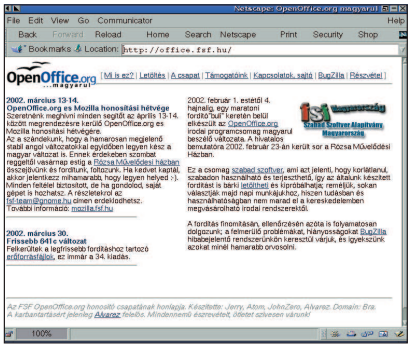
Ismét magyarodunk...

Lapzárta előtt kaptuk a hírt: újabb linuxos program magyarítására készülnek!.

Az FSF tagjai újabb, a magyar Linux-közösséget kellemesen érintő lépésre szánták el magukat. Az OpenOffice.org sikeres magyarítása után most a Mozilla böngésző, levelező, html-szerkesztő következik. Ez a projekt, mint honlapukon írják, 2002. március 11-én indult útjára, illetve indult újra, azaz a 0.9.2-es Mozillához ***Horvát Szabolcs*** által elkészített magyar nyelvi csomag „félbemaradt" fejlesztésének a folytatása. Ezzel szintén nagy űr töltődik be országunkban, az angolul nem tudók pedig újabb lehetőséget



kapnak a szabad világhoz való csatlakozáshoz. Amikor egyik, a Linuxot irodai célra is használó cégnél egy ismerősöm megmutatta a felhasználóknak a teljesen magyar felületű irodai csomagot, mindenkiből kitört az örömujjongás. Most már olyan dolgokat is képesek létrehozni, amire eddig nem is mertek gondolni, mivel értik a menüpontokat és nem félnek kipróbálni az újabb dolgokat sem.

A magyarítás a Rózsa Művelődési Házban 2002. április 13-14-én kerül megrendezésre „OpenOffice.org és Mozilla honosítási hétvége" néven, melynek keretében többek közt szeretnék befejezni, kipróbálni és kijavítani a Mozilla fordítását, hogy az a hamarosan megjelenő 1.0-s változatban már a lehető legjobb minőségű legyen.

A Mozilla projekt céljai:

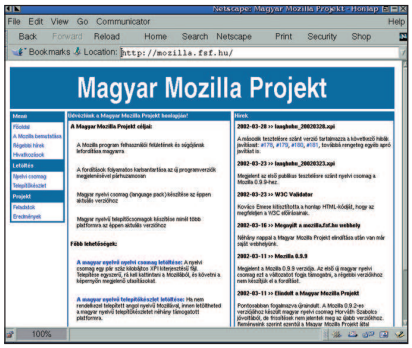
- A Mozilla program felhasználói felületének és súgójának magyarítása.
- A fordítások folyamatos karbantartása az új programváltozatok megjelenésével párhuzamosan.
- Magyar nyelvi csomag (language pack) készítése a legfrissebb változathoz.
- Magyar nyelvű telepítőcsomagok készítése minél több felületre a legújabb változathoz.

További érdekességek tölthetők le a következő oldalakról:

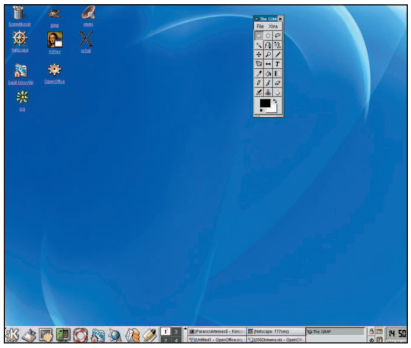
➲ http://mozilla.fsf.hu/
➲ http://www.fsf.hu/
➲ http://www.rozsamh.hu/

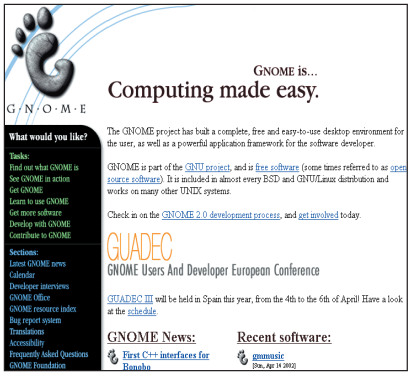
Sajnos ez az időpont lapzártánkkor túl esik, így az eredményről csak egy hónap múlva, a következő számban tudunk beszámolni, mint azt az OpenOffice.org esetében is tettük.

Azt hiszem, hogy az utóbbi időben felgyorsult linuxos fejlesztések is azt iga-



zolják, a Linuxnak igenis helye van az asztali gépek piacán is. Ez a hónap sem telt el egy kis vagdalkozás nélkül, az egyik internetes oldalon megjelent egy ősrégi írás, ami a Linux-Microsoft kettős párviadalát hivatott a csak pénzért megvásárolható rendszerek felére eldönteni. Miután elolvastam, az jutott az eszembe, milyen furcsa is a világ: mindannyiunk kedvence, a Linux hihetetlen sebességgel változik és fejlődik. Ebben a cikkben (azt hiszem, 1999-ből való) még fekete dobozként emlegetik a Linuxot – karakteres felület, csak guruknak! Természetesen ez akkoriban sem volt igaz, már akkortájt is lehetett nagyon jó, könnyedén kezelhető grafikus felületet találni. Én személy szerint az ablakkezelők közül az AfterStep mellett tettem le a voksomat: kicsi, gyors, megbízható, testreszabható – ezek azok a jellemzők, amelyek nekem számítanak. Azonban ne felejtsük el a Gnome-, a Ximian-Gnome- és a KDE-környezeteket sem. Ezek inkább amolyan „teljes grafikus környezetek", ahol mindennek megvan a beépített programja, és ha mégsem lenne, akkor egy külsőt hív segítségül. Felületünket grafikusan állíthatjuk be a legapróbb részletekig, bármit elvégezhetünk az ezernyi apró vagy kevésbé apró programmal, és ami a legfontosabb: tehetjük mindezt úgy, hogy a karakteres felülethez hozzás sem kell nyúlnunk.





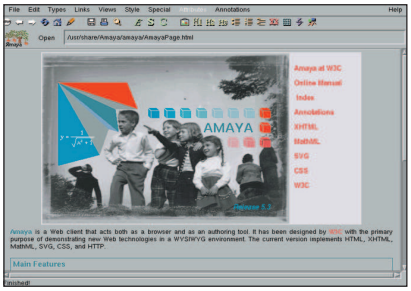
Örülök ennek a fejlődésnek, mivel így már a kevésbé elszántak is sikereket érhetnek el, és egyre nagyobb tábora lesz a szabad rendszereknek.

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

Programvadászat

Amaya 5.3

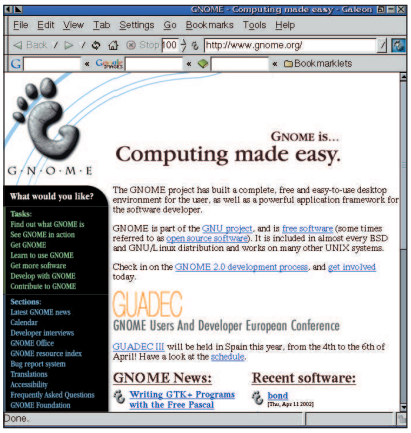
Az Amaya 5.3 a Word Wide Web Consortium ajánlásainak bemutatására készített grafikus böngésző és webszerkesztő. A program támogatja a HTML, az XHTML, a HTTP, a MathML, a CSS, az SVG és a PNG ajánlásokat. Megjegyzések hozzáfűzése a dokumentumokhoz



az RDF, az Xpointer és az Xlink segítségével lehetséges. A webszerkesztővel készített dokumentumok minden esetben megfelelnek a DTD-nek, azaz a Dokumentumtípus-meghatározásnak. A program a CD-mellékleten megtalálható és előre elkészített rpm- és tar.gz-csomagok segítségével telepíthető.

Galeon 1.2.0

Megjelent a Galeon újabb, 1.2.0-s változatszámmal rendelkező böngészője. A GTK-ban megírt kezelőfelülettel rendelkező program működésének alapfeltétele a Mozilla 0.9.9-es változatának jelenléte. Ez a rendkívül fürge és egyszerűen kezelhető böngésző – valószínűleg népszerűségének is köszönhetően – a Gnome és a Ximian Gnome alapértelmezett böngészője. A Galeon egyedülálló



sajátossága, hogy bármely letöltéskezelővel képes az együttműködésre. A beépített letöltéskezelőn kívül a WebDownloader for X, a Gnome Transfer Manager, a Wget, a ProZilla, továbbá bármely parancssorból is meghívható program alkalmazható. A friss kiadás a hibajavításokon túlmenően több újdonságot is tartalmaz. A menüsor az ***Oldal adatainak megjelenítése*** és az ***E-mail cím másolása*** elemekkel bővül. További fejlesztésen ment keresztül a ***Tab*** fül, amely ezentúl képes a ***Favicon.ico*** és az oldal címének helyes megjelenítésére.

Netscape 6.2.2

Megjelent a Netscape újabb változata. Az ingyenes böngésző rendelkezik egy levelezőprogrammal, egy HTML-szerkesztővel, az AOL csevegőjével, egy jelszókezelővel és egy ***Form Managerrel***. A 6.2.2-es változatszámmal rendelkező böngésző újdonságokat nem tartalmaz. Továbbra is a Mozilla 0.9.4-es kódjára épül, mellőzve a Mozilla újdonságait: a ***Tabbed***-et, a

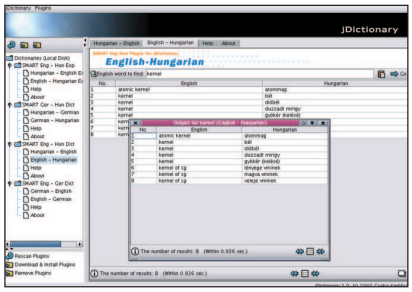
Favicon.ico megjelenítését és a nyomtatási előképet. A javításoknak köszönhetően a mind a sebessége, mind a megbízhatósága jelentősen növekedett. A Netscape újabb változatára történő frissítés minden felhasználó számára ajánlott.

Mozilla magyar nyelvi csomag

Elindult a Magyar Mozilla Project. Célja a program felületének és a súgónak a magyarítása, illetve telepítőcsomagok készítése minél több felületre a pillanatnyi változathoz. A fordítási munka rengeteg feladattal jár, hiszen a program folyamatosan fejlődik, új változatok és bennük új karakterláncok jelennek meg. A hibák és az új feladatok kitűzése, megvalósításuk nyomon követése a Bugzilla rendszerén keresztül zajlik. Jelentkezését itt teheti meg az, aki kellő kedvet, illetve elkötelezettséget érez a Mozilla magyar nyelvi csomagok elkészítéséhez. Az első nyilvános és kipróbálható nyelvi csomagok megjelenése és javítása után CD-mellékletünkre a jelenlegi második próbaváltozat került fel. Telepítése a böngészőből a ***langhuhu_20020328.xpi*** fájl megnyitásával történik. A Linux-változatokra történő telepítéskor ügyelnünk kell arra, hogy a ***Mozilla chrome*** könyvtárára írási joggal rendelkezzünk. A már telepített magyar nyelvi csomag frissítése esetén a műveletet az angol nyelvi beállítás alól ajánlott elvégezni.

jDictionary 1.0

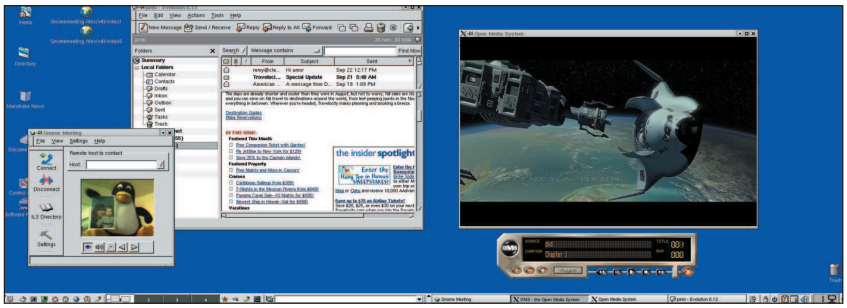
Megjelent a jDictionary 1.0 magyar fejlesztésű szótárprogram. A program szép grafikus felülettel rendelkezik és minden elterjedt operációs rendszeren futtatható. Működtetésének egyetlen feltétele egy a Java futtatására alkalmas rendszer. A szótárfájlok bővítményalapúak, azaz csak a használni kívánt

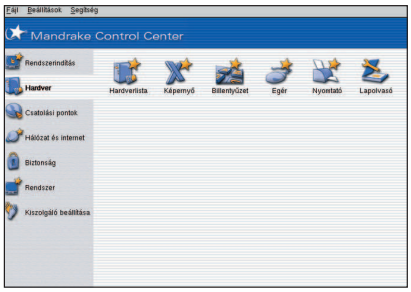


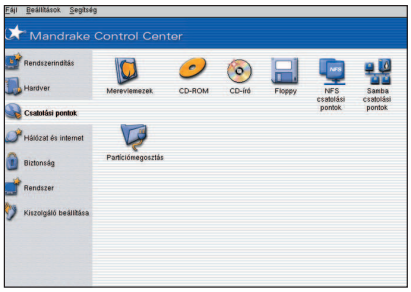
bővítményeket kell letölteni és telepíteni. Jelenleg az angol-magyar, a német-magyar és az angol-német bővítmények állnak rendelkezésünkre. Fejlesztés alatt áll a beszédszintetizátor és más nem kifejezetten nyelvi bővítmény is, például a lexikon. A program a ***jDictionary.tar.gz*** fájl kicsomagolása után a `java -jar jdictionary.jar` paranccsal indítható.

Mandrake 8.2

E havi CD-mellékletünkre a Mandrake Linux legfrissebb változata került fel. A világ egyik legelterjedtebb Linux-változata, mely népszerűségét elsősorban felhasználóbarát grafikus telepítőjének, valamint megbízhatóságának köszönheti. Könnyű telepítés, beállítás és használat jellemzi. Telepítése 15–20







percnél többet nem vesz igénybe – akár otthoni munkaállomásnak, akár internetkiszolgálónak telepítjük. Mind a telepítő, mind a kezelőfelület magyarított, így mindennapi használata az angol nyelvismerettel nem rendelkezők számára is gyerekjáték. Már maga az ingyenesen letölthető változat is minden olyan alkalmazást tartalmaz, amire a mindennapi használat során szükségünk lehet. Természetesen a Mandrake Soft fejlesztői kínosan ügyeltek arra, hogy csak a legjobb és a legmegbízhatóbb alkalmazások kerüljenek a CD-re.
A Red Hat-alapú Linux-változatokra jellemzően a programok telepítése és eltávolítása rendkívül egyszerű. A Red Hat ***Package Manager*** a feltelepített programokról egy adatbázis-nyilvántartást vezet, így a csomagok telepítése és eltávolítása is megfelelően összehangolt. Ezt az egyszerű telepítési módszert tovább könnyíti a Mandrake ***Szoftvercsomag-kezelője***, mely képes a függőségi viszonyok feloldására, azaz a hiányzó csomagok telepítésére.
A rendszer telepítése pár kattintással, illetve billentyűleütéssel elvégezhető. Választhatunk az ajánlott és a szakértői telepítési módszerek között. Bátran döntsünk a szakértői mellett, mert elnevezésével ellentétben sokkal egyszerűbb dolgunk és testreszabottabb rendszerünk lesz. Választhatunk a ***Telepítés***, a ***Frissítés*** – mely csak Mandrake 8.1 esetén javasolt – és a ***Csak a csomagok frissítése*** között is, amely teljes Mandrake rendszerünk frissítését eredményezi a már meglévő rendszerbeállítások módosítása nélkül. A meglévő rendszer helyreállítására, illetve frissítésére a ***Frissítés*** alkalmazása ajánlott. A grafikus telepítő pontról pontra végigvezet bennünket a telepítés menetén. Szinte semmit sem lehet elrontani: ha úgy érezzük, hogy valamit nem jól állítottunk be, a menüben a javítás erejéig bármikor visszaléphetünk. Miután beállítottuk a rendszer és a billentyűzet alapértelmezett nyelvét magyarra, következhet a lemezterületek létrehozása. Az egyszerűen és könnyen kezelhető ***DiskDrake*** átvállalja a lemezterület létrehozása gyötrelmének terheit. A varázsló segítségével lehetőségünk nyílik a felszabadított lemezterület otthoni munkaállomás, illetve kiszolgálótípusú felosztására is. Itt az ext2 típusú fájlrendszert érdemes a jóval korszerűbb ***ReiserFS*** naplózott rendszerére felváltani, így váratlan áramszünet esetén sem fog kellemetlen meglepetés érni bennünket. Ezután következik a lemezterületek formázása – ha a szakértői telepítést választottuk, a ***Home*** könyvtárban lévő adathalmazunkat egy kattintással megóvhatjuk a teljes megsemmisüléstől. A csomagok kiválasztása és telepítés után már csak az alapszintű beállítások következnek. A rendszergazda ***Root*** és a felhasználók bejegyzése után a hálózati beállítások következnek. A többi beállításhoz hasonlóan a ***Hálózatbeállító varázsló*** segítségével hálózatunkat önműködően felderíthetjük, majd könnyedén beállíthatjuk. Miután a telepítő elvégezte a rendszerindítási beállításokat, lehetőségünk adódik indítólemez készítésére. Elkészítése erősen ajánlott, hiszen rendszerindítási gond esetén ezzel a lemezzel még mindig életet tudunk lehelni kedvencünkbe. Amennyiben hibajavítás is szükséges, ezt a rendszer ***Failsafe*** módú indítása után végezhetjük el.
A Mandrake 8.2 elsősorban megújult és kimondottan látványos grafikus felületet és az előző változatokban előfordult hibák javítását tartalmazza. A 2.4.18-as

rendszermaggal a ***SuperMount*** szolgáltatás már tökéletesen működik: a CD, illetve hajlékonylemez behelyezése és a csatlakoztatási pont megnyitása esetén önműködően befűzi, míg a média eltávolításakor kifűzi azokat. Az előző változatokkal ellentétben adott a lehetőség az utólagos proxybeállítások elvégzésére, mely szolgáltatás a ***Mandrake Control Centerben*** található. Az irodai alkalmazások területén is történtek változások, a netről letölthető, illetve szabadon terjeszthető változatokban a Sun fizetős StarOffice csomagja helyett ezentúl az OpenOffice programot telepíthetjük, ezáltal pótolták az ezt megelőző Mandrake Linuxok sokat emlegetett hiányosságát. Az előző változatokkal ellentétben a telepíthető csomagok közé felkerült egy teljes értékű videolejátszó program, a Xine 0.9.8.-as változata. A program a legtöbb ismert videofájltípus lejátszására alkalmas és sikeresen megbirkózik a VideoCD-kel, az SVCD-kel és a DVD-kel is.
A Mandrake 8.2 használatát nyugodt szívvel ajánlhatom mind kezdő, mind haladó felhasználók számára. Rabulejtően könnyű telepítés és használat mellett sikerült megőriznie a Linux megbízhatóságát és erejét. Remélhetőleg leírásommal és a CD-mellékleten szereplő Mandrake 8.2-es változatával sikerül a kezdő Linux-felhasználók életét megkönnyíteni, és a Pingvin sötét oldalára átcsábítani.

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

IBM: új linuxos kiszolgáló és próbalabor

Az eServer-sorozat új tagja az x343, amellyel az IBM a távközlési ipart célozza meg. A cég nem titkoltan a Sun kiszolgálóival próbálja felvenni a versenyt, hiszen eddig csak jóval drágább, Unix-alapú kiszolgálóival tudta piaci terület igényeit kielégíteni.

Az x343 Intel-alapú kiszolgáló két egység magas, szekrénybe szerelhető gép. Két Intel Pentium III processzor kaphat benne helyet 2 GB hibajavító memória társaságában. A hálózati kapcsolatról kettős 10/100 Mb/s sebességű ethernet-csatoló gondoskodik, a további bővítés lehetőségét 6 PCI-foglalat biztosítja. Az üzembiztos működést a redundáns tápegység, a külső riasztópanel és a beépített rendszerfelügyeleti processzor biztosítja.
Ugyancsak a távközlési ipar igényeit szolgálja az a próbalabor, amelyet az IBM Oregonban, az Egyesült Államokban hozott létre. A laborban a programfejlesztők termékük működését x343-as gépeken és Linux operációs rendszeren vizsgálhatják. Az IBM olyan Linux-változat fejlesztését is fontolóra vette, amely kifejezetten a távközlési ipar résztvevőinek igényei szerint készülne.
➲ http://www.pc.ibm.com/us/eserver/xseries/x343.html

Hancom Office hordozható gépekre is

Habár a sajtóközlemény szerint a fejlesztő a Sharp Zaurus gépéhez jelentette be HancomMobileOffice nevű termékét, ez nem jelenti azt, hogy mini irodai csomagját más készüléken ne használhatnánk. A HancomMobileOffice három alkalmazásból áll. Szövegszerkesztésre a Mobile-Word áll rendelkezésre, táblázatkezelésre a Mobile-Sheet használható, bemutatókat pedig a Mobile-Presenter segítségével tarthatunk. Mindhárom – egyszerű szolgáltatáskészletű – program képes a Microsoft Office formátumainak kezelésére.

A csomag 30 napig használható változata ingyenesen tölthető le a fejlesztő honlapjáról, a teljes változat ára egyelőre ismeretlen, de várhatóan ötven dollár körül fog alakulni.
➲ http://mobile.hancom.com/

Elérhető a KDE 3.0

Letölthető a KDE 3.0, a népszerű ablakkezelő legújabb változata. A csomag nem kevesebb, mint ötven nyelven

érhető el, 3.1-es változatának megjelenésével ez a szám tovább növekszik. Lélegzetelállító újdonságokat ugyan nem találunk az új kiadásban, ám milliónyi ponton javítottak a csomag használhatóságán, hibáin és teljesítményén.
Talán ennél is fontosabb, hogy szorosan hozzá kötődve megjelent a KOffice 1.1.1 változata is. Az ingyenes irodai csomag egyelőre csak azoknak nyújt megfelelő megoldást, akik nem nagyon cserélgetnek másokkal MS Office-dokumentumokat, de várhatóan ez az állapot is csak augusztusig, az 1.2-es változat megjelenéséig fog fennállni.
➲ http://www.kde.org
➲ http://www.kde.hu

Megjelent a CrossOver Office 1.0

A CrossOver Office segítségével Linux alól is futtatható a Microsoft Office programcsomag, valamint a Lotus Notes – két olyan alkalmazásegyüttes, amelyet a nagyvállalatok előszeretettel telepítenek gépeikre. Akik nem akartak lemondani a sok helyen szabványként alkalmazott Microsoft Office szolgáltatásairól, eddig kénytelenek voltak Windows emulátort, majd Windowst telepíteni linuxos gépükre, ezzel azonban – ha jogtisztán tették – anyagi téren semmit sem nyertek. A CrossOver Office feleslegessé teszi az emulátor és a Windows beszerzését és telepítését, így a vállalatok számára is értékes megoldássá válhat.
„A CrossOver Office 1.0-s változata csak az első lépés” – mondta ***Jeremy White***, a CrossOver alapítója és elnöke. „Idén még további népszerű alkalmazások támogatását meg szeretnénk valósítani, így többek közt a CrossOver termékei révén a Linux egyre erősebb versenyzővé válhat az asztali operációs rendszerek piacán.”
A CrossOver Office ára 54,95 dollár, amelyből nagyobb mennyiségű vásárlás esetén további kedvezményeket lehet kapni.
➲ http://www.codeweavers.com

Közös magyarországi tesztlabort létesített a Compaq és a Microsoft

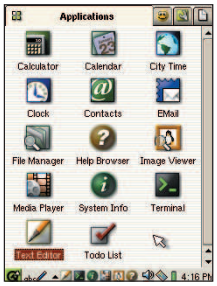
Az új, világszínvonalú, mintegy ötvenmillió forintos beruházással létrehozott Compaq-Microsoft .NET laborban a két cég ügyfelei még a bevezetés előtt meggyőződhetnek arról, hogy alkalmazásaik valóban megfelelő teljesítménnyel futnake a Compaq Datacenter megoldásain. Hasonló próbákat eddig csak nyugat-európai laborokban lehetett végezni, most azonban a magyarországi vásárlók előtt is megnyílt a hazai kipróbálás lehetősége. A labor idén ingyenesen vehető igénybe. Felszereltsége kiváló, hazánkban egyedül itt található kifejezetten tesztcélokat szolgáló Windows 2000 Datacenter Server. Ha az érdeklődők teljes informatikai hátteret szeretnének kipróbálni, 128 Kb/s sebességű bérelt vonal, GPRS mobiltelefonok, zsebtitkárok, hordozható számítógépek és további kiszolgálók állnak rendelkezésükre.

Robotizált Linux

A linuxizált robot elnevezés talán pontosabb lenne, hiszen a japán Kawada új emberszabású robotját Linux irányítja. A HRP-2P nevű robot operációs rendszere az RT Linuxon alapuló ART Linux, amelyet a különleges adatfeldolgozási, robotikai és rendszervezérlési feladatoknak megfelelően módosítottak, és amelyet a vásárlók a célfeladatnak megfelelően szabadon testreszabhatnak.
Az 58 kg súlyú, 154 cm magas robot végtagjainak mozgási szabadsága hasonló az emberéhez, ennél fogva alkalmas arra, hogy emberi munkaerőt váltsanak ki vele. A HRP-2P-t ugyanis néhány sorstársával ellentétben nem szórakoztatásra tervezték, hanem kemény munkára. Az alkalmazási lehetőségek köre rendkívül széles, hiszen most is számos olyan munkakör van, amelyben emberek nap mint nap kockáztatják az életüket.
➲ http://www.zdnet.com
➲ http://www.kawada.com

Megjelent a Trolltech Qtopia

A Qt norvég fejlesztője büszkén jelentette be Qtopia névre hallgató, mobil eszközökhöz készített alkalmazáskörnyezetét. Valószínűleg a Qtopia nagy népszerűségre tesz majd szert, hiszen segítségével hatékonyan és könnyen készíthetők beágyazott Linuxot futtató eszközök – jó példa erre a Sharp Zaurus nevű zsebtitkára, amelyről a Linuxvilág oldalain is olvashattunk már. A Qtopia számos előnyös jellemzővel rendelkezik, többek között készüléktől független, magas fokon testreszabható, a rendszermaggal együtt mindössze 6–8 MB memóriát igényel, és a forráskódja is hozzáférhető, ami a környezet további módosítását is lehetővé teszi.



A Qtopia fő összetevői maguk az alkalmazások, az asztali géppel végzett összehangoláshoz használt program, a Qt/Embedded-környezet, a Java, a különféle beviteli módszerek, valamint a honosítás támogatása, és nem utolsó sorban a forráskód.
➲ http://www.trolltech.com/products/qtopia/

Linux alá is elérhető a DivX 5.0

A DivX 5.0 Mac OS és Linux alá írt változata sajnos egyelőre csak lejátszani tud, ha magunk szeretnénk filmeket kódolni, továbbra is windowsos gépre van szükségünk, vagy a 4.02-es változatot használhatjuk. A fejlesztők ígérete szerint a linuxos csomaggal hamarosan már kódolni is lehet.
A DivX 5.0-s változata több javítást, módosítást tartalmaz, amelyek a jobb együttműködést és minőséget, valamint a nagyobb teljesítményt szolgálják. Képes MPEG-4 formátumú filmek kezelésére, a kódolóval előállított adatfolyam továbbításához szükséges sávszélesség pedig – beállítástól függően – 28 Kb/s és 9 Mb/s között rugalmasan méretezhető.
➲ http://www.divx.com/divx/maclinux.php

Megjelent a Mandrake Linux 8.2-es változata

A legsokoldalúbb és legtöbb szolgáltatást nyújtó operációs rendszer, amely valaha is elérhető volt a nagyközönség számára – így jellemzi a Mandrake saját 8.2-es terjesztését. Szükségszerű, hogy a készítő elfogult legyen a saját termékével szemben, ám a tényekkel nem érdemes vitatkozni. A cég legújabb terjesztésében 2.4.18-as rendszermagot találunk, javítottak a rendszer Firewire-támogatásán, valamint többek közt nem jelent gondot az USB 2.0 vagy az ATA133 felületen csatlakozó eszközök használata sem.
A telepítő nem kevesebb, mint negyven nyelven futtatható, de a ***Vezérlőközpontot*** is újratervezték. Érdekesség az Rfbdrake, amely távoli X-munkamenet indítását teszi lehetővé, így belenézhetünk segítségre szoruló ismerősünk grafikus felületébe.

Mandrake Linux

Akik kiszolgálót építenek, választhatják a kiszolgálókhoz készített rendszermagot, amely támogatja az 1 GB-nál nagyobb memóriával és egynél több processzorral rendelkező gépeket. Természetesen nem maradhattak el a legújabb – naplózásra is képes – fájlrendszerek sem, ha pedig valamilyen meghatározott célra állítunk össze linuxos gépet, a mindössze 65 MB helyet foglaló legkisebb telepítés is hasznos lehet. Ha nem tartjuk teljesen értelmetlennek a csoportmunkát segítő szolgáltatásokat, telepíthetjük a PHP Groupware-t, a megszokott Apache-t, a PHP-t, a MySQL-t – természetesen rajtuk kívül sok más is megtalálható a terjesztésben.
A Mandrake Linux 8.2-es változata több összeállításban érhető el, a ProSuite- és PowerPack-összeállítások már a StarOffice 6.0-s változatát is tartalmazzák.
➲ http://www.linux-mandrake.com/en/

Év végére piacra kerülnek a Windows-alapú mobiltelefonok

A Sendo – egy 1999 végén alakult angol vállalkozás, amelyben a Microsoft alig tízszázaléknyi részesedéssel rendelkezik – bejelentette, hogy év végére első Smartphone 2002-alapú zsebtitkár-mobiltelefon készülékei megvásárolhatók lesznek.

A Z100-as készülék rokonszenves jellemzőkkel rendelkezik: egyaránt képes a 900, 1800 és 1900 MHz-es GSM-hálózatok használatára, és GPRS-adatátvitelt is folytathatunk vele. Súlya mindössze 99 gramm, 176×220 képpontos TFT LCD-kijelzője 65 000 színt képes megjeleníteni. Egyelőre nem tudni, hogy mennyi RAM kerül bele, de 32 MB flashmemória mindenképpen a rendelkezésünkre áll majd, amit MMC- és SD-kártyákkal tovább bővíthetünk. Központi processzora a TI ARM 9-es magjára épül, készenléti ideje 200 óra, beszélgetési ideje pedig 2 óra.
A Sendo nemrég megvásárolta a Tao Group intent multimedia Java-környezetének használati jogát is, és elérhetővé kívánja tenni Z100-as készülékein.
A Tao környezete megfelel a Sun rá vonatkozó követelményeinek és Java-szabványainak.
A Tao Group környezetét felhasználva a hálózati operátorok – azaz a mobiltelefon-szolgáltatók – egyedi alkalmazásokat telepíthetnek ügyfeleik készülékére, és értéknövelt szolgáltatásokat nyújthatnak számukra.
A cég még nem jelentette be, hogy a telefont hol lehet majd kapni, de várhatóan Spanyolországban, Franciaországban és Németországban is felbukkannak majd a készülékek, így az előfizetéssel együtt 400 eurós árú mobil beszerzése valószínűleg hazánkban sem okozhat majd gondot.
➲ http://www.sendo.com

Kiadták az Unicode-szabvány 3.2.0 változatát

Az Unicode Consortium elkészült az Unicode karakterszabvány új változatával. A hosszan tartó munka eredményeként a legjelentősebb fejlesztések a matematikai és műszaki területeken használt karaktereknél történtek, amelyekből 1600 darab új került a szabványba, megkétszerezve ezzel a rendelkezésre álló jelek számát. A szervezet honlapjára már csak azért is érdemes ellátogatni, mert a szabványok leírásaiban nagyon érdekes, egzotikus írásjeleket találhatunk.
➲ http://www.unicode.org/

Egyelőre maradhat a Lindows név

A Lindowsnak lassan a körülötte gyűrűző kakaskodás lesz a legjobb reklám: a windowsos programok futtatására tervezett operációs rendszer nevét a Microsoft még decemberben bíróságon kifogásolta. A cég érvelése szerint a név túlságosan közel áll a jól ismert Windows névhez, amely viszont bejegyzett kereskedelmi védjegy. A Lindows név alkalmas a Windows ismertségének kihasználására, ezért tulajdonosa arra kérte a bíróságot, hogy tiltsa el használatától a San Diego-i Lindows.com Inc.-t. *John C. Coughenour* bíró azonban alaptalannak vélte a Microsoft félelmét, sőt, a Windows (magyarul ablakok) nevet túlságosan általánosnak ítélte ahhoz, hogy védjegy lehessen.

A Microsoft nem nyugodott bele a döntésbe, és újabb vizsgálatot kért, természetesen más bíró részvételével.
➲ http://www.lindows.com/

Megújult kétprocesszoros Tyan alaplap

A Tyan bejelentette K7-es alaplapjának utódját, amelyet egyszerűen K7X névre kereszteltek. Az AMD-760 lapkakészlet köré épülő lapba AMD-processzorokat helyezhetünk. A Tyan terméke egészen elképesztő, a kiszolgálókhoz készült alaplapokból már ismert képességekkel rendelkezik. A nagysebességű memóriaelérésről 266 MHz-es rendszersín gondoskodik. Az összesen 4 GB PC 2100-as DDR-memória befogadására képes memóriafoglalatokat – gondolva a rendszerépítő cégekre, akik sokszor helytakarékos megoldásokat keresnek – 25 fokkal megdöntve építették rá. A bővítést szolgálja a két 64 bites és a három hagyományos, 32 bites PCI-foglalat – már ha tudunk valamit bővíteni a gépen, hiszen hálózati kapcsolatot kettős 10/100 Mbit sebességű ethernetcsatolóval tud létesíteni, a merevlemezeket pedig választhatóan beépített Ultra-160 SCSI-vezérlő várja. VGA-kártyára sem lesz gondunk, az alaplapra egy ATi Rage XL vezérlő került, és ha ez nem felelne meg, még mindig kihasználhatjuk a 4x AGP-foglalat nyújtotta lehetőségeket.
➲ http://www.tyan.com

Nagyfelbontású Lexmark-nyomtatók

A Lexmark Z-sorozatának új, nagyteljesítményű tagjai messze túllépnek a megszokott tintasugaras nyomtatókon. A sorozat legkisebb tagja is 1200×1200 dpi felbontásra képes, a legnagyobb Z65-ös pedig – amely akár beépített hálózati kártyával is megvásárolható – 4800×1200 dpi felbontást teljesít, miközben gyorsnyomtatásnál akár 21 oldallal is végez egy perc alatt. A Z65 két papírtálcával gazdálkodik, amelyek legfeljebb 250 lap befogadására képesek, és USB 2.0 csatlakozóval rendelkezik. A festékpatronokat – cián, magenta, sárga és fekete – természetesen külön cserélhetjük benne.
➲ http://www.lexmark.com

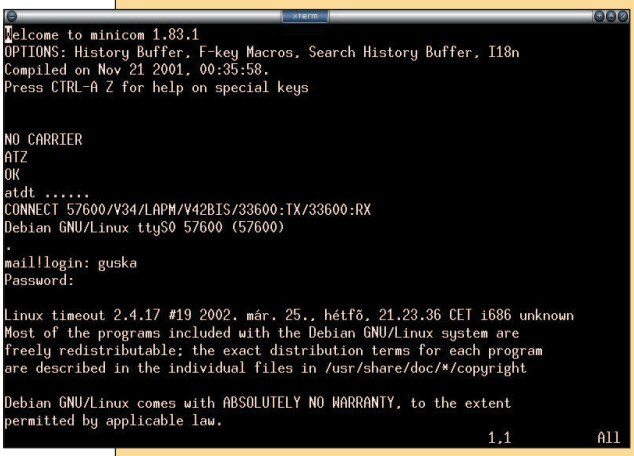
Hamarosan linuxos karóránk is lehet

Sajnos pontosan még mindig nem lehet tudni, hogy az IBM–Citizen együttműködésben készülő WatchPad mikor kerül az üzletek polcaira. Biztató, hogy dolgoznak rajta, és már az 1.5-ös változatnál tartanak.
A WatchPad új változata 320×240 képpontos kijelzőt kapott, amely negyede a szabványos VGA-felbontásnak, és a zsebtitkároknál megszokott felbontással megegyező. Rendelkezik ujjlenyomat-leolvasóval is, így nemcsak maga a karóra tehető biztonságossá, de számos biztonsági alkalmazáshoz is felhasználható. Az aprócska készülék érzékeli a kar mozgását is, valamint vibramotorral, hangszóróval és mikrofonnal rendelkezik. Li-ion akkumulátora egyelőre csak 6 órás üzemidőt képes biztosítani, ám az IBM az év végére az egész napos működést is el szeretné érni.
A programok oldaláról is érdekes alkotás a WatchPad, ugyanis 2.4-es rendszermag fut rajta, amit Microwindows grafikus felület egészít ki. Az operációs rendszer egy 32 bites RISC processzoron fut, amelynek órajelét – energiatakarékossági okokból – 18 és 74 MHz között változtatja. Bluetooth és infravörös kapcsolatot egyaránt tudunk létesíteni vele, így nemcsak aprócska zsebtitkárként, hanem beléptetésre, fizetésre és üzleti bemutatóknál kiegészítő eszközként is használható.
➲ http://www.trl.ibm.com/projects/ngm/index_e.htm0

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 – 2. rész

Ígéretet tettem rá, hogy folytatom ezt a témát, méghozzá a kiszolgálófelügyelet témakörével. Nézzük, mi haszna lehet egy Linuxra kötött telefonnak a kiszolgálókörnyezetben! Már egy kiszolgáló üzemeltetése esetén is érdemes folyamatosan figyelemmel kísérnünk gépünket, hiszen egy behatolást vagy kiszolgálóleállást általában olyan események szoktak megelőzni, amelyekből a legtöbb esetben ki lehet találni, hogy baj lesz.



Ez természetesen csak akkor lenne lehetséges, ha a nap 24 órájában a konzol előtt ülve figyelnénk a naplóbejegyzéseket. Amennyiben előre gondolkodunk, és megfelelő eszközt választunk a naplók elemzésére, hamar értesülhetünk arról, hogy a lemez be fog telni, vagy valaki behatolással próbálkozik. Szinte mindegyik elemzőprogram lehetőséget kínál rá, hogy a figyelmeztetést levélben jutassa el hozzánk, vagy az adatokat egy másik programnak adja át. Ebben az esetben mi sem egyszerűbb, mint ha az előző cikkben leírtak alapján a Gnokiit SMS-küldésre állítjuk be, valamint rávesszük a naplózó programot, hogy a Gnokiin keresztül küldjön nekünk értesítést. Ha a figyelőprogram nem ad módot az SMS közvetlen küldésére, legyünk ötletesek! Az adatokat akár egy pár soros héjprogram megalkotásával átadhatjuk, például az alábbi módon:

- A figyelőprogram csak elektronikus levelet tud egy megadott címre küldeni – semmi baj, írjunk egy Perl-programot, amelyet meghívunk a Procmailből, így a levél tartalma máris a kezünkben van, és egyszerűen át tudjuk adni a Gnokiinak:
  ```
  Procmail:
  :0 c | /home/guska/sms-kuldo.pl
  ```
  Ebben az esetben a levél teljes tartalmát átadtuk az *sms-kuldo.pl* programnak, amely esetleg 160 karakterre darabolja fel, és a `gnokii --sendsms` segítségével eljuttatja a telefonunkra, továbbá egy fájlba természetesen feljegyzést készít az üzenet küldéséről, hogy tudjuk, mennyi SMS-t küldtünk.
- A figyelőprogram csak egy meghatározott naplófájlba tud bejegyzést írni – ugyancsak semmi gond: vagy írunk egy olyan programot, amely az időzítőből futtatva akár másodpercenként megnézi, történt-e bejegyzés, és ha igen, meghívja a Gnokiit, vagy egy kicsit bonyolultabb programot készítünk, amely a fájlt folyamatosan olvassa.

Gondos figyelemmel válasszuk meg a programot, amely a figyelmeztetéseket létrehozza, hiszen előfordulhat, hogy egy túlbuzgó felhasználó POP3-kéréseit a gép DoS-támadásnak veszi, és fölöslegesen ébreszt fel minket. Arra is nagyon ügyeljünk, hogy egy folyamatos hibánál – ha már értesültünk róla (például az egyik kiszolgáló leállt) – a gép ne küldjön másodpercenként üzenetet, hanem időintervallumok legyenek. Például a `mon` nevű program lehetőséget nyújt, hogy amennyiben a gép elérhetetlen, késleltetést adjunk meg. Ha nem érhető el, küldjön SMS-t, majd ötpercenként próbálkozzon, de utána már csak akkor üzenjen SMS-sel, amikor a gép újra elérhető. Ez azért is fontos, mert ha jól állítjuk be, a gép csak abban az esetben küld értesítést, amikor felhasználói beavatkozásra van szükség, és nem ér rá reggelig. Ebben az esetben bátran úgy állíthatjuk be saját mobilunkat, hogyha a kiszolgálóra kötött telefonszámról az éjszaka kellős közepén SMS érkezik, hangosan csipogjon, egyéb esetekben csak rezegjen, hiszen csak akkor van értelme az egésznek, ha megbízhatunk benne. Pont ezért nem megbízható az elektronikus levéllel történő értesítés, hiszen az több levelezőkiszolgálón is átmegy, ahol meghamisíthatják, esetleg lehallgathatják. Igaz, már SMS-t is lehet hamísítani, de pontosan ismerni kell hozzá a küldő és fogadó számát, valamint az üzenetek felépítését. Természetesen egyik sem nyújt százszázalékos biztonságot a hamisítással szemben.
A kiszolgáló soros kapujára kötött telefonnak további nagyon hasznos felhasználása – ha beépített modemmel rendelkezik –, hogy kihasználjuk a modem adta lehetőségeket, és szükség esetén akár távolról is beléphetünk a gépre. Jókor jöhet ez a szolgáltatás, ha például:

- leálltak a távoli elérési lehetőségek: SSH, Telnet;
- egy félresikerült tűzfalbeállítással kitiltottuk magunkat;
- a kiszolgáló `init 0` szintre jutott – például egy lemezhiba miatt;
- a kiszolgáló elérhetetlen, mivel az Internet megállt.

Ebben az esetben az `mgetty` csomag lehet a segítségünkre. Ennek a remek programnak számos felhasználási módja van:

- konzolként,
- adatbehívópontként és
- faxbehívópontként használhatjuk.

Ha az első lehetőséget vesszük figyelembe, akkor kiszolgálónk egy másik gépről történő felügyeletét egy soros kábel árából megoldhatjuk. Ez csak akkor hasznos, ha

egy szolgáltatónál legalább két gépet egymás közelében helyezünk el. Ebben az esetben, ha mondjuk a tűzfalbeállításban valamit elszúrtunk, bejelentkezhetünk a másik gépre, és egy minicom (terminálprogram) segítségével átjelentkezünk a másik konzoljára.

A fenti példa megvalósításához a következőket tegyük: a */etc/inittab* állományba írjuk be a következő sort:

```
S0:12345:respawn:/sbin/mgetty
➥-s 57600 ttyS0
```

Ahol az `S0` a jelenlegi soros kapu, amire a kábelt kötjük, tehát COM1 esetén `S0`, kettő esetén `S1` és a többi, a `-s` kapcsoló a csatlakozási sebesség csúcsértékét adja meg, a `ttyS0` ugyancsak a kábel helyét határozza meg.

A */etc/mgetty/mgetty.config* állományba a következőket írjuk:

```
port ttyS0 # Itt ugyanazt a soros
➥kapuszÆmot  rjuk, mint az inittabba.
direct y
speed 57600 # Itt ugyanazt a
➥sebessØget  rjuk, mint az inittabba.
toggle-dtr n
```

Egy init q parancs után már működik is. Ellenőrizni a */var/log/mgetty/ttyS0.log* állományban tudjuk, vagy a `ps aux | grep mgetty` paranccsal. Ezek után a másik gépről egy minicom program segítségével rá tudunk jelentkezni az `mgetty`-re, ahol is egy egyszerű naplózóablakot kapunk.

Amennyiben mobilról akarunk a gépre jelentkezni, egy kábelre lesz szükségünk, amelyet ugyancsak az egyik soros kapura „akasztunk”. Az összes többi beállítás az előzőhöz hasonló, valamint az inittabban a `-n` kapcsolóra van szükségünk, mivel ez határozza meg, hogy a gép hány csörgés után vegye fel a telefont. A soros kapu sebességét 9600-ra vagy 19200-ra állítsuk (ez a szám a kapu és a telefon közti sebességet árulja el).

```
S0:12345:respawn:/sbin/mgetty
➥-s 19200 -n 2 tyS0
```

A telefonos betárcsázást az első megoldással kombinálhatjuk, ilyenkor egy másik gépről és telefonról is lehetőségünk lesz bejelentkezni.

Az egyszerű kábeles bejelentkezés akkor is hasznos lehet, ha se másik gép, se telefon nincs, amire rákössük, mivel több szolgáltatónál előfordul, hogy bár lehetőség van hozzáférni a gép konzoljához, de éppen nincs AT (régi) típusú billentyűzetük vagy átalakítójuk, csak PS/2. Ilyenkor vagy magunkkal viszünk egyet, vagy ha ott derül fény a gondra, szerezni próbálunk egyet. Az esetek többségében az AT-billentyűzet beszerzése 1–2 óránál több időt szokott igénybe venni. Helyette elővesszük soros kábelünket és egy PDA-t vagy notebookot, amelyen egy terminálprogrammal be tudunk jelentkezni a gépre – billentyűzet és monitor nélkül is.

Régen egy kiszolgálóteremben fordult elő, hogy lustaságból nem állítottam be az `mgetty`-bejelentkezést. Ment is szépen sokáig, amíg egy rendszermagfrissítés miatt a gépet újra nem kellett indítani. A gép újraindult, és ahogyan várni lehetett, egy apró gond miatt nem indult el többször. Kirohantam a helyszínre, hogy megnézzem, mi történt. A régi rendszermagot nem lehetett elindítani, mert se monitor, se régi típusú billentyűzet nem állt

rendelkezésre. Megközelítőleg kétórás futkosás után egy közelben lakó ismerőstől elkoboztam a billentyűzetét, és vakon begépeltem az `mgetty` beállítását. Ezek után már egy konzolról be tudtam jelentkezni a gépre és megoldottam a rendszermag-fordítási hibát, amely abból ált, hogy az ext2 fájlrendszer-támogatást nem fordítottam bele, így természetesen ext2-alapú rendszerem működésképtelen lett. Utolsó lépcsőként megpörgettem a Lammer-számlálót, mivel alapszintű probléma miatt fáradtam fölöslegesen. Azóta nem felejtem el a soros kapun történő kapcsolattartási lehetőséget a rendszerbetöltő programba is belerakni, hiszen ha se monitor, se billentyűzet nincs kéznél, fontos, hogy már az indításkezelő program (LILO vagy Grub) szintjén kapcsolatot tudjunk tartani (például kiválaszthatjuk, hogy melyik rendszermag töltődjön be).

Kapcsolódó címek

Lammer-számláló ➲ http://dawn.elte.hu/~raas/lc.html
Minicom ➲ http://www.clinet.fi/~walker/minicom.html
Mgetty ➲ ftp://alpha.greenie.net/pub/mgetty/source/

Ehhez a */etc/lilo.conf*-ba a következőket írjuk:

```
serial=0,19200n8 # 0 a soros kapu
➥szÆma, 19200 a kapcsolat sebessØge,
➥n8 az adatÆtvitel t pusa
```

Majd futtassuk a `lilo` parancsot. A következő indításnál a terminálprogram bekapcsolása után a PDA-n, illetve a notebookon figyelemmel kísérhetjük a rendszer indítását, és szükség esetén beavatkozhatunk.

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.

Elefánt a beágyazott porcelánok boltjában

A Microsoft célba vette következő áldozatát, a beágyazott Linuxot, ám a beágyazott Linux közössége készen áll a harcra.

A Microsoft monopólium jól ismert trükkjeinek egyikét vetette be, amikor 2001 utolsó negyedévében „Miért érdemes a Microsoft Windows XP Embedded rendszert választani, és miért nem beágyazott Linuxot?" címen hosszadalmas összehasonlítást közölt az újonnan bemutatott Windows XP Embedded és a beágyazott Linux-rendszerekről. A következőkben röviden erről lesz szó. Eszerint a beágyazott rendszert fejlesztő bármilyen eszköz tervezésének megkezdése előtt, első döntéseképpen kiválasztja az operációs rendszert. Mindegy, hogy egy egyetlen gyártóhoz kötődő alapról kíván-e áttérni kereskedelmire, vagy kereskedelmi alaprendszerek közötti váltást készít elő, a figyelembe veendő szempontok a következők:

- gyors piacra lépési lehetőség;
- biztos, bővíthető rendszermag, amely az összes tervezetnél felhasználható;
- fejlett műszaki megoldások, amelyek támogatják a legújabb lehetőségeket;
- előrelátható folyamatok a termék életciklusa alatt.
- emellett várhatóan a lehető legalacsonyabb árat próbáljuk elérni valamelyik neves gyártónál, aki képes a teljes fejlesztési folyamathoz támogatást adni.

Ezután a szóban forgó írás szerzője végiglépked a két rendszer összehasonlításán, és – amint várható volt – úgy találja, hogy az XP minden szempontból előnyösebb választás: átfogó, minden részletre kiterjedő, egyedülálló, más rendszerekkel együttműködő, világméretű, kipróbált megoldás – továbbá a Linux sem ingyenes, az OEM felhasználói szerződés viszont előnyös.

A Linux-közösség válasza

A Microsoft beágyazott Linux-ellenes kiadványának tanulmányozása után a ➲ http://LinuxDevices.com nyilvánosságra hozott egy felhívást, amelyben cselekvésre buzdítja az érdekelteket, hozzátéve: „a beágyazott Linux nem egyetlen uralkodó gyártó terméke, hanem egymással együttműködő (és közben versengő) kisebb-nagyobb vállalkozások és több ezer egyéni fejlesztő erőfeszítésének eredménye" – a felhívás mielőbbi, tömeges válaszadást sürgetett a Microsoft által intézett támadásra.

A Linux-közösség hamarosan felvette a kesztyűt, és gyors visszavágás érkezett, a Lineo és a LynuxWorks pedig egy hosszabb és részletesebb, helyenként mulatságos cáfolatot jelentetett meg. Az alábbiakban a számos hozzászólásból tálalunk néhány idézetet:

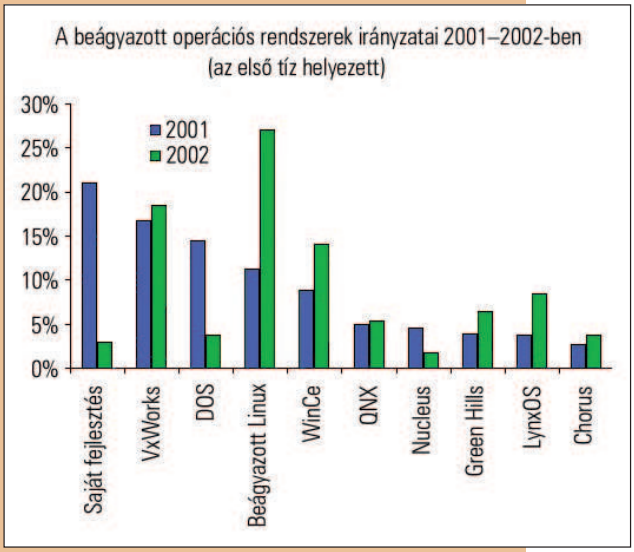
- „Nem ez az első alkalom, hogy az MS saját készítésű, a Windows által igen, de a Linux által nem támogatott szolgáltatások az összehasonlító elemzésével próbálja a Linuxot kedvezőtlen színben feltüntetni. Egy Németországban élő számára – én is innen írok – ez a fajta PR teljesen idegenül hat, hiszen hazámban összehasonlító hirdetések nincsenek engedélyezve. Az első ilyen hirdetéseket a Linuxszal kapcsolatban láttuk az újságokban, és mivel a Linux nem egy vállalat, hanem szabad termék, így nem perelhet más céget annak hirdetéseiért. Sosem értettem, miért csinálják ezt. Ha félretesszük a szabadságra és az önállóságra vonatkozó természetes igényünket, a történteket úgyis felfoghatjuk, hogy az MS elárulta nekünk, hol javíthatunk még a Linuxon, és milyen érveket tudunk felvonultatni mellette a lehetséges vásárlóknak. Minden hasonló cikk mindig ugyanazt a visszahatás váltja ki – a végén mindig az MS jár rosszul."
- „A Microsoft szokás szerint azt csinálja, amihez a legjobban ért: a fejlesztők feje felett egyenesen a vezetőkre hat, akik felülről hozzák a döntéseket. A beágyazott termékek fejlesztői szeretnek saját megoldásokat kidolgozni, hiszen az igények is sokfélék. Az eltérő eszközök másfajta területeken használhatók. A nagyokosok szeretnek mindenre jó termékeket vásárolni, amit természetesen vállalati szabványként terjesztenek el. Ők ugyanis mindenhez értenek, enélkül az MS sem lehetne egyeduralkodó a piacon. Nem tudom, hogy a beágyazott termékek piacán mekkora a befolyásuk, de a mobiltelefonok, zsebtitkárok, árusítóhelyek stb. fejlődését figyelve a nagyokos-szakmai vezető arány egyre romlik."
- „A Microsoftnak nagyon jó oka van arra, hogy aggódjon hosszútávú szerepe és jövedelme miatt, mivel befolyása egyre kisebb, kapcsolata a számítógépgyártókkal és a beágyazott termékek fejlesztőivel egyre lazább. A Microsoft az x86-32 asztali gépek piacát uralja, és itt is fog kimúlni."
- „Az, hogy a Microsoft hogyan értelmezi a *beágyazott* szót, még a vékony ügyfelek ötleténél is érdekesebb. A Microsoft Kenyérpirító egy négy Pentium IV processzort tartalmazó készülék lenne – mindkét oldalon két-két processzorral –, és a pirítós elkészítésének idejére valószínűleg parancssori módba váltana. Az, hogy egy teljes operációs rendszert, egy webböngészőt és további programokat kell belesűríteni 4 MB NVRAM-ba és 8 MB RAM-ba, kész átok számukra. Beágyazott: ez a szó számukra azt jelenti, hogy egy csontig lecsupaszított rendszer talán megelégszik 32 MB memóriával. Természetesen így is annyi erőforrást igényel, hogy azon egy átlagos e-kereskedelmi weboldal is működne, ha operációs rendszerként Linuxot választanánk. Hacsak nem elvakultan követed ezt a képet, sosem fogod megérteni, hogy honnan jönnek és hová tartanak."
- „A Microsoft pontosan azt a tényt pocskondiázza, hogy a Linux választási lehetőséget ad a fejlesztőknek: „Például legalább öt különböző ablakkezelő és legalább négy, egymással versengő webböngésző van...". A Microsoft elképzelései szerint az, hogy a fejlesztőnek ki kell választania azt az ablakkezelőt vagy webböngészőt, amely a legjobban felel meg az elvárásainak, kész katasztrófa. Az a rengeteg – gyakran nem is használt – szolgáltatás, amit például a Microsoft Internet Explorer biztosít, számos alkalmazásnál feleslegesnek bizonyulhat, a program

több megabájtos méretéről nem is szólva. Egy olyan megrögzött Microsoft számára, amely a választási lehetőséget rossz dolognak tartja, ezek a szempontok természetesen másodlagosak."

LynuxWorks

„Mi is elvégeztünk néhány vizsgálatot, és néhány megjegyzést szeretnénk fűzni az új Windows XP Embedded termék mint a beágyazott termékek piacának egyik új szereplője megjelenéséhez. Általában elmondható, hogy az operációs rendszer használhatósága a beágyazott termékek piacán korlátozott, a Linuxszal szemtől szembe állítva nem állja meg a helyét. Az XP beágyazott termékként számos hiányossággal bír. Egyes helyeken ugyan jól használható, de ezek száma – mint látható lesz – korlátozott. A Microsoft terméke továbbra is kedvezőtlen választás mind méret, mind teljesítmény tekintetében."
Összefoglalásképpen elmondható, hogy a Windows XP az alábbi okok miatt jelent rosszabb megoldást a Linuxnál, ha beágyazott rendszert kell választani:

- ***Méret*** – a Windows XP 5–15 MB memóriát foglal, míg a beágyazott Linux memóriaigénye 259 KB.
- ***Teljesítmény*** – a piac elismerte, hogy a Linux legalább olyan jó teljesítményt nyújt a kiszolgálók esetében, mint a Windows. A Windows XP ellen szóló további tényezőket figyelembe véve (méret és bonyolultság) a beágyazott alkalmazások esetében minden bizonnyal még inkább a Linux felé billen a mérleg nyelve.
- ***Kiforrottság*** – a Linux több mint negyven éve fejlődő operációsrendszer-modellt követ. Az együttműködési kérdések, a teljesítmény és az általános tervezési eljárások különösen hosszú ideje fejlődnek a gyakorlati tapasztalatok alapján. A fejlesztéseket sokféle rendszeren elvégezték.
- ***Testreszabhatóság*** – a Linux magas fokon testreszabható, valamint jellemzően memóriában szegényebb rendszerekre fejlesztették, ellentétben a Windows XP-vel, amely nagyméretű, memóriaigényes környezetből érkezett.
- ***Újítások*** – a Linux nyílt forráskódjának köszönhetően sokkal inkább az új ötletekre nyitott fejlesztők eszközévé vált, mint bármely Windows-termék.
- ***Külső támogatás*** – Linux alá többezer nyílt forrású alkalmazás, illesztőprogram és rendszermag-kiterjesztés érhető el, valamint kereskedelmi termékek is vásárolhatók. Ez az érték minden bizonnyal összevethető a Windows XP-alkalmazások számával.
- ***Hálózatkezelés*** – minden fontosabb hálózati protokoll, biztonsági szolgáltatás és kiterjesztés elérhető Linux alatt. Jelentős részük előbb futtatható Linuxon, mint egyéb operációs rendszereken.
- ***Biztonság*** – a Linux nyílt forráskódja a „több szem többet lát" elvnek köszönhetően hihetetlen mértékben fejlődik. Kedvező hatása különösen a biztonsági protokollok területén érezhető, mivel tervezésük és megvalósításuk kiválóan leírt és ismert. Remek példa erre az NSA nemrég kiadott biztonságos Linux-terjesztése.
- ***Együttműködés*** – a Linux-kiszolgálók kulcsfontosságú szerepet játszottak az Internet fejlődésében, így a különféle rendszerek példamutató mértékben képesek együttműködni egymással. Mivel a Java a Windows XP .NET keretrendszerének minden



előnyös tulajdonságával rendelkezik, a Linux lényegesen magasabb fokon képes együttműködni más rendszerekkel, mint a Windows XP.

- ***Költséghatékonyság*** – minden környezetben a fejlesztés igényli a legtöbb forrást. A teszteléseket és a telepítéseket eltérő környezetekben végző közösség nagyban hozzájárult a Linux sikeréhez. A beágyazott termékek sokszor rendkívül egyedi jellege miatt a magas fokon testreszabható Linux a költségek szempontjából különösen jó választásnak tűnik.
- ***Támogatás*** – a Microsoft egyetlen forrásból biztosít csak támogatást a termékeihez, korlátozva az egymással versengő ajánlatokat. A Linux világában ezzel szemben bőséges választék áll rendelkezésre a különféle vállalkozásokból, akik támogatást, egyéb megoldásokat és programokat kínálnak.
- ***Fejlesztői eszközök*** – míg a Windows XP csak IDE-környezetben végzett fejlesztésekre nyújt lehetőséget, addig a Linux az IDE-környezetek mellett nagy hatékonyságú, hagyományos Unix fejlesztői környezetet is biztosít.
- ***Megbízhatóság*** – a tények magukért beszélnek. A Windowst ritkán veszik számításba a küldetéskritikus alkalmazások futtatásakor, míg a Linux alkalmazása az ilyen esetekben teljesen elfogadott.

Lineo

„Megszokott, hogy a hasonló, a Microsoft részéről megjelentetett összehasonlítások következetesen a versenytársak termékeinek előnytelen vonásait próbálták meg kiemelni, miközben a Microsoft saját hiányosságairól elfeledkeztek. Ebben a dokumentumban a Microsoft szerzői látszólag elfeledkeztek róla, hogy intelligens és hozzáértő

programmérnökök, eszközgyártók és szerkesztők közösségéhez kell szólniuk, akik értenek a beágyazott programrendszerekhez. A Microsoft összehasonlításának tartalma és hangvétele alapján valószínűsíthető, hogy a versenytársak ajánlatait nem ismerő olvasókat feltételeztek…"

A Microsoft a webböngészőt és az ablakkezelést az alattuk működő operációs rendszerhez kötötte, és ezeket az összetevőket lényeges operációs rendszerösszetevőkként kezeli. A jelek szerint nem hisznek a termékek közötti különbségtételben vagy a választás lehetőségében. Ez a meggyőződés erősen áthatja a Microsoft összehasonlítását.

A Lineo viszont nem próbálja előírni az ügyfelei számára, hogy mit építsenek be a végleges termékbe. Például számos beágyazott termék esetében nincs szükség grafikus felületre vagy webböngészőre, így bármely beágyazott operációs rendszerekkel foglalkozó cégnek kár volna azt gondolnia, hogy ezek minden rendszernek elválaszthatatlan részét képezik. A Lineo Embedix beágyazott operációs rendszere ehelyett olyan alapvető szolgáltatáskészletet biztosít, amely egy teljesen működőképes operációs rendszer felállításához feltétlenül szükséges, így a fejlesztők kiválaszthatják azokat az elemeket, amelyekkel termékük kiemelhetik a versenytársaké közül. A valóság az, hogy ez a fajta rugalmasság sokkal bővebb körű újítási lehetőséget ad a fejlesztők kezébe, mint amit a Microsoft zárt forrású modelljében kaphatnak.

A Lineo teljesen nyílt forrású Linuxot kínál, a legyártott termékek után nem szed díjat. A fejlesztő alkalmazások ezreit módosíthatja szabadon, és használhatja a saját eszközén. Az *elhagyható* felhasználói szerződéspontok úgy kerülnek kialakításra, hogy a lehető legjobban megfeleljenek az ügyfél és a Lineo érdekeinek. A Windows XP Embedded esetében nincs semmiféle különbségtétel, a fizetendő jogdíj a mennyiséggel arányos, így a fejlesztők lehetősége a költségek csökkentésére meglehetősen korlátozott.

A Linux támogatása szintén versenyképes. A Linux nyílt forrású jellegének köszönhető, hogy számos különböző forrásból kapható támogatás – legyen szó akár belső, akár szerződéses, nyilvánosan elérhető (internetes) vagy kereskedelmi Linux-cégekről. Mit jelent ez a fejlesztők számára? Azt, hogy a linuxos cégek versengeni fognak az Ön cégéért, így a fejlesztők nagyobb mozgásteret kapnak, és saját piacra lépési, költség- és szolgáltatásigényüknek megfelelő ajánlatot választhatnak.

Miért támadják a beágyazott Linuxot?

Ahogy elül a csatazaj, érdemes elgondolkozni a kérdésen: miért pont most irányította a Microsoft beágyazott termékcsoportja torpedóit a beágyazott Linux ellen? Lássunk néhány indokot!

Az első ok: a Microsoft az általános beágyazott termékek piacán vesztésre áll a Linuxszal szemben. Számos piaci felmérés, köztük például az Evans Data Corporation „2001-es felmérés a beágyazott rendszerek fejlesztői között" tanulmánya következetesen azt állapítja meg, hogy a beágyazott Linux hatalmas léptekkel tört előre az elmúlt egy-két évben.

Az Evans Data Corporation legutóbbi adatai szerint – 2001-ben ötszáz fejlesztőt kérdeztek meg – a beágyazott Linux a harmadik legnépszerűbb, beágyazott termékhez választott operációs rendszer volt, egyedül a Wind River VxWorks és a Microsoft DOS-rendszere előzte meg, a Microsoft WinCE viszont mögé került (lásd az *ábrát*). Sokkal fontosabb ennél, hogy a tanulmány összefoglalása szerint a beágyazott Linuxnak komoly esélye van arra, hogy az elkövetkező egy év során megszerezze az első helyet, megelőzve ezzel mind a Wind River, mind a Microsoft termékét.

A második ok: a „PC utáni" korszak eszközeinek piacán hatalmas pénzek forognak. Ide kapcsolódik az is, hogy Microsoft azért foglalkozik egyre többet a beágyazott Linuxszal, mert a nagyobb eszközgyártók, így a Hewlett-Packard, a Sharp vagy a Motorola nemrég kezdte meg új, beágyazott Linuxot futtató termékei szállítását. Ezek között egyaránt találni kézi számítógépeket és tévés set-top boxokat – ezek olyan gyorsan növekvő piacok, amelyeken a Microsoft nyilván egyeduralomra szeretne szert tenni.

A zsebtitkárok területén, ahol a Microsoft jelentős részesedést ért el a piacvezető Palm kárára, a beágyazott Linux szintén befutó lehet. Különösen érdemes észrevenni, hogy a beágyazott Linux népszerűsége a Távol-Keleten töretlenül nő, ahol a legnagyobb tömegben gyártják a végfelhasználóknak szánt eszközöket.

A kézi számítógépek területével ellentétben a set-top szórakoztató készülékek és az autós számítógépek piacán nincs vezető szereplő. Ezeken a piacokon valószínűleg jóval nagyobb jogdíjbevételre lehet számítani, mint az asztali számítógépeken, tehát nem csoda, ha a Microsoft már a kezdetek kezdetén szeretné megfojtani a Linuxot.

Még nincs vége!

Az elkövetkező hónapokban érdemes odafigyelni ezekre a termékekre, valamint minden, a PC utáni korszakba tartozó eszközre. A beágyazott Linuxot fejlesztők – Lineo, MontaVista, Red Hat – háza tájáról érkező pletykák szerint ugyanis a szó szoros értelmében beágyazott Linuxot futtató termékek százai várnak megjelenésre, olyan termékek, amelyek nyilvánosan még nem mutathatók be, amíg gyártóik a szállításra készen nem állnak.

Mindent egybevetve: 2002 minden bizonnyal izgalmas év lesz a beágyazott Linux számára.

Kapcsolódó anyagok találhatóak a 33. CD Magazin/Beagyazott könyvtárában.

Linux Journal 2002. március, 95. szám

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

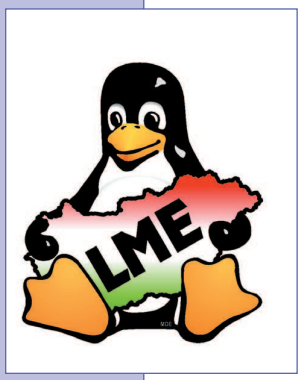
Tisztelt Linuxbarát-olvasók!

2002. március 23-án került megrendezésre a Linux-felhasználók Magyarországi Egyesületének 2002. évi rendes küldöttközgyűlése a MÁV BEIG előadótermében.
A közgyűlés elfogadta a leköszönő elnökség és az ellenőrzőbizottság beszámolóját, a pénzügyi beszámolót és a közhasznúsági jelentést.
Az elnökség 2001. évi munkáját a küldöttek több hozzászólás után elfogadták.
A 2002. évi pénzügyi tervet viszont nem szavazta meg a közgyűlés, mivel több előre tervezhető kiadás nem, vagy nem megfelelő formában szerepelt a tervben.
A terv módosítása és a következő közgyűlés elé terjesztése ezáltal az újonnan megválasztott elnökség első feladata lett.
Érdekes helyzet állt elő az elnökségi és ellenőrzőbizottsági tisztségre való jelöltállítás során, ugyanis igen nagyszámú jelölt jelentkezett, és az elnökségi feladatkörre két teljes csapat is pályázott – egy „független jelölt" társaságában.
A jelöltek bemutatkozása hosszú, de annál érdekesebb volt. Az első körben folytatott választás szoros eredményt hozott, és egy második fordulóra is szükség volt.
Végül a küldöttközgyűlés az alábbi személyeket választotta meg:

Elnökség:
Sári Gábor (saga@lme.linux.hu) elnök
Deim Ágoston (ago@lme.linux.hu) elnökhelyettes
Balázs Tibor (covek@devel.linux.hu) titkár
Magosányi Árpád (mag@lme.linux.hu)
Kovács Attila (k-atti-@lme.linux.hu)

Elllenőrzőbizottság:
Laky Norbert (lakyn@lme.linux.hu) elnök
Gibizer Tibor (gibzo@lme.linux.hu)
Tábor Viktor (Tabor.Viktor@lme.linux.hu)

Az egyesület elnöksége ezúton is köszöni azt a bizalmat, melynek eredményeképpen a 2002-es évben vezetőségi tagok lehetnek.
2001-ben az egyesület nehéz időszakot élt át, ennek egyik oka a MEH projekt által az egyesületre nehezedő felelősség terhe volt. Az előző elnökség és az egyesület tevékeny tagjai ezt a feladatot a tőlük elvárható szinten megoldották. A MEH projektről a küldöttközgyűlésen készült hangfelvétel segítségével bárki részleteket is megismerhet. A küldöttközgyűlés hangfelvételei Ogg Vorbis és MP3 formátumban is letölthetők a ➲ http://devel.linux.hu/kkgy/ címről.
Az új elnökség programjának fő üzenete az volt, hogy nem cselekszik semmi olyasmit, amit nem feltétlenül muszáj, viszont megpróbál segíteni abban, hogy a tagságtól érkező kezdeményezések sikeresen működhessenek. Ennek érdekében több olyan döntés is született, amelyek az egyesületi tagok tájékozódását segítik elő.
Az eddig zártnak tekinthető adatok a tagok számára hozzáférhetőkké váltak egy új, de zártkörű levelezőlistának köszönhetően. E listának minden tag részese lehet, amennyiben ez irányú elhatározását jelzi a listagazdának.
A nyilvános jellegű adatok vagy kérdések továbbra is a nyílt lme@lists.linux.hu levelezőlistán olvashatók.
Nagyon fontosnak tartja az elnökség, hogy az egyesület azokat a Linuxszal kapcsolatos tevékenységeket végezze, amelyeket a tagok is valóban fontosnak ítélnek meg.



Mint fentebb is írtuk, a különböző javaslatokat és gondokat az egyesületi levelezőlistákon tudjuk részletesebben megvitatni, illetve élőszóban a rendszeres szerdai összejöveteleken a Vízöntő Pinceklubban megbeszélni (címe: XIII. kerület, Váci út 50., a pontos időpontért pedig lásd az egyesület weboldalát a ➲ http://www.lme.hu/ címen).
A szerdai összejövetelek azért fontosak, mert az egyesület által végzett heti munkát itt tekintjük át. Jellemző a feladatok mennyiségére, hogy az eszmecserét igénylő pontok csupán mintegy negyedét sikerült az első küldöttközgyűlés utáni szerdán megbeszélni.
Az LME-tagsággal kapcsolatos kedvezménybővülések kapcsán a következő terveink fogalmazódtak meg:

- Minden tag évente egyszer ingyenesen jogosult legyen egy általa kiválasztott Linux-terjesztés CD másolatára.
- Terveink között szerepel a Linuxvilág magazinnal való szorosabb együttműködés és a SoftwareStation cégnél kedvezmények kiharcolása. Ezek pontos tartalma és mértéke azonban még egyeztetés alatt áll.

A küldöttközgyűlés elfogadta az egyesület Fegyelmi szabályzatát, továbbá sikeres aláírásgyűjtés történt annak érdekében, hogy az Alapszabály módosításának céljából az elnökség rendkívüli küldöttközgyűlést hívjon össze legkésőbb 2002. május 15-ig.
Ennek értelmében a rendkívüli küldöttközgyűlés időpontja: 2002. május 4., szombat 10 óra, helye a MÁV BEIG nagyterme a Nyugati téren.
Napirendi pontok:

1. Az egyesület Alapszabályának módosítása.
2. A 2002-es költségvetés vitája és a tervezet elfogadása.
3. A 2001-es módosított mérleg elfogadása.
4. Az LME jövője (vita).

A rendkívüli küldöttközgyűlést megelőző küldöttválasztó gyűlést előtte két héttel, szerdán a szokásos egyesületi találkozón, a Vízöntő Pinceklubban tartjuk.
Mivel a közgyűlés nem zártkörű, minél több LME-tag és vendég megjelenésére számítunk, különösen „Az LME jövője" címmel tervezett vitára.

Üdvözlettel: az LME elnöksége

Nyílt forráskódon alapuló rádió

Doc Searls Wild Bill Goldsmith-szel, a KPIG és a Radio Paradise tulajdonosával beszélget a nyílt forráskódon alapuló rádió fényes jövőjéről.

A KPIG kivételes állomás. Olyan színvonalas kereskedelmi rádióadó, amelyet teljes egészében saját lemezlovasai programoznak. Komoly rádióállomás rendkívüli humorérzékkel megáldva. A körülötte formálódó közösség számára legalább olyan állandó, mint egy szobor a bíróság épülete előtt – ez viszont sokkal érdekesebb. Egyesek szerint az adó stílusa a „mutáns cowboy rock and roll" kifejezéssel írható le legjobban, de nehéz bármilyen címkét ragasztani egy olyan állomásra, amely igazibb, viccesebb és jobb, mint (szerény véleményem szerint) az ország összes zenét sugárzó adója együttvéve. Az állomás gyökerei a legendás KFAT-ig nyúlnak vissza, sőt azon túl a KRAB csillagködig, amely a nem kereskedelmi rádióállomások lazán összefüggő hálózata volt, és amelyből a hatvanas években a „közösségi rádió" és a „független rádió" típusai fejlődtek ki. Ennek történelme ugyanolyan régi és gazdag, mint a Unixé.
A hasonlóság azonban nem merül ki ennyiben.
A KPIG és hívei, akik eredeti hangokkal, programozási ötletekkel és zenei válogatásokkal járulnak hozzá, komolyan veszik a szabad és nyílt szavakat. A KPIG egészen véletlenül a kaliforniai Freedom (Szabadság) városban székel, és az egyik legrosszabbul vehető jelet sugározza a Salinas-Monterey-Santa Cruz térségben. Ha Monterey-öblöt körülvevő hegyek medencéjében tartózkodsz, jó eséllyel foghatod. Ha máshol vagy, el kell látogatnod a ➲ http://www.kpig.com webhelyre, ahol sokféle folyam közül választhatsz, többek közt van itt egy 128 Kb-es MP3-jelforrás, amelynél szebben hangzó jelet ritkán adnak ki magukból a hangszóróid.
A KPIG volt az első kereskedelmi rádióadó, amely kilépett a Webre, és azóta is úttörő szerepet játszik ott. Amikor a múlt tavasszal az AFTRA kitalálta, hogy háromszoros árat kér a hálózaton sugárzott reklámokért, sok kereskedelmi adó befejezte az internetes adást. A KPIG jobbat talált ki: amikor reklám következik, a webes folyam hallgatói kellemes helyettesítő műsort hallgatnak. A rádió munkatársait és hallgatóit kivéve mindenkit megdöbbent, hogy rossz vételi lehetőségei ellenére a KPIG a helyi Arbitron-értékelésekben állandóan a csúcs közelében tartózkodik. Ez mindenképpen siker. Ráadásul egy csomó nyílt forráskódon alapuló módszert alkalmaznak, amelyek forradalmasíthatják a rádiózást, ha a megfelelő számítógépes és rádiós megszállottak együtt dolgoznak. Ez az, ami miatt a cikk megszületett.
A KPIG főszakembere, *„Wild Bill" Goldsmith* régi KPIG-es egészen a KFAT-os időkig visszamenően. Jelenleg a kaliforniai Paradise-ban lakik, ahol a KPIG mellett saját projektjét, a Radio Paradise-t is kitartóan fejleszti. Miután gyakorlatilag egy időben több levelet kaptam barátaimtól Seattle-ből, New Yorkból és Észak-Karolinából, elhatároztam, hogy felkeresem Billt. A leveleket rádiós öreg rókák írták, akik ugyan már más területeken dolgoznak, de rögtön megérzik az igazi tehetséget, ha meghallják.
Ahogy ránéztem a KPIG-re és a Radio Paradise-ra, rögtön tudtam, hogy többről van szó, mint jó rádióműsorról. Műszakilag mindkettő könnyen testreszabható megoldás nyílt forráskódú programok és általános számítástechnikai eszközökből felépítve. Ami különleges benne, az, hogy Bill olyasmit rakott össze, ami egységbe foglalja a zenekönyvtárat, a webhelyet, a teljes hangrendszert, a könyvelést, az üzemeltetést és a parancsfájlokból vezérelhető árukapcsolási szolgáltatásokat, amelyekkel esetlegesen pénzt hozó kapcsolatokat (boltok, művészek vagy bármi más) lehet reklámozni. Mindezt oly módon alakította ki, hogy a felhasználó szabhatja meg az élő működtetés és a gépesítés arányát.
Ez tetszik nekem. Gyermekkorom óta nagy rádiórajongó vagyok, rádióamatőr is voltam (a mai napig az egyetlen kód, amit ismerek, a Morse). A becenevem, a Doc azokból az időkből származik, amikor ezen a néven jelentkeztem be a KFAT-szerű WDBS rádió műsoraiba a hetvenes évek Észak-Carolinájában. A mai kereskedelmi rádiózás csak nyomokban emlékeztet arra, amit a rádiózás aranykorában szerettünk. A rádiózás felélesztése is közös szenvedélyünk *Phil Hughes*-szel, aki a Linux Journal kiadója és szintén régi rádiós. Szívesen emlegetjük a Seattle-beli KRAB hőskorát.
Írtam Billnek, és megkérdeztem, hogy pontosan mit hozott létre a két rádióadónál. Ezt válaszolta:
„Az egész egy programcsomagon alapul, ez választja ki és ütemezi a zenét, a zeneszámokat bárhová elküldi a hálózaton, fogadja és rendszerezi a hallgatóknak a lejátszott számokkal kapcsolatos visszajelzéseit.
A programokat tekintve minden százszázalékosan nyílt forrású: Linux, PHP, Perl, Postgres és Icecast.
Meg vagyok győződve róla, hogy ami a ➲ http://www.radioparadise.com címen látható, az a rádiózás jövőjét vagy csak egyszerűen a minőségi rádiózást jelenti, azaz interaktív, művészi értelemben szigorúan ellenőrzött (minden okkal jelenik meg, semmi sem véletlen), teljesen független a rádió- és zeneipar befolyásától (amennyire képes vagyok rá), és elsősorban a hallgatók önkéntes támogatására épít.
Ez nem az a dolog, ami az embert gazdaggá teszi, viszont bárki, aki elég tehetséges, kényelmesen megélhet belőle anélkül, hogy feladná a függetlenségét – mindig is erre vágytam."
Javasoltam Billnek, hogy bővebben is beszélgessünk el, amivel egyetértett. Október végén került sor a találkozásra, miután a KPIG legújabb honlapjának elkészítését befejezte.

Doc: *Tetszik az új honlap. A Webcast-hivatkozások és a pillanatnyi számlista rögtön elöl van. Tetszik a naplószerű dolog is az oldal közepén. Nagyon élővé teszi a helyet. Igazi rádió.*
Bill: Hamarosan további naplószerű dolgok kerülnek fel, miután az adó munkatársai birtokba veszik.

Doc: *Régóta gondolkodom azon, hogy a Linux és a nyílt forráskód fejlesztőinek közössége tehetne valamit a kereskedelmi rádiók helyzetének megingatása érdekében. Úgy látom te vagy az, aki e téren az úttörő munkát végzi. Szerinted érdemes a te utadat követni?*

Bill: Egyértelműen igen. Amit a két rádióállomásnál véghezvittem, fokozatosan teljes nyílt forráskódú programcsomaggá fejlődik, amellyel rádióállomást és hozzá szervesen kapcsolódó honlapot lehet tervezni és karbantartani. Gyakorlatilag minden, amit a KPIG.com-on látsz, nyílt forráskódú programok segítségével készült. Zárt forrású program sehol nem játszott szerepet.

Doc: *Mesélj arról, hogy mit láthatok, ha meglátogatom a KPIG honlapját. Hogyan épül fel a tartalom, és milyen kiszolgáló áll mögötte?*

Bill: Az oldalakat gyakorlatilag egy linuxos gépen futó Apache építi fel. Bizonyos tartalmak egy PostgreSQL-adatbázisból dinamikusan épülnek be. Sok kis apró dolgot – HTML-kódrészleteket – parancsfájlok építenek be az oldalakba. Ilyen például a *Now Playing* mező. Az itt látható tartalom kódját az a rendszer írja ki, amelyet a lemezlovasok használnak a zeneszámok kiválasztására és lejátszására, és amely egy másik Linux-kiszolgálón futó hasonló program. A lemezlovas egy nem nyilvános weboldalt használ, ezen keresztül szabályozza, hogy mi kerüljön a kimenő folyamba.

Doc: *Más szavakkal a lemezlovas egy belső weboldal felett rendelkezik, amely vezérlőpultként működik. Ez valamiféle párbeszédre ad lehetőséget.*

Bill: Igen, ennek segítségével választják ki és ütemezik a zeneszámokat. A képernyő teljesen olyan, mint bármelyik más gépesített rádió esetében. Nem úgy néz ki, mint egy weboldal. A legtöbben nem is tudják, hogy weboldallal van dolguk, mert teljes képernyős üzemmódban fut. Ezt a képernyőt használják a hanglejátszásért felelős kiszolgáló vezérlésére, így lehet előre programozni. Tetszőleges mélységig előzetesen megvizsgálható, hogyan is hangzik a műsor, utána elmehetnek. Megtehetnénk – bár ez nem célunk –, hogy ilyen módon teljesen gépesített rádióadót készítsünk – a lehetőség adott. Ezen a módon működik a Radio Paradise és a Smooth Jazz (➲ http://www.smoothjazz.com).

Doc: *A Smooth Jazz is a tiéd?*

Bill: A műszaki háttér igen, azaz a gépesítés és a honlapot felépítő rész.

Doc: *Ez a háttér felelős mindenhol a parancsfájlok elkészítéséért?*

Bill: Amikor a lemezlovas megnyom egy gombot, ami elindít valamit, vagy a rendszer saját maga indít el valamit, mert a lemezlovas erre utasította, akkor nemcsak a zene lejátszása indul el, hanem a Now Playing oldal is frissül. A *Now Playing* mezőben szereplő dal lecsúszik az alatta levő lista tetejére. Összesen négy dalcímet láthatunk, az éppen sugárzottat és a legutóbbi hármat. Mindegyik dalnak saját oldala van. A rendszer azt az oldalt is megírja, amelyet akkor látsz, amikor a zeneszámok listájára kattintasz, ez az elmúlt hat órában lejátszott számokat sorolja fel. Ezeket valójában egy másik gépen állítjuk össze, azután FTP-vel átvisszük a webkiszolgálóra, és beépítjük a főoldalba.

Doc: *Az összes zenének a merevlemezen kell lennie valahol?*

Bill: Nem, a legtöbb ugyan ott van, de akadnak kivételek. Ha a lemezlovas a KPIG-nél olyan számot akar lejátszani, ami nincs az adatbázisban, beírhatja a címét, hogy a rendszer megjelenítse.

Doc: *Minden KPIG-en hallható szám MP3?*

Bill: Jó minőségű MP3-ak. A legtöbb 256 Kb-es, de van néhány 192-es is. Az összeset egy Linuxot futtató gép játssza le egy 90 dolláros hangkártya segítségével. Az egész rendszer kijött 600 dollárból. Nagyon olcsó, ezért lehet rá vállalkozást építeni. Ez azonban különleges vállalkozás: a rádiózás következő nemzedéke.

Doc: *Meglepődve tapasztaltam, milyen kevéssé használja a közönség és a nem kereskedelmi csatornák az MP3-at mint a zene kódolásának és weben keresztüli átvitelének eszközét. A legtöbben Realt használnak, néhányan a Windows Médialejátszóját. Első ránézésre ez drágább megoldásnak tűnik. Azért van így, mert nem ismernek jobbat?*

Bill: Igen.

Doc: *Tehát csak a márkanév miatt.*

Bill: Talán 80–90 százalékban a márkanév miatt. Egy ingyenes Real kiszolgáló felállítása egyszerű dolog, ha valaki nem akar egyszerre húsz embernél többet kiszolgálni. De semmi egyéb. Rendben, legyen 95 százalék a márkanév szerepe. És a tehetetlenség. Sokan akkor kezdtek a Reallal foglalkozni, amikor még az volt a legjobb megoldás, és most nincs okuk váltani. De haver, ha nekem fizetnem kellene a Real kiszolgáló használatának jogáért, azonnal váltanék az MP3-ra, a QuickTime-ra vagy bármilyen más ingyenes megoldásra. Még a használt sávszélesség költségét is csaknem lehetetlen visszanyerni.

Doc: *Van-e a Realnak bármi előnye bizonyos sebességeknél?*

Bill: A Real legújabb kodekjei sokkal inkább ki vannak hegyezve az alacsonyabb bitsebességekre, mint az MP3. Tényleg jobb a minőségük a 32 k alatti bitsebességeknél. Ez az a bizonyos öt százalék – de a különbség nem jelentős.

Doc: *Elég érett a rendszered ahhoz, hogy dobozos terméket hozz belőle létre?*

Bill: Szeretném, ha a rendszer elterjedne, de nem gondolom, hogy dobozos termékként kellene árulnom.

Inkább egy nyílt forrású projekt magja szeretnék lenni. Nem akarok központi szerepet játszani benne, csak használni. Elsősorban ezért építettem. Foglalkozásomra nézve nem vagyok programozó. Eleget tanultam, hogy felépíthessem a rendszert – ez elég nagy tudásanyagot jelent, és örülök, hogy mindezt tudom –, de egy kissé nagy ahhoz, hogy egyedül kezelni tudjam. Másoknak is be kell szállniuk a fejlesztésbe.

Doc: *Gondoltál már arra, hogy felteszed a SourceForge-ra?*

Bill: Számos lehetőséget tartok számon. Jelenleg azt szeretném, ha egy kívülről érkezett ember megvizsgálná, felmérné, hogy meddig jutottunk. Ez meglehetősen nagy projekt. Magába foglalja a műsorszórási tevékenység mindhárom különböző tevékenységcsoportját. Nem olyan eszköz, amit egy hobbiból rádiót működtető ember használhat, csak olyanoknak való, akik komolyan kimagasló minőségű rádiót szeretnének működtetni 21. századi eszközök felhasználásával.

Doc: *Hány ember tudja, hogy mit jelent a kimagasló minőség a rádiózásban?*

Bill: A szakmában alig valaki. Az összes olyan ember, aki nyitott lenne mondanivalóm befogadására, már rég elhagyta a szakmát. A KPIG kivétel. Ha nem kötődnék a KPIG-hez, semmi közöm nem lenne a kereskedelmi rádiózáshoz. Csak a saját dolgaimmal törődnék.

Doc: *A KPIG sok szempontból formabontó.*

Bill: Rengeteg munkatársunk van a közepes méretű adók viszonylatában. Élő lemezlovasok adnak egész nap. Senki más nem tudja ezt. Még San Franciscóban sem látni ilyet, már nem.

Doc: Sokféle folyam közül lehet választani, de gyakorlatilag mind MP3. Ez azért van, mert minden zeneszerető számítógép-tulajdonosnak van valamilyen MP3-lejátszója?

Bill: Nem annyira elterjedt, mint a Real vagy a Windows Médialejátszó. De a Real az MP3-folyamokat is szépen lejátssza. Mindenesetre elég nagy azoknak az embereknek a száma, akik rendelkeznek a Real, a Winamp vagy az iTunes programok valamelyikével.

Doc: *Nézegettük az Apple OS X-et, amely a Darwinra épül, a BSD egy változatára. A vele adott MP3-lejátszó az iTunes, amely rádióvevőt is tartalmaz, és jól bemutatja, milyen nehéz csoportosítani a rádióadókat. Az adatokat az úgynevezett Kerbango-adatbázisból veszi (a Kerbango egy nemrégiben felvásárolt linuxos rádiótársaság volt, a 3Com vette meg és az év elején becsukta).*

Bill: Az Apple egy volt kerbangós barátomat fogadta fel, hogy részmunkaidőben karbantartsa az iTunes adatbázisát. Ő a KPIG-et az „Americana" csoportba sorolta, és a Radio Paradise az „Alt/Modern Rock" csoportba került. Valaha az „Americana" csoportba tartozott, de megkértem, hogy változtassa meg.

Doc: *Ki fog komolyan rádiózni, van-e ebben üzlet?*

Bill: Úgy vélem, egy sokoldalú ember vagy egy kis csoport számára ez jó üzletet jelenthet. Mostanság, ha minden jól megy, egy internetes rádió akár két-három teljes munkaidőben foglalkoztatott embert is el tud tartani.

Doc: *Miből jön a bevétel?*

Bill: Egyelőre az összes sikeres próbálkozás, amelyről tudok, a hallgatók adományaiból tartja fenn magát.

Doc: *Ez a közösségi rádió modellje.*

Bill: Igen, és ezt sokkal jobban alkalmazható a helyzetre, mint a kereskedelmi rádió modellje, amelyet ahányan csak megpróbáltak az internetes rádiózásban hasznosítani, annyian belebuktak. Amíg létezik reklámmentes versenytárs, nincs ok rá, hogy valaki a reklámokkal teletűzdelt adást hallgassa, hacsak az nem sokkal jobb.

Doc: *Úgy látom, a legjobb internetes adók, akár a legjobb hagyományos adók – kevés akad – olyanoknak játszanak, akik ismerik az anyagukat, vagyis lejátsszák neked a lemezgyűjteményüket.*

Bill: Ez a jó öreg földalatti URH-rádiózás.

Doc: *Nem sok ember vette észre a kereskedelmi rádiózás egyik alapvető törvényszerűségét, hogy a fizető ügyfelek és a fogyasztók nem azonos csoportba tartoznak. Ez a hátrány a nem kereskedelmi adóknál nem jelentkezik. A hallgatók egyben fizető ügyfelek is. És a hálóra könnyű feltenni egy perselyt, így az emberek adhatnak valamit a műsorszórónak a szolgáltatásért cserébe. Ezt egy autórádióval nem lehet megtenni.*

Bill: Ez a fizetési rendszer figyelemreméltóan jól működik a Radio Paradise-nál.

Doc: *Hogyan?*

Bill: Íme, a saját tapasztalatom. Nagyjából egy évvel ezelőtt időnként leveleket kaptam egyes emberektől, akik ilyesmiket írtak: „Hé, szívesen fizetnék ezért – elfogadsz adományokat?" Mindig azt válaszoltam: „Nem, nem. Tartsd meg a pénzedet." Ez egészen addig így ment, amíg le nem ültünk a feleségemmel, és el nem kezdtük kiszámolni, hogy miből fizetjük ki ezt az egészet. Ezt mondtuk: „Emlékszel, az emberek már ajánlottak fel pénzt. Vajon mi lenne, ha lehetővé tennénk az adakozást?" Feltettünk egy hivatkozást, és a folyamba egy kis hirdetést a lehetőségről, nem túl tolakodóan, mindössze egyszer vagy kétszer naponta. Egy kis cikk került erről a honlap hírei közé, és egy hivatkozás a lap aljára. Az első hónapban 2000 dollár adományt kaptunk. Ez pontosan 2000-rel több volt, mint az előző havi bevétel. Mostanra a 3000-et tűztük ki célul, és 3000–3500 között kapunk minden hónapban, meg még 300–400-at az üzleti kapcsolatokon keresztül, mint amilyen a CDNOW.

Doc: *Mióta?*

Bill: Május óta.

Doc: *És ebből megéltek?*

Bill: Ez elég ahhoz, hogy egy pár tanácsadói projektemet abbahagyhassam.

Doc: *Nem rossz kezdet.*

Bill: Igen. A csatorna többé-kevésbé önfenntartó.

Doc: *Fognak nőni a költségeid?*

Bill: Jelenleg a sávszélesség ingyen van. Amikor ez az időszak véget ér, a költségek megkétszereződnek, de azt hiszem, a bevételek is. Egyre többen hallgatnak minket.

Doc: *Gyűjtesz adatokat a hallgatókról?*

Bill: Most kezdjük el a KPIG-nél, teljesen önkéntes alapon. A hallgatók több mint nyolcvan százaléka a munka-

helyén hallgat minket. Sokan közülük otthon dolgoznak kábel- vagy DSL-hozzáférésen keresztül. A munkahelyeken szintén igénylik az emberek a háttérzenét, és a számítógép kéznél van. Főként az Egyesült Államokból hallgatnak, körülbelül nyolcvan százalék. Európából is elég sokan. Minden korosztály képviselteti magát, de főként a 30–40 évesek.

Doc: *Mit gondolsz a Live365-ről, amely 35 000 MP3-folyamot szolgáltat? Nekem három felületen is rendszerösszeomlást okozott ez az óriási mutánslény.*

Bill: Megkövetelik, hogy az ő felbukkanó kis lejátszójukat használd. Egy kis javascriptes valamit.

Doc: *De sikerült-e üzleti sikert elérniük az egyedi MP3-folyamok összegyűjtésével?*

Bill: Szerintem egyértelműen van piaca az internetes rádiók alá hátteret adó cégeknek. A Live365 lényegében ezzel foglalkozik. Bevételüket a hirdetésekre alapozzák, amit én helytelenítek. Úgy tűnik, hogy nagyjából véletlenszerűen hirdetéseket illesztenek be a folyamba. Eredeti üzleti modelljük – mint mindenki másnak is – az volt, hogy a honlapjukon elhelyezett reklámcsíkokból fognak meggazdagodni, amikor a honlapnak már kismillió látogatója van. Mindannyian tudjuk, milyen jól működött ez.

Doc: *Mindannyian sok bajt kerülhettünk volna el, ha használni tudtuk volna az elnémító gombot a távirányítón.*

Bill: (nevet) A Live365 tényleg fizettet az új emberekkel a folyam befogadásáért. De több ezret ingyen fogadtak be. A Radio Paradise is átcsúszott a kerítés alatt, mi is ott vagyunk. Körülbelül további kétszáz hallgatót tudunk ezúton szerezni.

Doc: *Hány folyam fut egyszerre?*

Bill: A legtöbb 800–900 körül volt. A KPIG is nagyjából hasonló. Talán kicsit több.

Doc: *Milyen bitsebességen hallgat titeket a legtöbb ember?*

Bill: A Radio Paradise-t az emberek háromnegyede 128 k-n hallgatja.

Doc: *Gondolom, főként a kábel és a DSL miatt.*

Bill: Igen, a 128 k nagyon szépen jön mind kábelen, mind DSL-en.

Doc: *Várható-e valamilyen alapvető hatékonysági gond, ha a hallgatók száma emelkedik?*

Bill: Természetesen, az egész elképzelhetetlenül nem hatékony, de szerintem ez nem fog változni. Ráadásul ma már a sávszélesség és az eszközök is annyira olcsók, hogy nem érné meg az egész Internetet újra feltalálni, hogy hatékonyabb lehessen.

Doc: *Aggódsz-e a Disney, az RIAA és társaik javasolta új törvények miatt?*

Bill: Igen. Ha elérik, amit akarnak, a hozzám hasonló emberek tönkremennek. Nem lenne rá lehetőség, hogy költséghatékonyan végezzem a munkámat.

Doc: *Vég nélkül kellene foglalkozni a digitális jogokkal.*

Bill: Nem tudom elhinni, hogy működni fog, hogy keresztül tudják erőszakolni a dolgot. Egyszerűen túl nagy az ellenállás. A fogyasztók oldaláról a legnagyobb az elutasítás. Az MP3-fájlok szabad cserélgetésével megízlelték, hogy milyennek kell lennie a világnak, és ezt a szabadságot semmiért nem akarják feláldozni.

Doc: *Mégis, amikor a Napster meghalt, nem sokan sírtak. Az emberek továbbléptek.*

Bill: Igen, mert sok más lehetőség is van. Ott a Gnutella, amelynek jó Linux-ügyfele van, maces is. Igaz, egyik sem olyan jó, mint a Napster volt fénykorában.

Doc: *Szóval nem aggódsz.*

Bill: Nem, kifejezetten derűlátó vagyok.

Linux Journal 2002. január, 93. szám

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

Talán maga a silányság is szabadalmazva van már?

Vajon mi a közös a háromdimenziós kördiagramban, a tankönyvekben, a génprofilozásban és a hirdetések hatékonyságmérésében? Például az, hogy *Gregory Aharonian*, a Bustpatents.com atyja ezt a négyet tartja minden idők legalkalmatlanabb szabadalmainak. Ezt a listát a Scientific American kérésére állította össze, és a természettudományi folyóirat közölte is az eredményt. Mind a négy átment (vagy inkább megbukott? – nézőpont kérdése) a Patent and Trademark Office (az amerikai Szabadalmi és Védjegyhivatal) nyilvánvalósági próbáján.

Szintén Aharoniantól tudjuk, hogy a közelmúltban különböző okok miatt érvénytelenített szabadalmak között találhatók az alábbiak:

- állatfej átitatására szolgáló eszköz;
- összetett képek videofeldolgozása;
- üzenetrögzítő telefonrendszerhez;
- szabadforgalmú értékpapírral elkövetett csalást észlelő és feldolgozó eszköz.

Ki tudja, ez utóbbi talán nem vette észre saját magát.

Linux Journal 2002. április, 96. szám

Doc Searls

Linuxot az oktatásba!

Ha elég korán kezdjük a Linuxszal való ismerkedést, több és jobb álláslehetőség vár ránk az egész világon.
A nepáli Bachhauliban lévő Shree Bachhauli Gimnáziumban negyvenhárom válogatott diák tanulja a programozás mesterségét. Így talán távol maradnak az illegális gyermekmunkától és könnyebben bekerülhetnek a felsőoktatásba, majd miután végeztek, rendes álláshoz juthatnak. Az iskolában tizennégy tanár jut hatszáz diákra, de a programozásórákat kis létszámú csoportokban tartják német és svájci önkéntesek segítségével, akik a Ganesha's Project nevű vállalkozásnak dolgoznak. Adománygépeket használnak nyílt forrású alkalmazásokkal, Linuxszal. Ennek segítségével az amúgy is elszegényedett oktatási rendszer jelentősen csökkentette a költségeit.

A Ganesha's Project legfőbb célja, hogy a nepáli gyerekeknek legyen egy, a gyermekmunkán kívüli választásuk – fogalmazott *Kirstin Boettcher* német grafikusnő, aki szintén a vállalkozásnak dolgozik. Hozzátette, hogy Nepálban – mint általában a harmadik világbeli országokban – nem használják ki a fejlődés adta lehetőségeket. „Tevékenységünk lényege, hogy számítógépes hátteret nyújtsunk a szegényeknek, amely lehetővé teszi számukra a tanulást, a művelődést."
A Ganesha's Project csak egy példa arra, hogy a pénz- és oktatóhiánnyal küszködő iskolák világszerte a Linuxot választják azért, hogy olcsó, de annál hasznosabb legyen a tananyag. Az Apple 1980-as akciójához hasonlóan, amikor az iskoláknak Apple II gépeket adományoztak, a Linux-forgalmazók és az önkéntes szervezetek újabb és újabb felhasználókat nyernek meg tevékenységükkel. A program sikerének kulcsa a nyílt forrású programok használata. *Peter Farina* számítástechnikát tanít az illinoisbeli Lombard Montini katolikus középiskolában. A Chicago külvárosában található iskolában nyílt forrású rendszereket használnak. Farina szerint amint a gyerekek az alapvető Linux-parancsokat megtanulják és megismerkednek a fájlrendszerrel, nagyon érdekli őket a téma. Idővel rájönnek, hogy egy sor szabad programot találnak, ami egészen más, mintha kalózmásolatot szerezznének egymástól.
Farina iskolája részt vesz a SuSE „Ingyenes Linuxot az amerikai iskoláknak" programjában. A vállalkozás Államok-szerte több mint kétezer példányt ajándékozott az iskoláknak.
„Ez a lehetőség megmutatja a Nyílt Forráskód irányvonal előnyeit, és mindenkit, azaz diákot, tanárt, rendszergazdát és informatikust egyaránt ráébreszt arra, hogy nem kell súlyos összegeket költeni operációs rendszerekre és a frissítésükre" – mondja *Dirk Hohndel*, a „SuSE Amerikában" vállalkozás elnöke.
A Ganesha's Project önkénteseihez hasonlóan Farina is egy szegény iskolában tanít, ahol a költségek csökkentése érdekében kezdett el Linuxot használni. „Eljutottunk odáig, hogy bővíteni szerettük volna a hálózatot" – meséli. „A fizikai megvalósítás a helyén van, de az engedélyek (license) igen drágák, ami megbénít minket. Valamilyen megoldást keresek arra, hogy a szolgáltatás színvonalát anélkül javíthassuk, hogy egy vagyont fizessünk ki a semmiért".
Farina tanítási segédeszközként használja a Linuxot, sőt a diákjainak azt is megtanítja, hogyan építsenek egy kisebb hálózatot. Úgy véli, a legnagyobb akadályt az képezi, hogy meggyőzze a kollégákat egy másik operációs rendszer használatának megtanulásáról. A legtöbbjük épphogy csak használni tudja a Microsoft Windows-termékeket, és sok fejfájást okozna számára, ha hirtelen rájuk erőltetné a Linux használatát.
A New York-i Beacon School tanulói, a szülők és a tanárok naponta anélkül használják a Linuxot, hogy észrevennék. *Shantanu Saha*, a New York-i Oktatási Központ technikai igazgatóhelyettese hangsúlyozta, hogy a Beacon School hálózata teljesen Linux-alapú. Az iskola weboldala, amelyet egy Red Hat-kiszolgáló biztosít, napi híreket és közérdekű hirdetéseket tartalmaz. Ezenkívül egy szülő és tanár közötti közlési rendszer fut rajta, amelynek hiányát nehéz volna pótolni.
„Az oldalakat főként a diákok fejlesztik, és minden alkalommal bővítik" – jegyzi meg Saha. „Idén az érdeklődő iskoláknak egy Linuxot bemutató, népszerűsítő foglalkozást vezettem."
Saha úgy tervezi, hogy a térség iskoláiba hetven hálózati kiszolgálót telepítenek, melyek a legfőbb levelezési és webes feladatokat végzik el. „Úgy vélem, a követendő példa a következő: telepítsünk Linux-rendszerű hálózati kiszolgálókat az iskolákba, tanítsuk meg a tanárokat és a rendszergazdákat az alapvető üzemeltetési feladatokra, és irányítsuk ezeket a kiszolgálókat távolról anélkül, hogy rendszergazdát küldenénk ki az iskolákhoz. Szeretném más iskolákban is megismételni a sikereket" – tette hozzá, és elmondta, hogy a Beacon School New Yorkban a legfejlettebb számítógépes háttérrel rendelkező iskolák egyikéve vált.
Ehhez hasonló sikereket szeretnének világszerte a különböző önkéntes szervezetek megvalósítani nyílt forrású oktatást segítő vállalkozásaikkal. Az egyik ezek közül a *Paul Nelson* és *Eric Harrison* által vezetett vállalkozás, mely a Multnomah megyei oktatási körzetben tevékenykedik az Oregonbeli Portland környékén. Egy úgynevezett K-12 Linux Projectet hoztak létre, mely arra hivatott, hogy a segítségével újra életre keltsék az iskolák elöregedett gépparkjait – természetesen Linux segítségével. „Az iskolák régi számítógépeket kapnak" – mondja Nelson, aki az utóbbi húsz esztendőben a Riverdale-i körzetben oktatóként, később pedig rendszergazdaként dolgozott. Ezek a gépek üres merevlemezzel érkeznek, tehát nincs rajtuk operációs rendszer. Gépenként mintegy száz dol-

lárba kerülne, hogy Microsoft-termék fusson rajtuk.
A Linux használatával nincs szükség új és gyors gépekre. A K-12 Linux Project egy központi kiszolgálót használ, ez biztosítja a Gnome-ot az iskola gépei számára. Nelson mostanában több száz számítógépet felügyel abban a két körzetben, ahol dolgozik. Azt állítja, a vállalkozás sikeres, a gyerekeknek nincs szükségük külön képzésre, csak kézbe veszik az egeret és kattintgatnak. Napok kérdése és szakértőkké válnak. A gyerekek már csak így tanulnak. „Úgy véljük, az operációs rendszer segítse és ne hátráltassa ezt a folyamatot. A K-12 Linux Project lavinaszerűen terjed" – mondja Nelson. „Londoni és belize-i iskolákat is bevontunk a programba, sőt malajziai és fülöp-szigeteki iskolák is érdeklődnek. A Linux óriási lehetőségeket rejt magában."
Nelson véleménye szerint a Nyílt Forráskód irányvonal a legjobb megoldás az iskolák számára, hogy a nemzetközi piacon versenyképesek maradjanak. „A rendszer ingyenes, a kereslet óriási – az iskolák viszont szegények" – teszi hozzá. „A vállalkozás segítségével három gyerekre jut egy gép. A gépeket hálózaton keresztül tudjuk felügyelni, legyenek azok az épület bármelyik részén – errre eddig soha nem volt lehetőség."
Az iskolák számára hasonló szolgáltatást biztosító társaságok vagy önkéntes szervezetek nem csak termékeiket népszerűsítik – igen, a piacgazdaságban a kapitalista ösztön természetszerűleg jelen van.
A MandrakeSoft alkatrész-forgalmazókkal együttműködve biztosít számítógépes hátteret a szegényebb iskoláknak, Los Angelestől kezdve Kanadán át Mexikóig.
„Az elkövetkezendő két évben elérjük, hogy a mexikói iskolák nagy része linuxos gépet futtat majd" – mondja *Daniel Morales*, a MandrakeSoft amerikai tagozatának elnökhelyettese. „Programcsomagokat ajándékozunk és néhol kiszolgálókat is annak érdekében, hogy a rendszer teljes legyen. Társaságunk hangsúlyozza az oktatás jelentőségét, hogy új tehetségeket képezhessenek ki a Nyílt Forráskód irányvonal számára."
Végül megállapíthatjuk, hogy a Linux beférkőzik az iskolákba, és az Apple-höz hasonló elismertséget élvez Amerika oktatóinak körében. A sikerhez nagymértékben hozzájárul az, hogy a rendszerprogram ingyenes. Saha úgy gondolja, hogy az a tudás, amelyet így szerez meg a diák, valamint a számolás, olvasás és írás tanulása együttesen szükséges a sikerhez.
Aki tudni szeretne valamit a számítástechnikáról, az ismerje meg az „Egy Operációs Rendszert", a Unixot. Saha véleménye szerint azok a diákok, akik ismerik és használják a Linuxot, majd pedig a Unixot, könnyedén fognak jól fizető álláshoz jutni. Még a Nepálban élők is.

Linux Journal 2002. március, 95. szám

John D. Biggs
író és tanácsadó a new yorki Brooklynból.

Leng a vörös zászló

Közismert tény, hogy a Kék Óriás egymilliárd dollárt költ a Linuxra, hiszen a pénz jókora része hirdetésekben és reklámhadjáratban ölt testet.
Az már kevésbé nyilvánvaló, hogy az 1,2 milliárdnál is több embert képviselő kínai kormány úgy juttatja kifejezésre a Linux iránti szeretetét, hogy a hazai változatra, a Red Flag Linuxra való áttérést ösztönzi.
A Red Flaget 1999-ben alkották meg a Tudományos Akadémián, amelynek vezetője *Jiang Mianheng*, *Jiang Zemin* elnök fia. A fejlesztéshez az állami tulajdonú Shanghai NewMargin Venture Capital nyújtott anyagi támogatást.
A Red Flag célja, hogy gátat vessen a Microsoft Windows operációs rendszer terjedésének a kínai számítógéppiacon. Ennek leghatásosabb módja, legalábbis a kínai kormány szerint, egy már amúgy is népszerű választási lehetőség terjesztése, amely – egy kommentátor szavaival élve – „a kódolás tekintetében teljesen átlátható". Ennek érdekében az állami intézményeket és állami tulajdonban levő vállalatokat a Red Flag Linux átvételére buzdítják.
Ez is egyfajta stratégia, de létezik egy másik is: kikerülik a szerzői jogi szabálytalanságokat és a „szoftverkalózkodást" olyan program használatának támogatásával, amelynél ezek a kérdések fel sem merülnek. Egy további elképzelés szerint a hazai vállalatok programjait vásárolják, amelyek a Linuxra építenek, és nem a Windowsra. A Gartner szerint a beijingi városi önkormányzat hat helyi szállítóval kötött szerződést, a hetediket, a Microsoftot viszont elutasította. A sikeres hat egyike a Red Flag volt. Nem meglepő, hogy a Linux egyre nagyobb mennyiségben tűnik fel az asztali gépeken, legalábbis a kiskereskedelemben.
„Legutóbbi kínai utazásom alkalmával feltűnt, hogy – túl a nyelvi különbségen – a számos nagyáruházban árusított Intel-megfelelő PC-k megjelenése kissé eltér a megszokottól. Közelebbről megnézve őket rájöttem, hogy GNU/Linux fut rajtuk" – írja *Dan Gillmor* a San Jose Mercury Newstól.

Linux Journal 2002. április, 96. szám

Doc Searls

Új termékek

Compaq ProLiant BL e-Class

A Compaq által kifejlesztett ProLiant BL e-Class olyan nagyteljesítményű pengekiszolgáló, amelyet nagyvállalati rendszerekbe szánnak. Egyszerre 280 kiszolgálótorony szerelhető be egyetlen szabványos 42U méretű állványba. A tervezés során a nagyvállalati adatközpontok és az xSP-környezetek igényeit vették figyelembe, így a kiszolgálóban alacsony fogyasztású Pentium III processzor, legfeljebb 1 GB ECC memória, 30 GB merevlemez és két 10/100 ethernetkártya található. A 3U gépház támogatja a menet közben cserélhető tápegységeket és a hűtőrendszert, a hálózati kapcsolatokat és a beágyazott módszereket. A felhasználók egyszerre használhatnak Windows 2000 és Linux operációs rendszereket.

Adatok: Compaq Computer Corporation, PO Box 692000, Houston, Texas 77269, telefon: 1-800-282-6672,
➲ http://www.compaq.com

Ch 2.1

A SoftIntegration most megjelent Ch 2.1 programja C/C++ parancsfájl-értelmező, amely oktatási és nem üzleti célokra ingyenesen használható. A Ch 2.1 a C nyelv C++-osztályokkal bővített változata, támogatott a C90, a C99 legnagyobb része, a POSIX, az X11/Motif, az OpenGL, az ODBC, az XML és a GTK+. Támogatja továbbá az általános matematikai függvényeket, a mátrixszámítást és a lineáris rendszerek számításaihoz való bonyolult numerikus algoritmusokat. A Ch a gyors alkalmazásfejlesztés eszköze. Segítségével kipróbálás céljából megvalósítható a keresztfelületes héjprogramozás, a rendszerfelügyelet, a feladatok gépesített végrehajtása, a valós idejű interaktív számítások, a gyors prototípuskészítés és a két-, illetve háromdimenziós rajzolás.
Adatok: SoftIntegration, Inc., 216 F Street, #68, Davis. California 95616, telefon: 530-297-7398,
e-mail: sales@softintegration.com,
➲ http://www.softintegration.com

eclipse.org C/C++ IDE

Az eclipse.org nyílt forráskódú C/C++ integrált fejlesztőkörnyezetet készített (IDE-t) a Linux munkaállomásokon futó eclipse-felület számára. Az eclipse C/C++ fejlesztőkörnyezete grafikus felületen futó forráskódszerkesztőt, beépített hibakeresőt és a parancssor elérésének lehetőségét tartalmazza, így bonyolultabb hibakeresési kéréseket is ki lehet adni. A program kapcsolódik az eclipse-felületen már meglévő Java-támogatáshoz. Az eclipse olyan nyílt forrású környezet, amelyben sokféle módszert megvalósító alkalmazásfejlesztő eszközöket lehet létrehozni, összekapcsolni és telepíteni.

A C/C++ IDE az eclipse webhelyéről tölthető le.
➲ http://www.eclipse.org

NIC Express 1.0

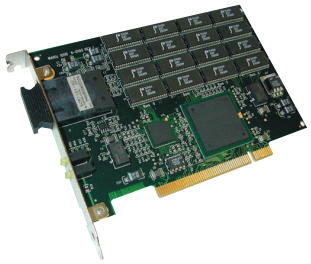
A NIC Express 1.0 gép- és gyártófüggetlen nyalábolási megoldás két vagy több hálózati kapcsolat között folyó hálózati forgalom terheléselosztására. A segítségével növelhető az átviteli teljesítmény és a hibatűrés. A NIC Express úgy is növeli a hibatűrést, hogy átfésüli a kiszolgálóhoz kapcsolódó hálózatot. Megtalálja a fizikai és logikai hibákat, és új útvonalakat jelöl ki a forgalom számára, ezáltal a hálózat két végpontja között az egy helyre összpontosuló hibák kikerülhetők. Támogatja a 2.4-es és újabb rendszermagokat, valamint minden 10/100/1000 ethernetes hálókártyával és 2/3/4-es szintű kapcsolóval együttműködik.

NIC EXPRESS
The Missing Link in Network Availability

Adatok: IP Metrics Software, Inc., 416 North Main Street, Suite #231, Euless, Texas 76039, telefon: 1-877-358-1007,
e-mail: sales@ipmetrics.com,
➲ http://www.ipmetrics.com

1000-es sorozatú ATM-kártyák

Az ImageStream Internet Solutions már szállítja új, 1000-es sorozatú ATM hálózati kártyáit, többek között DS3/E3 és OC3 kártyákat PCI és PCM kiszerelésben. Az új kártyákat elsősorban útválasztókba, kiszolgálókba és próbaberendezésekbe szánják. Az 1000-es sorozatú ATM-kártyák megfelelnek a 32 bites PCI 2.1 előírásnak, és támogatják az ATM-csomagok szétvagdalását és összerakását, hogy a PCI-sín teljesítménye



a kis csomagok mellett is a lehető legjobb legyen. A PCI és PCM kiszerelésű kártyák között található az 1001-DE, amely kétfeladatú DS3/E3 kártya, és az 1001-O3S is, amely többmódusú és egymódusú alkalmazások számára kialakított OC3 kártya.
Adatok: ImageStream, Inc., 7900 East 8th Road, Plymouth, Indiana 46563, telefon: 1-800-813-5123,
e-mail: info@imagestream.com,
➲ http://www.imagestream.com

Geodesic Suite

A Geodesic Suite alkalmazások fejlesztéséhez, kipróbálásához és telepítéséhez használható elemző- és hibakereső programokat tartalmaz. A csomag része a Great Circle, amely a fejlesztés során alkalmazható, webes felületen keresztül elérhető hibakereső környezet; és a Geodesic Runtime Solutions, amely beépül a telepített alkalmazásokba, és felismeri a kiszolgálóoldali hibákat, majd anélkül oldja meg őket, hogy az alkalmazás futását zavarná; továbbá a Geodesic Analyser, amely futási időben települ, és jelenti a teljesítményt befolyásoló szűk keresztmetszeteket és a megbízhatósági gondokat. 64 bites Itanium és 32 bites Pentium processzorokkal használható.
Adatok: Geodesic Systems, 414 North Orleans, Suite 410, Chicago, Illinois 60610, telefon: 1-800-360-8388,
e-mail: sales@geodesic.com,
➲ http://www.geodesic.com

A hónap szakmai tanácsai

A SuSE nem indul, a Red Hat nem jelenik meg

Dell Dimension L866r gépemre SuSE Linuxot telepítettem, amely LILO-t használ a rendszer betöltésére. Rendszerindításkor egy nagy halom 0 és 1-es szám önti el a képernyőmet. Telepítettem a Red Hat 7.2-t, amely a Grubot használja, és ez működött is. A 7.2 (akárcsak a többi Red Hat a 4.x óta) Nokia Multigraph 447x monitoromat sajnos nem tudja kezelni. Írtam nekik levelet, de válaszra sem méltattak.
Jim MacDonald, jimm@mediaone.net

Szerintem az a legegyszerűbb megoldás, ha a Red Hat használata mellett döntesz, és a monitort kézzel állítod be. A monitorodhoz az alábbi címen találod meg az üzemmódok beállításait: ➲ http://www.ibiblio.org/pub/linux/distributions/redmondlinux/redmond/build38/live/usr/share/hwdata/Monitors.
Christopher Wingert, cwingert@qualcomm.com

A BIOS valószínűleg nem megfelelő értékeket ad át a LILO-nak. Be van állítva a BIOS-ban a merevlemez geometriájának lefordítása? Ez a beállítás általában `"large disk support"` vagy LBA-mód néven szerepel. Segítségével a cilinderek száma egy kisebb számra (remélhetőleg 1024 alá) csökken, így a LILO a teljes lemezterületet láthatja. Ha ez nem lehetséges, győződj meg arról, hogy a rendszermagot tartalmazó lemezrész (/ vagy */boot* terjesztéstől függően) teljes egészében az 1024. cilinder alatt helyezkedik-e el.
A hiba azt is jelentheti, hogy a LILO félreértelmezte a lemez geometriáját. Közölnöd kell a LILO-val a helyes geometriát, vagy meg kell adnod a `"linear"` beállítást. További részletekért olvasd el a 2. 2. fejezetet a ➲ http://www.linuxdoc.org/HOWTO/mini/LILO-2.html oldalon.
David Brown, david@caldera.com

A következő oldal hivatkozást tartalmaz egy olyan */etc/X11/XF86Config* fájlra, amely a Nokia Multigraph 447x monitorhoz való, amilyen neked is van: ➲ http://www.geocities.com/SiliconValley/Peaks/3233/linux.html
Felipe E. Barousse Boué, fbarousse@piensa.com

Keresés a /etc/hosts fájlban és a DNS-ben egyaránt

Egy olyan nevet szeretnék feloldani, amelyik nincs a DNS-ben, de megtalálható a */etc/hosts* fájlomban, ám az `nslookup` nem hajlandó a */etc/hosts*-ban keresni. Azt szeretném, ha mindkettőben keresne, először a */etc/hosts*-ban, utána a DNS-ben. A */etc/nsswitch.conf* fájlom tartalma:

```
"hosts: files [NOTFOUND=continue] dns"
```

Sokféle Unix-rendszerrel dolgozom, és a többi unixos gép (Sun és HP-UX) működik ezzel a beállítással. Mit kell beállítanom Linuxban?
Jim Booker, jim.booker@verizon.com

Az `nslookup` nem keres a */etc/hosts*-ban, ez a rendes viselkedése. A `host` parancs viszont mindkettőben keres. A HP-UX és a Sun `nslookup` parancsát valószínűleg módosították, hogy másképp viselkedjen.
Marc Merlin, marc_bts@valinux.com

Az RPM nem hagy frissíteni

Red Hat rendszeren az RPM 4.0.3-1.03-as változata úgy tűnik, nem veszi figyelembe a `--nodeps` és a `--force` kapcsolókat, és továbbra is vizsgálja a függőségeket. Emiatt képtelen vagyok bizonyos csomagokat frissíteni. Például az

```
rpm --nodeps --force -Uvh
➥db3-3.2.9-4.i386.rpm
```

parancs a következő hibát adja:

```
failed dependencies:
libdb-3.1.so is needed by pam-0.72-26
libdb-3.1.so is needed by
➥sendmail-8.11.0-8
```

```
# rpm --nodeps --force
➥-Uvh pam-0.75-14.i386.rpm
error: failed dependencies:
  libdb-3.2.so is needed by pam-0.75-14
```

Yi Zhao, yzhao2@yahoo.com

Az efféle üzenetektől általában úgy szoktam megszabadulni, hogy egyetlen rpm-hívással frissítem az összes csomagot:

```
rpm -Uvh --force --nodeps
➥pam-0.75-14.i386.rpm  sendmail-xxx
➥db3-3.2.9-4.i386.rpm
```

Mario M. Bittencourt Neto, mneto@bunti.com.br

Az USB-billentyűzet nem működik

Compaq Presario 5000-es számítógépemre már sokféle rendszert telepítettem, többek között Red Hat 7.1-et is. Gondjaim támadtak azonban a Red Hat 7.2 telepítése közben. A telepítőprogram nem fogad adatokat a billentyűzetről. USB-s Compaq Internet PC-billentyűzettel rendelkezem. A telepítőprogram visszafejlődött, és most már nem tudja kezelni az USB-s billentyűzeteket? Ha igen, hogyan lehet a hibát kikerülni?
Brian W. Masinick, masinick@yahoo.com

Próbáld ki, hogy a BIOS-ban a „Legacy Keyboard Support"-ot állítod be.
Christopher Wingert, cwingert@qualcomm.com

Tudom, hogy Red Hat Linuxot használsz, de ha valakinek a Mandrake 8.1-es változatával támadt hasonló gondja, arra az a megoldás, hogy a */etc/sysinit/usb* fájlban a következő bejegyzésnek szerepelnie kell: `KEYBOARD_AT_START=NO`. A Linux csak ekkor működik rendesen az USB-s billentyűzettel.
Felipe E. Barousse Boué fbarousse@piensa.com

Linux Journal 2002. április, 96. szám

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

Adatvédelem az Interneten

P3P – többé nem kell unalmas adatvédelmi nyilatkozatokat olvasnunk minden egyes honlapon.

A személyes adatok felhasználásáról szóló nyilatkozatban bizonyos honlapok tulajdonosai kinyilvánítják, hogy adatainkat csak egyes meghatározott célokra használják föl. Elvárható lenne, hogy megvizsgáljuk a meglátogatott honlapok adataink felhasználására vonatkozó elveit, és elkerüljük azokat a lapokat, amelyek a személyes adatokat számunkra el nem fogadható módon kezelik.

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

Ha a honlap tulajdonosa nem tartja be a nyilatkozatban foglaltakat, a törvény például gondatlanság vagy csalás címén elítélheti. Vétkes felelőtlenség vagy a nyilatkozat tartalmának szándékosan megtévesztő megfogalmazása esetén a honlap tulajdonosa szintén jelentős kártérítés megfizetésére kötelezhető.

Ennek ellenére internethasználat közben a legtöbben nem törődünk a személyes adataink védelmével. Például nem vesszük a fáradtságot, hogy elolvassuk az adataink felhasználásáról szóló nyilatkozatot, amelyre majd' minden oldalon található hivatkozás. Vagy feltételezzük, hogy a honlap tulajdonosa kifejezett beleegyezésünk nélkül nem gyűjt és terjeszt adatokat rólunk, vagy már nem is törődünk azzal, hogy a személyes adatainkat nyilvánosan terjesztik. Ami engem illet, arra a következtetésre jutottam, hogy a személyes adataimért vívott csata elveszett, mivel már nincs energiám megtenni mindazt, ami a megvédésükhöz szükséges. Abbahagytam az adatvédelmi nyilatkozatok olvasását. Még azokat a tájékoztatókat is figyelmen kívül hagyom, amelyekben a bankom személyes pénzügyi adataim titokban tartásának lehetőségére hívta fel a figyelmemet (lefogadom, hogy az olvasók döntő többsége ebben hasonlít rám!). Olyan sok esetben folyik adatgyűjtés és -terjesztés, hogy lehetetlennek tűnik a személyes adatok védelmével bajlódni.

Egy barátom hívta föl a figyelmemet a P3P szabványra. A World Wide Web Consortium (W3C) által javasolt „keretrendszer az adatvédelem elveihez" („Platform for Privacy Policy") szabvány lehetővé teszi, hogy a felhasználók egyszerűen és hatékonyan kézben tartsák személyes adataik kezelését a hálózaton.

Danny Weitzner, a W3C társadalmi tartományának vezetője és a P3P-bizottság elnöke a következőképpen írta le a szabványt az Egyesült Államok szenátusának Kereskedelmi, Tudományos és Közlekedési Bizottsága előtt:

„A W3C és annak tagjai aggódással figyelik a személyes adatok kezelését az Interneten, mert a felhasználók nem fogják maradéktalanul kihasználni a Világháló lehetőségeit, ha ilyen kockázatokkal találják szemben magukat. A felhasználók többsége minden további nélkül hajlandó bizonyos adatokat megosztani a hálón. Ugyanakkor az alapvető emberi méltóság megköveteli, hogy ésszerű mértékben ellenőrizhessük, mely adatainkat szolgáltatjuk ki a nyilvánosságnak. Célunk, hogy az Internet alapszerkezetébe foglaljuk ennek a felhasználói ellenőrzésnek az építőköveit.

A P3P használatba vételéhez nem szükséges más, mint beállítani a böngészőben, hogy ellenőrizze, vajon a meglátogatott honlap használja-e a P3P-szabványt. Beállíthatjuk a szabványt nem támogató honlapok teljes tiltását, vagy egyszerűen csak körültekintőbben járhatunk el, ha ilyen honlapokkal osztjuk meg személyes adatainkat.

A böngésző önműködően tölti le a P3P-t használó helyekről azt a géppel feldolgozható XML-dokumentumot, amely a honlapnak a személyes adatok felhasználásáról szóló irányelveit magába foglalja. Így a böngésző meg tudja határozni, hogy a honlap tulajdonosa megígéri-e a személyes adatok védelmét, vagy az adatainkat kiadja-e másoknak.

Beállíthatjuk, hogy böngészőnk ne fogadjon el olyan honlapokat, amelyek nem felelnek meg az adatvédelemmel kapcsolatos elvárásainknak, vagy azt, hogy adatainkat ezekre a helyekre ne küldje el.

Többé nem kell majd hosszadalmas (és unalmas) adatvédelmi nyilatkozatokat olvasnunk minden egyes honlapon. Ehelyett a böngészőprogram és segédprogramjai önműködően és hatékonyan biztosítják személyes jogainkat XML-állományokat töltve le a honlapokról, még mielőtt megnyitnánk azokat.

A legfőbb programfejlesztő cégek, természetesen a Microsoftot is beleértve, részt vett a W3C P3P bizottságában. A létrejött szabványt fogyasztóvédelmi szervezetek is támogatják, köztük az Electronic Frontier Foundation.

A személyes adataink védelméhez való jog annyira alapvető számunkra, hogy tiszteletben tartását gyakran magától értetődőnek tekintjük. Személyes jogainkat azonban szorgalmas munkával kell biztosítanunk. A programok, amelyeket létrehozunk, hasznosnak bizonyulhatnak adataink védelmének biztosításához – és a P3P-szabvány olyan segédeszköz, amely pontosan ezt teszi.

Linux Journal 2002. április, 96. szám

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).

Természeti erők

Ebben a hónapban Doc a gyerekek kíváncsiságáról elmélkedik.

Csaknem három éve, hogy ***Dr. Sugara Mitra***, az NIIT Kognitív Rendszereket Kutató Intézetének vezetője feltűnés nélkül internetes fülkéket kezdett telepíteni Új-Delhi szegénynegyedeiben azokra a helyekre, ahol a gyerekek az idejüket töltik. A ***Falba vágott lyuk***-nak nevezett program a „legkevésbé erőszakos oktatásra" tett kísérlet volt. A hagyományos oktatással szemben az alaptétel itt az, hogy „ha nincs irányított oktatás, akkor bármely tanulásra alkalmas környezet, amely megfelelő kíváncsiságra sarkall, tanulást eredményez."

Nézzük meg, mi történt a program webhelyének (➲ http://www.niitholeinthewall.com) beszámolója szerint: a kísérlet célja annak megállapítása volt, hogy az embereket érdekelné-e egy felügyelet nélkül, a szabadban, minden felhasználói utasítás nélkül otthagyott internetalapú fülke. Valamint, hogy működőképes lehet-e egy minden felügyelet nélküli, szabadban hagyott fülke.

A NIIT-iroda fala, ahol a számítógépet elhelyezték, közvetlenül egy nyomornegyeddel határos, ahol sok 0–18 év közti gyerek él. E gyerekek közül sokan nem járnak iskolába, a többiek állami iskolában tanulnak, ahol kevés a felszerelés, hiányoznak a jó tanárok, és a tanulók sem túlságosan lelkesek. Ezek a gyerekek az angol nyelvet sem ismerik valami jól.

A kísérlet eredményei nagyon bíztatók. Az internetes fülke megnyitását követő három hónapon belül kiderült, hogy a többnyire a nyomornegyedből jövő gyerekek minden tervezett oktatási irányítás nélkül szert tettek bizonyos szintű számítógéphasználói ismeretre. Tudnak böngészni és letölteni az Interneten, és dolgoznak az MS Paint programmal. A negyedik hónapban felfedezték az olyan lehetőségeket, mint a könyvtárak készítése, kivágás és beillesztés, parancsikonok létrehozása, ablakok mozgatása és átméretezése, és billentyűzet hiányában is használni tudták az MS Wordöt rövid üzenetek megírására. A fülke ma is működik, és naponta körülbelül 80 gyerek használja.

A rákövetkező két és fél évben Dr. Mitra és munkatársai négy indiai város 29 különböző falánál helyeztek el Internetre csatlakoztatott számítógépeket. A kísérlet folytatódik, de már eddig is számos megállapítást tettek:

1. Egészében véve a felhasználók a felszerelést nem rongálják és nem károsítják.
2. A tanulás társas tevékenység. A gyerekek csoportosan gyorsabban tanulnak, mert a tagok meg szeretnék osztani a többiekkel, amit megtanultak.
3. A visszahúzódó gyerekeket sem hagyják ki. Főleg a lányok látnak el szervezői feladatokat, elzavarják a géptől a képernyőre tapadókat, hogy így a csöndesebb gyerekek is sorra kerülhessenek. A gyerekek még tanfolyamokat is szerveznek egymásnak.
4. A felnőttek nem vesznek részt, bár ők is úgy gondolják, hogy a fülkék hasznosak.

A legtöbbet mondó felfedezés talán az volt, hogy Dr. Mitra kísérleti alanyainak nem tetszik, hogy kísérleti alanyok. Az egyik, a csoporttól kapott hindi nyelvű üzenet így szólt: „Megtaláltuk és bezártuk azt a dolgot, amivel minket figyelnek." A kutató teljesen fellelkesedett: „Annyira örültem! Nem hiszem, hogy tanárként bármilyen nagyobb öröm érhetne, mint amikor a gyerekek megvernek a saját térfeleden."

Most tartsunk egy kis szünetet, és vizsgáljuk meg, mennyire ironikus is ez. Ezekben a fülkékben a Microsoft Windows operációs rendszerének különféle változatai futottak (és futnak), ami pénzbe kerül. Az önállóan tanuló tömegek operációs rendszere minden vitán felül a Linux, amely ingyenes. A Linux az egyetlen operációs rendszer, mely jogosan állíthatja, hogy az egész világnak megfelel. Nézzünk meg egy idézetet a világ másik végéből: „A Linux nagyon fontos szerepet játszhat a latin-amerikai és karibi modernizációban, mivel olyan hálózatok építhetők vele, melyek sok-sok egyetemnek, főiskolának, iskolának és oktatási központnak elérhetővé teszik az Internethez való csatlakozást, lehetővé téve számukra, hogy ezt a csodálatos eszközt használják tudományos és kulturális szintjük fejlesztésére. Egyszóval, a LINUX az az eszköz, melynek révén csökkenhet az országok közti „technikai szakadék". „Ennek az idézetnek az ENSZ a forrása és a ➲ http://ctrlaltesc.org közölte le.

Egy oktatási intézményrendszernek nehéz elismernie a gyerekek természetes kíváncsiságának mindent felülmúló erejét, és éppúgy nehéz egy üzleti intézménynek elismernie a Linux és a többi szabad és nyílt forrású program hasznosságát. A hagyományos oktatás azt feltételezi, hogy a gyerekek üres edények, akiket le kell ültetni egy teremben, és beléjük kell tölteni a tantervekben előírt tartalmat. Dr. Mitra kísérlete bizonyítja, hogy ez tévedés. A hagyományos programok azt feltételezik, hogy az üzlet pusztán az előre csomagolt bitek árusításáról szól. A Linux bizonyítja, hogy ez tévedés, egyszerűen azáltal, hogy mindenhol használják. A kereskedelmi programokat nem éri kár, hiszen a világ nagy, és az üzleti életben elég hely van, hogy mindenki elférjen. De nem érjük el a világ számtalan milliárd kíváncsi lelkét, ha minden erőfeszítésünkkel azon vagyunk, hogy csomagolt biteket adjunk el nekik. Ennél nagyobb keretekben és gyakorlatiasabban kell gondolkodnunk. A világban sokkal szélesebb körben kell használni a technikát, ha azt szeretnénk, hogy sokak részesedjenek a jólétből, amely ma kevesek kiváltsága. Arról van szó, hogy fel kell használni a természetben található anyagokat, melyek megújulnak, és minél inkább kihasználjuk őket, annál inkább fejlődnek. És nincs olyan természetes környezetben elérhető építőelem, melyre jobban ráillene ez a leírás, mint a Linuxra – feltéve természetesen, hogy nem a gyermekek természetes kíváncsiságával vetjük össze.

Linux Journal 2002. március, 95. szám

Doc Searls (doc@ssc.com) a Linux Journal szerkesztője, havi rovata a Töprengő, valamint a Cluetrain Manifesto társszerzője.

A PPPD beállítása Linuxban (1. rész)

Az Internethez való kapcsolódás könnyebb lehet, mint gondolnánk. Tony kétrészes cikkében a modemek Linux alatti beállítását ismerteti.

Manapság asztali számítógépükre egyre többen telepítenek Linuxot. Sajnos sokan nehézségekbe ütköznek, amint olyan lehetőséget próbálnak ki, amelyről ma már senki sem hajlandó lemondani: kapcsolódni szeretnének az Internetre. Milyen akadályokkal találják is szemben magukat ezek a felhasználók? Több is jelentkezik, de a legfontosabb akadály az MS Windows távoli elérése hivatalos lehetőségének vagy a Mac által biztosított ConfigPPP-nek a hiánya. A végeredmény az, hogy sok felhasználó a gépére hasonló feladatokat ellátó rendszereket telepít, úgymint Gnome betárcsázót, illetve Linuxconf, KDE tárcsázó (dialer) stb. programot, anélkül téve azonban, hogy ezekről a programokról különösebb ismeretekről rendelkeznének.
Túlságosan is nagy lehetősége van egy olyan helyzet kialakulásának, amikor semmilyen önműködő beállítóprogram sincs telepítve a gépre, például a felhasználó sem a Gnome-ot, sem a KDE-t nem jelölte ki telepítésre. Ezek után miért ne vállalnánk a kihívást, hogy megtanuljuk az internetkapcsolat kézi beállítását? Ez a cikk éppen a modemek Linux-rendszer alá történő telepítéséről szól, amelynek megoldása, úgy tűnik, számos felhasználónak fejtörést okoz. Elmondom, hogyan is kell beállítani a PPPD-t, amelynek előfeltétele természetesen a modem jó beállítása.

Előzetes ismeretek

Először a héjprogram használatával célszerű megismerkednünk: hogyan kell könyvtárat váltani, egy könyvtár tartalmát listázni, bármilyen szerkesztőprogram segítségével állományokat szerkeszteni, valamint az X-rendszerben virtuális konzolokat és terminálokat használni.
Cikkünkben feltételezzük, hogy olvasónk nem szoftveres modemmel rendelkezik, amely csak a Windowszal képes együttműködni. Bár az ilyen modem használata is lehetséges, beállítása sok fáradsággal jár, és túlmutat jelen írás keretein.

A modem megkeresése

Elsőként meg kell állapítanunk, tulajdonképpen hol is van a modem. Azt a soros kaput kell keresnünk, amelyhez a modem kapcsolódik. Ez még abban az esetben is szükséges, ha beépített modemet használunk, ugyanis a modemkártyán is található soros kapu. A számítógépbe feltehetőleg két soros kapu lett beépítve. Nagyon is valószínű, hogy az egyiket már az egér használja, hacsak nem PS/2 vagy USB-csatolófelületű egerünk van. A Unix-rendszerben minden fizikai berendezés a */dev* könyvtár egy-egy bejegyzéseként jelenik meg. Ez a könyvtár minden a számítógépre telepített készülékre nézve tartalmaz egy bejegyzést. A soros kapukat a `ttyS` betűsorozattal szokták jelölni, amelyeket 0-tól 3-ig terjedő, egyjegyű szám követ.

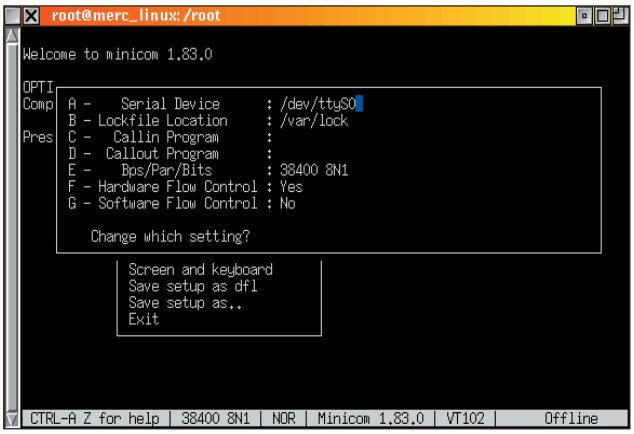
```
cd dev
ls -l ttyS*
crw-------   1 merc    tty    4,
➥64 Aug   3 10:24 ttyS0
crwxr-xr-x   1 root    tty    4,
➥65 Aug   3 10:25 ttyS1
crw-------   1 root    tty    4,
➥66 May   6   1998 ttyS2
crw-------   1 root    tty    4,
➥67 May   6   1998 ttyS3
```

Vajon melyikhez csatlakozik a modem? A válasz: attól függ, hogy melyik aljzatba dugtuk be a modem kábelét, illetve a belső modemnél az dönti el, hogy hogyan állítottuk be. Ha már megszoktuk, hogy a soros kapukra COM1, COM2 stb. névvel hivatkoznak, azt is tudnunk kell, hogy ezek Linuxban használatos megfelelői a következők: COM1=ttyS0, COM2=ttyS1, COM3=ttyS2, COM4=ttyS3. Amennyiben nem tudjuk megállapítani, hogy a modem melyik csatlakozóba lett bedugva, rövid úton kideríthetjük.
Rendszerint létezik egy modemnévvel ellátott közvetett hivatkozás – ez a Windows szóhasználatban nagyjából a parancsikon megfelelője –, amely az egyik soros kapura mutat. Az én rendszeremben például:

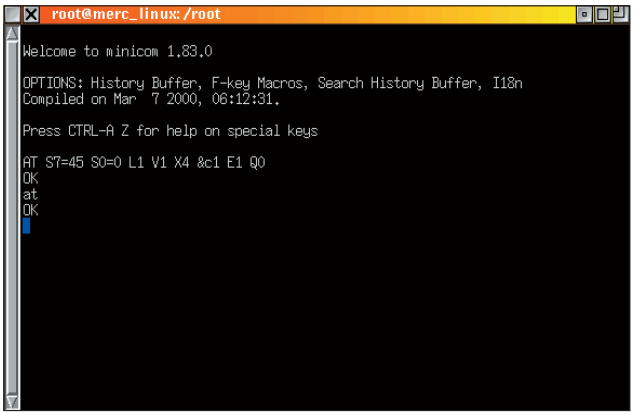
```
ls -l modem
lrwxrwxrwx 1 root root 5 Feb 7 2000
➥modem ->ttyS1
```

Az én esetemben a modem a második soros kapura – a `ttyS1`-re – van kötve, azaz a fenti példában bemutatott közvetett hivatkozás azt jelenti, hogy bármilyen program, amely a */dev/modem* állományt használja, valójában a */dev/ttyS1* kapun dolgozik.
Sose felejtsük el, hogy rendszerünk eltérhet a példában bemutatottól, sőt még az is előfordulhat, hogy a `ttyS0` vagy `ttyS1` kapukra mutató közvetett hivatkozások sem léteznek.
Most azt a célt tűztük ki, hogy egy olyan bejegyzést hozunk létre a */dev* könyvtárban, amely a megfelelő soros kapura mutat, vagyis arra, amelyiken a modem kapcsolódik.
Nos, először is meg kell vizsgálnunk, melyik kapura van csatlakoztatva a modem. Gépeljük be a `minicom` parancsot. Ahhoz, hogy ellenőrizzük, a `minicom` képes adatcserét lebonyolítani a modemmel, gépeljük be az `at` parancsot, és üssük le az ENTER-t. Ha OK válasz érkezik, akkor a `minicom` a */dev* könyvtárbeli megfelelő állományt használja a modem elérésére. Máskülönben a `minicom` valamilyen okból kifolyólag nem tud „párbeszédet” folyatatni a modemmel. Amennyiben a modemtől nem érkezik megerősítő válasz, elsőként ellenőrizzük, hogy hová lett kötve, vagyis melyik soros kapuhoz csatlakozik. Ez a feladat a `minicom` programból könnyen elvégezhető: nyomjuk meg a CTRL-A billentyűkombinációt, majd az O-t. Fontos megemlíteni, hogy a minicom néha úgy

van beállítva, hogy CTRL helyett az ALT gombot használja. Amennyiben ez a helyzet, ne felejtsük el, hogy a CTRL billentyű helyett az ALT billentyűt kell lenyomnia, vagyis a fenti példában a ALT-A után az O billentyűt.
A képernyőn a `minicom` beállítási képernyője jelenik meg. Most válasszuk ki a ***Serial port setup*** (a soros kapu beállítása) lehetőséget, majd üssünk ENTER-t.
Ezt követően bökjünk az A-ra, és válasszunk másik soros kaput. Az éppen beállított soros kaput (***/dev/ttyS0***) változtassuk meg például ***/dev/ttyS1***-re (*1. kép*). Lépjünk ki a `minicom` programból és indítsuk újra. Nyomjuk meg az ENTER gombot, majd mentsük a beállítást a ***Save default as dfl*** (Beállítás mentése alapértelmezettként), végül válasszuk a ***Exit*** (Kilépés) lehetőséget. Ekkor a minicom programból a CTRL-A, majd az Q megnyomása után kiszálltunk.



1. kép Erről a képernyőről megtudhatjuk, a minicomnak melyik soros kaput kell használnia, hogy sikeres adatcserét folytathasson a modemmel



2. kép Így kell működnie a minicomnak, ha a modem OK választ küld

Futtassuk ismét a `minicom` programot, hogy ellenőrizni tudjuk, vajon végez-e adatcserét az általunk választott soros kapuval. Üssük le az ENTER billentyűt, és ha ezúttal megkaptuk az OK-t, máris megtaláltuk a modemet (*2. kép*). Egyébként az ebben a fejezetben leírt lépéseket mindaddig ismételnünk kell, amíg a keresett soros kapura rá nem bukkanunk, tehát a modemtől OK választ nem kapunk (az `at` parancs begépelésekor a `minicom` parancs futtatását követően).
Mindig arra gondoljunk, hogy a modem csatlakozási pontjának keresése közben a közönséges hibakeresési módszereket kell követnünk. Különösen fontos meggyőződnünk róla, hogy a modem be van-e kapcsolva, csatlakoztatva van-e a géphez, megfelelő-e a tápellátása, és megbízható-e a számítógéppel való kapcsolata. Ezek a vizsgálatok talán jelentéktelennek tűnnek, de éppen magától értetődő jellegük miatt gyakran figyelmen kívül szokás hagyni őket. Ennek nyomán órákat tölthetünk el egy modem beállításával, amely nincs a géphez csatlakoztatva, sőt talán be sincs kapcsolva.
Ha a modemtől megérkezik az OK válasz, akkor minden működésre kész, és már csak néhány teendőt kell elvégeznünk. A `minicom` programon belül a CTRL-A, majd O billentyű használatával a beállító képernyő jelenik meg; itt válasszuk a ***Serial port setup*** (Soros kapu beállítása) lehetőséget, majd üssük le az ENTER billentyűt.
Mindjárt az első sorban szembetűnik, hogy a `minicom` melyik állományt használja a modemmel folytatott adatcsere lebonyolítására. Ha ez a ***/dev/modem***, akkor minden helyesen van beállítva a rendszerben, és máris cikkünk ***Párbeszéd a modemmel*** részére ugorhatunk.
Amennyiben a modemtől OK választ kapunk, de a `minicom` egy másik soros kapura van beállítva, a ***/dev*** könyvtárban létre kell hoznunk a modem névvel ellátott közvetett hivatkozást. Ez a ***/dev/modem*** bejegyzés fogja biztosítani a modem Linux-rendszerbeli működését – tekintsünk vissza: pontosan ezt tűztük rövid távú célul. Vessük papírra, hogy a `minicom` melyik kaput használja, most tételezzük fel, hogy ez a ***/dev/ttyS0***.
Lépjünk ki a beállítási képernyőből, vagyis üssük le az ENTER billentyűt, és válasszuk az ***Exit***-et (Kilépést), és szálljunk ki a minicom programból is a CTRL-A, majd a Q billentyűk leütésével. Ezzel vissza is tértünk a héjprogramhoz. Most lépjünk be a ***/dev*** könyvtárba, és hozzunk létre közvetett hivatkozást a modem számára az alábbi módon:

```
cd /dev
ln -sf ttyS0 modem
ls -l modem
lrwxrwxrwx  1 root  root  5 Aug  3
➥12:32 modem ->ttyS0
```

Természetesen a `ttyS0` kaput a `minicom` beállítási képernyőjén talált kapubeállítással kell helyettesítenünk. Most ismét futtassuk a `minicom` programot. Az `AT` parancs begépelésével és az ENTER leütésével győződjünk meg róla, hogy minden továbbra is működik. A megerősítő `OK` válasznak ismét meg kell érkeznie (*2. kép*).
Újra lépjünk a minicom beállító képernyőjébe – CTRL-A és O –, majd válasszuk a ***Serial port setup*** (Soros kapu beállítása) lehetőséget. Most az A billentyű lenyomásával a ***/dev/modem*** bejegyzéssel végezzük el a ***/dev/ttS0*** állomány helyettesítését (1. kép). Bökjünk rá az ENTER-re, válasszuk a ***Save default as dfl***-t (Beállítás mentése alapértelmezésként), és lépjünk ki a beállítóképernyőből (Exit). Hogy a `minicom` `az œj beÆll tÆsokkal` megfelelően működjön, ki kell lépni a `minicom` `programb l`, majd újra kell indítanunk: CTRL-A és O.
A minicom most már a ***/dev/modem*** készüléket használja. Gépeljük be az `AT` parancsot, erre meg kell kapnunk az OK választ (2. kép). Ha ez a helyzet, gratulálunk a sikeres beállításhoz.
A modem beállítása a legkritikusabb lépés az Internethez történő kapcsolódás során. Sok felhasználó nem tudja, hogy modemje melyik soros kapuhoz van kapcsolva. Amint az előző fejezetben láttuk, a modem már korábban jól be lett állítva, most csak meg kellett állapítanunk, hol van, és a ***/dev*** könyvtárban létre kellett hoznunk a helyes bejegyzést, amely a megfelelő készülékállományra mutat.

Párbeszéd a modemmel

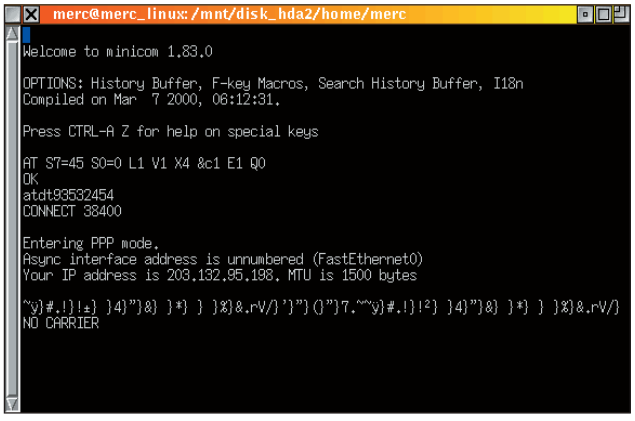
Érdemes megjegyezni, hogy maga a `minicom` semmit sem tud az `AT` parancsról vagy az OK válaszról. Fő feladatai valójában rendkívül egyszerűek: megjeleníti a soros kapu felől érkező karaktereket, illetve a felhasználó által a billentyűzeten begépelt karaktereket elküldi a soros kapu felé.

Bizonyos értelemben a modem olyan, akár egy robot, amely a géppel a soros kapun keresztül társalog. Ez az oka annak, hogy a `minicom` bizonyos értelemben megengedi, hogy „magánbeszélgetést" folytassunk a modemmel.

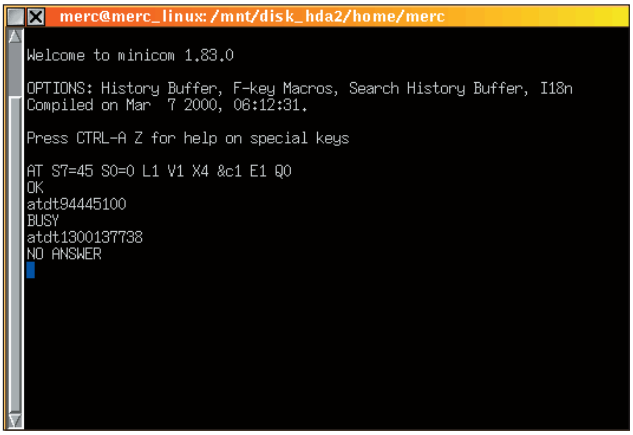
A fenti példában a felhasználó küldte az `AT` jelet, majd a modem válaszolt az OK< Enter > jelsorozattal.

Miféle párbeszédet folytathatunk a modemmel? Tény, hogy minden modem képes az at-parancskészleten alapuló Hayes-parancskészlet fogadására. De némely alkatrészgyártó a szabványon túl érdekes bővítésekkel is rendelkezik. Ha kíváncsiak vagyunk rá, melyek ezek, nem kell mást tennünk, csak tanulmányozzuk át őket a felhasználói útmutatóban. Például az ALT-1, ALT-2, ALT-3 minden modem esetén ugyanazt jelenti: a hangszóró hangerőszabályzóját alacsony, közepes vagy nagy teljesítmény fokozatra állítja. Annak ellenére, hogy e parancsok közül nem soknak az ismeretére van szükségünk, talán mégis szeretnénk kipróbálni néhány parancsot, hogy egy kicsivel többet tudjunk modemünk működéséről. Mindig gondoljunk arra, hogy az `AT&F` és `aT&W` parancsokkal bármikor visszaállíthatjuk a gyári értékeket, amennyiben bizonytalan működésű parancsokat használtunk, amelyek elrontották a korábban jól működő beállítást, sőt talán még a kapcsolatlétesítést is lehetetlenné tették.

Az `ATDT93355100` parancsot tárcsázásra használhatjuk, jelen esetben a 93355100 telefonszám hívására. A `DT` jel (tone) üzemmódot jelent, míg a `DP` impulzus (pulse) üzemmódot jelent. Az utóbbit akkor kell használni, ha vidéki környezetben analóg telefonközponthoz kapcsolódunk. Kíséreljünk meg `ATDT` parancsot küldeni a modemnek, ezután adjuk meg az internetszolgáltató telefonszámát, majd kísérjük figyelemmel, mi történik. Az én esetemet a *3. kép* szemlélteti. Mint az világosan látszik, az alkatrészek összehangolása után – ez mindkét oldalon sok összesípolással jár – a modem szépen küldi az `CONNECT 52500` üzenetet, amely arra utal, hogy a kapcsolat hiba nélkül létrejött. Természetesen emellett léteznek más üzenetek is: `NO DIALTONE`: a modemhez nincs telefonvonal kapcsolva; `BUSY`: a telefonvonal foglalt; `NO ANSWER`: nincs válasz stb. (*4. kép*). Az én esetemben a kapcsolat sikeresen létrejött, és amint kapcsolódtam, internetszolgáltatóm rejtélyes jelsorozatot küldött, amely valahogy így kezdődött: `~}#.!}!-} }4} }&} }*} } }%)`. Arról van szó, hogy internetszolgáltatóm a PPP-démonommal szeretne kapcsolatot teremteni, ugyanakkor én a `minicomot` futtatom, ami azt mutatja, amit internetszolgáltatóm küldött a PPP-démonomnak.



3. kép Ilyen volt az én képernyőm, miután internetszolgáltatómmal sikerült kapcsolatot teremtenem



4. kép Két szám tárcsázásának eredménye: az első szám foglalt volt, a második pedig nem felelt

Innentől kezdve mindig tartsuk észben, hogy a modem ezentúl nem hajlandó az a parancsaira válaszolni, minden soros kapura küldött adat modulálva lesz, és a vonal másik végére lesz küldve. Ugyanakkor a vonal másik végéről érkező adatok demodulálva lesznek, és a soros kapura lesznek átirányítva – innen származik a modem elnevezés. A számítógép, jobban mondva a soros kapu minderről semmit sem tud. Minden pontosan úgy működik, mintha a gépet a másik oldallal soros kábel kötné össze. Tény, hogy ha két géppel rendelkezünk, ezeket a soros kapukon egy soros kábel segítségével keresztül könnyedén összeköthetjük a PPPD-vel.

Csak arra kell figyelni, hogy fordítós kábel legyen, amit használunk, vagyis az első gép küldő tűcsatlakozója a másik gépnél fogadó csatlakozó legyen és fordítva.

A következő hónapban a számítógép egy olyan beállításáról fogok beszélni, amelynek segítségével elérhető, hogy az Internethez mindenféle bonyolult eszköz nélkül csatlakozni tudjunk. A lényeg az, hogy mardjunk vonalban!

Linux Journal 2002. február, 94. szám

Tony Mobily
(merc@mobily.com) a Login olasz számítógépes magazin műszaki szerkesztője, valamint linuxos vizsgára készít fel, továbbá rendszergazda és táncoktató. A nyelvek közül többek között az angolt, az olaszt, a C-t és a Perlt ismeri. Tony honlapja a ➲ http://www.linuxcertification.com címen kereshető fel.

Kapcsolódó címek

➲ http://www.tldp.org/HOWTO/PPP-HOWTO/
➲ http://www.topology.org/linux/pppd.html
➲ http://oh3tr.ele.tut.fi/~oh3fg/ppp/ppps.html

Kisvállalkozások levélkezelése

Rövid beszámoló a Hírép Kft. egy évvel ezelőtti belső és külső levelezési rendszerének, illetve a munkatársak közös, osztott internethasználatának kiépítéséről.

A rendszer kezdetben egy Mandrake Linux 7.2-alapú kiszolgálóra épült, de körülbelül három hónapja áttértünk Mandrake 8.1-re. A Linux-kiszolgáló üzembe helyezése óta folyamatosan, hiba nélkül üzemel. Néhány hónapja a cég Novell NetWare 3.11 kiszolgálóját is megszüntettük, helyette a fájl- és nyomtatókiszolgálói feladatokra a Samba kiszolgálót állítottuk csatasorba. Az eddig hibátlanul működő NetWare cseréje az alábbi két okból vált szükségessé: a hosszú fájlneveket gond nélkül szerettük volna használni, valamint a gép már nagyon régi volt, új kiszolgálót akartunk használatba venni. Az új Linux-rendszert futtató gép főbb alkatrészei: 512 MB RAM, 40 GB IDE merevlemez.

A cég jelenleg – az LNX névre keresztelt Linux-kiszolgálón kívül – 24 Windows NT/2000/98/95 ügyfélgéppel és az összekötő ethernethálózattal rendelkezik. Az *1. ábra* a vállalat jelenlegi számítógéphálózatának felépítését mutatja.

A belső TCP/IP hálózati címeket a saját 192.168.1.0 hálózati címtartományból állítottuk be. Címeket saját használatra mindenki szabadon használhatja, ugyanis az útválasztó programok nem továbbítják az ilyen IP-című csomagokat. A PPP-kapcsolat kezdetben egy 56 KB/s-os analóg modem volt, amit fél éve egy ISDN-vonal váltott fel. A Mandrake Linux 7.2 és a 8.1 ISDN-kapcsolatot beállító programok hibátlanul működtek, ezért az ISDN-kapcsolat beállítása könnyű feladat volt. Mandrake 8.1 esetén a HardDrake használatát javaslom, ahol az ***ISDN Adapters*** választása esetén egy varázsló segítségével a következő kapcsolatokat állíthatjuk be: analóg és ISDN-kapcsolat, ADSL és egyéb kábelmodemes összeköttetés. A PPP-kapcsolat természetesen nem a legeszményibb módja az Internetre csatlakozásnak, de a UPC, a Matáv és egyéb adatkapcsolati szolgáltatók a cég földrajzi környezetében jelenleg a korszerűbb megoldásokat nem biztosítják.

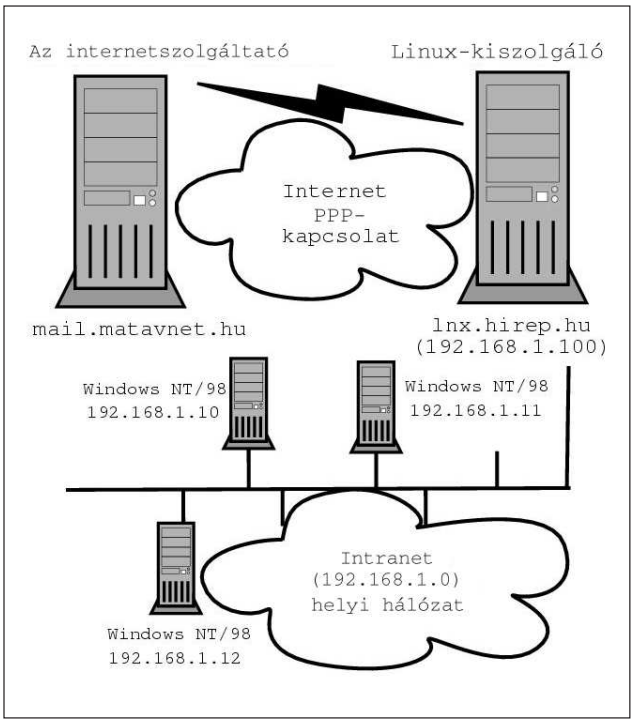
A feladat megoldásához használt programcsomagok

A Linux telepítése során még a következő csomagok telepítésére is szükség volt:

- postfix-20010228-15mdk.rpm (ez a levélkiszolgáló program),
- ppp-2.41-2mdk.rpm (a PPP-kapcsolat miatt),
- imap-2000c-7mdk.rpm (az IMAP-kiszolgáló használatára szükségünk lesz),
- fetchmail-daemon-5.9.0.-4mdk.rpm, fetchmaiconf-5.9-0-4mdk.rpm, fetchmail-5.9.0.-4mdk.rpm (a Fetchmail program a leveleket a kiszolgálóról a POP3-protokoll segítségével fogja letölteni), webmin-0.87-4mdk.rpm (a távoli és könnyű felügyelhetőség érdekében mindenkinek ajánlom). A fenti összetevők telepítése utáni legkisebb programbeállítási igényeket később ismertetem.

A megvalósított levelezési rendszer működési logikája

Az internetszolgáltatók (továbbiakban: ISP – Internet Service Provider) azt a lehetőséget kínálják, hogy egy postafiókba gyűjtik azokat a leveleket, amelyek tartományneve azonos. Az ISP-nél a beérkező levelek számára bejegyeztettük a hirep.hu tartománynevet. Általában három lehetőség közül lehet választani, ha postafiókot kérünk: az első esetben mindenkinek saját felhasználói fiókja van: ***xy@hirep.hu***; a másodikban létezik egy ***info@hirep.hu*** postafiók, ahová ISP-nk levélkiszolgálója minden olyan levelet betesz, ahol címzettként a ***@hirep.hu*** tartománynév szerepel. A Hírép Kft. esetében egy harmadik megoldásra kötöttünk szerződést. Ez abban

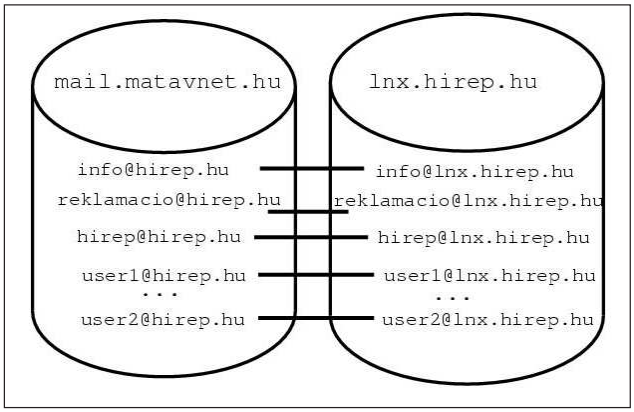


1. ábra A Hírép Kft. hálózata

különbözik a másodiktól, hogy az ISP számára fel kellett sorolnunk a létező címzetteket, mert ezek az ISP-levélkiszolgálón az „info” postafiók állandó aliasneveiként lettek kialakítva. Ezzel megteremtődött a lehetőség, hogy bárki levelet küldhessen a cég és az alkalmazottak számára, de további feladatként jelentkezett a levelek központilag felügyelt elérhetősége.

A Kft. saját (***lnx.hirep.hu*** nevű) Linux-kiszolgálóján a Postfix levélkiszolgálót helyeztük üzembe. Az ISP-nek leadott felhasználói neveket ezután a saját kiszolgálónk felhasználói közé is felvettük.

A *2. ábra* azt mutatja, hogy az ISP-nél kezelt egyes levélcímek és a saját levélkiszolgálókon kialakított postafiókok között milyen összerendelődést (e-mail mapping) alakítottunk ki. Ezt a kapcsolatot majd a Fetchmail program beállítófájlja fogja leírni. Levelezőrendszerünk úgy fog működni, hogy a belső leveleket a ***user@_xy@lnx.hirep.hu*** címre kell küldeni, amelyek azonnal továbbítódnak, hiszen ehhez az internetkapcsolatnak nem kell élnie. Természetesen küldhetünk levelet a ***user_xy@hirep.hu*** címre is, de ez csak a PPP-kapcsolat és a



2. ábra A szolgáltatótól kapott és a helyi levélkiszolgáló fiókjainak összerendelése (helyi címek kialakítása)

Fetchmail lefutása után fog bekerülni a címzett postaládájába. A külvilág számára (a @hirep.hu tartományon kívüli levélcímek) írt, illetve az onnan kapott levelek csak a PPP-kapcsolat feléledésekor tudnak továbbítódni, illetve beérkezni. A külső leveleket a Fetchmail program által kezelt POP3- (Post Office Protocol 3) eljárás segítségével töltjük le a helyi postafiókokba (a 2. ábra összerendelése szerint). A várakozó kimenő levelet pedig a Postfix önműködően továbbítja az SMTP (Simple Mail Transfer Protocol) segítségével. A Windows-munkaállomások az Outlook vagy az Outlook Express programokat használják. Programokba beállított postafiókokat az IMAP- (Internet Message Access Protocol) postafiók hozzáférési módszere szerint használjuk. Előnye, hogy a levelek az ***lnx.hirep.hu*** kiszolgálón maradnak, ami főleg a közösen használt postafiókok esetén (info, reklamacio, hirep) hasznos, mert így mindenki olvashatja őket (természetesen a postafiók jelszavának ismerete szükséges).
A fenti ***ISP/Saját Linux-kiszolgáló/Windows munkaállomások*** szerkezetű kialakításnak egy járulékos feladata is létezik: Az IP-Chains rendszermag-szolgáltatást kihasználva a munkaállomások számára megnyílik az osztott internethasználat lehetősége.

Az Internetre történő csatlakozás (PPP-kapcsolat felépítése és megszüntetése)

A Mandrake 8.1 Linuxban a PPP-kapcsolat beállítása után (a ***/dev/ppp0*** csatoló meghatározása) a PPP-csatoló felépítésére az `ifup ppp0` parancsot használhatjuk. A következő lista ezt a parancsot használva építi fel a PPP-kapcsolatot (a parancsfájl neve: `startinet.sh`):

```
#! /bin/sh
# Ha sz ksØges felØp tj k a PPP-kapcsolatot
DB=`ifconfig | grep "ppp" | wc -l`
if [ ${DB} = 0 ]; then
   echo "A PPP-csatol  elind tÆsa..."
   ifup ppp0
   /etc/ppp/firewall-masq
else
   echo "A PPP-csatol  mÆr mßk dik,
➥hasznÆlhatja..."
fi
exit 0
```

A parancsállomány az `ifconfig` utasítás segítségével megállapítja, hogy a PPP-kapcsolat már létezik-e, hiszen bárki más is elindíthatta. Amennyiben a ***/dev/ppp0*** csatoló nem létezik, akkor felépíti. A `firewall-masq` parancsfájl ismertetésére is nemsokára sor kerül.

A PPP-kapcsolat lebontására a következő parancsfájl (neve: `stopinet.sh`) szolgál:

```
#! /bin/sh
echo "A PPP-kapcsolatot bontom..."
ifdown ppp0
kill -9 `ps -ax | grep "ppp-watch" | awk
➥'{print $1 }' | sed 1q`
echo "A PPP-kapcsolat lebontva..."
exit 0
```

Az `ifdown ppp0` lebontja a kapcsolatot, de az `ifup` egy figyelőfolyamatot is elindít, amit nekünk kell kilőni. Ennek a figyelőfolyamatnak egyébként az a feladata, hogy újraépítse a véletlenül megszakadt kapcsolatokat. A visszafelé hajló hiányjelek között megfogalmazott parancs a `ppp-watch` figyelőfolyamat mindenkori PID-jét adja vissza, amelyre már gond nélkül használhatjuk a `kill` parancsot. Egy folyamat név szerint történő megszüntetése annyira hasznos feladat, hogy a fenti `kill` parancs működésének tanulmányozását mindenkinek a figyelmébe ajánlom. Segítségül annyit érdemes tudni, hogy a `ps -ax` az összes futó folyamatot kilistázza, amiből a `grep` kiveszi azokat a sorokat, amelyekben a `ppp-watch` szerepel. Ezekből az `awk` lecsípi az első szavakat (ezek már a PID-ek lesznek). Az így kialakult egyszavas sorokból a `sed 1q` kiveszi az első sort, ami most csak egyetlen szó, azaz a `ppp-watch` PID-je. A kapott szó a `kill -9 ...` parancs része lesz, mert a `...` jelek makróhelyettesítést hajtanak végre a parancsban.
A névfeloldás biztonsága érdekében az ISP-névkiszolgálót érdemes felvenni a ***/etc/resolv.conf*** fájlba, ami nálunk így néz ki (ezeket az ISP használja):

1. lista A firewall-masq.sh

```
#!/bin/sh
# Interface to Internet
EXTIF=ppp+
ANY=0.0.0.0/0
ipchains -P input ACCEPT
ipchains -P output ACCEPT
ipchains -P forward DENY

ipchains -F forward
ipchains -F input
ipchains -F output

# Deny TCP and UDP packets to privileged
# ports
ipchains -A input -l -i $EXTIF -d $ANY
➥0:1023 -p udp -j DENY
ipchains -A input -l -i $EXTIF -d $ANY
➥0:1023 -p tcp -j DENY

# Do masquerading
ipchains -A forward -j MASQ
echo 1 > /proc/sys/net/ipv4/ip_forward
```

2. lista A firewall-standalone.sh

```
#!/bin/sh
EXTIF=ppp+
ANY=0.0.0.0/0
ipchains -P input ACCEPT
ipchains -P output ACCEPT
ipchains -P forward DENY

ipchains -F forward
ipchains -F input
ipchains -F output

# Deny TCP and UDP packets to privileged
# ports
ipchains -A input -l -i $EXTIF -d $ANY
➥0:1023 -p udp -j DENY
ipchains -A input -l -i $EXTIF -d $ANY
➥0:1023 -p tcp -j DENY
```

```
nameserver 195.228.240.248
nameserver 194.38.96.75
```

A */etc/network* fájl tartalma:

```
NETWORKING=yes
FORWARD_IPV4="yes"
HOSTNAME="lnx.hirep.hu"
DOMAINNAME=hirep.hu
GATEWAY="145.236.225.49"
GATEWAYDEV=""
NISDOMAIN=""
```

Az osztott internethasználat Linux-oldali beállítása

Az osztott internethasználatot az IP-címláncolás (IP Chain) segítségével oldjuk meg. Ennek rengeteg angol és magyar nyelvű leírása létezik (lásd irodalomjegyzék), ezért most csak a szolgáltatást használó két parancsfájlt adjuk meg (mindkettő a Mandrake Linux része, nem saját alkotás).
Az első, a `firewall-masq.sh` nevű parancsfájl (*1. lista*) megteszi azokat a beállításokat, ami után lehetővé válik az osztott internethasználat.
A parancsfájl lefutása és a PPP-kapcsolat felépülése után az ***lnx.hirep.hu*** internetes átjáró lesz a belső windowsos gépek számára. Eme osztott internethasználat kifogástalan működését bármelyik helyesen beállított (a pontos beállítást lásd később) windowsos gép parancsablakából kiadott `ping www.linux.org` parancs visszaigazolja.
Ezt az osztott használati lehetőséget a *2. listában* látható parancsfájl képes megszüntetni.
Ezzel a belső Windows-gépek intenetelérési lehetősége megszűnik. Természetes az ***lnx.hirep.hu*** gép még most is látja az Internetet, de a windowsos gépek csomagjait többé nem továbbítja a külvilág felé.

A külső levelek letöltése a Fetchmail program használatával

A Fetchmail program képes rá, hogy a POP3-protokoll használatával leveleinket a helyi postafiókokba töltse le.
Az rpm-csomag telepítése után a */etc/fetcmailrc* beállítófájlt

3. lista A hirepmail.sh

```
#! /bin/sh
DB=`ifconfig | grep "ppp" | wc -l`
echo "Levelek lekØrØse indul: "
echo `date`

if [ ${DB} = 0 ]; then
   echo "A PPP-csatolÆs elind tÆsa..."
   ifup ppp0
   /etc/ppp/firewall-masq
   fetchmail
   echo "A levelek let ltve..."
   echo "A PPP-kapcsolatot bontom 180
   ➥mÆsodperc mælva..."
   sleep 180

   # VÆrunk, hogy a levØlk ldØsi sort
   # az SMTP ki r tse
   W = `mailq | grep "Mail queue is empty"
   ➥  | wc -w`
   while [ $W = 0 ]
     do
          sleep 60
          W = `mailq | grep
          ➥"Mail queue is empty" | wc -w`
     done
   # VÆrakozÆs vØge

   ifdown ppp0
   kill -9 `ps -ax | grep "ppp-watch"
   ➥  | awk '{print $1 }' | sed 1q`
   echo "A PPP-kapcsolat lebontva..."
else
   echo "A PPP-csatol  mÆr mßk dik..."
   fetchmail
   echo "A levelek let ltve...
   ➥A PPP-kapcsolat tovÆbb Øl..."
fi
exit 0
```

kell helyesen kialakítanunk. Minden felhasználónak egy helyi ***.fetchmailrc*** leírófájlja is lehet, mely esetben a Fetchmail program futtatásakor az ott meghatározott forgatókönyv fog végrehajtódni. Nézzük meg, hogy néz ki egy általános Fetchmailrc fájl:

```
poll mail.matavnet.hu
protocol POP3
localdomains lnx.hirep.hu
no dns
no envelope
user info with password "Itt_a_jelsz _van" to
     hirep
     info
     reklamacio
     user1
     user2
     ...
     user_utolso
here
```

A `poll...` azt mondja meg, hogy postafiókunk a ***mail.matavnet.hu*** ISP-nél található. A második sorban afelől rendelkezünk, hogy a levelek letöltése a POP3 szerint történjen. Ezután az ***info*** postafióknév és jelszavának megadása jön. Itt fontos megjegyezni, hogy az ***info@hirep.hu*** az alapcím, a többi címünk ennek az aliasneve. Az azonosítási eljárás után elkezdődik a levelek letöltése. Ekkor jut szerephez, hogy összerendeltük az ISP- és a saját fiókneveinket. Egy ***inyiri@hirep.hu*** címre küldött levél az ***lnx.hirep.hu*** kiszolgáló ***inyiri*** postafiókjába töltődik. Amennyiben olyan letöltött levél is van, aminek az ***lnx@hirep.hu*** kiszolgálón nincs azonos nevű postafiókja, akkor az a szabály, hogy a levél mindig annak a felhasználónak a postaládájába töltődik, aki a Fetchmail programot elindította. Megjegyzés: a fenti beállításfájlban az eredeti felhasználók nevét user1, user2, ..., user_n-re írtam át.

4. lista A crontab-ütemezés

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
01 07 * * 1-5 root hirepmail.sh >> /var/log/hirepmail.log
01 09 * * 1-5 root hirepmail.sh >> /var/log/hirepmail.log
01 11 * * 1-5 root hirepmail.sh >> /var/log/hirepmail.log
01 13 * * 1-5 root hirepmail.sh >> /var/log/hirepmail.log
01 15 * * 1-5 root hirepmail.sh >> /var/log/hirepmail.log
01 20 * * 1-5 root hirepmail.sh >> /var/log/hirepmail.log
45 16 * * 0   root hirepmail.sh >> /var/log/hirepmail.log
# Mandrake-Security : if you remove this comment, remove
➥the next line too.
```

A levélletöltő parancsfájl felépítése

Ennyi előzetes után nézzük meg a *3. listán* látható levélletöltő parancsfájlt.
Ez a parancsfájl a Fetchmail lefuttatásával a leveleket úgy tölti le, hogy előtte – ha szükséges – az internetkapcsolatot is felépíti. Amennyiben az internetkapcsolat már létezik, úgy a levelek letöltése után nem szakítja meg, viszont ha a kapcsolatot a `hirepmail.sh` parancsfájl építette fel, akkor a `ppp0` csatolót is megszünteti, azaz bontja kapcsolatot a Világhálóval. Két fontos dolgot érdemes a parancsfájl kapcsán kiemelni. Az egyik az, hogy a Fetchmail blokkoltan fut, azaz a parancsfájl végrehajtása a Fetchmail utáni sorral csak akkor folyatódik, ha ténylegesen letöltöttük az összes levelünket. A másik érdekesség a `mailq` parancs által vezérelt `while` ciklus. Erre azért van szükségünk, hogy a PPP-kapcsolat addig ne bontódjon, amíg a kiszolgálónkról a külvilágnak szánt összes levél nem továbbítódik. A `mailq` parancs feladata egyébként az, hogy kilistázza a Postfix-kiszolgáló által még nem továbbított, de továbbításra váró leveleket.

A levélletöltés önműködővé tétele

A következő feladat az, hogy a `hirepmail.sh` parancsfájlt valakinek valamilyen időközönként meg kell hívnia. Ezt jelenleg a */etc/crontab* fájlban meghatározott crontab-ütemezés teszi meg (lásd *4. lista*).
A */etc/crontab* fájlban azt is meg kell mondani, hogy a program kinek a nevében indul el. E helyütt a rendszergazdát adtuk meg. A ***hirepmail.log*** naplófájlban az egyes letöltések folyamatosan figyelemmel kísérhetők. Megjegyzés: a `startinet.sh`, `stopinet.sh` és a `hirepmail.sh` fájlokat a */usr/bin* könyvtárba célszerű másolni, mert onnan mindenki (a crontab is) könnyen elérheti. Az ütemezés jelenleg olyan, hogy hétköznapokon (ahol 1–5 van) napi 6 alkalommal, hétvégén pedig csak egyszer, 16 órakor töltődnek le a levelek. Természetesen leveleinket kézzel, a `hirepmail.sh` parancs segítségével bármikor letölthetjük, illetve elküldhetjük.

A Postfix levélkiszolgáló beállítása

A Postfix rpm-csomag telepítése után csak két apró változtatást kell a */etc/postfix/main.cf* fájlban megtennünk.

```
# Ezen a gØpen kereszt l k ldj k a leveleket
# az SMTP-vel:
relayhost = mail.matavnet.hu
# A levelek addig vÆrakozzanak a k ldØsre,
# am g nincs PPP-kapcsolat:
defer_transports = smtp
```

Megjegyzés: minden új linuxos felhasználó egyben levelezési címet is jelent. A „ppapai” linuxos felhasználó felvételével egyben egy ***ppapai@lnx.hirep.hu*** levélcím is születik. A postafiókok fizikailag a */var/spool/mail* könyvtárban egy-egy fájl formájában jelennek meg. További részleteket a man Posfix parancs kiadásával manlapjairól és a programcsomaggal együtt szállított leírásból tudhatunk meg. Ennek ellenére a fentieknél több tájékoztatásra nincs szükségünk, ha a Postfixet csak egy fekete doboznak képzeljük el.

Az IMAP-démon beállítása

Ennek a kiszolgálófolyamatnak az a feladata, hogy az IMAP-ügyfeleket (Outlook, Outlook Express) kiszolgálja. Az rpm-csomag telepítése után semmilyen további beállításra nincs szükség. Nézzük meg a szolgáltatáskezelő vagy a `ps -ax | grep imap` paranccsal, hogy fut-e ez a kiszolgáló. Szükség esetén ezt a szolgáltatást állítsuk működőre, mert a windowsos levelezőügyfelek igényelni fogják.

A Windows-munkaállomások beállítása

A Windows-munkaállomások beállítása egyrészt a TCP/IP-protokoll, másrészt az Outlook vagy az Outlook Express beállításából áll.
A TCP/IP-beállításnál a következő értékeket kell megadnunk:

- a gép IP-címe 192.168.1.x (netmask=255.255.255.0) legyen,
- az átjáró (gateway) az ***lnx.hirep.hu*** belső címe: 192.168.1.100,
- a DNS-kiszolgálók: 195.228.240.248 és 194.38.96.75 (természetesen más cím is lehet, de a mi ISP-nk ezt adta),

- a *hosts* fájlba a következő sort szúrjuk be: 192.168.1.100 *lnx lnx.hirep.hu*.

A beállítási folyamat után a `ping lnx` parancsnak jeleznie kell, hogy a névfeloldás működik.
Az Outlook Express beállításának értékei a következők legyenek (a postafiók neve most inyiri):

- az SMTP-kiszolgáló neve: *lnx*,
- a levélcím: *inyiri@hirep.hu*,
- a levelezési protokoll: IMAP (vigyázat, ne a POP3-at válasszuk!),
- a fiók neve: inyiri,
- a jelszót ne adjuk meg, jobb, ha a fiókunk elérése során minden esetben újra és újra megadjuk.

Állítsuk be azt is, hogy az Outlook Express induláskor a leveleket ne próbálja letölteni, illetve elküldeni.
Az ügyfélgépeken a saját használatú fiókok elérése mellett mindenki elérheti a közösen használt „info”, „reklamacio” és „hirep” postafiókokat is, így ezeket is fel kell venni az Outlookba.
Megjegyzés: a jelenlegi levelezési rendszer kis hiányossága, hogy a belső levelezést jobb a *@lnx.hirep.hu* címmel megadni, mert így a levelek egyből továbbítódnak. Erre azonban a „válasz” szolgáltatás nem működik, hiszen a feladó levélcímek mind *@hirep.hu* végűek, ezért a „továbbítás” használata a megoldás. Természetesen belső leveleinket a *@hirep.hu* címmel is küldhetnénk, de akkor lassan kezelődnek, bár a „válasz” működni fog.

Az lnx.hirep.hu szolgáltatásainak közvetlen használata

A fenti beállítás már biztosítja a levelezés önműködő használatát. A gyakorlatban azonban az alábbi feladatok megvalósítására is szükség van:

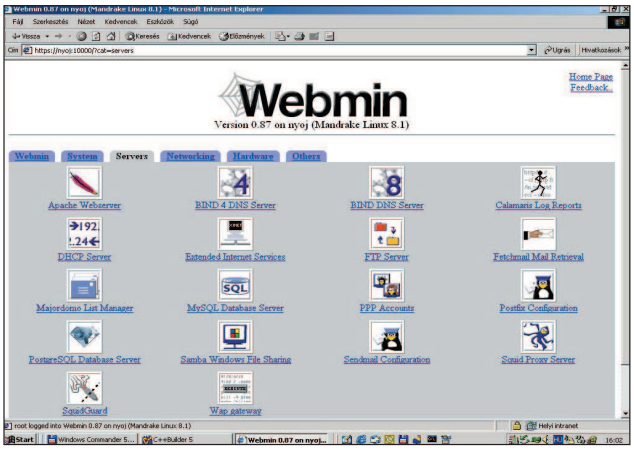
- internethasználat bármelyik Windows-gépen,
- azonnali levélletöltés vagy -elküldés (sürgős eset kezelése).

A cégnél két kiemelt felhasználó létezik, akik ismerik a rendszergazdai jelszót. Ők a `telnet lnx`, utána `su` parancs együttes használata segítségével kiadhatják a `startinet.sh`, `stopinet.sh`, illetve a `hirepmail.sh` parancsokat.

Egyéni postafiókok használata

Több alkalmazott is rendelkezik a freemail, hotmail, mailbox... helyeken kialakított postafiókokkal. Ennek kezelésére a helyi *.fetchmailrc* használható. Az inyiri felhasználónak például fiókja van a következő két címmel: *nyimre@freemail.hu* és *nim@mailbox.hu*. Ekkor a */home/inyiri/.fetchmailrc* tartalmát a következőre érdemes állítani:

```
poll freemail.hu
protocol POP3
localdomains lnx.hirep.hu
no dns
no envelope
user nyimre with password "Itt_a_jelsz _van" to
     inyiri
here

poll mailbox.hu
protocol POP3
localdomains lnx.hirep.hu
no dns
```




```
no envelope
user nim with password "Itt_a_jelsz _van" to
     inyiri
here
```

A fenti vezérlő parancsfájlban az inyiri felhasználó beállítása csak dokumentációs célokat szolgál, mert sem a „nyimre”, sem a „nim” fióknév nem fog megegyezni az „inyiri” helyi fióknévvel. Ekkor azonban a fetchmail azon szabálya lép életbe, hogy annak tölti le a leveleket, aki a programot elindította. Ennek megfelelően a saját külön leveleink letöltése a következő négy lépésből áll:

1. Az arra feljogosított munkatársunkat megkérjük, hogy indítsa el a `startinet.sh` parancsfájlt (bár elképzelhető, hogy épp nekünk is megvan hozzá a jogunk).
2. A `telnet lnx` (például inyiri felhasználóként).
3. A `fetchmail` parancs lefuttatása.
4. Az internetkapcsolat megszüntetése.

Továbblépési lehetőségek

A UPC Chello szolgáltatása a vállalat földrajzi környezetében még az idei év során elérhető lesz. A jelenlegi ISDN-kapcsolat helyett a Chello 24 órás internetelérési szolgáltatását tervezzük megvásárolni. Ebben az esetben nem lesz rá többé szükség, hogy postafiókjainkat az ISP-től kapjuk, így az e-mail mapping megszüntethető, és leveleinket közvetlenül – az SMTP használatával – a saját kiszolgálónkra fogjuk fogadni. Jelenlegi levelezési rendszerünk előnye, hogy erre a korszerűbb és rugalmasabb szolgáltatásra már most teljesen fel van készítve, így az átállás valószínűleg egyszerű lesz.

Irodalomjegyzék

1. ➲ http.//www.linux.org – Linux Documentation Project: Linux network administrators's guide (2nd edition)
2. Fred Butzen-Christopher Hilton: Linux-hálózatok
3. A *Linuxvilág* 8–11. számaiban megtalálható Postfix-, IMAP- és POP3-cikkek

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

Sendmail: alapismeretek és beállítás

Útmutató azoknak, akik most állítják üzembe első levelezőkiszolgálójukat.

Az Internet folyamatos növekedésével az elektronikus levelezés egykettőre az információk széles körben való terjesztésének legfontosabb eszközévé vált. Ahogy növekszik a gyors, olcsó és megbízható elektronikus levelezés igénye, úgy fordul egyre több személy a Linuxhoz, mely gyors, olcsó és megbízható megoldást nyújt.
A Sendmailt eredetileg ***Eric Allman*** fejlesztette ki 1979-ben `delevermail` néven, és először a BSD 4.0 része volt. Nem volt túl rugalmas, és a fordítása közben kellett végrehajtani a beállításokat. Ahogy növekedett a TCP-protokoll jelentősége, és más tényezőknek köszönhetően is nyilvánvalóvá vált, hogy a `delevermail` kevés ahhoz, hogy eleget tegyen az új kívánalmaknak. ***Eric Allmannak*** az alapoktól kiindulva újból meg kellett alkotnia a Sendmailt, és munkájának eredménye minden MTA-k anyja lett. A Sendmail nem utasítja el azokat az üzeneteket, amelyek nem felelnek meg a protokolloknak – úgy lett megtervezve, hogy tűrőképessége igen nagy legyen ezekkel az üzenetekkel szemben. Azoknak, akik még sohasem állítottak üzembe levelezőkiszolgálót, cikkünk bemutatja, hogyan is kell beállítani a Sendmail 8.11.2-es változatát a Red Hat Linux 7.1 friss telepítése után.
A Red Hat Linux 7.1 telepítése közben alapértelmezésként a Sendmail 8.11 is feltelepül. A Red Hat az eltelt évek során elért fejlődésének köszönhetően a telepítési folyamat rendkívül egyszerű. Bár ez a cikk nem foglalkozik a telepítési mód részleteivel, a Red Hat CD-s telepítőkészletében további tájékoztatás található.
Ahhoz, hogy új levelezőkiszolgálónkat működésre bírjuk, először a DNS-sel kapcsolatos beállításokat kell elvégeznünk. Elsőként is az új levelezőkiszolgáló nevét és IP-címét adjuk hozzá a DNS-kiszolgálóhoz, majd az `nslookup` segítségével ellenőrizzük a címet:

```
[root@testmail /root]# nslookup -sil
➥testmail.blank.com
Server:         192.168.100.1
Address:        192.168.100.1#53
Name:   testmail.blank.com
Address: 192.168.100.134
```

Az is lényeges, hogy a rendszergazda végezze el a fordított DNS-beállításokat, megakadályozva, hogy a levelek kiküldése késedelmet szenvedjen. A korszerű levelezőkiszolgálók többsége a levelek továbbításánál ellenőrzésként a fordított feljogosítást (reverse lookup) használja. Ennek a beállításnak a helyességét is ellenőrizzük úgy, hogy az IP-címre vonatkozóan kiadjuk az `nslookup` parancsot:

```
[root@testmail /root]# nslookup -sil
192.168.100.134
Server:         192.168.100.1
Address:        192.168.100.1#53

134.100.168.192.in-addr.arpa    name =
➥TESTMAIL.blank.com.
```

Mint látjuk, a DNS-bejegyzések a helyükön vannak és megfelelően működnek, ezért továbbléphetünk, és elkezdhetjük a Sendmail tényleges beállítását. Az alapbeállítások szerint a Red Hattel szállított Sendmail az SMTP-forgalmat csak a helyi gépen engedi meg. A `netstat -nl` parancs kimenete (*1. lista*) minden olyan kaput megmutat, amelyen démon figyel; tekintsük csak meg a `127.0.0.1:25` sort. Ez azt jelenti, hogy a kiszolgáló a 25-ös (SMTP) kapura érkező kapcsolatokat csakis a visszacsatoló címen figyeli.
Ez meggátolja, hogy a leveleződémon a helyi gépet kivéve bármely más helyről elektronikus leveleket fogadjon el. Hogy megoldjuk ezt a gondot, meg kell mondanunk a Sendmailnek, hogy a kapcsolatokat a külső csatolón is figyelje. Új kiszolgá-

1. lista A „nestat -nl” kimenete

```
[root@testmail /root]# netstat -nl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address State
tcp        0      0 0.0.0.0:32768           0.0.0.0:* LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:* LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:* LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:* LISTEN
udp        0      0 0.0.0.0:32768           0.0.0.0:*
udp        0      0 0.0.0.0:667             0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
Active UNIX domain sockets (only servers)
Proto RefCnt  Flags     Type     State      I-Node   Path
unix  2       [ ACC ]   STREAM   LISTENING  1119     /dev/gpmctl
unix  2       [ ACC ]   STREAM   LISTENING  1172     /tmp/.font-unix/fs7100
```

lónk esetében csak egy ethernet hálózati kártya van, és `eth0` a külső csatoló elnevezése. Ellenőrizzük az `eth0` csatoló IP-jét, ehhez mindössze az `ifconfig` parancsot kell lefuttatnunk (*2. lista*). A beállításoktól függően elképzelhető, hogy az IP más, mint a DNS-kiszolgáló által beállított cím, példánkban azonban a címek megegyeznek.

2. lista Az inconfig parancs kimentése

```
[root@testmail /root]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:60:97:DE:E9:99
          inet addr:192.168.100.134  Bcast:192.168.100.255
          Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12421 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0xe000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Ennek a gépnek az `eth0` csatolón `192.168.100.134` a címe. Amikor megállapítottad, mi a pontos cím, a */etc/sendmail.cf* állományt szerkeszd át, és a Sendmail démont úgy állítsd be, hogy ezt a címet figyelje:

```
# SMTP daemon options

O DaemonPortOptions=Port=smtp,Addr=127.0.0.1,
➥Name=MTA
```

A fenti sort a következőképpen kell megváltoztatnod:

```
O DaemonPortOptions=Port=smtp,
➥Addr=192.168.100.134, Name=MTA
```

Ha ezzel végeztél, mentsd a fájlt, és a `/etc/init.d/sendmail` parancs segítségével indítsd újra a Sendmail démont:

```
[root@testmail /root]# /etc/init.d/sendmail
➥restart
Shutting down sendmail:                    [  OK  ]
Starting sendmail:                         [  OK  ]
[root@testmail /root]#
```

Most a `netstat -nl` parancs segítségével ellenőrizd (*3. lista*), hogy megtörtént-e a változás. Mint látod, a kimenet egyértelműen mutatja, hogy a (Sendmail) démon az `eth0` csatolónak kiosztott `192.168.100.134`-es IP-cím 25-ös kapujára figyel.
Most, hogy a Sendmail már elfogad kívülről jövő kapcsolatokat, be kell állítanunk, hogy mely tartományneveket fogadhatja el. Ennek megadására szolgál a */etc/mail/local-host-names* állomány. Csak annyit kell tenned, hogy beírod ebbe az állományba a tartomány nevét, a ***blank.com***-ot:

```
# local-host-names - include all aliases for
# your machine here.
blank.com
```

Ha a fenti adatokat mentettük ebbe a fájlba, a Sendmail démont a */etc/init.d/sendmail restart* parancsfájllal újra kell indítanunk. A Sendmail több tartomány számára érkező elektronikus leveleket is el tud fogadni ugyanazon a kiszolgálón. Ha új tartományt szeretnénk felvenni, a tartományok nevét be kell illesztenünk ebbe a fájlba.
Most már jól működő levélkiszolgálónk van a helyi gépen. Képes elektronikus leveleket fogadni bárhonnan a világból, de csak a helyi gépről tud leveleket küldeni, illetve továbbküldeni. Egy másik alapértelmezésben szereplő biztonsági beállítás nyomán a Sendmail nem feltétlenül engedélyezi egy levélnek a továbbküldését sem, megakadályozva ezáltal, hogy kiszolgálónkról levélszemetet (kéretlen reklámleveleket – spamet) küldhessenek. Amennyiben a felhasználók közvetlenül a kiszolgálóra jelentkeznek be, ezt a beállítást nem szükséges módosítani. Ha azonban a szervezet felépítése olyan, mint a legtöbbé, akkor a felhasználók távoli helyszínekről is használják az elektronikus levelezést. Ha a felhasználók olyan levelező ügyfélprogramokat használnak, mint például a Kmail vagy az Outlook Express, meg kell engednünk, hogy ezek a gépek az új kiszolgálón keresztül küldhessék a leveleket, de semmiképpen nem szeretnénk teljesen megnyitni a gépünket, hogy korlátlanul lehessen rajta keresztül leveleket küldeni. Ezért az alábbi sort kell beleírnunk a */etc/mail/access* állományba, majd mentése után le kell futtatnunk a `make access.db` parancsot (*4. lista*).
A `make access.db` parancs lefuttatásával az új beállítás az ellenőrző adatbázisban mentődik, melynek segítségével a Sendmail meghatározza, ki is küldhet tovább elektronikus leveleket kiszolgálónk közvetítésével. Így hát a ***blank.com*** tartományon belülről jövő kapcsolatok leveleiket új levélkiszolgálónkon keresztül küldhetik tovább, a kívülállók viszont nem tudják majd ezt a szolgáltatást használni. Alhálózati IP-tartományt (például `192.168`) is megadhatunk, hogy így korlátozzuk a felhasználhatóságot a tartományon belülre. Ne feledjük, hogyha ezt a beállítást korlátozás nélkülire állítjuk, a levélszemétküldők hatalmas mennyiségű levelet küldhetnek a rendszerünkön keresztül!
Most, hogy a világon bárhonnan tudunk elektronikus leveleket fogadni, beállítottuk a tartományt és engedélyeztük a levelek továbbküldését az erre jogosult ügyfeleknek, itt az idő, hogy a levelek távolról történő elérését is engedélyezzük. Ez az IMAP vagy a POP segítségével valósítható meg. A kiszolgáló alapbeállítás szerinti telepítésekor nem minden olyan csomag kerül fel a rendszerre, amely a POP, illetve az IMAP működéséhez szükséges. Ezek a szolgáltatások az imap-2000-9 rpm-csomag telepítésével válnak elérhetővé. Nézzük meg, hogy ez a csomag fel lett-e telepítve, tehát adjuk ki a követ-

3. lista Változás ellenőrzése

```
[root@testmail /root]# netstat -nl
Active Internet connections (only servers)
Proto Recv-Q send-Q Local Address
                    ➥Foreign Address State
tcp        0       0 0.0.0.0:32768
                    ➥0.0.0.0:* LISTEN
tcp        0       0 0.0.0.0:111
                    ➥0.0.0.0:* LISTEN
tcp        0       0 0.0.0.0:22
                    ➥0.0.0.0:* LISTEN
tcp        0       0 192.168.100.134:25
                    ➥0.0.0.0:* LISTEN
udp        0       0 0.0.0.0:32768
                    ➥0.0.0.0:*
udp        0       0 0.0.0.0:667
                    ➥0.0.0.0:*
udp        0       0 0.0.0.0:111
                    ➥0.0.0.0:*
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State
                    ➥I-Node Path
unix  2      [ ACC ]     STREAM
      ➥LISTENING     1119   /dev/gpmctl
unix  2      [ ACC ]     STREAM
      ➥LISTENING     1172
/tmp/.font-unix/fs7100
[root@testmail /root]#
```

kező parancsot: `rpm -aq | grep -i imap`. Ha nem találunk ilyen csomagot, helyezzük be a CD-ROM-meghajtóba a Red Hat 7.1 második telepítő lemezét, és fűzzük be a lemezt. Ehhez adjuk ki a `mount /dev/cdrom /mnt/cdrom` parancsot:

```
[root@testmail mail]# mount
➥/dev/cdrom /mnt/cdrom
mount: block device /dev/cdrom is
➥write-protected, mounting read-only
(Successful Mount of Read-Only Media)
```

Ha sikerült a lemezt befűznünk, az

```
rpm -Uvh /mnt/cdrom/RedHat/RPMS/
➥imap-2000-9.i386.rpm
```

paranccsal telepíthetjük a csomagot.

```
[root@testmail mail]# rpm -Uvh
/mnt/cdrom/RedHat/RPMS/imap-2000-9.i386.rpm
Preparing...
########################################
[100%]
   1:imap
########################################
[100%]
```

Mint látjuk, ha most kiadjuk az rpm keresésére szolgáló `rpm -aq | grep -i imap` parancsot, kimenetként az IMAP-csomagot kapjuk:

4. lista

```
blank.com                   RELAY

# Check the /usr/share/doc/
➥sendmail-8.11.2/README.cf file for
# a description
# of the format of this file.
➥(search for access_db in that file)
# The /usr/share/doc/sendmail-8.11.2/
➥README.cf is part of the
sendmail-doc
# package.
#
# by default we allow relaying from
# localhost...
localhost.localdomain            RELAY
localhost                         RELAY
127.0.0.1                         RELAY
blank.com  RELAY

[root@testmail mail]# make access.db
[root@testmail mail]#
```

```
[root@testmail mail]# rpm -aq | grep -i imap
imap-2000-9
[root@testmail mail]#
```

A megfelelő csomagok telepítését követően engedélyeznünk kell, hogy az új levelezőkiszolgálóhoz POP3-mal is lehessen kapcsolódni. Ezt úgy valósíthatjuk meg, hogy az ***ipop3*** nevű állományt a ***/etc/xinetd.d*** könyvtárban átírjuk. A `disable` (kikapcsolva) kapcsolónál szereplő értéket írjuk át no-ra (nem), és mentsük a fájlt. Fontos, hogy a kis- és nagybetűknek olyanoknak kell lenniük, ahogyan a fájlban szerepelnek:

```
# default: off
# description: The POP3 service allows remote
# users to access their mail

#               using an POP3 client such as
# Netscape Communicator, mutt,

#               or fetchmail.
service pop3
{
  socket_type        = stream
  wait               = no
  user               = root
  server             = /usr/sbin/ipop3d
  log_on_success     += USERID
  log_on_failure     += USERID
  disable            = no
}
```

Most a `xinetd` démont újra kell indítani, ugyanis az új beállítás csak ekkor fog életbe lépni. Ehhez az alább látható módon a `restart` (újraindítás) parancsot az /etc/init.d/xinetd nevű parancsállományra vonatkozóan ki kell adnunk:

```
[root@testmail xinetd.d]# /etc/init.d/xinetd
➥restart
Stopping xinetd:                           [  OK  ]
Starting xinetd:                           [  OK  ]
[root@testmail xinetd.d]#
```

Most küldj egy próbalevelet az új kiszolgálóra, és kedvenc POP-ügyfélprogramoddal kapcsolódj a kiszolgálóhoz. Ha minden jól ment, leveleidhez most már a POP-protokollon keresztül is hozzá tudsz férni.
Végezetül az új kiszolgálóval kapcsolatban a teljesítményről is szót kell ejtenünk. Amennyiben az ügyfelek a kapcsolatokat egy tűzfal mögül kezdeményezik, lehetséges, hogy panaszkodni fognak a lassú POP-kiszolgálós kapcsolat miatt. Ennek oka, hogy levelezőkiszolgálónk IDENT-kapcsolatot kezdeményez az ügyfelekkel, így ellenőrizve az ügyfél azonosságát. Ha erre a kérésre nem érkezik válasz, a kiszolgáló csak némi holtidő (timeout) után válaszol, ami alapesetben öt másodpercre van állítva. Ezt az értéket 1 másodpercre is lehet csökkenteni, így az IDENT által okozott késedelem nagy része megszüntethető. Ha ezt az értéket meg szeretnénk változtatni, a */etc/sendmail.cf* állományt át kell szerkesztenünk, és a holtidőnél szereplő értéket a kívánt értékre kell változtatnunk:

```
# timeouts (many of these)

#O Timeout.ident=5s

ezt a k vetkezıre kell Æt rnunk:

O Timeout.ident=1s
```

Most tehát levelezőkiszolgálónk működik, és szolgáltatásokat nyújt a felhasználóknak. A Sendmailnél még számos olyan beállítási lehetőség van, melyek meghaladják ennek a cikknek a kereteit. A Linux nagyon megbízható teherbíró felületet nyújt az elektronikus levelezés lebonyolításához. Ha további tájékoztatást szeretnél a Sendmailről, keresd fel a ➲ http://www.sendmail.org címet.

A Linux Journal honlapjáról ➲ http://www.linuxjournal.com

Eric Jorn Seneca
Unix-rendszermérnökként dolgozik a louisianai Baton Rouge-ban.

Linux-tábor Szerencsen

2002 nyarán immár második alkalommal vehetnek részt az érdeklődők a Linux-felhasználók Magyarországi Egyesületének támogatásával rendezett Linux-táborban, Szerencsen.
Az egyhetes táborokon két váltásban, június 30–július 7., valamint július 8–13. között lehet résztvenni. A tábor programjának középpontjában az érdeklődőknek a Linux terén szerzett ismereteinek bővítése áll. A szerencsi gimnázium számítógéptermeiben rendezik a délelőtti tanfolyamokat, melyeken a tábor lakói saját gépeiket is használhatják, ebben az esetben a rendezők segítenek a Linux telepítésében. A Linux ismerete egyébként nem feltétele a jelentkezésnek.
A tanfolyam anyagának elsajátításában országosan elismert linuxos szakembereik segítenek, a tematikát az igényeknek megfelelően alakítják ki.
A résztvevőket a gimnázium épületéhez közeli Szerencsi Középiskolai Kollégiumban helyezik el. Itt természetesen lehetőség nyílik internetezésre, levelek olvasására is. Kulturált elhelyezést ígérnek a rendezők 2, 3 és 4 ágyas szobákban.
A nap további részében számos egyéb program is várja az érdeklődőket: többféle sportolási lehetőség, esténként borozás a helyi borospincékben – Szerencs a Tokajhegyalja tájegység része –, és az egyhetes programot, immár hagyományosan, bográcsozás zárja. A táborozók rokonokat, barátokat,

gyerekeket is hozhatnak, a szakmai programok időpontjában az ő idejüket is megszervezik a tábor vezetői, többek közt biciklitúrákra is lehetőség adódik Szerencs környékén.
A 2001-es tábor sikerén felbuzdulva a szervezők idén nyáron egy helyett két csoportot szerveznek, bár a második alkalmat csak akkor tartják meg, ha elegendő résztvevő jelentkezik rá.
A részvételi díj tartalmazza a szállást, a napi háromszori étkezést, a tanfolyamokat, valamint a borkóstolót. Amennyiben május 30-ig jelentkeznek az érdeklődők, 19 000 Ft-ért vehetnek részt a Linux-táborban, az LME tagjai kedvezményt is kapnak. Akit csak a szakmai program érdekel, és a szállást, étkezést maga kívánja megszervezni, annak a szervezők – május 30-ig – mindössze 7000 Ft-ért teszik lehetővé a tanfolyam látogatását. Szívesen várnak gyermekeket is szülői kísérettel, akik féláron vehetnek részt a táborban.
Milyen ismereteket nyújtanak az érdeklődőknek a délelőtti tanfolyamok? Mint 2001-ben, idén is alap, haladó és profi szintű témaköröket mutatnak be az oktatók.
Az első nap a Linux telepítésével kezdődik, így a legkevesebb ismerettel rendelkezőket az alapoknál kezdik bevezetni az ismeretekbe. Az alapszintű témák közé tartozott 2001-ben, és várhatóan idén is a Linux-alapokkal valamint, a hálózattal, a betárcsázó rendszerrel való ismerkedés. A haladó témák iránt érdeklődők megismerkedhetnek többek közt a rendszerbiztonság, az alapvető hálózati és belső biztonság, a levelezőkiszolgáló, a proxykiszolgáló témaköreivel. A profi témák a következők: webkiszolgáló, hálózati biztonság magasfokon, belső biztonság felsőfokon. A fenti ismeretkörök hallgatóság érdeklődéséhez vannak igazítva, mivel a tavalyi jelentkezéskor leadott igények alapján alakították ki őket a tábor szervezői.
További tájékoztatás és a jelentkezési lap a ➲ http://linuxtabor.webhome.hu címen található. Ugyanitt a tábor szervezői minden felmerülő kérdésre választ adnak.

Szabó Ágnes

Ördögűzés

Sorozatunk előző részében betekintést nyerhettünk a Linux alatti fájlkezelés rejtelmeibe. Most a háttérben futó programok kezelési lehetőségeivel ismerkedünk meg.

A Linux többfeladatos operációs rendszer. Ez mindössze annyit jelent, hogy egyszerre több programot, alkalmazást képes futtatni. Valójában szó sincs valódi párhuzamos futtatásról, egyszerre csak egy program fut, de egy ütemező algoritmus segítségével a rendszer – bizonyos idő elteltével – mindig egy másik, szintén futásra kijelölt alkalmazásnak adja át a futási lehetőséget.

Ez az ütemező eljárás meglehetősen bonyolult, működésének ismertetése meg is haladná e cikk kereteit. Ezért legyen elég annyi, hogy ezt a feladatot is a rendszermag látja el. Természetesen, mint minden valamire való többfeladatos rendszerben, a Linux sem hagyja szabadon garázdálkodni a futó programokat. Az alkalmazás futásának teljes ideje alatt komoly megszorítások vonatkoznak rá, és ha csak az egyiket is áthágja, egyből kinn találhatja magát a rendszer memóriájából.

Írásomban a párhuzamosan futtatott alkalmazások irányításával kezelésével, foglalkozunk, végezetül pedig pár szót ejtünk egy a Unix világában kulcsfontosságú fogalomról: a démonokról.

A Linux többfeladatos, úgynevezett multitaszkos operációs rendszer. Ez mindössze annyit jelent, hogy egyszerre több programot és alkalmazást képes futtatni. Valójában mindig csak egyetlen program fut, de a rendszer egy ütemező algoritmus segítségével – bizonyos idő elteltével – mindig egy másik, szintén futásra kijelölt alkalmazásnak adja át a futási lehetőséget.

Azt a legkisebb egységet, amely egy ilyen rendszerben párhuzamos feldolgozásra kerülhet, folyamatnak (process), illetve feladatnak (task) hívjuk. Egyszerűbben: ha bármilyen programot elindítunk, a rendszer létrehoz egy új folyamatot, amelyben az alkalmazás a többi, már meglévő folyamattal együtt fut majd.

Az éppen futó folyamatok listáját a `ps` parancs segítségével kaphatjuk meg. Ha ezt az utasítást kapcsolók nélkül adjuk ki, csak az általunk elindított programok kerülnek listázásra. Ha kicsit bővebb adatokra vágyunk, például az összes folyamatot meghatározott adataival együtt szeretnénk látni – például ki és melyik terminálon futtatja és a többi –, a `ps aux` parancsot használjuk. Az így kapott, általában többképernyős listában az utolsó oszlop tartalmazza a futó programok neveit. Az első oszlop azt a felhasználót jelöli, aki az alkalmazást elindította. Számunkra a második oszlop lesz még igazán fontos, ugyanis ez tartalmazza az úgynevezett PID-et (Process ID), ami nem más, mint egy azonosítókulcs, amely az adott folyamatot azonosítja. Minden folyamatnak egyedi azonosítója van. Ha egy folyamat egy kódszámot kapott, azt a kódszámot innentől kezdve egyik folyamat sem kaphatja meg (egészen a rendszer újraindításáig), még akkor sem, ha az adott program esetleg régen befejezte a futását.

Sajnos, kevés a tökéletes program, ezért előbb-utóbb számíthatunk rá, hogy valamelyik le fog fagyni. Az ilyen helyzetek eléggé kellemetlenek az olyan „egyfeladatos” rendszerekben, mint amilyen például az MS-DOS: ebben az esetben kizárólag a számítógép újraindításával kezelhetjük a gondot a leghatékonyabban.

Egy valódi többfeladatos rendszernél azonban nem kell ilyen mélyreható eszközökhöz folyamodnunk, mivel egy folyamat nem veszélyeztetheti a többi futó alkalmazást, illetve magát az operációs rendszert. Ha valamelyik programunk netán lefagy vagy rendellenesen működik, egyszerűen csak meg kell kérnünk a rendszermagot, hogy küldjön neki egy üzenetet (signal), amelyben a futás befejezésére szólítja fel. Ezt a `kill` utasítással tehetjük meg. Kapcsolóként a leállítandó folyamat PID-jét kell megadnunk.

Egyes esetekben akkora gubanc is keletkezhet, hogy az adott alkalmazás nem képes az általunk küldött utasítást „kezelni”. Ilyenkor erőszakosabb fellépésre van szükség: a `kill` utasítás után írjunk egy `-9` kapcsolót, például: `kill -9 192`. A `-9` hatására a rendszermag se szó, se beszéd a futó alkalmazást eltávolítja a memóriából. Ez ellen egy felhasználói program sem képes védekezni, tehát jaj annak az alkalmazásnak, amelyik erre a sorsra jut! A `kill` utasítás eredeti feladata egyébként nem az, hogy az éppen futó programjainkat megsemmisítsük a memóriából, hanem az, hogy segítségével különböző rendszerüzeneteket (signal) küldhessünk a folyamatoknak. A `kill` parancs hivatalos írásmódja: `kill -`*`jel`* `PID`.

A `-signal`-hoz a jel nevét (például `KILL`, `TERM` stb.) vagy számát írhatjuk. Amennyiben nem adunk meg jelet, akkor a 15-ös számú `TERM` jel kerül elküldésre. Ez felszólítja az alkalmazást, hogy azonnal zárja be az összes megnyitott állományt és fejezze be a futását.

A `-9` a `KILL` jel küldését teszi lehetővé. Ezt a lehetőséget csakugyan a legutolsó esetben használjuk, ugyanis egészségtelen, ha a nyitott állományokat nem zárják le megfelelően.

A `kill` mellett meg kell említenünk a `killall` utasítást is.

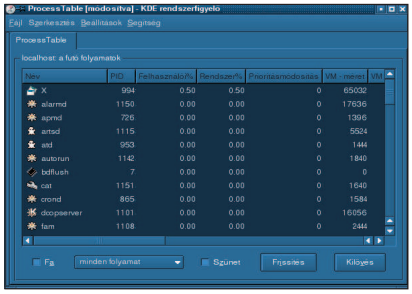
Itt kapcsolóként nem a PID-et, hanem az alkalmazás nevét szükséges megadnunk. Eredményeként az összes ilyen nevű futó program befejeződik. Természetesen, amennyiben ezt az utasítást nem rendszergazdaként adtuk ki, csak az általunk elindított alkalmazásokra lesz hatással.

A folyamatok kezelésére másfajta és egyszerűbb módszer is létezik. Ilyen például a KDE-hez tartozó ***KDE rendszerfigyelő***, de kötelességünk megemlíteni a konzolos `top` nevű programot is. Mélyedjünk el egy kicsit az éppen futó folyamatok listájában! Már a lista méretéből is arra következtethetünk, hogy a háttérben sokkal több minden fut, mint ahány alkalmazást indítottunk.

Észrevehetjük, hogy viszonylag sok a rendszergazda által indított, általában `d` betűre végződő program, például a `kerneld`, `syslogd`, `inetd`, `nfsd` stb. Ezek nem mások, mint az összes Unix-alapú rendszer alappilléreit képző démonok (daemon). A démon görög eredetű szó, eredeti jelentése „közvetítő”. Az elnevezés olyan programokat takar, amelyek a rendszer indulásától egészen az újraindításig futnak, ám idejük nagy részét általában semmittevéssel töltik, csak egy meghatározott esemény bekövetkeztekor lendülnek munkába.

Minden démon kizárólag egy egyéni fe-

ladatot lát el. Például a `syslogd` feladata a rendszerben történő dolgok naplózása. Ha nincs semmi naplóznivaló, „mély álomba szenderül”, de amint újabb feljegyezni való érkezik, azonnal munkába áll. Kiegészítésként megjegyezzük, hogy a



Linuxban (és számos más, többfeladatos operációs rendszerben) a legalapvetőbb műveletek elvégzése is ezen az elven alapszik. Például a lemezre való írásért felelős eljárások is folyamatosan külön feladatként (task) futnak, de érdemi tevékenységet csak akkor végeznek, ha valamit írni kell a merevlemezre. Ezek a feladatok mélyen a rendszermagban, a felhasználói folyamatok alatti szinteken helyezkednek el, és egy felhasználó sem bolygathatja meg őket. A démonok azonban a „felhasználói rétegben” futnak, tehát a rendszergazda bármikor leállíthatja, illetve újra elindíthatja őket.
A Unix-rendszerekben szinte az összes feladatot a démonok látják el. Például a különböző hálózati kiszolgálók (web, FTP stb.) is démonként futnak.
A démonok lelkivilágának jobb megértéséhez a dolgot egy kézzelfogható példával szemléltetem, mégpedig a `cron` démonnal (senkit ne tévesszen meg, hogy a név végén nincs `d` betű!). E démon segítségével a felhasználók által előre meghatározott feladatok elvégzését tehetjük önműködővé. Például beállíthatjuk, hogy a program minden hajnalban 5 órakor hívja meg a `halt` parancsot, amely a rendszer leállását eredményezi (éjszakai letöltögetőknek roppant hasznos szolgáltatás!).
A `cron` démon tehát a Linux elindítását követően betöltődik, ettől kezdve minden percben megnézi az adatbázisában, hogy akad-e valamilyen tennivalója. Ha van, elvégzi a kijelölt feladatot, ha nincs, a következő percig „hibernálja” magát.
Most nézzük meg, hogyan is működik ez a gyakorlatban! Ahhoz, hogy egy felhasználó kiaknázhassa a `cron` démont által nyújtott szolgáltatásokat, azonosítójának szerepelnie kell a */etc/cron.allow* állományban. Ezt a fájlt csak a rendszergazda írhatja! A `cron` démon feladatait az úgynevezett ***crontab*** állomány tárolja. Minden felhasználóhoz külön crontab-fájl tartozik, amely a */var/spool/cron* könyvtár alatt található. A crontab felépítése nagyon egyszerű: minden bejegyzést külön sorba kell írnunk, és egy bejegyzés hat oszlopból áll. Az első öt a feladat elvégzésére kijelölt időpontot jelzi (perc, óra, nap, hónap, a hét napja), az utolsó pedig magát a feladatot. Ha valamelyik időpont helyére *-ot (csillagot) írunk, az minden lehetséges értéket helyettesíteni fog. A „hét napja” oszlopnál a nulla jelenti a vasárnapot, egy a hétfőt és így tovább. Eredeti példánkhoz visszakanyarodva az ötórai gépleállítás crontabja valahogy így nézne ki:
`5 0 * * * /sbin/halt`.
Biztonsági okokból crontab-állományunkat nem szerkeszthetjük csak úgy kézzel. Erre a feladatra a `crontab` parancsot kell használnunk. Ha új bejegyzést szeretnénk létrehozni, hozzunk létre egy új állományt a saját könyvtárunkban, amelybe az új feladatot az előbb ismertetett formátumban írjuk be. Ezután adjuk ki a `crontab fÆjlnØv` utasítást, és máris bekerült a cron adatbázisába. Bejegyzést törölni a `-r` kapcsolóval tudunk, egy már meglévő átszerkesztéséhez pedig a `-e`-t kell használnunk.
A démonok tehát a rendszer indításától fogva betelepszenek a memóriába. De vajon a rendszer melyik része tölti be őket? Ezt a feladatot az úgynevezett indító parancsfájlok végzik el, amelyek minden rendszerinduláskor lefutnak (előző számunkban már említettük a héjakat, amelyek hasonlóak a DOS-os kötegelt (batch) állományokhoz, csak sokkal fejlettebbek). Ezeknek a héjaknak előre „megmondják”, hogy melyik démonokat kell betölteniük.
Az indító parancsfájlok valójában egyetlen démont sem tölt be, hanem csak a saját indítóhéjaikat hívja meg. Ezek az indítóhéjak általában a */etc/rc.d/init.d* könyvtár alatt találhatók.
Hogy a rendszer indulásakor melyik indítóhéjat használjuk, az úgynevezett futási szintek (runlevels) segítségével adhatjuk meg. A Linux-rendszerekben hét futási szint van megadva (0–6-ig). Minden egyes futási szintben leolvasható, hogy milyen démonokat indítsunk el, illetve állítsunk le.
A nullás és hatos a rendszerleállításra szolgál, tehát értelemszerűen az összes démon leállítását tartalmazza. Az 1-es általában csak a legszükségesebb összetevőket tartalmazza, a 2-es és a 3-as a standard beállítást. A 4-esnek és az 5-ösnek tulajdonképpen nincs meghatározott szerepe, esetükben általában a grafikus bejelentkezők is elindulnak.
Ezt a rendszert, ha gondoljuk, kedvünkre megváltoztathatjuk, de nincs sok értelme.
Hogy a rendszer melyik futási szinttel induljon, a */etc/inittab “id:* kezdetű sora írja le. Ha úgy látjuk jónak, változtassuk meg nyugodtan, csak arra vigyázunk, nehogy 0-ra vagy 6-ra állítsuk, mert kellemetlen meglepetésben lehet részünk (a rendszer elindulása után egyből a leállítási szintbe lép át).
Miután rendszerünk teljesen betöltődött, a különböző futási szintek között kedvünkre váltogathatunk, erre használható az `init <œj futÆsi szint>` utasítás. Ha az új futási szinthez 6-ot írunk, a kiadott parancs a `halt` utasítással lesz egyenértékű.
Végezetül nézzük meg, hogyan is szerkeszthetjük a futási szinteket! A démonok inicializációs scriptjeit a */etc/rc.d/init.d* könyvtár alatt lelhetjük meg. Minden futási szinthez külön könyvtár tartozik, amelynek elnevezése: */etc/rc.d/rc.x* (az x a futási szint számát jelöli). Kukkantsunk be az egyik ilyen könyvtárba! Mint láthatjuk, a démonok incializációs scriptjeire mutató közvetett hivatkozásokkal van tele. A hivatkozás nevének utolsó része megegyezik a démonindító parancsfájlok nevével (ez általában a démon nevével is azonos). Az első karakter csak `S` vagy `K` lehet. Az első esetben a héj a `start` értékkel hívható meg, tehát az adott démon elindul. Az utóbbi esetben pedig a `stop` érték lesz átadva, tehát a démon befejezi a futását.
Ha nem akarjuk, hogy egy démon például a 3-as futási szinten betöltődjön, egyszerűen távolítsuk el a közvetett hivatkozást. Az ***init.d*** könyvtár alatti héjakat kézzel is meghívhatjuk, így mi magunk a futási szintektől függetlenül is démonokat tölthetünk be, illetve távolíthatunk el a memóriából.
A következő részben már teljesen más vizekre evezünk: a Linux alatti fájlrendszerkezelést vesszük nagyító alá, tehát mindenféle lemezterületet és egyéb cserélhető lemezegységet fogunk fájlrendszerünkhöz be- és kifűzni.

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 (5. rész)

Ha valamit elrontottunk, azt vissza is állíthatjuk.

Amennyiben valamit eltévesztünk, a ***Menu > Undo more*** menüpont kiválasztásával vagy az ***Undo*** ikonnal visszaléphetünk (*1 f kép*). A ***C-x u*** vagy ***C-_*** ugyanezt teszi. Ha ezeket a parancsokat ismételten kiadjuk, egyre régebbi állapotokat állíthatunk vissza. Ha a visszalépések sorát egy akármilyen jelentéktelen parancs kiadásával (például a kurzormozgató ***C-f*** megnyomásával) bárhol megtörjük, az addigi visszaállítások is visszavonhatóvá válnak (redo). Az XEmacs a visszaállítás céljára alapértelmezetten általában 20 000 karaktert tud megjegyezni. Ez az érték az ***Options/Customize/Emacs/Editing/Undo/Threshold*** menüpontban állítható.

A regiszterek

Ha egy szövegrészt többször egymás után be akarunk illeszteni, célszerű a ***C-x r s r*** teljes billentyűvel egy regiszterbe átmásolni. A regiszterek olyan helyek, ahová szöveget, könyvjelzőket, számokat és állományneveket menthetünk későbbi használatra. A regiszterbe mentettek mindaddig ott maradnak, amíg újabb dolgot nem helyezünk el ugyanoda. A regiszterek neveiként egyetlen betűt használhatunk, azaz a fenti parancsban az utolsó r betű a regiszter neve, amit más betűvel is helyettesíthetünk. Később a ***C-x r i r*** paranccsal a szöveget máshová is beilleszthetjük. Ha elfelejtjük, hogy mi található egy adott nevű regiszterben, tartalmát az ***M-x view-register RET r*** paranccsal nézhetjük meg. Könyvjelzőket, azaz olyan szöveghelyeket, ahová később vissza akarunk ugrani, a ***C-x r Szóköz d*** paranccsal menthetünk regiszterbe, aminek a mostani példában ***d*** a neve. Az ugrást a ***C-x r j d*** parancs végzi. Lehetőség van még az ablakok vagy keretek helyzetének regiszterekben való mentésére is a ***C-x r w v*** és az ***M-x frame-configuration-to-register RET v*** parancsokkal. A visszaállításhoz ismét a ***C-x r j v*** teljes billentyűt kell használnunk. Most *v* a regiszter neve. Az ablakok és keretek visszaállításakor a beállításban nem szereplők láthatatlanná válnak, de nem törlődnek. Számokat is tehetünk a regiszterekbe a ***C-u szám C-x r n n*** paranccsal, és az értéküket a ***C-u szám C-x r + n*** beírásával meg is növelhetjük. Az első példában ***szám*** számot teszünk az ***n*** regiszterbe, a másodikban az ***n*** regiszterben lévő számot növeljük meg ***szám*** értékkel. Az így mentett számot a ***C-x r g n*** teljes billentyűvel szúrhatjuk be az ***n*** nevű regiszterből. Fájlokat is tehetünk regiszterekbe. Ha például a *~/.emacs* beállítási fájlba beírjuk a

(set-register ?f '(file . "~/.emacs"))

sort, később a ***C-x r j f*** paranccsal egy lépésben meglátogathatjuk a gyakran használt állományt. Az idézőjelek közé a választott fájl nevét kell írni. A ***?f*** a regiszter neve.

Átfordító utasítások

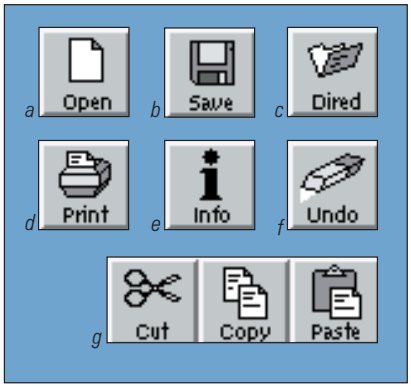
A ***C-t*** átfordító utasítással karaktereket cserélhetünk meg. Ha a ***Linuxvilág*** szóban a kurzorral ráállunk az ***i*** betűre, és megnyomjuk a ***C-t*** billentyűket, akkor a ***Linuxivlág*** szót kapjuk, azaz a *v* betű helyébe az ***i*** került, a ***i*** helyébe pedig a *v*. Az ***M-t*** szavakat cserél fel. Ha a ***transpose Linuxvilág*** szavaknál az első szóra állunk a kurzorral, akkor az ***M-t*** használata a ***Linuxvilág transpose*** eredményt adja. A ***C-x C-t*** sorokat, a ***C-M-t*** kifejezéseket fordít át.

Betűátalakítások

A következő teljes egy vagy több szót billentyűkkel kisbetűssé vagy nagybetűssé alakíthatunk:

M-l – a pont utáni betűket a szó végéig kisbetűssé alakítja
M-u – a pont utáni betűket a szó végéig nagybetűssé alakítja
M-c – (1.) a következő szó első betűjét nagybetűssé és a többit kisbetűssé alakítja, (2.) ha a szó közepén vagyunk, akkor a kurzor alatti betű lesz nagybetűs, a kurzor pedig a szó végére ugrik

A fenti parancsoknak pozitív- vagy negatívszám-kapcsolót is átadhatunk. A ***C-u 20 M-u*** parancs például 20 egymás után következő szót alakít át nagybetűssé. Mivel a kurzor mindig a szó végére ugrik, a teljes billentyűk nyomogatásával egymás után több szót is átalakíthatunk. Ha eközben ki akarunk hagyni egy-egy szót, nyomjuk meg az ***M-f*** billentyűt.



1. kép

A következő parancsok nem egy szóra, hanem egy kijelölt területre hatnak:

C-x C-l – a kijelölt területet kisbetűssé alakítja,
C-x C-u – a kijelölt területet nagybetűssé alakítja.

Az ***M-c*** billentyűnek megfelelő területet átalakító parancs használata ritkább, ezért egyetlen billentyűhöz sem kötötték:

M-x capitalize-region

Ez a parancs a kijelölt terület minden szavának első betűjét nagybetűssé, a többit kisbetűssé alakítja.

Könyvjelzők

A kurzor pillanatnyi helyzetébe a ***C-x r m könyvjelzőnév RET*** paranccsal szúrhatunk be könyvjelzőket, amik a regiszterektől annyiban különböznek, hogy hosszú neveik lehetnek, és rejtve a szövegben az Emacs bezárása után is megmaradnak, mivel a program kimenti őket a *~/emacs.bmk* nevű fájlba:

```
(("emacs-mark"
  ((filename .
➥"/home/ratio/.emacs")
```

```
    (front-context-string .
➥"XEmacs to avoid ")
    (rear-context-string .
➥"macs-custom for ")
    (position . 3371))))
```

A fenti példa az emacs-mark nevű könyvjelzőt elhelyezi az *.emacs* beállítófájlba a 3371. pont helyzetébe. A már meglévő könyvjelzőket a ***C-x r l*** teljes billentyűvel listázhatjuk ki, ahol egyúttal szerkeszthetjük is őket. A könyvjelző mód szerkesztési parancsait ilyenkor a ***C-h m*** súgó paranccsal tekinthetjük meg. A ***C-x r b könyvjelzőnév RET*** paranccsal ugorhatunk a kívánt helyre, és beíráskor a ***NYÍL*** billentyűket a mini átmeneti tárban használhatjuk a már beírt könyvjelzők előhívására. Egy már meglévő könyvjelzőt az ***M-x bookmark-delete RET könyvjelzőnév RET*** paranccsal törölhetünk.

Címkék

Programozás közben gyakorta előfordulhat, hogy nem emlékszünk rá pontosan, hogy az adott függvény mit csinál. Ilyenkor célszerű megkeresni és megnézni, hogy mi van benne. Ehhez nyújtanak segítséget a címkék (tags). Az Emacs képes önműködően címkéket rendelni többek közt a C-, C++, Java-, Lisp-, Pascal- és Perl-kódhoz. Ezt a munkát most az ***etags*** program végzi el helyettünk, amely a fájl végződéséből és tartalmából felismeri, hogy melyik programozási nyelvről van szó:

etags file1 file2 ...

A címkéket tartalmazó állomány alapértelmezetten a TAGS címketáblázat fájl lesz, amit a program a munkakönyvtárba tesz. Ha programunkhoz új függvényeket adunk, akkor ezt a fájlt időről időre célszerű frissíteni. Egyszerre több címketáblázat fájlunk lehet, de közülük egyszerre csak egy lehet aktív. Az alapértelmezettől eltérőt az ***M-x visit-tags-table*** paranccsal választhatunk. Miután címketáblázatunk elkészült, az ***M-. címke RET*** paranccsal kereshetünk benne. Ha nem tudjuk, milyen címkék közül választhatunk, a ***Tab*** megnyomásával listát kapunk. A listából a középső egérrel a megszokott módon választhatunk. A ***C-u M-.*** teljes billentyű megkeresi a legutóbb megtalált címke következő előfordulását, ha van ilyen. Negatív kapcsolóval visszaugrik az előzőleg megtalált címkéhez, tehát ilyenkor a ***C-u - M-.*** teljes billentyűt kell gépelnünk. Az ***M-***, billentyű a pillanatnyi tárolóban nézi meg, hogy nincsen-e még egy találat. Az ***M-x list-tags RET fájlnév RET*** az állományhoz tartozó címkéket listázza ki.

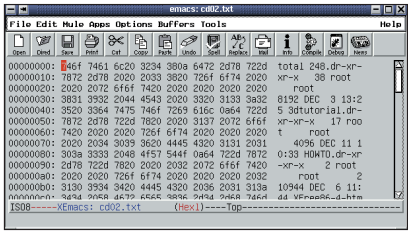
Héjparancsok

Ha héjparancsokat akarunk végrehajtatni, nem kell kilépnünk az Emacsból, elegendő, ha a mini átmeneti tárolóba beírjuk az ***M-! parancs RET*** parancsot. Például kiírattathatjuk az aktuális munkakönyvtárat:

M-! pwd RET

Ki is listázhatjuk a tartalmát:

M-! ls -l RET



A listázás eredménye egy új Emacs-ablakban jelenik meg. Ha akarunk, az ***M-x shell*** paranccsal egy ablakban új héjat nyithatunk.

Bináris fájlok szerkesztése

Bináris állományok hexadecimális szerkesztésére a fájlt az ***M-x hexl-find-file*** paranccsal kell behívni, ami működésbe hozza a ***Hexl*** főmódot, amelyben a következő parancsok használhatók:

C-M-d szám RET – egy tízes számrendszerben megadott bájtot szúr be,
C-M-o szám RET – egy nyolcas számrendszerben megadott bájtot szúr be,
C-M-X szám RET – egy tizenhatos számrendszerben megadott bájtot szúr be,
M-g szám RET – a tizenhatos számrendszerben megadott címre ugrik,
M-j szám RET – a tízes számrendszerben megadott címre ugrik,
C-x [– egy 1 kilobájt nagyságú szövegrész elejére ugrik,
C-x] – egy 1 kilobájt nagyságú szövegrész végére ugrik,
C-c C-c –kiléphetünk a ***Hexl*** módból.

Súgó az Emacsban

Az XEmacs rögtön induláskor eligazítást ad arról, miképpen juthatunk segítséghez. Mihelyt megnyomjuk az F1 gombot, észre fogjuk venni, hogy nem a Windowsban megszokott Súgó rendszerhez jutunk. A kezdő felhasználó nem minden

alap nélkül úgy érezheti magát, hogy ez a Súgó olyanok számára készült, akik már mindent tudnak az Emacsról, és csak az emlékezetüket akarják felfrissíteni. A legalul lévő visszhangterületen üzenetet kapunk, hogy gépeljünk be egy *?* kérdőjelet. Miután megtesszük, listát kapunk a beüthető billentyűkről. Elsőként nyomjuk meg az '***a***' betűt, majd a mini átmeneti tárolóba írjunk be egy olyan karakterláncot, ami sejtésünk szerint része a keresett parancsnak. Tegyük fel, olyan parancsokat keresünk, amik tartalmazzák a ***language*** szót. Mivel programozók vagyunk, nem szeretünk többet gépelni, mint amennyi feltétlenül szükséges, ezért csak annyit írunk be, hogy ***lan***. Miután megkaptuk a keresés eredményét, láthatjuk, hogy az Emacs minden olyan parancsot kilistáz egy ablakban, ami tartalmazza a fenti betűket, mint például a ***describe-lang uage-environment*** vagy a ***balan ce-windows*** ***describe-lang uage-environment*** vagy a ***balan ce-windows*** parancsokat. Gyakorlásképpen nézzük meg, hogy milyen eredményt kapunk a ***fon*** vagy a ***des*** szótöredékek begépelésekor! A ***q*** megnyomásával elhagyhatjuk a ***Súgó*** ablakot. Most gépeljük be a hasonlóan működő és hasonló feladatot ellátó ***M-x apropos*** parancsot. Nyomjuk meg ismét az ***F1*** billentyűt, majd üssük le a ***b*** betűt! Most az Emacsban használatos billentyűkombinációkról kapunk listát. A ***c*** betű megnyomása után azt a billentyűkombinációt kell a mini átmeneti tárba begépelnünk, amiről súgást akarunk kapni. Például a ***C-x 5 2*** beírása után a következő válasz jelenik meg: ***C-x 5 2 runs the command make frame***. Ha ezt a parancsot az ***F1 c*** előtag nélkül gépeljük be, az Emacs egy új keretet fog létrehozni. Próbáljuk ki ezt is! Az ***F1 f Tab*** listát készít az Emacs Lisp-függvényekről, az ***F1 k*** rövid leírást ad a megnyomott billentyű feladatáról, az ***F1 w*** pedig egy parancs begépelésekor

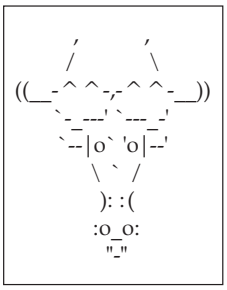
kinyomtatja, hogy milyen billentyűkkel hívhatjuk elő. Az ***F1 l*** az utolsó száz begépelt karaktert mutatja meg. Ez jól jön adatvesztéskor vagy hibakereséskor, mivel megnézhetjük, hogy miket is csináltunk. Ha egy menüpont leírását szeretnénk olvasni, írjuk be a ***C-h k*** billentyűkombinációt, és az egérrel válasszuk ki a kívánt menüt! Próbáljuk ki az ***F1*** megnyomása után a visszhangterületen látható többi billentyűt is!

A kezdőknek elsőként az oktatófájlon célszerűbb átrágniuk magukat, amit a ***C-h t*** vagy ***F1 t*** kombinációkkal indíthat el. Megnézhetjük még a man- és infooldalakat a ***man (x)emacs*** vagy ***info (x)emacs*** parancsokkal a héjban, vagy elolvashatjuk azokat magában az Emacsban a ***Help*** menüpont alatt és az XEmacsban az ***Info*** ikont megnyomva (*1 e kép*).

Az ***F1 i*** szintén az infooldalakat hozza elő, az ***F1 F*** pedig a ***Gyakori Kérdések***-et. A ***Súgó***-ban szóközzel lapozhatunk, és a ***q*** megnyomásával léphetünk ki belőle.

Utószó

Mostani rövid sorozatomhoz a SuSE Linux 7.2 profi kiadását használtam. Feltehető, hogy más terjesztések alapértelmezetten nem tartalmazzák a legelső részben ismertetett szerkesztők mindegyikét, ezért pillanatképeket készítettem róluk, amelyeket a CD-melléklet Magazin/Emacs könyvtárába helyeztem.



Néhány más, a témához kapcsolódó anyag is található a teljesség igénye nélkül. Nem hiszem, hogy az olvasók közül bárki egy ültő helyében elejétől a végéig el fogja olvasni ezt az írást, de nem is ez volt a célom. Az Emacs iránt érdeklődők üljenek le a gép elé, és kezdjék el használni a fent leírtak alapján. Nem garantálhatom, hogy az Emacs mindenkinél ugyanígy fog működni, de nagy eltérések nem lehetnek. A billentyűkombinációkat nem bemagolni kell, hanem készségszinten az ujjainkban rögzíteni. Vannak olyanok, akik már évek óta használják az Emacsot, és nem tudják megmondani, hogy ezt vagy azt a parancsot milyen teljes billentyűvel lehet előhívni, de amikor leülnek a számítógép elé, gondolkodás nélkül beütik azt. Lehet, hogy őrültségnek tűnik a sok felsorolt billentyű és parancs megtanulása, és még nagyobb őrültségnek az, hogy még azokat is megtanuljuk, amelyeket most nem említettem meg, de számos gyakorlott programozó egybehangzó véleménye szerint az Emacs megismerése hosszú távon kifizetődő. Nyilvánvaló, hogy a ***vi*** vagy az Emacs más, mint amit más operációs rendszerekben eddig megszoktunk, de az a tény, hogy furcsa és szokatlan, még nem lehet a minősítés alapja. Ha valakinek van olyan angol, német vagy magyar nyelvű leírása az Emacsról, ami nem szerepel a CD-mellékleten, szívesen fogadnám. Érdekes lenne arról is hallani, hogy ki és miképpen alakította az Emacsot a saját lelkületének megfelelőre.

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 fájlrendszer sértetlenségének ellenőrzése CVS-sel

A számítógépre történő behatolás felfedezésének legegyszerűbb módja, ha a fájlrendszer változásait figyeljük.

Erre a feladatra a Tripwire a legjobb ismert megoldás, amely időről időre önműködően átvizsgálja a fájlrendszert, és minden engedély nélküli változást jelent a rendszergazdának. Sokféle változást tud észlelni a Tripwire, többek közt a jogosultságokét, a fájltípusét, a fájlleíró számét (inode), a hivatkozások számáét, a felhasználóazonosítóét (UID) és a csoportazonosítóét (GID), a méretét, a hozzáférés idejéét, a módosítás idejéét, a fájlleíró változásának idejéét és a hamisíthatatlan ujjlenyomat változásáét (amelyet a tíz ujjlenyomat-készítő függvény bármelyikének alkalmazásával létrehozhatunk, például MD5 és Snefru). Ebben a cikkben bemutatjuk, hogy a Concurrent Versions System (CVS – párhuzamos változatkezelő rendszer) Perllel összegyúrva alkalmassá tehető hasonló fájlrendszer-ellenőrző feladatok ellátására, mint amire a Tripwire képes.

Tripwire

A Tripwire telepítésének leegyszerűsített folyamata az alábbi néhány lépésben foglalható össze:

1. telepítsd az operációs rendszert, és azonnal futtasd a Tripwire-t a konzolról, mielőtt még a gépet a hálózatra kötnéd;
2. a fájlrendszer adatait tartalmazó Tripwire adatbázist tartsd írásvédett adathordozón, a rendszertől elkülönítve;
3. rendszeresen futtasd a Tripwire-t, hogy a jelenlegi fájlrendszer adatai és az eredeti adatbázis közötti különbségek kiderüljenek.

CVS

A CVS-t elsősorban programfejlesztők használják, segítségével a forráskód különböző változatai rendszerbe szervezhetők. Az egyik leghasznosabb tulajdonsága, hogy képes nyomon követni az adott forráskóddarab (vagy közönséges szöveg) időbeli változásait attól az eredeti változattól kezdve, amelyet a kezdetekkor raktak a CVS-raktárba. Ha a fejlesztő változtatni akar a forráskódon, kikéri a raktárból, megváltoztatja, és az eredményként előálló kódot visszateszi a raktárba. A CVS önműködően megnöveli a változatszámot, és nyomon követi a változásokat. A módosítások raktárba tétele után a `cvs diff` parancs segítségével pontosan látható, hogy mi változott az új változatban az előző (vagy bármelyik másik) változathoz képest.
Mostanra nagyjából értjük, hogy milyen lépésekből áll a Tripwire alkalmazása. Megmutatjuk, hogy a CVS hasonlóképpen használható, de egy nagy előnye van a Tripwire-rel szemben: a szöveges beállítófájlok különböző változatainak kezelése. Ha gépedet feltörték, és hozzáadtak egy felhasználót a */etc/passwd* fájlhoz, a Tripwire jelenti a tényt, hogy bizonyos számú fájltulajdonság megváltozott, például a módosítási idő vagy az ujjlenyomat, de nem tudja megmondani, hogy melyik felhasználót adták hozzá.
A behatolásérzékelés fő gondja szokott lenni, hogy a vakriasztások száma gyakran nagyon magas, és ez alól az általunk vizsgált megoldások sem kivételek. A rendszerek számától, és így a bonyolultságtól függően a vakriasztások olyan nagy mértékben is jelentkezhetnek, hogy a rendszergazda nem tudja átlátni és kezelni őket. Ezért ha a felhasználó hozzáadásának példáját vesszük, jó lenne, ha a támadásérzékelő program jelentené, hogy kit adtak hozzá a */etc/passwd* fájlhoz, mert ebből könnyebben eldönthetjük, hogy a hozzáadás jogosan történt-e, vagy valami másról van szó. Ez sok óra megtakarított időt jelenthet, mert ha biztossá válik, hogy a rendszert feltörték, az eredeti állapotok visszaállítására az egyetlen biztos megoldás csak az egész operációs rendszer újratelepítése.

Adatgyűjtés

Megtartjuk a Tripwire módszerét, azaz pillanatfelvételt készítünk a kiindulási fájlrendszerről, és a megfigyelendő rendszertől független adathordozón tároljuk. Ehhez valamilyen módon össze kell gyűjtenünk a távoli rendszerek adatait, és egy helyi CVS-raktárban kell tárolnunk. Hozzunk létre egy CVS-nek szentelt gyűjtőgépet a hálózaton! Minden fájlrendszerfigyelő szolgáltatást ezen a gépen egyetlen felhasználó futtat. A szóba jöhető figyelőfeladatok a következők lehetnek: a nyers adatok begyűjtése, az adatok összehangolása a CVS-raktárral és riasztás küldése jogosulatlan adatváltozás esetén. Az SSH-protokollt használjuk a hálózaton keresztül a távoli gépek adatainak átvitelére. Az egyszerűség kedvéért a támadásérzékelő rendszer felhasználójának RSA-kulcsát minden célrendszer rendszergazdájának ***authorized_keys*** fájljába elhelyezzük. Ez lehetővé teszi, hogy ez a felhasználó rendszergazdaként futtathasson parancsokat a célrendszeren, anélkül, hogy az SSH-n keresztül jelszót kellene megadnia. Most, hogy összeállt az általános kép a gyűjtőgépről, megkezdhetjük az adatok gyűjtését.
Kétféle típusú adatot kell a távoli rendszerekről begyűjtenünk: ASCII beállítófájlokat és a parancsok kimeneteit. A parancsok kimeneteinek begyűjtése általában szélesebb kategória a fájlok begyűjtésénél, hiszen a távoli fájlrendszereket valószínűleg nem akarjuk teljes egészében lemásolni. Többek között a következő két parancs szolgáltat fontos adatokat a megfigyelés szempontjából: `md5sum` (MD5-ujjlenyomatot készít a fájlokhoz) és `find / -type f -perm +6000` (azokat a fájlokat keresi meg, amelyeknek az felhasználóazonosító és csoportazonosító jogosultságbitjei be vannak állítva). Az 1. (33. CD Mazazin/CVS könyvtár) és 2. *listán* olyan Perl-kódot mutatunk be, amely adatokat gyűjt a távoli rendszerekről és figyeli az adatok időbeli változásait.
Az 1. lista Perl-kódjában a `Net::SSH::Perl` modult használjuk három adathalmaz begyűjtésére a `192.168.10.5` című gépről, lehívjuk néhány fontos rendszer bináris MD5-ujjlenyomatát, néhány beállítófájl tar-archívumát és az felhasználóazonosító (UID), illetve csoportazonosító (GID) jogosultságbitekkel rendelkező fájlok listáját. A 7. és 8. sorban adjuk meg a célszámítógép IP-címét és a távoli felhasználó nevét, amelyet a `collector.pl` használ majd a bejelentkezésre. Emlékeztetünk, hogy a helyi felhasználó kulcsa már ott van a távoli rendszeren, ezért a bejelentkezéskor nem szükséges a jelszó

2. lista A cvschecker.pl

```
#!/usr/bin/perl

use File::Find;
use strict;
use vars qw(@Files);

my $email = ·root@localhost·;
my $host = "192.168.10.5";
my $cvsroot = "/usr/local/hbids_cvs";
my $collectorCmd =
➥"/usr/local/bin/collector.pl";

system "cvs -d $cvsroot checkout -ko $host";
chdir $host;
system "$collectorCmd"; ### fÆjlok
➥megszerzØse a $host-r l

find(\&findfiles, "/home/mbr/${host}");

### ellenırizz k, hogy megvÆltoztak-e a
➥fÆjlok
for my $file (@Files) {
    if (`cvs -d $cvsroot commit -m . $file`)
{
        my $cvsfile = $file;
        $cvsfile =~ s|^\S+?$host|$host|;
        $cvsfile = $cvsroot ."/" . $cvsfile
        ➥.",v";
        my $prerev = "";

        ### az rlog hasznÆlatÆval kapjuk meg
        ➥az elızı vÆltozatot
        open RLOG, "rlog $cvsfile|";
        my $found = 0;
        while (<RLOG>) {
            if (/^revision\s(\S+)/) {
                $prerev = $1;
                $found++;
            }
            last if ($found == 2);
        }
        close RLOG;

        ### a k l nbsØgek felder tØse a ·cvs
        ➥diff· hasznÆlatÆval
        my $diff = `cvs -d $cvsroot
        ➥diff -r        \

$prerev $file`;
        open TMP, "> /tmp/change";
        print TMP $diff;
        close TMP;
        $file =~ s|^\S+?$host||g;

        ### a vÆltozÆsok elk ldØse levØlben
        `mail -s "MegvÆltozott fÆjl: $host:
        ➥$file"    \
                                $email <
        ➥/tmp/change`;
    }
}
exit 0;

### az  sszes fÆjl feldolgozÆsa a helyi
➥k nyvtÆrszerkezetben,
### kivØve a k nyvtÆrakat Øs a CVS-fÆjlokat
sub findfiles() {
    my $file = $File::Find::name;
    unless (-d $file || $file =~ /CVS/) {
        push @Files, $file;
    }
    return;
}
```

megadása. A 10–13. sorokban a rendszer szempontjából fontos binárisok kis példalistája szerepel, ezekre kell majd az MD5-ujjlenyomatot elkészíteni. Hasonlóképpen a 15–18. sorokban adjuk meg, hogy mely fájlokról készítünk a helyi gépen biztonsági mentést. A 20–24. sorok egy helyi fájlt rendelnek a távoli gépen lefuttatandó parancshoz, minden parancs kimenete ebben a fájlban fog a helyi gépen tárolódni. A 27–33. sorokban rejlik a gyűjtőalkalmazás lényege, a `n_command()` alprogram hívása (36–49. sorok) a `%Cmds` tömbben tárolt minden parancsra. A `run_command()` minden egyes végrehajtásakor új `Net::SSH:Perl` objektumot hoz létre, amelyen keresztül SSH-kapcsolatot nyit a távoli gép felé, bejelentkezik és végrehajtja az alprogramnak átadott parancsot.

A 2. lista Perl-kódja az elektronikus levelek létrehozásáért felelős, amelyek riasztanak, ha a `collector.pl` által begyűjtött fájlok bármelyike megváltozott. Ez úgy történik, hogy kikérjük a pillanatnyi `192.168.10.5` modult a CVS-raktárból (12. sor), lefuttatjuk a `collector-pl`-t (14. sor), hogy friss változataink legyenek a távoli parancsok kimeneteiből és a fájlokból a helyi könyvtárszerkezetben, majd minden egyes (esetlegesen módosult) fájlt visszahelyezünk a raktárba (20. sor). Ellenőrizzük a `cvs commit` parancsának visszatérési értékét (20. sor). Ebből eldönthetjük, hogy a fájl megváltozott-e, mert a CVS önműködően növeli a változatszámot és követi a változásokat. Ha egy bizonyos fájl megváltozott, a `cvschecker.pl` kiszámítja az előző változatszámot (27–36. sorok), végrehajtja a `cvs diff` parancsot a mostani és az ezt megelőző változat között, hogy megkapjuk a különbségeket (39–40. sorok), és a változásokat (47–48. sorok) elküldi arra a levélcímre, amit a 7. sorban adtunk meg.

Támadások érzékelése

Lássuk a *collector.pl*-t és a *cvschecker.pl*-t munka közben pár támadáson keresztül. Tegyük fel, hogy a célrendszer egy Red Hat 6.2-es gép, a fájlrendszer adatait a gép hálózatba kötése előtt elraktároztuk, és a gép IP-címe `192.168.10.5`.

1. példa

Tegyük fel, hogy a `192.168.10.5` gépet feltörték, és rendszergazdaként a következő parancsot adták ki:

```
# cp /bin/sh /dev/... && chmod 4755 /dev/...
```

Ez a */bin/sh* héjat a */dev/* könyvtárba másolja "..." fájlnévvel, és

beállítja az UID bitet. Mivel a fájl tulajdonosa a rendszergazda, és a rendszer bármelyik felhasználója végrehajthatja, a támadónak csak a */dev/...* elérési utat kell ismernie, és bármilyen parancsot futtathat rendszergazdaként. Magától értetődően értesülni szeretnénk róla, hogy ez történt a `192.168.10.5` rendszerrel. A gyűjtőgépen elindítjuk a `cvschecker.pl`-t, ekkor a root@localhost egy olyan levelet kap, amelyből egyértelműen kiderül, hogy a */dev/...* egy új SUID-fájl:

```
From: hbids@localhost
Subject: MegvÆltozott fÆjl: 192.168.10.5:
suidfiles
To: root@localhost
Date: Sat, 10 Nov 2001 17:35:13 -0500 (EST)

Index: /home/mbr/192.168.10.5/suidfiles
===================================================
RCS file:
/usr/local/hbids_cvs/192.168.10.5/suidfiles,v
retrieving revision 1.3
retrieving revision 1.4
diff -r1.3 -r1.4
4a5
> -rwsr-xr-x 1 root root 512668 Nov 10
➥18:40/dev/...
```

2. példa

Tegyük fel, hogy a támadó rendszergazdaként a következő két parancsot adja ki:

```
# echo "eviluser:x:0:0::/:/bin/bash" >>
➥/etc/passwd
# echo "eviluser::11636:0:99999:7:::" >>
➥/etc/shadow
```

Figyeljük meg, hogy az eviluser UID-je és GID-je a */etc/passwd* fájlban 0, és a */etc/shadow*-ban nincs titkosított jelszóbejegyzés. Emiatt a rendszer bármelyik felhasználója az `su - eviluser` begépelésével jelszó nélkül is rendszergazdai jogosultságokat szerezhet. Ahogy az előbb a `cvschecker.pl` futtatása után, most a következő levelet találjuk a rendszergazda postafiókjában:

```
From: hbids@localhost
Subject: MegvÆltozott fÆjl: 192.168.10.5:
➥/etc/passwd
Delivered-To: root@localhost
Date: Sat, 10 Nov 2001 17:43:17 -0500 (EST)
Index: /home/mbr/192.168.10.5/etc/passwd
===================================================
RCS file:
/usr/local/hbids_cvs/192.168.10.5/etc/passwd,v
retrieving revision 1.2
retrieving revision 1.3
diff -r1.2 -r1.3
26a27
> eviluser:x:0:0::/:/bin/bash
```

és

```
From: hbids@localhost
Subject: MegvÆltozott fÆjl: 192.168.10.5:
➥/etc/shadow
Delivered-To: root@localhost
Date: Sat, 10 Nov 2001 17:43:18 -0500 (EST)

Index: /home/mbr/192.168.10.5/etc/shadow
===================================================
RCS file:
/usr/local/hbids_cvs/192.168.10.5/etc/shadow,v
retrieving revision 1.2
retrieving revision 1.3
diff -r1.2 -r1.3
26a27
> eviluser::11636:0:99999:7:::
```

Összegzés

A fájlrendszer változásainak észlelése hatékony módszer a behatolás felfedezésére. Ebben a cikkben bemutattuk, hogy egyszerű Perl-programok segítségével a CVS házi készítésű támadásérzékelő rendszerré fejleszthető. Mostani munkahelyemen, az USinternetworkingnél, amely egy nagy ASP a marylandi Annapolisban, hasonló (bár sokkal kiterjedtebb) egyedi rendszert használunk. Az USiOasis nevű rendszer több száz hálózatba kötött gép fájlrendszerének sértetlenségét ellenőrzi. A gépeken sokféle operációs rendszer, többek közt Linux, HPUX, Solaris és Windows, és sok különböző kiszolgálóoldali alkalmazás fut. A rendszer háttere egy MySQL-adatbázis, egy elég nagy CVS-raktár és egy egyedi web-, illetve CGI-felület, amelyet nagyrészt Perlben írtak. A változások követése a CVS-raktárral még egy fontos előnnyel jár: létezik egy remek megjelenítő eszköz, a Pythonban írott ViewCVS. Az operációs rendszer és az alkalmazások beállítófájljainak a CVS-ben tartása nemcsak a támadásérzékelés szempontjából előnyös, hanem többek között a hálózati és alkalmazáshibák elhárítására, katasztrófa utáni helyreállításra és a rendszer beállításainak követésére is alkalmas.

Linux Journal 2002. február, 94. szám

Michael Rash
(mbr@cipherdyne.com) vezető biztonsági mérnök az ASP-nél a marylandi Annapolisban. Az University of Marylandon szerezte diplomáját alkalmazott matematikus szakon. 1998 óta foglalkozik a Linuxszal. Szabadidejében a Prince George filharmonikus zenekarban hegedül.

Kapcsolódó címek

AIDE ➲ http://www.cs.tut.fi/~rammer/aide.html
FCheck ➲ http://www.geocities.com/fcheck2000
➲ http://portal.acm.org.
Ha egy gépet feltörtek, nagyon is elképzelhető, hogy egy betölthető rendszermagmodullal magát a rendszermagot is megváltoztatták. Ez a parancsvégrehajtásba történő beavatkozást a rendszermag szintjén teszi lehetővé, ezért rendszer szempontjából lényeges binárisokba nem kell a hátsó kapukat építeni. Lásd például:
➲ http://www.uberhaxr.net/kis
A */etc/passwd* és */etc/shadow* fájlok felépítéséről bővebben a megfelelő man(5) súgóoldalak szólnak.
ViewCVS ➲ http://viewcvs.sourceforge.net

A Subversion vállalkozás

Ismerkedjünk meg a hatékonyabb változatkezelő rendszer elképzelésével és megvalósításával.

E vállalkozás a CVS-rendszert (Concurrent Versions System) kiváltó programcsomag megalkotását tűzte ki célul. Aki valamilyen nyílt forráskódú program fejlesztésén munkálkodik, minden bizonnyal dolgozott már a CVS-sel, és talán még azt is fel tudja idézni, hogyan tanulta meg egy forráskódfa névtelen ellenőrzését a Világhálón keresztül, az első programfordítást, vagy éppen az első CVS-változások figyelemmel kísérését. Aztán egyszer csak elérkezett a végzetes nap, amikor megkérdezte az egyik barátját, hogyan is kell átnevezni egy állományt:
– Ó, azt nem lehet! – hangzott a válasz.
– Micsoda? Hogy érted azt, hogy nem lehet? – kérdezett vissza.
– Szóval, az állományt csak törölni tudod, majd új néven felvenni a nyilvántartásba.
– Jó, akkor kérdezzük meg a CVS-gazdát! Vajon ő képes-e kézzel módosítani a CVS-rendszer RCS-állományait és képes-e újra működőképessé tenni a rendszert?
– Jaj, csak most jut eszembe, hogy könyvtárt se próbálj meg törölni!
Az ember a szemét forgatta és egyre csak sóhajtozott. Hogyan lehet az ilyen egyszerű dolgok végrehajtása ennyire bonyolult?

A CVS öröksége

Kétség sem férhet hozzá, hogy a CVS a Nyílt Forráskód Közösségének programbeállító kezelőrendszerévé (SCM) fejlődött, és tegyük hozzá, joggal vált azzá. A CVS olyan nyílt forráskódú program, amelynek gördülékeny fejlesztési modellje az együttműködést egymástól nagy földrajzi távolságokra levő programozók százai számára teszi lehetővé. Sőt van, aki még azt is megkérdőjelezi, hogy az olyan webhelyek, mint a Freshmeat vagy a Sourceforge a CVS nélkül képesek lettek volna-e úgy felvirágozni, mint amilyen sikerrel a mai napig működnek. A CVS és a hozzá tartozó káoszszerű fejlesztési modell mostanra a Nyílt Forráskód kultúrájának elemi részévé vált. Akkor mi a baj a CVS-sel?
Mivel a CVS a háttérben az RCS tárolási rendszert használja, csak az állományok tartalmi változásainak követésére képes, a faszerkezetbeli változásokéra nem. Ennek következtében a felhasználó számára nincs lehetőség az előzmények elvesztése nélkül az állományok másolására, áthelyezésére vagy átnevezésére. A faszerkezetbeli átrendezések minden esetben csúnya kiszolgálóoldali csűrés-csavarásokat jelentenek.
Az RCS a kiszolgálóoldalon képtelen a bináris állományok hatékony tárolására, az elágazási és jelölési műveletek igen lelassulnak. De a CVS a hálózatot is gyenge hatékonysággal használja, mint ahogy számos felhasználó bosszankodik a hosszú várakozási idők miatt: ezek abból adódnak, hogy az állománybeli módosulások küldése egyirányú – a kiszolgálótól az ügyfél felé és sohasem fordítva –, és a bináris állományok mindig teljes terjedelmükben kerülnek átvitelre.
A fejlesztő szempontjából a CVS-alapkód tulajdonképpen a történelmi javítások egymásra helyezett rétegei. Gondoljunk csak arra, hogy a CVS életének kezdeti szakaszában az RCS-t használó héjprogramok tömkelege létezett. Ez az örökség a programkódok értelmezését, karbantartását és kiterjesztését nehézkessé teszi. Például a CVS hálózati képessége csak „oda lett ragasztva”, de valójában sohasem volt igazi kiszolgáló-ügyfél rendszer. A CVS rendszer gondjainak megoldása óriási feladat, és ekkor még csak néhány gyakori panaszt említettünk.

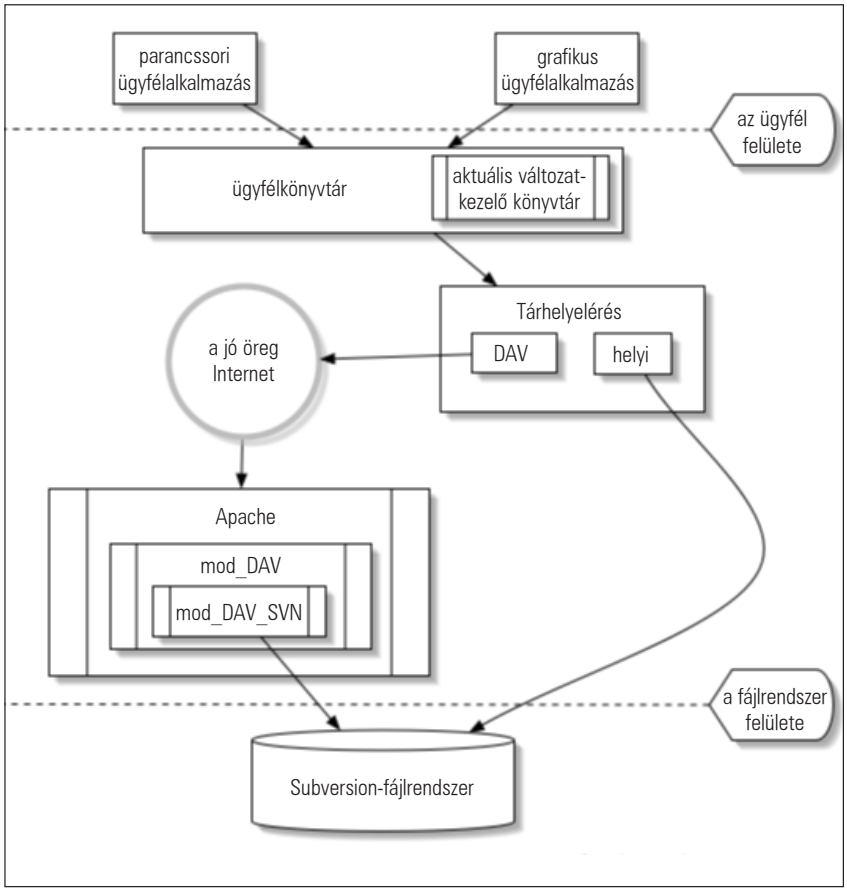
Lépjünk be a Subversion rendszerbe!

1995-ben ***Karl Fogel*** és ***Jim Blandy*** megalapította a Cyclic Software nevű céget, amely a CVS-rendszer kereskedelmi alapú fejlesztésére és a felhasználók számára nyújtott terméktámogatásra vállalkozott. A Cyclic megjelentette a CVS első hálózatképes, nyilvános változatát. 1999-ben Karl Fogel könyvet adott ki a CVS-ről és az általa lehetővé tett nyílt forráskódú programfejlesztésről (➲ http://cvsbook.red-bean.com). Karl és Jim már régen tervbe vették a CVS kiváltását; Jim még egy új raktárrendszer (repository) tervét is felvázolta. Végül 2000 februárjában a Collabnetnél (➲ http://www.collab.net) dolgozó ***Brian Behlendorf*** Karlnak teljes munkaidős állást ajánlott, és a CVS-rendszert kiváltó program elkészítésével bízta meg. Karl kiválasztotta a munkatársait, a csapat összeállt, és májusban megkezdődött a munka.
A fejlesztőcsapat néhány egyszerű célt tűzött ki maga elé: eldöntötték, hogy a Subversionnak működés szempontjából helyettesítenie kell a CVS-t. Mindenre képes lesz, amit a CVS el tudott végezni – megőrizve ugyanazt a fejlesztési modellt, ugyanakkor megoldja a CVS tervezési (hiányosságokból fakadó) hibáit. A jelenlegi CVS-felhasználók a programfejlesztés célközönsége; bármelyik CVS-felhasználónak csekély erőfeszítés révén képesnek kell lennie elkezdeni a Subversion használatát. Bármilyen további szolgáltatást csak másodlagos jellegűnek minősítettek – legalábbis az 1.0-s változat elkészülte előtt.
E cikk írásakor a programfejlesztés már több mint egy éve folyt és számos kiváló programozó vett benne részt önkéntesen végzett munkájával. A Subversion a CVS-hez hasonlóan nyílt forráskódú vállalkozás.

A Subversion szolgáltatás

Az alábbiakban felsorakoztatjuk azokat az érveket, amelyek azt támasztják alá, hogy a Subversion megjelenését nagy örömmel üdvözölhetjük:

- ***Valódi másolatok és átnevezések***: a Subversion raktárrendszere egyáltalán nem használ RCS-állományokat. Ezek helyett olyan virtuális állományrendszert alkalmaz, amely nyomon követi a faszerkezetek időbeli változását (lásd alább). Az állományok és könyvtárak egyaránt változatszámmal lesznek ellátva. És végre az ügyféloldali `mv` és `cp` parancsok pontosan úgy viselkednek, ahogyan az elvárható tőlük.
- ***Atomi műveletek***: egy művelet vagy teljes mértékben bekerül a raktárrendszerbe vagy egyáltalán nem kerül be.
- ***Fejlett hálózati réteg***: a Subversion



A Subversion tervezési rétegei

hálózati kiszolgálóprogramja az Apache, az ügyfél- és kiszolgálógépek a WebDAV (2) protokoll segítségével értenek szót egymással (lásd alább a ***Subversion tervezése*** című szakaszt).

- ***Gyorsabb hálózati elérés***: a módosítások tárolására és mindkét irányú továbbítására bináris eltérésfigyelő algoritmust használnak, tekintet nélkül arra, hogy az állomány szöveges vagy bináris típusú-e.
- ***Állományrendszer-jellemzők***: minden egyes állományhoz vagy könyvtárhoz egy eloszlástábla tartozik. Tetszőlegesen kiválasztott kulcs/érték párokat adhatunk meg és tárolhatunk a felhasználókra, a jogokra, az ikonokra, az alkalmazástulajdonosra, a MIME-típusokra, a személyes megjegyzésekre stb. vonatkozóan. Ez a felhasználók számára nyújtott általános szolgáltatás. A tartalomhoz hasonlóan az állományjellemzők is kapnak változatszámot. Némelyik tulajdonság kijelölése önműködő, ilyen például az állomány MIME-típusa, ami azt jelenti, hogy a továbbiakban már nem nagyon kell a `-kb` kapcsolóra gondolnunk.
- ***Bővíthető és átalakítható***: a Subversion nem cipel „történelmi" terheket, a rendszer közös használatú C-könyvtárakra épül, jól meghatározott alkalmazási felületekkel együtt. Mindez a Subversiont rendkívül jól karbantartható, más alkalmazások és nyelvek számára kitűnően használható rendszerré teszi.
- ***Könnyű áttérés***: a Subversion parancssoros ügyfélprogramjának működése nagyon hasonlít a CVS-hez, a fejlesztői modell azonos, úgyhogy a CVS-felhasználóknak az áttéréssel nem lehet túl sok gondjuk. A `cvs2svn` raktárrendszer-átalakító program fejlesztése folyamatban van.
- ***Szabad program***: a Subversion az Apache, illetve BSD-szerű, nyílt forrású felhasználási engedéllyel jelenik meg.

A Subversion felépítése

A Subversion felépítése modulrendszerű, megvalósítása C-könyvtárak halmazán alapul. Minden egyes rétegnek jól meghatározott célja és felülete van.

Általában véve a kódfolyam a diagram tetejéről indul és lefelé halad, és minden réteg csatolófelületet biztosít a felette levő réteg számára (*ábránkon*).
Járjuk be ezeket a rétegeket az alsótól kezdve!

A Subversion állományrendszere

A Subversion állományrendszere nem rendszermagszintű állományrendszer, amelyet az operációs rendszerbe kell telepíteni – mint például a Linux ext2 állományrendszert, ehelyett inkább a Subversion raktárrendszerére utal.
A raktárrendszer, minthogy egy adatbázis tetejére épül, amely jelenleg a Berkeley DB, tulajdonképpen ***.db*** állományok halmaza. A könyvtár viszont hozzáfér ezekhez az állományokhoz, és egy C alkalmazási felületet emel be, amely egy állományrendszert utánoz, nevezetesen változatszámokat is rögzítő állományrendszert. Ez azt jelenti, hogy a raktárrendszerhez hozzáférő programot írni olyan, mint más állományrendszer számára alkalmazói programfelületet készíteni: a szokásos módon nyithatunk meg állományokat és könyvtárakat olvasásra és írásra. A lényeges eltérés az, hogy ez az állományrendszer sohasem veszít el adatot, a régi állományok és könyvtárak – mint afféle történelmi alkotások – megőrződnek. A CVS a kiszolgálóoldalon az RCS-állomány(ok) révén állományonkénti alapon módosítási számokat tárol, ezzel szemben a Subversion teljes fákat lát el változatszámmal.
A raktárrendszerben végzett apró módosítások teljesen új állományrendszerfát hoznak létre egyszerű, de mindenre kiterjedő módosításszámlálóval.
A módosított állományok és könyvtárak lemezre íródnak, miközben a régebbi állományokról visszatölthető mentés (back-up) készül, és a legfrissebb változathoz képest csak a változásokat tárolja. A változatlan bejegyzésekre az elosztott tárolási rendszerben szerepelnek hivatkozások, tehát a raktárrendszer ilyen módon nemcsak állományokat képes változatszámmal ellátni, de egymásba ágyazott könyvtárakat is.
Végül meg kell említenünk, hogy a Berkeley DB-hez hasonló adatbázis használata révén a Subversion más fontos rendszerszolgáltatást is képes nyújtani: az adatok sértetlenségét, atomi írásműveleteket, helyreállíthatóságot és a háttérállományok gyors előállítását. A részletesebb tájékozódás érdekében keresse fel a ➲ http://www.sleepycat.com címet.

A hálózati réteg

A Subversion rendszer mindenhol magán viseli az Apache nyomát. A Subversion középpontjában működő ügyfélprogram az Apache Portable Runtime (APR) hordozható futásidejű könyvtárat használja. Ez azt jelenti, hogy a Subversion ügyfélprogramja bárhol lefordítható és futtatható, ahol a HTTPD futni képes. Jelenleg ide értendő az összes Unix-változat, a Win32, BeOS, OS/2, Mac OS X, sőt talán még a NetWare operációs rendszer is.

A Subversion nem kizárólag az APR-től függ; a Subversion-kiszolgálót maga az Apache httpd démon testesíti meg. Jó-jó, de miért pont az Apache-ot választották? Végül is azért esett rá a választás, hogy ne kelljen újra feltalálni a kereket, azaz mindent nulláról kezdeni. Az Apache nyilvános forrású, komoly használatra szánt rendszer, amely már kiállta az idők próbáját, ráadásul még mindig bővíthető. Nagy a hálózati terhelhetősége, sok felületen képes futni, és még a tűzfalakkal is együtt képes működni. Számos hitelesítési protokollt használni tud. Hálózati adatcsatorna (pipeline) és gyorstár (cache) feladatok is rábízhatók, és a Subversion az Apache kiszolgálóként való felhasználásával mindehhez a szolgáltatáshoz ingyen jut hozzá. Szóval, miért is kellett volna mindent az elejéről kezdeni?

A Subversion saját hálózati protokolljaként a WebDAV-ot alkalmazza. A DAV (Distributed Authoring and Versioning) – elosztott szerzői és változat-nyilvántartó rendszer – önmagában is nagy téma (➲ http://www.webdav.org), dióhéjban azt lehet elmondani róla, hogy a HTTP kiterjesztése, amely a Világhálón állományok olvasását, illetve írását és változatkezelését teszi lehetővé. A Subversion reméli, hogy meglovagolhatja e protokoll lassan növekvő támogatását; a Win32, Mac OS X és a Gnome legfrissebb állományböngészőinek mindegyike képes ennek a protokollnak a használatára. Egymással való együttműködésül (remélhetőleg) már egyre inkább afféle ráadássá válik.

Azon felhasználók számára, akik csak helyi a lemezen használják a Subversion raktárrendszerét, az ügyfélprogram ugyanúgy működik, ám ilyenkor nincs szükség hálózatra. A raktárrendszer-hozzáférési réteg (RA) olyan leegyszerűsített alkalmazói felület, amelyet a DAV és az RA helyi elérést támogató könyvtárai valósítanak meg. Ez a könyvtárosított módosításkövető rendszer előnye és a CVS felett aratott óriási győzelem. A CVS egyfajta programot használ a helyi és egy másikat a hálózati raktárrendszer elérésére. Esetleg úgy gondolja, hogy új hálózati protokollt lenne kedve készíteni a Subversion-rendszerhez? Ha a kérdésre igennel felel, készítse el az RA alkalmazói programfelületet alkotó új könyvtárat.

Ügyfélkönyvtárak

Ügyféloldalon a Subversion munkamásolat könyvtára a rendszer működéséhez szükséges adatot különleges */SVN* könyvtárakban tárolja, amelyek a cél szempontjából a CVS-rendszerben található */CVS* nyilvántartó könyvtárakhoz hasonlítanak.

Egy átlagos */SVN* könyvtár belsejébe vetett pillantás viszont a szokásosnál jóval többet árul el magáról a rendszerről. A bejegyzések állomány XML-t tartalmaz, amely a munkamásolat könyvtár pillanatnyi állapotát írja le, és alapvetően ez az, amely a CVS-bejegyzések, a gyökér- és raktárrendszer-állomány szolgáltatásait egyesíti magában.

Ezenkívül sok minden más is megtalálható itt, ami nincs jelen a */CVS* könyvtárban, többek között a változatszámmal ellátott tulajdonságok tárolási helyei – ezeket fentebb a Subversion által nyújtott szolgáltatásoknál metaadatoknak neveztük – és minden egyes eredeti állományhoz gyorstár. Ez utóbbi szolgáltatás teszi lehetővé, hogy a helyi módosításokról és változat-újraszámozásról hálózati elérés nélkül is kimutatást tudjunk készíteni. A hitelesítési adatokat ugyancsak */SVN* könyvtárakban tárolják, semmint holmi *.cvspass*-szerű egyszerű kis állományban.

A Subversion ügyfélkönyvtárra van bízva a legnagyobb felelősség. Az a tiszte, hogy ötvözze a munkamásolat könyvtár feladatait a rendszerraktár-elérés könyvtár teendőivel, hogy ezekkel bármilyen alkalmazásnak a legmagasabb szintű felhasználói felületet biztosítsa, amely módosítási-szabályozási műveletet szeretne végezni.

Vegyük például a `svn_client_checkout()` eljárást, amely tulajdonságként egy URL-t használ. Ezt az URL-t átadja a rendszerraktár-elérési könyvtárnak és megnyit egy hitelesített munkamenetet (session) egy bizonyos rendszerraktárral. Majd a rendszerraktártól kér egy adott fát, amelyet elküld a munkamásolat könyvtárba, ekkor a teljes munkamásolat a lemezre íródik – vagyis a */SVN* könyvtárak és minden egyéb.

Az ügyfélkönyvtárat olyanra tervezték, hogy bármilyen alkalmazás használhassa. A Subversion forráskódja szabványos parancssori ügyfélprogramot tartalmaz, de az ügyfélkönyvtár tetejébe tetszőleges számú grafikus felületű felhasználói programot ültetni sem nehéz feladat. Remélhetőleg ezek a felhasználói felületek egy nap majd sokkal jobban fognak szerepelni, mint a jelenlegi CVS grafikus felületek, hiszen nem többek a CVS parancssori ügyfélprogram köré telepített törékeny csomagolásnál. Ezenkívül a megfelelő SWIG-hozzárendeléseknek (➲ http://www.swig.org) az alkalmazói programfelületet nagyszámú programnyelv számára hozzáférhetővé kell tenniük: Java, Perl, Python, Guile stb. A CVS legyőzésénél sokat számít az, hogy mindenütt jelen van.

A Subversion jövője

A Subversion 1.0 megjelenését jelenleg 2002 elejére tervezik. Az 1.0-s közzétett változat megjelenését követően a Subversiont olyan bővítmények befogadására szemelték ki, mint az i18n-támogatás, intelligens összeválogatás, jobb változáskészlet-kezelés, ügyféloldali bedolgozó programok, a kiszolgálógépnél pedig továbbfejlesztett szolgáltatások. A kívánságlistán szerepel még néhány ötlet, amelyekben a rendszerraktár elosztása, majd arról másodpéldány készítése kerül szóba.

Végül hadd idézzünk egy gondolatot a Subversion Gyakran Ismétlődő Kérdéseiből (GYK): „Mi (még) nem próbálunk meg új utat törni az SCM-rendszerekben, sőt még csak meg sem kíséreljük utánozni az SCM-rendszerek legjobb tulajdonságait. A CVS-rendszer kiváltására törekszünk. Ha három év múlva a Subversiont a Nyílt Forráskód Közösségében az SCM-rendszerek szabványaként fogják számon tartani, a vállalkozás már sikerrel járt. De a jövő egyelőre még ködös és titokzatos.

Végsősoron a Subversionnek ezt a helyet a saját érdemei alapján kell kiérdemelnie.

Javítófoltokat viszont továbbra is szívesen fogadunk.

Linux Journal 2002. február, 94. szám

Ben Collins-Sussman
már 11 éve dolgozik programozóként és rendszergazdaként különböző intézményeknél. Ben jelenleg a Collabnetnél, a Subversion fő támogatójánál tevékenykedik. Honlapja a ➲ http://www.red-bean.com/sussman címen olvasható.

A Linux-csomagszűrő felépítése (1. rész)

A Linux-csomagszűrőről szóló kétrészes sorozatunk első részében Gianluca elmeséli, milyen utat tesz meg egy csomag a rendszermagon keresztül, míg eljut a rendeltetési helyére.

Talán néhányan közületek emlékeznek még „Linux-csomagszűrő: bájtok elfogása a hálózatban" című cikkemre, mely a Linuxvilág júniusi számában jelent meg, és a Linux-rendszermagba épített csomagszűrő működését taglalja. Abban a cikkben a csomagszűrő működését tekintettük át, ezúttal pedig a rendszermag azon részébe ásom be magam, amely a szűrésért felelős, és egyúttal a rendszermag csomagkezelő részébe is betekintést nyújtok.

Az előző cikk témái

Legutóbbi cikkemben felmerült néhány dolog a Linux csomagkezelő rendszerével kapcsolatban. Legjobb lesz, ha ezek közül a legfontosabbakat most újra átvesszük:

- A csomagok átvétele először a hálózati kártya eszközmeghajtójának szintjén zajlik, pontosabban annak megszakításkezelő eljárásában, és az eszközmeghajtó továbbítja egy felsőbb szintre, ahol később feldolgozásra kerül.
- Amennyiben a fogadás és protokollértelmezés művelete során a rendszer túlságosan elfoglalt, néhány csomagot eldob. Az elkövetkezőkben, amint a csomag egyre közelebb ér a felhasználói szinthez, az alacsonyabb szintű protokollok adatai elvesznek.
- Foglalati szinten, közvetlenül mielőtt a csomag elérne a felhasználói szintre, a rendszermag ellenőrzi, hogy az adott csomag számára létezik-e nyitott foglalat – amennyiben nincs, a csomagot eldobja.
- Az ezt követő szinten a Linux-rendszermag `PF_PACKET` nevű szolgáltatása érhető el. Ezzel olyan foglalatokat lehet létrehozni, amelyek közvetlenül a hálózati kártya meghajtójától fogadnak adatokat. Ily módon minden egyéb protokollkezelő eljárás átugorható, és bármilyen csomagot elkaphatunk.
- Az ethernetkártya többnyire csak az olyan csomagokat továbbítja, amelyek magának a rendszermagnak szólnak, minden egyebet eldobnak. Mindazonáltal létezik egy olyan beállítás, amely lehetővé teszi, hogy a hálózati kártyán áthaladó összes csomag MAC-címétől (promiscuous mód) függetlenül továbbítódjon.
- Legvégül pedig a foglalaton egy szűrőt is elhelyezhetsz, hogy csak azok a csomagok továbbítódjanak, amelyek megfelelnek a szűrő beállításainak. Ezt egy `PF_PACKET` foglalattal összekapcsolva remek eszközhöz jutunk hálózatunk forgalmának hatékony elkapására (sniff).

Bár lehallgatónk a `PF_PACKET` foglalatokon alapul, a Linux foglalati szűrője nincs az ilyen eszközök előállítására korlátozva. Valójában ezzel a foglalattal egyszerű TCP- és UDP-csomagokat is elkaphatunk, hogy kiszűrjük a nem kívánt forgalmat – természetesen ez a lehetőség kevésbé szokványos. Az elkövetkezőkben többször hivatkozom e foglalatokra és foglalati felépítésre. Ezen írás keretein belül a két fogalmon ugyanazt értjük, mivel a foglalat valójában a rendszermag belsejében értelmezett foglalati felépítést takarja. A rendszermag számon tart egy foglalati felépítést, továbbá a foglalatok kezeléséhez is egyet, de a kettő közötti különbségtől most eltekintünk.

Egy másik lényeges szerkezet, amely még gyakran szóba kerül, az `sk_buff` (vagy más néven foglalati átmeneti tár), mely a csomagok rendszermagon belüli ábrázolásáért felelős. Ez a szerkezet olyan módon van felépítve, hogy az egyes fejlécadatok hozzáadása vagy elvétele a legkevésbé legyen költséges: nincs szükség adatok másolására, mivel az összes adatváltoztatást mutatók eltolásával oldották meg.

Mielőtt továbblépünk, nem árt, ha megvilágítjuk az eddigieket. A Linux foglalati szűrőnek nevével ellentétben semmi köze a Netfilterhez (lásd még a Linuxvilág 2001 októberi számában). A Netfilter valamikor a rendszermag 2.3-as sorozatának legelején jelent meg, és egészen más a szerepe. Bár a foglalati szűrőkhöz hasonlóan ennek is a csomagok felhasználói szintre juttatása a feladata, ez más célból történik: hálózati címváltásból (NAT), a csomagok felhasználói szintű módosításából, kapcsolatkövetésből, biztonsági megfontolásból és így tovább. Ha mindössze csak a csomagok elkapására (sniff) van szükséged, ennek legkézenfekvőbb eszköze a Linux-csomagszűrő. Most már folytathatjuk a csomagkövetést a számítógépre való belépéstől egészen egy felhasználói programnak történő kézbesítésig. Először egy általános foglalatot (nem `PF_PACKET`) veszünk szemügyre. Kapcsolati szinten egy ethernethálózatból indulunk ki, mivel ez a legelterjedtebb, és egyúttal a legkönnyebben érthető is. Alapvetően minden más kapcsolati szintű protokoll az ethernethez hasonlítható, többnyire nincsenek lényeges különbségek.

Ethernetkártya- és alacsony rendszermagszintű belépés

Mint előző cikkünkben már szó volt róla, az ethernetkártyának van egy hálózati szintű, úgynevezett MAC-címe, és az erre a címre érkező csomagokat emeli ki a hálózatból. Amikor a kártya egy olyan csomaggal találkozik, amely a saját címére szól vagy az úgynevezett Broadcast-címre (FF:FF:FF:FF:FF:FF ethernet esetén), a csomagot a memóriaterületére másolja. Miután az ethernetkártya a csomagot teljes egészében átvette, megszakítási kérelmet hoz létre, amelyet a hálózati kártya meghajtója kezel. Ezalatt a meghajtóprogram az egyéb megszakítások beérkezését letiltja, és többnyire a következő feladatokat hajtja végre:

- Lefoglal egy újabb `sk_buff` szerkezetet, az ***include/linux/skbuff.h***-ban található rendszermagszintű értelmezésnek megfelelően. Ez a szerkezet a rendszermag szemszögéből a beérkező csomagokat ábrázolja.
- A kártya átmeneti tárából az újonnan lefoglalt `sk_buff`

tárterületére másolja a csomagot, mely a DMA felhasználásával is megtörténhet.
- Meghívja a `netif_rx()` függvényt, ez az általános, csomagok beérkezésekor meghívásra kerülő függvény.
- Miután a `netif_rx()` visszatér, újra engedélyezi a megszakításokat, és befejezi a műveletet.

A `netif_rx()` függvény a rendszermagot a beérkezett csomaggal elvégzendő következő műveletre is felkészíti; az `sk_buff`-ot a CPU pillanatnyi beérkező várakozási sorába helyezi, és a `NET_RX` `softirq`-t (erre még visszatérünk) megjelöli, hogy az a `__cpu_raise_softirq()` rendszerhíváson keresztül meghívódjon. Ezzel kapcsolatban meg kell említenünk két fontos dolgot: először is, ha ez a várakozási sor már megtelt, az új csomagot a rendszer eldobja; másodszor: processzoronként csak egy várakozási sorunk van, az új késleltetett feldolgozómodellel (softirq) így több processzor esetén lehetővé válik a csomagok párhuzamos feldolgozása.
Ha egy valódi ethernetmeghajtót működés közben szeretnél látni, vess egy pillantást a NE2000 kártya PCI-meghajtójára, melyet a ***drivers/net/8390.c***-ben találsz; a hálózati kártya megszakítását az `ei_internet()` kapja el, majd meghívja az `ei_receive()` függvényt, ami a következőket hajtja végre:

- A `dev_alloc_skb()` függvényhíváson keresztül helyet foglal egy új `sk_buff` szerkezetnek.
- A csomagot kiolvassa a kártya átmeneti tárából (`ei_block_input()` függvény), és ennek megfelelően állítja be az `skb->protocol`-t.
- Meghívja a `netif_rx()`-et.
- A fentieket legfeljebb tíz egymást követő csomagra vonatkozóan megismétli.

Ennél jóval bonyolultabb megoldást mutat be a ***3c59x.c*** fájlban található 3COM-meghajtó, mely a csomagot a DMA segítségével másolja a kártya tárterületéről az `sk_buff` területére.

Hálózati szintű feldolgozás

Nézzük meg közelebbről a `netif_rx()` függvényt! Mint már említettem, ennek a függvénynek az a feladata, hogy fogadja a csomagot a hálózati kártya meghajtójától, és továbbítsa a magasabb szintű rétegek felé. Ez a függvény szolgál központi gyűjtőhelyül a különböző hálózati kártyameghajtóktól beérkezett csomagoknak, amelyekkel a felsőbb szintű protokollokat látja el.
Mivel ez a függvény megszakítási környezetben fut (azaz a korábban keletkezett megszakítás hatására fut le), működése közben a megszakítások le vannak tiltva, ugyanakkor rövidnek és gyorsnak kell lennie. Nem hajthat végre hosszadalmas ellenőrzést vagy más bonyolult feladatot, mert amíg ezen a függvényen belül vagyunk, csomagok veszhetnek el. Tehát a `netif_rx()` függvény nem tesz egyebet, mint a `softnet_data` tömbből kijelöli a csomag számára megfelelő várakozási sort, és a futtató processzortól függően kiválasztja a tömb indexét. Ezután ellenőrzi a várakozási sor állapotát, majd besorolja az öt lehetséges torlódási szint egyikére: `NET_RX_SUCCESS` (nincs torlódás), `NET_RX_CN_LOW`, `NET_RX_CN_MOD`, `NET_RX_CN_HIGH` (azaz alacsony, közepes vagy magas torlódási szint), illetve `NET_RX_DROP` (a rendszer a nagyon magas torlódási szint miatt a csomagot eldobja).
A `netif_rx()` függvény az esetleges hálózati szakadás elkerülése végett forgalomkorlátozó politikát tart érvényben, amely – ha a torlódás kritikus szintet ér el – a rendszert torlódásmentes állapotba juttatja vissza. Ennek egyik előnye, hogy segítségével a lehetséges DoS-támadások elháríthatók. Átlagos körülmények között a csomag végül a várakozási sorba (`__skb_queue_tail()`) továbbítódik, és a `__cpu_raise_softirq(cpuid, NET_IF_SOFTIRQ)` függvény hívódik meg. Ez utóbbi függvényhívás ütemezi a `softirq` futtatását.
Amikor a `netif_rx()` függvény befejezi munkáját, visszatérési értékével a hívó számára jelzi a pillanatnyi torlódási szintet. Ezen a ponton a megszakítási környezet véget ér, és a felsőbb szintű protokollok készen állnak arra, hogy a csomagot fogadják. A feldolgozás egy későbbi időpontra halasztódik, amikorra a megszakítások ismét engedélyezettek lesznek, és a feldolgozási idő kevésbé kritikus. Ez a késleltetett feldolgozási módszer a 2.2-es és 2.4-es rendszermagok között teljesen eltérő: a 2.2-es rendszermagok az úgynevezett `als felek`-kel dolgoznak, a 2.4-es rendszermagban a működés már a `softirq`-n alapul.

Alsó felek a softirq-k ellenében

Az `als felek`, vagyis az angol rövidítésnek megfelelően a `BH`-k részletes ismertetése meghaladja e cikk kereteit, így egy-két jellemző tulajdonságára csak pontokba szedve térünk ki.
Először is a `BH`-k tervezésekor az alapvető megszakítás környezetbeli politikának megfelelően elsődleges szempont volt, hogy ebben az üzemmódban a rendszermagnak a lehető legkevesebb számítást kelljen elvégeznie. Ezáltal ha bonyolultabb műveletet kellett valamilyen megszakításkérésre elvégezni, a megszakítási környezetben csak az ehhez szükséges `BH` került kijelölésre, anélkül, hogy bármilyen bonyolult művelet foglalta volna le a rendszert. Egy későbbi időpontban a rendszermag ellenőrizte a `BH`-maszkot, hogy eldöntse, valamelyik `BH` futásra vár-e, és amennyiben igen, még az alkalmazásszintű feladat előtt lefuttatta.
A `BH`-k egészen jól működtek, csupán egyetlen hátrányuk akadt: a felépítésük miatt ugyanaz a `BH` egy időben csak egyetlen processzoron futhatott. Ez több processzor között a több CPU-val működő SMP-rendszerekben mindennemű párhuzamosságot és feladatelosztást meggátolt, ugyanakkor a teljesítményt is jelentősen rontotta. A `softirq`-k a `BH`-k 2.4-es rendszermagbeli követőikkel, és az úgynevezett `tasklet`-ekkel együtt a rendszermag szoftveres megszakításcsaládjához tartoznak, azaz olyan kódok, melyeket igény esetén a rendszermag hajt végre, anélkül, hogy a feladatra rendelkezésre álló idő szigorúan meg lenne határozva. A legfontosabb különbség a `BH`-khoz képest, hogy egy `softirq` ugyanabban az időben akár több CPU-n is futhat. Több feladat egyidejű futása esetén a feladatok a rendszermag spinlockjaival hangolhatók össze.

A softirq-k működése

A `softirq`-k feldolgozómagja a `do_softirq()` függvényben található, a ***kernel/softirq.c*** fájlban. A függvény először egy bitmaszkot ellenőriz, majd ha a megfelelő `softirq` bit be van kapcsolva, meghívja a megfelelő kezelőeljárást.
A `NET_RX_SOFTIRQ` esetében a `net_rx_softirq()` függvény érdekel bennünket, ami a ***net/core/dev.c*** fájlban található.
A `do_softirq()` függvény három különböző helyről hívható meg a rendszermagban:

- a ***kernel/irq.c*** fájlban található `do_IRQ()` függvényből (ez az általános megszakításkezelő);
- rendszerhívások kilépésekor a ***kernel/entry.S*** fájlból;

- a `schedule()` függvényből a ***kernel/sched.c*** fájlból, mely a fő feladatütemező függvény.

Más szavakkal a `softirq` meghívódhat, ha a rendszer alkatrészmegszakítást kezelt, ha egy alkalmazásszintű program egy rendszerhívást kér, vagy ha új feladat (process) kerül végrehajtásra. Ily módon a `softirq`-kat olyan gyakran kezeli, hogy közülük egyiknek sem szükséges túl hosszan várakoznia. A kioldórendszer a már elavult `als felek` esetén pontosan ugyanígy működött.

A NET_RX softirq

Mint láthattuk, a csomagok a hálózatról a hálózati kártyán keresztül kerülnek a rendszerbe, majd egy várakozósorba jutnak, ahonnan a rendszer csak később dolgozza fel őket. Megfigyelhettük, miként folytatódik a feldolgozás a `net_rx_action()` függvényen keresztül. Most annak is eljött az ideje, hogy beletekintsünk ebbe a függvénybe.
E függvény feladata alapvetően nagyon egyszerű: a pillanatnyi CPU-hoz tartozó várakozósorról leemeli az első csomagot (`sk_buff`), majd végigfut a két csomagkezelő listán, és ennek alapján meghívja a megfelelő csomagkezelő függvényeket. Érdemes néhány szót ejteni erről a két listáról: vajon miként működnek, és hogyan épülnek fel? A két lista neve ***ptype_all*** és ***ptype_base***, és protokollkezelőket tartalmaznak az általános csomagokhoz, valamint bizonyos csomagtípusokhoz. A protokollkezelők önmagukat jegyzik be: vagy a rendszermag indulásakor, vagy bizonyos foglalati típusok létrejöttekor, megadva, hogy milyen protokolltípusok kezelésére képesek.
Az ehhez kapcsolódó függvény a `dev_add_pack()`, mely a ***net/core/dev.c***-ben található. Ez egy csomagtípus-szerkezetet (lásd az ***include/linux/netdevice.h*** fájlt) hoz létre, amely mutatót tartalmaz arra a függvényre, amelynek meg kell hívódnia, ha ilyen típusú csomag érkezik. A bejegyzés során minden csomagkezelő függvény belekerül a két lista valamelyikébe, azaz vagy a ***ptype_all*** listába (`ETH_P_ALL`-típusok esetén), vagy a ***ptype_base*** listába (minden egyéb `ETH_P_*` típus esetén).
A `NET_RX softirq` mindössze annyit tesz, hogy a csomagok protokolltípusához tartozó protokollkezelőket sorban meghívja. Először az általános protokollkezelők hívódnak meg (azaz a ***ptype_all*** protokollok), ahol az egyes protokollok fajtája nem számít, majd ezt követően az egyedi protokollkezelők következnek. Mint látni fogjuk, a `PF_PACKET` protokoll mindkét listába belekerülhet, attól függően, hogy az alkalmazás milyen foglalati típust választ. Másrészről az alap-IP-kezelő a második listába kerül, az azonosítója pedig `ETH_P_IP` lesz.
Amennyiben a várakozósor több csomagot is tartalmaz, a `net_rx_action()` az összes csomagon – amíg fel nem dolgozza a legnagyobb feldolgozható számú csomagot (`netdev_max_backlog`), vagy míg túl sok időt nem tölt el a csomagok feldolgozásával (a legtöbb rendszermag számára ez a túl sok idő egy pillanatot jelent, azaz körülbelül 10 ms-ot). Ha a `net_rx_action()`-nek nem sikerül befejeznie a feldolgozást, és csomagok maradnak a várakozósorban, akkor újra engedélyezi a `NET_RX_SOFTIRQ`-t, és a lemaradt csomagok egy későbbi alkalommal kerülnek feldolgozásra.

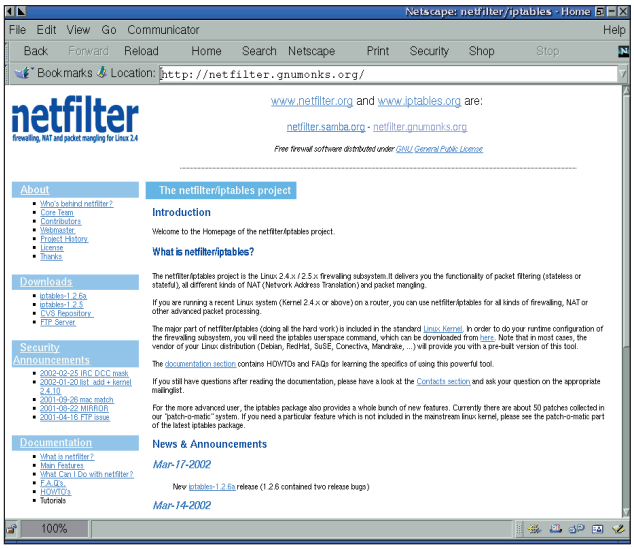
Az IP-csomagkezelő

Az IP fogadófüggvényre, nevezetesen az `ip_rcv()`-re (***net/ipv4/ip_input.c***), a rendszermag indulásakor bejegyzett csomagtípus-szerkezetre mutat az `ip_init()` függvény (a ***net/ipv4/ip_output.c*** forrásfájlban). Az IP bejegyzett protokolltípusa: `ETH_P_IP`. Ennek következtében az `ip_rcv()`-t a `net_rx_action()` függvény hívja meg a `softirq` feldolgozása során, ha egy `ETH_P_IP`-típusú csomag kerül feldolgozásra. Ez a függvény végez el minden kezdeti ellenőrzést az IP-csomagon, mely nagyjából annyiból áll, hogy ellenőrzi az IP-csomag épségét (az IP-ellenőrzőösszegét, az IP-fejlécmezőket, és a legkisebb jelentős csomagméretet). Ha a csomag megfelelőnek tűnik, az `ip_rcv_finish()` függvény hívódik meg. Csak mellékesen jegyzem meg, hogy ennek a függvénynek a meghívása keresztülmegy a Netfilter előútválasztó ellenőrzési pontján, melyet gyakorlatilag az `NF_HOOK` makróval valósítanak meg.
Az `ip_rcv_finish()` – még mindig az ***ip_input.c***-nél maradva – nagyrészt az IP-útválasztó szolgáltatásáért felelős. Ez ellenőrzi, hogy vajon a csomagot továbbítani kell-e egy másik gépre, vagy pedig helyben kell feldolgozni. Az első esetben elvégzi az útválasztást, és a csomagot a megfelelő hálózati eszköz felé irányítja; a második esetben a csomagot helyben kézbesíti. A szemfényvesztést az `ip_route_input()` függvény valósítja meg, mely az `ip_rcv_finish()` legelején hívódik meg, és amely eldönti a csomag következő feldolgozási pontját, a mutatót a megfelelő függvényre állítva be az `skb->dst->input`-ban. A helyben kézbesítendő csomagok esetén ez a mutató az `ip_local_deliver()` függvényre mutat. Az `ip_rcv_finish()` függvény egy `skb->dst->input()` függvényhívással zárul.
Ezen a ponton a csomagok már ténylegesen a felsőbb szintű protokollok felé utaznak. Az irányítás az `ip_local_deliver()`-hez kerül; ez a függvény felelős az IP-töredékek összeállításáért (abban az esetben, ha az IP-adat-

PF_PACKET foglalatok létrehozása

A `PF_PACKET` foglalat története akkor kezd el különbözni az általános foglalatokétól (pl. `PF_INET`), amikor a `socket()` rendszerhívás meghívódik. A `socket()` rendszerhívás (melyet a rendszermagban a `sys_socket()` kezel, a *net/socket.c* forrásfájlban található) a `sock_create()` függvényt használja fel, hogy eldöntse, milyen műveletek tartoznak a megadott protokollcsaládhoz. Ez az adat a `net_families` tömbből származik, melynek tartalma a rendszermag indulásakor dől el a `sock_register()` függvény hatására. Ezen a ponton a rendszermag képes arra, hogy meghívja az adott protokollcsaládhoz tartozó `create()` függvényt, mely létrehoz egy ennek megfelelő foglalati felépítést, és befejezi a létrehozási szakaszt.
`PF_PACKET`-ek esetén a `create()` függvény a *net/packet/af_packet.c* fájlban található *packet_create()* függvénynek felel meg. Egyebek között ennek a feladata, hogy a hivatkozásokat beírja a foglalati felépítésbe, a protokollra jellemző műveletekre. Ez nem egyezik meg a protokollcsaládra jellemző műveletekkel, mivel ez utóbbi csak egy mutatót tartalmaz a `create()` függvényre, míg a protokollra jellemző műveletek egy egész sor függvényhívásra tartalmaznak hivatkozást, amelyeket egy adott foglalaton végrehajthatunk (például `accept`, `connect`, `bind`, `ioctl`, `listen`, `sendmsg`, és így tovább – a teljes listához vess egy pillantást az *include/linux/net.h-ra*.)

gram szét van töredezve). Ezt követően a vezérlés átadódik az `ip_local_deliver_finish()` függvénynek. Mielőtt azonban ez meghívásra kerülne, egy újabb Netfilter-kapocs hajtódik végre (az `ip-local-ip`).
Az `ip_local_deliver_finish()` – mely az IP feldolgozási rétegben található utolsó függvény – hajtja végre a 3. réteg befejezéséhez szükséges még várakozó feladatokat. Az IP-fejléc adatait összerendezi, és továbbítja a 4. szinten lévő protokollnak, valamint megvizsgálja, hogy a csomag nem tartozik-e valamelyik nyers (raw) IP-foglalathoz, és ha igen, továbbítja a megfelelő kezelőnek (`raw_v4_input()`).



A nyers (raw) IP olyan protokoll, melynek segítségével az alkalmazásoknak lehetőségük nyílik arra, hogy olyan IP-csomagokat állítsanak össze, illetve fogadjanak közvetlenül, amelyek a 4. szinten még nem kerültek feldolgozásra. Ennek a protokollnak elsősorban hálózati alkalmazások veszik hasznát, ugyanis feladatuk elvégzéséhez szükségük lehet bizonyos csomagok elküldésére. Néhány jól ismert eszköz, mint a `ping` vagy a `traceroute` úgyszintén nyers IP-t használ a saját csomagjai felépítéséhez, hogy a csomagokhoz egyéni fejléc-adatokat adjon hozzá.
A nyers IP lehetőséget biztosít hálózati szintű protokollok felhasználói szinten történő kezelésére és létrehozására. Egy ilyen, felhasználói szinten létrehozott protokoll például az RSVP is (erőforrás lefoglaló protokoll). A nyers IP valójában nem más, mint a `PF_PACKET` megvalósítása az OSI (szabvány a nyílt rendszerek közti kapcsolattartásra) eggyel magasabb rétegében.
Ezt követően a csomagok tovább utaznak, és egy újabb rétegbe jutnak, ahol a rendszermag részéről egy újabb protokollkezelővel találják szemben magukat. Hogy ez a protokollkezelő melyik lesz, azt az IP-fejlécbe írt ***Protocol*** mező elemzésével dönti el a rendszer. Az ezen a szinten használt módszer, amely azt dönti el, hogy a csomag hogyan folytassa útját, nagyon hasonló ahhoz, mint amit a `net_rx_action()` függvénynél már megismerhettünk. Egy tábla `inet_protos` néven lett megadva, melyben az összes utólagos IP-kezelő protokoll be van jegyezve. A táblázat kulcsait a rendszer természetesen az IP-fejléc ***Protocol*** mezőjéből veszi. Ezt a táblázatot az `inet_init()` függvény tölti fel, mely a rendszermag indulásakor hajtódik végre (***net/ipv4/af_inet.c***). Ez a függvény hívja meg az `inet_add_protocol()`-t külön-külön a TCP-re, az UDP-re, az ICMP-re és az IGMP-re (csak akkor, ha a multicast, azaz a többrétegű címzés engedélyezett). A teljes protokolltáblázat a ***net/ipv4/protocol.c***-ben található.
Minden egyes protokoll számára létezik egy kezelőfüggvény: `tcp_v4_rcv()`, `udp_rcv()`, `icmp_rcv()` és `igmp_rcv()`, amelyek neve egyértelműen utal arra, hogy melyik függvény melyik protokollt kezeli. E függvények akkor kerülnek meghívásra, ha valamilyen feldolgozandó csomag vár rájuk. A visszatérési értékből dől el, hogy vajon egy ICMP- ***destination unreachable*** (Cél nem elérhető el) üzenetet kell-e küldeni válaszul a küldőnek. Ez olyankor fordul elő, amikor egy felsőbb szintű protokoll a beérkező csomagot egyetlen foglalathoz tartozóként sem ismeri el. Talán még emlékszel rá az előző cikkből, hogy az elfogás egyik alapelve az, hogy minden csomaghoz hozzájussunk – tekintet nélkül arra, hogy milyen kapura vagy milyen címre küldték őket. És itt (illetve az előbb említett `_rcv()` függvényeknél) ez az első akadály, mellyel szembetaláljuk magunkat.

Összegzés

Jelen pillanatban a csomag útjának már valamivel több mint felét megtette. Mivel szeretett újságunkban csak korlátozott hely állt rendelkezésünkre, a jövő hónapig a csomagot a 3. rétegben hagyjuk. Hátravan még a 4. réteg megismerése (TCP és UDP), a `PF_PACKET` kezelése és természetesen a csomagszűrő felépítése. Légy türelmes!

Linux Journal 2002. február, 94. szám

Gianluca Insolvibile
már a 0.99pl4-es rendszermag óta Linux-rajongó. Jelenleg hálózat- és digitális videokutatással, valamint -fejlesztéssel foglalkozik. A g.insolvibile@cpr.it címen érhető el.

További érdekességek

A Netfilterrel és az `ipfw`, IP Chains vagy IP Tablesszel kapcsolatban nagy segítségedre lehetnek Rusty „megbízhatatlan ismertetői” (Unreliable guides) a
➲ http://netfilter.gnumonks.org címen.

Biztonsági szempontból a Linuxvilág 2001. 10. számában található egy elemzés a Netfilterről: *David A. Bandel* A Netfilter megszelídítése című írásában.

Kapcsolódó címek

- ➲ http://www.tldp.org/LDP/nag2/
- ➲ http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2817396,00.html
- ➲ http://www.oreilly.com/catalog/linag2/
- ➲ http://www.unixreview.com/documents/urm0106p/
- ➲ http://kernelnewbies.org/documents/
- ➲ http://www.linux.org/books/various/admin2.html


linuxvilág
www.linuxvilag.hu

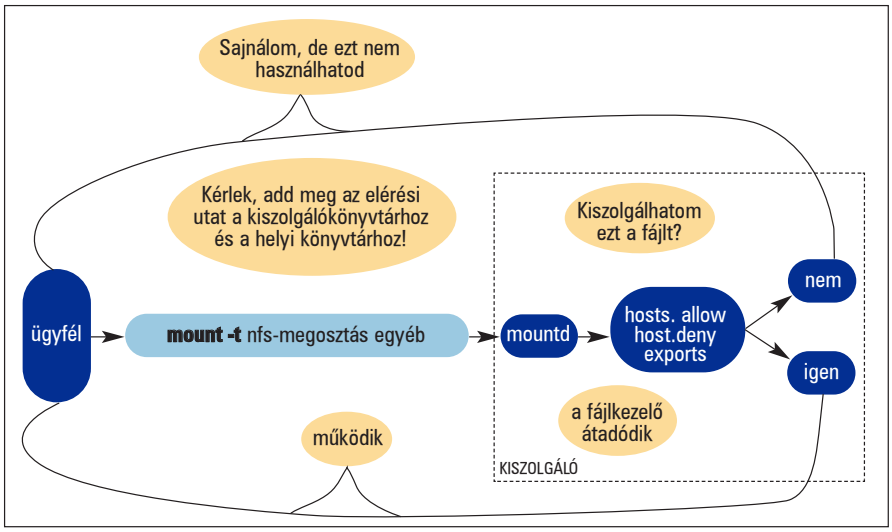
Fájlmegosztás NFS-sel

Olexij és Denis az NFS hálózati fájlrendszert mutatja be.

Előfordulhat, hogy több gépünk van, és szeretnénk a lemezterületeket, a különböző eszközöket vagy valamelyik CD-meghajtót megosztani. Erre használatosak a hálózati fájlrendszerek, például az NFS, amellyel az állományok és a források hálózati megosztása könnyen megvalósítható.
Az NFS használatával úgy dolgozhatunk egy távoli gép fájljaival, mintha a saját gépünkön lennének; a például tulajdonképpen a hálózat bármely gépéről ugyanazt a könyvtárat használhatjuk saját könyvtárként. Nagy előnye még, hogy nem kell több gépre másolgatni a fájljainkat, így nagy lemezterületeket takaríthatunk meg.
Az NFS-t a Sun Microsystems mérnökei vezették be 1985-ben. Igaz, régi, de még mindig jó, ráadásul folyamatosan fejlesztik, javítgatják. A Sun mostanában a linuxos NFS négyes kiadásán dolgozik. Mi most ugyanennek a harmadik kiadását ismertetjük – egy felhasználó számára ugyanis az egyes kiadások között nincs sok különbség.
Az NFS-t sem az ügyfél-, sem a kiszolgálógépre nem túl bonyolult telepíteni. Bemutatunk néhány egyszerű lehetőséget, de vigyázat: a legtöbb parancsot rendszergazdaként kell végrehajtani! Nézzük, hogyan is működik! Az NFS a legismertebb szolgáltatás, amely távoli eljáráshívást (RPC) használ. Például legyen a kiszolgáló neve ***tiger***, és a saját könyvtárak legyenek a ***/home*** alatt. ***Blackcat*** nevű gépünkről adjuk ki a következő parancsot:

```
mount -t nfs tiger:/home /home
```

A `mount` a fenti parancs alapján távoli eljáráshívás segítségével rákapcsolódik a kiszolgálógép `rpc.mountd` démonjára. A kiszolgáló ellenőrzi, hogy a ***blackcat*** jogosult-e hozzáférni a ***/home*** könyvtárhoz. Ha igen, akkor egy fájlkezelőt küld vissza, amelynek segítségével elérhetjük a ***/home*** könyvtár tartalmát. Ha nem szabad, hibaüzenetet ír ki. A kettőspont már jelzi a parancsban, hogy távoli fájlrendszert akarunk befűzni, így a `-t nfs` kapcsolót akár el is hagyhatjuk. Ha megkapjuk a fájlkezelőt, máris megszelídítettük a tigrist. A befűzés folyamata az *1. ábrán* látható.
Amikor a ***blackcat*** fájlt kér a hálózati fájlrendszerről, a rendszermag távoli eljáráshívást intéz az NFS-démon felé, amely általában az `rpc.nfsd`. A hívás tartalmazza a fájl nevét, felhasználó- és csoportazonosítóját, a démon ez alapján dönt, hogy kiszolgálja-e a kérést.



1. ábra Egy NFS-megosztás befűzése

Az NFS-kiszolgáló beállítása

Ha linuxos gépet szeretnénk NFS-kiszolgálóként használni, az `rpc.mountd` programot kell futtatnunk. A program segíti a befűzés folyamatát, majd a hívást átadja az `rpc.nfsd` programnak, amely minden további kiszolgálófeladatot elvégez. Az RPC-protokoll támogatja, hogy a kiszolgálógép rendszermagja teljesítményadatokat adjon ki, ami nagyban segíti az `rpc.lockd` és az `rpc.rquotad` munkáját. A 2.2.18-as és ennél újabb rendszermagok esetében az `rpc.nfsd` magától elindítja az `rpc.lockd` programot, tehát nem nekünk kell külön megtennünk. Tárkorlátok figyeléséhez az `rpc.quotad` démont kell futtatnunk.
Az eljáráshívás előnye azt jelenti, hogy az `inetd` internet-kiszolgáló ***/etc/inetd.conf*** fájljába nem kell beleírnunk.
Az NFS egy másik programot használ, a kapu-hozzárendelőt, (portmapper), amely segít megtalálni a hálózat NFS-szolgáltatásait. Az első védelmi vonalat, amely rendszerünket vigyázza, a ***/etc/hosts.allow*** és a ***/etc/hosts.deny*** nevű fájlok alkotják. Amennyiben a beérkező kérelemnek megfelelő bejegyzés szerepel a host.allow fájlban, a kérelem továbbjut. Ha itt nincs, az ellenőrzés a host.deny vizsgálatával folytatódik. Ha itt szerepel egy illeszkedő bejegyzés, a kérést a kiszolgáló elutasítja. Ha sem a ***/etc/hosts.allow***, sem a ***/etc/hosts.deny*** fájlban nincs megfelelő bejegyzés, a kérelmet akkor is továbbítja.
Ha a kiszolgálót például egy oktatási intézmény részére telepítjük, jobban oda kell figyelnünk a biztonságra. A hallgatók sok-sok gonoszságot művelhetnek: ellophatják mások jelszavát, „I love you” tárgyú leveleket küldözgethetnek vagy trójai falovakkal ajándékozhatják meg társaikat. Az NFS, főleg a régebbi kiadások, jókora biztonsági hézagot jelenthetnek, tehát jobb, ha a beállításkor odafigyelünk.
Az NFS súgóoldalán többet is megtudhatunk erről. Igaz, általában elég, ha az összes gépet letiltjuk, és csak néhányat engedélyezünk. A következő sorokat tegyük a ***/etc/hosts.deny*** nevű fájlba:

```
portmap:ALL
lockd:ALL
mountd:ALL
rquotad:ALL
statd:ALL
```

Ezek után senki nem tud az NFS-kiszolgálóra kapcsolódni, de mivel ezt nyilván nem akarjuk, nézzük a */etc/hosts.allow* állományt. A fájl szerkezete a következő:

```
dØmonlista: ... felhasznÆl @gØp-minta
```

Ha például a ***blackcat*** (`192.168.16.13`) és a ***tomcat*** (`192.168.16.24`) gépeknek engedélyezni akarjuk a hálózati fájlrendszerhez való hozzáférést, a következő sorokat adjuk a */etc/hosts.allow* fájlhoz:

```
portmap  :   192.168.16.13  192.168.16.24
lockd    :   192.168.16.13  192.168.16.24
mountd   :   192.168.16.13  192.168.16.24
rquotad  :   192.168.16.13  192.168.16.24
statd    :   192.168.16.13  192.168.16.24
```

Ha egy adott alhálózaton lévő összes gépet engedélyezni akarjuk, egy ilyenfajta lista meglehetősen hosszúra nőhet, ezért az egyszerűség kedvéért a szóban forgó gépek címét a következőképpen is megadhatjuk: az `192.168.16.0/255.255.255` számsorozat hozzáadásával például az összes olyan gépet engedélyezzük, amelynek IP-címe `192.168.16.0` és `192.168.16.255` közé esik.
Ez idáig csak nagyon általános lehetőségeket említettünk. Az NFS-szolgáltatás beállítófájlja (/etc/exports) az eddigieknél egyedibb. A fent említett két gép számára megnyitjuk a */home* és a */usr/doc* könyvtárakat:

```
/home     192.168.16.11(rw,root_squash)
➥192.168.16.24(rw)
/usr/doc 192.168.16.11(ro,root_squash)
➥192.168.16.24(ro)
```

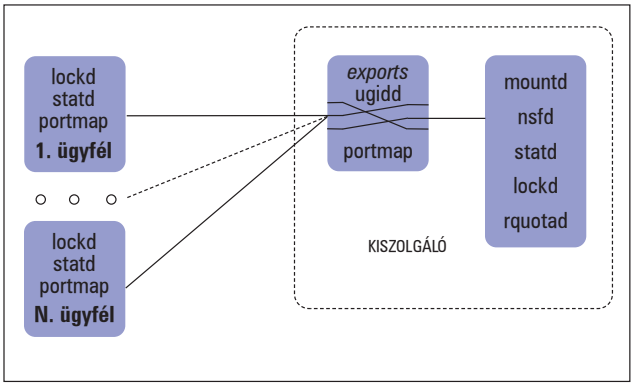
A fájl szerkezete egyszerű. A bal oldali mezőbe kerül a könyvtár neve, majd a gép IP-címe (zárójelben a jogosultságokkal). Fontos, hogy az IP-cím után nincs szóköz! A fenti példában írási és olvasási engedélyünk van a */home* könyvtárra, de a */usr/doc* könyvtárra csak olvasási jogunk.
A `root_squash` érték ahhoz kell, hogy megakadályozzuk az ügyfélgépről történő rendszergazdai hozzáférést. Ez azt jelenti, hogy hiába szerezte meg egy felhasználó, mondjuk egy hallgató az ügyfélgép rendszergazdai jelszavát, nem tud rendszergazdaként dolgozni a kiszolgálón lévő fájlokkal. Ez az alapértelmezett beállítás. A `squash` szó itt azt jelenti, hogy a rendszergazda csak a nobody nevű felhasználó jogait birtokolja.
Ha jól átgondoljuk a rendszergazdai hozzáférés példáját, észrevehetjük, hogy valamit a felhasználó- és csoportazonosítókkal kapcsolatban is tennünk kell. Ha ugyanis egy felhasználónak különböző felhasználóazonosítója (UID) és csoportazonosítója (GID) van az ügyfélgépen, mint a kiszolgálón, előfordulhat, hogy nincs joga minden állományát elérni, ellenben más felhasználók fájljaihoz teljes jogokat szerezhet.
Elkerülhetjük ezt a gondot, ha fájljainkra állandó felhasználó- és csoportazonosító hozzárendelést állítunk be. A következő példában ezt a */etc/nfs/home_map.cats* fájlban állítjuk be:

```
map_static=/etc/nfs/home map.cats
```

A hozzárendelést szabályozó fájl, a ***home_map.cats*** így nézzen ki:

```
# SzabÆlyok a gØpekhez:
#     kiszolgÆl          gyfØl
uid    0-60           -     # nobody lesz
uid    1061-1080   1041     # hozzÆrendelj k
                            # a 61-80 a 41-60-hoz
```

Az ügyfélen és a kiszolgálón lévő démonok adatcseréjét a *2. ábra* mutatja.



2. ábra Démonok közötti adatátvitel

Ha hálózatunkon működik NIS-kiszolgáló, a rendszer képes a NIS-szolgáltatást is használni az UID-GID kezeléséhez. Ekkor mást kell beállítanunk: `map_nis=kingdom.cat`.
Ez jobb megoldás, mert az NFS-kiszolgáló a szükséges adatokat a NIS `kindgom.cat` körzetnevének segítségével szerzi meg.
A NIS használatakor csoportokat is megadhatunk. Ha például a fájlhoz hozzáadjuk a `@macskatulajdonosok` értéket, akkor a `macskatulajdonosok` csoportjába tartozó felhasználók neve hozzáadódik az engedélyezettek listájához.
Miután az összes fájlt megfelelően beállítottunk, indítsuk el a szükséges alkalmazásokat. Mivel az NFS kapu-hozzárendelőt használ, először ezt kell indítanunk. Az újabb Linux-kiadásokban a `portmap` vagy az `rcp.portmap` rendszerindításkor elindul, amit a következő parancssorral könnyen leellenőrizhetünk:

```
ps axu | egrep portmap
```

Ha a `portmap` fut, a szabványos kimenetre valami ilyesmit ír ki:

```
daemon 99     0.0 0.3 1132 500 ?      S Jul11
➥0:02 /sbin/portmap
tiger  27837 0.0 0.3 1264 492 pts/0 R 17:03
➥0:00 egrep portmap
```

Az első sor azt mutatja, hogy a `portmap` nevű alkalmazás fut. Ha a kapu-hozzárendelő nem fut, indítsuk el saját magunk. Legtöbbször a */sbin* könyvtárban található, de ha nem, nézzük meg a */usr/sbin* könyvtárban. Ha ott sem lenne, akkor a programot telepítenünk kell, amely a ***netbase*** vagy a ***nkitb*** nevű csomagban van.
Ha a kapu-hozzárendelő már fut, indítsuk el a szükséges démonokat, pontosan ebben a sorrendben: `rpc.mountd`,

`rpc.nfsd`, `rpc.statd`, `rpc.lockd` és `rpc.quotad`. Az újabb kiadásokban megfelelő héjprogramokkal indíthatjuk őket. Debian alatt például a `/etc/init.d/nfs-kernel start` parancsot használhatjuk. Ha ezek a csomagok véletlenül hiányoznának, a fenti kiadásokból kölcsönvett programokat módosíthatjuk, vagy akár magunk is megírhatjuk őket. Az `rpc.quotad` démon csak akkor kell, ha a felhasználók lemezterület-felhasználását ellenőrizni akarjuk. Újabb rendszermagok esetében – ha kell – az `rpc.nfsd` démon hívja az `rpc.lockd`-t, de ha külön elindítjuk, akkor sem lesz semmi gond.
Miután minden programot elindítottunk, hajtassuk végre a következő parancsot: `rpcinfo -p`. Az utasítás kiírja futó programjainkat a kiadásuk számával és egyéb hasznos adatokkal együtt. A kapu-hozzárendelő (portmapper), a `mountd` és az `nfsd` mindenképp fusson, hogy a hálózati fájlrendszer-kiszolgálót használni tudjuk. A Linux NFS HOGYAN-ja segítséget nyújthat, ha véletlenül elakadnánk.

Az NFS-ügyfél beállítása

Győződjünk meg róla, hogy van-e a rendszerünkben NFS-támogatás. Nézzünk bele a */proc* könyvtárba:

```
tomcat>    cat /proc/filesystems
           ext2
nodev      proc
nodev      devpts
           iso9660
nodev      nfs
```

Az itt szereplő fájlrendszertípusokat támogatja a rendszermagunkkal. Ha a `nodev nfs` hiányzik, az azt jelenti, hogy az NFS-fájlrendszert támogató modul hiányzik. Ez esetben rendszergazdaként a következő paranccsal próbálkozzunk: `modprobe nfs`. Ha modul megtalálható, a paranccsal telepítettük is, így ha a */proc/filesystems* fájlt újra kiíratjuk, a kívánt sor szerepelni fog. Ha rendszerünk az NFS-t nem támogatja, nincs mese, rendszermagot kell fordítani, de ennek ismertetése túlmutat a jelen írás keretein.
Ha létezik NFS-támogatás, akkor a hálózati fájlrendszereket a példában vázolt módon tudjuk befűzni. A parancs szerkezete a következő:

```
mount -t nfs kiszolgÆl :k nyvtÆr_elØrØsi_œtja
➥helyi k nyvtÆr kapcsol k
```

A `kiszolgÆl` helyére az NFS-kiszolgáló gép nevét írjuk, a `k nyvtÆr_elØrØsi_œtja` helyébe pedig a befűzni kívánt könyvtár teljes elérési útját.
Nagyjából ennyi az egész, de hogy hálózati fájlrendszerünk befűzése jobban működjön, nézzük meg, milyen lehetőségeket tudunk beállítani. Később biztosan hasznunkra válik:

- `rsize=n` és `wsize=n`: az olvasásra és írásra vonatkozóan beállítja az egyszerre elküldött és fogadott adatok mennyiségét – ebben a sorrendben. Az alapértelmezés rendszermagkiadástól függ, és általában 1024 bájt pozitív egész szám többszöröse. Minél nagyobb adatmennyiséget tud egyszerre feldolgozni, annál hamarabb végez, így jó, ha nagyobb értéket állítunk be. 4096-ra vagy 8192-re állítva sokat javulhat a szolgáltatás sebessége.
- `timeo=n` és `retrans=n`: az első érték azt mutatja, hogy az NFS-ügyfél hány tizedmásodpercenként próbálkozzon újra csatlakozni a kiszolgálóra, ha a hálózat vagy a kiszolgáló épp nem érhető el. Az alapértelmezés 7. Ha a próbálkozási idő elérte a 60 másodpercet, megkétszerezi ezt a számot és újraindítja a kérelmet. A második szám azt mutatja, hogy ez utóbbit, mármint a megkétszerező-újraindító folyamatot az NFS-ügyfél hányszor tegye meg. Alapértelmezés szerint ez az érték 3.
- `hard` és `soft`: beállítja, hogy mit tegyen az NFS-ügyfél, ha nem éri el az NFS-kiszolgálót. Ha a `hard`-ot választjuk, a következő válasz mellett tovább próbálkozik: `server not responding, still trying`. Így ha a kiszolgáló újra elérhetővé válna, a folyamat onnan folytatódik, ahol abbamaradt. Ha viszont a `soft`-ot választjuk, akkor az NFS-ügyfél be- és kimeneti hibát jelez, ami hibás fájlokat eredményezhet. Vigyázzunk tehát ezzel a lehetőséggel.
- `intr`: megszakíthatjuk az NFS-hívást, ami akkor hasznos, ha a kiszolgáló sokáig nem érhető el.

Az `intr` és `hard` az ajánlott beállítás, és mivel a `hard` az alapértelmezett, elég az `intr`-t megadni. Ezekkel az értékekkel valószínűleg javítottunk az NFS-kapcsolat teljesítményén.
Hálózati fájlrendszert rendszerindításkor is befűzhetünk, ha a */etc/fstab* fájlt módosítjuk. Ennek minden sora legalább négy, de általában hat mezőből áll:

1. eszköznév,
2. befűzési pont,
3. a fájlrendszer típusa,
4. kapcsolók,
5. dump,
6. az ellenőrzés sorrendje.

```
tiger:/nfs1/home /home nfs rw,hard,intr 0 0
```

Az első érték a befűzendő eszközt azonosítja. Példánkban a tiger nevű NFS-kiszolgáló */nfs1/home* könyvtára lesz a */home* könyvtár. A negyedik mezőben a fájlrendszer típusát adjuk meg, ami jelen esetben `nfs`. Az ötödik mezőt a régi dump adatmentő rendszer számára tartják fenn, a hatodik mező pedig megadja, hogy az eszköz a rendszer főkönyvtára (1), egyéb ellenőrziendő könyvtár (2), vagy nem ellenőrizendő (0). Indítsuk el még az `rpc.lockd` és az `rpc.statd` démonokat, amit héjprogramok segítségével tehetünk meg. Ezennel beállítottunk egy működő NFS-ügyfelet.

Hálózati fájlrendszerünk tervezése

A kérdés az, hogy miféle adatokat tehetünk a hálózati fájlrendszerre? A válasz egyszerű: mindenfélét, így hálózatunkon lemeznélküli gépek is működhetnek. A rendszer tervezésénél figyelembe kell vennünk, hogy kétféle fájlról beszélhetünk: léteznek megosztható és nem megosztható adatok. A második típusba tartozó adatokat a saját gépünkön kell tartani – például az eszközfájlokat nem lehet megosztani.
Kisebb rendszer esetében a */home* könyvtárat tehetjük a kiszolgálógépre. Közepes és nagy rendszerek építésekor a */home* könyvtárat feloszthatjuk. A felhasználói könyvtárak mérete nem állandó, de ezt tárkorlátrendszerrel szabályozni tudjuk.
Mi a helyzet más könyvtárak esetében? Néhányuk állandó méretű, némelyik folyamatosan változik. A megosztható állandó méretű könyvtárak közül a */usr* és a */opt* említhető meg, de az állandó méretűek közül például nem osztható meg például a */etc* és a */boot*. A folyamatosan változó könyvtárak közül megosztható a */var/mail* és a */var/spool/news*, de nem lehet megosztani a */var/run* és a */var/lock* könyvtárakat.
Sokan a */usr/bin* és a */bin* könyvtárakat nem teszik hálózati fájlrendszerre, mert ekkor lassabban lehet elérni őket. Úgy gon-

doljuk, egy átlagos felhasználó nem veszi észre a különbséget egy hálózati fájlrendszerről hívott program és egy helyi lemezről indított alkalmazás között. Így tehát majdnem minden fájl feltehető az NFS-kiszolgáló lemezeire, hogy hely jusson MP3-fájljaink vagy akár egy másik operációs rendszer számára. Igaz, akármit tehetünk a hálózati fájlrendszerre, a szokás mégis az, hogy az NFS-kiszolgálón a felhasználói könyvtárakat, a leírásokat és a súgórendszert tartjuk. A ritkábban használatos programokat is nyugodtan a kiszolgálóra tehetjük, így például a */opt* és a */usr/local* könyvtárak tartalmát. A */usr/doc* és a */usr/share* könyvtárakat úgyszintén megoszthatjuk.
Miféle kapcsolókat kell használnunk a `mount`-tal ezekben az esetekben? Mint említettük, a felhasználóikönyvtárakra az `intr` és a `hard` beállítás ajánlott. Így biztosak lehetünk benne, hogy nem vész el adat. A */usr/doc* könyvtárak esetében viszont nem biztos, hogy ez a megfelelő beállítás. Ha például a súgórendszert próbáljuk elérni, de a kiszolgáló nem működik, feleslegesen terheljük a hálózatot. Ez esetben jobb a `soft` beállítás, miáltal a folyamat a hibaüzenet megjelenésével megszakad. Hasonlóan járjunk el a többi, nem túl fontos fájl esetében is. Leveleinket inkább más szolgáltatásokkal érjük el, használjunk IMAP-ot, POP-ot vagy NNTP-t.

Az NFS-kapcsolat sebességének javítása

Állítsuk be a legmegfelelőbb `rsize` és `wsize` értékeket. Az alapértelmezett beállítás nem biztos, hogy jó, előfordulhat ugyanis, hogy hálózati kártyánk a nagy adattömböket nem tudja kezelni. Régi alkatrészekkel vagy rendszermagokkal ez gyakran előfordul. Ha újabb kártyánk van, nagyobb érték beállításával próbálkozzunk. Esetenként a kapcsolat sebességét négy-ötszörösére növelhetjük, ezért jó, ha kísérletezünk.
A sebesség mérésének legegyszerűbb módja, ha létrehozunk egy fájlt a kiszolgáló lemezére és megmérjük a létrehozás idejét. Bármilyen adatot tartalmazhat a fájl, csak a kísérlet kedvéért hozzuk létre, így akár csupa nulla is lehet. Hozzuk létre a `dd` parancs segítségével, és a bemeneti fájl a */dev/zero* legyen. A parancs az időt is méri.
Mekkora legyen az ideiglenes fájlunk? Legyen kétszer vagy háromszor akkora, mint a kiszolgálógép memóriája. Ha a gépben 2 MB RAM van, 4 MB legyen. Előfordulhat, hogy – ha túl sok memória van a kiszolgálógépben – nem tudunk ekkora fájlt létrehozni, de 256 MB legtöbbször elég. A `df` paranccsal megnézhetjük, hogy mennyi szabad hely van a lemezen.
Az ügyfélgépen dolgozva a hálózati fájlrendszert újra az 1024-es `rsize` és `wsize` méretekkel fűzzük be. Futtassuk le a következő parancsot:

```
time dd if=/dev/zero of=/home/tempo/verbatim
➥bs=16k count=16384
```

A */dev/zero* fájlt használjuk bemenetként, így olvasása nem rontja el a kapott adatot. Parancsunk 16 384 darab 16 KB-os nulla bájttömböt küld, így 16 000×16 384/1024 kilobájt jön létre, ami 256 MB-nak felel meg. Jegyezzük meg az így kapott időt. Majd mérjük meg az olvasás és a */dev/null*-ra küldés idejét is:

```
time dd if=/home/tempo of=/dev/null bs=16k
```

Jegyezzük meg ezt az időt is, majd az ideiglenes fájlt letörölhetjük:

```
rm /home/tempo/ verbatim
```

Ezt ismételjük meg háromszor, majd számoljuk ki az átlagos olvasási idő/írási idő hányadost. Válasszuk le az NFS-fájlrendszert:

```
mount /home
```

Ezután fűzzük be újra, most 2048-as `rsize` és `wsize` értékekkel. Majd ismételjük meg a fenti eljárást egészen 32 768-as `rsize` és `wsize` értékekig.
Lássuk az eredményt! Minél gyorsabb az adatátvitel, annál kevesebb a mért idő. A legmegfelelőbb értékeket állítsuk be a */etc/fstab* fájlban, majd számításainkat tegyük biztonságos helyre.
Rövidebbé tehetjük az eljárást, ha újabb hálózati kártyánk és hármas kiadású NFS-rendszerünk van. Ez esetben induljunk 32 768-ról, és csökkentsük a számot. A kettes kiadással ne próbálkozzunk, ez ugyanis csak a régebbi rendszermagokkal, Solarisszal vagy SunOS-sel működik.
Úgy találtuk, hogy 2.2.19-es rendszermaggal a megfelelő érték 4096, ezért próbáljuk ki a 2048-as és 8192-es értékeket. Az újabb, 2.4.6-os rendszermag esetében ez az érték 8192 volt, de azt tapasztaltuk, hogy a régebbi rendszermag sokszor jobban teljesít. Igaz ez nagyban attól függ, hogy milyen alkatrészeket használunk. Mindenképpen vigyázzunk, ha NFS-kiszolgáló gépünkre új rendszermagot fordítunk.
TCP/IP segítségével kapcsolódó ügyfelek esetén a legjobb az 1024, vagy az annál is kisebb érték. Akkor is működik, ha az ügyfél és a kiszolgáló között útválasztó van.

Az NFS-kiadásokról néhány szóban

Miközben a `mount` végzi feladatát, az NFS ellenőrzi a protokollváltozatokat. Néha a következő hibaüzenet jelenik meg: „a kiszolgáló NFS kiadása régebbi, mint az ügyfélgépen lévő rendszermag által támogatott”.
A hálózati fájlmegosztás gondolata olyan régi, mint maga a hálózat, így több megoldás is kínálkozik. Az NFS teljesítményének javításához váltsunk a 2000-ben bemutatott 4-es kiadásra, vagy röviden NFSv4-re. Az NFS feltalálója, a Sun Microsystems támogatja a linuxos NFSv4 projektet. Megtartották az eddigi kiadások összes lehetőségét, de a biztonságot a megbízható RPC-folyamatok bevezetésével fokozták. A régebbi kiadások másik gyenge pontja az volt, hogy nem támogatták a soknyelvűséget. Az új kiadásban ezt is megoldották. Gyorstárak alkalmazásával a fájlműveletek sebességét is növelni tudták. A megfelelő sebesség eléréséért folytatott kísérletezést is elfelejthetjük, ugyanis az új kiadás magától megteszi. Nem bánjuk meg, ha erre állunk át, bár a linuxos NFSv3 néhány NFSv4-es lehetőséget is támogat.

Linux Journal 2002. január, 93. szám

Olexij Tykhomyrov
(tiger@ff.dsu.dp.ua) 1994 óta Linuxot használ. A Dnepropetrovski Nemzeti Egyetem Kísérleti Fizikai Tanszékén dolgozik, ahol fizikát és kommunikációt tanít. Rajong a fiáért, Misáért, aki Tigrisnek szólítja, mert a diákok állítólag félnek tőle. Tigris szeret úszni és utazni.

Denis Tonkonog,
Tigris volt tanítványa ugyanott dolgozik, és szintén kedveli az utazást. Hobbija a puskával halászás. Barátai Fekete Macskának szólítják, nem tudni, miért. E-mailt a következő címen vár: denis@ff.dsu.dp.ua.

Ablakkezelők (4. rész)

Az AfterStep és a WindowMaker.

Az ablakkezelőket bemutató sorozatunk utolsó részében két, egymás utódjának tekinthető ablakkezelő bemutatására kerítek sort. A két ablakkezelőt meglehetősen sokat használtam, de az utóbbi egy-másfél évben az AfterSteppel sajnos megbízhatósági gondjaim támadtak, és azóta emiatt szinte csak a WindowMakert használom. Mindkét ablakkezelőhöz fordítgattam annak idején néhány dolgot, ezek szolgálják a mostani leírás alapját is.

Az AfterStep

Az AfterStep NextStep-felületen alapuló eszköz, amely a felhasználók igényeinek megfelelően változatos munkakörnyezetet képes létrehozni. Néhányak szerint nemcsak egyszerű ablakkezelő, hanem teljes eszköztárral felszerelt önálló grafikus felület. Az AfterStep a hasonló tudású ablakkezelőkhöz képest jellemzően igen kis memóriaigénnyel bír. Természetesen a felület által megkövetelt memória nagy mértékben az általa kezelt (háttér)képek méretétől függ.
Az AfterStep a régebbi, ez az ablakkezelő szolgált a későbbiekben bemutatásra kerülő WindowMaker alapjául. Ha már az ablakkezelő történetéről ejtünk néhány szót, érdekes lehet, hogy ennek az ablakkezelőnek az alapja is az fvwm volt, amit ***Robert Nation*** írt (az fvwm pedig a twm-en alapult).
Az AfterStephez vezető változások a Bowman-projektből indultak. A későbbiek folyamán számos fejlesztő ***Alfredo Kojima*** (kojima@WindowMaker.org) vezetésével a WindowMaker-projektbe igazolt át.

Melyek az AfterStep saját elemei?

- A NextStephez hasonló fejléc, fejlécgombok, keretek és sarkok.
- Az AfterStep Wharf, ami a GoodStuff-változattól kezdve létezik. A szerzői jogi bonyodalmak elkerülése végett nem hívják „dock”-nak.
- A NextStep-stílusú menük.
- A lenyíló menü a fő ablakban.
- NextStep-megjelenésű ikonok. Az alapikonok azonosak a NextStep felületén használtakkal, bár mások is beállíthatók.
- A Lapozó (Pager) képernyő-megjelenítéssel.
- Jól és könnyen használható fájlok, amelyek a felület barátságossá tételére szolgálnak.
- A ***Start*** menü bővíthető almenükkel.
- Ablaklista, feladatlista, ami vízszintesen és függőlegesen is elhelyezkedhet.
- Számos saját modult és alkalmazást tartalmaz.
- Az fvwm-1.x moduljai is jól használhatók.
- Ikonalkalmazások (appsicon).

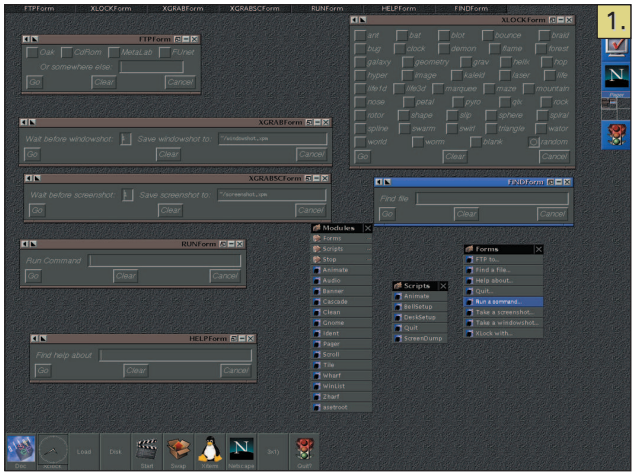
Az AfterStepet több helyről is beszerezhetjük: a hivatalos oldal a ➲ http://www.AfterStep.org címen található. A következő FTP-helyről szintén elérhető: ➲ ftp://ftp.AfterStep.org/pub/.
Az alkalmazások feltöltésére a következő cím szolgál: ➲ ftp://ftp.AfterStep.org/incoming/.
A mai ablakkezelők már általában egyetlen közös menüt használnak, illetve vesznek át a rendszertől. A régebbi ablakkezelőkben ez nem így volt, illetve még ma sem így van, ha a rendszermenü helyett más menüt szeretnénk használni.
A menüt több könyvtár tartalmazza, így például:

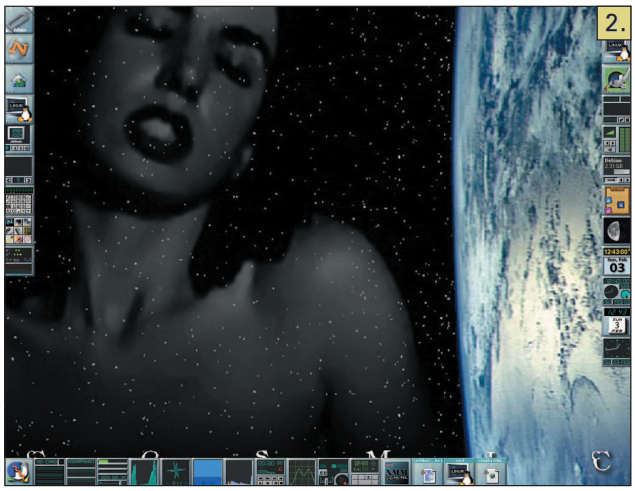
- ***/etc/X11/AfterStep/hookdir/Debian*** – gondolom, más rendszereken a rendszermenüt tartalmazó könyvtárak valahol ehhez a könyvtárhoz igen közel, egy hasonló könyvtárszerkezetben helyezkednek el.
- ***~/GnuStep/ Library/AfterStep/start*** – ez a könyvtár a saját beállításaidat tartalmazza, tartalma csak az adott felhasználó számára érhető el.
- ***/usr/share/AfterStep/start*** – ez a könyvtár az AfterStep saját menüjének beállításait tartalmazza.

Az egyes menüpontok egy-egy különálló fájlt jelentenek, amelyek az egyes alkalmazásokat mintegy parancsfájl segítségével indítják el. E fájl neve általában azonos a menüponttéval, de arra is lehetőség nyílik, hogy a menüpont nevét a fájlon belül adjuk meg. A menü könyvtárszerkezete megegyezik az indítófájlokat tartalmazó könyvtárak felépítésével.
Az alábbi példa két indítófájl tartalmát mutatja: az első a ***b_AfterStepdoc*** fájlt szemlélteti, ami az AfterStep saját leírásához a rendszer alapértelmezett böngészőprogramját (általában a Netscape-t) nyitja meg.

```
bash-2.05a$ cat b_afterstepdoc
Exec "AfterStep documentation"
➥exec afterstepdoc &
# Futtasd ezen a nØven ezt a programot.
MiniPixmap "mini-as.xpm"
# Ikonja legyen ez a kØp.
A mÆsodik pØlda egy elvÆlaszt  sort
➥( res sort) illeszt be a men be:
bash-2.05a$ cat a_nop
Nop ""
```

Az AfterStep egyik leginkább a sajátjának tekinthető eszköze a Wharf. Feladata az alkalmazások ikonokról történő indítása. Az ikonok könyvtárszerűen csatolhatók egymásba: ha például egy kezdő ikonsorral rendelkezünk, amelyben az ***Office*** fül is megtalálható, amikor rákattintunk, mire merőlegesen egy másik ikonsor nyílik ki, amiből akár az OpenOffice-t, akár az Abiwordöt is elindíthatjuk. A Wharfot már nagyon régen használtam, de annak idején minden számomra szükséges alkalmazás indítását a Wharfból végeztem. A Zharf nevű hasonló szerepű alkalmazást nem használtam, mert a Wharf alkalmazása elégségesnek bizonyult. A beállításokat tartalmazó fájlok felépítése, illetve megjelenése hasonló, sőt a szerepük is azonos. A Zharf amolyan második Wharfnak tekinthető.
A felület további beállításait a ***feel*** és ***look*** fájlok tartalmazzák – ezeknek az AfterStep saját könyvtárában saját könyvtáruk van. E fájlok viszonylag könnyen átszerkeszthetők, hiszen eléggé beszédesek, általában tartalmazzák a beállítási lehetőségeket, így nekünk csak a sorkiemelést kell módosítanunk.
Az AfterStep telepítése után a telepítéskori eredeti ***feel*** és ***look*** fájlok közül válasszuk ki leginkább megfelelőt, és azt szerkesszük át testhezálló formátumra.
Az AfterStep néhány saját parancsfájlt és űrlapot (form) is





tartalmaz. A parancsfájlokkal kezdeném: könnyen magyaríthatók, de a magyarított parancsfájlokat csak azonos változatú AfterStepek között lehet hibátlanul cserélgetni! Jó hír azonban, hogy a title szóra rákeresve a parancsfájlokban, illetve az adott parancsfájlt elindítva hamar rájöhetünk, hogy melyik angol szöveg, illetve szó helyére milyen magyar szöveg illik – és ez nyugodt szívvel arra változtatható meg, amire csak akarjuk! Például:

```
Title {Ok} ==> Title {Rendben}
```

A szöveg hosszára kell csak figyelned, nehogy a beírt szöveg vége takarásba kerüljön.
Az űrlapok az AfterStep főkönyvtárában a ***forms*** fájlban találhatók. Annak idején én is készítettem kettőt: egy tárcsázót, illetve egy rendszergazdaablak-megnyitót. Szerintem, ha ebbe a fájlba belepillantunk, mindenféle magyarázat nélkül viszonylag egyszerűen tudunk majd új formokat készíteni.
Nem igazán tartozik az ablakkezelők leírásához, de az AfterStep-leírás készítésekor gondom akadt a szükséges telepítőprogramok megtalálásával, mert a hibaüzenetek csak a szükséges, de hiányzó fájl nevét mutatták meg, azt azonban nem, hogy a helyes működéshez milyen csomag is szükséges. Némi segítség ehhez:

```
rpm -qpil *.rpm
➥>/tmp/amelyik_fÆjlba_akarod_tenni.txt
```

Ez egy fájlba gyűjti például a Red Hat rpm-jeinek leírását, ami egy szerkesztőprogrammal átnézhető.
Debian esetében egyszerűbb a dolgunk:

```
apt-cache search amit_keres nk
```

Az ablakkezelőhöz tartozik még a Winlist, amely egy tálcaszerű eszköz az ablak tetején. Az AfterStep Lapozója is megkülönböztet munkafelületeket, illetve munkaterületeket. A Munkaterületek közötti váltáshoz elegendő az egérmutatót az asztal megfelelő szélére húznunk, és a váltás azonnal megtörténik.
Az Ikon-alkalmazások (appsicon) olyan kisméretű alkalmazások, amelyeket ikonméretűre terveztek és az ikonsorokban futtathatók. Az AfterStepnek is vannak saját alkalmazásai, de a WindowMaker jóval több ilyen alkalmazással rendelkezik, ezért ezekkel a maguk helyén foglalkozom részletesen.
Az AfterStep is rendelkezik grafikus beállítóeszközzel, az Ascp-vel. Az Ascp – amit akár ***AfterStep Control Panel***-nek is nevezhetnénk – olyan program, amely a NextStep beállítópanelét emulálva segíti az AfterStep beállítását. Ez az AfterStep hagyományosan kézzel történő beállításainak elvégzését a beállítófájlokban valóban nagyon könnyűvé teszi. Az Ascp a 0.9-es változatig Tcl/Tk-t és TkStepet használ, ami a ***steprc*** fájlt ***.only***-ra írja át. A 0.9-es változattól kezdődően az Ascp felhagyott a Tcl-alapú szemlélettel. Az új Ascp teljes mértékben a GTK+-on alapul. A későbbi változatok közül némelyek az 1.1.1-es GTK+-n, és az 1.1.2-es változatú glib-en alapulnak. Figyelmeztetés: az AfterStep a ***.steprc***-t az 1.2 változat óta nem használja, tehát ha a régebbi Tcl-alapú Ascpt használjuk, csak AfterStep-megfelelő módban szabad futtatni:

```
AfterStep -f {yeoldsteprc}
```

Ez az eszköz a mostani Linux-terjesztésekbe valamiért sajnos nem kerül be.

WindowMaker

A WindowMaker is igen elegáns ablakkezelő, és ugyancsak hasonlít az OpenStep-felülethez.
A fejlesztői változat a hivatalos oldalról (➲ ftp://ftp.WindowMaker.org/pub/beta/srcs/) tölthető le.
A hazai hivatalos oldal a ➲ http://www.extra.hu/WindowMaker címen érhető el.
Ajánlott még a ➲ http://wm.current.nu/downloads.html oldal megtekintése is, amelyen az utolsó megbízható és a fejlesztői változat egyaránt fellelhető. A fejlesztői változatok általában önmagukban is igen megbízhatók, de ha a legújabb változatot szeretnéd, az is nagy biztonsággal használható.
A WindowMaker nagysága az úgynevezett Appicon-alkalmazásokban rejlik. Ezek – mint említettem – olyan ikonok (az apró fejléc nélküli és lekicsinyített alkalmazások), amelyeket a legtöbb alkalmazás indítása után az ikondobozban helyez el – egészen az alkalmazás futtatásáig. Ha például az Xtermet futtatjuk, egy sarokbeli ikon figyelmeztet rá. Ezeket az ikonokat az úgynevezett Dockba vagy a Clipbe húzhatjuk át. Hogy miképpen tehetünk új alkalmazást a Dockba? (Ez az az ikon vagy ikonsor, ami alapértelmezésben a képernyő jobb oldalán található, lásd *2. kép*).
Indítsuk el az alkalmazást! Ha ilyenkor a képernyő alsó sarkában egy kicsi ikon található, át kell húznunk a dokkolt ikon(ok)ra. Egy felvillanó fehéralapú négyzet válik láthatóvá, ami azt jelzi, hogy közel van a dokkoláshoz, és az ikon várható helyét mutatja. Úgy győződhetünk meg róla, hogy a legkö-

zelebbi indításkor az alkalmazás itt jelenik majd meg, hogy az ***Exit session*** lehetőséggel kilépünk a WindowMakerből.
A Clip esetén ugyanígy helyezhetünk el dokkolt alkalmazásikont. Fontos megemlítenem, hogyha a Clip és a Dock közel egymáshoz van, alkalmazásaid dokkolásánál gondjaid támadhatnak. Ezért a Clipet először messzire el kell vinni a Docktól, amit a kezdőikonra történő kattintással és annak mozgatásával tudunk elérni. Amennyiben dokkoltunk egy alkalmazást, a jobb gombbal rákattintva egyéb lehetőségeket is be tudunk állítani.
Az itt elhelyezett ikonok kétfélék lehetnek:

- nem WindowMaker-alkalmazás, amelynek csak az indítóikonja helyezhető el az ikonsorokban.
- WindowMaker-alkalmazás, amely eleve ikonméretűre lett készítve.

Következzék néhány ilyen alkalmazás rövid leírással (2. kép)!

- Clipben a képernyő bal oldalán, fentről lefelé:
  Clip beállítóikon, Netscape-indító, OpenOffice-indító, Kylix-indító.
  – wmxres: az X-asztal felbontását lehet vele váltani, hasonló a CTRL+ALT+PLUSZ MINUSZ (+ -) gombok működéséhez.
  – wmcb: egyfajta vágólap, ahova szövegeket illeszthetünk be és vághatunk ki.
  – Wmcalc: számológép.
  – Wmbutton: a rajta lévő alkalmazásikonok elindítják a csatolt alkalmazásokat.
  – WmSpaceWeather: az „űridőjárás”-mutató. Pontosan nem tudom, milyen adatokat és mi alapján mutat.
- Ikonbox az asztal alján, balról jobbra:
  Valamely alkalmazás segédikonja.
  – wmwave: a kiválasztott ethernetkártya statisztikáját mutatja.
  – Wmtop: a top legaktívabb három alkalmazását mutatja be működésük sorrendjében.
  – Wmsysmon: rendszermonitor.
  – Wmload: a rendszerterhelést mutatja, hasonló az ***xload***-hoz.
  – Wmcube: pontosan nem tudom, mi a rendeltetése, különböző térbeli alakzatokat forgat.
  – Wmbubble: első ránézésre egy kacsa úszkál jobbra-balra az ikonon, ám ez valójában egyfajta rendszermonitor. Az adatok csak akkor válnak láthatóvá, ha az egérmutatót fölé húzzuk, illetve két másik alkalmazás indítására is használható.
  – Wmavgload: egy újabb rendszermonitor.
  – Wmusic: az XMMS irányítására szolgáló alkalmazás.
  – Wmscope: a hangkimenet jeleit mutatja (nekem sajnos nem sikerült megfelelően beállítanom). Alapesetben egy változó szinuszgörbéhez hasonló ábrát mutat.
  – Wmix: audiokeverőpult.
  – Wmcdplay: CD-lejátszó.
  – Wmxmms: ez az eszköz is az XMMS indítására és kezelésére szolgál.
- Dock az asztal jobb oldalán, fentről lefelé:
  – wmdock: ehhez csatolhatók az alkalmazások.
  Xterm: alapesetben az Xtermet indítja, de nálam az rxvt-t:
  `rxvt -g 80x40 -fg green -bg black -fn 8x16`
  – wprefs: a WindowMaker egyik grafikus beállítóeszközét indítja.
  – Wmppp: a PPP-kapcsolat indítására és felügyeletére szolgáló eszköz.
  – Wmmixer: hangkeverőpult több csatornával.
  – Wmmount: a meghajtók csatolására, leválasztására szolgál, a beállításai a ***/etc/system.wmmount*** fájlban találhatóak.
  – Wmpinboard: jegyzetfüzet, időzített üzenet megjelenítési lehetőséggel.

Az alkalmazások dokkolását is eléggé egyszerű megszüntetni. Ha az adott alkalmazás nem fut, a dokkolt ikont csak le kell húzni az asztal közepére (távol a Docktól és Cliptől), és el fog tűnni. Ha a program fut, akkor le kell nyomni a Meta (váltó) (általában a CTRL) billentyűt, és az ikont ki kell húzni a Dockból. A másik lehetőség, hogy a jobb gombbal rákattintunk az ikonra, és a megjelenő menüből kiválasztjuk a ***Kill***-t, majd az ikont a képernyő közepére húzva elengedjük – ilyenkor a dokkolás általában megszűnik.
A WindowMaker helyi beállításai a ***~/GNUstep*** könyvtárban találhatók. E fájlok és feladatuk röviden:
~/GNUstep/WindowMaker/WindowMaker: a fő-fő beállítófájl. Ez a fájl állítja be a billentyűzetkapcsolókat, a betűkészleteket, a képeket és a fókuszmódokat.
~/GNUstep/WindowMaker/WMWindowAttributes: ez a fájl az alkalmazásokhoz tartozó vezérlő-„tulajdonságokat” tartalmazza. Olyan lehetőségek találhatók itt, mint például hogy milyen ikonokat használjon. Jelen esetben ehhez a legjobb hozzáférést – a fejlécre történő jobb gombos kattintásra megjelenő menüben – az ***Attributes*** lehetőség kiválasztása adja.
~/GNUstep/Defaults/WMState: ez a fájl a mindenkori dokkolt alkalmazásokhoz önműködően jön létre és a beállításaikat tartalmazza. Kézzel történő szerkesztése nem ajánlott!
~/GNUstep/Defaults/WMRootMenu: ez az a fájl adja meg, hogy milyen fájlt használjunk főmenüként. A Window Maker 0.19.0-tól lecserélhető a ***~/GNUstep/Defaults/WindowMaker*** könyvtárban lévő ***plmenu***-nek hívott fájlokkal, ekkor a menü szerkesztésére a ***WPrefs***-t használhatjuk.
~/GNUstep/Library/WindowMaker/menu: ez a fájl alkalmazható a főmenü szerkesztésére.
~/GNUstep/Library/WindowMaker/plmenu: ugyanaz a menüfájl ***property*** listaformátumban – a ***WPrefs***-nek szüksége van rá ahhoz, hogy a menüt szerkeszteni lehessen vele.
Az ablakkezelő nem rendelkezik lapozóeszközzel, mivel a képernyők közötti váltást a gyorsbillentyűk, illetve a Clip szolgálja. A rendszermenü az asztalra kattintva érhető el. A beállításoktól függően a futó alkalmazások listája is az asztalra kattintva érhető el. A WindowMaker rengeteg saját témával rendelkezik, amelyek beállítása az alkalmazásindító menüből és a grafikus beállítóeszközökből is elérhető.
A WindowMaker két grafikus beállítóeszközzel rendelkezik: a Wprefs alapból telepítésre kerül, bár szerintem ez a kevesebb beállítást engedélyező eszköz. A másik eszköz a ***Wmakerconf***. Szerintem jobban használható, bár mindenkinek mások az elképzelései. Én azért szeretem jobban, mert például témákat úgy lehet telepíteni és kiválasztani, hogy a későbbi felület megjelenése látható.
A WindowMaker saját hangkiszolgálóval is rendelkezik, amit a WindowMaker indítása előtt kell elindítanunk.
Helyszűke miatt most sajnos csak ennyi fért bele a leírásokba. Remélem, a különböző ablakkezelők bemutatása felkeltette érdeklődéseteket az ilyen grafikus felületek iránt.

Kapcsolódó anyagok találhatóak a 33 CD. Magazin/Ablak könyvtárában.

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.

PHP-vel biztonságosan (2. rész)

A figyelmes programozás szempontjain túl szó esik a PHP témába vágó beállítási lehetőségeiről és a PHP-fejlesztések ez irányú jövőjéről.

A múlt havi számban megjelent cikk folytatásaként nagyrészt a PHP körüli rendszeradminisztrációs feladatokról fogunk szólni. Vizsgálódásunk tárgya még a biztonsági beállítások mellett a PHP ezen területen várható jövője (részben jelene) is.

A PHP védelmi lehetőségei

A ***php.ini***, a PHP beállításait tartalmazó fájl számos olyan lehetőséget tartalmaz, amelyek használatával PHP-alkalmazásainkat jóval biztonságosabb környezetben tudhatjuk. Ez kiterjed az adatok védelmére, a hibajelzések megfelelő kezelésére. Akár egyes PHP-elemek, -függvénynek letiltása is lehetséges. Érdemes mindjárt azzal kezdenünk, hogy a szószátyár hibakezelőt kicsit gatyába rázzuk.

A hibakezelés beállítása

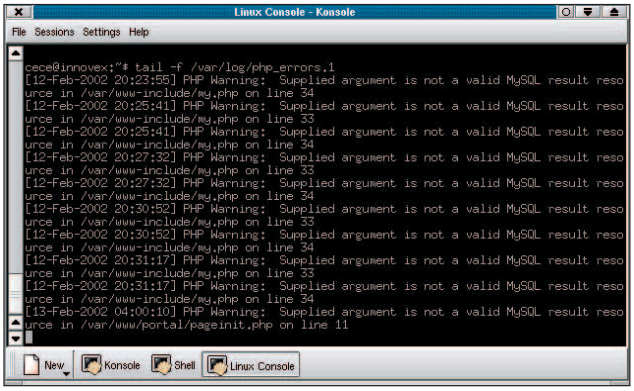
Célszerű mindjárt a PHP alapértelmezett hibakezelési beállításait megváltoztatni. Az `error_reporting` eredeti értéke (`E_ALL & ~E_NOTICE`) tökéletesen megfelelő hibakezelési szint, minden rendben is van vele. A `display_messages` kapcsoló bekapcsolt értéke viszont egy éles kiszolgálón megengedhetetlen. Ezt a kapcsolót `On` vagy `Off` állapotba állításával vezérelhetjük, mégpedig aszerint, hogy a PHP-hibaüzenetek a weblapba ágyazva ott jelenjenek-e meg, ahol a hiba keletkezett. Valójában a PHP hibaüzeneteihez egyedül az üzemeltetőnek (vagy a fejlesztőnek) van köze. Programfejlesztés, hibakeresés során esetleg kényelmesebb lehet bekapcsolva tartani, így a hibamentesítés gyorsabban folyhat. A nagyközönség elé tárt kiszolgálón viszont megéri a következő beállításokat végrehajtani:

```
display_errors = Off
log_errors = On
error_log = <a fÆjl, ahovÆ a hibÆkat  rhatja >
```

E beállítások hatására két dolog történik:

- A hibás működést okozó látogató a hibaüzenetet nem látja. Valaki hibát hozhatott létre véletlen módon, programhibába ütközve, de lehetséges, hogy ez a valaki rendszerünk gyenge pontját puhatolva kényszeríti programunkat hibás működésre.
- A hibák naplózása révén mi viszont minden hibát láthatunk, nem csak a magunk által létrehozottakat. Így a menet közbeni felmerülő működési rendellenességekről is első kézből kaphatunk adatokat.

A `display_errors` kapcsoló mellett a `log_errors` kapcsoló `On` állapota gondoskodik arról, hogy minden egyes hiabüzenet naplófájlba kerüljön. Ez a naplófájl az ***php.ini*** `error_log` beállításával adható meg. Értéke vagy egy fájl teljes elérési útvonalával, vagy pedig a `syslog` szó lehet, ekkor a rendszernaplózó feladata lesz a hibaüzenet naplóbejegyzéseinek lekezelése. Több tartományt kiszolgáló rendszereken a naplózófájlt tartományonként külön-külön is beállíthatjuk, kihasználva, hogy ezt az Apache beállításaiban is módosíthatjuk:




```
<VirtualHost azegyikdomenem.lx>
ServerName azegyikdomenem.lx
ServerAdmin cece@balaton.hu
DocumentRoot /var/www/azegyikdomenem.lx/
php_admin_value error_log
/var/log/php/azegyikdomenem.lx-error.log
... egyØb beÆll tÆsaink ...
</VirtualHost>
```

E kis átállítgatások következtében PHP-s rendszerünk egy csapásra felügyelhetővé válik. Az sem megvetendő előny, hogy innentől nem vagyunk arra utalva, hogy a hibát egyáltalán bejelenti-e valaki, és ha igen, milyen minőségben. Lássuk, mit lehet még tenni!

Beillesztendő PHP-kódok védelme

Az előző cikkben volt pár példa arra nézvést, hogy mit tehetünk PHP-programunk csak `include()`-ra szánt részeinek védelme érdekében. Mindazok a pótmegoldások, amelyekről ott értekeztem, csak akkor jöhetnek szóba, ha a kiszolgáló nem a mi kezünkben van, és nincs e fájlok védelmére beállítva. Valós védekezésképpen ezeket az állományokat a nyilvánosság elé tárt könyvtáron kívül, külön kell tárolni.
A függvényeket és egyéb programrészleteket tartalmazó fájlok könnyebb elérése érdekében hasznos beállítani, hol keresse őket a PHP. Lehetne ugyan teljes elérési útvonalak megadásával is ügyködni, de ez jócskán visszadobná a PHP-ben írt rendszer hordozhatóságát. A ***php.ini***-ben az `include_path` segítségével adható meg egy vagy akár több könyvtár is, ahol a beillesztendő fájlokat sorban keresni fogja. Az egyes lehetséges könyvtármegjelöléseket kettősponttal kell elválasztani, valahogy így:

```
include_path .:/usr/local/php/include/
```

No és természetesen igény szerint akár tartományonként is megadhatjuk ezeket az ***httpd.conf***-ban – az `error_log` példájára.

magic_quotes-huncutságok

A PHP jelenlegi változata több helyen nyögi még a PHP3 hagyatékait. Támogatottságukat ugyanis nem lehet egyik napról a másikra eltüntetni, mivel rengeteg PHP-ben írt oldal mondaná fel a szolgálatot egy rendszerfrissítés után. Nem pont a legfájdalmasabb, de az egyik ilyen maradvány a `magic_quotes_gpc` bekapcsolt állapota a PHP hivatalos alapbeállításaiban. Hatására a PHP a bejövő adatokon önműködően végrehajt bizonyos átalakításokat. Minden egyszeres és kétszeres idézőjelet fordított perjellel (backslash) véd, természetesen magát ezt a jelet is. Ezzel a lépéssel, vagyis hogy minden érkező adatot így alakít, sok kellemetlen beavatkozástól védheti meg az amúgy óvatlan programfejlesztőt. Csakhogy e szolgáltatás miatt sajnos a PHP-programozók java a tapasztalat szerint valóban gyanútlanná vált.
A teljesen kezeletlen és ellenőrizetlen beérkező adatok egy SQL-lekérésbe vagy -parancsba kerülve nagy veszélyforrást jelentenek, hisz a következő kis SQL-adatmódosító parancs könnyedén támadófelület lehet:

```
$qstr =  SELECT * FROM bizalmasadatok
WHERE tulajdonos= $PHP_AUTH_USER
➥AND tipus=$tipus ;
```

Itt a `$tipus` helyére a terv szerint egy típust azonosító, legördülő menüből választható számnak kell kerülnie. Az űrlapot mentve és a legördülő menüt módosítva viszont ez is a `$tipus` változóba kerülhet: `1 OR (tulajdonos= masvalaki )`. Ekkor egy ilyen lekérést kapunk:

```
SELECT * FROM bizalmasadatok
WHERE tulajdonos= $PHP_AUTH_USER
➥AND tipus=1 OR (tulajdonos =  masvalaki )
```

Ezáltal a lekérés minden rekordra teljesül, ahol az adott típussal a saját bizalmas adataink is szerepelnek, és a „masvalaki” belépőkódú ismeretlen adatai is a listához csapódnak. Megszokott (terv szerinti) esetben a másik felhasználó bizalmas adataihoz nem férhetünk hozzá, de egy ilyen kívülről megbütykölt lekéréssel ez mégis megtörténhet. A legjobb lenne, ha figyelnénk, hogy `$tipus` csakis egy 0 és 50 közti egész szám lehet, minden más értéket eldobva. Ehelyett ott van a `magic_quotes_gpc`, amely a fenti gondot annyival oldja meg, hogy a két aposztrófot egy-egy fordított perjellel levédi, így a lekérés a jogtalan adatszerzést meghiúsítva SQL-hibát hoz. Ez a védőbástya igencsak minimális védelmet nyújt. Előfordulhat ugyanis olyan lekérés is, ahol minden szűrőfeltétel számszerű. Talán megfontolandó a számszerű adatokat is idézőjelek közé szorítani, mert ekkor ez és a `magic_quotes_gpc` jól együttműködik. Hangsúlyozom azonban, hogy a biztos védelmet csakis alapos adatellenőrzéssel lehet elérni.
Frissítéskor a telepítő szerencsére az eredeti beállításokat nem bántja, így nem kell attól tartanunk, hogy a `magic_quotes_gpc` beállítás a tudtunk nélkül egyszer csak megváltozik. Érdemes viszont észben tartani, hogy a jövőt illetően a PHP ezen képessége is nemkívánatossá vált. A PHP-terjesztésekben található ***php.ini-optimized*** állomány tartalma is ezt igazolja. Ez már az új elvárásoknak felel meg, ámde a régebbi alkalmazásokkal nem együttműködő beállításokkal rendelkezik. A PHP fejlesztőgárdája a jövőben a `magic_quotes_gpc` alapbeállítását `Off` állásba fogja helyezni Az adatok kezeléséről tehát mindenkinek magának kell gondoskodnia. A dolog oka prózai: ezzel is rá kívánják venni a PHP-ben fejlesztőket, hogy minden egyes bejövő adatot maguk ellenőrizzenek. A túl nagy kényelem segíti a figyelem lankadását, márpedig egy korrektül megírt programnak `magic_quotes_gpc` nélkül is biztonságosan kell üzemelnie.

Safe Mode

A PHP-t is adó tárhelyszolgáltatás egyik alapfeladata, hogy minden egyes a kiszolgálón futó PHP-parancsfájl ugyanazon felhasználóazonosítóval fusson. Ez az azonosító azonos az Apache webkiszolgálóhoz beállítottal. Ez előhív egy igen fájdalmas probléma, amelynek során a szerveren ügyködő előfizetőket nem tudjuk védeni egymástól, hiszen minden PHP egy ***nobody*** vagy ***www-data*** vagy hasonló nagy közös felhasználó neve alatt működik. Ezáltal egy egyszerű program segítségével az egész webkiszolgáló terület átböngészhető. Márpedig ez nyilvánvalóan nem szerencsés, mivel ilyen módon bárki hozzáférhet a kiszolgálón a másik felhasználó legféltettebb PHP- és egyéb állományaihoz. Például ahhoz is, amelyben a kényes adatokat tartalmazó MySQL-adatbázishoz tartozó felhasználó azonosító-jelszó párost tárolja.
A jelenlegi Apache-beállításokban nincs is lehetőség arra, hogy a különböző virtuális kiszolgálók különböző felhasználóazonosító alatt fussanak. Egyedül ***suexec***-kel fordított Apache esetén tudjuk ezt a CGI módban működő PHP-vel ezt mégis megtenni – ami viszont rengeteg egyéb rizikót hoz be a képletbe. Megoldást ebben az Apache 2.x változatai fognak hozni, ahol már minden további nélkül virtuális szerverenként állítgatható lesz az `User` és `Group` Apache fordítói utasítás, és ez a modulként beépülő PHP esetén is hatni fog. Eképpen a kiszolgálón előfizető felhasználók már nem figyelhetnek be mások féltett könyvtáraiba.
Addig sem maradhat tétlen az ember. A fenti problémák orvoslására született meg a ***Safe Mode*** PHP-kiegészítés. Ezt a biztonságosabb üzemeltetést biztosító futási módot a ***php.ini***-ben tudjuk ki-bekapcsolni:

```
safe_mode = On vagy Off
```

Ilyen bekapcsolt állapotban a PHP több oldalról is igyekszik levédeni a kiszolgálón tartózkodó adatokat. Egyrészt korlátozható a fájlrendszerben az a terület, amit a PHP látni képes. Ezt és magát a biztonságos üzemelést biztosító kapcsolót az Apache beállító állományán keresztül akár virtuális kiszolgálónként is kapcsolgathatjuk:

```
<VirtualHost www.vedett.hu>
    .. egyØb beÆll tÆsok ..
    php_admin_value safe_mode 1
    php_admin_value open_basedir
/var/www/html/safephphost/
</VirtualHost>
```

Ha bekapcsoljuk a `safe_mode` állapotot, a PHP a PHP-fájlok tulajdonosi viszonyainak is figyelmet szentel. Innentől a PHP-parancsfájl csak a tulajdonosával azonos birtokviszonyú fájlokkal tud valamit kezdeni. Más esetben a műveletet egy `Warning: SAFE MODE Restriction in effect.` kezdetű hibaüzenettel visszautasítja. Ez a korlátozás a PHP összes fájlokat kezelő vagy bármilyen műveletre felhasználó parancsára érvényes.
Az `open_basedir`-ben megadott könyvtáron kívül ekkor a PHP nem lát, és ugyanez történik a PHP által meghívott parancssori alkalmazásokkal is. Ez a védelem persze korántsem

tökéletes, hisz nem a megfelelő szinten próbálja orvosolni a problémát. Az `open_basedir` a `safe_mode` ki- vagy bekapcsolt állapotától függetlenül működőképes. Az `open_basedir` után több könyvtár is megadható, itt szintén kettőspont az elválasztó karakter. Fontos megjegyeznünk azt is, hogy az `open_basedir` valójában nem könyvtárat ad meg, hanem az elérési útvonal kezdeti karaktereit. Tehát a */var/www/phpinc* a */var/www/phpinc/* könyvtárnak felel meg, de ugyanígy a */var/www/phpinclude* vagy a */var/www/phpincs* könyvtáraknak is megfelelhet. Érdemes tehát az ebből előálló galibákat elkerülendő a megadott könyvtárakat záró perjellel ellátni.
A `safe_mode`-nak is létezik egy, az `open_basedir`-hez hasonló könyvtármegadási lehetősége. A PHP csak a `safe_mode_exec_dir` után megadott könyvtáron belül hagyja, hogy a futtatható állományokat a `system()`, `exec()` vagy hasonló függvényekkel meghívhassuk. A végrehajtó operátor, azaz a balra dőlő aposztróf, amelynek segítségével egyszerű módon lehet parancssori utasításokat futtatni, a `safe_mode` bekapcsolt állapotában egy az egyben le van tiltva.
A `disable_functions` ***php.ini*** fordítói utasítás után a vesszővel elválasztva felsorolt függvények a programok számára nem lesznek elérhetőek. Végső finomításként talán érdemes körülnézni, hisz pár veszélyesnek (és a feladat szempontjából feleslegesnek) ítélt függvény lekapcsolása is meghosszabbíthatja a nyugodtan átaludt órák számát.

Mérföldkő: PHP 4.1.0 és register_globals

A PHP népszerűségét jórészt annak köszönheti, hogy könnyedén és gyorsan készíthető vele az Internetre vagy a belső hálózatra szánt alkalmazás. E könnyedség egyik oka, hogy a PHP alapesetben az űrlapokról, URL-ekből, sütikből érkező adatokat azonnal változóban tálalja. Ez rengeteg terhet vesz le a programozó válláról, de sajnos így könnyedén lehet biztonságosnak egyáltalán nem mondható kódot írni. Emellett további könnyítés az is, hogy változóinkat nem szükséges a programkód elején előre deklarálni. Emiatt viszont kívülről bármilyen nevű és értékű változó bejuttatható a futó PHP-programba.
Egy példa, ami jól mutatja ennek veszélyeit
(forrás: ➲ http://www.php.net):

```
<?php

if (authenticate_user()) {
  $authenticated = true;
}
...

?>
```

A fenti példa feltételezi, hogy a felhasználó azonosítást egy `authenticate_user()` nevű PHP-függvény végzi (a felhasználónév-jelszó páros kiértékelésével). Rendes körülmények között a fenti kódrészlet az `$authenticated` változónak abban az esetben ad „igaz” értéket, ha a belépés sikeres volt. A kód további részében a program ezt a változót használja fel a jogosultság megállapítására. Feltéve, hogy ezt a PHP-ben megírt oldalt ***nagyon_nagy_titkaim.php***-nek nevezik, a következő kézzel módosított URL-lel hívva a jogtalan behatolás meg is történt:

```
nagyon_nagy_titkaim.php?authenticated=1
```

Természetesen egy kis adalékkal a fenti kód is megfelelő lenne, mert a feltétel kiértékelése előtt elég lenne az `$authenticated` változót „hamis” értékkel ellátnunk. Ezt az `else`-ágban is megtehetnénk, így védve ki a csalást. Ezeknél azonban sajnos néha sokkal kevésbé szembetűnőbb logikai hibákat is ejthetünk a programtervezés során, és a programunk esetleg kívülről manipulálhatóvá válhat.
A hosszú távú tapasztalatok – és a tény, hogy a PHP mennyire elterjedt – nyomán született meg az a döntés, amelyet a tavalyi év végén a PHP fejlesztőcsapata hozott meg. A 4.1.0 változattól a ***php.ini***-ben a `register_globals` néven ismert kapcsoló alapértelmezésként kikapcsolt állapotban kerül az újonnan telepített PHP beállításait tartalmazó ***php.ini*** fájlba. E kapcsoló segítségével lehet a PHP fenti tulajdonságát, azaz a kívülről jövő adatokból automatikusan változókat létrehozó képességét ki- és bekapcsolni. Jelenleg a `register_globals` `Off`-ra állítása még csak ajánlás, de a fejlesztők nem titkolt szándéka az, hogy idővel ezt a szolgáltatást teljes egészében kivegyék a PHP-ből. Ezáltal a PHP-ben való fejlesztés kissé nehézkesebbé válhat, de ez a nehézség szükségszerű.
Az ajánlásnak eleget tenni és meglévő programjainkat mindezen körülményeknek megfeleltetni meglehetősen nagy feladat.
A nagy traumát enyhítendő bevezetésre került hét új, az átállást és a jövőbeni programozást valamelyest megkönnyítő asszociatív tömb, amelyek minden futás során automatikusan létrejönnek:

- $_GET - a webcímbe (URL) ágyazott, valamint a GET-eljárással elküldött űrlapadatok elérhetősége. Tartalmában a régóta jól ismert `$HTTP_GET_VARS` tömbnek felel meg.
- $_POST – a `$HTTP_POST_VARS` megfelelő párja. A POST-eljárással küldött űrlapadatok elérésére szolgál.
- $_COOKIE – a „süti” forrásból érkező adatokat tartalmazó `$HTTP_COOKIE_VARS` megfelelő párja.
- $_SERVER – a `$HTTP_SERVER_VARS` rövid nevű társa.
- $_ENV – kitalálhatatlan, de a `$HTTP_ENV_VARS`-nak felel meg.
- $_REQUEST – nem, nem, ilyen eleddig nem volt: ez a `$_GET`, `$_POST` és a `$_COOKIE` tömbök uniója. Itt megtalálható minden olyan adat, ami a kiszolgálón kívülről származik. Más szóval azok, amelyekben nem bízhatunk vakon.
- $_SESSION – a PHP saját munkamenet- (session) kezelését megvalósító moduljának tárolt adatai.

Ennyiben nem merül ki a fejlesztőknek az átállni szándékozókat célzó segítőkészsége. A fenti hét tömb PHP-kódunkban minden további nélkül bárhol elérhető. Függvényből hivatkozva rá szükségtelen mindjárt az elején a `global` paranccsal a függvény felségterületére behúznunk. További könnyítés, hogy a `$_SESSION` tömbbe közvetlenül írt adatok ugyanúgy bekerülnek a körforgásba, mintha egy `session_register()` segítségével tettük volna meg. Ilyen tulajdonsággal a többi tömb (a hétből) nem rendelkezik.

Programkövetés

És mindezeknek a végére hagytam még egy nagyon fontos szempontot. A PHP-t folyamatosan fejlesztik, a nyilvánosságra került hibákat (köztük a biztonsági szempontból érdekeseket) időről időre javítják. Fontos tehát telepített PHP-nkat állandó jelleggel frissíteni.

Heilig (Cece) Szabolcs
(cece@php.net) Veszprémben él. Hobbija, kedvenc időtöltése és munkája is a programozás. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalatoskodni.

A Squirrelmail és bővítményei

Ígéretemhez híven részletesebben is megnézzük a Squirrelmailhez található bővítményeket.

A bővítményeknek (plugins) fontos szerep jut a használhatóság terén. A Squirrelmail mindenfajta kiegészítő nélkül csupán néhány pluszt tartalmaz, bár ez is jelentős haladás a régebbi változatokhoz képest, amelyek az alaptelepítés után nem sok mindent tudtak felmutatni. A legfrissebb változat az alábbi lehetőségeket kínálja: naptár, felügyelő, helyesírás-ellenőrző, fordító, levélszemét- (spam) felderítő. Ezek csak a legfontosabb kiegészítők, rajtuk kívül található még néhány a csomagban. A további bővítményekhez, illetve frissítésükhöz a ➲ http://www.squirrelmail.org/plugins.php címen juthatunk hozzá.

Alapértelmezésként bővítmények egyike sincs telepítve. Tegyük meg a legfontosabbakkal: lépjünk be a program főkönyvárába:

```
root@mail# cd /var/www/squirrelmail
```

és adjuk ki a következő parancsot:

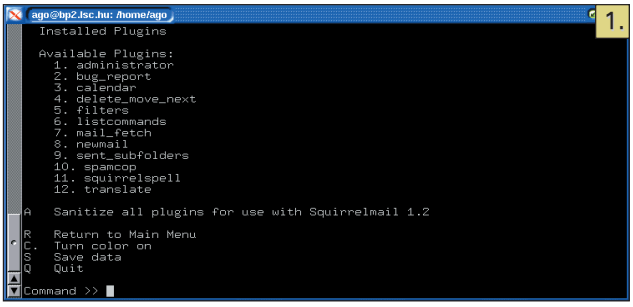
```
root@mail#/var/www/squirrelmail# ./configure
```

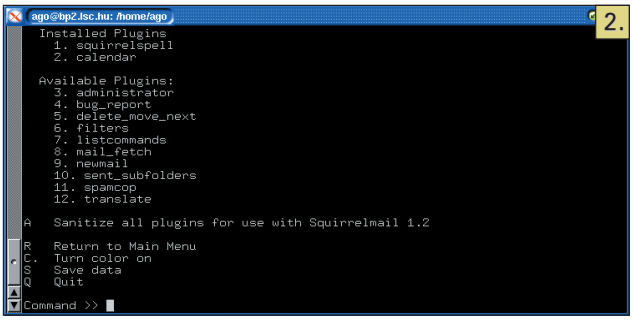
Ekkor az előző számból ismert, rendkívül egyszerű kinézetű, de hallatlanul hatékony beállítófelülethez érkezünk. Most válasszuk ki az eddig még nem használt nyolcas pontot – az *1. képhez* hasonlót kell látnunk.

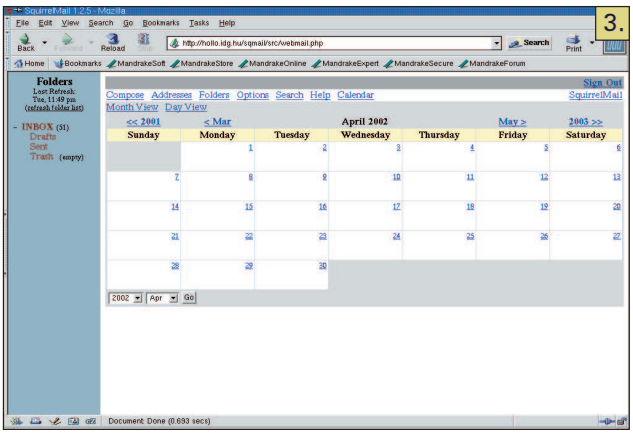
Telepítsük először a naptár- és a helyesírásellenőrző modult. Ehhez nem kell mást tennünk, mint a billentyűzeten a bővítmények neve mellett álló számot lenyomni. Egy kevéske angolnyelv-ismeret, illetve a mellékelt és a fenti címen megadott leírás, valamint az ismertetők olvasása nem árthat nekünk. Ha a naptárt (calendar) és a helyesírás-ellenőrzőt (squirrespell) telepítettük a levelezőbe, a *2. képhez* hasonlónak kell fogadnia minket. A beállításokat az **s**, majd az ENTER billentyű lenyomásával mentsük! Ha most belépünk a levelezőbe, a változás első ránézésre nem is fog feltűnni. A menüpontok között azonban rögtön feltűnik a ***Calendar***. Ezt kiválasztva egy haviképhez jutunk, ahol a pillanatnyi keltezés mellett a ***Today*** felirat található és egy dátumra kattintva a következő kép bukkan fel. Kattintsunk például április 1-jére, ekkor a *3. képen* látható átmenetben lesz részünk: itt valamely óra melletti ***Add*** feliratra kattintva az *5. képhez* hasonló látvány köszönt bennünket.

A hozzáadott esemény mind a napi, mind a havi nézetben megjelenik. Ha kívánjuk, a napi nézeten megjelenő hivatkozásokkal ezt később szerkeszthetjük (***Edit***), illetve törölhetjük (***Del***) is.

A másik bővítmény, amit részletesen megvizsgálunk, a helyesírás-ellenőrző. Ezt levél írásánál láthatjuk megjelenni: az eddigi menüpontok mellett feltűnik a ***Check Spelling*** gomb. Működéséhez szükség van az ***ispell*** nevű alkalmazásra, mely már magyar helyesírás szerint is tud szöveget ellenőrizni. A magyar ispell a ➲ http://www.szofi.hu/gnu/magyarispell/index.html címen érhető el. A levél szövegének beírása után a ***Check Spelling*** gomb elindítja a kiszolgálón a helyesírás-ellenőrzőt, és átadja a szöveget. Ha hibát talál benne, javaslatot tesz a hiba kijavítására, és lehetőségünk nyílik a helyesbítésre.

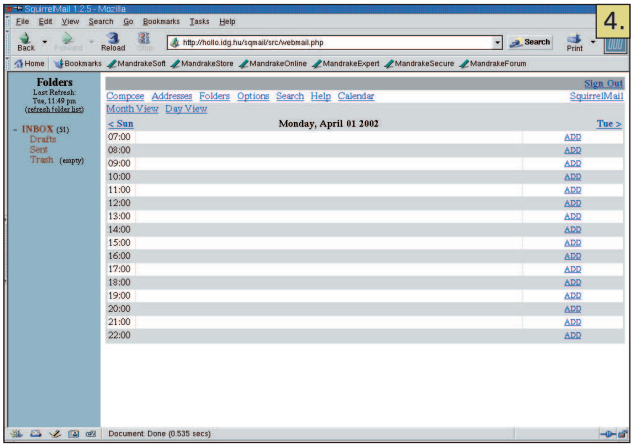


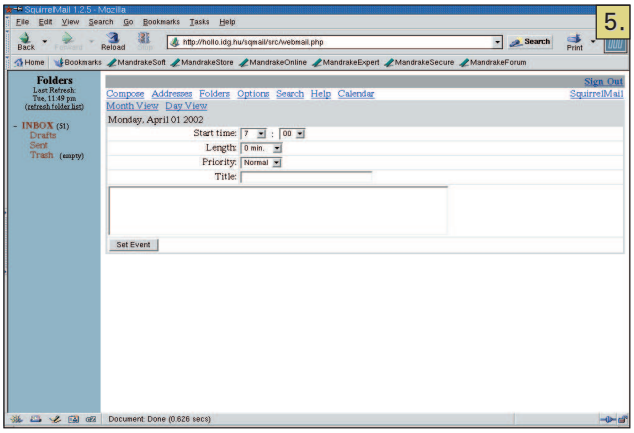


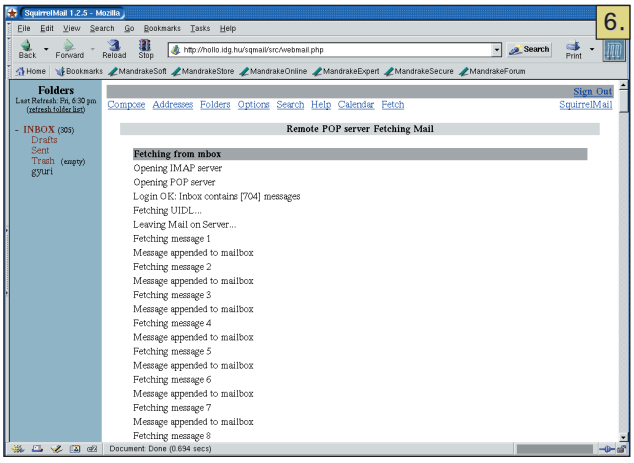


Néhány egyéb modul, amelyeket rendkívül hasznosnak tartok:

- ***Address Take***: lehetővé teszi, hogy egy levélből kiemeljük a címeket és a címtárunkba helyezzük őket.
- ***AddressBook Import-Export:*** CSV-be (vesszőkkel elválasztott lista) mentett címjegyzéket állíthatunk elő, illetve tölthetünk be a saját címtárunkba. Ilyen formában képes menteni, illetve olvasni belőle az Outlook és az Outlook Express is.
- ***Password Forget:*** az újabb böngészők hajlamosak rá, hogy megjegyezzék a begépelt felhasználói nevekhez tartozó jelszavakat. Ez biztonsági szempontból igen hátrányos lehet – ezen segít ez a bővítmény. Használatával a felhasználó még akkor sem tudja a jelszavát megjegyeztetni a böngészővel, ha akarja, éppen ezért ez a személyes kedvencem.







- ***Filters:*** a levelet önműködően tudjuk különböző alkönyvtárakba küldeni a címzett (To), a másolati példány (CC), a Küldő (From) és a Tárgy (Subject) alapján.
- ***SpamCop:*** lehetővé teszi, hogy a levélszemetet (spam) a megfelelő modul kezelje. Telepítése után egy Report As Spam hivatkozást helyez el a levél olvasását lehetővé tévő ablakban. Ha erre kattintunk, elkezdődik a folyamat, melynek eredményeképpen a küldőt mint levélszemétterjesztőt bejelentjük. Ennek eredménye lesz, hogy a postmaster@kuldocime.hu címre egy levél fut be, ami felhívja a postamester (postmaster) figyelmét, hogy a felhasználók között valaki „rendetlenkedik”.
- ***List Commands***: telepítése után a felhasználó, ha levelezőlistáról kapott levelet, egy újabb menüsort láthat, amelyben a levelezőlista személyét érintő kezeléséhez kap lehetőséget, többek között a le- és feliratkozáshoz, az archívum böngészéséhez stb.
- ***Mail Fetch:*** külső POP3-kiszolgálóról tölt le leveleket és az IMAP postafiókjába teszi, ami az IMAP-protokoll alapértelmezett levéltároló könyvtára.
- ***Delete Move Next:*** újabb hivatkozásokkal bővíti a levél olvasása közben elérhető lehetőségeket. Ilyen például a ***Delete & Next*** (Töröl és következőre ugrik) vagy a ***Delete & Previous*** (Töröl és az előzőre ugrik).
- ***Timeout User:*** ez a bővítmény önműködően kilépteti a felhasználót, amennyiben egy adott ideig tétlen volt. Rendkívül hasznos, ha sok olyan felhasználónk van, aki nem szeret kilépni az alkalmazásból.
- Qoute-of-the-day at login: a bejelentkező képernyőn egy idézet fog minket köszönteni, mely véletlenszerűen kerül kiválasztásra. A `fortune` programot is képes használni, ekkor az idézeteket annak adatbázisából választja ki.

A bővítmények telepítésével tehetjük teljessé rendszerünket, előtte azonban mindig győződjünk meg róla, hogy hibátlanul együttműködnek-e a Webmail magjával. Nem arról van szó, hogy a Squirrelmail eltüntetné a leveleket vagy hasonló csúnyaságot tenne, de lehet, hogy sokat lassít rajta vagy zavaró lesz a működése. Ennek jó példája, ha olyan modult rakunk fel, amely önműködően frissíti a könyvtárak tartalmát. Én egyszer feltelepítettem ezt a modult, de túl rövid időre állítottam be a frissítést és majd megőrültem, amikor a könyvtárak adatait tároló rész levélolvasás közben tíz másodpercenként újra letöltődött. Kipróbálásához a legjobb eszköz asztali számítógépünk, ha azon kialakítunk egy Webmail rendszert és helyben ellenőrizzük, megkíméljük magunkat attól, hogy a felhasználók esetleg (jogosan) panaszkodjanak, vagy valami mégis balul üssön ki.

A változatosság gyönyörködtet

Úgy tartják, hogy az életben a legjobb dolgok is csak akkor elég jók, ha változatosak. Nem kivétel ez alól egy webmail rendszer sem (megjegyzés: vannak ennél sokkal jobb dolgok is). A változatosság egy webmail rendszer esetében leginkább a témázhatóságot érinti, hiszen a kinézet az, ami alapján sok mindent megítélünk. Kedvenc rendszerünk természetesen sok grafikus kiegészítővel rendelkezik – már alapcsomagként is. Ha ezek közül egy sem nyeri el tetszésünket, az eredeti felépítés tanulmányozása után akár mi magunk is készíthetünk különböző képeket. A már előre elkészített témák mellé bemásolhatjuk a sajátunkat és utána a Squirrelmail beállítóprogramjából már el is érhetjük majd. A téma kiválasztásához lépjünk be a rendszerbe, majd az ***Options*** (Opciók) menűpont alatt válasszuk ki a ***Display Preferences*** pontot (magyar fordítása az elég rosszul sikerült Általános beállítások). Itt a legfölső lenyíló listából kiválaszthatjuk a nekünk tetsző témát és az oldal alján a ***Submit*** (Elküld) gomb megnyomásával betölthetjük azt. A következő részben befejezzük a Squirrelmail ismertetését és egy új témába kezdünk, mely szintén a levelezéshez kapcsolódik.

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.

Digitális mozgóképsorok saját kezűleg

MEncoder – kikövezett út a hatékony tárolás felé.

Sorozatunk előző részében próbát tettem a mozgóképtárolás területén szerzett ismeretanyagunk rendszerezésére, valamint az elméleti alapok lefektetésére. Kísérletem sikerességének reményében a jelen és a következő írásban sokkal inkább a tettek mezejére szeretnék lépni, és bemutatom az általam ismert gyakorlati lépéseket. Mondanivalóm gerincét most is az mplayeres vonal alkotja. Az előzőekben tett ígéretemhez híven, be kívánom mutatni az MPlayer-csomag szerves részét képező MEncodert, mely – mint nevéből is látszik – a lejátszás inverz folyamatát végzi, és a segítségével videókat készíthetünk és tömöríthetünk. Sokat gondolkodtam azon, vajon hogyan is érzékeltethetném számotokra azt a meglepődést, amelyet a program használata keltett bennem, így megmutattam az egyik – videós körökben nagy tiszteletnek örvendő – ismerősömnek, aki a redmondi operációs rendszerrel szerzett komoly tapasztalatokat a témával kapcsolatban. Ez az ismerősöm csak ennyit tudott mondani: „Nagyon profi!”

A MEncoder mint eszköz

A program igazából az MPlayer különleges változata, egyfajta kiterjesztése, ezért ezt az alapfelfogást szem előtt tartva célszerű képességeit, jellemzőit elemeznünk. A MEncoder tulajdonképpen nem más, mint egy olyan MPlayer, amelynek kimenete a monitor helyett más kodekek bemenetét képezi, tehát tisztán tükrözi a jó öreg unixos filozófiát, miszerint a programok láncba fűzve kiválóan képesek egymással kapcsolatot tartani. Az MPlayer dekódoló része a lánc első tagja, amely a kép és hang kicsomagolása után a megjelenítési réteg helyett újabb kodekeket fűz a folyamatba, és ezek valamilyen más formában újra bekódolják az adatfolyamot, majd visszaírják az adattárolóra. Ennek következményeképpen minden olyan fájl „átalakítására” képesek vagyunk, amelyet az MPlayer le tud játszani, tehát a bemeneti fájlformátumok halmaza igen nagy. A kimenet egyelőre sajnos csak AVI lehet, és az alkalmazható kép- és hangkodekek listája is szegényes, megvalósításuk jelenleg is folyik, ennek ellenére már most is kiválóan használhatók (a későbbiek folyamán azt is látni fogjuk, hogy miért).

Használatbavétel

A használatbavétel gyakorlatilag megegyezik az MPlayerről szóló cikkben leírtakkal (Linuxvilág 2002., 3. szám, 28. oldal). Mindenképpen érdemes elolvasni, hiszen nem szeretném ismételni magam, tehát néhány helyen feltételezem az előző anyag tartalmának ismeretét. Az előkészületek során szinte pontosan ugyanúgy kell eljárnunk, mint az MPlayer telepítésénél. A forrás letöltése után a fordítás során egyszerűen a `make` parancs kiadásával létrejön: mind az MPlayer, mind a MEncoder lefordul. Célszerű a CVS-változatot használni, mivel rengeteg újítás (amely természetesen a MEncoder mellett az MPlayer részre is vonatkozik) még nincs benne a 0.6-os üzembiztos változatban, így előfordulhat, hogy a cikk során olyan szolgáltatásokat is kihasználok, amelyek a régebbi változatokban még nem találhatók meg – remélem, ez nem vezet félreértésekhez. A kodekek telepítésére hasonlóképpen kell ügyelnünk, mint az eddigiekben, hiszen a csomag mindkét része ugyanazokra az alapokra épülve dekódol. A hatékony munka érdekében nagyon ajánlott a DivX4Linux kodek telepítése, mert a MEncoder a tömörítésre egyelőre ezt tudja igazán jól használni (a DivX4Linux kodek a ➲ http://www.divx.com címről tölthető le). Ha hangot is szeretnénk kódolni, szükségünk lesz a LAME MP3 kodekre, melynek forrását a ➲ http://www.sulaco.org/mp3/ címről tölthetjük le. A futtatható állomány nem elég, magunknak kell fordítani és telepíteni. A kodekek telepítése után természetesen legalább a MEncodert újra kell fordítanunk. Ha a fentiekkel megvolnánk, jöhet az igazi munka!

A legfontosabb gyakorlati jellemzők

Mint már említettem, az MPlayer által ismert összes fájlformátumból kedvünk szerint kódolhatunk: lehetőségünk nyílik például V4L-megfelelő TV tunerekről történő kódolásra is, valamint minden olyan lineáris adatfolyamot feldolgozhatunk, amelyet az MPlayer szabványos bemeneteként meg tudunk adni. A folyamat során használhatunk külső audiosávokat, beépített képátméretezést, alkalmazhatunk többmenetes DivX4-et, változó bitrátájú MP3-kódolást, de a kép- és hangsávok változtatások nélkül történő másolására is lehetőségünk van (direct stream copy). Alapvetően mindig a kapcsolók, lehetőségek két nagy csoportját adjuk meg a MEncoder számára: az egyik a bemeneti adatfolyamra és annak kezelésére vonatkozik, a másik a kimenettel kapcsolatos tulajdonságokat fekteti le.
A bemenetre vonatkozó megkötések teljesen megegyeznek az MPlayernél alkalmazottakkal: a bemeneti forrásra ugyanúgy hivatkozunk, kérhetjük külső audiofájl lejátszását, vagy épp a hang nélküli megjelenítést, megadhatjuk a dekódoláshoz alkalmazandó video- és hangkodekeket, kodekcsaládokat. Számunkra azonban a MEncoder kimenete és annak jellemzői az érdekesek. A kimenet tulajdonságai további két csoportra bonthatók: egyik része az audio-, másik a videosávokra vonatkozik.
A kimenetre alkalmazott videokodeket a `-ovc` *`kodek neve`*, a hangkodeket pedig a `-ovc` *`kodek neve`* kapcsolókkal választhatjuk ki, továbbá a rendelkezésre álló kodekek listája a `-ovc help`, illetve a `-oac help` kapcsolókkal kérdezhető le. Attól függően, hogy melyiket választottuk, további kapcsolók segítségével lehet megadni az egyes kodekekre jellemző tulajdonságokat.
A `-divx4opts` a DivX4, a `-lavopts` a libavcodec, a `-lameopts` kapcsoló után pedig a LAME mp3 kodek tulajdonságai adhatók meg. Az egyes kapcsolókat itt szóköz nélkül, kettőspontokkal elválasztva kell átadni. Nézzük meg a két legfontosabb kodekhez használható finombeállításokat! Nem térnék ki minden egyes kapcsolóra, inkább csak azokat írom le, amelyeket a mindennapos esetek során szoktunk használni (a teljes lista egyébként a MEncoder leírásában is megtalálható).

- A Divx4Linux legfontosabb kapcsolói (`-divx4opts`):
  `help`: ezzel kérdezhetjük le a

Kulcsképkockák

A mozgóképek tömörítésének alapja: az egymást követő képkockák csak kis mértékben különböznek egymástól, ezért elég csupán az egyes képekhez viszonyított különbséget tárolnunk. Azokat a képeket, amelyekhez képest csak a különbségeket tároljuk, kulcsképkockáknak nevezzük. (A többi kép nem is igazi kép, inkább csak válozásvektorok, amiből az eredeti kép kiszámítható.) Ez egészen addig megy így, amíg a különbségek már akkora méretűek, hogy célszerűbb beszúrni egy új képkockát, és az azt követő képeket pedig ahhoz viszonyítani. (Ezek általában vágásoknál, gyors mozgásoknál találhatók egy-egy filmben.) A valóságban ez természetesen bonyolultabb, a hatékonyabb méretcsökkentés érdekében általában a kulcshoz viszonyított különbségekhez viszonyított (különbségekhez viszonyított... és folytathatnám) képeket tárolnak a kulcsképkocka után.

kapcsolók listáját és jelentésüket.
`br=<szÆm>`: a tömörítési bitrátát (a tömörítés mértékét) határozza meg. A kódolásnál Alapértelmezetten változó bitrátát alkalmaz. Ez azt jelenti, hogy a mozgókép pillanatnyi jellemzőitől függően kisebb-nagyobb mértékben változtat a tömörítés mértékén, így például gyors mozgásoknál sem romlik a kép minősége.
`key=<szÆm>`: meghatározza, hogy legfeljebb hány képkocka követheti egymást anélkül, hogy kulcsképkockát helyeznénk el. A kulcsképkockák (keyframe) olyan képek, amelyekhez a tömörítés során az azt követő képet viszonyítják, ezenkívül összehangolási célokat is szolgálnak. Alapértelmezés szerint a kodek, ha szükségesnek érzi, ezeket a képkockákat önműködően szúrja be, ám például hosszú állóképeknél előfordulhat, hogy ez akár fél percig sem történik meg, ami később, a pozicionálásnál gondot okozhat, ráadásul hibaérzékennyé válik. E kulcsképkocka-mentes tartományok méretét szabályozhatjuk a fenti kapcsolóval.
`q=<1-5>`: segítségével a tömörítés minőségét határozhatjuk meg. Alapértelmezetten 5 – ez a legjobb minőségű és a leglassabb is egyben. Ahogy csökken a minőség, úgy nő a feldolgozási sebesség. Például:

```
-divx4opts
➥br=1800:q=3:key=250
```

- A LAME mp3 kodek kapcsolói (`-lameopts`):
  `help`: ezzel kérdezhetjük le a kapcsolók listáját és jelentésüket.
  `br=<szÆm>`: a tömörítési bitrátát (tömörítés mértékét) határozza meg. Ez a kodek alapértelmezetten állandó tömörítési arányt használ, a kimenetet a pillanatnyi bemenettől függetlenül ugyanakkorára tömöríti. A változó tömörítési arány használatára is lehetőség nyílik, ám ezzel könnyen állíthatunk elő olyan filmeket, amit aztán Windows alatt nem lehet majd lejátszani, épp ezért együttműködő képességének megőrzése végett nem árt tartózkodnunk tőle.
  `q=<szÆm>`: ugyanaz, mint a DivX4Linux-kodek esetében.
  `mode=<0-3>`: meghatározhatjuk a többcsatornás kimeneti MP3 (hangsáv) tulajdonságait. Négy lehetőség közül választhatunk: stereo, joint-stereo, dualchannel, mono. Például:

```
-lameopts br=192:q=4:mode=1
```

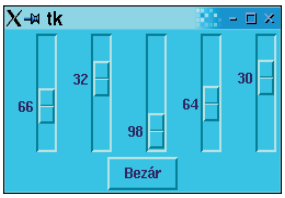
Arra is módunk van, hogy a kép- vagy hangsávokat feldolgozás nélkül csak átmásoljuk, ekkor a `-oac copy` vagy a `-ovc copy` kapcsolókkal kell a műveletet elindítanunk, és a kodekek jellemzőit nem adhatjuk meg, hiszen nincs is értelme. Ezt a lehetőséget akkor érdemes használni, ha például DVD-ről „mentjük le" filmjeinket, és meg szeretnénk őrizni az eredeti AC3-hangot. Ilyenkor a képkimenetnek megadjuk a divx-kodeket, a hangot pedig feldolgozás nélkül átmásoljuk. Ez természetesen nem az egyetlen alkalmazási lehetőség: ezzel a módszerrel a hibásan összefésült (összefésületlen) vagy a hibás indexű AVI-k is javíthatók. A kép és hangsávokat egyszerűen másoljuk át egy másik fájlba, közben önműködően összefésülődnek, és az index is újból helyesen jön létre.
A kimeneti kodekek kapcsolókkal való ellátásán kívül még számos olyan adatot adhatunk meg, amely befolyásolja az elkészült filmet. Mindenekelőtt a `-o` *`kimeneti fÆjl`* kapcsolót kötelező megadnunk. Általában szeretnénk tudni, hogy a megadott kapcsolókkal létrehozott AVI hogyan fog kinézni, még mielőtt nekikezdünk a fél napig tartó tömörítésnek. Ilyenkor csak egy rövid részletet célszerű kódolni, így ellenőrizve a filmet. Erre egyrészt a pozícionáló kapcsolók szolgálnak, másrészt megadhatjuk, hogy a program hány képkocka tömörítése után álljon le. A `-ss <  :pp:mp>` kapcsoló hatására a kimeneti fájl kezdete a megadott idő lesz, a `-endpos <  :pp:mp>` kapcsolóval megadott időpontnál pedig a bementi fájl feldolgozása befejeződik. A `-ss` kapcsoló után a `-frames` kapcsolóval – a képkockák számával – is megadhatjuk, hogy a tömörítés mikor érjen véget. Kellően rövid időintervallum megadásával könnyedén ellenőrizhetjük a kódolás eredményét. Ez természetesen csak egy lehetőség, a közvetlen kép- és hangsáv másolásával összekötve részleteket másolhatunk ki egy filmből, levághatunk belőle. A `-pass <1/2>` kapcsolóval határozhatjuk meg, hogy egy többmenetes divx-kódolás esetén épp melyik lépést alkalmazzuk. Ha túl nagy felbontású forrásunk van, a képet úgy átméretezhetjük, hogy a kimeneti fájl kisebb legyen, de akár nagyíthatunk is rajta, ha kedvünk tartja. Néha az átméretezés kifejezetten szükséges, mivel az MPEG-2 formátumok fizikai felbontása az esetek nagy többségében eltér a helyes képaránytól, így ezt a dolgot is a méretezéssel kell megoldanunk. A pontos értékeket a `-x <kimeneti szØlessØg>` és a `-y <kimeneti magassÆg>` kapcsolókkal adhatjuk meg. A `-sws <0-2>` kapcsolóval az átméretezési eljárást határozhatjuk meg, amely a kimeneti kép minőségét befolyásolja. Három lehetőség közül választhatunk: fast bilinear (ez a leggyorsabb), bilinear, bicubic (ez a legjobb minőségű). Egyelőre hiányolom a programból a filmek széléről történő keretek levágásának lehetőségét (cropping), ám a fejlesztők remélhetőleg ezt a szolgáltatást is hamarosan megvalósítják.
A következő cikkben részletesen szeretnék beszélni a többmenetes kódolásról, a szabványos bemenetről történő kódolásról, amellyel több fájl összefűzésére nyílik lehetőség és a különböző bemeneti egységekről történő közvetlen tömörítésről. Szeretném továbbá egy példán keresztül bemutatni egy DVD-lemez tartalmának áttömörítését, valamint az egyéb más fájlokból történő kódolást is, úgy, hogy közben a teljes folyamatot elemezni tudjuk. Megpróbálom bemutatni a videotömörítés lépéseit. Kapcsolódó anyagok a 33 CD Magazin/MPlayer könyvtárban találhatóak.

Komáromi Zoltán
(komi_@freemail.hu)
21 éves, a BME hallgatója, mellette PHP-programozóként dolgozik. Kedvenc területe a multimédia.

Tkinter felsőfokon – Python-szálak Tkinterből

Ebben a részben a Python szálkezelési lehetőségeivel ismerkedünk meg, egyúttal azt is megtudhatjuk, miként érhetjük el grafikus felületről ezeket a lehetőségekkel.

A Python-szálak használatával lehetőségünk nyílik arra, hogy egyetlen programon belül több végrehajtási szálat hozzunk létre. A szálakat alprogramokként képzelhetjük el, amelyek teljesen ugyanúgy néznek ki, mint a főprogram, ugyanazt a memóriaterületet használják, ugyanazokkal a fájlkezelőkkel bírnak, és még a végrehajtódó kód is teljesen egyező. Az egyetlen különbség közöttük az, hogy az egyes szálakban a végrehajtás a kód más-más részén helyezkedik el. Alapesetben a program futása egyetlen szálon hajtódik végre, vagyis a folyamatnak (process) saját memóriaterülete és egyéb erőforrásai vannak, melyeket az `os.fork()`-kal vagy `os.system()`-mel létrehozott új folyamatok nem érnek el.
Szálak létrehozására akkor van szükség, ha egy program által kizárólagosan használt erőforrást többféleképpen is el szeretnénk érni, vagy pedig ha programunknak két vagy több működő feladatot kell folyamatosan ellátnia. Ez az utóbbi főként a GUI-k használata során merül fel, amikor a felhasználói felület is folyamatos frissítést kíván, illetve a programnak egyéb, halasztást nem tűrő, olykor hosszadalmas teendői is lehetnek – ha ezek a főprogrammal egy szálon futnának, azt eredményeznék, hogy a felhasználói felület látszólag huzamosabb időszakokra teljesen lebénul.
A többszálú programok futása úgy zajlik, hogy a folyamat, vagyis a teljes program számára rendelkezésre álló idő megoszlik a program szálai között. Ez annyit tesz, hogyha például egy programban tíz szál fut, beleértve az elsődleges szálat is, akkor az egyes szálakra a teljes folyamatra összesen jutó idő 1/10 része esik.
A szálak (thread) használata során programunkat különleges gondossággal kell megtervezni, főleg az egyes szálak adatelérését és a közös adatok módosítását tekintve. Ha egy adatot a program több különböző szála nagyjából megegyező időben módosít, a programfutás kimenetele onnantól kezdve teljesen bizonytalanná válik. Ennek elkerülése érdekében a programban a kritikus részeknél kölcsönös kizárások (úgynevezett `mutex`-ek), vagy egyéb összehangoló eszközök használata szükséges, biztosítva, hogy egy adott részt egyszerre csak egy szál hajthasson végre.
A Python parancsértelmezőjében egy időben egyszerre csak egy szál futhat, melyet egy szigorúan szabályozott központi `lock`-oló eljárás biztosít. A parancsértelmező dolga, hogy bizonyos időközönként más szál kerüljön végrehajtásra. Alapértelmezésben ez a „bizonyos időközönként” 10 bájtkódnyi időt jelent. Ennyi utasítást hajt végre a Python, mielőtt a következő szálnak adná át a vezérlést. Ezen a `sys.setcheckinterval()` függvény segítségével változtathatunk.
Ha Python-programjainkból olyan külső C/C++ modulokat használunk, amelyeket nem kifejezetten többszálas végrehajtásra terveztek, akkor a külső program futása során az összes többi szál várakozni kénytelen. Szerencsére a Pythonban található modulok nagy részét felkészítették a többszálas végrehajtásra, így ezekkel nagy valószínűséggel nem lesz gondunk. Itt kell még megjegyeznünk, hogy többszálú programok futtatása esetén programunk a jelek és megszakítások fogadásakor furcsán viselkedhet. Például a billentyűzetmegszakítás kivételét bármely szál elkaphatja, ellenben a `signal`-modulban meghatározott jeleket csak a program elsődleges szála.



A Python-szálak ugyanúgy működnek minden POSIX-szálakat (pthreads) támogató operációs rendszer alól, a Windowst, a Solarist és a Linuxot is beleértve. Ellenben előfordulhat, hogy a rendszerhez adott Python-csomagot az adott rendszeren szálak támogatása nélkül fordították, ebben az esetben a Python parancsértelmezőjét nekünk kell újrafordítanunk. A Debiant használók helyzete ismét könnyű, hiszen esetükben a rendszerhez egy teljes, rendkívül zsúfolt Python jár, mely természetesen a szálak támogatását is tartalmazza.

Ismerkedés a szálak gyakorlati megvalósításával

A Pythonban a szálak kezelésére két modul is a rendelkezésünkre áll, az egyik a `thread`, a másik a `threading`. Ebben a cikkben az utóbbival fogunk foglalkozni, mely magasabb szintű szálkezelési felületet biztosít. A `threading`-modul – mely valójában maga is a `thread`-modulra épül – a `Thread` nevű osztályra épül. Ez az osztály az egész modulnak a lelke, az ebből származtatott új objektumok lesznek a programunk egyes szálai.
A `Thread` osztály a következő kezdeti értékekkel hívható:

```
Thread(group=None, target=None, name=None,
➥args=(), kwargs={})
```

A `group` értéknek a `ThreadGroup` használata során lesz majd szerepe, melynek megvalósítása még várat magára. Egyelőre az értéke kötelezően `None`. A `target` érték határozza meg a létrejövő szál által hívott első függvényt, melynek értéke alapesetben csak akkor lehet `None`, ha a `Thread` osztályt egy leszármaztatott osztályban használjuk és a `run()` eljárását felülírjuk. Ennek a `run()` eljárásnak a dolga lenne egyébként a `target` érték által meghatározott függvény meghívása.
A `name` érték értelemszerűen a szál nevét határozza meg, míg az `args` értékben egy `tuple`-ot, vagyis listát kell megadni, amely a `target` függvény értékeit határozza meg. A `kwargs` az `args`-hoz hasonló, azzal a különbséggel, hogy ebben nevesített kulcsszóparamétereket adhatunk át a `target` függvénynek.
Az *1. listában* egy egyszerű példaprogram látható, mely jól

1. *lista* Ismerkedés a szálakkal (thread1.py)

```
1. #! /usr/bin/python
2.
3. import threading
4. import time
5. import random
6.
7. def thrFunc(par):
8.     print "KezdØs: %d (i: %d)" % (par, i)
9.     time.sleep(random.choice((range(1,9))))
10.    print "    VØg: %d (i: %d)" % (par, i)
11.
12. for j in range(0, 15):
13.    t = threading.Thread(target =
       thrFunc, args = (j,))
14.    t.start()
15.
16. for i in range(0, 10):
17.    print "--- i: %d ---" % i
18.    time.sleep(1)
19.
20. print "V GE"
```

illusztrálja a szálak egyidejűségét, ugyanakkor egy közös változó használatával az is jól látható, hogy egy másik szál által megváltoztatott változó a közös adatterület miatt az összes szál változóira hatással van.
A program futása akkor ér véget, miután minden szál befejezte a tevékenységét. Ezt a viselkedést egyszerűen leellenőrzihetjük, ha a programból eltávolítjuk a „`for i`”-vel kezdődő bekezdést. Így az elsődleges szál futása azonnal végetér, de a program mindaddig nem lép ki, míg a többi szál nem végzett. Ha a 13. sor után beszúrunk a `t.start()` elé egy `t.setDaemon(1)` sort, akkor a létrejövő szál „Daemon” üzemmódban fog futni, vagyis az így létrejövő szálakra a program nem fog várakozni. Ebben az esetben programunk futása a VÉGE felirat megjelenítése után azonnal véget ér, anélkül, hogy az egyes szálak futása befejeződne. Ez azért van így, mert a program futása az elsődleges szálhoz kötött, így annak befejeződése egyben a program befejeződését is jelenti.

Összehangolás

Az egyes szálak közötti összehangolást az úgynevezett `mutex` objektumok, a `threading` modul használata esetén egyszerűen a `Lock()` függvény által visszaadott objektumok felhasználásával végezhetjük el. Összehangolásra akkor van szükség, ha egy változót egy több szál használta adatterületen módosítunk, ilyen módon elkerülhetjük programunk megzavarodását, mivel pontosan meghatározhatjuk, hogy egy változón melyik szál mikor és hogyan módosíthat.
A `Lock` objektumok használata rendkívül egyszerű. A példányosítást követően, vagy a `Lock()` függvény által visszatért objektumot felhasználva a következő metódusokat érhetjük el:

```
acquire([blocking = 1])
```

Az eljárásnak van egy nem kötelező értéke, amely alapértelmezésben 1, vagyis ha a `Lock` objektumot már lefoglalta valaki, akkor addig vár, míg fel nem szabadul. Ha ennek értéke `0`, és a `Lock` foglalt, akkor hamis értékkel azonnal visszatér.

```
release()
```

Ezzel az eljárással egy már lefoglalt `Lock`-ot szabadíthatunk fel. Programjainkban a kényes részeket e két utasítás között kell elhelyeznünk, így biztosítva, hogy egyszerre csak egy szál dolgozhat rajta.
A `Lock` objektummal nagyjából megegyező működése az `RLock` objektum, azzal a különbséggel, hogy az ezzel létrehozott objektumok többször is lefoglalhatók, vagyis az `acquire()` eljárás egy adott szálból több alkalommal is meghívható. Hogy az `RLock` felszabaduljon, a `release()` eljárást ugyanannyiszor kell meghívni, mint a lefoglalásért felelős párját.

Ugyanez Tkinterből

Grafikus felületek készítése esetén ügyelnünk kell arra, hogy a grafikus objektumok frissítéséért felelős `mainloop()` függvénynek a program elsődleges szálából kell futnia. Ha így járunk el, sok kellemetlenségtől kímélhetjük meg magunkat.
A Tkinter használatával a szálak kezelése lényegében ugyanúgy működik. A 2. listában (33. CD Magazin/Tkinter könyvtár) szereplő példánkban a külső szálak semmilyen hatással nincsenek a grafikus felületre, mivel a grafikus objektumok frissítése az elsődleges szálon belülről, segédváltozók segítségével történik.
A programból a ***Bezár*** gombra kattintva lehet kilépni. Amennyiben a programban a létrejövő szálak esetében a Daemon módot elmulasztjuk megadni (27. sor), abban az esetben a program ablakának bezárása esetén (ha nem a ***Bezár*** gombra kattintva akarunk kilépni), a program futása nem szakad meg, csak a program ablakai tűnnek el, de a program tovább fut.
A legtisztább kilépési mód, ha a ***Bezár*** gombra kattintunk, így a `self.ok` változó – mely a szálak futását vezérli – nullára vált, tehát a külső szálak sorra megszakadnak.

Összegzés

Ebben a részben a Python szálainak kezelésével kapcsolatos módszereket vettük át. Ezek ismeretében már belekezdhetünk saját több szálon futó programunk írásába is. A cikkben ismertetett módszereken kívül még egyéb lehetőségek is rendelkezésre állnak, melyek megismerésében a leírás nagy segítséget jelenthet.

A cikkhez tartozó példaprogramok megtalálhatók a lemezmellékleten, a 33. CD Magazin/Tkinter könyvtárában.

Gludovátz Gábor
(ggabor@sopron.hu) Szeret nyelveket tanulni, jól beszél angolul és németül, egy cseppet franciául, és ha már a nyelveknél tartunk, kedvencei közé tartozik még a Java, a C++ és a Python is.

Kapcsolódó címek

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

Virtuális CD-ROM-váltó építése

Használjunk olcsó, méretezhető Linux-kiszolgálót CD-ROM-lenyomatok megosztására a windowsos ügyfelek számára!

Írásom a virtuális CD-ROM-váltók (Virtual CD-ROM Jukebox, a továbbiakban VCDJ) Samba és Linux segítségével történő készítését tárgyalja. A VCDJ olyan hálózati kiszolgáló, amely nagyszámú CD-ROM tartalmához biztosít hozzáférést úgy, hogy eközben mindössze egyetlen CD-ROM-meghajtóra van szükség. Emellett elérhetővé teszi az ISO 9660 CD-ROM-lenyomatokat is, így egyetlen CD-íróval akár az összes lemez másolata is elkészíthetjük.

Miért jó ez az egész?

Egy CD-váltó általában fájlkiszolgáló (vagy valamilyen célkészülék), amelyhez egy CD-ROM-torony is csatlakozik. Nagyszámú CD-ROM-lemez tartalmának elérését biztosítja a hálózati ügyfelek számára, gyakran SMB/Windows Networking használatával. Haszna abban mutatkozik meg, hogy a felhasználóknak nem kell fizikailag előkeresniük egy-egy lemezt, ha programokat akarnak telepíteni vagy adatokra van szükségük. Ennek a megoldásnak ugyanakkor hátrányai is akadnak.

A megosztható CD-k számát a toronyban található meghajtók száma korlátozza. Ha további lemezeket akarunk elérhetővé tenni, újabb meghajtókat kell beszereznünk. A lemezeknek folyamatosan a meghajtókban kell lenniük, így egyéb célra nem használhatjuk őket. A lemezekről nehéz úgy – különösen rendszerindításra alkalmas – másolatot készíteni, hogy kiveszszük őket a kiszolgálóból, mert ez elérhetetlenné teszi őket a felhasználók számára.

Egy VCDJ segítségével túlléphetünk ezeket az akadályokon. Működése eltérő a megszokott fájlkiszolgálókétól, mivel azokéval ellentétben nem a lemezek tartalmát tárolja, hanem ISO 9660 lenyomatokat. Felállítása után a lenyomatot és a benne tárolt állományokat egyaránt elérhetjük, az eredeti CD-ket pedig külön tárolhatjuk, ahol eltűnésüktől nem kell tartani.

Míg a hagyományos CD-ROM-kiszolgálók teljesítőképességét a CD-meghajtók száma korlátozza, addig a VCDJ teljesítménye a rendelkezésre álló lemezhelytől függ. A merevlemezek viszont jelenleg olcsóbbak, mint a CD-ROM-tornyok, és előnyösebb feltételekkel méretezhetők. Egy 40 GB-os merevlemez egyetlen IDE- vagy SCSI-csatlakozót igényel, viszont 57, egyenként 700 MB-os CD anyagának tárolására alkalmas. Ennek kiváltására 57 CD-meghajtót kellene csatlakoztatni a kiszolgálóhoz, ami a gyakorlatban kivitelezhetetlen.

Munkahelyemen rendkívül értékes segítséget jelent a VCDJ, amikor a rendszeresen frissített program-előfizetések anyagát kell elérhetővé tennünk. Korábban sikertelenül próbáltuk nyomon követni, hogy kinek melyik lemezt adtuk kölcsön, most egyszerűen Windows-tartományi hozzáférésük révén érhetik el őket. Bármely rendszerindításra használható lemezről bármikor készíthetünk másolatot, az eredeti lemezeket pedig elzárva tartjuk.

Az építőelemek

Ha saját Virtual CD-ROM Jukebox építésébe kezdünk, az alábbiakra lesz szükségünk:

- Valamilyen újabb Linux-változatot futtató számítógépre egy CD-meghajtóval. Ezt a meghajtót fogjuk használni az ISO 9660 CD-lenyomatok elkészítésére (a korongokon általában ISO 9660 fájlrendszert használnak, a CD-k képeire tehát ISO 9660 lenyomatokként fogunk hivatkozni).
- Elegendő lemezhelyre a merevlemezen a megosztani kívánt CD-lenyomatok tárolására.
- Hurokeszközre (loopback) az ISO 9660 lenyomatokban tárolt állományok eléréséhez.
- Automounterre az ISO 9660 CD-lenyomatok önműködő befűzéséhez.
- Hálózati megosztásokra kiszolgálására beállított Sambára.

ISO 9660 lenyomatok készítése a lemezekről

Az első feladat: ISO 9660 lenyomatok elkészítése a CD-kről. Gyakorlatilag bármely olyan eszköz használható erre a célra, amellyel korongokat lehet másolni, de a Linux-terjesztések ISO lenyomatjai például az Internetről is letölthetők.

Linuxon legegyszerűbben a `cat` segítségével készíthetünk lenyomatot. Helyezzük be a megfelelő CD-t a meghajtóba! Ellenőrizzük, hogy a */mnt/images* könyvtár létezik-e! Ha a CD-ROM-meghajtó blokkos elérésű eszköze a `hdc`, a lenyomatot az alábbiak szerint készíthetjük el:

```
cat /dev/hdc > /mnt/images/image1.iso
```

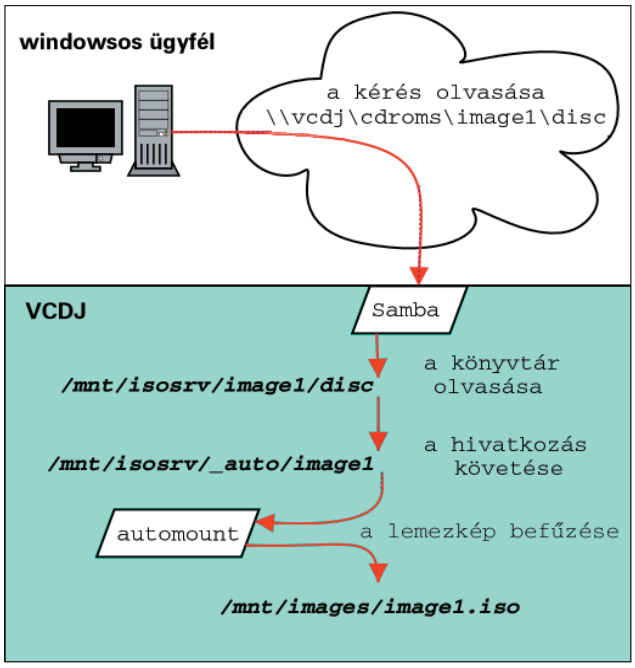
Természetesen a lenyomat ennél beszédesebb nevet is kaphat. A lenyomat elkészítése eltarthat egy kis ideig, és ugyanezt a műveletet kell megismételni az összes szükséges lemezzel. Miután a lenyomatok elkészültek, szeretnénk elérni a tartalmukat. Az általános módszer erre a hurokeszköz használata az alábbiak szerint:

```
mount -t iso9660 -o loop,ro
➥/mnt/images/image1.iso /mnt/isosrv/image1/
```

Ezzel a befűzési paranccsal ISO 9660 fájlrendszerben tárolt adatokat fűzünk be a rendszerbe hurokeszköz használatával. A hurokeszköz a rendszermag egyik ügyes szolgáltatása, segítségével egy fájlt – esetünkben a */mnt/images/image1.iso* nevűt – eszközként kezelhetünk, azaz merevlemezként vagy CD-ROM-meghajtóként. A fenti parancs a lenyomatot csak olvasható módban fűzi be. A CD-lenyomat tartalma a */mnt/isosrv/image1/* könyvtár alatt érhető el.

Az automounter beállítása

Ha a VCDJ számos CD-lenyomatot tartalmaz, nem lehet mindet egyszerre, állandó jelleggel befűzni. Következő lépésünk tehát az automounter beállítása. Az ő feladata lesz, hogy az ISO 9660 lenyomatokat szükség szerint fűzze be a rendszerbe, majd adott idő után – ha már nincs rájuk szükség – leválassza őket. Azért kell ezt megtettünk, mert a hurokeszköz segítségével csak korlátozott számú fájlrendszert lehet egyszerre befűzni.



Hálózati ügyfelek adatelérése megosztott lemezeken

Remélhetőleg nem akarjuk az összes CD-lenyomatot egyszerre használni, tehát az automounter használata nem jelent gondot (az automounter telepítéséről és alapvető beállításairól lásd *Marcel Gagné* Tux Knows It's Nice to Share, Part 4 című cikkét a ➲ http://www.linuxjournal.com/article.php?sid=5298 címen). Elsőként a */etc/auto.master* állományt kell átszerkesztenünk, vagyis az alábbi sort kell hozzáfűznünk:

```
/mnt/isosrv_auto /ect/auto.isosrv  timeout=60
```

Ellenőrizzük, hogy a */mnt/isosrv_auto* könyvtár létezik-e! A változások az automounter újraindításával léptethetők életbe. Hozzuk létre a */etc/auto.isosrv* fájlt, és másoljuk bele az alábbiakat:

```
image1  fstype=iso9660,ro,loop
➥:/mnt/images/image1.iso
```

Minden önműködően befűzendő ISO 9660 CD-lenyomathoz hozzunk létre egy hasonló sort. A VCDJ befűzése után az alábbihoz hasonló üzenetet kell látnunk:

```
automount(pid782) on /mnt/isosrv_auto type
➥autofs (rw,fd=5,pgrp=782,minproto=2,maxproto=3)
```

(A különféle számértékek rendszertől függően mások lehetnek.)
Az automountert *nem* kell újraindítani, ha módosítjuk a */etc/auto.isosrv* fájlt. Eddig az automountert arra utasítottuk, hogyha egy folyamat olyan fájlt vagy könyvtárat próbál elérni, amely a */mnt/isosrv_auto/image1/* könyvtár alatt található, akkor fűzze be az *image1.iso* lenyomatot. Ha bizonyos ideig senki nem használja a könyvtárat, a lenyomat leválasztódik a rendszerről.

A fájlrendszer megtervezése

Van még egy említésre méltó kérdés. Jelenítsük meg a */mnt/isosrv_auto/* könyvtár tartalmát:

```
ls /mnt/isosrv_auto/
```

Ha a közelmúltban senki sem fért hozzá a CD-lenyomatot tartalmához, a könyvtár üresnek fog látszani. Amennyiben viszont kifejezetten a CD anyagára vagyunk kíváncsiak, megjelenik a tartalma.

```
ls /mnt/isosrv_auto/image1
```

Most lépjünk vissza, majd listázzuk ki újra a */mnt/isosrv_auto/* könyvtár tartalmát. Az image1 megjelenik. Az automounter hamarosan újra le fogja választani a lenyomatot, így a könyvtár ismét üresnek fog látszani.
A gond az, hogy a fentiek szerint a felhasználóknak az összes elérni kívánt CD-ROM nevét ismerniük kell. A könyvtárakat nem lehet majd tallózni, ami természetesen elfogadhatatlan. A megoldást egy újabb */mnt/isosrv* nevű könyvtár létrehozása jelenti. Lépjünk be ebbe a könyvtárba, majd hajtsuk végre a következő parancsokat:

```
mkdir image1
cd image1
ln -s ../../isosrv_auto/image1 disc
```

A fentieket minden egyes ISO 9660 lenyomat esetében ismételjük meg.
A */mnt/isosrv* könyvtár tartalmát listázva tehát az összes elérhető lenyomatot megjeleníthetjük, akár befűzte őket az automounter, akár nem.

A Samba beállítása

A Samba telepítésével és kezdeti beállításaival kapcsolatban érdemes átnézni *Marcel Gagné* Tux Knows It's Nice to Share, Part 5 című cikkét
(➲ http://www.linuxjournal.com/article.php?sid=5297).
Ellenőrizzük, hogy megfelelően, azaz az felhasználói szerződéseknek megfelelően védjük-e a CD-ROM-ok tartalmát

Hogyan ellenőrizhetjük le a lenyomatokat?

Időnként az ISO 9660 lenyomatok hibásak, ezért nem árt őket ellenőrizni. Ehhez a lenyomatot közvetlenül az eredeti CD-vel kell összehasonlítanunk. Fűzzük be a CD-ROM-ot és az ISO 9660 lenyomatot is (a hurokeszköz segítségével az *ISO 9660 lenyomatok készítése a lemezekről* című részben leírtak szerint).
Futtassuk az alábbi parancsfájlt – feltéve, hogy az eredeti CD-ROM a */mnt/cdrom* könyvtárba, az ISO lenyomat pedig a */mnt/isosrv/image1* könyvtárba van befűzve:

```
#1/bin/sh
cd /mnt/cdrom && find .  type f | sort |
➥sed  e  s/^/ /  e  s/$/ /  | cargs cksum
➥> CDROM.sums
cd /mnt/isosrv/image1 && find .  type f |
➥sort | sed  e  s/^/ /  e
 s/$/ /  | xargs cksum > IMG.sums
echo
diff CDROM.sums IMG.sums | more
```

Ha az echo parancs után megjelenik valami, a lenyomat hibás (illetve előfordulhat, hogy a CD-ROM-meghajtó hibásodott meg).

A fájlrendszer felépítése

Az automountert úgy állítottuk be, hogy az *image1.iso* fájlt a */mnt/isosrv_auto/* könyvtárba fűzze be. Ha egy folyamat a */mnt/isosrv_auto/image1/* könyvtárat próbálja elérni, az automounter befűzi az ISO 9660 lenyomatot. Amennyiben azonban egy folyamat a */mnt/isosrv_auto/* könyvtárat éri el, az automounter nem tesz semmit.
Ennek kiküszöbölésére egy másik */mnt/isosrv/* nevű könyvtárat készítünk. Ebben a könyvtárban a */mnt/isosrv_auto/image1/* könyvtárról */mnt/isosrv/image1/* néven közvetett hivatkozást hozhatunk létre. A közvetett hivatkozás mindig látható lesz, ha egy folyamat a */mnt/isosrv/* könyvtárat olvassa. Amikor a folyamat a */mnt/isosrv/image1* nevű közvetett hivatkozást próbálja elérni, a rendszermag az automounter segítségével az *image1.iso* fájlt befűzi a */mnt/isosrv_auto/* könyvtárba. A közvetett hivatkozás (*/mnt/isosrv/image1*) ezután a most befűzött eredetire fog mutatni.
A megoldás azonban nem tökéletes: ha egy folyamat a */mnt/isosrv/* könyvtárat éri el, amely az ISO 9660 fájlokra mutató közvetett hivatkozásokat tartalmazza, valójában az összes metaadatot is el fogja érni (például a módosítás dátumát, a fájlok típusát és a hozzáférési engedélyeket). Ezért az automounter kénytelen lesz az *összes* olyan ISO 9660 lenyomatot befűzni, amelyre ebből a könyvtárból hivatkozunk (jó példa erre az `ls -la` parancs). Ezzel azonban nem tudjuk az automountert az eredeti célnak megfelelően használni.
A megoldás: a */mnt/isosrv* könyvtáron belül az összes ISO 9660 lenyomathoz külön könyvtárakat kell készíteni, majd ezeken a könyvtárakon belül kell a */mnt/isosrv_auto/* könyvtárban található befűzési pontokhoz a közvetett hivatkozásokat létrehozni.

– természetesen amennyiben minden anyag nyílt forrású, ezzel nem lesz gondunk.
Ne feledjük, célunk az, hogy az ISO 9660 lenyomatokat és tartalmukat Sambán keresztül egyszerre tegyük elérhetővé. Ehhez a */etc/smb.conf* (vagy a */etc/samba/smb.conf*) fájlt is módosítanunk kell, és hozzá kell fűznünk a következő sorokat:

```
[isoimages]
       comment = iso9660 CD ROM lenyomatok
       path = /mnt/images/
[cdrom]
       comment = a CD-lemezek tartalma
       path = /mnt/isosrv/
```

Indítsuk újra a Sambát. Üljünk át egy windowsos ügyfélhez (vagy bármely géphez, amely SMB-ügyfélként használható), és tallózzunk a VCDJ tartalmára. Két új megosztást fogunk látni: az ***isoimages*** és a ***cdroms*** nevűt. Az ***isoimages*** könyvtárban lesznek az ISO 9660 lenyomatok, a ***cdroms***-megosztásban pedig a lenyomatok tartalma lesz elérhető. Ha befűzzük a VCDJ-t, a következőhöz hasonló üzenetet láthatunk:

```
/mnt/images/image1.iso on
➥/mnt/isosrv_auto/image1
➥type iso9660         (ro,loop=/dev/loop0)
```

Összegzés

Az első ábrán láthatjuk, mi történik, amikor egy hálózati ügyfél a ***cdroms*** megosztásban található adatokat próbálja elérni. Megjegyzem, ha az ügyfél az ISO 9660 lenyomatot éri el, a Samba közvetlenül a */**mnt**/**images**/* könyvtárból szolgálja ki, megkerülve a közvetett hivatkozásokat és az automountert. Újabb ISO 9660 lenyomatokat adva a rendszerhez egyre inkább értékelni fogjuk a VCDJ szolgáltatásait. Szárnyalásunknak egyedül a rendelkezésünkre álló lemezhely szabhat határt, a tárhely bővítésének költsége és munkaigénye azonban messze alatta marad az újabb és újabb CD-ROM-meghajtók beszerelésének. Most már minden munkatársunk elérheti az összes CD-ROM-ot akár a munkahelyéről, akár otthonról, és soha többé nem kell azzal foglalkoznunk, hogy kinek melyik CD-t adtuk kölcsön.

Linux Journal 2002. április, 96. szám

Jeremy Impson (jeremy.impson@lmco.com) vezető tudományos kutató a Lockheed Martin Systems Integration nevű cégnél, a New York állambeli Owegóban.

Keresés a HOGYAN-okban

Debian Woodyt használok és a */usr/doc/HOWTO/en-txt* könyvtáramban a HOGYAN-okat mind tömörítve tárolom. Így azonban nem tudok bennük keresni. A legutóbb arra voltam kíváncsi, hogy melyik HOGYAN-ban esik szó az ncurses-ről. Szerencsére a feladatot tökéletesen megoldja a `zgrep` parancs. Éppen héjprogramozással foglalkoztam, és elgondolkoztam a parancs működésén. Érdekességképpen írtam a feladatra egy kis bash-programot. Talán másoknak is a hasznára lehet. A parancsfájl a pillanatnyi könyvtár minden gzippel tömörített állományában a megadott kifejezésre keres, és a találatokat a sorszámmal együtt kiírja:

```
#!/bin/bash

if [[ $1 == "" ]]; then
     echo "HasznÆlata: $0 {kifejezØs}";
     exit;
fi

for FILE in 'ls -1'; do
     if [[ 'file $FILE | cut -d"
     ➥" -f2' != "gzip" ]]; then
       continue;
     fi
     LIST="";
     for LINE in 'zcat $FILE | grep -n
     ➥"$1" | cut -d: -f1'; do
         LIST="$LIST $LINE ;
     done
     if [[ $LIST != "" ]]; then
         echo "$FILE:$LIST";
     fi
done
```

Fülöp Balázs

Zope-termékek

Reuven folytatja a Zope felfedezését. E hónapban megtudhatjuk, hogy a Zope-termékek miképpen teszik lehetővé a szolgáltatások újrahasznosítását.

Múlt hónapban egy pillantást vetettünk a nyílt forráskódú Zope alkalmazáskiszolgálóra. Pontosabban láthattuk, hogyan használhatjuk a Zope DTML (Dynamic Template Markup Language) tagjait egyszerű dinamikus oldalak előállítására, illetve hogyan kezelhetjük weblapunkat mindössze egyetlen böngésző segítségével.
Csakhogy aki már dolgozott a DTML-lel, tudja, hogy hamar elveszti a varázsát, amint valami összetettebbet próbálunk meg vele alkotni. A DTML akkor a leghatékonyabb, ha csak mértékkel használjuk, vagy ahol használata magától értetődik; ha olyan DTML-oldalakat készítünk, ahol fél tucat egymásba ágyazott feltételes (`<dtml-if>`) tagot kell alkalmaznunk, oldalunk hamar olvashatatlanná és nehezen karbantarthatóvá válik, nem beszélve arról, hogy nemigen lesz modulárisnak nevezhető.
A DTML másik nagy gondja, hogy központi elhelyezés helyett különálló dokumentumokban tárolódik. Így ha több helyen is újra akarunk egy szolgáltatást hasznosítani, a DTML-eljárásokat és dokumentumokat le kell másolnunk. Ez azt jelenti, hogyha új szolgáltatást akarunk bevinni avagy egy régit megváltoztatni, akkor az összes másolaton végig kell mennünk és el kell végeznünk a szükséges módosításokat.
A nehézségre a Zope-termék jelenti a megoldást. Minden Zope-termék tulajdonképpen egy objektumosztály (vagy osztálykészlet), amelyet a weblapunkon akárhány példányban létrehozhatunk.
Ebben a hónapban a Zope rugalmasságának magvát alkotó Zope-termékkel fogunk foglalkozni. A telepítést követően néhány meglévő terméket próbálunk ki, majd Pythonban a saját termékünket is elkészítjük.

Mire képes a termék?

A Zope-termék tulajdonképpen egy kódból, grafikából és DTML-ből álló csomag, amely egy adott újrahasznosítható szolgáltatást tesz elérhetővé. Ha például egy olyan egyszerű lapot szeretnénk készíteni, amely kiírja a pillanatnyi időt, a következő DHTML-dokumentumot kell létrehoznunk:

```
<p>A pontos idı: <dtml-var name="ZopeTime"
➥fmt="fCommon">.</p>
```

De mi a helyzet, ha lapunkat ki szeretnénk bővíteni, és az időjárás-előrejelzést is meg akarjuk jeleníteni, amit egy másik kiszolgálóról töltünk le HTTP-n keresztül? A DTML nem jó megoldás; még ha segítségével el is tudnánk készíteni a saját szolgáltatásunkat, az eredményt biztos, hogy nehéz lenne kezelni, és legalább ilyen ronda ügy lenne megírni is. Minthogy azonban a Zope-termékek Pythonban készültek, tetszés szerinti Python-modult használhatnak, és kimenetüket HTML-ben vagy bármilyen együttműködő formátumban jelentethetik meg.
Mivel minden termék önálló egésznek tekinthető, egységenként feltelepíthetjük vagy eltávolíthatjuk őket, akkor is, ha számos osztályt használnak.
Természetesen ez nem azt jelenti, hogy minden termék önmagában működik; hiszen az egyik termék nyugodtan használhatja a másik által nyújtott szolgáltatásokat.
A termékek és a DTML-dokumentumok mellett a Zope további két lehetőséget nyújt a dinamikus tartalom megjelenítésére: a Python-parancsfájlok (amelyeket szintén egy termék valósít meg) lehetővé teszik, hogy kisméretű Python-programokat írjunk és használjunk a Zope-ban. A másik lehetőséget választva a termékeket a ZClasses nevű rendszeren keresztül is létrehozhatjuk, szerkeszthetjük és használhatjuk. A ZClasses lehetővé teszi, hogy az új termékeket (és a hozzájuk tartozó osztályokat) mindössze a böngésző és a DTML segítségével rakjuk össze.
E négy lehetőség igen nagy rugalmasságot biztosít, de a kezdő Perl-programozóknak néha nehéz eldöntetni, melyiket is válasszák. Beehive ***The Book of Zope*** című művében azt ajánlja, hogy a projekthez eleinte ZClassest használjunk, majd mikor már mindenki megegyezett a tervben, a kódot a teljes értékű termékek szintjére tegyük át. Minél összetettebb feladatokat oldunk meg, annál valószínűbb, hogy a DTML és Python-parancsfájlok helyett a Zope-termékeket választjuk.

A termékek kezelése

A Zope-termékek csaknem minden szempontból jól kezelhetők a ***Zope intézőben*** (management screen), amit Zope-kiszolgálónkon a ***/manage*** URL alatt érhetünk el. A termék intézőjének elővarázslásához a bal oldali keretben kattintsunk a ***control panel*** hivatkozásra, majd az így feljövő ***Zope vezérlőpult*** fő keretében válasszuk a ***Product Management*** (termékkezelés) hivatkozást. Itt a Zope-termékek listáját találjuk, valamint a képernyő tetején egy ***Add product*** (termék felvitele) gombot. A Weben keresztül is szerkeszthető termékeket (ilyen például a ZClasses, amelyet az előbb ismertettünk röviden) nyitott doboz jelöli, a beépített Zope-termékeket pedig zárt. A zárt doboz egyszerűen annyit jelent, hogy a terméket magát a Weben keresztül nem módosíthatjuk. Természetesen a legtöbb termék megengedi, hogy webes felületen keresztül egy vagy több tulajdonságát testreszabjuk, de a termék mindig változatlan marad, hacsak a forráskódot meg nem változtatjuk.
Minden termék valójában a Zope telepítési könyvtára alól nyíló ***lib/python/Products***-ban található alkönyvtár. A Sessions termék a ***lib/python/Products/Sessions***, a Transience termék pedig a ***lib/python/Products/Transience*** könyvtárban helyezkedik el (rendszeremen a ***/usr/local/zope/*** könyvtárba helyeztem a Zope-ot, így nálam a Sessions a ***/usr/local/zope/lib/python/Products/Sessions/*** könyvtárban helyezkedik el). A termékek könyvtára Python-kódot, szövegfájlokat és könyvtárakat tartalmaz:

- ***__init__.py***: ez az, amit a Zope modulbetöltéskor megkeres és végrehajt. Az ***__init__.py*** alapérték beállító eljárása (initialize method) többek között meghívja a

`context.registerClass` osztályt, amely (mint neve is sugallja) tudatja a Zope-pal, hogy termékünk egyáltalán létezik. Ismerteti, milyen szöveget kell majd a */manage* képernyőhozzáadás menüjében megjeleníteni (ezt a `meta_type` kapcsolóval oldja meg), és hogyan készíthetünk a termékből egy új példányt, ha az ***Add*** gombot megnyomják (ez a `constructors` kapcsolóval adható meg).

- ***README.txt***: miként neve is mutatja, ez a termékhez tartozó README (OLVASSEL) állomány. A ***Vezérlőpulton*** a termék nevére kattintva többek közt egy README táblát is felhoz. Itt a fájlrendszerben való keresgélés nélkül olvasgathatjuk a ***README.txt*** tartalmát. Ha a termék könyvtára nem tartalmaz ***README.txt*** nevű fájlt, a README tábla sem jelenik meg a képernyő tetején.
- ***version.txt***: ez a fájl tartalmazza a termék kötőjelekkel (-) elválasztott pillanatnyi változatszámát. A Foo termék 1.2.3-as változata például a következő tartalmú ***version.txt*** fájllal rendelkezik: Foo-1-2-3. A vezérlőpult ezt az adatot fogja megjeleníteni.
- ***Help*** fájlok: a termék egy help (súgó) könyvtárat tartalmazhat, ahol azok a fájlok találhatók, amelyek tartalmát a ***help*** hivatkozásra kattintva a Zope megjelenítheti. A Súgófájlok gyakran strukturált szöveget tartalmaznak, amely a Perl POD dokumentumrendszeréhez hasonló szellemiségű, egyszerűségre törekvő formázási rendszer. Strukturált szöveg egyszerű szövegszerkesztővel is könnyen elkészíthető, és ugyanilyen könnyen olvasható a `less`-hez hasonló szabványos Linux-eszközökkel.

A Zope a pillanatnyi termékek listáját csak indításkor nézi végig. Ez azt jelenti, hogyha új terméket telepítettünk, a vezérlőpultról a Zope-kiszolgálót újra kell indítanunk.

A termékek telepítése

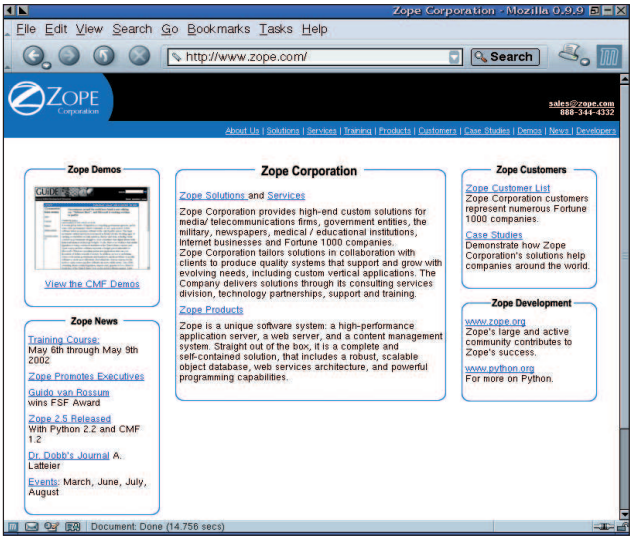
Most hogy láttuk, mit tartalmazhat egy átlagos termék, ideje feltelepítenünk egyet! A Zope honlapjáról töltsük le, a ***lib/python/Products*** könyvtárba csomagoljuk ki, majd a Zope-ot indítsuk újra. Ha minden jól ment, frissen telepített termékünk meg kell jelenjen a ***Vezérlőpult*** képernyőjén. Ráadásul a termék új példányait a webszerkezet tetszőleges helyén máris létrehozhatjuk. Példaképp a Zope Squishdot termékével készítsünk egy Slashdot-másolatot. Első lépés a Squishdot-másolat letöltése a ➲ http://www.zope.org/Products címről. A Squishdot-csomagot több társával egyetemben a Feedback csoport alatt találjuk, és valószínűleg valahol az elsők közt kell keresnünk a termékek listájában. Kattintsunk a hivatkozásra, ami a Squishdot letölthető változatához visz bennünket – e sorok írásakor a legújabb változat az 1.3.0-s. Figyeljük meg, hogy egy viszonylag összetett termék is milyen kis méretű; a Squishdot általam letöltött változata ugyanis alig volt több, mint 256 KB.
A Squishdotot telepítéséhez ki kell csomagolnunk a ***lib/python/Products*** könyvtárba. Feltételezve, hogy az újonnan letöltött fájlokat a */downloads* könyvtárba tesszük, a Squishdotot a következőképpen csomagolhatjuk ki:

```
# Itt Æll tsuk be sajÆt Zope k nyvtÆrunkat
export ZOPE=/usr/local/zope

# vÆltsunk a termØkk nyvtÆrba
cd $ZOPE/lib/python/Products

# Squishdot kicsomagolÆsa az aktuÆlis
# k nyvtÆrba
tar -zxvf /downloads/Squishdot-1-3-0.tar.gz
```

A régebbi Zope-termékeket nem a ***lib/python/Products***-ban állva, hanem a Zope saját könyvtárból kell kicsomagolni. Úgy tűnik, sajnos nemigen van olyan módszer, amivel kicsomagolás nélkül egyértelműen el lehetne dönteni, miként csomagolták össze a terméket, így nem marad más hátra, mint a kicsomagolás:

```
tar -ztvf /downloads/ProductName.tar.gz
```

Ha a fájlnevek ***lib/python/Products*** útvonallal kezdődnek, akkor kicsomagolás előtt a ***$ZOPE/lib/python/Products*** helyett a ***$ZOPE*** könyvtárba kell váltanunk.
A Squishdot telepítéséhez egyetlen dolgot kell csak megtennünk: csomagoljuk ki a terméket. Mivel azonban a Zope csak induláskor nézi meg a termékeket, a kiszolgálót még az előtt újra kell indítanunk, mielőtt a rendszeren Squishdot-példányokat készíthetnénk. Erre a legjobb módszer, ha a ***Vezérlőpulton*** a ***Restart*** (újraindítás) gombra kattintunk. Ne essünk pánikba, ha

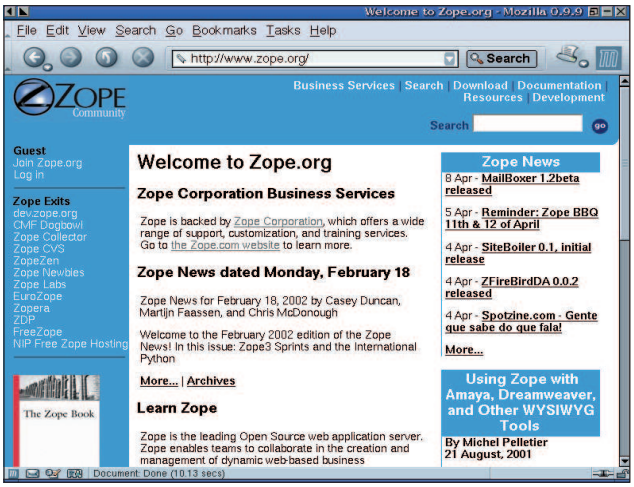


a ***Restartra*** kattintás után a böngésző arra figyelmeztet, hogy a kiszolgáló nem fut, vagy ha valami fura kinézetű Python kivételnyomelemzés (exception backtrace) jelenik meg. Egyszerűen várjunk néhány másodpercig, majd a bal oldali keretben megint kattintsunk a ***Vezérlőpult*** hivatkozásra, és már működnie is kell. Termékünk sikeres felvételét ellenőrizendő a ***Vezérlőpulton*** visszatérhetünk a ***Termékkezelő*** (Product Management) lapra. Amennyiben a frissen feltelepített termék (jelen esetben a Squishdot) nem jelent meg a listában, ellenőrizzük, hogy jól sikerült-e kicsomagolnunk, illetve a jogosultságok lehetővé teszik-e, hogy a Zope-felhasználó elérje a termék állományait.

A termék használata

Ettől kezdve lehetőségünk nyílik rá, hogy a Zope-kiszolgáló saját (/) könyvtárába mozgatása által új Squishdot-honlapot hozzunk létre. A listából ki kell választanunk a Squishdot-lapot, majd kattintsunk az ***Add*** (hozzáadás) gombra. Ez meghívja `context.registerClass` constructors kapcsolójában megnevezett eljárásokat, amelyet a Squishdot ***__init__.py*** programjának beállítófüggvénye indít el.
Máris elkezdtük Squishdot-lapunk létrehozását. Csakhogy a figyelmeztető levelek elküldéséhez a Squishdot a Zope MailHost objektumát is használja (amely az SMTP-kiszolgálónak felel meg). Ha még nem készítettük el és nem határoztuk meg a `MailHost`-ot, a Squishdot beállítóképernyője emlékeztet minket erre.

Amikor a Squishdot a `MailHost` után kutat, a keresést a pillanatnyi könyvtárban kezdi. Ha nem talál `MailHost`-objektumot, a könyvtárfán felfelé folytatja a keresést , amíg el nem éri a /-t, vagy nem talál egy `MailHost`-objektumot. Bár mindez egyszerű dolognak látszik, nagyon jól megvilágítja a Zope alapfilozófiáját jellemző szerzeményezés elgondolást. Egyben azt is jelenti, hogy a különböző Squishdot-honlapok a leveleket különféle SMTP-kiszolgálókon keresztül küldhetik,



egyszerűen úgy, hogy egynél több `MailHost`-objektumot készítünk. Azaz egy teljes körben alapértelmezett `MailHost`-ot adhatunk meg a gyökérben (/), amit aztán az alkönyvtárakba helyezett `MailHost`-objektumokkal szükség szerint felülbírálunk. A szerzeményezéses elgondolás átszövi a Zope-ot, ami abban nyilvánul meg, hogy helyileg szinte bármit megadhatunk és újradefiniálhatunk – a `MailHost`-okat, a felhasználókat, de akár a fejléceket és a stíluslapokat is.
Jelen esetben a `MailHost`-ot a terméklistából kiválasztva, majd az ***Add***-ra kattintva a / könyvtárban egy `MailHost`-példányt hozunk létre. Tekintve, hogy a `MailHost` objektum az SMTP-kiszolgálónak felel meg, ezen objektum beállítása meglehetősen magától értetődő, mindössze arra van szükség, hogy a Zope-kiszolgáló SMTP-kiszolgálóját megadjuk. Minthogy a legtöbb linuxos gép saját levelezőkiszolgálót futtat, a `localhost` valószínűleg megfelelő érték lesz.
A kötelező ID (azonosító) mező célja a `MailHost` egyedi azonosítása a működő könyvtárban, hiszen a Zope az URL-eken belül az objektumok egyértelmű azonosítására ID-ket használ. Miként a fájlnév egyedi azonosító a könyvtárban, a Zope objektum-ID is egyedi azonosító egy könyvtárban vagy egy másik objektumban. A választható ***Title*** (cím) mező inkább az emberek, semmint az alatta fekvő Zope-kiszolgáló kedvéért van itt; ha megadjuk, ez az objektumcím fog a Zope-kiszolgáló csatolófelületén megjelenni.
Miután `MailHost` objektumunkat létrehoztuk, visszatérünk a Zope / gyökérhez tartozó fő kezelőfelületéhez. Láthatjuk, hogy új `MailHost` objektumunk (amit egy kis borítékikon jelképez) az általunk megadott névvel együtt megjelent az objektumok listájában.
Készen állunk, hogy létrehozzuk Squishdot-helyünket. A listából kiválasztva, majd a jobb felső sarokban található ***Add*** gombot lenyomva vegyük fel az új Squishdot-hely objektumot, válasszunk egy ID-t (azaz URL-elérési utat), tetszés szerinti címet, levélgazdát (mailhost), majd Squishdot-lapunkhoz válasszunk ki még pár alapvető kapcsolót. Én például ID-ként az ***atf*** szó mellett döntöttem, az egyéb beállítási lehetőségeket pedig az alapértelmezett értéken hagytam.
Ha Squishdot-lapomat meg szeretném nézni, utasítom a böngészőt, hogy jelenítse meg a ➲ http://localhost:8080/atf/ címet. A Zope észleli a ***/atf*** kérelmet és látja, hogy egy Squishdot-objektumra hivatkozunk. Ezután a Zope megkéri az objektumot, hogy jelenítse meg magát. Nagy valószínűséggel egy bemutatkozó képernyőt fogunk látni, ami erősen hasonlít a Slashdotra, de a Zope hajtja meg.
Annyi Squishdot-lapot készítünk, amennyit csak akarunk, szem előtt tartva, hogy minden honlapnak saját egyedi ID-vel kell rendelkeznie. Így készíthetünk egy moderált lapot, egy moderálatlan lapot és egy másik, belső lapot a szervezet saját használatára – mindegyik saját URL-lel és védett saját felhasználóhalmazzal, valamint csoportokkal rendelkezik.
A Squishdot-honlap (site) módosításához a módosítandó objektum nevéhez egyszerűen csapjuk hozzá a ***/manage*** karaktereket, például: http://localhost:8080/atf/manage. E sor a Squishdot-lapunkhoz rendelt Zope-kezelőfelületet hozza be. A képernyő tetején található táblákat használva csaknem minden olyan kapcsolóját átállíthatjuk, amelynek köze van a Squishdothoz, a moderátorszabályoktól kezdve a honlap nevének színéig.

Összefoglalás

Ebben a hónapban a Zope-termékekkel ismerkedtünk meg: láttuk, hogyan tölthetünk le, telepíthetünk és állíthatunk be új termékeket a rendszerünkön. Bár a termékek természetüktől fogva az egyszerű DTML-lapoknál jóval összetettebbek, központosított kódjuk és nagyobb rugalmasságuk folytán komoly feladatokra sokkal alkalmasabbak, mint a DTML.
A következő hónapban azt vizsgáljuk meg, miképpen írhatunk saját Zope-termékeket Python- és DTML keverék használatával.

Linux Journal 2002. március, 95. szám

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 forrású Zope-terméket a Zope Corporation (Digital Creations) készíti és végzi a karbantartását. Több adatot a Zope honlapján érhető el ➲ http://www.zope.com

A Zope-közösség fő fejlesztési oldala a ➲ http://www.zope.org. Erről az oldalról letölthetjük a program legújabb változatát, hozzászólhatunk a vitákhoz, letölthetjük a Zope-terméket, tanulhatunk a Zope-ról és részesei lehetünk a Zope-közösségnek.

A Squishdot termék a saját oldalán érhető el ➲ http://www.squisdot.org

Többet tudhatunk meg a Python nyelvről a ➲ http://www.python.org címen.

Működjünk együtt!

Marcel újra megvizsgál néhány régi asztali környezetet, és kifürkészi, milyen jövőt is hozhat a 3D grafikus felület.

Igen, François. Úgy néz ki, mintha az 1980-as évekből származna. Mert így is van. Igazad van, mon ami. Igen, még mindig a Linux-rendszeremet futtatom, de a hatás rém nosztalgikus, non? Számos példa szerepel a mai menünkön, és azt hiszem, a vendégeink el lesznek ragadtatva. Ó, már itt is vannak!

Bonjour, mes amis, Isten hozott titeket Chez Marcelnél, a Linux világának legkiválóbb borospincéjében és kitűnő Linux-konyhájában. Ínyetek bizsereg már a várakozástól, nem igaz? Engedjétek, hogy hűséges pincérem az asztalhoz vezessen titeket, és a menüre is vetünk rögvest egy pillantást. François, kérlek, hozd fel a pincéből az 1989-es Pessac-Léognant, mert tökéletesen illeni fog a most következő fogásokhoz.

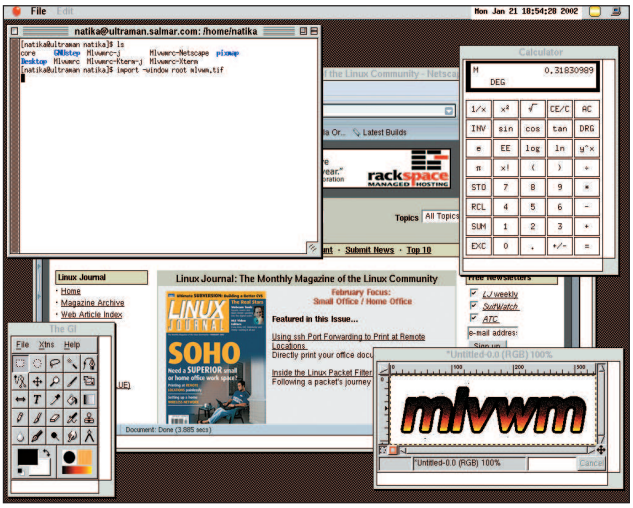
Meghatározás szerint az együttműködő képesség azt jelenti, hogy egy műszaki megoldás jól együtt tud dolgozni egy másik módszerrel, vagy egy program egy másik programmal. Elmondom nektek, mes amis, hogy ez a dolgok gépi oldalára vonatkozik, ami arra késztet, hogy bánatomat egy jó adag karamellikőrbe fojtsam. Ha meggondolom, ezt anélkül is megtehetem, hogy tekintettel lennék a dolgok kimenetelére, non?

Miről is beszéltem, mes amis? Á, oui. Mi a helyzet a program és a géppel dolgozó ember kapcsolatával? A Linuxszal való munkát az teszi nagyszerűvé, hogy a választásnak olyan lehetőségét kínálja, amit egyetlen másik rendszer sem. Ez a választási lehetőség olyan rendkívüli, hogy Linux-rendszerünknek akár egy olyan korábbi „operációs rendszer”-arcát és -légkörét is választhatjuk, amit ismertünk és kényelmesen tudunk vele dolgozni. Végül is a belső számít, non?

Otthoni gépünkön alighanem a csillogó új Gnome vagy KDE asztalt használjuk, mégis vágyakozva gondolunk vissza fiatalságunk asztali rendszerére, legyen az akár Amiga, akár egy öreg Macintosh, akár az a különleges, mindenütt megtalálható asztali felület 1995 tájáról. Semmilyen körülmények között nem cserélnénk el a Linux asztal hatékonyságát, a megjelenés azonban más dolog.

Amikor a Gnome vagy KDE asztalt elindítjuk, tulajdonképpen ablakkezelőt futtatunk az X Window rendszer (vagy egyszerűen X) grafikus felülete felett. A legtöbb Linux-rendszeren ez az Xfree86 kiszolgáló. Amennyiben rendszerünk szöveges bejelentkezéssel indul és a `startx` paranccsal elindítjuk az X-et, már készen is állunk a kísérletezésre.

Ha rendszerünk grafikus beléptetéssel kezd, érdemes visszatérnünk a szöveges bejelentkezéshez, hogy kipróbálhassuk a következő recepteket. Jelentkezzünk ki asztalunkból, váltsunk át az egyik virtuális konzolunkhoz (a Ctrl-Alt-Fn billentyűegyüttessel, ahol az Fn egy funkcióbillentyűt takar 2-től 6-ig), és rendszergazdaként jelentkezzünk be. Adjuk ki az `init 3` parancsot, mire a grafikus beléptetéskezelő befejezi működését. (Számos más módszer is létezik erre.) Esetleg a grafikus beléptető maga is rendelkezik egy a karakteres módba való átváltáshoz szükséges lehetőséggel. Az asztalokat elindító főprogram a `xinit`. A `startx` program csak egy parancsfájl, ami a `xinit`-et hívja (néhány kapcsolóval és



1. kép Az MLVWM működés közben

értékkel). Még a KDE ablakkezelőt is elindíthatjuk vele az alábbi példa szerint:

```
xinit /usr/bin/startkde
```

Ha ezt a módszert használjuk, bizonyosodjunk meg arról, hogy indításkor a teljes útvonalat használjuk. Egy másik lehetőség, ha a saját felhasználói könyvtárunkban egy ***.xinitrc*** fájlt hozzunk létre benne egyetlen sorral:

```
exec startkde
```

Ebben az esetben ablakkezelőnket a hagyományos `startx` paranccsal indíthatjuk. És amiről most beszélni fogunk, maguk a klasszikusok, mes amis, és ez az 1989-es vörös is a leghatározottabban klasszikus. François, kérlek, tölts a vendégeinknek! Amíg élvezettel szürcsölgetitek italotokat, hadd mutassam meg mai első asztali klasszikusunkat: ***Takashi Hasegawa*** MLVWM programját, amely nevét a Macintosh-Like Virtual Window Manager (Macintosh-szerű látszólagos ablakkezelő) rövidítéseként nyerte. Honlapjának meglátogatásával kezdve töltsük le a legfrissebb forrást, és fordítsuk le:

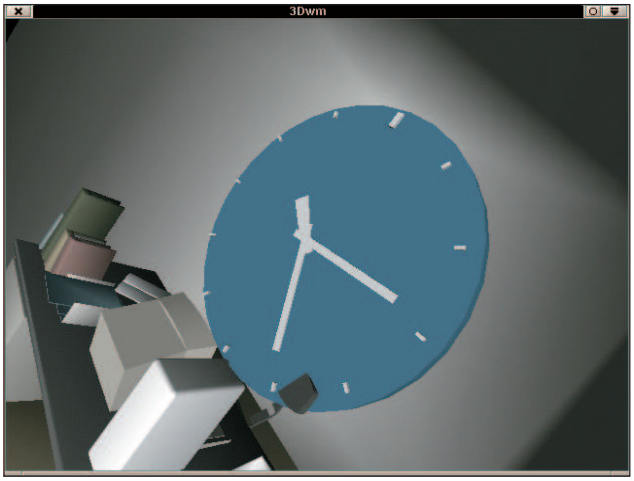
```
tar -xzvf mlvwm091.tar.gz
cd mlvwm091
xmkmf -a
make
make install
```

Ha már itt vagyunk, esetleg érdemes a miniikonkészletet (ami valójában az FVWM ablakkezelővel kerül forgalomba) is letölteni és kicsomagolni:

```
cd /home/natika/pixmaps
tar -xzvf mini-icons.tar.gz
```

Mielőtt elindítanánk új (vagy inkább régi) Macintosh-kinézetű asztalunkat, először a ***.mlvwmrc*** fájlt kell létrehoznunk. Az mlvwm fordításának könyvtárában egy ***sample_rc*** könyvtárat

2. kép Ki mondta, hogy az Amiga halott?



3. kép A 3dwmmel belépünk a harmadik dimenzióba

találunk. Az összes itt található fájlt egyszerűen másoljuk át felhasználói könyvtárunkba, majd az ***mlvwmrc*** fájlt nevezzük át. Folytassuk barátságos felhasználónkkal, natikával:

```
cd /home/natika
cp /pathto/mlvwm091/sample_rc/* .
mv mlvwmrc .mlvwmrc
```

Majdnem készen vagyunk. Az alapértelmezett beállítófájl jó hely a kezdéshez, de a `pixmap`-ekhez vezető útvonal szerkesztésére is szükségünk lesz. Nézzük a következő sort, és változtassuk meg úgy, hogy az elérési útvonal a saját rendszerikonokra mutasson (vagy az imént telepített miniikonokra):

```
IconPath /usr/local/include/X11/pixmaps:
➥/home2/tak/bin/pixmap
```

Most a `xinit /usr/bin/X11/mlvwm` paranccsal indítsuk el. Az *1. kép* működés közben mutatja az MLVWM-et. Mindez újra a miénk. Na milyen, mes amis?
Azok számára, akiknek barátságos felületre volt égető szükségük, az Amiga mindig a rendelkezésére állt. Múlt időt használtam volna? Mais non, ha nem is maga az Amiga, de a lelke továbbél ***Marcus Comstedt*** AMIWM programjában. Kezdjük honlapjának meglátogatásával és a forráskód letöltésével. Ezután, mint korábban oly sokszor, fordítsuk le:

```
tar -xzvf amiwm0.20pl48.tar.gz
cd amiwm0.20pl48
./configure
make
make install
```

Olvassuk el a mellékelt README-t, ahol a beállítófájl lehetőségeit találjuk. Ezek a futásidejű beállítások vagy a helyi ***.amiwmrc*** fájlból, vagy az átfogó ***system.amiwmrc*** állományból érkezhetnek. Pillantsunk a *2. képre*, hogy lássuk, miként fest az AMIWM felülete.
Ezek a különféle asztalok lehetőséget adnak arra, hogy a múlt ízeiből kóstolót kapjunk. A KDE és a Gnome a jelen. De mi a helyzet a jövővel-övel-övel? Hallotta valamelyikőtök a visszhangot? Bárhogyan is, ***Niklas Elmqvist***, ***Robert Karlsson***, ***Steve Houston***, ***Antony Suter*** és társaik azon dolgoznak, hogy képet alkothassunk a jövőről. Lépjünk be a 3dwm-be, ebbe a háromdimenziós asztali programba, ami kiszakít bennünket az ablakkezelők unalmas síkságából, és elvisz a mélység világába. Igazából a 3dwm csapat az szeretné, ha elfeledkeznénk a program nevének „wm” (window manager – ablakkezelő) részéről, és ablakkezelő helyett úgy tekintenénk a 3dwm-re, mint háromdimenziós felhasználói környezeteket fejlesztő eszközre.
Pillantsunk a *3. képre* egy ízelítőért, hogy megtudjuk, mi is következik most. Készen álltok, hogy belépjünk a harmadik dimenzióba? Akkor indulhatunk!
Mivel egy 3D-alkalmazásról van szó, szükségünk lesz a Mesa vagy az OpenGL programozói könyvtárakra, valamint a projekt XML részeihez az ***expat*** könyvtárakra, de az SDL könyvtárakat sem nélkülözhetjük. Az SDL-csomag (amely a ➲ http://www.libsdl.org címről tölthető le) változatszámára oda kell figyelnünk, mert csak az 1.2 felettiek fognak megfelelően működni. Ez indításként kicsit soknak tűnik, szerencsére az újabb Linux-rendszercsomagok nagy része ezeket az összetevőket már telepítve tartalmazza. A 3dwm lefordításának van még néhány további feltétele. A legfontosabb ezek közül az OmniORB csomag, amely az AT&T's UK terméke – nekik köszönhetjük a VNC-t is (ezt a kitűnő hálózatfelügyeleti eszközt). Kicsit szokatlan, amit most fogok javasolni. Az esetek nagy részében könnyű szívvel tudom ajánlani a forráskód lefordítását, de ebben az esetben talán mégis a bináris állomány letöltése a jobb megoldás. Mivel a függőségek nem kerülhetők meg, azok, akik rpm-alapú rendszert használnak, megfontolhatják az rpm-forráskód letöltését és fordítását a saját rendszerükhöz történő jobb illeszkedés érdekében. Én is ezt tettem, azaz újrafordítottam az rpm-csomagot:

```
rpm --rebuild omniORB-3.0.4-0.src.rpm
```

Ez eltarthat egy darabig. Ha elkészült, a fordítás könyvtárában (ami rendszerenként más és más lehet) lennie kell egy omniORB és egy omniORB-devel csomagnak. Az én Red Hat próbarend-

szeremen a kész rpm-csomagok a */usr/src/redhat/RPMS/i386* könyvtárba kerültek. A Debian-felhasználók is találhatnak a rendszerükhöz előre elkészített csomagokat.
Ezután ezt a lépést későbbre hagyva a 3dwm fordításával folytathatnánk, de mégis kezdjük inkább ezzel, hogy el ne feledkezzünk róla. A következő sorok hozzáadásával módosítsuk */etc/profile* fájlunkat:

```
OMNIORB_CONFIG=/etc/onmiORB.cfg
OMNIORB_LOGDIR=/var/omninames
```

Az *omniORB.cfg* fájl még nem létezik, ezért most létre kell hoznunk egyet, amelynek a következőt kell tartalmaznia:

```
ORBInitialHost localhost
ORBInitialPort 8088
```

Figyeljünk rá, hogy a *localhost*-meghatározás nem szükségszerűen felel meg gépünk beállításainak. Mivel a gépemet egy tartományban lévő hálózati gépként indítom, nekem a megfelelő teljes nevet kellett ide beírnom. Előfordulhat, hogy ezt a saját gépünkön is meg kell tennünk. Most az omniNames-kiszolgálót a következő paranccsal indítsuk el:

```
omniNames --start 8088 &
```

A 3dwm letöltési oldalán található a meshio forrása is, amely a 3D-modellek állományainak betöltését végző programozói könyvtár. Igen, ennek a receptnek az elkészítéséhez bőven akad tennivalónk, de biztosíthatlak titeket, hogy nem fogtok csalódni. Fordítsuk le a meshiót. Örülni fogtok, mert viszonylag gyors művelet:

```
tar -xzvf meshio-0.2.0.tar.gz
cd meshio-0.2.0
make
make install
```

Elérkezett magának a 3dwmnek a fordítása. Írásom idején a változatszáma 0.3.1, ami technikailag egy alfaállapotú programot jelent, de mi, Linux-felhasználók szeretünk a borotva élén táncolni, nem igaz? A fordítás megfelelő végbemeneteléhez be kell állítanunk a PYTHONPATH változót:

```
export
➥PYTHONPATH=/usr/lib/python2-1/site-packages
```

Most már fordíthatunk:

```
tar -xzvf 3dwm-0.3.1.tar.gz
cd 3dwm-0.3.1
./configure
make
```

Ha ezzel elkészültünk, a *tdwmrc* beállítófájlt (amelyet a fordítás könyvtárában, a */etc* alatt találunk) másoljuk a *~/.tdwmrc* fájlba. Nyissuk meg kedvenc karakteres szerkesztőnkkel, és bizonyosodjunk meg a *deault.zorn* elérési útjának helyességéről. Ne feledjük, ez a fordítás könyvtárától függően változhat. Most indítsuk el a megjelenítés kiszolgálóját:

```
cd server
./tdwm-server
```

Ha eddig minden rendben zajlott, egy fekete téglalapnak kell a képernyőn megjelennie. Ez már fél siker. Érdekesen hangzik, de a 3dwm egy ügyfél–kiszolgáló-alkalmazás, és a 3D-ügyfélprogram ebben az ablakban fog futni. Ahhoz, hogy belevethessük magunkat a háromdimenziós világba, egy ügyfélprogramot is el kell indítanunk. A 3dwm honlapján találunk egy *3dwm-data.tar.gz* fájlt, amit le kell töltenünk. Ez néhány mintafájlt tartalmaz 3D-modellekkel és felületi mintákkal. A fájlt egyszerűen ki kell csomagolni, fordításra nincs szükség:

```
cd clients/geoclient
./geoclient office.3ds
```

Egy iroda háromdimenziós képe jelenik meg a 3dwm-kiszolgáló ablakában (asztallal, monitorral, billentyűzettel és egérrel). Ha kicsit furcsán fest, próbáljuk meg a következőképpen bejárni. A mozgás a CTRL billentyű és az egérgombok megfelelő együttesével történik, meghatározásukat egyébként a korábban említett *default.zorn* fájl tartalmazza. Ha a CTRL gombot és a bal egérgombot lenyomva tartjuk, az egér mozgatásával a középpont körül mozgunk. A CTRL billentyű nyomva tartása mellett a jobb egérgombot használva a középpont közelítése-távolítása lehetséges. A CTRL billentyű a középső egérgombbal a középpontot helyezi át. A CTRL-SHIFT a bal egérgombbal együtt jobbra-balra forgást, míg a jobb gomb használata a látómező változását eredményezi. A geoclient mellett más ügyfelet is indíthatunk, de ennek felfedezését már rátok bízom.
A 3dwm csomaggal együtt egy háromdimenziós órát is kapunk, és amint a 3D-órámon láthatjátok, a záróra ismét elért minket. François, légy szíves vendégeink poharát utoljára teletölteni. Merci, mon ami.
Akár a múltban találjuk meg a számunkra eszményi munkakörnyezetet, akár a jelenben, esetleg a jövőben, megbizonyosodhattunk affelől, hogy Linuxszal főzve az együttműködő képesség nemcsak jól hangzó reklámszöveg – ez mindennapi munkánk szerves része.
A votre santé! Bon appétit!

Linux Journal 2002. április, 96. szám

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

Kapcsolódó címek

3Dwm ➲ http://www.3dwm.org

AMIWM ➲ http://www.imagemagick.org

Marcel borlapja
➲ http://www.marcelgagne.com/nfwine.html

MLVWM
➲ http://www2u.biglobe.ne.jp/~y-miyata/mlvwm.html

Az OmniORB honlapja
➲ http://www.uk.research.att.com/omniORB

Ruby

Thomas bemutatja a Ruby programozási nyelv erejét és rugalmasságát.

A Ruby érett, korszerű, tisztán objektumközpontú programozási nyelv. Írásmódja tömör és állandó, így egyszerre könnyen olvasható és tanulható, valamint rugalmas és kifejező. Ha eddig olyan nyelven programoztál, ami erejét egy hatalmasra püffedt API-ból merítette, meg fogsz lepődni azon, hogy a Ruby API milyen kicsi és mégis milyen hatékony. Mivel szorosan együttműködik az alatta futó operációs rendszerrel, és szinte nevetségesen egyszerű bővíteni, a Ruby egyszerre nagyon hatékony és sokoldalú programozási nyelv. Merész állítások? Nézzük, mi rejtőzik a kijelentések mögött!
A példa kedvéért egy egyszerű Ruby programon fogunk dolgozni, amely egy ideiglenes könyvtárból kitörli a megadott napnál idősebb állományokat. Ezen az alkalmazáson keresztül fogom bemutatni a Ruby alapszintű írásmódját, valamint néhány fontosabb képességét. A teljes programot az első lista tartalmazza. (33. CD Magazin/Ruby könyvtár) A programot konzolról a `./purge.rb` *`tmp_dir ÆllomÆny_max_kora_napokban`* paranccsal futtatjuk, ahol is második értékként napokban számolva azt a kort adjuk meg, amelynél idősebb állományokat a könyvtárból törölni akarunk. Természetesen a programot a `crontab`-ból is meghívhatjuk.

Ruby-alapok

A Ruby objektumközpontú nyelv, amely lehetővé teszi az adatok és eljárások objektumba zárását, az öröklést, valamint támogatja a többalakúságot. A nyelvben minden, még a legegyszerűbb adattípusok is (például karakterlánc, egész) objektumként van megjelenítve. Sőt, az állandók és az osztályok is objektumok. Ennél fogva a Ruby tisztán objektumközpontú nyelv. Az egyetlen kivételt e tekintetben a vezérlőszerkezetek képezik, vagyis néhány olyan kifejezés, mint például a `for`, `if`, `while` stb. Ezek nem objektumok.
Amint a 2. listában láthatjuk (33. CD Magazin/Ruby könyvtár), hogy a `delete_older` eljárás adja meg a program általános logikáját: belépni egy adott könyvtárba és megkeresni a törlendő állományokat.
Akik a szigorúan típusos nyelvekhez (pl.: Java, C++) vannak szokva, furcsának találhatják, hogy az eljárásmegadásban az értékek típusát nem adtuk meg. A Ruby dinamikusan típusos nyelv. Vagyis a változónak nincsen típusa, de annak az objektumnak, amire a változó hivatkozik, van. Ezért nem szükséges a típusmegadás. A dinamikus típusosság az osztályöröklléssel szemben az objektumkompozíciót részesíti előnyben. Nem befolyásolhatjuk az objektumok típusát, amelyek értékként adódnak át az eljárásoknak, így kevésbé kell aggódnunk a bonyolult öröklési rendszer miatt, hiszen az objektumok eljárásoknak történő átadása többé nem a többalakúságtól függ. Így egyszerűbb és könnyebben újrafelhasználható kódhoz jutunk.
A Ruby-eljárások megadása ismerősnek fog tűnni a Python-programozók számára. A két nyelvben a megadás tulajdonképpen azonos módon történik, a választható értékek használata által szintén ugyanúgy járunk el. Ha egy érték választható, akkor az eljárás meghívásakor elhagyható. Az érték elhagyása esetén az eljárás az érték alapbeállításával hívódik meg.
A Ruby-eljárások megadásában a visszatérési érték sem szerepel. Mivel a nyelv dinamikusan típusos, nem szükséges megadni a visszaadott érték típusát. Hacsak egy visszatérési értéket nem explicit módon a `return` kifejezéssel adunk meg, akkor – miként a Lisp nyelvben – az eljárás legutoljára kiértékelt kifejezését kapjuk vissza.
Az eljáráshívások tulajdonképpen a célobjektumnak küldött üzenetek. Ez ***Smalltalk***-örökség. A *`cØl. zenet(ØrtØkek)`* formátumú kifejezések valamennyi objektumközpontú programot alkalmazó programozó számára ismerősek. Egy objektumnak küldött üzenet az objektum egy adott eljárását hívja meg. Az objektumok között zajló valamennyi kapcsolattartás üzenetek formájában történik.
A Rubyban az eljárásoknak kétfajta megnevezése létezik: osztályeljárások és példányeljárások, vagy röviden csak eljárások. A példányeljárásokat példánnyá vált osztályokban, elterjedt megnevezéssel élve objektumokban hívjuk meg. Az osztályeljárásokat olyan osztályokban hívjuk meg, amelyekből nem származtattunk példányt, ezek olyanok, mint a Java vagy a C++ statikus eljárásai. Ezért az osztályeljárásokat programkönyvtár-eljárásoknak is felfoghatjuk. Ezek az eljárások nem tudnak az objektumok tagváltozóival dolgozni.

Konténerek

Vizsgáljuk meg a programunkat meghívó alábbi kódot, amely a parancssorban adott értékeket dolgozza fel:

```
path = ARGV.shift or raise
➥"HiÆnyzik az œtvonal"
age  = ARGV.shift or raise "HiÆnyzik a kor"
```

A program meghívásakor a parancssorban átadott paramétereket az ARGV tömb objektum tárolja. A `shift` eljárás visszaadja a tömb első elemét, miközben azt a tömbből eltávolítja. A Ruby fejlett tömbosztállyal rendelkezik. A tömbök dinamikusak: képesek a saját méretük megváltoztatására. Mivel a tömbök is objektumok, sem a memóriafoglalások, sem a tartományon kívüli véletlen elemelérések nem okoznak gondot. Az osztály eljárásai egy tömb feldolgozását az indexei vagy az elemei segítségével teszik lehetővé, de úgy is dolgozhatunk velük, mintha vermek, halmazok vagy sorok lennének. A tömbök megfordíthatóak és rendezhetők. A tábla kezeléshez a hash osztályt alkalmazzuk. Az *1. listából* vett következő sor jól mutatja, mennyire elegánsan kezeli is a Ruby a tömböket:

```
Dir.entries(full_name) - [·.·, ·..·]).empty?
```

A `Dir.entries(full_name)` egy tömbbel tér vissza, amely a könyvtár összes állományát tartalmazza. Ezután a `-` műveletjel segítségével az állományok listájából kivonjuk a `[·.·, ·..·]` tömböt. Majd meghívhatjuk az `isEmpty` eljárást, hogy megvizsgáljuk, üres-e a könyvtár. Ha a tömb üres, vagyis az `isEmpty` igaz logikai értékkel tér vissza, a könyvtárban nem található több állomány.

Hibakezelés

Miután a meghíváskor átadott paraméterek feldolgozásra kerültek, a `delete_older` eljárás hívódik meg:

```
delete_older(path, age_in_seconds)rescue
➥puts "Error: #$!"
```

Előfordulhat, hogy a végrehajtás során hiba lép fel. Ha például a programnak kapcsolóként nem létező könyvtárat adtunk meg, hiba keletkezik, amikor a Ruby a legelső alkalommal próbálja a `Dir` osztály valamelyik eljárását meghívni. A fenti kód nemcsak a `delete_older` eljárást hívja meg, hanem azokat az esetleges hibákat is kezeli, amelyek a futtatás során előállnak. Ebben a `rescue` utasítás játssza a kulcsszerepet. Amikor hiba lép fel, a Ruby-értelmező a hibát egy úgynevezett kivételobjektumba csomagolja. Ez az objektum végiglépked az előzőleg lezajlott hívásokat tartalmazó vermen, egészen addig, amíg olyan kódot nem talál, amely képes az adott típusú kivételt kezelni. Ha a kivétel ilyen módon nem kezelhető, a program futása abnormálisan megszakad. A keresés eredménye az `stderr`-re íródik. Ez a módszer nem hibakódokat ad vissza, mint a héjprogramok vagy a C nyelv, ezáltal kevesebb beágyazott utasítást, spagettilogikát és egyszerűen jobb hibakezelést eredményez.

Ha a `rescue` utasítás mellett az `ensure` parancsot is használjuk, elérhetjük, hogy egy adott kódrészlet minden körülmények között lefusson. Kapcsold össze azt a lehetőséget, hogy saját kivételeket írhatsz, hogy kivételek létrehozására készítheted fel a kódodat (a `raise` utasítás segítségével, ahogy a ***Konténerek*** nevű lista is mutatja), valamint azt a képességet, hogy a kivételekből a program egy adott kódrészlet lefuttatásával vagy a hibát okozó kód újrafuttatásával felépülhet – és máris a legnagyszerűbb hibakezelő szerkezet áll a rendelkezésedre, amivel csak valaha dolgod volt.

Fejlettebb tulajdonságok

Térjünk vissza a `delete_older` eljáráshoz, hogy szemügyre vegyük a Ruby néhány fejlettebb szolgáltatását (lásd 33. CD Magazin/Ruby könyvtár 2. lista).

A második sorban a `Dir` osztályon belül a `foreach` utasítást hívjuk meg; a `foreach` az iterációs tervezési minták egyik megvalósítása. Ha objektumközpontú programozással foglalkozol, de még nem olvastad a Négyek bandájának úttörőnek számító könyvét, a ***Tervezési minták***-at, akkor jobban teszed, ha minél hamarabb beszerzel egy példányt. Az interátor nem az egyetlen minta, amely a Ruby nyelvben beépítetten megtalálható. A singleton, publisher/subscriber, vizitor és delegációs minták szintén megvalósítottak. Szükség esetén további minták adhatók a nyelvhez, de az előbb felsoroltak a nyelv alapvető részét képezik.

A `foreach` utasítás egy könyvtár összes állományát bejárja. A `foreach` meghívása után egy kódblokkot látunk, amelynek a kezdete és vége igencsak hasonlít a szabványos Java vagy C++ kódblokkokhoz. A kapcsos zárójelek közötti kódrészletet blokknak hívjuk, amely olyasmi, mint egy eljáráson belüli eljárás. A blokk sosem akkor hajtódik végre, amikor a program futása eléri a blokkot. Amikor a `foreach` eljárás beolvasta a könyvtár egy állományát, a vezérlést átadja a blokknak. A blokkon belüli kód ekkor végrehajtódik, majd a vezérlés visszakerül a `foreach`-re, ami újabb állományt olvas be a könyvtárból. Ez a folyamat addig ismétlődik, míg a ciklus végül a könyvtár összes állományán végiglépked.

Ahelyett, hogy az iterátorok működését segítő osztályokat kellene írnunk, mint a Java vagy a C++ nyelvben, a Ruby rendelkezik a `yield` utasítással, amely lehetővé teszi, hogy az iterátorokat eljárásként valósítsuk meg. Ez remek példája annak, mennyire kifejező és rugalmas a nyelv. Azáltal, hogy egy ötlet megvalósításakor nem kell az alapokhoz visszanyúlnunk, a Ruby lehetővé teszi, hogy magára a feladatra összpontosíthassunk.

Mint korábban már említettük, az eljárások hívása a célobjektumnak küldött üzenettel történik `cØl. zenet` formátumban. A cél megadása nélkül csak az adott osztály helyi eljárásait érhetjük el. Programunk a célobjektum megadása nélkül hívja meg a `puts` utasítást, ami a rendszermag egyik eljárása. Mivel nem adtunk meg célobjektumot, és a `puts` nem helyi eljárása annak az objektumnak, amiből hivatkozunk rá, honnan tudja az értelmező, hogy melyik eljárást hívtuk meg?

A varázsszó a bekeverés (Mix-in). A bekeverés alapvetően azt teszi lehetővé, hogy valamely osztályból öröklődés nélkül hívhassunk meg egy olyan eljárást, amely máshol lett megvalósítva (a bekeverésről bővebben a Linuxvilág 2001. áprilisi számának 47. oldalán olvashatunk, a Mix-in osztálytípus című cikkben). A bekeverés nem új ötlet, és nem is kizárólag a Ruby sajátja. De feltétlenül nagy szerepe van abban, hogy a Rubynak tiszta, érthető írásmódja van.

Még csak nem is remélhetem, hogy egy ekkora terjedelmű cikkben a Ruby összes jellemzőjét be tudom mutatni. Inkább arra bátorítalak, hogy olvasd el ***David Thomas*** és ***Andrew Hunt*** Ruby-programozás (Programming Ruby) című könyvét, amiben bőséges és részletes magyarázatot találsz az olyan fogalmakra, mint modulok, álnevek (alias), névterületek, megjegyzések, dinamikus eljáráshívások, rendszerkapcsolodások, elosztott programozás és hálózati programozás. Érdemesnek tartom még megemlíteni, hogy a Ruby épp olyan jó szabályos kifejezéstámogatással bír, mint a Perl, a CGI-t is támogatja, sőt saját Apache-modulja van, a mod_ruby.

Összegzés

A Ruby csupán újabb tagja lenne a parancsnyelvek családjának? Biztosan nem. Valami más, több, újabb és izgalmasabb jelent meg a nyílt forrást fejlesztők japán berkekből. A Ruby a programozók legjobb barátja. Bár elsősorban parancsnyelvnek tervezték, bebizonyította, hogy nagyobb munkafolyamatokban is megállja a helyét. Olyan izgalmas jellemzőkkel bír, amelyeket a választható programnyelvekben csak mostanság kezdenek el megvalósítani. A Ruby tehát mindenképpen megér egy próbát.

Köszönetnyilvánítás

Különleges köszönet szaklektoraimnak: ***Armin Roehrl***-nek az ➲ http://approximity.com-tól, amiért átnézte a kéziratot, és a végső változat szerkesztésében segítségemre volt. ***David Thomas***-nak a ➲ http://pragmaticprogrammer.com-tól, amiért nagymértékben javított az eredeti mintaprogramon, és átnézte a kéziratot. ***Kent Dahl***-nak és ***Sean Chittenden***-nek a kézirat átnézéséért. Végül, de nem utolsósorban ***Magnus Lie Hetland*** Python-gurunak az értékes segítségért.

Linux Journal 2002. március, 95. szám

Thomas Österlie
tanácsadó a norvégiai telephelyű ConsultIT A/S cégnél, ahol a Unix-kiszolgálókon rendszerfejlesztéssel és számítógépes biztonsággal foglalkozik.

Együtt(működve) könnyebb

Számlázóalkalmazások, FTP-ügyfelek, hőmérséklet-figyelés és sokan mások…

Számomra az együttműködés a 106 féle Unix-változat egy csapatba gyúrását jelenti a Microsoft saját elképzelések szerint fejlesztett, az RFC-khez csak ritkán igazodó, roppant idegesítő operációs rendszereivel. A Unix-változatok kiválóan megvannak egymással, annak köszönhetően, hogy követik a szabványok előírásait. A Microsoft „100% Microsoft" elgondolása és ügyesen elhelyezett, az együttműködési lehetőségeket rontó akadályai ugyanakkor soha ki nem merülő bosszúságforrást jelentenek a rendszergazdák számára, akik közül sokan úgy vélik, a „0% Microsoft" sokkal jobban hangzó jelmondat lenne. Eközben a Microsoft rejtett fejlesztési felületei több gondot jelentenek, mint az időnként ugyancsak mozgó célpont módjára viselkedő Linux. Szerencsére léteznek olyan programozói csapatok – például a Samba fejlesztői –, amelyek a Microsoft minden próbálkozása ellenére megteremtették a közös munka lehetőségét. A Microsoft-termékek és a Linux tökéletes együttműködése azonban sajnos továbbra is messzi cél.

GnuLedger

Teljes mértékben Linuxra áttérni kívánó ismerőseim gyakran panaszkodnak, hogy semmilyen személyi pénzügyi rendszert nem találnak. Üzleti célokra jó pár kiváló alkalmazás érhető el, de a személyi használatnál sajnos más a helyzet. Tudnék ugyan legalább három ilyen programot említeni, ám adatbázisként mindegyik egyszerű állományokat használ, és csak helyben futtatható. A GnuLedger ellenben webböngészőn keresztül érhető el, így a világon bárhonnan hozzáférhetünk, célszerűen a HTTPS-protokoll felett. Mivel az adatokat a MySQL segítségével tárolja, a lekérdezések gyorsak, és némi SQL-tudással akár a GnuLedger nélkül is elvégezhetők. Az egyetlen gond a GnuLedgerrel az, hogy – legalábbis az általam letöltött legújabb változat esetében – nem kapunk hozzá táblákat: mindet magunknak kell létrehoznunk, márpedig egy átlagos otthoni felhasználó valószínűleg nem erről álmodik. Néhány átfogó, mindenki számára jól használható tábla minden bizonnyal sokat lendítene a GnuLedger sikerén. Esetleg nem akarja valaki elkészíteni, majd közzétenni őket? A futtatáshoz szükséges: MySQL, webkiszolgáló, Perl, Number::Format, DBI, DBD::mysql, Math::Currency és Math::FixedPrecision Perl-modulok.

➲ http://webaccountant.sourceforge.net

phpPgAdmin

Ha ismered a phpMyAdmint, számodra a phpPgAdmin sem tartogat meglepetéseket. A webalapú PostgreSQL felügyeleti program a rendkívül népszerű phpMyAdmin átültetése. Segítségével néhány kattintással elintézhető a Postgres felügyelete. A telepítés egyszerű, és te választhatod ki, hogy csak az egyik vagy az összes adatbázist vele akarod-e kezelni. A futtatáshoz szükséges: webkiszolgáló PHP- és pgSQL-támogatással, webböngésző és PostgreSQL.

➲ http://phppgadmin.sourceforge.net

hardmon

Ha az alaplapodon megfelelő érzékelők vannak (egyelőre leginkább az Asus termékei jönnek szóba, de a támogatott alaplapok köre lassacskán bővül), grafikus felületen követheted figyelemmel a géped állapotát (például a processzor hőmérsékletét). Mivel én Panamában élek, és jó néhány gépnek nem jutott légkondicionált hely, a program nyáron kiváló szolgálatot tesz, mivel azonnal jelzi, ha a hőmérséklet túl magasra emelkedik, és jobb egyik-másik programot leállítani. Egy- és többprocesszoros gépeken egyaránt működik. A futtatáshoz szükséges: lm_sensors, libXpm, libX11, libm és glibc.

➲ http://www.fcoutant.freesurf.fr/hardmon.html

Freemoney

Ha van egy kis üzleted, ahol rengetegféle termék értékesítésével foglalkozol, érdemes megismerned a Freemoney-t. Ehhez elég ellátogatnod a fejlesztők honlapjára, majd a próbaoldalt választanod. A kiterjedt szolgáltatásokat nyújtó számlázási csomag akár webes, akár hagyományos boltok számára megfelelő. A telepítésével sajnos nagyon sokat kell dolgozni, és leírása is hiányos. Nem ötperces munka lesz, de ha végzel, egyszerű használhatósága el fog bűvölni. A futtatáshoz szükséges: Interchange, webkiszolgáló, Perl- és Bundle::Interchange-, DVD:Pg-, DBI & Schedule::At Perl-csomagok, valamint PostgreSQL.

➲ http://www.freemoney.org

gFTP

E hónapban könnyű volt a három évvel ezelőtti anyagból választani. Valódi gyöngyszem az FTP-ügyfelek között. Mind grafikus, mind szöveges módban remekül használható, az átviteleket FTP- (aktív és passzív), SSH-, SFTP- vagy http-protokollon keresztül képes bonyolítani. Több szálon fut, így egyszerre több letöltést is végezhetünk vele. Képes a letöltések sorba állítására, a segítségével akár két távoli rendszer között is továbbíthatunk anyagokat, és sok egyéb kényelmi szolgáltatást nyújt még. Ha gyakran mozgatsz állományokat különféle rendszerek között, mindenképpen próbáld ki. A futtatáshoz szükséges: gftp-text: libglib, libnsl, glibc; gftp-gtk: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, libpthread, glibc.

➲ http://gftp.seul.org

Linux Journal 2002. április, 96. szám

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