatólag a Gimp központi részévé avatja majd. Williamson jelenlegi terve egy apró GEGL-re épülő szerkesztőprogram írása, amellyel a nagy képállományok memóriakezelését, a többszálú működést, nagy szerkesztőfákat és egyéb erőforrásigényes folyamatokat próbálhat ki.
Azok az osztályok, amelyek a képállományok és a memória kezelését végzik, le lettek választva a képfeldolgozást végző osztályokról. Ez olyan képkezelő írását teszi lehetővé, amellyel például a grafikai műveletek egyéni módon rendezhetők, vagy egyedi memória-, illetve gyorstárkezelő megoldásokat találhatnak. Az osztályok, amelyek a grafikák részeként adatokat hordoznak az operátorokról (bemenetek, kimenetek, érdekeltségi tartomány – minden külső operátorjellemző) külön lettek választva a képfeldolgozó osztályok belső operátor adataitól. Ez a grafikákat könnyebben áttekinthetővé teszi olyan feladatok számára, mint amilyen a többszálú működés megvalósítása.
„A GEGL még mindig nagyon korai szakaszban van, számos osztály vár még kidolgozásra" – mondja Williamson. „Nincs még hivatalos változat, de a forrás letölthető az anonim CVS-ről. A szerkezeten mostanában nemigen változtattam." A GEGL teljes 16-bites képfeldolgozóegység a jövőbeli Gimp és egyéb projektek számára.

Összegzés

Mind a GEGL, mind pedig a FilmGimp önként jelentkezőket vár, akik segítenének a programozásban. Williamson elmondja, hogy a PDI, az ILM, az ICT és a Sony már kifejezte érdeklődését, de programozót eddig még nem biztosítottak. Williamson szívesen fogadja a képfeldolgozó műveletek, a memóriakezelő kódok és a többszálú működés iránt érdeklődő programozókat, és várja a GEGL projekthez csatlakozni kívánók jelentkezését. „A könyvtár kiegészítése és a képműveletek kódolása igen sok időbe telik" – jegyzi meg Williamson. „A GEGL a Gimp jövőjének igen fontos része." Azoknak a programozóknak, akik a GEGL határidejét túl távolinak találják, Williamson azt ajánlja, hogy egy évig a FilmGimp fejlesztésében segédkezzenek. „A FilmGimpet olyan eszközök hozzáadásával szeretnénk továbbfejleszteni, amelyek a felhasználók számára lehetővé teszik a képsorozatokon való műveleteket. Szeretnénk egyszerűsíteni az egyik képről a másikra való rajzolás és másolás folyamatát" – mondja Dahllöf, a FilmGimp karbantartója.
Az R&H saját flipbook-lejátszót, a Flickset használja. Ez a szolgáltatás, amellyel a felhasználó a villogást észlelheti, hiányzik a FilmGimpből. A felhasználók szűrőket is használnak, de ezek közül néhány, amely a FilmGimpben is megtalálható, a mozgófilmes munkáknál haszontalan. A felhasználók a szűrők nagyobb fokú beállítási lehetőségét igénylik."
A nyílt forrású programok tárgyában ***Green***, az R&H vezető programmérnöke a következőket mondja: „Nagy vita folyik a FilmGimpre fejlesztett programjaink forráskódjának felszabadítása körül. A legtöbb program használata a kívülállók számára meglehetősen bonyolult lenne, hiszen terméknyomkövető rendszerünk, a PTS köti őket. Enélkül semmi sem fog működni."
Green szerint az egyetlen érv, amely mellette szól, a betanítás magas költségeinek mérséklése. Az R&H új alkalmazottai az első hónapot kizárólag tanulással töltik. Sokat segítene, ha az egyetemet olyan tehetségek hagyhatnák el, akik már az R&H eszközein gyakorolhattak.

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.

Kapcsolódó címek

BEAST/BSE ➲ http//:beast.gtk.org
FilmGIMP ➲ http//:film.gimp.org
GEGL ➲ http//:www.gegl.org
gegldev egroup ➲ http//:gegldev@gegl.org
Gimp ➲ http//:www.gimp.org
Gimp-film News Group ➲ http//:https://lists.xcf.berkeley.edu/mailman/listinfo/gimp-film
Rhythm & Hues ➲ http//:www.rhythm.com
Silicon Grail ➲ http//:www.sgrail.com

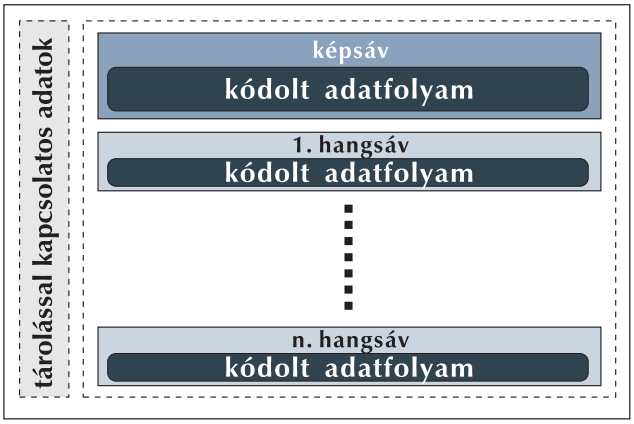
Digitális videózás (2. rész)

Alapvető filmformátumok – és ahogyan ezt az MPlayer látja.

Reményeim szerint sorozatunk előző részében sikerült átfogó képet adnom az MPlayer lejátszóról. Az áttekintések azonban az esetek jelentős részében természetükből fakadóan felületesek, és mivel jelenleg egy igen összetett témát taglalunk, ez most különösképpen igaz. Ezen a felületességen szeretnék változtatni a sorozat következő részeiben. A linuxos videolejátszás mellett a digitális mozgóképek világát is be szeretném mutatni – mindjárt az elejétől kezdve. Éppen ezért, hogy az alapokkal tisztában legyünk, inkább a fájlformátumokról, és kevésbé az MPlayerről fogok beszélni. Abban a reményben kezdek neki beszámolómnak, hogy aki az előző cikkben ismertetett nem túl egyszerű telepítési folyamatot túlélte, a sikerélmény mellett mélyebb érdeklődésre is szert tett, ezért szívesen fogadja az itt leírtakat.

Értem én, hogy benzin, na de mitől megy?

Ha azt mondom, hogy film a számítógépen, akkor sokan csupán a megjelenő képkockákra gondolnak, ám ahhoz, hogy mindez lehetővé váljon, feltétlenül szükség van egy digitálisan tárolt mozgó képsorra. Jelen esetben a tároláson van a hangsúly. Amióta csak a digitális filmezés gondolata megfogalmazódott, a kérdéskör alapproblémája, hogy digitalizált filmjeinket miként tároljuk. Az ilyen formában, tehát digitálisan történő tárolás egyik nagy hátránya hatalmas helyigénye. Ebből következően a rögzítés egyetlen járható útja, hogy a tárolt adatot tömörítjük (általában veszteséggel), majd valamilyen formában tároljuk. Kezdetben ehhez kapcsolódóan sem szabvány, sem eljárás nem létezett, ezért szükségessé vált a megalkotásuk. Gond csupán abból adódott, hogy az élénk érdeklődés és az egymással folytatott versengés miatt többen is digitális mozgóképformátumok fejlesztésébe kezdtek, amelyet igyekeztek a saját arculatukra szabni. A legtöbben egymástól eltérő szempontok szerint hoztak létre tömörítési eljárásokat, tárolási módokat, melynek eredményeképp ma egyfajta káosz övezi a mozgóképtárolással kapcsolatos kérdéskört. Ennek átláthatóságát úgy javíthatjuk, ha a tárolás miként-



1. ábra A médiatartály logikai felépítése

jét két fő részre osztjuk. Az egyik része az adatok kódolása (tömörítés), a másik pedig a kódolt adatok tárolása. Fontos különbséget tennünk köztük és élesen elválasztani őket egymástól, valamint megértenünk, hogyha digitális filmformátumról beszélünk, az nem csak a lejátszáshoz szükséges kodeket jelenti. Egy fájlformátumon belül többféle kodeket is alkalmazhatunk. Gondoljunk csak bele: egy AVI (ez a tárolási mód) képkockái akár Indeo, DivX3, DivX4 stb. kodekkel is tömörítve lehetnek, tartalmazhat akár WAV, MP3, AC3 stb. formátumú hangot – ezek azonban általában kívülről nem látszanak, a lejátszók ugyanis a tömörítési eljárást átlátszóvá teszik számunkra. Mi csupán annyit érzékelünk, hogy AVI-val van dolgunk, amely – mint tárolási formátum – a rá jellemző sajátosságokkal rendelkezik. Ebből már kiderül, hogy egy filmformátum a tárolási mód és az alkalmazható kodekek kombinációjaként jön létre, nem csoda hát, hogy közel százfélével is találkozhatunk. Ezek az utóbbi időben csoportokba kezdenek rendeződni, egyes kodekeket csak bizonyos formátumok esetén alkalmaznak, valamint minden fájlformátumhoz egy szűkebb kodekcsoportot kapcsolhatunk, és ezáltal rendszerezettebbé, áttekinthetőbbé válhat számunkra ez a zűrzavar.

Médiatartályok

Itt az ideje, hogy nevén nevezzük a dolgot: a médiatartály (media container) mint legnagyobb logikai tárolóegység szerepel a filmek terén, és alapvetően meghatározza a képsor tulajdonságait, a lejátszás módját és lehetőségeit (*1. ábra*). A tartály formátuma egyrészt jellemző az adatok tárolásának módjára (blokkos, lineáris stb.), az egyes sávok (egy fájlon belüli adatfolyamok) egymáshoz való kapcsolatára, összerendelésük módjára; másrészt a film sajátosságaira vonatkozó adatokat tartalmazza, például a felbontást, a képarányt, az időzítéseket, a képkockaszámot és a képek futamidejét (vagy olyan adatokat, amelyekből ezek kiszámolhatók). Ez természetesen nem azt jelenti, hogy minden formátum az összes ilyen adatot magában foglalja, mindegyikben csak a formátumra jellemző értékek találhatók.

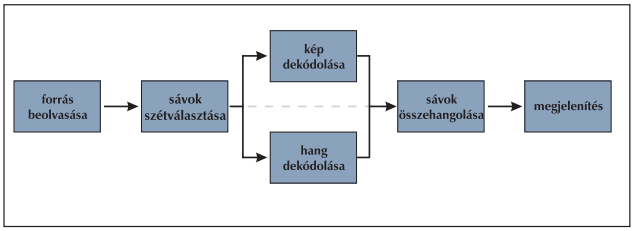
A fentiekből látható, hogy egy ilyen médiatartályt átgondoltan és megfontoltan kell kialakítani, hiszen az előre nem látható nehézségekkel is számolni kell. A téma sokrétűsége következtében általában egy előre kitűzött célnak (ezt a későbbiekben látni fogjuk) próbálják megfeleltetni.

A legelterjedtebbek a Microsoft fejlesztésű AVI (Audio-Video Interleaved) és az MPEG (Motion Picture Expert Group) (➲ http://www.iso.org/iso/en/commcentre/pdf/MPEG0009.pdf) által készített MPEG-1, MPEG-2 stb. formátumok. Az AVI-szabvány a tárolt adatokat illetően szinte szabad kezet ad, a fejlécben megadott formátumú kép és hangsávokat tartalmaz párhuzamosan, mely sávok típusa igen sokféle lehet. Ezzel szemben az MPEG formátumai általános fejléc nélküli, blokkokból álló adathalmazok, a képre, hangra jellemző adatok pedig az egyes blokkok fejléceiben vannak tárolva. Az MPEG eredetileg a VideoCD-

szabvány kifejlesztéséhez jött létre, ennek következtében itt nem csupán egy médiatartályról beszélhetünk – ebben az esetben a fájlformátum az alkalmazott kodekkel szorosan összekapcsolódik. Később azonban látni fogjuk, hogy az MPEG-szabványok esetében is szétválasztható a tartalmazó objektum és a használt kodek, ugyanis a későbbi MPEG-vívmányok sem szakítottak az MPEG-1 fájlformátumának alapvető hagyományaival, inkább csak kiterjesztették azt.

Lejátszás

A későbbi érthetőség szempontjából leírom, hogy a videolejátszók hogyan nyerik vissza a bonyolult módon kódolt, tömörített, összerendelt adatfolyamból az eredeti mozgó képsort. A leírás alapját az MPlayer által alkalmazott folyamat képezi, de szinte az összes egyéb lejátszó is ugyanezt a stratégiát alkalmazza.



2. ábra A videolejátszás folyamata

Először is abból a tényből adódóan, hogy a fájlok sávokra oszlanak, beolvasás után ezeknek a sávoknak a szétválasztása (demultiplexing) következik. Ez a legtöbb esetben a kódolt kép és hang szétválasztását jelenti, amely már a médiatartályra jellemző módon történik.
Az eredményül kapott két vagy több adatfolyam a megfelelő kodekek bemenetét képezi. A kodekek úgynevezett fekete dobozokként vesznek részt a folyamatban. Ez azt jelenti, hogy a lejátszó nem ismeri a kódolási, tömörítési eljárásokat, a használandó kodeket a fejlécek alapján azonosítja be. A többi már a kiválasztott dekódoló modul teendője. Ezt követően igen jelentős feladatnak számít az egyes sávok (például a kép- és a hangsáv) összehangolása. Ne feledjük, hogy az egyes leképező eszközök (hangkártya, videokártya) további átmeneti tárat használnak, ami szintén megnehezíti az összehangolást (*2. ábra*).

Az MPEG mint médiatartály

Minden MPEG-objektum néhány képkockából álló csoportokból épül fel, amelyek teljesen függetlenek egymástól. Ennek eredményeképp az MPEG-fájlok „streamelhetők”, ami azt jelenti, hogy a film folyamatos lejátszásához pusztán a pillanatnyi és a néhány következő blokk is elegendő, tehát lehetővé válik a távoli fájlrendszerről történő lejátszás, vagy egy sugárzott adásba történő bekapcsolódás. Ez a tény másfelől hibavédelmet is nyújt.
Az MPEG némileg ellentmond az idáig felépített filozófiának, hiszen ezeket a fájlokat a tömörítési eljárásokkal együtt fejlesztették ki, tehát a médiatartály és a kodek fogalma valamennyire mégis összekapcsolódik, logikailag azonban elkülönülnek egymástól. Az eddigiekben az MPEG-ről mint médiatartályról beszéltem, most nézzünk belé egy kicsit! Háromféle típusa terjedt el igazán: a VCD (VideoCD) vagy szabványos MPG fájl, az SVCD (Super VideoCD) és a DVD. Mindhárom formátum nemzetközi szabvány. A VCD MPEG-1, az SVCD és a DVD MPEG-2 (az elnevezés itt arra utal, hogy az MPEG-2 formátumhoz kifejlesztett kódolási eljárásról van szó) kódolást alkalmaz a képanyagra, ám a hangkodekek terén már nem ilyen egységes a helyzet. A VCD-kben, MPG-ekben a hang tömörítésére MPEG layer1, layer2, layer3-at egyaránt használnak, az SVCD szabvány az ilyen alakú filmek számára a layer2-es hangot írja elő, míg a DVD-k leginkább AC3 hangot tartalmaznak (néha azonban layer2-es hangsáv is előfordul). Mint látható, az MPEG-fájlok, mivel szabványokhoz kötődnek, viszonylag átlátható rendszert alkotnak: az egyes típusokhoz az alkalmazandó kodekek jól behatárolhatók.
Az MPlayer gyakorlatilag az összes MPEG formátumú fájlt képes lejátszani – a beépített libavcodec segítségével.
Az ffmpeg12 videokodek mindkét videosávot (MPEG-1, 2) le tudja játszani. Az ffmp3 segítségével az MPEG layer1, 2, 3 kódolású hangsávok, míg az ac3 hangkodek segítségével a DVD-khez tartozó hangsávok válnak elérhetővé (ez abból a szempontból érdekes, hogy a libavcodec család nyílt forrású, ez azonban nem mondható el mindegyik kodekről).

Mit tud az AVI?

Ez a formátum egész más filozófiát követ, mint az MPEG, a két kategória igazából nem összehasonlítható. Az AVI-t eredendően nem a kész filmek végleges tárolására, sokkal inkább feldolgozásra, vágásra használják. Az természetesen már más kérdés, hogy az új DivX kodekek segítségével jelentős méretcsökkenés érhető el, ezzel számolva filmjeinket már érdemes így tárolni.
A fájl egyetlen fejléccel rendelkezik, az ebben felsorolt jellemzők határozzák meg a lejátszás módját. Alapvetően két formája létezik: „interleaved” és „non-interleaved”. Az elsőben a kép- és hangsávok összefésülten helyezkednek el, a másikban egymás után folytatólagosan. Egy videofájl legfeljebb 2 GB méretű lehet és 99 hangsávot tartalmazhat. A kodekekre szinte semmilyen megkötés nem vonatkozik, a kép és a hang bármilyen módon tömörítve lehet. Az időzítés is többféle módon oldható meg: az egyik eljárás, amit a legtöbb helyen alkalmaznak, a fejlécben megtalálható bitráta/mintavételezési ráta értékén alapul. Hátránya, hogy a hibásan létrehozott fájlok lejátszásánál a kép és a hang elcsúszhat egymástól, és ezzel elrontja a filmet. A másik módszer ritkábban használatos, de az előző hibával is megbirkózik. Ez az összefésülés alapú időzítés. Az eljárás nem a fejlécben megadott értéket használja, hanem az összefésült audio- és videocsomagokból viszonyított elhelyezkedést számol. A kép tömörítésére jelenleg leggyakrabban a DivX, DivX4 kodekek használatosak, a hangsáv pedig legtöbbször MPEG layer3 vagy AC3 formátumban található.
Az MPlayer is elboldogul szinte bármelyik kódolási eljárással, sőt mindkét időzítési módot ismeri. Alapértelmezetten a DivX formátumokat (libavcodec család), valamint a fent említett hangformátumokat is le tudja játszani.
Ezzel természetesen nincs vége a listának, számos támogatott fájlformátum ismertetése megtalálható még a program leírásában, sőt, a technikai részben igen érdekes adatokat találhatunk a lejátszó tényleges működéséről – a lejátszás folyamatától egészen a programban alkalmazott függvények leírásának szintjéig! Bátran ajánlom mindenkinek, akit a digitális mozgókép-visszaadás részletekbe menően érdekel.

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

Könnyű álmok (12. rész)

Az előző számban megjelent cikkünkben megkezdtük az ismerkedést az Interneten használt legfontosabb protokollokkal. Most ezt az utat járjuk tovább.

Először tekintsük át a levelezésnél használt legfontosabb protokollokat! A levelek küldésére és fogadására az SMTP (Simple Mail Transfer Protocol) szolgál. Az Interneten elhelyezkedő kiszolgálók e protokoll segítségével fogadják és továbbítják a leveleket. Nem minden számítógép rendelkezik azonban megfelelő erőforrásokkal (lemezterület, folytonos internetkapcsolat, állandó IP-cím) ahhoz, hogy kihasználhassa ezt a lehetőséget. Ezért két általánosan elterjedt protokoll is létezik, amelyek kisegítő szerepet játszanak. Az egyik a POP3 (Post Office Protocol – Version 3), amely a levelek letöltését teszi lehetővé, a másik pedig az IMAP4 (Internet Message Access Protocol – Version 4), ez még szélesebb körű szolgáltatást nyújt.

Mielőtt a protokollok ismertetésébe belekezdenénk, tekintsük át, hogy a levelezés milyen veszélyeket is jelent a hálózatunkra nézve. A legismertebb veszélyről már mindenki hallott: ez a vírusok terjedése. A vírusok a felhasználók képzetlenségét vagy kényelemszeretetét kihasználva terjednek, és károkat okoznak a számítógépeken. Sok elterjedt levelezőprogramban sajnos folyamatosan súlyos biztonsági hibákat találnak, amelyeket a támadók vagy a féregprogramok (worm) kihasználhatnak. Ilyenek lehetnek például az érvénytelen MIME-csatolások vagy a túl hosszú fejlécek hibás kezelése. Ezen hibák ellen a csomagszűrés-alapú megoldások nem nyújtanak védelmet.

A cikkben szereplő példákat ismét képzeletbeli tűzfalunkra írtuk, ahol az `eth0` csatoló a védett, míg az `eth1` csatoló az Internet felé néz. Az elozó cikkhez hasonlóan itt is csak az INPUT láncon mutatjuk be a beállításokat. Feltételezzük, hogy 2.2.x rendszermag esetén a nem SYN-es csomagokat a csomagszűrő szabályaink elfogadják. 2.4.x rendszermagsorozat esetén a rendszernek az ESTABLISHED állapotú csomagokat el kell fogadnia.

SMTP

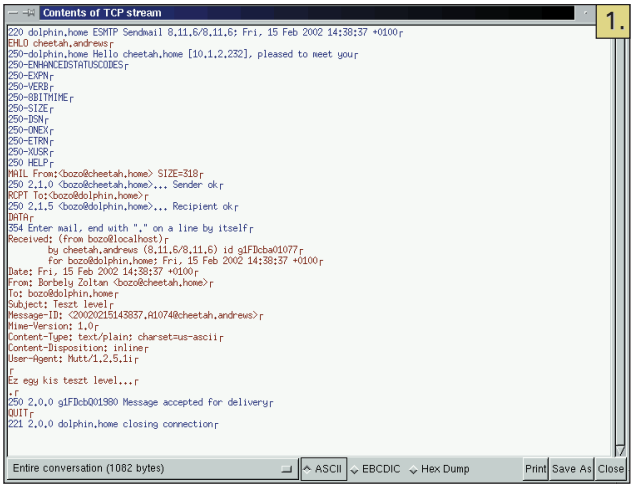
Az SMTP-protokoll egyike az Internet legrégibb protokolljainak, feladata a levelek továbbítása és fogadása. Az SMTP szövegalapú protokoll, ami TCP-t használ. A kiszolgáló bejegyzett kaput használ, ami a `25/tcp`. Az SMTP-protokoll `store & forward` (tárol és továbbít) alapú. Ez azt jelenti, hogy amelyik SMTP-kiszolgáló veszi a levelet, az vagy a felhasználó postaládájába helyezi, vagy tárolja és a címzett felé továbbítja. Amikor a kézbesítés sikertelen (például a címzett nem létezik, vagy a beállított időkorláton belül sem sikerült a levelet továbbítania), akkor a hibáról egy levelet küld a feladónak, majd a kézbesíthetetlen levelet eldobja. A hibalevél feladója egy különleges cím, a <> (azaz a cím üres). Sajnos, néhány vírus ellen védekezésnek „köszönhetően” sokan anélkül letiltották az ilyen levelek fogadását, hogy a következményekbe belegondoltak volna. Így nemcsak a vírus által küldött leveleket dobják el, hanem az MTA-k által küldött hibaleveleket is. A `store & forward` működés miatt az egyes SMTP-kiszolgálók (szaknyelven gyakran MTA-knak – Message Transfer Agentnek hívják őket) natív proxyként képesek működni. Amikor egy MTA levelet kíván küldeni, a DNS-kiszolgálótól megkérdezi az adott tartomány levelezőkiszolgálóinak címeit. Ezek a címek vagy MX [1.], vagy A rekordokban vannak tárolva. A címek közül választ egyet (egy tartomány általában több levelezőkiszolgálóval is rendelkezik, így az elsődleges kiszolgáló elérhetetlensége esetén a levelek nem a feladó SMTP-kiszolgálóján várakoznak), majd TCP-kapcsolatot kezdeményez a megadott levelezőkiszolgáló 25-ös kapujával, ahol egy MTA fogadja. Az elküldendő leveleket továbbítja, majd zárja a kapcsolatot. Erre láthatunk példát az *1. képen*.

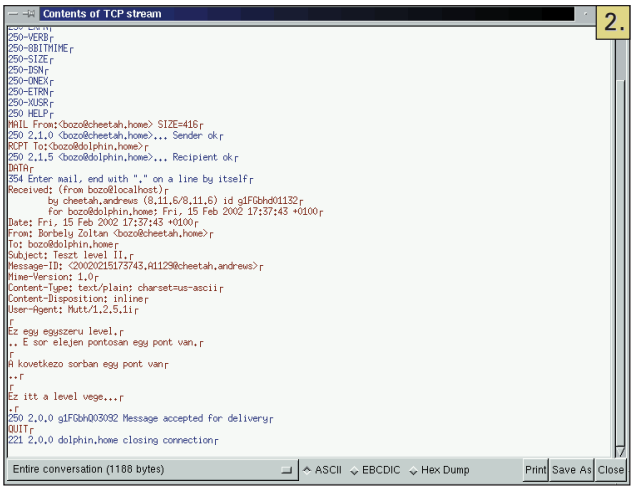
A kép az `etheral` program [2.] segítségével készült, és az ügyfél (***cheetah***) és a kiszolgáló (***dolphin***) között zajló TCP-forgalmat mutatja. Az ügyfélgép üzenetei pirossal, a kiszolgáló üzenetei kék színnel jelöltek. A protokoll menete a következő: az ügyfél TCP-kapcsolatot nyit a kiszolgáló 25-ös kapujára, ahol a kiszolgáló egy úgynevezett „greeting” üzenettel várja. Válaszul az ügyfél azonosítja magát, ami az EHLO vagy HELO üzenettel történhet, erre a kiszolgáló válaszol. Az EHLO üzenet esetén a kiszolgáló közli az általa támogatott protokollkiegészítéseket. E szakasz után következik a levelek tényleges elküldése. A levél küldését a `MAIL FROM` parancs jelzi (az SMTP-protokoll a kisbetűket és a nagybetűket a parancsokban nem különbözteti meg). A `MAIL FROM` parancs paramétere a `reverse path`, mely a küldő levélcíme. Mint látható, az SMTP MTA a küldőt elfogadta, ezt jelzi a 250 kódú válaszüzenet. Ezt követi a címzett(ek) megadása, jelenleg csak egy címzettet adtunk meg. A címzettek az `RCPT TO` parancs(ok) paramétereiben helyezkednek el. A ***dolphin*** gép elfogadta a címzettet. Miután a címzetteket megadtuk, a levél tartalmának átvitele következik. A levelet a `DATA` parancs vezeti be, és egészen addig tart, amíg egy sorban csak egy pont (`"."`) karakter áll. Mint láthatjuk, a levél első része a fejléc, ezt követi a levéltörzs. A fejlécet a törzstől egy üres sor választja el.

A levél lezárása után látható, hogy a kiszolgáló a levelet elfogadta. Mivel a ***cheetah*** gépnek most nincs több elküldendő levele, a `QUIT` paranccsal a kapcsolat bontását kezdeményezi. Ezután a TCP-kapcsolat lebomlik, a levéllel kapcsolatos további teendők (például a felhasználó postaládájába helyezés vagy a továbbküldés) már a ***dolphin*** gép feladata.

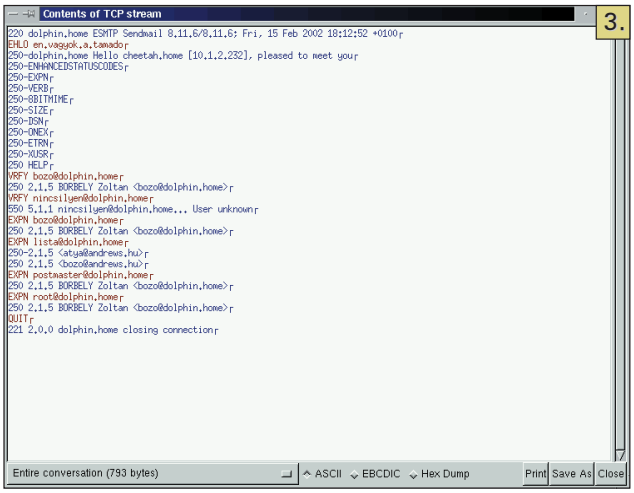
A `DATA` parancs után említettük, hogy a levél végét egy olyan sor jelzi, ami csak egy pontot tartalmaz. Mi történik akkor, ha olyan levelet írunk, ahol ez szintén előfordul? Erre láthatunk példát a *2. képen*. Itt jól látható, miként oldja meg ezt a gondot az SMTP-protokoll. A küldő program – amennyiben a sor elején pontot talál – még egy pontot szúr be elé. Így a sor elején álló pont a karaktertovábbítás folyamán mindig megkétszereződik. A vevő-MTA a sor elején álló pont felismerése után megvizsgálja, hogy sorvége következik-e (ez a levél végét jelentené), vagy más karakter található. Ha nem sorvége jelet talál, a plusz pontot eltávolítja. Ennek köszönhető, hogy

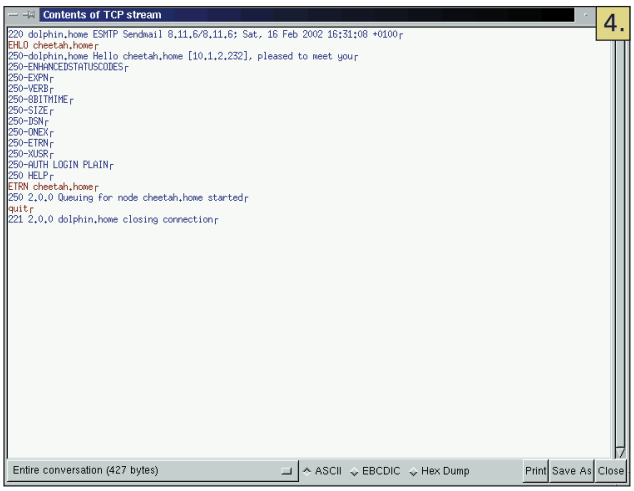
a címzett pontosan ugyanabban a formában kapja meg a levelet, ahogy a feladó azt megírta.
Most nézzük meg, hogy az általános levelezési kockázatok mellett milyen SMTP-re jellemző veszélyek leselkedhetnek még ránk. Az SMTP MTA-k egyedi parancsokkal bírnak ahhoz, hogy egy postaláda meglétét ellenőrizzék, vagy egy alias (nem valódi postaláda – a neki címzett levelet más címzett(ek) kapják





meg) tagjait felderítsék. Ezek hasznosak lehetnek hibakereséshez, azonban a támadók vagy a levélszemétcím-gyűjtők (akik az Internetről gyűjtik a levelezési címeket, amelyeket a szemetelők számára árulnak is) ugyancsak kihasználhatják. A postaláda létezésének ellenőrzése a támadók számára lehetővé teszi, hogy a rendszeren létező felhasználókat feltérképezzék. Ennek oka, hogy a felhasználói név általában megegyezik a postaláda nevével. Az alias kifejtése segítségével a támadó kideritheti, hogy ki üzemelteti a számítógépet. Elég csak megnézni, ki kapja a `root` vagy `postmaster` leveleit (alapszabály, hogy a `root`-felhasználó nem levelezik, hiszen egy levelezőügyfél hibája végzetes következményekkel járhat a rendszer biztonságára nézve). A `VRFY` és `EXPN` parancsok kihasználására mutat példát a *3. kép*. Itt a `telnet` parancs segítségével kapcsolódtunk a ***dolphin*** levelezőkiszolgálójára, majd kézzel adtuk ki a parancsokat. Ezeket a parancsokat a kiszolgálóban célszerű kikapcsolni. Minden korszerű MTA képes e parancsok letiltására. Mint a protokollból is látható, az elektronikus levelek hamisítása roppant egyszerű feladat. Soha ne bízzunk meg egy hagyományos elektronikus levélben! A hamisítás ellen a legegyszerűbb és legjobb megoldás a levelek digitális aláírása. A legtöbb linuxos levelezőprogram (MUA – Mail User Agent) képes együttműködni valamilyen fejlett titkosító programmal (például gpg, pgp stb.). Napjainkban egyre nagyobb gondot jelentenek a szemetelők (spammer), akik kéretlen reklámjaikkal folyamatosan bombáznak bennünket. Tökéletes megoldás sajnos nem létezik ellenük, most csak arra vonatkozóan adunk néhány tippet, miként akadályozhatjuk meg, hogy a reklámokat a gépünkön keresztül küldjék. Az Interneten az SMTP MTA-k régi időkben bárkitől bárki számára elfogadtak levelet. Ezt a szemetelők alaposan ki is használták. Manapság ez a beállítás veszélyessé vált, hiszen a szemetelők felfedezik és reklámleveleiket az áldozat kiszolgálóján át továbbítják, ami jelentős hálózati terhelést és tekintélyveszteséget okozhat.





Az ilyen MTA-kat a szaknyelv `open relay`-nek hívja. A levéltovábbítást csak a védett hálózat számára célszerű engedélyezni, illetve azonosításhoz kell kötni.
Az Internet elterjedésével megjelentek olyan kisebb cégek, akik SMTP-alapú levéltovábbítást (számukra a POP3 vagy az IMAP4 nem jelentett megoldást) igényeltek, de nem rendelkeztek állandó internetkapcsolattal. Az ilyen szükséglet kielégítésére fejlesztették ki az ETRN-protokollbővítést, ami a `TURN` parancs (az ügyfél és a kiszolgáló viszonyának megfordítása) korszerűbb és biztonságosabb változata. Amennyiben a levél hagyományosan nem továbbítható, az SMTP MTA várakozási sorá-

ban marad, és a kiszolgáló meghatározott időközönként ismételt továbbítást kísérel meg. Az ideiglenes kapcsolattal rendelkező gépek számára (például telefonos kapcsolat) a továbbítási kísérletek közötti több óra túl hosszú, a próbálkozások idejének csökkentése pedig a kiszolgálóknak jelent felesleges terhelést. Az `ETRN` parancs célja, hogy az ügyfélgép Internetre csatlakozásakor a megadott kiszolgálókhoz fordul, és kéri őket, hogy kezdeményezzék a nekik szánt levelek azonnali továbbítását. Erre láthatunk példát a *4. képen*. Ahogy a képen is látszik, az ügyfél (***cheetah***) arra kéri a kiszolgálót (***dolphin***), hogy a ***cheetah.home*** tartományra vonatkozó leveleket küldje el számára. A levelek továbbítása a hagyományos SMTP-protokollon, egy független TCP-kapcsolaton át történik. Az `ETRN` használatához általában a DNS-be bejegyzett név szükséges, így csak az állandó IP-című ügyfelek használhatják.

A korszerű SMTP MTA-k egyre fejlettebb módszerekkel rendelkeznek: lehetővé teszik az azonosítást, valamint a köztük levő csatorna titkosítását is. Az azonosítás a kapcsolódó MTA vagy felhasználó kilétének megállapítására szolgál, célja pusztán a levél továbbíthatóságának igazolása. Ez azonban semmilyen védelmet nem jelent a levelek hamisításával szemben.

A levelek illetéktelen elolvasása ellen a csatorna titkosítása nem tökéletes megoldás, hiszen csak az adott csatornára vonatkozik. Amennyiben a levél végcélja nem ez az MTA, a levél a következő továbbítási lépésnél akár kódolatlanul is közeledhet.

A levél a köztes MTA-kon és a postaládában eredeti formájában található meg, tehát az így nyújtott védelem nem azonos magának a levélnek a kódolásával. A csatorna titkosításának elsődleges célja, hogy az azonosítási adatok ne kódolatlanul kerüljenek a hálózatra.

Az SMTP-levelezést nem célszerű pusztán csomagszűrés segítségével védeni. A legjobb módszer, ha a DMZ-ben (a DMZ – a DeMilitarized Zone határvédelmi fogalom: olyan alhálózatot hívunk így, amely sem védett hálózatunknak, sem az Internetnek nem része, lásd még Linuxvilág, 5. szám 40. oldal.) elhelyezzük a levelek továbbítását végző kiszolgálót, amely nem tartalmaz helyi kézbesítést (azaz nincsenek rajta postaládák). Az ilyen kiszolgáló célja, hogy az Internetről beérkező leveleket a cég belső levél kiszolgálóinak ossza szét, és a belülről jövő leveleket az Internetre továbbítsa. Ezen a kiszolgálón fontos a levelek fejlécét és tartalmát is vizsgálni, így a támadások egy része (például túl hosszú a levélfejléc vagy vírusos a levél) kiszűrhető. Hasznos, ha az SMTP MTA elkülönítőben (jail) foglal helyet. Az elkülönítők (jail) célja, hogy az esetlegesen bejutott támadót megakadályozza a kiszolgáló egyéb alrendszereinek támadásában. Ilyen környezet kialakításáról sorozatunk későbbi részeiben fogunk szót ejteni.

Kis cégek esetén nem mindig engedhető meg egy csak erre a célra fenntartott gép használata. Ilyenkor az SMTP-továbbítási feladatot a tűzfalra is bízhatjuk, de a beállításoknál fokozott gonddal kell eljárnunk.

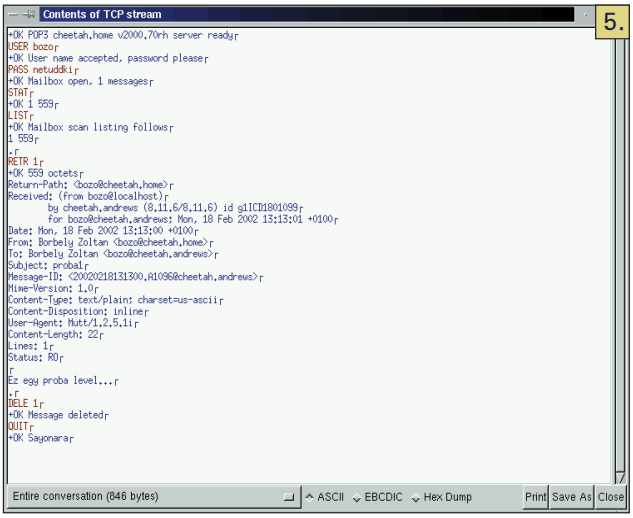
A helyi postafiókokba történő kézbesítés és egyéb felhasználói szolgáltatások (például ***~/.forward*** jellegű állomány, ahol a levél célja megváltoztatható vagy program indítható) szinte semmilyen esetben sem célszerűek. A jogokat a levéltovábbításhoz az elengedhetetlen minimumra kell csökkenteni, az MTA-t pedig elkülönítőbe (jail) kell zárni.

Ha feltételezzük, hogy a tűzfalunkon fut az SMTP MTA, a csomagszűrőben engedélyezni kell a `25/tcp` kapu elérését. Ezt a védett hálózat számára elérhetővé kell tenni, hogy a tűzfalon át SMTP-vel levelet kaphasson.

Amennyiben a külvilágból a leveleket SMTP-protokollon át kapjuk (állandó IP-címmel rendelkezünk), akkor az onnan érkező SMTP-kapcsolatokat is engedélyezni kell. Ha a külvilágból POP3- vagy IMAP4-protokoll segítségével töltjük le leveleinket, szerencsés, ha a kiszolgálónk SMTP-kapuja kívülről nem érhető el. Amennyiben a gép levélfogadást és -továbbítást nem végez (csak leveleket küld), tanácsos, hogy az adott gépen ne fusson MTA. A levelek küldésekor a legtöbb MUA a ***/usr/lib/sendmail*** programot elindítva továbbítja a levelet. Az MTA-nak nem feltétlenül a ***Sendmailnek*** kell lennie – kompatibilitási okokból szinte bármelyik MTA tartalmaz programot ezen a néven. A név egyszerűen „történelmi” okokból alakult így.

Fontos figyelmet fordítanunk egy apróságra, amit sokan elfelednek: a levelet fogadó MTA-k sok esetben visszakapcsolódnak a küldő `auth`-kapujára (más néven `ident`, ez a `113/tcp` kapu), amely a kapcsolatot létesítő felhasználó kilétének megállapítására szolgál. A protokoll nagyon régi és egyáltalán nem biztonságos. Nem biztos, hogy egy lehetséges támadóra tartozna, hogy egy hálózati kapcsolatot kezdeményező program milyen felhasználóként fut. Ha azonban az ilyen kapcsolatokat a `DENY`-szabály segítségével eldobjuk, a levél továbbítása előtt a címzett MTA hosszan várakozhat. Ezért célszerű, hogy az `auth`-kapura érkező kéréseket utasítsuk el (lehetőleg `TCP RST`-csomaggal).

Ezt kétféle módon tehetjük meg: vagy a csomagszűrőből utasítjuk el (ez a biztosabb), de az RST-elutasítás használatára



csak a 2.4.x rendszermagsorozat esetén nyílik lehetőségünk, vagy meggyőződünk arról, hogy gépünkön nem fut az `auth`-szolgáltatás, és az `auth`-kapura beérkező kapcsolatokat elfogadjuk. A második megoldás azért lehet veszélyesebb, mert a szolgáltatás a későbbiek során „véletlenül” (például valamelyik későbbi csomag telepítésfüggő csomagként felhozza) települhet és elindulhat.

Amennyiben nem ragaszkodunk a TCP RST-elutasításhoz (így a kapcsolódó gép láthatja, hogy a TCP kaput csomagszűrő védi), úgy használhatjuk az ipchains REJECT akcióját is. Az alábbi példabeállítással csak a védett hálóról fogadunk el leveleket.

2.2.x rendszermagsorozatnál:

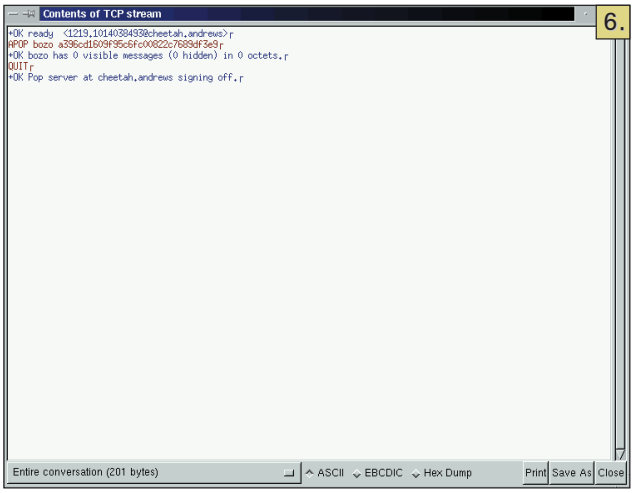
```
ipchains -A input -i ! eth0 -d 0.0.0.0/0
➥smtp -p tcp -j DENY
ipchains -A input -i eth0 -d
➥tuzfal_belso_ip_cime smtp -p tcp -j ACCEPT
ipchains -A input -d 0.0.0.0/0 auth -p tcp
➥-j REJECT
```

2.4.x rendszermagsorozatnál:

```
iptables -A INPUT -p tcp -i ! eth0 --dport
➥smtp -j DROP
iptables -A INPUT -p tcp -i eth0 -d
➥tuzfal_belso_ip_cime --dport smtp -j ACCEPT
iptables -A INPUT -p tcp --dport auth
➥-j REJECT --reject-with tcp-reset
```

POP3

A POP3-protokoll lényege, hogy egyszerű ügyfélgépek (például asztali számítógépek) számára is lehetővé tegye a levelezést (a POP3-protokoll nem támogatja levelek küldését).
A levél ilyenkor SMTP-protokollon keresztül egy kiszolgálóra érkezik, ahol a felhasználó postaládájába kerül. A POP3-protokoll célja, hogy a felhasználó postaládájában levő leveleket letöltse, majd letörölje onnan. A protokoll által nyújtott szolgáltatások köre igen behatárolt, a fő cél az egyszerű megvalósíthatóság volt. Aki szélesebb körű támogatást szeretne



6.

(például mappák kezelése), az IMAP4-protokollt használja.
Az *5. képen* egy a POP3-ügyfél és -kiszolgáló közti jellemző párbeszédet mutatunk be.
Ebben a példában a ***cheetah*** nevű gép a kiszolgáló (kék színnel), míg a ***dolphin*** az ügyfél (piros színnel jelölve). A kiszolgáló itt is üdvözlő sorral fogadja az ügyfelet, minden válasz egy állapotjelzővel (`+OK`, `-ERR`) kezdődik, és ezt követi az üzenet.
A belépés a `USER` és `PASS` parancsokkal történt. Mint látható, ez esetben a felhasználó neve és jelszava minden védelem nélkül kerül továbbításra. Belépés után a `STAT` paranccsal megnéztük, hogy hány levél vár ránk (1 levél 559 bájt hosszan). Ezután a `LIST` parancs hatására a levelek listáját is megmutatja. A sorszámozás eggyel kezdődik, a sorszámok kiosztását a POP3-kiszolgáló a belépéskor végzi el. Ezekután letöltjük az első levelet, majd le is töröljük. A törlés ilyenkor még nem hajtódik végre, csak miután kilépünk (`QUIT`). Fontos megfigyelni, hogy a levél hogyan kerül továbbításra. Ha jobban megnézzük, itt is az SMTP-protokoll esetén megismert levélvégejelzés használatos.
A fejlettebb POP-kiszolgálók pusztán a levél elejének letöltésére is lehetőséget adnak. Ez akkor lehet hasznos, ha modemes kapcsolattal rendelkezünk, és nem szeretnénk az olykor nagyméretű levélszemeteket vagy egyéb kéretlen leveleket letölteni. Néhány ügyfél lehetővé teszi, hogy a levélfejlécek letöltése után leveleinket törölhessük. Ennek használatához az ügyfélnek és a kiszolgálónak támogatnia kell a `TOP` parancsot.

A jelszó nyílt elküldése (főleg az Interneten át) nem szerencsés. Kikerülésére több megoldás is született: az egyik az `APOP` parancs használata, amelyre a *6. képen* láthatunk példát. Itt a kiszolgáló üdvözlő üzenetében egy rejtvény található (a példában ez a `<1219.1014038493@cheetah.andrews>` karaktersorozat). A cél, hogy erre az `APOP` parancsban a megfelelő választ adjunk. Az `APOP` parancsban el kell küldeni a felhasználó nevét, valamint a választ, ami a rejtvény és a jelszó egymás után írásából származó karaktersorozat MD5-értéke karakterlánc formában. A válasz így tehát kiszámítható (amennyiben a jelszó „netuddki”):

```
$ echo -n '<1219.1014038493@cheetah.andrews>
➥netuddki' | md5sum
a396cd1609f95c6fc00822c7689df3e9  -
```

Ezenkívül az IMAP4-protokollnál bevezetett többféle azonosítási módszer használata is lehetséges. Ezekre az IMAP4-protokoll ismertetésekor térünk ki. A POP3-protokoll bővítése a TLS (Transport Layer Security – az SSL utódja) használatát is lehetővé teszi. Célja, hogy a csatorna a jelszó vagy a levél lehallgatásától, valamint az egyéb támadásoktól védve legyen.
A POP3-protokoll egyszerűsége ellenére a kiszolgálómegvaló-

7.

sítások sajnos sokszor súlyos biztonsági hibákat tartalmaznak. Amennyiben belső POP3-kiszolgálót kívánunk üzemeltetni, az elérhetőség körét célszerű a szükségesre korlátozni. Ne feledkezzünk meg arról, hogy a cikk elején vázolt általános levelezési gondok a POP3-protokollt ugyanúgy érintik. A protokoll csomagszűrése egyszerű: a kiszolgálók szabványos `110/tcp`-kapun várnak az ügyfelekre. Ezenkívül használhatják még a `995/tcp`-kaput is, ami POP3S, ami SSL-es védelmet jelent. A POP3S szükségessége fokozatosan csökken, mivel a szabványos változat is képes már TLS-t használni.
Amennyiben a belső ügyfélgépek szeretnék az Interneten elhelyezett POP3-kiszolgálókat elérni, ellenőrizzük a kiszolgálót, valamint az ügyfélprogramot, hogy támogatja-e a titkosítást (TLS vagy POP3S), illetve milyen azonosítást használ.
Az alábbi példa-beállítás lehetővé teszi, hogy a védett gépek elérhessék a külső POP3- és POP3S-kiszolgálókat.

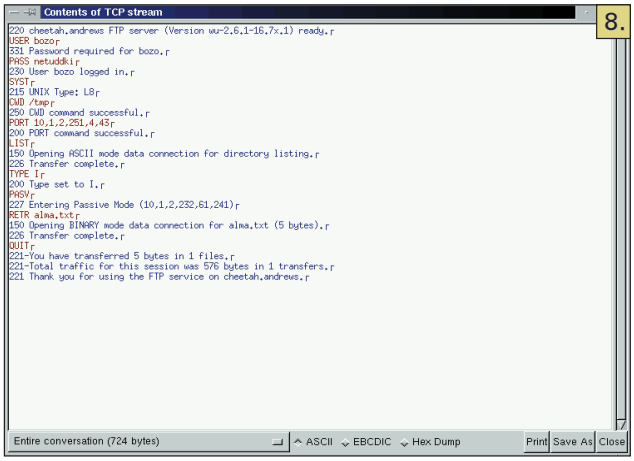
2.2.x rendszermagsorozatnál:

```
ipchains -A input -i eth0 -d 0.0.0.0/0 pop3
➥-p tcp -j ACCEPT
```

```
ipchains -A input -i eth0 -d 0.0.0.0/0 995
➥-p tcp -j ACCEPT
```

2.4.x rendszermagsorozatnál:

```
iptables -p tcp -A INPUT -i eth0 --dport pop3
➥-j ACCEPT
iptables -p tcp -A INPUT -i eth0 --dport 995
➥-j ACCEPT
```



8.

IMAP4

Az IMAP4-protokoll célja, hogy a felhasználó több helyről is kezelhesse a postaládáit. Ez a POP3-protokollal szemben nem csak a levelek letöltésére és törlésére készült. A levelek ilyen esetben a kiszolgálón maradnak, a felhasználó ott végezhet velük műveleteket. A protokoll csak a levelek kezelését végzi, a küldésre az SMTP-protokoll használható.

A protokoll meglehetősen bonyolult, ezért csak ízelítőt adunk a működéséből. A TCP-kapcsolat felépítése után az ügyfelet egy üdvözlősor várja, amelyben a kiszolgáló által támogatott kiegészítések is fel vannak sorolva. Az ügyfél ezután kérést küld a kiszolgálónak. Minden kérés elején egy egyedi azonosító helyezkedik el, ezt követi a parancs, valamint a parancshoz tartozó paraméterek. A *7. képen* ezek jól láthatók, valamint az is, hogy miként lehet egy IMAP-kiszolgálóra bejelentkezni. Mint a kép is mutatja, a jelszó ez esetben is kódolatlanul halad át a hálózaton, amely komoly biztonsági hibát jelenthet. Ez ellen többféle módon is védekezhetünk, az egyik lehetőség a megfelelő azonosítási módszer használata. A fejlettebb IMAP-kiszolgálók a `LOGIN` parancson kívül lehetővé teszik az `AUTHENTICATE` parancs használatát, amelynek segítségével számos azonosítási módszer áll a rendelkezésünkre (például Kerberos és többféle challenge-response azonosítás). Ezenkívül a kapcsolatot magát is védhetjük titkosítási eszközökkel: elterjedt módszer a kapcsolat SSL-be ágyazása (IMAPS), és egyre több kiszolgáló támogatja a TLS-t mint protokollbővítést.

Az IMAP-protokollt csomagszűrővel védeni egyszerű, bár a protokoll összetettsége miatt a kiszolgálóprogram kiválasztására fokozottabban ügyelni kell. Az IMAP-kiszolgálók alapesetben ügyfeleikre a `143/tcp`-kapun várakoznak . Az `"IMAP over SSL"` a `993/tcp`-kaput használja.

Az alábbi példa beállítás lehetővé teszi, hogy a védett gépek elérhessék a külső IMAP4- és IMAP4S-kiszolgálókat.

2.2.x rendszermagsorozatnál:

```
ipchains -A input -i eth0 -d 0.0.0.0/0 imap
➥-p tcp -j ACCEPT
ipchains -A input -i eth0 -d 0.0.0.0/0 993
➥-p tcp -j ACCEPT
```

2.4.x rendszermagsorozatnál:

```
iptables -p tcp -A INPUT -i eth0 --dport imap
➥-j ACCEPT
iptables -p tcp -A INPUT -i eth0 --dport 993
➥-j ACCEPT
```

Állományok mozgatása

A levelezés áttekintése után nézzük meg, miként szokták szállítani a különböző állományokat az Interneten. E szempontból főképpen két protokoll népszerű: az FTP és a HTTP. Az általános kérdésekről (például vírusok és trójai programok) természetesen itt sem szabad megfeledkeznünk, bár kivédésükre most nem térünk ki. A HTTP protokollról és a Web veszélyeiről későbbi írásunkban szólunk részletesebben, most az FTP protokollt vizsgáljuk meg.

FTP

Az FTP szintén az Internet egyik régi alapprotokollja, hiszen az állományok mozgatásának szükségessége már a kezdetek kezdetén felmerült. A protokoll célja, hogy állományok mozgatását tegye lehetővé két számítógép között, a rendszerek eltérő sajátosságainak figyelembevételével. Mit is értünk ez alatt? Még az olyan egyszerű dolgok, mint a szöveges állományok is különbözőek lehetnek a rendszerek között. A DOS/Windows-alapú rendszereken a szövegállományok sorait `CR-LF` bájtsorozat határolja, míg a Unix-alapú rendszerek csak egy `LF` karaktert használnak a sor végének jelölésére. Léteznek olyan rendszerek is, amelyek a sorok végét egy `CR` karakterrel jelzik. A régebbi időkben ennél különösebb eltérések is léteztek a rendszerek között, de az ilyen rendszerek az idők folyamán háttérbe szorultak.

Az FTP (a legtöbb protokolltól eltérően) két független csatornát használ. Az egyik a parancscsatorna, ahol az ügyfélprogram utasíthatja a kiszolgálót, a másik az adatcsatorna, ahol az állományok és a könyvtárlisták közlekednek. Míg a parancscsatorna a kapcsolat ideje alatt fennmarad, addig az adatcsatorna szükség esetén megnyílik, majd az állomány- vagy könyvtárlista átvitele után lebomlik. Az adatcsatorna felépítésére két módszer létezik. Alapesetben a kiszolgáló a kezdeményező fél. Ekkor az ügyfél egy dinamikus TCP-kaput hoz létre, amelynek címét tudatja a kiszolgálóval. A kiszolgáló a `20`-as TCP-kapuról kapcsolódik az ügyfél megadott kapujára – ezt hívják aktív módnak. A másik mód esetén a `PASV` parancs hatására a kiszolgálóprogram egy dinamikus kaput hoz létre. A létrehozott kapu címét a válaszban megadja. Ebben az esetben az ügyfélgép kezdeményezi az adatcsatorna felépítését a kiszolgáló felé. Szaknyelven ezt passzív módnak hívják. A *8. képen* mind az aktív, mind a passzív módra láthatunk példát. A */**tmp*** alkönyvtár listáját aktív módban töltöttük le, az ***alma.txt*** állományt pedig passzív módban. Mind a `PORT`, mind a `PASV` parancsnál a címet az alábbi módon kell értelmezni: az első négy bájt az IP-cím, az utolsó kettő a kapu címe két bájtra bontva. A nagyobb helyiértékű bájt szerepel elöl. A képen ezek szerint az ügyfélgép a `PORT` paranccsal azt jelentette be, hogy a `10.1.2.251:1067` TCP-kapuján vár a kiszolgáló által küldött adatra, a `PASV` parancs válaszából kitalálható, hogy az ügyfélnek az adatért a `10.1.2.232:15857` TCP-kapura kell csatlakozni.

Mint a példából is jól látható, a teljes protokoll (és benne a jelszó) szövegalapú, és a jelszó is nyílt szövegként kerül továbbí-

tásra. Ez súlyosan veszélyeztetheti a rendszereink biztonságát. Az eddig említett protokollokkal (SMTP, POP3, IMAP4) szemben az FTP protokoll biztonsági kiegészítése ugyan megtörtént, de nem terjedt el széles körben. Az FTP protokollal kapcsolatban egyéb biztonsági gondok is felmerülnek. Képzeljük el, hogy egy támadó állományt helyezhet el az FTP-kiszolgálón, és egy megtámadandó rendszer megadott kapujára (például a `telnet`-kapura) állományátvitelt kezdeményez. Amennyiben a támadó az állományt megfelelően állítja össze, úgy az „állományátvitel" leple alatt be tud jelentkezni az adott gépre, és parancsokat tud kiadni, hiába nem lenne képes közvetlenül az adott szolgáltatásra csatlakozni. Ezt a támadási formát a „FTP bounce attack" névre keresztelték. Megakadályozására az FTP-kiszolgálók aktív mód esetén megkövetelik, hogy a célcím a parancscsatorna forráscíme, és a célkapu 1023 feletti legyen. Egy másik lehetséges támadási forma az állományok ellopása. Ezt passzív módú kapcsolatok esetén lehet kihasználni. Ilyenkor a támadó az FTP-kiszolgáló dinamikus kapuira próbál kapcsolódni. Amennyiben a kiszolgáló a passzív átvitel során folyamatosan növekvő kapukat használ, a támadó a pásztázandó tartományt egy könnyen végigpróbálható kis szeletre tudja korlátozni. Ebben az esetben a támadó más felhasználók állományaihoz is hozzáférhet.
Az adatkapcsolat forráscíme a parancscsatorna forráscímével ugyanebben az esetben is vizsgálható lenne, de ezt az FTP-kiszolgálók nem mindegyike teszi meg. Mindkét galibát a külön adatcsatorna jelenléte okozza, ám az ezáltal okozott nehézségek sora ezzel még nem ért véget. Egy FTP-ügyfelet vagy -kiszolgálót hagyományos csomagszűrővel védeni rémálom. Amennyiben FTP-ügyfeleket kell védeni, megfelelő megoldás lehet a passzív mód, hiszen ilyenkor mind a parancscsatorna, mind az adatcsatorna kimenő kapcsolat. Ne felejtsük el azonban, hogy az adatcsatornának mind a forrás-, mind a célkapuja dinamikus, így szinte tetszőleges TCP-kapcsolatot ki kell engednünk (minden kapcsolatot, amelynek forrás- és célkapuja 1024 feletti).
Az aktív mód engedélyezése még veszélyesebb. Ebben az esetben a kívülről érkező csomagokat be kell engedni, ha forráskapujuk 20-as és célkapujuk 1023-nál nagyobb. Ne feledjük, hogy ebben a tartományban helyezkednek el az olyan programok, mint az NFS-kiszolgáló vagy az X-kiszolgáló. Ennek hatására a támadó a 20-as forráskapuról kapcsolatot képes kezdeményezni többek között ezekre az alkalmazásokra is.
Állapottartó csomagszűrőkkel (SPF – Stateful Packet Filter) a helyzet sokat javult, hiszen ezek figyelik a parancscsatornát, és az adatcsatorna létesítésére vonatkozó szabályokat maguk helyezik és távolítják el. Szerencsére a 2.2-es Linux-rendszermagsorozat (masquerading) része ilyen, 2.4-es rendszermag esetén pedig a Netfilter hagyományos forgalomirányítás esetén is képes a kezelésére. Amennyiben az FTP protokollt Linux-tűzfalunk csomagszűrőjével szeretnénk védeni, ne felejtsük el betölteni az FTP-protokollt támogató modulokat (2.2-es sorozat esetén az `ip_masq_ftp.o`, 2.4-es sorozat esetén a `ip_conntrack_ftp.o`, és NAT használata esetén `ip_nat_ftp.o` modulokat).
Az SPF-tűzfalak sem mindig védenek meg azonban az FTP protokoll hibáitól. Az FTP protokoll kellő védelme csak alkalmazásszintű tűzfalakkal lehetséges. Amennyiben csak anonim FTP-t kívánunk használni (eléggé gyakori eset), célszerű lehet egy FTP proxy használata, például a Squid gyorstáré. A Squid (lásd még Linuxvilág 2–3. szám, 74. oldal) webböngészővel egyszerűen használható, valamint az FTP- mellett a HTTP protokollt is támogatja. Külön előnye, hogy gyorstárként is működik, így használatával értékes hálózati sávszélességet nyerhetünk. Az ilyen nagyméretű programokat célszerű elkülönítőbe (jail) zárni, és amennyiben lehetséges, külön gépre elhelyezni. Az FTP beállítás a 2.2.x rendszermag esetén jelentősen eltér masquerading használatakor. Mivel masquerading használatakor a beállítás sokkal egyszerűbb, most csak a hagyományos megoldást mutatjuk meg (amikor nincs szükség címfordításra).

2.2.x rendszermagsorozatnál:

```
ipchains -A input -i eth0 -d 0.0.0.0/0 ftp
➥-p tcp -j ACCEPT
```

amennyiben a passzív módot szeretnénk engedélyezni:

```
ipchains -A input -i eth0 -s 0.0.0.0/0 1024:
➥-s 0.0.0.0/0 1024: -p tcp -j ACCEPT
```

amennyiben az aktív módot szeretnénk engedélyezni:

```
ipchains -A input -i eth1 -s 0.0.0.0/0 20
➥-d 0.0.0.0/0 1024: -p tcp -j ACCEPT
```

2.4.x rendszermagsorozatnál:

```
iptables -A INPUT -p tcp -i eth0 --dport ftp
➥-j ACCEPT
iptables -A INPUT -p tcp -i eth0 -m state
➥--state RELATED -j ACCEPT
```

Áttekintés

Az előző fejezetekben néhány gyakran használt internetes protokoll tulajdonságaival, valamint veszélyeivel ismerkedtünk meg. Nem szabad azonban elfelejteni, hogy a programokban lévő megvalósítási hibák külön veszélyt jelentenek. E programok sajnos általában rendszergazdai jogosultságokkal futnak, hogy 1024 alatti kaput foglalhassanak maguknak, a felhasználót azonosíthassák a rendszer adatbázisaiból, vagy átválthassanak a felhasználó jogaira. Emiatt egy a futtatásának korai állapotban sikeresen megtámadott program (mielőtt még rendszergazdai jogosultságait eldobhatta volna) igen súlyosan veszélyezteti a rendszer egészének biztonságát. A rendszergazda ellen még az elkülönítők (jail) sem mindig jelentenek tökéletes védelmet.

A fenti protokollokkal kapcsolatban érdemes elolvasni a 30. CD-n (Magazin/Konnyu) található irodalomjegyzékben lévő írásokat.

Hivatkozások

[1.] Linuxvilág: 2002. 1–2 szám, 50. oldal – Könnyű álmok: A protokollok (11. rész)
[2.] Linuxvilág: 2001. 8. szám, 56. oldal – Könnyű álmok: Hálózati forgalom vizsgálata (8. rész)

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.

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.

Ablakkezelők (3. rész)

Az ablakkezelőkről szóló cikksorozatunk harmadik részében két igen kedvelt programról és változataikról lesz szó.

A felhasználók többsége gyorsaságáért és üzembiztosságáért kedveli az IceWM-et. Tapasztalataim alapján elárulhatom, hogy minden változata hibátlanul kezeli a KDE, a Gnome és a rendszermenü által indítható programokat. Ezt csak azért emeltem ki, mert már előfordult olyan eset is (főleg a KDE programjaival és az OpenGL-lel kapcsolatban), hogy valamely ablakkezelő rendszer bizonyos alkalmazásokat hibásan kezelt. Mint az ablakkezelőkről szóló sorozatunk első részében jeleztem, a Debian alatt elérhető ablakkezelőkről kívánok írni. A Debian alatt az alábbi IceWM-függő csomagok telepíthetők:

- fspanel: ez például a KDE-tálca egyszerűsített változata. Alkalmas a programok megjelenítésének kezelésére, illetve a munkaterületek közötti váltásra;
- grun: GTK-alapú programindító párbeszédablak;
- icewm-lite: az ablakkezelő egyszerűsített változata;
- icewm: maga az ablakkezelő;
- icewm-common: az IceWM-változatokhoz szükséges közös fájlok;
- icewm-gnome: a Gnome-felület elemeinek grafikus megjelenését átvevő IceWM-változat;
- dfm: egyszerű fájlkezelő;
- IceConf: az IceWM grafikus beállítóeszköze;
- iceme: grafikus menüszerkesztő program az IceWM-hez;
- IcePref: egy másik beállítóeszköz az IceWM-hez;
- icewm-themes: témafájlok az Ice Window Managerhez.

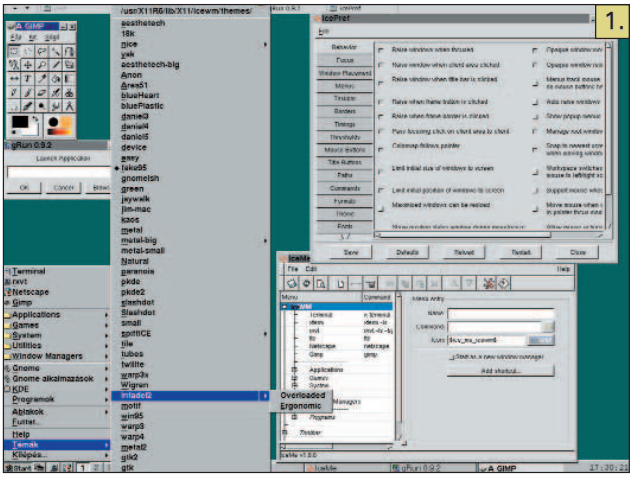
Mint az előbbi listában is látható, az IceWM három változatban telepíthető. Az *1. képen* az IceWM alapváltozata látható. A lite változat annyiban tér el tőle, hogy nincs benne tálca és startmenü. A Gnome-változatban pedig a Gnome-beállításokat átvéve a keret formája, illetve az ablakok kitöltése is a Gnome alatt láthatóval azonos.
Az IceConf és IcePref eszközök az ablakkezelő beállításait teszik lehetővé. Az IcePref sokkal több beállításra ad lehetőséget, mint az IceWM Configurator, de az utóbbi olyan lehetőségeket is tartalmaz, amelyeket a másik nem.
Ha nem grafikus bejelentkezést használunk, az IceWM a ***$HOME/.Xclients*** fájlba a következő sort kell beírni:

```
exec icewm
```

IceWM ablakkezelése egérrel

Bal gomb: ablak kiválasztása és előtérbe hozása, az ablakkeretre kattintva átméretezésre nyílik lehetőség.
Jobb gomb: ablak mozgatása, kattintáskor a helyi menü megjelenítése

- Fejlécparancsok egérhasználatkor:
  Bármely gomb: mozgatja az ablakot.
  ALT + BAL gomb: az ablakot a háttérbe küldi.
  BAL GOMBOS dupla kattintás: az ablak méretét teljes méretűvé növeli vagy visszaállítja.


1.


2.

  KÖZÉPSŐ GOMB dupla kattintás: az ablakot a fejlécbe rejti, illetve lenyitja.
- Tálcaparancsok:
  BAL GOMB: aktiválja a kiválasztott ablakot tartalmazó munkaterületet; az ablak méretét az eredeti méretre állítja vissza vagy a legkisebbre veszi.
  SHIFT + balgombos kattintás: az ablakot a jelenlegi munkaterületre teszi át.
  CONTROL + balgombos kattintás: az ablakot elrejti vagy visszaállítja.
  KÖZÉPSŐ GOMB: az ablakot előtérbe, illetve háttérbe teszi
  SHIFT + középső gombos kattintás: az ablakot a jelenlegi munkaterületre teszi át.
  CONTROL + középső gombos kattintás: rejti az ablakot
  JOBBGOMBOS kattintás: megnyitja a helyi menüt.
- A fontosabb parancssori kapcsolók:
  `-c CONF_fÆjl`: beállítja, hogy melyik beállításokat tartalmazó fájlt használja.

`-v`: a felület változatát jeleníti meg.
`-n`: nem használja a beállításokat tartalmazó fájlt, csak a beépített beállításokat.
`-t` téma: a megadott témát fogja használni.

- Témák kezelése:
  A kiválasztott témák az ablakkezelő betűkészletét, a felhasznált színeit, az ablak keretének méreteit és a felhasznált képeket egyszerre adják meg. A témafájlok a ***libpath/themes*** könyvtárban találhatók. Ezen könyvtárak tartalmazzák a téma- és a hozzájuk szükséges ***.xpm*** fájlokat. A kiválasztott téma a `Theme` beállítási értékkel, vagy a `-t` parancssori kapcsolóval adható meg a ***preferences*** fájlban.

```
Theme = "nice/default.theme"
```

Érdekes témákat tölthetünk le a ➲ http://icewm.themes.org/ címről.

Enlightenment

Azt hiszem, még mindig ez az egyik legszebb ablakkezelő. Pár éve az volt vele a fő gond, hogy az akkori gépek erőforrásait nem kímélve a grafikus munkaállomást nagyon lelassíthatta. Ez a jelenlegi, viszonylag erősebb gépeken már rég nem okoz gondot. Esetében szinte minden grafikus felületen állítható be. A felületi beállításokat az egér jobb gombjával az asztalra történő kattintás után megjelenő menü elemeivel végezhetjük el. A *2. képen* látható néhány alkalmazás, amely ebből a menüből indítható el. Az alkalmazásmenüt ugyanitt az egér bal gombjával érhetjük el. A középső gomb használatával egy újabb alkalmazásmenü jelenik meg néhány beállítással kiegészítve (például itt válthatunk témát). A középső gombra kattintás után megjelenő menüben nyílik lehetőségünk az Enlightenment saját kis alkalmazásainak elindítására (*3. kép*). Az így megjelenített alkalmazások (némelyik ablaka esetenként nem nagyobb, mint 1x1 cm) igen sok kisegítő lehetőséget tartalmaznak. Bemutatásuk azonban kitöltené az ablakkezelőkről szóló sorozat e részét. Fontos megemlítenem, hogy ezek a kis ablakok a *Lapozó*-hoz hasonlóan a munkaterületekkel együtt mozognak – mindig a pillanatnyilag használt asztalra. Kilépéskor, majd az újbóli belépéskor – nem tudom, miért – ezen ablakok egy része nem nyílik meg magától, a többség azonban rendesen megnyílt.
Az indításkor megjelenő asztalon egy pagert (nevezzük lapozónak), illetve egy iconboxot (ikondobozt) találunk, valamint a képernyő felső szélén a dragbart (munkaterület-váltót).
A Enlightenment-felület lehetővé teszi az ablakeseményekhez rendelt hangok lejátszását. Ehhez az ***esound*** eszközt kell telepíteni. Alapesetben a telepítés során az `esound` a számítógép indításakor démonként elindul. Ha ez valamiért nem így történik, az `esd &` parancsot kell kiadnunk, például a ***.Xinitrc*** fájlból az Enlightenment indítása előtt.

A felület felépítése

Az Enlightenment-felület felépítése a következő: megkülönböztetünk munkaterületeket és a hozzájuk kapcsolódó egyes asztalokat. Ezt a megkülönböztetést (fvwm, afterstep) természetesen számos más ablakkezelő is használja. A különbség hozzájuk képest csupán annyi, hogy itt a ***Többasztalos beállítás***-sal egyszerre több munkaterületet meg tudunk jeleníteni (*4. kép*). A ***Többasztal beállításai*** ablakban adhatjuk meg a munkaterületek számát, a ***Virtuális asztalok beállítása***-inál pedig az egy munkaterülten elérhető asztalok számát és egy-

A képernyőkímélő beállítása

Az elérhető képernyőkímélők próbája és használatának engedélyezése, továbbá a képernyőkímélő-indítás és a kapcsolódó jelszóvédelem beállításai.

Beállítások
A beállítások két helyen tárolódnak:

- az egyik X-forrás adatbázisában, amelyek az alapbeállításokat tartalmazzák,
- és mint említettem, a helyi beállításokat a felhasználók saját könyvtárában az *.xscreensaver* fájl tartalmazza.

A beállítások legegyszerűbb eszköze az `xscreensaver-demo`. Az itt elvégzett beállítások csak akkor lépnek életbe, ha a beállításokat tartalmazó fájl újra beolvasásra kerül (*File/Restart Daemon*). Ez a következő parancsot hajtatja végre:

```
xscreensaver-command -restart
```

Az XscreenSaver indítása két helyről lehetséges.
1. Az XscreenSaver indítása központilag (rendszergazdai jogosultság szükséges hozzá):
a, az XscreenSaver indítása anélkül, hogy bárki is bejelentkezett volna. Az */usr/lib/X11/xdm/Xsetup* fájlba a következő sorokat írjuk be:

```
xhost +localhost
xscreensaver-command -exit
xscreensaver &
```

Ilyenkor az `xscreensaver`-t a rendszergazda futtatja, ezért a feloldáshoz az ő jelszava szükséges.
b, az XscreenSaver újraindítása, ha valaki belép:
ha az előbbi beállítások már megvannak, az */usr/lib/X11/xdm/Xsession* fájl elejére az alábbi sorokat kell beírnunk:

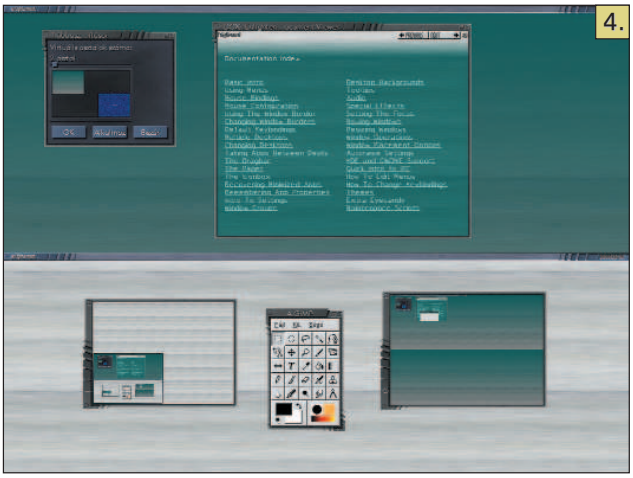
```
xscreensaver-command -exit
xscreensaver &
```

Ekkor a felhasználó belépésekor az XscreenSaver újraindul, és a felhasználó beállításait olvassa be.

2. Az XscreenSaver indítása egyénileg.
Ez esetben a szükséges indítóparancsot (*~/.xsession*) a helyi X-et indító fájlba kell beleírnunk:

```
xscreensaver-command -exit
xscreensaver &
```

Előfordulhat, hogy az ablakkezelő saját indítóeszközzel rendelkezik, ilyenkor ezeket a beállításokat a saját indítófájlába is beírhatjuk (a WindowMaker *autoexec* fájlja, ezt használom most).



máshoz való elhelyezkedésüket. Az első munkaterületen a munkaterület-váltó segítségével egyszerre több munkaterületet is meg bírunk jeleníteni.

Telepítés

Az Enlightenment forrásból történő telepítése során van néhány telepítési lehetőség, amit érdemes megemlíteni:
`--cache-file=F`JL` A gyorstárelérésre a megadott FÁJL-t fogja használni, melynek eredményét a *./config.cache* fájlba menti. A FÁJL értékét */dev/null*-ra állítva a gyorstár használatát a `configure` használatakor nem engedélyezzük.
A `--help` a `configure` parancs lehetőségeit jeleníti meg.
A `--quiet`, `--silent`, `-q` a fordítás során nem jeleníti meg a make parancs visszajelzéseit.
A `--srcdir=KONYVTAR` a fordítónak az Enlightenment forráskódjának helyét adja meg. Ezt általában a `configure` parancs önműködően is meg tudja tenni.
A `--version` kiírja a `configure` által használt Autoconf változatszámát és kilép.
Debian esetében a szükséges csomagok az alábbiak (kapcsolódó *lib*-ek nélkül):
`epplets` – kis alkalmazások (Eppletek) az Enlightenmenthez
`e16keyedit` – gyorsbillentyűszerkesztő az Enlightenmenthez
`e16menuedit` – az Enlightenment menüszerkesztője
`enlightenment` – az Enlightenment ablakkezelője
`enlightenment-data` – az Enlightenment adatfájljai
`enlightenment-theme-bluesteel` – a Hunchback Enlightenment témája
`enlightenment-theme-brushedmetal` – zenei fájlok a BrushedMEtal-Tigert Enlightenment témához
`enlightenment-theme-ganymede` – a cK Enlightenment témája
`enlightenment-theme-shinymetal` – a Raster Enlightenment témája
`eterm -- Enlightenment` – a saját terminálemulátora
A fentieken kívül még szükség lehet az esoundra és a kapcsolódó fájlaira, amennyiben az ablakeseményekhez hangot szeretenénk rendelni.
Egyetlen dolog zavart: a megjelenített betűk mérete és kódolása. Ezen úgy segítettem, hogy megnéztem, melyik téma betűkészlete jeleníti meg a betűkészleteket olyan módon, ahogyan én szeretném, majd rendszergazdaként a */usr/share/enlightenment/themes/Jo_Tema_neve/ttfonts* könyvtárban lévő összes *.ttf* fájlt átmásoltam a használni kívánt téma azonos könyvtárába, és az ott lévő betűkészletet felülírattam. Én úgy tudom, hogy a betűméret megváltoztatása csak a beállításokat tartalmazó fájlokban módosítható (a font szóra keresve), de ha valaki más, egyszerűbb megoldást tud, nagyon megköszönöm a segítségét!

Összegzés

A fentiekben tárgyalt ablakkezelők nagyon kellemes és megbízható felületet kínálnak munkavégzéshez, játékhoz, és más bokros teendőinkhez. Az ablakkezelőkről szóló következő és egyben utolsó részben az AfterSteppet és a WindowMakert mutatom be.

Forrás: cikkemhez az IceWM, az Enlightenment és a XscreenSaver leírásait használtam fel.

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

Gyakorlati fenyegetettségelemzés és kockázatkezelés

Attól, hogy fenyegetettségelemzést végzünk, még nem fogunk nyugodtabban aludni, de segít abban, hogy felébredve helyesen cselekedjünk.

Aki már régebb óta olvassa ezt a rovatot, tapasztalhatta, hogy szeretem, ha a műszaki megoldásokhoz, eszközökhöz és technikákhoz elegendő háttéradat áll rendelkezésre.
A biztonság hatalmas témakör, és az egyetlen lehetőség, hogy eligazodjunk a változatok, módszerek és fekete mágia tengerében, az, ha megpróbáljuk megtalálni a közös pontokat ebben a biztonsági kirakójátékban.
Az egyetlen darabka, amely minden biztonsági feladatban közös, a fenyegetettség. Fenyegetettség nélkül ugyanis nincs értelme a biztonsági feladatokról beszélni. Csakhogy vajon mennyi időt töltünk a fenyegetettség felmérésével és értékelésével ahhoz képest, hogy mennyit szánunk a biztonsági megoldások telepítésére és (remélhetőleg) karbantartására? Valószínűleg igen keveset. Még ha így is van, ne keseredjünk el nagyon, hiszen még a biztonsági szaktanácsadók is túl keveset foglalkoznak a fenyegetettségelemzéssel.
Természetesen, nem azt mondom, hogy órákat töltsünk el vele. Álláspontom az, hogy eszményi esetben csak kritikus rendszereink sértetlenségét és elérhetőségeit kell rendszerezetten végigböngészni; végül a kevésbé szükséges, de fontos rendszerek fenyegetettségét is legalább egyszer szervezetten végig kell gondolni. Vagyis nekünk van valamink (vagyon), a Rosszfiúk pedig szeretnék ezt a valamit.
Mielőtt elmerülnénk a fenyegetettségelemzés rejtelmeiben, előbb érdemes lefektetnünk néhány fontos alapelvet és meghatározást. Először is, mit jelent egyáltalán a fenyegetettség? Nagyon egyszerűen: a fenyegetettség a vagyon, a sérülékenység és a támadó együttese.
A vagyon bármi lehet, amit csak meg szeretnénk óvni. Adatbiztonsági értelemben a vagyon legtöbbször adat, számítógépes rendszer vagy számítógépes hálózat. Ezen „vagyontárgyak" sértetlenségét (adatok esetében a bizalmasságukat) szeretnénk megőrizni.
A sértetlenség a jogosulatlan változtatások hiányát jelenti. A jogosulatlan változtatások eredményeképpen a számítógép vagy az adat sértetlensége megszűnik. Ez jelentheti azt, hogy hibás adatot szúrnak a valós adatok közé, a törvényes adatok egy részét törlik vagy megváltoztatják. Számítógépek esetében azt takarja, hogy a támadó a beállításfájlokat úgy változtatja meg, hogy a jogosulatlan felhasználók a rendszert képesek lesznek helytelen módon használni.
Adataink legalább egy részének a bizalmasságát is meg szeretnénk őrizni. Ez valamelyest más feladat, mint a sértetlenség, hiszen a bizalmasság passzív módon is károsítható. Hogy valaki megváltoztatja az adatainkat, könnyen felfedezhetjük és kivizsgálhatjuk – egyszerűen az eredeti és a károsított adat összehasonlításával. Amennyiben azonban a támadó adatainkat jogosulatlanul másolja le (azaz ellopja), a felderítés és a kárfelmérés sokkal nehezebb, hiszen az adat tulajdonképpen nem változott.
Tegyük fel, hogy ABC Társaság rendelkezik egy SMTP-átjáróval, amely a bejövő leveleket dolgozza fel. Ez az SMTP-átjáró két vagyonértéket képvisel. Az első maga a kiszolgáló, melynek tökéletes működése az ABC Társaság napi üzletmenetében igen fontos szerepet tölt be. Más szavakkal az ABC Társaságnak meg kell védenie SMTP-átjárója sértetlenségét, hogy az e-mail szolgáltatása ne szakadhasson meg.
Másodszor ez az SMTP-átjáró tárolja a rajta keresztül érkező leveleket. Így ha az átjáró rendszere megsérül, a bizalmas levelekbe beleolvashatnak, illetve fontos adatokhoz férhetnek hozzá. Az SMTP-átjáró védelme tehát az ABC társaság e-mail adatainak bizalmasságának és sértetlenségének megőrzésében is fontos szerepet játszik.
Az első lépés minden fenyegetettségelemzés esetében tehát a védendő vagyonértékek, illetve a vagyonértékek védendő minőségének, tulajdonságainak a meghatározása.

Sebezhetőségek

A második lépés az érték, illetve a vele közvetlen kapcsolatban álló rendszerek ismert és kézenfekvő sebezhető pontjainak meghatározása. Természetesen az ismert gyengeségekre sokkal könnyebb megoldást találni, mint a tisztán elképzeléseken alapulókra (legalábbis így gondolnánk, mégis jelentős számú számítógép fut az Interneten alapértelmezett, változatlan operációs rendszerrel). Ennek ellenére mindkét fajtát meg kell próbálnunk beazonosítani.
Az ismert gyengeségek programjavítások használatával, körültekintő beállítással, vagy a terjesztő hirdetőtábláin és a nyilvános fórumokon megjelenő utasítások betartásával könnyen javíthatók. Azok a gondok, amelyek ily módon nem enyhíthetők, további vizsgálatot és mérlegelést igényelnek, és vagy külső módszerekkel védhetők (például tűzfallal), vagy el kell őket fogadni, mint az adott rendszer tevékenységének velejáróját.
Az ismeretlen sérülékenységeket meghatározás szerint általánosan kell kezelni, de a jelentőségük ettől még cseppet sem csökken. Ezt a legkönnyebben egy példa segítségével tudjuk bemutatni.
Térjünk vissza ismét az ABC Társasághoz. A levelezés rendszergazdája a Sendmailt szereti futtatni az ABC Társaság SMTP-átjáróján, mivel jól ért a Sendmail beállításaihoz, és eddig jól megfelelt a céloknak. Ugyanakkor nem táplál ábrándokat a Sendmail biztonsági kérdéseit illetően; folyamatosan figyeli a biztonsági hirdetéseket, és minden javítást, frissítést azonnal felrak, amint megjelenik. Az ABC Társaság így az ismert Sendmail-kiskapuk ellen elég jól védett.
Az ABC igazán menő e-mail rendszergazdája azonban nem elégszik meg ennyivel. Igaz, biztos benne, hogy a Sendmail már biztonságosan javított és beállított, de azt is tudja, hogy korábban voltak veremtúlcsordulási kiskapuk, amelyek gondokat okoztak, különösen akkor, ha a Sendmailt rendszergazdaként futtatták (rendszergazdaként ugyanis a futó folyamat eltérítése a rendszergazdai jogosultság megszerzésével egyenértékű). Ezért aztán a Sendmailt `chroot jail` alatt futtatja (a teljes rendszer egy alhalmazán) root felhasználó helyett `mail`-felhasználóként, és ennek megfelelően állítja be a Sendmail `SafeFileEnvironment` és `RunAsUser` feldolgozási kapcsolóit. Ezáltal az SMTP-átjáró többszintű védelemre is szert tesz,

Leírás	Becsült ár
Helyreállítás: megbeszélés egy külső céggel (4 óra×150 dollár)	600 dollár
Kieső termelés (2 óra per 10 munkás x átlagosan 17.5 dollár/óra)	350 dollár
FAX-papír, hőalapú (thermal) (1 tekercs 16 dollár)	16 dollár
Távolsági FAX-kapcsolatok (20 x átl. 2 perc×0,25 dollár/perc)	10 dollár
Összes SLE egynapos SMTP-kiszolgáló elleni DOS-támadás esetén	***950 dollár***

1. táblázat Részletezett egyedi várható veszteség

és már nemcsak az ismert gyenge pontok ellen védett, hanem bizonyos ismeretlen kiskapuk ellen is, amelyek ugyan károsíthatják a Sendmailt, de egyúttal remélhetőleg nem okozzák a teljes rendszer károsodását.

Támadók

A fenyegetettség-kirakójáték utolsó darabkája, amit megvizsgálunk, mielőtt a fenyegetettségelemzés rejtelmeibe belevetnénk magunkat, maga a támadó. A támadók, vagy ahogy néha nevezik őket, a „művészek" (actors), igen változatosak lehetnek, a várhatóaktól (elégedetlen exalkalmazottak, csintalan fiatalok) kezdve egészen a „különös, de igaz" (kábítószerszövetségek, kormányzati ügynökségek, ipari kémek) típusokig. Ha a lehetséges támadókat számba vesszük, kiderül, hogy szinte minden típusuk előfordulhat; feladatunk ilyenkor, hogy megbecsüljük, vajon melyik támadó a legvalószínűbb.
Jó ökölszabály ilyen esetekre, ha számba vesszük, milyen támadók ellen tervezték is a fizikai biztonsági rendszereket, majd azokat a földrajzi korlátoknak megfelelően módosítjuk. A két dolog között párhuzam vonható: ha költséges zárat szerelünk fel a számítógépterem ajtajára, senki sem kérdezi meg tőled, hogy „tényleg azt hiszed, a karbantartók ellopják ezeket a gépeket, ha hazamegyünk?"
A számítógépes biztonság semmiben sem különbözik ettől. Gyakran hallani olyasmit, hogy „az én adataim érdektelenek; senki sem akarhatja feltörni a rendszeremet", pedig nincs más választásunk, mint feltételezni, hogy ha valamilyen támadás ellen sérülékenyek vagyunk, azt valaki ki fogja használni, még akkor is, ha el sem tudjuk képzelni, vajon miért teszi. Nyilvánvalóan nem az a fontos, hogy megértsük a támadó észjárását, hanem hogy beazonosítsuk és mérsékeljük a támadható gyengeségeket.

Egyszerű kockázatelemzés: ALE-k

Miután összeállítottuk az értékek és a gyenge pontok listáját (és végiggondoltuk a lehetséges támadókat), a következő lépés a köztük lévő kapcsolat feltárása, illetve a mennyiséghatározás. Egy egyszerű módszer, amivel lemérhetjük a kockázat mértékét, az éves várható veszteség (annualized loss expectancies, azaz ALE-k) kiszámítása.
Az összes gyenge pontot minden egyes értékkel párosítva előbb megbecsüljük az adott érték pótlásának vagy helyreállításának költségét (egyszeri veszteséggel számolva), majd megállapítjuk a gyenge pont várható éves előfordulási gyakoriságát. Ezekután a kettőt összeszorozva kapjuk meg a sérülékenység éves várható veszteségét.
Más szavakkal: minden gyenge ponthoz kiszámítjuk a következő értéket: egyszeri várható veszteség (költség) × (várható) éves gyakoriság = éves várható veszteség.
Például egy kisvállalkozás, a Mommenpop Kft. ki szeretné számítani az SMTP-átjárójuk elleni DoS (denial of service) támadásokra vonatkozó ALE-értéket. Tegyük fel továbbá, hogy az e-mail szolgáltatás kulcsfontosságú az üzletmenethez; mind a tíz alkalmazottjuk e-mail alapján számláz az ügyfeleknek, így ad munkabecsléseket a jövendőbeli vásárlóknak, illetve más hasonlóan fontos üzleti kapcsolattartást folytat. Ugyanakkor a hálózatkezelés nem éppen a szakterületük, így egy helyi tanácsadócégre bízzák az levelezőkiszolgáló kezelését.
A korábbi, átlagosan egynapos kimaradások a termelékenységet körülbelül az egynegyedével csökkentették, amely visszaszámolva napi két órát ad ki alkalmazottanként. Tartalékrendszerük egy faxgép, de mivel székhelyük egy kisvárosban található, ez távolsági hívásokkal jár és meglehetősen drága.
Lehet, hogy kicsit talán bonyolultabban hangzik, mint amilyen valójában; táblázatban kifejezve azonban máris sokkal kevésbé elrettentő (lásd az *1. táblázatot*).
A következő lépés az adott gond várható éves előfordulásának (Expected Annual Occurrence avagy EAO) megbecslése. Ezt számként gond per év hányadosként szokás kifejezni. Példánkat folytatva tegyük fel, hogy a Mommenpop Kft. ez idáig még semmiféle kémkedésnek vagy más, a versenytársak által elkövetett támadásnak nem volt célpontja, és legjobb tudásunk szerint a levelezőkiszolgáló elleni DoS-támadások legvalószínűbb forrásai vandálok, gengszterek, zűrös emberek és más véletlenszerű idegenek lehetnek.
Elfogadhatónak tűnik az a becslés, miszerint ilyen támadás két- vagy háromévente körülbelül egyszer fordul elő; de legyünk óvatosak és mondjunk kettőt. Kétévente egy támadás átlagosan 0,5 gondot jelent évente, azaz az EAO 0.5. Illesszük be ezt az értéket az ALE-képletbe:

```
950 ($/problØma)   0.5 (problØma/Øv) = 475 ($/Øv).
```

A Mommenpop SMTP-átjáró elleni DoS-támadások ALE-értéke tehát 475 dollár évente.
Most tegyük fel, hogy valamilyen terjesztő megpróbálja rábeszélni a céget saját fejlesztésű Linux-tűzfalának kereskedelmi tűzfalra való cseréjére; ez a termék beépített SMTP-proxyval rendelkezik, ami segít csökkenteni, de nem szünteti meg az SMTP-kapu DoS-támadásokkal szembeni érzékenységét. Tegyük fel, hogy ez a termék 5000 dollárba kerül. Még ha a költségeket három évre osztjuk is el (10% kamattal számolva ez évente 2166 dollárt tesz ki), egy ilyen tűzfalfejlesztés nem tűnik igazán jogosnak egyetlen kockázatforrás kiküszöbölésére.
A *2. táblázat* a képzeletbeli cégünk SMTP-átjárójának fenyegetettségelemzését mutatja be kicsit teljesebb formában, ahol nemcsak az ALE-eket, hanem számos más, a vagyonértékeinket célzó számértéket is bemutatunk, illetve különféle biztonsági célokat is láthatunk. Ebben a példaelemzésben a vásárlói adatok bizalmassága minősült a legfontosabb kockázati értéknek; ha ugyanis ezeket fürkészik ki vagy ezekbe nyúlnak bele, a cég könnyen elveszítheti a vásárlóit (hiszen megrendül a Mommenpoppal szembeni bizalom), ami végsősoron a jövedelem megcsappanását jelenti. E veszteségek különböző jelentőségét mutatják az egyes gyenge pontokhoz tartozó egyszeri várható veszteségábrák. Hasonlóképpen a különféle becsült éves előfordulási mennyiségek az egyes gyenge pontok tényleges kihasználásának relatív valószínűségét mutatják.

Érték	Biztonsági cél	Sérülékenység	SLE$/konfliktus	ARO konfliktus/év	ALE$/év
SMTP-átjáró	Rendszerintegritás	Sendmail-hibák	2400 dollár	0,5	1200 dollár
		Egyéb rendszerhibák	2400 dollár	0,5	1200 dollár
	Rendelkezésre állás	DOS-támadás	950 dollár	0,5	475 dollár
Bizalmas e-mail (ügyfél számlaadat)	Adatbizalmasság	Kémkedés az Interneten vagy az ISP-nél	50 000 dollár	2	100 000 dollár
		SMTP-átjáró feltörése	50 000 dollár	0,5	25000 dollár
		Rossz szándékú belsős	150 000 dollár	0,33	49500 dollár
	Adatintegritás	Hamisított levél a vásárlónak/-tól	10 000$ dollár	1	10 000 dollár
		Továbbítás alatti változtatás az Interneten vagy az ISP-n	10 000 dollár	0,25	2500 dollár
		SMTP-átjáró feltörése	10 000 dollár	0,5	5000 dollár
Nem bizalmas e-mail (művelet-adatok)	Adatintegritás	Továbbítás alatti változtatás az Interneten vagy az ISP-n	3000 dollár	0,25	750 dollár
		SMTP-átjáró feltörése	3000 dollár	0,5	1500 dollár

2. táblázat ALE-alapú példa fenyegetettségmodell

Minthogy a 2. *táblázatban* látható példaelemzést táblázat formájában adtuk meg, a sorokat tetszés szerint könnyen rendezhetjük. A 3. *táblázat* ugyanezt az elemzést mutatja be sérülékenység szerint rendezve.

Hasznos lehet, ha az azonos sérülékenységekhez tartozó ALE-ket összeadjuk. A leveleknek az Interneten töltött idő alatt vagy az ISP-n történő megváltoztatásának eredménye: 2500 dollár és 750 dollár, összesen tehát 3250 dollár ALE-érték. Ha a képzést feladó tanácsadó például 2400 dollárt kér három félnapos tanfolyamért, ahol a dolgozóknak azt mutatják be, hogyan kell az ingyenes GnuPG programot a dokumentumok aláírására használni, akkor a kiképzési díj ezzel a veszélyforrással arányban áll. Azt is láthatjuk, hogy bizonyos ALE-k egyes sérülékenységekkel kapcsolatban állnak. A 3. ábrán megfigyelhetjük, hogy az alsó három ALE az SMTP-átjáró károsítása miatt bekövetkezett veszteségeket gyűjti össze. Más szavakkal az SMTP-átjáró sérülése nem csak a termeléskiesés és a vaskos helyreállítási költségek miatt okoz veszteséget (1200 dollár bármelyik ALE esetében, a 3. táblázat tetején), hiszen a levéladatok károsodásának kockázata további 31 500 dollár veszteséggel is fenyegeti a céget, így az ide tartozó ALE értéke összesen 32 700 dollár. Látható, hogy a levelezés kikémlelésének vagy módosításának veszélye igen magas. A Mommenpop Kft. jobban tenné, ha máris hívná azt a 2400 dolláros oktatót.

Az ALE-n alapuló elemzőeszközök egyik nagy hátránya a részrehajló szemléletmód (figyeljük meg, a fenti leírásban milyen gyakran szerepeltek a „valószínű” és a „megfelelő” szavak), és épp emiatt a végeredményt a gyakorlati adatok helyett alapvetően a tanulmány készítőjének tapasztalata és tudása határozza meg.

Ez módszer ráadásul nem ad igazán jó lehetőséget az ALE-k összehasonlítására (az olyan rövidke listáktól eltekintve, mint a 2. és a 3. táblázat).

Az ALE-módszer előnye egyszerűségében és rugalmasságában rejlik. Bárki, aki elegendő ismerettel rendelkezik a saját rendszerének felépítéséről és működtetési költségeiről, és nagyjából tisztában van a jelenlegi főbb IT-s (információtechnológiai) biztonsági irányvonalakkal (például olvassa a CERT jelenlegi és korábbi tanácsadó és az összetűzésekről szóló jelentéseit), a környezetről már könnyedén hosszú ALE-listát képes készíteni. Ha ezt a listát táblázatos formában jelenítjük meg, a különféle költségek és gyakoriságok megbecslése különösen könnyű. Annak ellenére, hogy ez a módszer valóban erőteljesen elfogult (amit a kockázatvizsgálatokban teljes mértékben nem is lehet kizárni), igen fontos és hasznos eszköz a kockázati tényezők összeszámlálásában, mennyiségi becslésében és a kockázatok súlyozásában. Az éves várható veszteségek jól szerkesztett listája sokat segíthet nekünk abban, hogy IT-s biztonsági kiadásainkat arra gyenge pontra költsük, amely a leginkább számít.

Egy másik megoldás: Schneier támadási fadiagrammja

Bruce Schneier, az Applied Cryptography (Alkalmazott kriptográfia) szerzője a kockázatelemzés egy másik formáját vezette be: a támadási fákat. A támadási fa nagyon tömören egy adott célpont elleni támadási lehetőségek szemléletes megjelenítése. A támadási célpontot (target) gyökércsomópontnak nevezzük (root node), a cél eléréséhez szükséges alcélokat pedig levélcsomópontoknak hívjuk (leaf nodes).

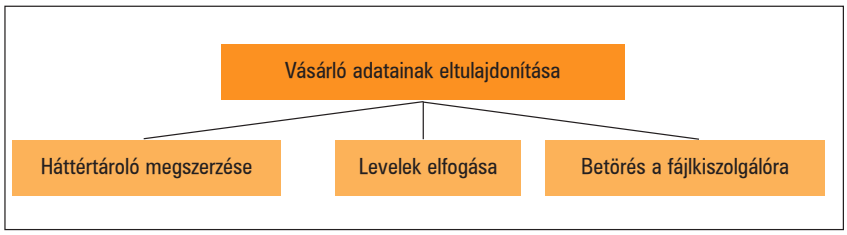
A támadási fa elkészítéséhez először is meg kell neveznünk egy gyökércsomópontot. Támadási cél lehet például az „ellopni a Mommenpop Kft. vásárlóinak bejelentkezési adatait”. Ennek közvetlen módozatai a következők lehetnének:

1. A Mommenpop-fájlkiszolgáló szalagjainak megszerzése,
2. A Mommenpop Kft. és a vásárlói közti e-mailforgalom elfogása és
3. Interneten keresztüli betörés a Mommenpop-fájlkiszolgálóra.

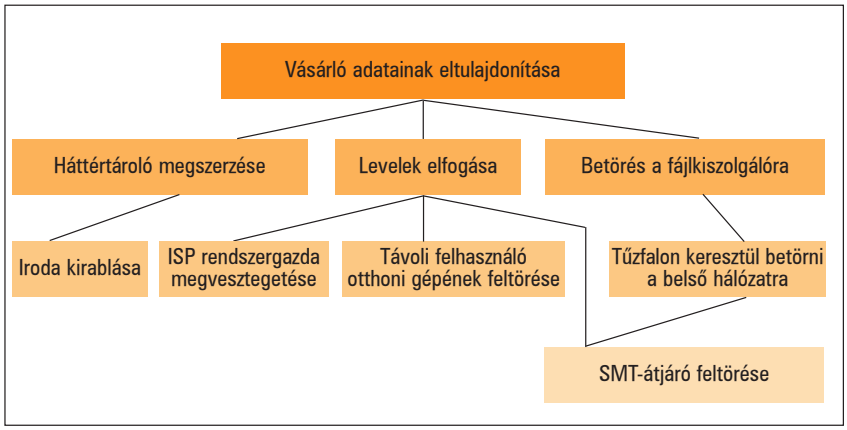
Érték	Biztonsági cél	Sérülékenység	SLE $/konfliktus	ARO konfliktus/év	ALE $/év
SMTP-átjáró	Rendszerintegritás	Sendmail-hibák	2400$	0,5	1200$
SMTP-átjáró	Rendszerintegritás	Egyéb rendszerhibák	2400$	0,5	1200$
Bizalmas e-mail (ügyfél számlainformáció)	Adatbizalmasság	Rossz szándékú belsős	150000$	0,33	49500$
Bizalmas e-mail (ügyfél számlainformáció)	Adatintegritás	Továbbítás alatti változtatás az Interneten vagy az ISP-n	10000$	0,25	2500$
Nem bizalmas e-mail (művelet információk)	Adatintegritás	Továbbítás alatti változtatás az Interneten vagy az ISP-n	3000$	0,25	750$
Bizalmas e-mail (ügyfél számlainformáció)	Adatintegritás	Hamisított levél a vásárlónak/-tól	10000$	1	10000$
Bizalmas e-mail (ügyfél számlainformáció)	Adatbizalmasság	Kémkedés az Interneten vagy az ISP-nél	50000$	2	100000$
SMTP-átjáró	Rendelkezésre állás	DOS-támadások	950$	0,5	475$
Bizalmas e-mail (ügyfél számlainformáció)	Adatbizalmasság	SMTP-átjáró feltörése	50000$	0,5	25000$
Bizalmas e-mail (ügyfél számlainformáció)	Adatintegritás	SMTP-átjáró feltörése	10000$	0,5	5000$
Nem bizalmas e-mail (műveletinformációk)	Adatintegritás	SMTP-átjáró feltörése	3000$	0,5	1500$

2.táblázat Ugyanez a példa sérülékenység szerint rendezve

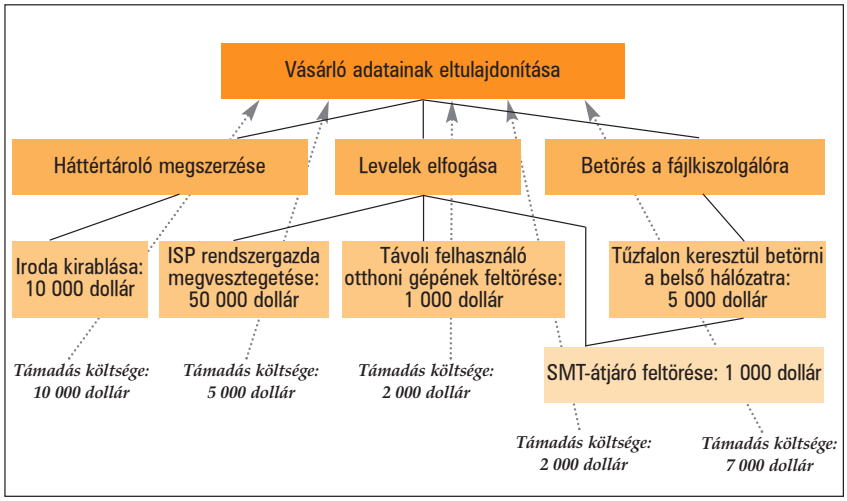
Ez a három alcél (levélcsomópont) helyezkedik el közvetlenül a gyökércsomópont alatt. (Lásd a *4. ábrát*)
Most minden egyes levélcsomóponthoz meg kell keresnünk azokat az alcélokat, amelyek az adott levélcsomópont eléréséhez feltétlenül szükségesek. Ez lesz a következő levélcsomópont-rétegünk. Ezt a lépést addig ismételgetjük, amíg a kívánt mélységet és összetettséget el nem érjük. Az *2. ábra* egy egyszerű, de többé-kevésbé teljes támadási fát mutat be a Mommenpop Kft. esetében.
Kétségtelen, hogy további kiegészítő leveleket is kitalálhatnánk az *5. ábrán* jelölt két réteghez, sőt akár új rétegeket is készíthetnénk. De tegyük fel, hogy a jelen környezet meglehetősen jól biztosított a belső támadásokkal szemben (elég ritka, hogy valóban így is van), illetve hogy egy kívülálló számára ezek a leginkább keresztülvihető támadási metódusok.
A példából sok minden kiolvasható: a háttér adathordozó megszerzése legkönnyebben az irodába történő behatolással oldható meg; a belső fájlkiszolgáló feltörése a tűzfal keresztültörését foglalja magába, az elfogott levelek segítségével pedig három különféle módon férhetünk hozzá az adatokhoz. Azt is leolvashatjuk róla, hogy bár a tűzfal támadása a legjobb módszer a Mommenpop Kft. SMTP-kiszolgálójának feltöréséhez, egy másik, közvetlenebb módszer is adódik: az elfoglalt átjárón keresztül haladó levelek elolvasása.
Ezek nagyon fontos adatok. Lehet, hogy a cég több pénzt szándékozik a tűzfalra költeni, de megeshet, hogy úgy dönt, jobban megéri neki, ha a pénzt és az időt az SMTP-átjáróra fordítja. Legalább ilyen fontos az is, hogy láthatjuk az egyes támadási célok közötti kapcsolatokat, amit ezzel a fával még nem végeztünk el. Ha a támadási fát a kívánt mélységig felrajzoltuk, elkezdhetjük felbecsülni az egyes leveleket. Például minden egyes levélhez árcédulát kapcsolhatunk, amely az adott cél eléréséhez szükséges becsült pénzösszeget jelképezi. Ha minden támadási vonalat árcédulákkal jelöltünk meg, könnyen megbecsülhetjük a különféle támadási módok egymáshoz viszonyított költségét. A *3. ábra* az árcédulákkal kiegészített támadási fát mutatja be (a pontozott vonalak a támadási utat jelölik).
A 3. ábrában úgy találtuk, hogy a betörés elég költséges támadási forma, hiszen az elfogatást és a börtönt is megkockázatja. Senki sem fogja ezt a munkát nekünk megfelelő ellenszolgáltatás nélkül elvégezni. Ugyanez igaz az ISP rendszergazdájának megvesztegetésére; még egy megvásárolható ISP-alkalmazott is kétszer meggondolja, érdemes-e elvesztenie a munkáját és priuszt szereznie.
A feltörés egy kicsit más – bár éppúgy törvénytelen, sokkal kevésbé tartják kockázatosnak, mint a betörést. Továbbá a legtöbb szervezet számítógépes védelmét sokkal kevésbé nehezebb áttörni, mint a fizikai védelmet.
Azt mondják, egy tűzfal keresztültöréséhez egy kicsit nagyobb tudás kell, mint amennyivel egy átlagos script-kiddie rendelkezik, továbbá némi időt és erőfeszítést igényel; így aztán ez is viszonylag drága cél. Az SMTP-átjáró feltörése már sokkal könnyebb, és ha már beazonosítottunk egy vagy több távoli felhasználót, akkor az adott felhasználó otthoni gépe jó eséllyel könnyedén feltörhető. Ezért ezt a két célt olcsóbbnak minősítettük.
A támadástípusoknak megfelelő képzettségű bűnözők bérlési költsége alapján e példában a legígéretesebb támadási forma az SMTP-átjáró feltörése és a távoli felhasználók gépeinek elfoglalása. A Mommenpop Kft. úgy tűnik, jobban teszi, ha hálózata peremvidékét, az SMTP-kiszolgáló biztonságát és a távoli elérési lehetőségeket kicsit jobban megvizsgálja.
Természetesen a levelekhez nem csak költséget rendelhetünk. Beilleszthetünk kétértékű (boolean) mennyiségeket is, mint például kivitelezhetőeket és nem kivitelezhetőeket; a „nem



1. ábra Gyökércsomópont három levélcsomóponttal



2. ábra Egy részletesebb támadási fa



3. ábra Támadási fa költségbecsléssel

kivitelezhető" a támadási fa bármely pontján egyúttal azt is jelenti, hogy az egész ág kivitelezhetetlen.
Írhatunk ide befektetett munkamennyiséget is, órában vagy percben. Röviden: egyazon támadási fát többféle módszerrel is megvizsgálhatunk, és rendszerünk gyengéiről olyan felbontású képet alkothatunk, amilyet csak akarunk. A *6. ábrában* látható költségbecslések mind azt feltételezték, hogy a támadás kivitelezéséhez a támadónak fel kell bérelnie valakit. Ezeket az értékeket egészen másképpen kellene számítani, mint ha a támadók maguk is tapasztalt rendszerkalózok – ilyen esetben az időbecslés minden csomópontra hasznosabb a költségbecslésnél.

Védekezés

A fenyegetettségelemzés végső célja az lenne, hogy meghatározzuk, milyen szintű védelmet kell felhasználnunk azokon a területeken, amelyeken rendszerünk gyengének tűnik.

Háromféleképpen csillapíthatjuk a kockázatot. A védelmi stratégiákat aszerint csoportosíthatjuk, hogy a támadó számára csökkentjük a vagyon értékét , mérsékeljük az adott gyengeségeket, illetve semlegesítjük vagy megelőzzük a támadásokat. A vagyon értékének csökkentése nem tűnik túl járható útnak, de gondoljunk csak bele: a vagyon értékét a támadó számára kell csökkentenünk, és nem a jogos felhasználók/tulajdonosok számára. A legjobb példa a titkosítás: az alkalmazott támadástípus a cikkben említett példák mindegyikében nagyjából semlegesíthető, amennyiben megfelelő levéltitkosító programot alkalmazunk. Az adatvédelem másik stratégiája a gyengeségek megszüntetése vagy mérséklése. A programjavítások, foltok jó példák erre: minden egyes Sendmail-hiba évek óta arra ösztönözte a fejlesztőket, hogy foltokat adjanak ki, amelyek az adott hibát kijavítják.
A gyengeségek csökkentésére még jobb példa a védekező kódolás. Ha forrásfájljainkat olyan szűrőkön futtatjuk keresztül, amelyek mondjuk megtalálják a hibás határvizsgálatokat, akkor elősegíthetjük, hogy programunk ne legyen sérülékeny a veremtúlcsorduláson alapuló támadásokkal szemben. Ez a módszer sokkal hatékonyabb, mintha egyszerűen, minden ellenőrzés nélkül kibocsátanánk a programot, és várnánk a hibajelentéseket.
A legtöbb figyelmet érdemlő védelmi stratégia azonban a következő: el kell űzni a támadót, mielőtt még hozzáférhetne a sérülékeny rendszerekhez. Nyilvánvaló megoldás erre a tűzfal. A tűzfalakat azért készítjük, hogy megakasszuk a támadást. Az elérést korlátozó szerkezetek – mint a felhasználónév–jelszó sémák, az azonosító nyelvi egységek (token) és az intelligens kártyák (smart cards) – úgyszintén ebbe a csoportba tartoznak, hiszen feladatuk a megbízható és a nem megbízható felhasználó (azaz a lehetséges támadó) közti megkülönböztetés. Nem árt azonban tudni, hogy az azonosítási módszereket a gyenge pontok megerősítésére is felhasználhatjuk (például SecurID nyelvi egységeket használunk egy nem megfelelő jogosultságrendszerrel rendelkező webalkalmazás azonosítási rétegéhez). Mára legyen elég ennyi.

További érdekességek találhatók a 30. CD Magazin/Gyakorlati könyvtárban.

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.

A Linux és a Samba egy országos laboratóriumban

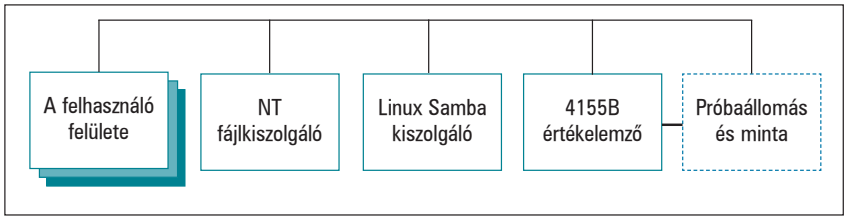
Az alábbiakban a Linuxnak és a Sambának a VCSEL-nek nevezett rendkívül kis méretű lézerek kutatásában történő felhasználásáról olvashatunk.

Nemrégiben a Linux és a Samba adott választ a marylandi Adelphiben található Hadi Kutatólaboratórium (ARL) igényeire. Intézetünk csúcstechnológiai kutatást végez a lézerek egy különleges fajtájánál, és ezen eszközök teljesítményének kipróbálása közben rendkívül nagy mennyiségű adatot gyűjtünk össze. A próbához használt felszerelésünket a hálózaton át rá tudtuk egy Samba kiszolgálóra kötni. Ennél a módszernél az a csel, hogy a beállítások miatt a felhasználók úgy látják, mintha az intézet NT-n futó fájlkiszolgálóján levő adatokat érnék el. Részletesen is el fogom magyarázni a felállást, de a kulcs az, hogy az NT-gépen egy hálózati parancsikont hoztunk létre, mely a Samba-megosztásra mutat, a linuxos gépet pedig a hálózaton láthatatlanná tettük. Az *ábra* a hálózat felépítését mutatja be.

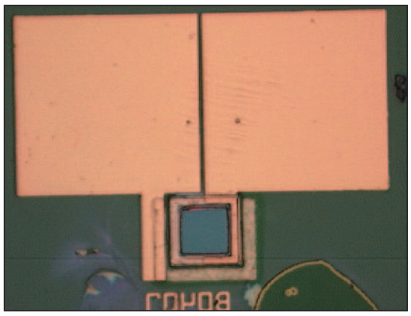
Intézetünk VCSEL-nek (felületi üreges függőlegesen sugárzó lézer) nevezett, rendkívül kis lézereket fejleszt, melyek a fénytani kutatás általános területébe tartoznak. Könnyen megesik, hogy egy négyzetmilliméternyi felületen több mint 60 lézert helyezünk el, és előfordul, hogy a lézereket tartalmazó lapka teljes átmérője mindössze 7,5 centiméter. Így hát megeshet, hogy egyetlen lapkán több ezer alkatrész található. A *1. képen* egy jellegzetes VCSEL látható. A legfőbb próbát, amelyet minden VCSEL teljesítményének ellenőrzéséhez lefuttatunk, az áramerősséget, a fényerőt és a feszültséget mérő ILV-görbének nevezik. Alapjában véve azt vizsgáljuk, hogy a befektetett energia mennyi fényt eredményez. Mivel az elemzőprogramok többsége a felhasználók asztali gépén található, szükségük van rá, hogy a feldolgozatlan adatokhoz onnan férjenek hozzá. A felhasználók a szokások rabjai. Az intézettel kapcsolatos adatok elérése mindig is azt jelentette, hogy el kell menni az NT-kiszolgálóra. Mivel a felhasználók hozzászoktak, hogy az adatokat az NT-s gépről kapják meg, nem akartuk rákényszeríteni őket, hogy mással kísérletezzenek. Igyekeztünk számukra mindent áttekinthetővé tenni, és azt a látszatot próbáltuk kelteni, mintha az NT-kiszolgálóról kapnák az adatokat. Azért, hogy a felhasználóknak az NT-s gépen keresztül kelljen menniük, a linuxos gépet a hálózatról nézve láthatatlanná tettük. Az adatokat elérő felhasználók azonosításában az NT-gép biztonságára támaszkodunk.

A kipróbálás beállításai

A VCSEL-ek jellemzésében két készüléknek jutott kulcsfontosságú szerep. Az első a mintavételező állomás, amely tulajdonképpen néhány apró szondával és egy fénymérővel egybeépített mikroszkóp. A szondák energiát bocsátanak az eszközre, mi pedig a fénymérővel megmérjük az áramot. A második készülék az Agilent gyártmányú 4155B típusú értékelemző. Az elemző úgy lett programozva, hogy pásztázza végig az áramerősség szintjét, valamint mérje meg a feszültséget és a fényerőt. Alapvetően két módon lehet vezérelni: a készüléken található kezelőszerveken és a GPIB-csatolón keresztül. Noha teljesen igaz, hogy a GPIB-kapu a tudományos berkekben népszerű csatoló, és mutatósabb tesztek elvégzését teszi lehetővé azáltal, hogy a próbabeállításokat számítógép vezérli, és az adatok összegyűjtésére is képes, azonban vezérlő számítógépünk a laboratóriumi helyszíntől mintegy ötlábnyira helyezkedik el, és nem lehet közelebb hozni. Emiatt nehéz elkezdeni a próbát, amikor a szondák a helyükre kerültek. Szerencsére fő tesztünket egyszerűen a készülék kezelőfelületén be lehet állítani. A próbát úgy szoktuk végrehajtani, hogy a szondákat a mikroszkóp nézőkéjének segítségével helyezzük el, óvatosan odanyúlunk és az értékelemzőn megnyomjuk a próbát indító gombot, majd pedig mentjük az adatokat. A *2. képen* a laboratórium felszerelése látható.



A hálózat felépítése



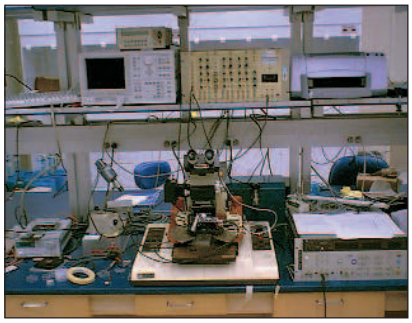
1. kép Jellemző VCSEL: a nagy téglalapok a próbát végző szondák érintkező felületei, a lézerrel besugárzott tényleges terület a kis szürke négyzet alul középen

Működés

Ha a próbát sikerült rendben lefuttatnunk, mentenünk kell az adatokat. A 4155B háromféle módon képes menteni az adatokat: GPIB, hajlékonylemez és TCP/IP. Az elemzőt nem GPIB-vel vezéreljük, ezért ez szóba sem jön. A hajlékonylemez támogatja a 3,5" lemezeket, de ezek a lemezek hamar megtelnek, és hurcolászni kell őket. Több laboratóriumi területen is dolgozunk, ezért előfordult már, hogy egy lépést újra kellett kezdenünk, mert épp eltűnt egy hajlékonylemez. Az általunk összeállított válasz a TCP/IP-támogatás alapján működik.

Linux

Az értékelemző támogatja a TCP/IP-t, pontosabban az NFS-t. Az elemzőt még pingelni is lehet. Be van jegyezve a laboratórium DNS-ébe, ezért IP-cím és



2. kép Mintavételező állomás (alul középen) és a 4155B-típusú értékelemző (a fehér doboz bal oldalon felül)

név alapján is pingelhető. Felesleges és rossz alkatrészekből sikerült összeállítanunk egy linuxos gépet. Szó szerint három számítógép alkatrészeiből raktunk össze egyet. A kormánynak ez semmibe sem került, és arra, amire kell, éppen jó. A telepítésnél a legújabb változat, amely nekünk megvolt és támogatta a Pentium 133-as gép alkatrészeit, a RedHat 6.2-es volt, ezért ezt raktuk fel, és a Bastille tűzfallal, illetve a legfrissebb foltokkal szilárdítottuk meg. Ezenkívül kikapcsoltuk az összes szükségtelen szolgáltatást, és felraktuk az SSH-t. Gondosan felosztottuk a merevlemezen elérhető területet, így végül körülbelül 1,5 GB jutott az adatoknak. A beállítások elvégzésére összesen három órát fordítottunk.

NFS

Az értékelemző ismeri az NFS-t is, ezért a következő lépés ennek a beállítása volt. A */etc/exports* fájlba mindössze egy sort kellett beírnunk:

```
/home/guest/hptestdata
➥192.168.10.29(rw)
```

A vendég (guest) saját könyvtárában létrehoztuk a ***hptestdata*** könyvtárat, és újraindítottuk az `nfsd`-t. Ez a sor csak egyetlen IP-címnek engedi meg a könyvtár befűzését. Az értékelemző kezelőszervén megadtuk a szükséges beállításokat, és megnyomtuk a ***mount*** (befűzés) gombot. Elsőre természetesen nem indult. A hibaelemzés csak egy percet vett igénybe, és az analizátor és a vendégfiók azonosítószámának összehangolása megoldotta a gondot. Az NFS beállításával összesen öt percet töltöttünk el.

Samba

A Samba rendkívüli program, és nagyon sok mindenre képes. Ez egyszerű feladat, szemléltetésül */etc/smb.conf* fájlunk a listán (30. CD Magazin/Samba) látható. A rendszer biztonságában döntő szerepet játszó adatokat, mint például a hálózat tartománynevét itt és a */etc/exports* fájlban is megváltoztattuk. A fájl legfontosabb része a ***hptestdata*** megosztás létrehozása és csak olvashatóvá tétele. Azért kell csak olvashatóvá tenni, nehogy a felhasználók véletlenül kitörölhessék az adatokat. Időnként kiürítjük a könyvtárat, de csak azután, hogy minden felhasználóval egyeztettünk. A Samba beállításának másik része az, hogy a rendszerindítófájlokat úgy módosítsuk, hogy az `nmbd` leálljon. Az általunk használt rendszerbeállítások mellett nem szeretnénk, ha a gép látszódna a hálózaton, ezért azt sem akarjuk, hogy az `nmbd` elvégezze a névfeloldást. A saját terjesztésed leírásában megtalálhatod, hogy melyik az a fájl, amelyet módosítanod kell. A Red Hat 6.2-ben az ***S91smb*** fájlt változtattuk meg, és az `nmbd`-t indító sorokat megjegyzéssé tettük, azaz a megfelelő sorok elejére beszúrtunk egy # jelet. Hogy erre a hálózati beállításra később is emlékezzek, a magyarázó sort is módosítottam, ami most már annyit mond, hogy az `nmbd` nem indul el. Rendes körülmények között a program azt jelzi vissza, hogy az `nmbd` elindul. A hozzáférést a mi tartományunkra korlátoztuk, ezzel kívülről nem férhetnek hozzá a géphez. A beállítások elvégzése összességében jó néhány órányi ügyeskedésbe telt.

NT-beállítások

Utoljára az NT-s gépet kellett beállítanunk. Ezt a trükköt még sehol sem láttuk, ezért gondoljuk, hogy nagyon jópofára sikeredett. A linuxos gép számára létrehoztunk egy adatmegosztást, a felhasználók az asztali gépükről ide fordulnak az adatokért. Ezután az UNC (egységes elnevezési szabvány) segítségével egy hálózati parancsállományt készítettünk, amit az adatmegosztási pontra irányítottunk. A Samba-megosztást a hálózaton egy percre láthatóvá kellett tennünk – ekkor hoztuk létre a parancsállományt a könyvtárban. Egyszerűbb volt így tennünk, mint megszenvednünk a megkettőzött fordított perjelek helyes beállításával. Amikor a felhasználó az NT-kiszolgálóhoz fordul, a megosztott könyvtárat fogja látni. Ha kétszer rákattint, a könyvtárat meg is nézheti. Amennyiben viszont duplán kattint a könyvtáron, akkor anélkül, hogy észrevenné, a linuxos gép próbaadatokat tartalmazó könyvtárába jut. Erre a trükkre azért van szükség, mert a Windows nem képes olyan hálózati meghajtót megosztani, amelyet már befűzött. Az eredeti tervem az volt, hogy az NT-s gép a Samba-megosztást befűzi egy meghajtóra, és majd onnan lesz megosztva. Miután rájöttünk, hogy a Windows nem képes a befűzött meghajtókat megosztani, és bevetettük ezt a trükköt, összesen öt percet töltöttünk a beállításokkal.

Összegzés

A Linux és a Samba a laboratórium olyan igényét elégítette ki, amelynek kiszolgálására másként nem lett volna mód. A módszer a felhasználók számára is átlátható, mert továbbra is ugyanarra a központi helyre kell menniük az adatokért; éppannyira biztonságos, amennyire az intézet NT-kiszolgálója, és szó szerint ingyen készült el, ugyanis a kiépítéséhez teljes egészében összeszedegetett alkatrészeket használtunk fel.

A jövő

Ez a megoldás azonban még nem nyújt teljes biztonságot. Egy agyafúrt számítógép-felhasználó megnézheti a hálózati parancsállomány tulajdonságait, és utána létrehozhat egy a Samba-kiszolgálóra mutató közvetlen parancsállományt, megkerülve ezzel az NT-n levő biztonsági eljárást. A másik lehetőség az lenne, hogy a linuxos gépet és az `smbmount`-ot használva megosztást készítünk az NT-s gépen, majd pedig az NFS segítségével exportáljuk a próbát futtató készülékre. Az NT-megosztást sikerült a linuxos géphez csatolnunk, majd az NFS-sel exportálnunk és befűznünk a 4155B készülékhez. Továbbra is gondot jelent viszont, hogy erre a megosztásra írni is tudjunk, ezt azonban még az `smbmount` beállításainak használatával sem sikerült megoldanunk. Reméljük, a közeljövőben lesz időnk rá, hogy ezzel a feladattal is újból foglalkozzunk.

A források megtalálhatóak a 30. CD Magazin/Samba könyvtárban.

Brian Gollsneider (balra) és *Mike Martin* (jobbra) a Marylandi Egyetem villamosmérnök szakos, másod-, illetve első diplomáján dolgozó hallgatói. A Hadi Kutatólaboratórium tudósaival együtt vizsgálják a VCSEL-eket. Brian Gollsneider elérhető a gollsneb@glue.umd.edu címen.

A Squirrelmail

A Squirrelmail a legtöbb terjesztésben már telepíthető csomagként is szerepel, azonban érdemes a használatuktól eltekintenünk. Miért? Két okból is. Először: az újabb változatokkal számos új szolgáltatás, illetve jobb teljesítmény érhető el. A második számomra jelentős előny, hogy az új változatokban általában a hibajavítások is megtalálhatók. A hibák széles skálán mozognak, a biztonsági lyukak attól kezdve, hogy más felhasználó leveleit is el tudjuk olvasni egészen odáig terjednek, hogy – bizonyos körülmények között – távolról bárkik bármilyen parancsot végre tudnak hajtani a kiszolgálón. Ehhez ráadásul további kisebb, inkább csak boszszantó jelenségek járulnak. Az alkalmazást ugyanis PHP nyelven írták, ezért a benne rejlő hibákat más is kihasználhatja. Miért választottam mégis ezt a webes levelezőügyfelet? Mert minden hibája ellenére szolgáltatásainak sora a legjobbak közé emeli, és a többi ügyfélprogramnak – vagy programozási nyelvnek – is megvan a maga hibája mind biztonsági, mind az elvárt működés terén. Kezdjünk hát neki a telepítésnek, hogy felhasználóink minél hamarabb használhassák.

Az első lépések

Legelső lépésként töltsük le a programot a ➲ http://www.sqirrelmail.org címről. A legújabb változat jelenleg a 1.2.5-ös. Ez már fel van készítve a PHP új biztonsági és egyéb szolgáltatásaira. Ha letöltöttük, csomagoljuk ki a tömörített fájlt – feltételezve, hogy a lent szereplő csomagot töltöttük le és Debian-rendszert használunk, valamint a */var/www/mail* könyvtár létezik.
Ha nem, nulladik lépésként hozzuk létre a könyvtárat:

```
root@mail # mkdir /var/www/mail,
```

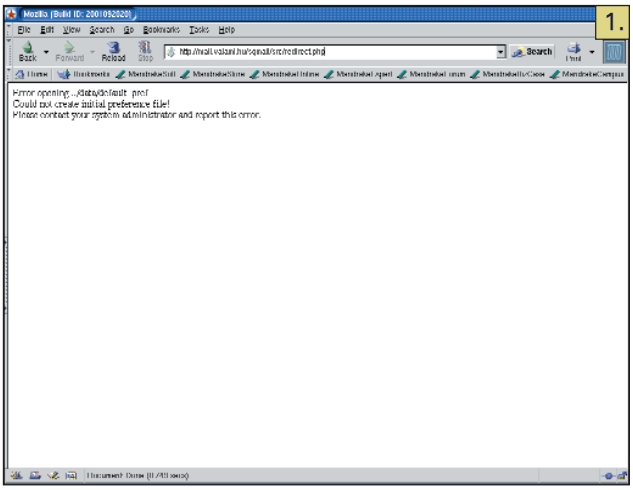
majd:

```
root@mail # tar xvfz squirrelmail-1.2.5.tar.gz
➥-C /var/www/mail
```

Ekkor a csomagban lévő összes PHP-kód, nyelvi modul és a kép a könyvtár alatt kerül kicsomagolásra. Mivel ez PHP-értelmező nyelv, fordítási teendőnk nincsen (olyan kereskedelemi termék is kapható, amely a PHP-kódot bináris formára képes alakítani).
Mielőtt azonban azt hinnénk, hogy készen is vagyunk, ellenőrizzük a webkiszolgáló beállítófájlját. Keressük meg, szerepel-e benne olyan bejegyzés, amely a kiszolgálót a PHP-kód értelmezésére utasítja, illetve rendelkezik-e az utasítások fordításához szükséges modullal. Azt feltételezve, hogy a kiszolgálóprogram az Apache és a rendszer a Debian, keressük meg a következő fájlokat (vagy fájlt): */etc/apache/httpd.conf*, illetve */etc/apache/srm.conf*. A *httpd.conf*-ban a következő bejegyzést kell látnunk:

```
LoadModule php4_module
➥/usr/lib/apache/1.3/libphp4.so
```

Ez tölti be az Apache-kiszolgálóhoz tartozó PHP-modult. Előfordulhat, hogy más lesz a neve – én forrásból telepítettem. A csomagból felrakott PHP-modul ugyanúgy megfelel, mint a saját magunk által fordított. Amennyiben csomagból telepítettük, telepítésekor ennek a sornak kell szerepelnie a beállítófájlban, ha ez mégsem történne meg, akkor feltétlenül írjuk bele. Velem már előfordult, hogy ugyan beíródott a beállítások közé, de # jel előzte meg, így a webkiszolgáló



a PHP-kódokat teljesen érthető módon nem értelmezte. Ha a PHP-t csomagból rakjuk fel, az IMAP-modult is telepítenünk kell ahhoz, hogy működő Squirrelmail-telepítéssel rendelkezzünk. Debian-rendszer esetén a többi beállítást az *srm.conf* fájlban, más rendszer esetén mindezt még mindig a *httpd.conf*-ban folytatjuk. Keressük meg, hogy a következő sor szerepel-e a beállítások között:

```
AddTypeapplication/x-httpd-php .php .phtml*
```

Enélkül a sor nélkül ugyanis az Apache még mindig csak a kódokat írja ki, és nem fordítja le, illetve futtatja azokat. Összefoglalva: ha az Apache-val és a hozzávaló PHP-modul(okk)al és a megfelelő beállításokkal rendelkezünk, kipróbálhatjuk az ügyfelet. Egy böngészőbe írjuk be:

```
http://kiszolgalocime/mail
```

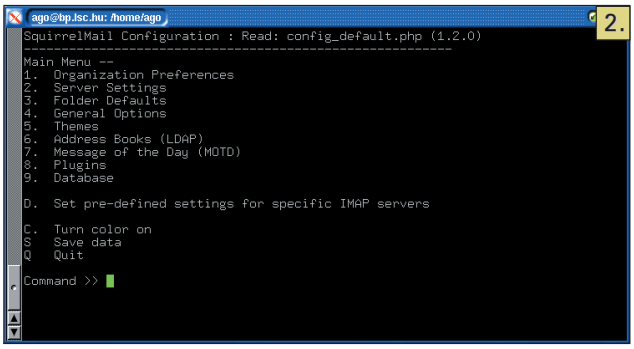
Sajnos a következő kép fogad bennünket: *1. kép*.
Mindig ebbe fogunk ütközni, ha a jogosultságok nem megfelelők a fájlrendszeren.Vegyük figyelembe, hogy a PHP-kódokat a webkiszolgáló futtatja. Mivel a program bizonyos könyvtárait írni akarja, ezekre a webkiszolgálót futtató felhasználónak írási joggal kell rendelkeznie.
Lépjünk be a */var/www/mail* könyvtárba és az alapértelmezett data könyvtárat adjuk át a webkiszolgálót futtató felhasználó jogosultságának:

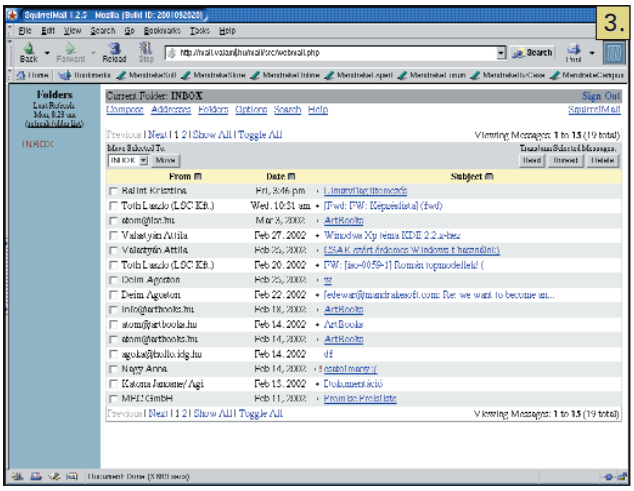
```
root@mail:/var/www/mail# chown
➥-R www-data.www-data data/
```

A felhasználónév nem Debian-rendszer esetén általában ***nobody*** vagy ***httpd***. A megfelelő eredményt értelemszerűen a megfelelő felhasználót behelyettesítve érhetjük el.
A következő hibajelenség akkor fogad(hat) minket, ha a program környezetét nem állítottuk be megfelelően.
A hiba javításához a kiszolgálón lépjünk be a */var/www/mail* könyvtárba, ahol indítsuk el a `configure` programot. Ez nem a szokásos programok fordítása előtti parancsfájl, hanem egy Perl-program, mely segít elvégezni a beállításokat.

```
root@mail:/var/www/sqmail# ./configure
```

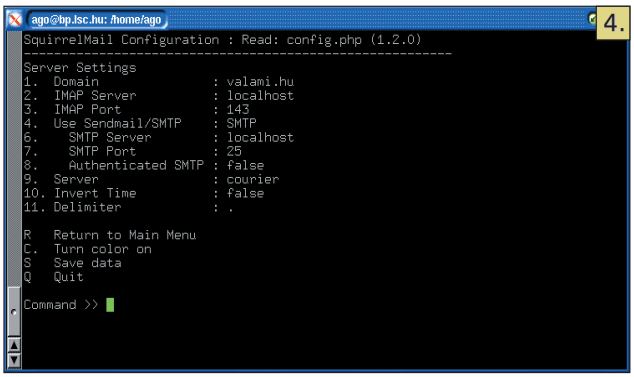
Ekkor a *2. képhez* hasonlót láthatunk.
Itt a menüpontok között haladva állíthatjuk be a kiszolgálón





futó programokhoz illő környezetet. A leggyorsabb út, ha a billentyűzeten lenyomjuk a ***D*** betűt, és kiválasztjuk, melyik IMAP-kiszolgálót használjuk a számítógépünkön. Ekkor, ha a többi beállítás is megfelelő, a rendszert már használhatjuk is. Siker esetén a *3. képhez* hasonló látvány fogad bennünket.
Előfordulhat, hogy a kiszolgáló beállításai nem megfelelőek. Ezért nézzük meg, hogy melyek a legfontosabb beállítások, illetve milyen lehetőségeink vannak még a beállítóprogramban. A főmenüben a második pontot választva (***Server Settings***) juthatunk el a kiszolgálóoldali beállításokhoz. A *4. képen* szereplők kell fogadjanak bennünket.
Beállítási lehetőségeink a következők:

1. `Domain`: melyik tartományhoz tartozunk, azaz mi fog látszani tartománynévként a kimenő leveleken.
2. `IMAP server`: hol található az IMAP-kiszolgáló. Itt adhatunk meg hálózati címet vagy a kiszolgáló nevét is. Látszik tehát, hogy ha egy központi levelezőkiszolgálóval rendelkezünk, akár több webkiszolgálót is elétehetünk, csak be kell állítanunk, hogy hol találhatók meg.
3. `IMAP port`: melyik kapun hallgatódzik az IMAP-kiszolgáló. Ez alapértelmezettként a 143-as, de nyugodt szívvel beállíthatunk ide nagyobb (1024 vagy afölötti) kapuszámot is.
4. `Use Sendmail/SMTP`: a levélküldés módját határozhatjuk meg. Ha a ***Sendmail*** módot választunk, a Squirrelmail a küldéshez a kiszolgálón található Sendmail binárist fogja használni. Ez a bináris az összes levelezőkiszolgálóban (MTA) megtalálható, hogy a „nagypapával”, a Sendmaillel megőrizzék a megfelelőséget. Ennél jobb ötletnek tartom, ha a levél elküldése SMTP-n keresztül történik, mert azt sok esetben gyorsabb átadni egy olyan démonnak, amely állandóan figyel az adott kapun, mint elindítani egy binárist a fájlrendszeren.



5. Ha a Sendmail binárist választottuk, megjelenik az 5-ös pont, hogy beállítsuk, hol található meg a rendszeren a Sendmail binárisa.
6. Ha az SMTP-t választottuk a küldésre, a kiszolgáló helyét adhatjuk meg – természetesen itt is választhatunk más kiszolgálót, akár a helyi gépet is.
7. Az SMTP-kiszolgálóhoz tartozó kaput adhatjuk meg, itt is eltérhetünk az alapértelmezett 25-ös kaputól.
8. `Authenticted SMTP`: használ-e az SMTP-kiszolgáló valamilyen azonosítást a levél elfogadására. Alapértelmezettként nem, és ha `smarthost`-ként a Squirrelmail futtató kiszolgáló rendelkezésére áll, akkor nem is kell. Ezt csak akkor engedjük meg, ha a gépek mások által nem elérhető alhálózaton tartózkodnak.
9. Server: az IMAP-kiszolgáló típusát állíthatjuk be (`courier`, `cyrus` stb.)

A másik két beállítás az időzónákra és a könyvtárak közötti IMAP-jellemző elválasztásra vonatkozik, de ezeket bízzuk a rendszerre, hadd kezelje őket. Az IMAP-beállításoknál használjuk az alapértelmezett értékeket.

Következő cikkemben a Squirrelmail alá telepíthető bővítményekről és levelezéshez kapcsolódó érdekességekről esik szó.

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.

Bevezetés a Tkinter használatába (4. rész)

Záróakkordok…

Bevezetésünk végéhez érkeztünk. Mostanra már tudjuk, hogyan néz ki egy Tkinter-program és megismerkedtünk a rendelkezésünkre álló lehetőségekkel. Ha figyelmesen követtük a cikksorozat eddigi részeit, és olykor a leírást is elővettük, valószínűleg már magunk is képesek vagyunk egyszerű programok írására.
Sorozatunk elkövetkező részeiben zavarosabb vizekre, ismeretlen tájak felé evezünk, vagyis komolyabb témákkal foglalkozunk. Szolgáljon ez a rész hídként, amellyel a már ismertet az ismeretlennel összekötjük!
Minden program egyik legfontosabb jellemzője a sebessége. Különösen igaz ez az olyan programokra, amelyek folyamatos felhasználói beavatkozást igényelnek, egy kérdőívhez olvasnak be adatokat, képeket jelenítenek meg, vagy éppen kiszámolnak valamit.
Az ilyen programoknál a felhasználó elvárja, hogy miután leüt egy billentyűt vagy megnyom egy gombot az alkalmazás ablakában, azonnal valamilyen válasz következik be: megjelenik az adott betű, beugrik valamilyen új ablak – tehát a program valamiféle jelét adja annak, hogy bizony működik. A felhasználó sokszor a program e viselkedése alapján dönt két alkalmazás között, és előfordulhat, hogy azt ítéli meg jobbnak, amelyik minden eseményre azonnal reagál, még ha a kiválasztott program az adott dolgot rosszabbul végzi is el.
Képzeljük el, hogy szöveget gépelünk be egy szövegbeviteli mezőbe, netán a beviteli mezők között ugrálunk a TAB gombbal. Kevés dolog zavaróbb annál, mintha a betűk nem jelennek meg azonnal, ahogy beírjuk őket. Olyan esettel is találkoztunk, hogy űrlapok sorozatát kellett kitöltenünk, és minden egyes űrlap után hosszú másodperceket kellett arra várnunk, hogy a következő ablak betöltődjön.
A most következőkben néhány olyan trükkel ismerkedünk meg, amelyekkel alkalmazásainkat gyorsabbá, hatékonyabbá és kezelhetőbbé tehetjük.

Kapcsoljunk sebességbe!

Ha már programindításnál a felhasználó kedvében szeretnénk járni, ne várakoztassuk meg feleslegesen! Az egyik legtöbb időt igénylő folyamat a Python-kód bájtkódra történő lefordítása, mely végül ténylegesen futni fog. Ha programunk sok-sok ezer sort tartalmaz, ez bizony nem kevés időt vesz igénybe. Viszont tudjuk, hogy a rendszer moduljainkat önműködően bájtkódra fordítja le és tárolja is őket ilyen formában, ***.pyo*** vagy ***.pyc*** kiterjesztéssel. A következő futáskor – ha a bájtkód újabb, mint a ***.py*** kiterjesztésű fájl – anélkül, hogy az eredeti kóddal bármit kezdene, a Python azonnal a bájtkódhoz fordul. Programunk indítófájlját a Python ugyanakkor lefordítja, de a bájtkódú állományt nem tárolja. Ez nem éppen szerencsés, tekintve ha alkalmazásunk logikája és a temérdek kód ebben az indítófájlban van tárolva. Nincs más teendőnk, mint ezt a fájlt modullá alakítani, és létrehozni egy néhány soros fájlt, ami ezt a modult importálja.
Miről is van szó? Tegyük fel, hogy van egy alkalmazásunk, mely a következő programsorokat tartalmazza:

```
from Tkinter import *
import Pmw

class EgyOsztaly(Dialog):
        sok ezer sor

root = Tk()
peldany = EgyOsztaly(root)
```

Ha ezt a fájl közvetlenül a Pythonnal hívjuk meg, a Python értelmezője minden egyes alkalommal bájtkóddá fordítja le, és csak azt követően hajtja végre.
Alakítsuk át a program utolsó két sorát függvénnyé, és szúrjuk be eléjük a `def indito:` sort! Ha most ezt az `indito` eljárást egy másik indítófájlból hívjuk meg, akkor ezt a sok ezer kódsort tartalmazó modult máris csak egyszer, a legelső alkalommal fordítja le nekünk a Python, és az eredményt bájtkódba menti!
Az új indítófájlunk így néz ki:

```
import RegiIndito
RegiIndito.indito()
```

És máris rengeteg időt takarítottunk meg, mert a Pythonnak minden alkalommal csak két sort kell lefordítania, a többi már a bájtkód formátumú fájlokból töltődik be.
Ha a Pythont már első indításkor az `-O` kapcsolóval hívjuk meg, programunk még gyorsabb lesz, mivel a Python egyszerűsíti nekünk a kódot, illetve a bájtkódba nem fordít bele olyan dolgokat, amelyekre futáskor nincs szükség (például nem tárolja minden művelethez kapcsolódóan, hogy eredetileg a forráskód melyik sora tartalmazza azt). Az ilyen módon fordított modulok ***.pyo*** kiterjesztést kapnak, mely a ***.pyc*** fájlokhoz hasonlóan a futtató rendszertől úgyszintén függetlenek.

Egyszerűsítsünk!

A legfontosabb szabály, hogy programunkat úgy tervezzük meg, hogy a Python értelmezője minél kevesebb időt töltsön programunkban, és inkább a saját belső függvényeiben tengesse az idejét. Ez annyit tesz, hogyha egy műveletre létezik valamilyen Python-függvény, azt részesítsük előnyben. Ha mindent saját magunk kódolunk, a ciklusok futása során értékes időt veszítünk, ezért jobban tesszük, ha a Python C-ben írt függvényeire bízzuk magunk.
Gyakori eset, hogyha egy adott függvényben egy külső változóra van szükség, a függvény elején a külső változót egy helyi változónak adjuk értékül. Ez viszonylag egyszerű művelet, de ha a függvényünk mondjuk, valamilyen ciklusból hívódik meg, amely akár több ezerszer végrehajtódik, akkor ilyen apróságoknak is érdemes figyelmet szentelnünk. Így mielőtt helyi változót hoznánk létre, gondoljuk meg, valóban szükség van-e rá. Ha a helyi változóhoz a függvényből csak egyetlen alkalommal férünk hozzá, nem érdemes rá értékes órajelciklusokat pazarolni. Ellenben ha a függvényünkben többször hivatkozunk rá, netalántán a függvény egy újabb ciklust tartalmaz, akkor jobban tesszük, ha létrehozzuk azt a helyi változót. Ilyen módon elérhetjük, hogy a külső

változót csak egyszer kelljen elérni, azután pedig használhatjuk a jóval gyorsabban elérhető helyi változónkat.
Vannak esetek, amikor megszokásból létrehozunk változókat, amiket aztán soha többet nem használunk, vagy csak egyetlenegyszer. Ilyen helyzetben mindig tegyük fel magunknak a kérdést: hogyan oldható meg a dolog egyszerűbben? Ha például egy `Label()` objektumot hozunk létre, amelyre csak egyszer hivatkozunk – amikor kitesszük a képernyőre –, akkor jobban tesszük, ha változó megalkotása helyett a Pythonra bízzuk a dolgot, és egy választékos `Label().pack()`-kel oldunk meg mindent.
Ha mégis ciklusok írására adjuk a fejünket, gondoljuk át még egyszer, nincs-e az már adott feladatra belső függvény… Biztos nem megoldható a dolog se a `map()`-pel, se a `reduce()`-szal, se a `filter()`-rel? Ha nem, hát nem.
Vegyünk egy példaciklust:

```
for i in range(10000):
    for j in range(100):
      import EgyModul
      a = a + EgyModul.EgyOsztaly.EgySzam * 2
```

Ez a ciklus tízezer alkalommal egy százelemű listát hoz létre, 10 000-szer 100 alkalommal importál egy modult, és ugyanennyiszer fér hozzá egy külső változóhoz. Ha a fenti kódot picit átírjuk, jelentős sebességnövekedést érhetünk el:

```
import EgyModul
b = EgyModul.EgyOsztaly.EgySzam

szaz = range(100)

for i in range(10000):
    for j in szaz:
        a = a + b * 2
```

A második kód nyolcszor gyorsabb az elsőnél! Lássuk, hogyan értük el: a második esetben a modul csupán egyetlenegyszer importálódik, a százelemű listát egyszer hozzuk létre, és külső változót is egyszer olvasunk be. Ha az `a` kezdőértékének nullát veszünk, az `EgyModul.EgyOsztaly.EgySzam` értékének pedig hármat, akkor mindkét esetben hatmilliót kapunk végeredményként. Láthatjuk tehát, hogy egyáltalán nem mindegy, hogyan oldunk meg egy nehézséget – akár egy egyszerű számolást is.
Vegyünk két kifejezést:

```
szoveg = szoveg1 +      + szoveg2 +      + szoveg3
```

és

```
szoveg =  %s %s %s  % (szoveg1, szoveg2, szoveg3)
```

A második esetben – C-kódban gondolkodva – az egész egy egyszerű függvénnyel megoldható, míg az első esetben memóriaműveletek sorára van szükség. Így már nem olyan meglepő, hogy az utóbbi megoldás közel ötször gyorsabb az elsőnél.
Egyetlen szabály van tehát, amit fejben kell tartanunk, mégpedig az, hogy a kódunk minél rövidebb és egyszerűbb legyen. Gondolkodjuk a Python „fejével”, és amit csak lehet, bízzunk rá.

Tegyünk úgy, mintha gyorsak lennénk!

Mint már szó volt róla, grafikus felület tervezésénél a felhasználó számára a legfontosabb annak látszata, hogy programunkat gyorsnak lássa. Ha a képernyő előtt ülve napjában többször is űrlapok során kell végigverekednie magát, nem mindegy, mennyit kell várnia az egyes űrlapok között. Az időigényes műveleteket lehetőség szerint úgy kell csoportosítani, hogy a felhasználónak inkább egyszer kelljen hosszabb ideig várnia.
Ha a felhasználót sokszor váratjuk, programunkat lassúnak fogja érzékelni, míg ha csak a kitöltés befejeztével kell várnia, ő már léphet is tovább a következő munkájára, programunk pedig a háttérben dolgozik.
Figyeljünk arra is, hogy az olyan műveleteket végképp kerüljük el, melyeket a grafikus felület kirajzolásakor hajtanánk végre. A lassan, akadozva felbukkanó ablakok a felhasználóban azt az érzést keltik, hogy a program hibásan működik.
Hasznos, ha az elemeinket tartalmazó kereteket (Frame) csak azután tesszük ki a képernyőre, miután a keretben található elemek elkészültek. Ha fordítva járunk el, a felhasználó esetleg villogást tapasztal – ez annak következménye, hogy az elemkezelő minden új elem kihelyezésekor a többi helyzete alapján számolásokat végez, hogy az új hova is kerüljön.
A felhasználó tényekedése által keltett eseményekre a lehető leggyorsabban próbáljunk meg válaszolni. A puszta egérmozgatás is események egész sorát indítja el, ami – ha csak az egérmutató koordinátáira van szükségünk – nem is baj, de ha egy ilyen gyakran keletkező eseményhez olyan függvényt rendelünk, amely bonyolult elemet rajzol ki, akkor ennek a rendszer teljesítményére nézve súlyos következményei lesznek.
Azért erre is van megoldás: ha mi mégis ilyen bonyolult műveletet szeretnénk elvégezni, akkor lehet, hogy elegendő minden 5. eseményre válaszolunk, vagy meghatároznunk, hogy a kirajzolás legfeljebb csak bizonyos időközönként hajtódjon végre, a többi esetben pedig egyszerűen hagyjuk figyelmen kívül az eseményt.

Keressük meg a szűk keresztmetszetet!

Ha programunk futása során úgy érezzük, hogy mozgása minden igyekezetünk ellenére mégis döcögős, és nem tudjuk, melyik függvényben időzik el, jó szolgálatot tehet a `profile` modul. E remek kis modul segítségével könnyen megtalálhatjuk, hol akad el programunk futása.
Amennyiben programunk egy `indito()` függvény meghívásával kezdi el a munkát, annyit kell tennünk, hogy programunk elejét a következőképpen alakítjuk át:

```
import profile
profile.run( indito() )
```

Miután ez lefutott, a profiler megjeleníti a statisztikát, hogy programunk hány másodpercet töltött el az egyes függvényekben.
Ez a statisztika szedett-vedett, tegyük kicsit rendbe:

```
import profile
import pstats

profile.run( indito() ,  eredmenyek.p )
p = pstats.Stats( eredmenyek.p )
p.sort_stats( cumulative ).print_stats(10)
```

Ebben az esetben a **profile.run()* létrehoz egy ***eredmenyek.p*** nevű fájlt, mely a profiler kimenetét tartalmazza a `pstat` modul által emészthető formában. Ezt követően a fájlt a `Stats()` függvénnyel beolvassuk, és megjelenítjük a tíz legtöbb időt igénybevevő függvényt.

Gludovátz Gábor
(ggabor@sopron.hu) Kedvenc időtöltéseinek egyike a programozás és a Linux lelkivilágának alaposabb megismerése. Gyakran éjszakázik a monitor előtt ülve hosszú kódsorok társaságában, és ha ideje engedi, a soproni erdőkben teker kedvenc bringájával.

Bemutatkozik a Zope

A Zope alkalmazáskiszolgáló birodalmában tett villámlátogatásunk során a Zope-ot először mint webfejlesztőrendszert vizsgáljuk meg.

Mindegy, milyen programnyelvet vagy operációs rendszert használunk, a webfejlesztés alapvetően HTML-fájlok, képek, önálló programok, hibrid HTML, illetve -kódvázlatok és adatbázis-kapcsolatok tervezéséről szól. A tapasztalt fejlesztők könnyedén tudnak Perlről PHP-re vagy ASP-re váltani, hiszen az egyik nyelvről a másikra váltás az elgondolás szintjén csak a lehető legkisebb változást okozza. A paradigmák effajta közeledése a programozóknak igen kényelmes, igaz, ennek az az ára, hogy a webfejlesztői közösség berkeiben egyre gyakrabban tapasztalhatjuk az önelégültség és a lustaság jeleit.
Szerencsére itt van nekünk a nyílt forráskódú Zope alkalmazáskiszolgáló, amely felrázhat minket langymeleg elégedettségünkből, és a webfejlesztés teljesen új irányába vezethet bennünket. A Zope, amelyet a Zope Corporation (korábbi nevén Digital Creations) működtet, szinte mindent megváltoztat, amit a webfejlesztésről valaha is tudni véltünk. Ugyan továbbra is dinamikusan készülő tartalommal és relációs adatbázisokkal dolgozunk, de a Zope ezt a piacon ma megtalálható alkalmazáskiszolgálóktól teljesen eltérő módon teszi.
Ebben a hónapban a Zope-ot webfejlesztési szemszögből vizsgáljuk meg. Remélem, sikerül rávilágítanom, miben is különbözik a Zope minden más környezettől. Az általa képviselt finomság és hatékonyság bizony igen esélyes versenyzővé emeli a nyílt forrás rajongóinak kegyeiért folytatott harcban.

Mi is a Zope valójában?

A Zope megértésének egyik fő nehézsége, hogy valójában nem egy, hanem több dologról van szó. Egyszerűbben fogalmazva: a Zope minden, amire csak a webalkalmazások fejlesztéshez szükségünk lehet. A Zope telepítésével egy egyszerű HTTP-, FTP- és Web-DAV-kiszolgálót (amely Zserver néven ismert), egy objektum-adatbázist (ZODB), illetve egy kiszolgálóoldali alkalmazások fejlesztéséhez szükséges keretrendszert kapunk.
A Zope legnagyobb része Pythonban íródott, ami azt is jelenti, hogy a különféle felületek között könnyedén mozgatható. Bár köztudott, hogy a Perl, a Python vagy a Java nyelveken írt programok Linux- és Windows-rendszereken egyformán könnyen futtathatók, még mindig elég szokatlan, hogy egy kiemelkedő nyílt forrású rendszer Windows alatt is fut.
Egy átlagos honlap statikus HTML-fájlok, sablonok és grafikus állományok keveréke. Mikor a HTTP-kiszolgáló kérelmet fogad, először is meghatározza, hogy milyen típusú fájlkérésről van szó (ezt vagy a fájl kiterjesztéséből, vagy a MIME-típus alapján tudja eldönteni), majd ha szükséges, a megfelelő programsorokat végrehajtja, végül visszaküldi a HTTP-választ.
A Zope esetében azonban semmi sem tárolódik a fájlrendszeren. Ehelyett a teljes laptartalom és a programok a ZODB objektum-adatbázisban tárolódnak (az adatbázis maga általában egyetlen hatalmas lemezállomány formájában létezik, amelyet menteni lehet, vagy le lehet másolni, így készítve biztonsági mentést a Zope-kiszolgálóról.)
Ha például egy egyszerű HTML-fájlra van szükségünk, létre kell hoznunk egy „file” objektumot a ZODB-ban. Ha a honlapon képet szeretnénk megjeleníteni, a ZODB-ban egy képobjektumot kell létrehoznunk. Ugyanígy ha egy bizonyos URL meghívása esetén egy adott kódot szeretnénk futtatni, ezt a kódot is a ZODB-ban tároljuk.
Szerencsére az elemek tárolása és lekérése a ZODB-ban igen egyszerű. A Zope csaknem mindent elvégez helyettünk, és néhány magas szintű kapcsoló beállítását már a telepítéskor lehetővé teszi. Továbbá a Zope-ot úgy fejlesztették ki, hogy webböngésző segítségével is tökéletesen vezérelhető legyen. Alkalmazhatjuk a Zope webalapú eszközeit, amelyek segítségével a honlapot anélkül szerkeszthetjük és karbantarthatjuk, hogy egyetlen böngészőn kívül bármi mást használnunk kellene. Tehát lehetőségünk nyílik rá, hogy a honlapon bármely tartalmat (HTML, képek vagy könyvtárak) mindössze a böngészőnket használva készítsünk el, módosítsunk vagy töröljünk. Ha szöveget (és kódot) inkább az Emacsban szeretünk szerkeszteni, semmint a Web szövegmezőiben, kapóra jön a ZServer FTP-felülete, amely a ZODB objektumait úgy jeleníti meg, mintha fájlrendszer-hierarchiát néznénk. Ha mi (illetve az általunk kitanított avatatlan csoport) a fájlok webkiszolgálóra való felviteléhez és lehozatalához korábban már használt FTP-t, akkor a ZServer FTP-megoldásával a megszokott lépéseket szinte változtatás nélkül folytathatjuk.

A Zope telepítése

Mindezen összetett szolgáltatások ellenére a Zope-ot meglepően egyszerű telepíteni. Le kell töltenünk a legfrissebb változatot a ➲ http://www.zope.org-ról (lásd a ***Kapcsolódó címek*** részt). Miután a letöltendő változatot kiválasztottuk, váltsunk Zope könyvtárunkba (ez általában a */usr/local/zope/*, de bármi megteszi), és csomagoljuk ki a Zope fájlt:

```
mkdir /usr/local/zope
cd /usr/local/zope
tar zxvvf /downloads/Zope-2.4.3-linux2-x86.tgz
```

A Zope archívállomány megnyitásával számos könyvtár bukkan elő:

- ***bin***, ahol a Zope indító és leállító parancsfájljai rejtőznek;
- ***doc***, amelyben a teljes Zope-leírás megtalálható;
- ***lib***, ebben a Python- és minden egyéb Zope-alkalmazás (más néven termékek (products)) tárolódnak;
- ***ZServer***, amely a Zserverhez szükséges osztályokat tartalmazza;
- ***utilities***, amely néhány Zope-pal kapcsolatos segédeszközt foglal magában.

Mielőtt nekikezdhetnénk a Zope-pal való munkának, előbb a telepítő parancsfájllal telepítenünk kell. Ezt a parancsfájlt azonban csak akkor futtassuk le, ha a Zope-fájlokat már a

végleges helyükre mozgattuk, mivel a telepítés bizonyos teljes rendszerre érvényes értékek beállításához a működő könyvtár nevét használja fel.
Miután a Zope telepítését befejeztük, a fő Zope könyvtárban található `start` parancsfájl segítségével akár el is indíthatjuk. Ez a parancsfájl a Zope HTTP-kiszolgálót alapértelmezés szerint a 8080-as kapun indítja el (egy másik kapun pedig az FTP-kiszolgálót). Ezeket az értékeket parancssori kapcsolókkal megváltoztathatjuk; a kapcsolók teljes listáját a `start -help` begépelésével kaphatjuk meg.
Bár a Zope főként Pythonban íródott, nem feltétlenül szükséges, hogy Python telepítve legyen a gépünkön. A Zope saját Python-másolattal rendelkezik, és ezt fogja használni, bármi legyen is telepítve az operációs rendszeren. Ezáltal a változatütközések és hibák esélye mérséklődik. Jelenleg a Zope Python 2.1-et használ (azaz az utolsó megbízható változatot), de hamarosan megjelenik a Python 2.2 – érdekes lesz megfigyelni, mikor fogja a Zope beilleszteni.
A telepítő parancsfájl nemcsak a Zope alapbeállítását végzi el, hanem a rendszer `admin` felhasználója számára egy nehezen kitalálható jelszót is készít. Hamarosan szükségünk is lesz erre a jelszóra, úgyhogy ne felejtsük el feljegyezni valahová.

Zope Felügyelet

Ha a Zope-ot üzembe helyeztük, hogyan használhatjuk? Nos, az első és legfontosabb lépés a böngészőnk elindítása, majd a saját gépünk felkeresése a http://localhost:8080/ URL megadásával. Így néhány kezdeti Zope-adathoz jutunk, többek között például a leírásra és a honlapokra mutató hivatkozásokhoz (előfordulhat, hogyha esetleg más is fut a 8080-ason, az gondot okoz, ugyanis például a `wwwoffle` is azon keresztül fut – a ford.). Hagyjuk itt ezt a bemutatkozó lapot, és lépjünk tovább a Zope fő kezelőfelületére, a http://localhost:8080/manage címre. Itt meg kell adnunk a karbantartó felhasználói nevét és jelszavát – használjuk azt az `admin`-jelszót, amit a telepítő parancsfájl írt ki nekünk, amikor a Zope-ot először telepítettük.
Amennyiben a jelszót helyesen gépeltük be, böngészőablakunk két függőleges részre oszlik: a bal oldali rész a Zope-kiszolgáló szerkezetét mutatja, míg a jobb oldali részben az éppen kiválasztott objektumot vizsgálhatjuk meg részletesen.

Alapvető tartalom készítése

Nem nehéz megérteni, mi történik akkor, amikor a */foo/bar* dokumentumot lekérjük az Apache-kiszolgálótól. Az Apache megnézi, van-e ***bar*** nevű fájl a ***foo*** könyvtárban a dokumentumgyökér alatt. Ha létezik ilyen fájl, az Apache beolvassa a lemezről, és a tartalmát HTTP-válaszként visszaadja. Ha a fájl CGI-program, akkor végrehajtja a programot, és a kimenetet adja vissza HTTP-válaszként; és végül, ha a fájl nem létezik, az Apache a felhasználó böngészőjének hibaüzenetet ad.
A Zope az URL-eket másképpen értelmezi; a */foo/bar* URL például azt jelenti Zope alatt, hogy a foo-objektumban található barobjektum megjelenítő eljárását meg kell hívni. Ha nem létezik foo- vagy barobjektum, a Zope hibaüzenetet ad vissza, amelyben arra figyelmeztet bennünket, hogy a ZODB-ben nem talált ilyen objektumot.
Természetesen, ha minduntalan megjelenítő eljárások írásával és objektumok létrehozásával kellene vesződnünk, a Zope sem volna több egy különleges programozói játékszernél. Szerencsénkre a Zope lehetővé teszi, hogy számunkra a ZODB hierarchikus fájlrendszernek látszódjon, ahol HTML-fájljainkat és grafikáinkat elhelyezhetjük.
Például egy egyszerű HTML-fájlt akár a böngészőnk segítségével is felvihetünk. Lépjünk a Zope saját könyvtárába (ide bármikor visszatérhetünk, ha a bal oldali keret bal felső részére kattintunk), az új állomány létrehozásához válasszuk a DTML Document lehetőséget a jobb felső sarokban található ***Select type to add...*** listából (amint azt hamarosan látni fogjuk, a ***DTML/Document Template Markup Language*** a Zope kibővített HTML-nyelve; a Zope szinte mindig DTML-re hivatkozik HTML-dokumentumok helyett).

DTML-értékek beállítása

Böngészőnkön egy rövid, három részből álló HTML-űrlapot láthatunk:

- Az id szövegmező – a Zope alatt minden elemhez egy ID (azonosító) tartozik, ezeknek az ID-knek az adott mappában mindig egyedinek kell lenniük. Az ID-k itt ugyanazt a szerepet töltik be, mint a lemezen a fájlnevek, azaz arra szolgálnak, hogy az URL objektumait azonosítsák.
  A */foo/bar* ***URL***-ben, a foo a könyvtárobjektum ID-je, a bar pedig ebben a könyvtárban található objektum azonosító ID-je. A Zope-dokumentumoknak hagyományosan nincsenek kiterjesztései (mint például a ***.html*** vagy a ***.gif***); mivel a rendszer pontosan tudja, hogy az egyes azonosítókhoz milyen típusú objektum tartozik, egy ilyen kiterjesztés használata felesleges lenne.
- A cím szövegmező – az URL-ekben az objektum ID-je jelenik meg, de a Zope belső kezelőfelületein már az objektum címével dolgozunk (az objektum címének semmi köze a HTML <title> (cím) taghoz, amit továbbra is nekünk kell elkészítenünk).
- A fájlelem lehetővé teszi, hogy a helyi gépről a HTTP-feltöltés használatával állományt töltsünk fel. Ez azt jelenti, hogy a DTML-állományt a helyi gépen kell létrehoznunk, majd amikor ellenőrizni akarjuk, fel kell töltenünk a Zope-ra.

Ebben a példában most a ***testdoc*** ID-t írjuk be, és a ***Test document*** címet adjuk meg. Bár ezt a dokumentumot is hozzáadhatnánk már a ZODB-hez, így még üresen maradna. Ezért az ***Add and edit*** gombra kattintunk, így szövegmezőhöz jutunk, ahol rögtön alapértelmezett tartalmat is találunk:

```
<dtml-var standard_html_header>
<h2><dtml-var title_or_id></h2>
<p>
This is the <dtml-var id> Document.
</p>
<dtml-var standard_html_footer>
```

Ha akarjuk, ezt a szöveget a szövegablakban át is szerkeszthetjük. Mikor végeztünk, kattintsunk a ***Save changes*** (változások mentése) gombra a képernyő alján. Ez a gomb visszavisz bennünket a szerkesztőablakra, de beilleszt egy emlékeztetőt, ami megmutatja, hogy a dokumentumot mikor módosították utoljára.

A DTML használata

DTML-dokumentumunk igen hasonló a HTML-hez, eltekintve attól, hogy néhány `<dtml-var>` kezdetű tagot tartalmaz.
A DTML tulajdonképpen programozási nyelv, ami sok mindent lehetővé tesz, de talán egyszerűbb (és jobb) úgy gondolnunk rá, mint egy igen erős kiszolgálóoldali beillesztő (include) szerkezetre. Ahogy a példadokumentum is bemutatja, a `<dtml-var>` alkalmazásával dokumentumunkba dinamikus értékeket illeszthetünk. Így a `<dtml-var standard_html_header>` a `standard_html_header`

ID-jű objektum tartalmát illeszti be, míg a `<dtml-var id>` a pillanatnyi dokumentum azonosítóját helyezi a tartalomba. Akár ki is találhatnánk: a `<dtml-var title_or_id>` a címet vagy az azonosítót jeleníti meg aszerint, hogy címet megadtunk-e vagy sem.
Mikor a Zope egy `<dtml-var standard_html_header>`-kifejezést talál, a `standard_html_header` azonosítójú objektumot először a pillanatnyi könyvtárban keresi. Ha létezik ilyen objektum, a `<dtml-var>` tagot ennek látható tartalmával helyettesíti. Ha nem létezik, a Zope a keresést a felsőbb könyvtárakban is megismétli. Így az egész szerkezetet végigjárja, míg végül el nem ér a saját könyvtárig.
Ez azt jelenti, hogy az önműködően beillesztett fejlécek és láblécek nemcsak a használt `<dtml-var>` tagoktól függenek, hanem DTML-dokumentumunk helyétől is. Ezt az igen lényeges és hasznos jelenséget, miszerint az objektum objektumhierarchiában elfoglalt helye befolyásolja a kimenetet, a Zope világában szerzeményezésnek (acquisition) nevezik. A szerzeményezés révén minden egyes könyvtárhoz más és más fej- és láblécet rendelhetünk. Ha dokumentumunkat az egyik könyvtárból a másikba helyezzük, megváltoztatjuk a Zope által használt keresési utat, amelyet a fejlécek és láblécek, illetve más objektumok azonosításához használ. Az objektum viselkedése tehát a Zope-ban nemcsak a meghatározástól függ, hanem az objektumhierarchiában elfoglalt helyétől is. Ezért a szerzeményezést gyakran írják le a biológiából ismerős „nature vs. nurture” (öröklött vagy tanult) vita megfelelőjeként: az objektum meghatározását tekinthetjük „öröklött tulajdonságnak” (nature), míg az objektumhierarchiában elfoglalt helyét „tanult képességnek” (nurture).
A dokumentum pillanatnyi tartalmának megtekintéséhez a keret tetején kattintsunk a ***View*** fülre. A dokumentumtartalmat pontosan úgy fogjuk látni, ahogy a felhasználó látná. Ha a dokumentumot ismét szerkeszteni szeretnénk, a legegyszerűbb, ha a bal felső sarokban található ***Root folder*** hivatkozásra kattintunk, kiválasztjuk a ***testdoc*** objektumot, majd a szövegmezőt tetszés szerint átszerkesztjük.
Ha a DTML szerkesztése közben hibát vétünk, nem árt tudni, hogy a Zope végtelen visszalépési lehetőséggel rendelkezik. Kattintsunk az ***Undo*** (visszalépés) fülre a DTML-szerkesztőablak jobb felső részében, és válasszuk ki azt a változatot, amelyikhez vissza szeretnénk térni. A végtelen visszalépés az egyik kedvenc szolgáltatásom a Zope-ban, nemcsak azért, mert lehetőséget ad hibáim kijavítására, hanem mert a nem programozók számára is könnyen elérhető, és bármilyen típusú objektummal működik.

Más DTML-eljárások

Számos DTML-tag létezik, valamennyit a dtml-kezdet különbözteti meg. A legtöbb DTML-tag egy vagy több értéket is elfogad, ahol minden érték az alapértelmezett működést változtatja meg egy kicsit. Példadokumentumunkat például átalakíthatjuk, hogy a következő egyszerű tagot használja:

```
<dtml-var expr>:
<p><dtml-var expr="5+10"></p>
```

A fenti kódban az `expr` egyszerű Python-kifejezés. A Zope kiértékeli a kifejezést, és az eredményt a dhtml-tag tag helyére illeszti. Egyszerű összeadás helyett valami érdekesebbet is kipróbálhatunk, például a `string` modulból használhatjuk a nagybetűsítő (capitalize) eljárást (ami önműködően importálódik a Zope-ba), ha az értékén belül nevezzük meg:

```
<p>
<dtml-var expr="_.string.capitalize(·abc·)">
</p>
```

Figyeljük meg, hogy a `string.capitalize()` függvényt nem hívhatjuk meg közvetlenül, csak a `_.string.capitalize()` -t.
Ugyan a DTML lehetővé teszi, hogy a `string`-modult ilyen módon használjuk, de soha nem lesz rá szükségünk, hogy közvetlenül hívjuk meg, olyan sok hasznos karakterkezelő függvényt találunk a DTML-be építve.
A DTML tervezőknek és nem programozó felhasználóknak készült azért, hogy saját dinamikus tartalmukat a kiszolgálóoldali ***include*** állományok gyenge írásmódjával és a kevés leírással való bajlódás nélkül készíthessék el. Természetesen egy nem programozónak a DTML-t nem olyan könnyű megtanulni, mint egyesek gondolnák, de minden bizonnyal sokkal könnyebb, mint egy teljes értékű programnyelvre tanítani meg őket. Ráadásul a Zope a DTML-dokumentum mentésekor bizonyos fokú hibaellenőrzést is végez, ami segít elkerülni néhány, a futásidejű nyelveknél gyakori hibát.

Összefoglalás

Ebben a hónapban villámlátogatást tettünk a Zope világának berkeiben, megismerhettük a Weben keresztüli szerkesztést, a karbantartó felületet, a szerzeményezést, és egy egyszerű DTML-dokumentumot is szemügyre vehettünk. A következő hónapban a Zope termékeit ismerjük meg – megnézzük, miképpen kell őket letölteni és telepíteni, illetve hogyan írhatjuk meg a saját változatainkat.

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

A nyílt forráskódú Zope-alkalmazást a Zope Corporation, régebbi nevén a Digital Creations kezeli és írja. Bővebb tájékoztatást a Zope Corporation honlapján találunk a ➲ http://www.zope.com címen.

A Zope hivatalos fejlesztői közösségének oldala a ➲ http://www.zope.org címen érhető el. Egyúttal ez az a honlap, ahonnan a legújabb változatokat letölthetjük, megvitathatjuk felmerült kérdéseinket, letölthetjük a Zope-termékeket, illetve tagjává válhatunk a Zope fejlesztőközösségnek.

Beehive Book of Zope c. könyve, melyet a No Starch Press és a Linux Journal Press közösen adtak ki, kitűnő bemutató írás Zope-programozóknak és -felhasználóknak egyaránt. A teljes körű ismertetés a következő havi Linuxvilágban jelenik meg.

A LinuxBIOS

Eric megmutatja nekünk, hogy a LinuxBIOS elfogadása és teljesítménye mennyire felkeltette a beágyazott és a fürtözött rendszerek fejlesztőinek figyelmét.

Több mint egy éve dolgozom a Linux NetworX-nél LinuxBIOS-fejlesztőként, és ez idő alatt rengeteg mindent meg kellett tanulnom. Most már bizton kijelenthetem, hogy a Linux rendszermagja igazi profi munka, a C nyelv pedig igazán magas szintű nyelv.

Mi is a LinuxBIOS?

Mikor a mikroprocesszor elindul, szépen sorban elkezdi végrehajtani a ROM-ban található utasításokat. Ezek az utasítások felelősek az alkatrészek felkészítéséért, elsősorban a RAM engedélyezéséért, majd pedig az operációs rendszer betöltéséért. Ennek a rendszernek a felépítése és kezelőfelülete gépről-gépre változó, de alapjában véve mégis mindenhol ugyanaz.

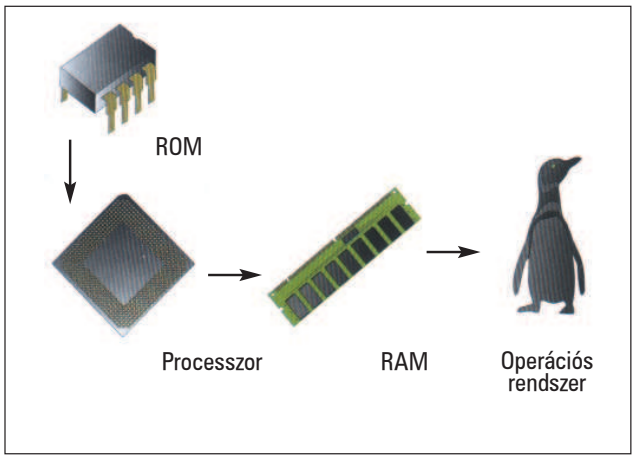
Az Alpha-alapú rendszer esetében a mikroprocesszor beolvassa az egész soros ROM-ot – azaz az SROM-ot – a gyorstárba, majd elkezdi végrehajtani az utasításokat. Az itt található kód feladata, hogy előkészítse a mikroprocesszort, a RAM-ot, és hogy egy flash EEPROM-ból betöltse az SRM-et. Ezt követően a mikroprocesszor végrehajtja az úgynevezett ***palcode***-ot (alapvetően a valódi Alpha-rendszermagot), előkészít még néhány alkatrészt, végül pedig elindítja az operációs rendszert. Mivel a gépi szinten futó program (firmware) két részre van osztva, az SRM frissíthető vagy akár teljesen lecserélhető. Az eredeti Alpha-elképzelés szerint a gépi kódnak operációs rendszerenként különböznie kell (az SRM-szintjén).

Az x86-os processzor 16-bites módban kezdi meg működését, és kezdetben a 16-bites CS regiszter a 64 K-s címterület végére mutat. A 8086-oson ez a cím 0xF000:0xFFF0 == 0xFFFF0, pont 1 MB alatt. 80386 vagy annál újabb mikroprocesszor esetében a cím 0xFFFFFFF0, pont 4 GB alatt. A 286-osnál és későbbi Intel-modelleknél a CS tartalma nem tölthető tetszőlegesen bármikor újra, emiatt a rendszer a ROM-ot a 0xFFFF0000 és 0xF0000 címeken is elérhetővé teszi.

Az Alphával ellentétben az x86-os rendszerek a ROM-ban található utasításokhoz egyenként férnek hozzá. Mivel a ROM valójában ISA-eszköz, kezdetben, amikor még nincs semmilyen gyorstár engedélyezve, a benne található kód végrehajtása nagyon lassú, illetve az is ennek a következménye, hogy a lapkakészletnek szinte teljesen fel kell állnia ahhoz, hogy a ROM-hoz hozzáférjünk, mivel az ISA eléréséhez elég hosszú utat kell végigjárnunk a processzortól az északi hídig, a PCI-síntől a déli hídig, és csak ezután érünk az ISA-hoz. Ha ezzel tisztában vagyunk és van egy jó alaplapunk is, nagymértékben megkönnyíti a helyzetünket, ha a rendszer indulásánál működésbe lépő eszközök valamelyikén végzünk hibakeresést.

Az alap PC BIOS feladata közé tartozik: az alkatrészek felkészítése az indulásra, az operációs rendszer betöltése, és az operációs rendszer futtatása közben bizonyos szolgáltatások biztosítása (többnyire a legszükségesebb eszközmeghajtók formájában).

A gépi szinten futó kódot a SPARC és a PowerPC esetében is meghatározták, melyet ***OpenBoot***, ***Open Firmware*** vagy az egykori ***IEEE1275***-ként is ismerünk. A szabványosított ***Forth firmware*** majdnem ugyanott található, ahol az Alpha esetében az SRM. Az ***Open Firmware*** esetében jó néhány egyéni vonással találkozhatunk: több processzoron és kiépítésen működőképes; ***Forth***-alapú bájtkódot használ, így a futtatható programok nem processzorfüggőek; ezenkívül a rendszer indításakor végrehajtódó utasítások legjavát a Forth-alapú bájtkódot átalakítva kapja.



1. ábra A rendszerindítás lépései

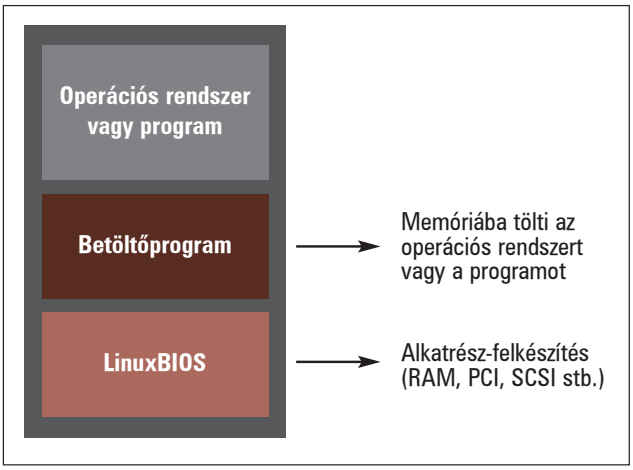
Az Itanium/IA64 úgynevezett EFI gépi szinten futó kódot használ, mely inkább felületfüggő, mint az ***Open Firmware***, mivel meghajtói vagy IA32- vagy IA64-kódra épülnek. Felépítését nézve is jóval bonyolultabb annál, az EFI ugyanis IP-vermet és fájlrendszermeghajtókat is tartalmaz. Hasonlóan az ***Open Firmware***-hez a kezdeti alkatrész-készenléti állapot itt sincs meghatározva.

A Linux-rendszermagnak a gépi kóddal szemben támasztott követelményei csekélyek, mivel az eszközöket közvetlenül hajtja meg, a BIOS közreműködése nélkül. Mivel a rendszermag nem használja a BIOS-t, az alkatrészek előkészítése a BIOS részéről egyszerűen felesleges. Ez a megközelítés nem csak a Linuxra jellemző, ugyanis nem ismerek olyan korszerű operációs rendszert, amely ne ezt az irányvonalat követné.

A jelenlegi operációs rendszereknek mindössze egy alapvető rendszerelőkészítő szolgáltatásra van szükségük. A különleges eszközmeghajtók és rendszerjellemzők, amelyeket az ***EFI***, az ***Open Firmware*** vagy akár a PCBIOS biztosít, egyáltalán nem szükségesek, kivéve azokat a részeket, amelyek az operációs rendszer betöltéséhez szükségesek. Mivel ezekre a kódokra nincs szükség, a LinuxBIOS-ban nem is találhatók meg.

A LinuxBIOS kódja elégséges ahhoz, hogy egy Elfben kódolt programot betöltsön a flash ROM-ból. Ilyen program lehet egy operációs rendszer rendszermagja mint a Linux-rendszermag, de a legtöbb esetben ez mégis valamilyen alkatrészvizsgáló program vagy rendszerbetöltő eljárás (például Memtest86, Etherboot vagy a RedBoot).

Ha a LinuxBIOS mellé rendszerbetöltőt rendelünk, alkalmas lesz az operációs rendszer betöltésére. A LinuxBIOS eredeti ötlete az volt, hogy a Linux-rendszermagot a ROM-ból töltsük be, és a rendszerbetöltő eljárást arra építsük. Az ***nbc*** nevű rendszerbetöltő program ezt az ötletet valósítja meg: a hálózaton keresztül betölti a Linux-rendszermagot, vagy valamilyen önműködő programot, majd a rendszert a ***kexec*** rendszermagfolt (kernelpatch) segítségével elindítja. Ez a megoldás nagyszerűen működik, ha 512 KB vagy annál nagyobb ROM áll a rendelkezésünkre, viszont a manapság kapható alaplapok többsége sajnos csupán 256 KB ROM-mal rendelkezik. Az x86-os felület esetén szinte teljesen lehetetlen a 360 KB-nál kisebb rendszermag összeállítása.



2. ábra A Linux BIOS betöltése

Ennek az akadálynak a megkerülésére – mely a nem elégséges ROM-méretből fakad – különböző elképzeléseket dolgoztak ki. Közülük az egyik a Tiara, mely teljes gépi kód- és rendszerbetöltő a SiS630-as lapkakészlethez; ilyen még az Etherboot, amelyet átdolgoztak, hogy együttműködjön a LinuxBIOS-szal, valamint a RedBoot, ez azonban még nem teljesen működőképes. Ezenkívül még különböző foltokat készítettek a LinuxBIOS-hoz is, amelyek segítségével ez a nehézség megkerülhető. Az Alpha gépi kódjának olyan programra van szüksége, amely ismeri az alaplapot, amelyen fut, és ez gondokat okozhat. Emiatt az újabb alaplapok támogatása a felmerülő frissítések mennyisége miatt esetenként nagyon nehéz lehet. A LinuxBIOS fejlesztése során megteszünk minden tőlünk telhetőt, hogy ezt a hibát elkerüljük.

A hagyományos x86-os megközelítéssel kezdünk: felkészítjük a Super-IO lapkákat a munkára, beállítjuk a kezdeti értékeket (például a soros kapukat az alapértelmezett címeikre, az IRQ-kat stb.), majd elérhetővé tesszük az IRQ útválasztási táblázatokat, illetve az SMP számára az úgy nevezett ***mp***-táblákat. Hosszú távon egy táblázatban tároljuk az azonnal használható, azonosított lehetőségeket. Ez a programlista tárolja, milyen alkatrészek állnak rendelkezésünkre, illetve beállítja, hogy ezek közül melyeket fogja a számítógép használni, vagy legalábbis megadja, hogy az egyes eszközök milyen erőforrásokat használnak.

A megoldás, melyen dolgozom, egy táblázat karbantartását teszi szükségessé, mely az eszközök egymáshoz és az alaplaphoz való kapcsolódásáról tárol adatokat. Ez a táblázat tartalmazza az összes olyan eszközt, melyet egyetlen azonnal használható (Plug-and-Play) felsorolás sem foglal magában, és elegendő adatot tartalmaz ahhoz, hogy az IRQ-útválasztás megfelelően kezelhetővé váljon. Ráadásul az ötlet úgy tűnik, megfelelően illeszkedik a 2.5-ös rendszermaghoz tervezett eszközfa-struktúrához. Az ACPI látszólag választható megoldást kínál, de meglehetősen PC-centrikusnak tűnik, emellett a feldolgozott byte-kód szükségtelen, sőt veszélyes is lehet.

Kivitelezhető a LinuxBIOS?

A rendszerindító ROM-hardver felépítése az első IBM PC óta rengeteget fejlődött, így manapság szinte minden számítógép helyben frissíthető BIOS-szal rendelkezik, mely hibás frissítés esetén az eredeti állapotába állítható vissza. Ennek következtében általános eljárásnak mondható, hogy az alaplapon a flash ROM-ot egy foglalatban helyezik el. A flashlapka a program számára lehetővé teszi, hogy frissítse a tartalmát, mindemellett a foglalat a lapka cseréjét is biztosítja, ha a frissítés valamilyen okból nem sikerül. Ilyen típusú kiépítéssel már lehetséges egyéni rendszerindító gépszintű programok előállítása. A frissítéshez nincs szükség speciális eszközre, és ha a fejlesztés során elromlik valami, visszaállítható az eredeti állapot.

A jelenlegi PC-kiépítések hátránya, hogy a normál 256 KB-s rendszerindító ROM-ok túlságosan kicsik. Ez a hely elég egy gépi kód számára, de nem elegendően nagy a Linux-rendszermagnak.

A Linux-rendszermag ugyanúgy képes LinuxBIOS-ból futni, mint az általános PCBIOS-ból, ha az átültetést megfelelően elvégezzük el. A LinuxBIOS-t a mai napig három alaplapra is sikeresen átültettem. A legutóbbi alaplapon már nem lehet különbséget tenni, hogy a rendszert a PCBIOS-ból vagy a LinuxBIOS-ból indítottuk-e el. Tehát a meglévő technikai akadályok ellenére úgy tűnik, hogy a LinuxBIOS-t sikerül, illetve már sikerült is átültetnünk újabb rendszerekre.

A fentieken túl az is kulcskérdés, hogy a fejlesztőnek hozzáférése legyen a megfelelő szintű dokumentációhoz. A számítógépgyártók eddig csak nehezen tűntek rábeszélhetőnek arra, hogy támogassák a LinuxBIOS-t, vagy akár arra, hogy megfelelő minőségű dokumentációt biztosítsanak valaki számára, aki végül lekódolná azt. A korlátozott támogatás vagy akár annak teljes hiánya nem minősül új dolognak a szabad programot használók számára, ezt a nehézséget eddig is megoldottuk valahogy, és most sem adhatjuk fel miatta a reményt. Nem szabad elfelejteni, hogy efféle erőfeszítések nélkül nem születnének olyan új gépek, melyeken szabad programot futtathatunk.

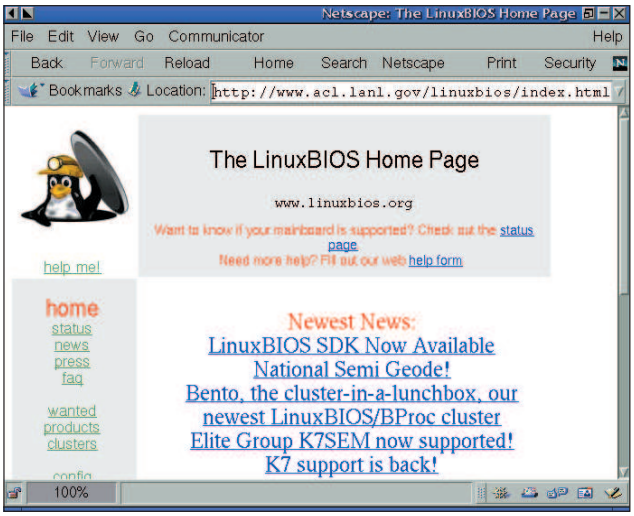
Milyen alkalmazások léteznek a LinuxBIOS-hoz?

Jelenleg két érdekeltségi kör dolgozik a LinuxBIOS-on: az egyik beágyazott rendszereket készít, míg a másik nagyteljesítményű számítógépfürtöket. Ezen alkalmazások számára az ősi x86 gépszintű program elégséges.

A LinuxBIOS nagy karrier előtt áll a beágyazott alkalmazások terén. Mivel a GPL licenc alatt fejlesztik, a LinuxBIOS teljességgel jogdíjmentes. A LinuxBIOS-nak mindössze 64KB elegendő, nem pazarolja a ROM-ot szükségtelen feladatokra. Mivel a LinuxBIOS modern felépítésű, rendkívül gyorsan indul, még kódoptimalizálás nélkül is.

2001 augusztusában a General Software-nek egy beágyazott rendszeren újraindítás után 0,8 másodperc alatt sikerült eljutnia a LILO-kiírásig. Ez az adott feladathoz képest elfogadhatónak mondható, de a LinuxBIOS esetén egy ilyen jó eredmény teljesen rutinszerű. Ha hidegindítás után a rendszermagot hálózatról töltjük be, a rendszerindulás egy kétprocesszoros kiszolgálófarm esetén mindössze két másodpercet vesz igénybe – a LinuxBIOS hatékonnyá tétele nélkül.

A LinuxBIOS eredményei a SiS-t olyannyira meggyőzték, hogy egy fejlesztőjüket megbízták a LinuxBIOS saját alaplapjaikra való átültetésével, megcélozva a beágyazott rendszereket és jól támogatott felületet hozva létre.
Számítógépfürtök esetén – amelyekkel a Linux NetworX foglalkozik – a LinuxBIOS úgyszintén nagy jövő előtt áll. A soros kapu az alapértelmezett rendszerkonzol, így nincs szükség videóalkatrészre. A soros kapcsolatok egy központi terminálkiszolgálón könnyedén átirányíthatók, hogy távolról hozzáférjünk a gépek rendszerkonzoljaihoz. A soros konzol kezdeti szakasza is előnyökkel jár, például a LinuxBIOS ezen keresztül



képes a különféle alkatrész- vagy egyéb hibák jelentésére. Egy egyszerű BIOS, még ha rendelkezik is soroskonzol-kiegészítéssel, a soros kaput túl későn nyitja meg ahhoz, hogy néhány hibát időben észlelhessen, és rendszerint egy üres CMOS végzetes hatással van a rendszerindításra nézve.
A legtöbb rendszeren a LinuxBIOS lehetővé teszi a hálózaton keresztüli rendszerindítást, ezáltal a DHCP-kiszolgálón keresztül az indítási beállításokon akár egy egyszerű módosítással változtathatunk. Mivel a kód szabad forrású, ha a hálózati tulajdonságok nem megfelelőek, könnyedén módosíthatunk rajtuk. A gyors újraindítás a LinuxBIOS esetén azt jelenti, hogyha valamilyen hibát keresel, és ehhez újra kell indítanod az egyik csomópontot, a gépújraindítási folyamat nem a rendszergazda idejét rabolja.
A LinuxBIOS nyíltsága és Linux-központúsága azt eredményezi, hogy beállításai Linux alól módosíthatók. Így minden, ami a felhasználó parancssorából módosítható, a hálózaton keresztül is megváltoztatható. Azonos gépekből összeállított telepek esetén ezzel rengeteg idő nyerhető, mivel a gépszintű programokon a változtatások központilag elvégezhetők, és nem kell őket külön-külön minden csomópontnál végrehajtani. Nagyszámú gép esetén a számítógéphibák jóval valószínűbbek, mint egyetlen gép esetén. A LinuxBIOS kis gépigényeivel – nem szükséges hajlékonylemez vagy CD-ROM-meghajtó, és merevlemezre sincs feltétlenül szükség – egy kevésbé drága, ugyanakkor jóval megbízhatóbb rendszert kapunk. Minél kevesebb gépet használunk, annál kisebb a kockázata egy esetleges számítógép-meghibásodásnak.
Fürtök esetén a csatlakoztatott számítógépek a LinuxBIOS segítségével beállíthatók, hogy viselkedjenek úgy, mintha egyetlen számítógép lennének, ahelyett, hogy úgy nézzenek ki, mint összekötött számítógép-csomópontok halmaza.

Milyen alkatrészeket támogat a LinuxBIOS?

A LinuxBIOS jelenlegi állapotában 13 különböző alaplapot támogat. A LinuxBIOS ugyanúgy fut az egyes x86-kiépítéseken, mint Alphán. Az AMD Athlon, az AMD Duron, a Pentium II és Pentium III, az Alpha 211264 CPU-k, az ALI m1631, a Digital Tsunami, az AMD 760, az AMD 760MP, az Intel 440BX, az Intel 440GX, a VIA VT8601, a SiS540, a SiS550, a SiS630 és a SiS730 már mind-mind képesek futtatni a LinuxBIOS-t. Most csak azokat soroltuk fel, melyeket a mostani LinuxBIOS már tartalmazza és működőképesek, de nem említettem a még fejlesztés alatt álló, illetve a LinuxBIOS-ban még nem szereplő átírásokat. Tehát annak ellenére, hogy az alkatrész-támogatás korlátozott, a lista mégis folyamatosan nő. A LinuxBIOS jelen pillanatban nincs egyetlenegy lapkakészlethez, gyártóhoz, vagy processzorkiépítéshez sem kötve.
A számítógép-támogatás minősége változó. A lapkakészlet oldaláról nézve, a SiS-lapkakészletek támogatása a legfejlettebb. Az Intelnek és az AMD-nek egyaránt megvan a maga alapvető politikája lapkakészleteik leírásával kapcsolatban, így mindkét gyártó támogatottsága elég jó. A Via leírásai nyilvánosan sajnos nem hozzáférhetők, így az ő termékeik támogatása meglehetősen nehéz.
A processzoroldalon a Compaq a lényeges részleteket nyilvánossá tette, így az Alphák támogatása kivitelezhető. Ezzel szemben mivel az Alpha-processzorok elég drágák, ráadásul jelen pillanatban a jövőjük is elég kétséges, a támogatásuk nem elsődleges szempont.
A Pentium II és Pentium III processzorok nagyon jó leírással rendelkeznek, kivéve ami a másodszintű (L2) gyorstárukat illeti, de a LinuxBIOS már azt is támogatja. Az AMD Athlon és Duron nem olyan jól támogatott, mivel az AMD nem tett nyilvánossá minden olyan részletet, amelyre a fejlesztésekhez szükség volna.
Az alaplapgyártó oldaláról a támogatás nem feltétlenül szükséges, mivel a legtöbb esetben az alaplapon található eszközökről az első pillanatban megállapítható, hogy mivel van dolgunk.
Az alaplapgyártók alaplapjaikhoz csak egyetlen gépszintű program támogatásában érdekeltek. Mivel a LinuxBIOS egyéb operációs rendszerekhez nem tartalmaz támogatást, különösen a Windowshoz nem, jelenlegi formájában a LinuxBIOS alkalmazásában nem különösebben érdekeltek.

Összegzés

A LinuxBIOS segítségével betekintést nyerünk abba, hogyan működik a gépszintű program, hogyan épül fel, hogyan írták és milyen a felhasználói szerződése. Ahogyan a számítógépek egyre egybevontak, a LinuxBIOS fokozatosan rugalmasabbá válik, és nagyobb lehetőséget biztosít a kód-újrafelhasználásra. Ennek köszönhetően remélhetőleg egyre elterjedtebbé válik majd, és ezáltal még nagyszerűbb LinuxBIOS-t kapunk.

Eric Biederman
(ebiederman@linuxnetworx.com a Linux NetworX-nél dolgozik programfejlesztőként. Elsődleges feladata a LinuxBIOS, valamint segédkezni a számítógéptelepeket karbantartó programok tervezésénél. Ha Eric éppen nem a LinuxBIOS-szal van elfoglalva, szívesen olvas sci-fi-t, vagy játszik a DOSEMU-val, emellett szabadidejét szívesen tölti a Salt Lake City mellett található Wasatch-hegyekben.

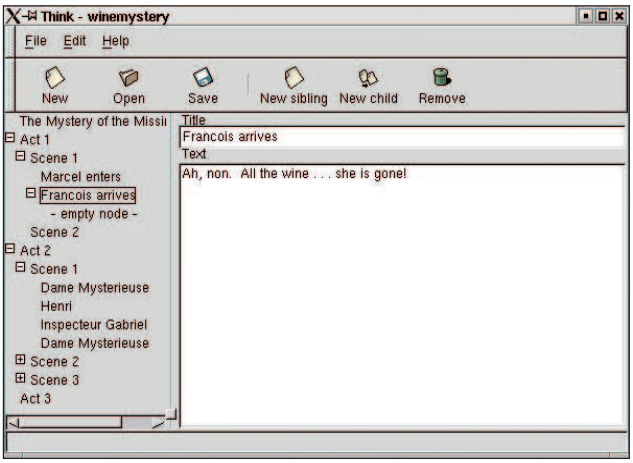
Parancsállományok a Weben és egyéb helyeken

Marcel a legközelebbi forgatókönyved vázlatkészítéséhez és összerendezéséhez mutat néhány egyszerű módszert.

François, mon ami, mit csinálsz? Látom, szabad utat engedtél alkotói vénádnak. Sárkányok, űrhajók, világkörüli Linux-rendszerek… lenyűgöző történet. Látom, az étterem is a díszlet részét fogja képezni – csodálatos! De vajon mi okból dolgoztatod ennyire a szürkeállományodat, François? Vagy úgy! A Linuxvilág 2002. áprilisi számának középpontjában a webes parancsállományok (web scripting) írása áll, és te épp forgatókönyvet írsz (script) egy bemutatandó internetes műsorhoz (a script szó parancsállományt és forgatókönyvet is jelent az angolban – a ford.).
A Világhálóra írsz forgatókönyveket? És miért korlátozod magad a Webre, miért nem írsz rádiós, televíziós vagy színpadi bemutatóra is? A megfelelő eszközzel és a Linuxszal bármire készen állsz.
Siess a pincébe, François, megérkeztek a vendégeink! Hozd fel az ausztráliai Margaret River Chardonnayt – remek bor: illatos és gondolatébresztő. Vite!
Amíg François felhozza a bort, elmagyarázom, mi történt, kedves barátaim. Hűséges pincérünk sajnos félreértette e havi vezérfonalunkat. Éppen egy forgatókönyv írásán fáradozik, remélve, hogy a világhálós bemutatója sikert arat majd. Ahelyett, hogy kiábrándítanánk, a mai menüt a téma köré szervezzük, vagyis történeteket fogunk életre kelteni.
Egy jelenet megtervezése és a tökéletes színpadi beállítás megteremtése némi fortélyt igényel ahhoz, hogy gondolatainkat megfelelően tudjuk elrendezni és áttekinteni. Pontosan ez járhatott ***Peter Teichman*** fejében is a Think nevű Gtk/Gnome-alkalmazás megírásakor, amellyel rangsor szerinti dokumentumvázlatot hozhatunk létre, majd az eredményt XML-dokumentumként menthetjük. Ha gyors pillantást akarsz vetni a Thinkre, látogass el a
➲ http://primates.ximian.com/~peter/think címre és töltsd le a legfrissebb forrást. A kód kibontása után a hagyományos `./configure` lépéshármast követhetjük:

```
tar -xzvf think-0.2.1.tar.gz
cd think-0.2.1
./configure
make
make install
```

A Think futtatásához a `think &` parancsot kell begépelned. Az *1. kép* a programot működés közben mutatja.
E kis gondolatrendszerező alkalmazása során csomópontokat és alcsomópontokat kell létrehoznunk, olyan nevet adva nekik, ami kicsit többet mond annál, mint az üres csomópont (empty node). Minden csomóponthoz szöveget rendelhetünk. Például a párbeszéd szövege a második jelenetet írja le, ami az 1. felvonás egyik alcsomópontja. E csomópontok mindegyike a „fogd és dobd” módszerrel helyezhető át a bal oldali lista tetszőleges helyére. A dokumentum szerkezetének átláthatóságához bármelyik főcsomópont egyetlen sorrá csukható össze.



1. kép Gnome Think

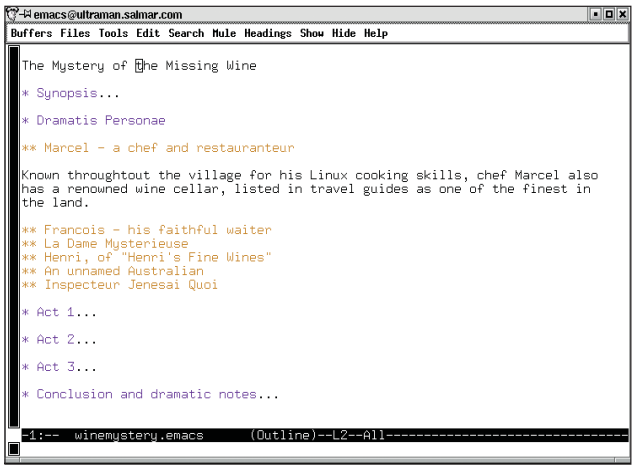
2. kép Belépés az Emacs Outline üzemmódjába

Természetesen nem kell olyasmit letöltened, amit nem akarsz. Jó eséllyel a rendszered már rendelkezik a legendás Emacs szerkesztővel. Grafikus felület futtatására sincs szükséged, hiszen az Emacs szöveges módban is elboldogul. Miért is ne próbálnánk ki a vázlatüzemmódot, amikor olyan egyszerűen használható?
Indítsd el az Emacsot az `emacs ÆllomÆnynØv` paranccsal. Ha az állomány még nem létezik, az Emacs létrehozza. Ezzel az Emacs szerkesztőmódjába jutunk. Ahhoz, hogy a vázlatszolgáltatást kipróbáld, az ESC-X-et, majd az ***outline-mode***-ot kell választanod (lásd a *2. képet*).
Most már csak el kell kezdenünk a gépelést. A szint (a Think kifejezését használva: csomópont) jelzéséhez csupán egy csillagot (*) gépeljünk be, alszint létrehozásához kettőt (**) és így tovább. Ezek alá a címsorok alá a többi szerkesztőprogramhoz hasonlóan bármit beírhatsz. Egy szint, alszint vagy faszerkezet elrejtéséhez az Emacs a CTRL-C szekvenciákat alkalmazza. Az első táblázatban láthatunk is néhányat. Természetesen, ha X grafikus módban futtatod az Emacsot, az egérrel a menüsoron egyszerűen csak a ***Show*** (mutat) és ***Hide*** (elrejt) menüpontokra kell kattintani. Amikor elrejtünk egy szintet, csak az első sora vagy a cím jelenik meg, amelyet három pont (…) követ, az ezt követő adat rejtve marad.
A *3. képen* éppen „Mystery of the Missing Wine (A lábakélt bor rejtélye)” című művemen dolgozom az Emacs vázlatmódját használva.
Amikor ***Christopher Tomaras Lansdowne*** vígjátéktervet akart írni, egy kis Perllel és XML-lel kezdett foglalkozni, majd a GScriptWriterben összekapcsolta őket. Ez olyan Gtk-eszköz, amellyel egy történetet írhatunk körül, új szereplőket vehetünk

fel, akiket legördülő listából választhatunk ki, amikor párbeszédeket vagy cselekményt szeretnénk hozzájuk rendelni. Ez az egyszerű kis program a dokumentum HTML-ként való mentésekor is szép munkát végez.
A GScriptWritert a
➲ http://cs.alfred.edu/~lansdoct/linux/gscriptwriter címre ellátogatva szerezheted meg. A forrás birtokában a telepítést így hajthatod végre:

```
tar -xzvf gscriptwriter-0.1.2.tar.gz
cd gscriptwriter-0.1.2
./gscriptwriter.pl
```

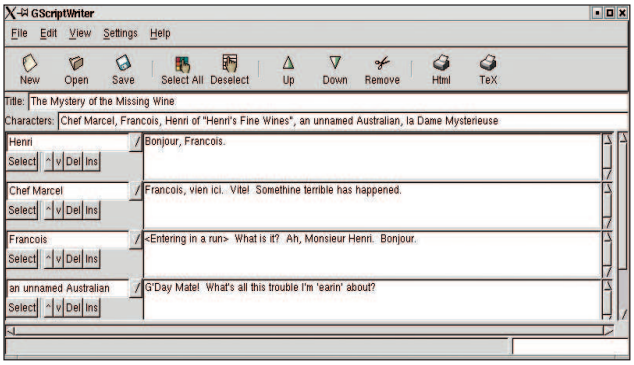
Mint már bizonyára észrevetted, a folyamatból hiányzik a fordítás művelete, egyszerűen futtasd a parancsállományt. Szép Gtk-felületet kapsz, amelyen keresztül a vígjátékok forgatókönyírásának minden feladata elvégezhető. Nézz csak



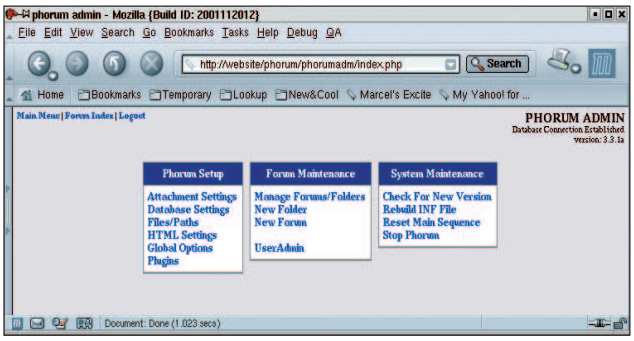
3. kép A titok felfedése Emacs-kiemeléssel

a *4. képre*! François keményen dolgozik egy rejtélyen, ami éppen az éttermünkben játszódik.
A forgatókönyv eseményeinek menete könnyen átszervezhető, csak ki kell választani a megfelelő sorokat, amelyek fel-lemozgathatók. A munkát mentsd, majd válaszd a HTML-exportálás lehetőségét. Eredményül szépen befejezett alkotást kapsz (természetesen csak a rejtély megoldása után). A GScriptWriter még befejezetlen, de a Perl-kódot a saját igényeid szerint könnyen módosítani tudod.
Néhányan úgy gondolják, hogy a történet nem más, mint amilyet a Slashdot.org-on olvashatunk, és soha nem lehet elég belőle. A gond könnyen orvosolható: böngésződbe írd be a ➲ http://bbspot.com/toys/slashtitle/index.html címet. Igen szórakoztató helyen találod magad, amely álvéletlen Slashdot-történeteket hoz létre. Szánj rá egy látogatást, mon ami, és meglátod, miről beszélek. De térjünk vissza tárgyunkhoz!
A „történet" szó számos különböző jelentést rejt. A hirdetőtáblák olyan együttes törekvést valósítanak meg, amelyek használói számára lehetővé teszik, hogy egy központosított feladaton a Világháló használatával közösen dolgozzanak. A történet egyetlen kiírás formájában egy elgondolás kezdetét közvetítve jelenik meg. Néhányan válaszolnak, folytatják az eszmecserét, és nemsokára vagy egy zűrzavar közepén, vagy a következő nagy dráma elütésében találod magad. Megfelelő környezetben egy rugalmas hirdetőtábla csodálatos eszköz lehet.

A következő fogás neve Phorum. A Phorum ingyenes, nyílt forráskódú alkalmazás, amely egyszerű, az Apache-éhoz hasonló felhasználási szerződéssel terjeszthető. Futtatásához Apache-kiszolgáló szükséges, amelyen engedélyezzük a PHP-t, valamint egy támogatott adatbázistípust kíván meg, amely például PostgreSQL, MySQL vagy Sybase lehet. A Phorum



4. kép François a GScripteWritert használja



5 kép A Phorum's Admin beállítóablaka

nagymértékben adatbázisfüggetlen, és telepítéskor futó adatbázis-kiszolgálók után kutatva önműködő keresést hajt végre.
A telepítést nagyon egyszerűnek fogod találni. Kezdd a Phorum honlapjának (➲ http://phorum.org) meglátogatásával, és töltsd le a legfrissebb forrást. A telepítést a forrás webkiszolgáló rendszeredbe való kicsomagolásával indítsd:

```
tar -xzvf phorum-3.3.1a.tar.gz
mv phorum-3.3.1a.tar.gz
➥/usr/local/apache/htdocs/phorum
cd /usr/local/apache/htdocs/phorum
```

Előfordulhat, hogy ennél a pontnál a `secure` parancsállomány futtatásával szeretnéd majd folytatni, amellyel a Phorum biztonságával kapcsolatos elérési utak és engedélyek állíthatók be. Ezt a parancsállományok könyvtárában találod: ***scripts/secure***
Ez különösen abban az esetben fontos, ha látszólagos webkiszolgálót futtatsz. Mivel a Phorum szerkezete közismert, valószínűleg meg akarod majd változtatni az admin könyvtár nevét. Ez az első kérdés, amelyre választ kell adnod. Javaslom, hogy a könyvtár ***.htaccess*** állománnyal való védelmére vonatkozó kérdésre is igennel felelj. Végül a Phorum megkérdezi, hogy az Apache-kiszolgáló milyen felhasználói név alatt fut. Esetemben ez www volt. A telepítés további része működés közben zajlik. Ehhez a böngésződbe a felügyeleti könyvtár

elérési útvonalát kell beírnod. Ha a biztonsági parancsállomány futtatása során ezt megváltoztattad, a megfelelő elérési utat kell használnod:

http://yourwebserver/phorum/admin

A Phorum telepített adatbázisodat megpróbálja önműködően felismerni. Ha több ilyen is létezik (mint nálam is), ezen a

Emacs-vezérlőbillentyűzetek

Mutat		*Elrejt*	
CTRL-C CTRL-A	Mindent mutat	CTRL-C CTRL-L	Levelek elrejtése
CTRL-C CTRL-E	Bejegyzést mutat	CTRL-C CTRL-T	Törzs elrejtése
CTRL-C CTRL-K	Elágazásokat mutat	CTRL-C CTRL-C	Bejegyzés elrejtése
CTRL-C CTRL-TAB	Gyerek mutat	CTRL-C CTRL-D	Alfa elrejtése
CTRL-C CTRL-S	Alfát mutat	CTRL-C CTRL-Q	Alszintek elrejtése
		CTRL-C CTRL-O	Egyebek elrejtése

6. kép Vicces dolgok a „Phorum"-on

ponton választanod kell közülük. Én telepítésem során a PostgreSQL-t választottam. A Submit (elküld) gombra kattintás után az adatbáziskiszolgáló nevét (localhost), az adatbázis nevét, a felhasználói nevet és jelszót kellett megadnom. Mielőtt továbbmennél és kitöltenéd ezeket a mezőket, hadd figyelmeztesselek valamire. Az adatbázis-felhasználónak egy valóban létező felhasználói névnek kell lennie, kivéve, ha az adatbázisod nem igényel nevet és jelszót (ami nem túl biztonságos dolog). A PostgreSQL esetén a felhasználó hozzáadása így történik:

```
createuser felhasznÆl i_nØv
```

A felhasználónak létrehozó és módosító képességekre lesz szüksége, majd egy üres adatbázist kell létrehoznod. A sajátomat (talán kicsit földhözragadtan) phorumnak neveztem el. Nincs időnk az összes adatbázisra kitérni, de ha úgy adódik, hogy PostfreSQL-t futtatsz, a `postmaster`-t mindenképpen `-i` kapcsolóval indítsd, hogy az adatbázishoz való kapcsolódást TCP-n és a Weben keresztül egyaránt lehetővé tedd:

```
/usr/bin/postmaster -D /var/lib/pgsql/data -i
```

Időközben – visszatérve a böngészőn keresztüli telepítéshez – kitöltöm a hátralévő mezőket, a Submit gombra kattintok, jutalmam pedig egy szép üzenet, mely szerint a tábla létrehozása és az adatbázis frissítése megtörtént. Egészségetekre, mes amis! De várjatok csak – ahogy nálunk mondják: van másik! A következő képernyőn a felügyeleti felhasználói név és jelszó megadására van szükségem, végül az utolsó képernyőn a Phorum címének és a felügyelő levélcímének a rögzítésére, legutoljára a Submitra kell kattintanom és kész is vagyunk. Ekkor a rendszer önműködően a felügyeleti bejelentkező képernyőre irányít. Írd be az imént megadott felügyeleti nevet és jelszót, majd itt az idő, hogy kicsinosítsd a saját vitafelületedet.

Megváltoztathatod a színbeállításokat, az oldalanként megjelenő üzenetek számát, megadható, hogy a felhasználók csatolhatnak-e állományokat és ezek mekkorák lehetnek. Az *5. kép* a Phorum felügyeleti képernyőjét mutatja.

Mivel témakörök nélkül a viták meglehetősen haszontalanok, talán néhány percet rá tudsz szánni a Phorum témakörökkel való feltöltésére. A kapcsolódó párbeszédablakok egyszerűek, könnyen használhatóak – a segítségükkel megadható, hogy ki adhat fel üzeneteket, a vita moderált-e, valamint a megjelenítés (szín, elrendezés) tulajdonságai is beállíthatók.

Miután mindezzel elkészültél, célszerű a felhasználók felé is kinyitni a Phorumot. Egyszerűségénél fogva akár egy már létező weblaphoz is kapcsolhatjuk, például keretként betöltve. A Phorumot használó számos oldal ezt a módszert követi. Jó megoldást érhetsz el úgy is, ha a böngészőjüket a honlapodon lévő phorumra irányítod:

http://your_website_address/phorum

Most már a felhasználók végrehajthatják egyedi beállításaikat, olvashatnak, üzeneteket írhatnak, mások leveleire válaszolhatnak. Hogy ténylegesen szükségük lesz-e felhasználói profilra, az a Phorum felügyeleti képernyőjén keresztül létrehozott vitaablak előzetes beállításain múlik. A *6. kép* a Phorumot eszmecsere közben mutatja.

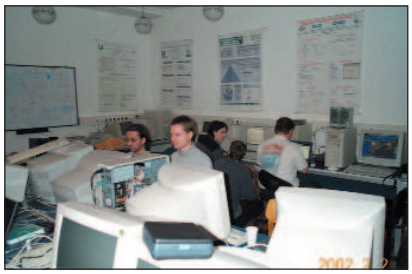
Kedves barátaim, ismét elérkezett a záróra ideje Chez Marcelnél. Még egyszer köszönöm, hogy ellátogattatok, a viszontlátásra! A votre santé!

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

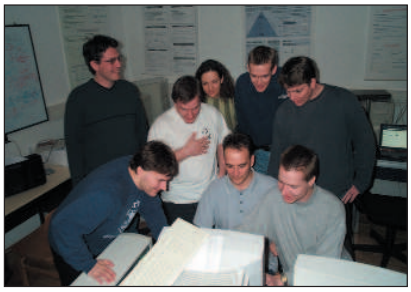
Végre magyar az OpenOffice!

Egy lelkes csapat egyetlen hétvége alatt elkészítette a fél éve húzódó munkát.

Amióta az irodai alkalmazások közül választani lehet, verseny van. A programok között válogathatunk fogyasztói ár, jogszerződés és tudás alapján. De miért van az, hogy amikor szabadon használható szövegszerkesztők egész sora áll rendelkezésre, mégis sokan fizetős termékeket vásárolnak? Nos, nézzük: különböző tudás, különböző felhasználói felület (aki hozzászokott az egyik programhoz, akár fizet is, csak ne kelljen másikat megtanulnia), a támogatás minősége, az elérhetőség és a megbízhatóság.
Egy fizetős programtól az ember elvárja, hogy könnyen használható és megbízható legyen, olyan szolgáltatásokat nyújtson, amelyekre igényünk van, és ha bajba kerülünk, legyen valamiféle elérhető támogatás. Egy szabadon használható termék esetén természetesen hajlandóak vagyunk engedni az igényekből, de vannak dolgok, amikhez ragaszkodunk. Egy irodában például ahhoz, hogy a programnak magyar felülete legyen.
Az egyik legismertebb ingyenes irodai programcsomag, a StarOffice, majd később nyílt forrású testvére, az OpenOffice.org magyar felülete már nagyon hosszú ideje várat magára. Bár voltak kezdemények, részleges magyarítások, sőt, az OpenOffice.org rendszeren alapuló fizetős változat is akad, mégis ami nyílt programok közösségének oly fontos lenne, egy teljes és egységes, ugyanakkor szabadon használható magyar felület egyszerűen nem létezett. Közben több vetélytárs is „megemberelte” magát, például a KOffice szinte teljes magyar felülettel rendelkezik.
Három lelkes szakember – ***Noll János***, ***Somogyi Péter*** és ***Tóth László*** – megunta a többéves vajúdást, és elhatározták, hogy egy hosszú hétvége alatt a teljes fordítást elkészítik. Ehhez természetesen meg kellett szervezni a munkát, ki kellett nyerni a fordítandó szövegeket a forrásból, majd készíteni kellett egy keretrendszert, amelynek segítségével a közel huszonegyezer kifejezést kezelni tudták. Magától értetődően szükség volt még egy erős gépre, melyen az összeépítést végezték, valamint egy folyamatos és szélessávú internetkapcsolattal rendelkező gépteremre is.
A háttérrendszer elkészítését Noll János vállalta magára. Egy PHP-alapú rendszerrel oldotta meg, hogy a lelkes fordítók távolról is be tudjanak kapcsolódni. Ez a rendszer a fordítandó kifejezéseket



egy PostgreSQL-adatbázisból ajánlotta ki, valamint a lektorok számára biztosított egy területet, ahol a beérkező fordításokat jóvá tudták hagyni, vagy el tudták vetni. A kész fordításokat Perl-programokkal varázsolták vissza a forrásba. A közösség segítőkészségére jellemző mozzanat volt, hogy a munka elkezdéséig (a Weben keresztül elérhető fordítórendszert két órával korábban már beüzemelték) a kifejezések 6,7 százalékát már a Weben keresztül le is fordították.
A fordításokat időközönként visszarakták a forrásba, majd összeépítettek egy új próbaváltozatot. A teljes programcsomag fordítása egy 1 GHz-es gépen hat órát vesz igénybe, szerencsére a fordítás során elegendő volt a változott részeket újrafordítani, majd újraépíteni a csomagot. Hogy ne kelljen minden változat kipróbálásához a nyolcvan megabájtos csomagot letölteni, az összeépítés után csak a megváltozott ***.res*** erőforrásfájlokat rakták ki a Webre.
Így egy újabb változat aránylag gyorsan (fél óra alatt) készülhetett el, a távolról segítők pedig csupán néhány megabájtnyi anyagot kellett letöltsenek.
A nyersfordítás az első nap alatt elkészült, így a második napot teljes egészében a lektorálásra tudták fordítani.



Erre szükség is volt, hiszen egy ilyen fordításnál számos hiba jelentkezhet. A második nap végén a lektorok által kijavított anyagokkal összeépített csomag végül is használhatónak bizonyult, így a hétvége minden szempontból sikeresnek mondható. Az utolsó változat került ki a fordítás honlapjára ➲ http://office.fsf.hu, illetve a Linuxvilág előző számának mellékletére is.

Merre tovább?

Természetesen sok munka van még az OpenOffice.org csomaggal. Egy újabb lektorálási kör várat még magára, valamint fontos, hogy minél jobb magyar helyesírás-ellenőrző rendszerrel is bírjon. A fordítás óta eltelt időszak eredményéből, valamint a visszajelzésekből egyértelműen látszik, hogy nagyon hasznos lenne egy következő hétvége. De máris újabb vadra fáj a csapat foga. A tervek szerint a következő hétvégén a Mozilla böngésző felületét fordítják le. Feladat tehát már van, sőt, a helyszínre is akadnak ötletek.

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

Öt rendszernek hét magja

David együttműködési gondokat keres a különféle terjesztések, rendszermagok és programok között.

Tudom, tudom, nem vagyok átlagos Linux-felhasználó. De megkérdezném: ki számít annak? Csak remélni tudom, hogy a legtöbb felhasználó nem találkozik azokkal a gondokkal, amelyekkel én nap mint nap szembesülök. Otthonomban általában öt gép található, vállalkozásom pedig számos kiszolgálóval és hozzáférési ponttal rendelkezik az Interneten. Egyre gyakrabban megütközöm azon, hogy mivel számos program nem található meg több terjesztésben, mások pedig meghatározott rendszermagra épülnek, végül négy különböző terjesztést használok hét különféle rendszermaggal azon az öt gépen, amit otthon tartok. A cikk írásának időpontjában (az utolsó rendszermagváltozat: 2.4.17-pre6) a legfrissebb Internet PhoneJACK (CVS) program a négy legújabb rendszermaggal nem működik, más programok fordításához pedig például a 2.4.4–2.4.8 változatú rendszermag szükséges. Néhány program fordításához és futtatásához a rendszermagok közötti átfedés nem engedhető meg, ezért található az öt gépemen hétféle rendszermag. A FreeS/WAN egyes rendszermagokkal lefordul, azonban nem az összessel. Bizonyos terjesztéseknek saját hibáik is akadnak. A Mandrake 8.1, amelyben rengeteg csiricsáré kiegészítőt találunk, nem tartalmazza a vezeték nélküli hálózatok használatához szükséges segédprogramokat, illetve nem támogatja a CardBust vagy az ORiNOCO kártyámat – természetesen megoldottam a nehézséget, de talán nem minden kezdőnek ez az álma. A Calderának rossz szokása, hogy a rendszermagot frissíti ugyan, de magát a rendszermagváltozatot már nem. Feljavítják a szükséges biztonsági foltokkal, ámde a forrás marad 2.4.2-es. Mondanom sem kell, hogy bizonyos programokat nem lehet lefordítani, hiszen 2.4.4-es vagy újabb rendszermag kell hozzájuk. A legtöbb olvasó számára bizonyára nem hatnak újdonságként a Red Hat terjesztéseivel kapcsolatos gondok. A megoldás? Tartok tőle, hogy semmit sem tudok javasolni. Abban viszont biztos vagyok, hogy amit én csupán bosszantó pepecselésnek tartok, azzal számos újoncnál betelhet a pohár. Nincs az a támogatás, amellyel a Linux a Microsoft-hatalmasság fölé kerekedhetne, ha ezeket a gondokat nem sikerül megoldani.

XNetworkStrength

Vezeték nélküli kártyát használsz? Szeretnéd tudni, hogy mennyire jó vagy rossz a jel minősége, de nem akarsz másodpercenként `iwspy`-t futtatni? Nos, ha X-felületet használsz, az XNetworkStrength megteszi helyetted. Miközben a munkádat végzed, folyamatosan a kapcsolaton tarthatod a szemedet. A futtatáshoz libX11 és glibc szükséges.
➲ http://gabriel.bigdam.net/home/xnetstrength

CGIpaf

Egyszerű és biztonságos módszert szeretnél biztosítani a felhasználók számára jelszavuk megváltoztatására? Ezek a CGI-eszközök lefordított C-programok, amelyek biztonságosak, ám lehetőleg csak HTTPS-protokollon keresztül tedd lehetővé az elérésüket (a jelszóváltoztatást nem túl jó ötlet nyílt csatornán keresztül intézni). A segítségükkel a felhasználók leveleket továbbíthatnak, illetve vissza is küldhetnek – hasonlóan a `vacation`-höz. Ellentétben a vacationnel, a program a levelezési listák kezelésére sajnos nem képes. A futtatáshoz szükséges: libdb1, libpam (elhagyható), libdl, glibc és webkiszolgáló PHP-támogatással.
➲ http://stafwag.f2g.net/cgipaf

Vipul's Razor

A múlt hónapban a levélszemétküldők ellen intéztem kirohanást, most pedig ismertetném az ellenszert. Ha MTA-t futtatsz (nekem több is van), a levélszemétnek egyetlen általános */etc/procmailrc* állománnyal útját állhatod (illetve ugyanezt egyéni *~/.procmailrc* segítségével is megteheted). Az enyém külön gyűjtőbe tesz minden szemetet. Körülbelül egy hónapig figyelemmel követtem, hogy milyen levelek kerülnek a gyűjtőbe, és egyetlen rendes levél sem futott vakvágányra. Mostanában nagyjából háromnaponta kapok valamilyen szemetet, amit a *razor*-kiszolgálóknak azonnal tovább is küldök. A levélszemét mennyisége múlt karácsony óta állítólag 650 százalékkal növekedett, de én köszönöm, jól elvagyok. A futtatásához szükséges a Perl és a következő Perl-modulok: Net::Ping, Net::DNS, Time::HiRes, Digest::SHA1, Mail::Internet, valamint erős akarat a levélszemét kiirtására.
➲ http://razor.sourceforge.net

NorthStar

A NorthStar segítséget nyújt az IP-cím hozzárendelések, valamint értelemszerűen a készülékek és elhelyezkedésük nyomon követéséhez. Igazából az benne a legjobb, ahogyan a hálózatokat, az eszközöket és a helyüket jeleníti meg. Ha nem csak néhány IP-címet, rendszert és telephelyet kell kezelned, mindenképpen próbáld ki. A futtatáshoz szükséges: webkiszolgáló, Perl és PostgreSQL.
➲ http://www.brownkid.net/NorthStar

Visszatekintő: mail-bounce

A kisméretű Perl-program segítségével a leveleket foghatod, és egyenesen oda küldheted vissza őket, ahonnan jöttek. Külön egyedi megjegyzést is fűzhetsz hozzájuk. Mivel az üzeneteket bármikor vissza lehet küldeni, nincs értelme órákat várni, ezért erre a célra a `procmail` használata ajánlott, bár tulajdonképpen bármilyen programot, akár a parancssort is használhatjuk. A futtatásához a Perl szükséges és a `procmail` ajánlott.
➲ http://www.spots.ab.ca/~gary/mail-bounce

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