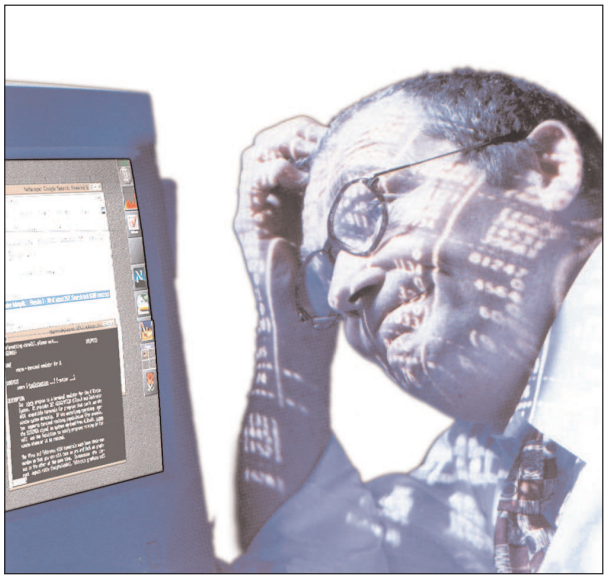
A Linux, az Iroda, a Multimédia meg a többiek

Pár hónapja gaztettet követtem el az egyik megrögzött windowsos ismerősöm ellen, leültettem egy linuxos gép és azt mondtam neki:
– Használd!
Kis idő elteltével megkérdeztem:
– Mit érzel?
– Nem tudom, kicsit mintha izzadna a tenyerem és félnék ettől a helyzettől – jött a válasz.
Ezután még mindig ott ült és mindenféle programok futtatásával, szöveg szerkesztésével, filmek lejátszásával, hang rögzítésével próbálkozott. Láttam az arcán, hogy sehogyan sem boldogul. Telt-múlt az idő (már vagy két órája nyüstölte a gépemet), amikor egyszer csak azt kérdezte:
– Miért nem működik semmi?
– Miért, nem működik?
– Nem!
– Mit szeretnél csinálni?
– Filmet nézni!
– Keress egy xine nevű programot, indítsd el, meg tudod vele nézni a filmet.
– Na, ez milyen ronda! Hol kell behívni a fájlt?
Miután nagyvonalakban elmagyaráztam, mit hol talál, újabb két óra telt el, de közben megváltozott az arckifejezése: izgatottság, öröm, gond és még számtalan ellentétes érzést figyelhettem meg rajta. Egyre jobban el tudott szakadni a beidegződésektől, tanulóreflexe beindult: ha valamit nem tudott, vagy valamivel nem boldogult, egyre ritkábban kérdezett engem, helyette terminálablak, böngésző, könyv került elő, és vadul gépelt mindenhova segítségért (ugyanis a Windows-súgó után a manoldalak kissé „szokatlanok"). Minél jobban megismerte a rendszert, annál jobban érdekelte a dolog, és mivel minden lehetőség adva volt, hogy a legnagyobb ismeretanyagot szedje össze, nagyon gyorsan tanult. Az egynapos „tortúra" végén felállt a gép elől, és azt mondta:
– Ez jó! Nekem is kellene egy ilyen.
Másnap vasárnap volt, így hát felhívtam:
– Itt állok a CD-immel, készülj, mert jövök és telepítünk!

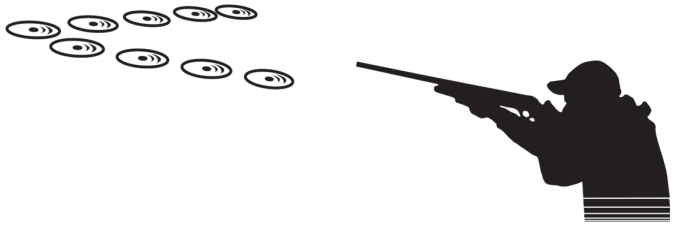


Kis mentegetőzés után kiderült, hogy a tegnapi eufórikus állapot után már kissé megszeppenve gondolt vissza a történtekre. Tetszett ugyan neki a dolog, de mint kiderült, este visszahódította szívét a Windows. Hogyan is történhetett? Hazament, leült a gép elé, bekapcsolta, a feje zsongott a sok újdonságtól, ám ahogy meglátta a „kórházzöld" hátteret, mindent elfelejtett, és már ismét csak a Windows volt a „nyerő". A régi beidegződések újra megerősödtek (kliketi-kliketi-klikk).
Mivel ez is amolyan „amerikai sikertörténet", természetesen nem lehet a vége más, csak az, hogy a jó mégis legyőzte a gonoszt... A valóságban viszont az történt, hogy aznap nem telepítettünk semmit, szép napos idő volt, és még elég meleg a kerthelyiségekben való ücsörgésre. A következő hétvégén azonban ismerősöm restelkedő felhangon hívott, hogy mégiscsak jó lenne, ha neki is Linux lehetne a gépén. Lett. Azóta felváltva használja a két rendszert, a Linux még csak hobbi, szabadideje rovására tanulgatja, de a legfőbb feladatokat még Windowsban oldja meg.
A fenti példa is jól mutatja, mennyire nehéz megszokni egy másik rendszert. Ismerősöm nagy előnye a szakmai angol nyelv ismerete, ezért tudta viszonylag gyorsan feldolgozni az Interneten és a manoldalakon talált anyagokat.

Mindemellett azonban a manapság használatos ablakkezelőknek és egységesített felületeknek köszönhetően számos olyan lehetőség működik, amit már megszoktak az emberek (húzd és ejtsd, kuka, háttérbeállítás, önműködő fájlmeghatározás és a megfelelő alkalmazással történő megnyitás stb.). Így azért könnyebb a leendő felhasználókat elcsábítani és rávenni őket arra, hogy Linuxot használjanak.
A programokkal pedig az a helyzet, hogy szinte minden feladatra lehet megfelelő programot (akár programokat is) találni, mivel ma már több ezer közül választhatunk. Az irodai programcsomag magyarul szól hozzánk és a helyesírásunkat is ellenőrzi, reméljük, hamarosan az első magyar nyelvű Linux is megjelenik, a fejlesztése most is javában zajlik.
Ne feledkezzünk meg az anyagi előnyökről sem: ha az embernek döntenie kell, hogy közepes gépet, hozzá operációs rendszert és programokat vásároljon-e, vagy nagyságrendekkel jobb gépet vegyen – olyat, ami szinte mindenre kifogástalanul megfelel – (esetleg ugyanazt a közepes gépet veszi, viszont fele annyiból, programok nélkül), és Linuxot használ rajta. Ez leginkább akkor látszik, ha több gépet kell vásárolni, hiszen ilyenkor egyáltalán nem mindegy, hogy tíz vagy esetleg húsz gépet tudnak-e a dolgozók rendelkezésére bocsátani.
Természetesen ez az elmélet sem hibátlan, rengeteg olyan helyzet van, amit Linuxszal nem, vagy csak nagyon körülményesen lehet megoldani – ezt mindig az adott helyzet dönti el.
A Linux most már nem csak a programozóké, azonban a rendszergazdáké és a guruké!

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

Programvadászat



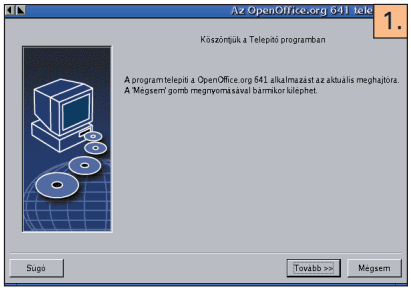
A jelenlegi korongunkon a legnagyobb értékkel bíró csomag minden bizonnyal a magyar nyelvű OpenOffice. Mint előző számunkban már hírt közöltünk róla, február első hétvégéjén OpenOffice-magyarítási maraton zajlott le. Sok lelkes Linux-hívő fáradságot nem kímélve három nap alatt elkészítette e nagyszerű irodai csomag magyar nyelvű változatát. Szerintem nagyon jól sikerült! A linuxos változat letölthető (természetesen a 28. CD Iroda/OpenOffice könyvtárában is megtalálható), a windowsos változat sajnos még nem készült el, így a projekt támogatói 40 000 forint „vérdíjat” tűztek ki rá. Remélem, hamarosan ez is létrejön. Maga az OpenOffice egy összetett, minden igényt kielégítő irodai programcsomag: szövegszerkesztővel, táblázatkezelővel, bemutatókészítővel, HTML-szerkesztővel, rajzolóprogrammal, valamint a Microsoft-termékekkel készített fájlokhoz való szűrőkkel rendelkezik, így nyugodtak lehetünk, egy elektronikus levélmellékleteként kapott fájl feldolgozása már nem okozhat gondot. Mindenki szempontjából nagyon fontos, hogy ez a termék teljesen ingyenesen férhető hozzá, ami azonban nem azt jelenti, hogy „mivel ingyenes, biztos valami buta, esetleg korlátozott változat”, hanem hogy nagyon jó minőségű terméket kapunk mindössze egy magazin vagy a letöltés áráért! Így – azt hiszem – ez esetben az ár-teljesítmény arányt egyetlen hasonló programéval sem lehet összevetni.

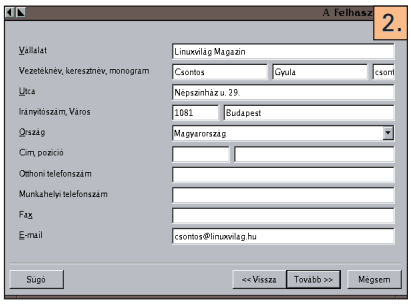
Az alábbiakban telepítésének egyszerű módját ismertetem, melynek során lépésről lépésre haladva mutatom be a feladatokat, hogy a programot a teljesen kezdő, ám a Linux iránt érdeklődő felhasználók is sikerrel tudják használni mind otthon, mind a munkahelyen, esetleg az iskolákban is.

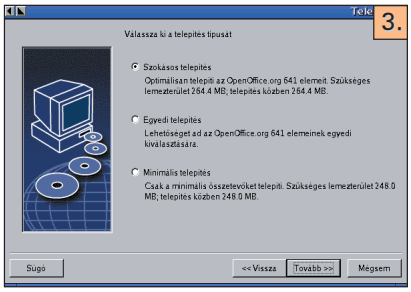
OpenOffice telepítési útmutató

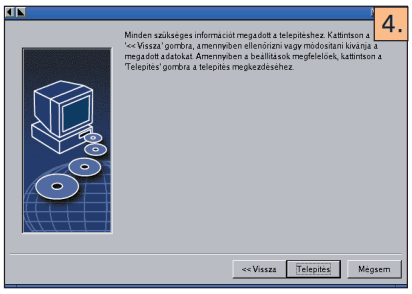
Nos kérem, nem kell többé félnünk az idegen nyelvektől, az OpenOffice-t magyar nyelven telepíthetjük.

Mint láthatjuk, már a telepítő első ablaka is ékes magyarsággal szól hozzánk. Miután a ***Tovább*** gombra kattintottunk, elolvashatjuk az átmeneti ***olvassel*** fájlt, ahol tájékoztatást kaphatunk a termékről. Ezután átbön-



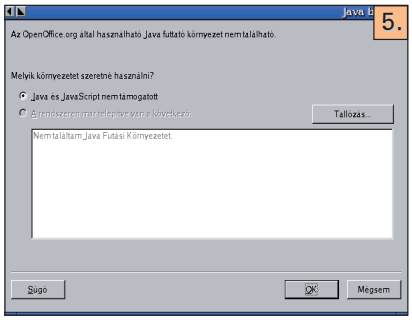


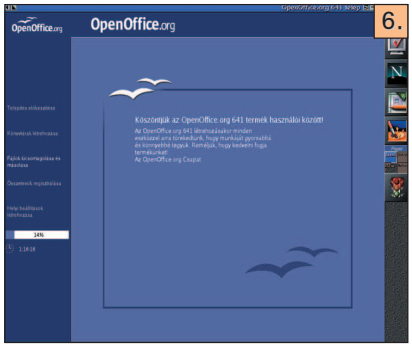




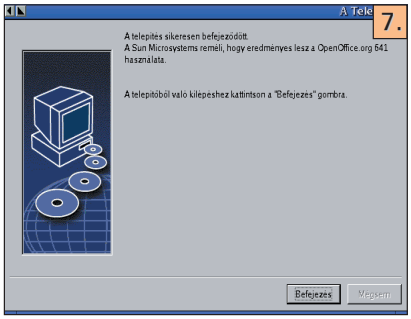
gészhetjük a felhasználói szerződést (ez angol nyelvű), majd elfogadva megkezdődhet a telepítés előkészítése: első lépésben személyes adatainkat (név, telefonszám, cég stb) adhatjuk meg. Továbbhaladva a telepítés típusát tudjuk kiválasztani, ami lehet ***Szokásos telepítés***, ekkor megközelítőleg 265 MB helyre lesz szükségünk. Az ***Egyedi telepítés***-t választva a program telepítését ízlésünknek megfelelően hangolhatjuk. Ezenkívül kedvünkre való lehet a ***Minimális telepítés*** is, ez a mód ugyanis a szükséges összetevőket minden „sallang” nélkül telepíti fel, ekkor megközelítőleg 248 MB helyre lesz szükségünk. Mint látható, a ***Szokásos*** és a ***Minimális*** telepítés között nincs igazán nagy különbség (legalábbis méretben), ezért szerintem – ha csak nem vagyunk nagyon helyszűkében – érdemes a ***Szokásos telepítést*** választani. Ezután a telepítő felkínálja az alapértelmezett könyvtárat, ami az én esetemben a ***/home/csontos/OpenOffice.org641*** volt. Ezt el is fogadtam, mivel azonban ilyen nevű könyvtár eddig még nem létezett, a ***„A '/home/csontos/OpenOffice.org641' könyvtár nem létezik. Létrehozza?”*** kérdésre igennel válaszoltam.

A *4. képen* a tényleges telepítés előtti utolsó lépés látható, ha igennel válaszolunk, innen már nincs visszaút! Ha valamin még módosítani szeretnénk, lépjünk vissza és tegyük meg, ha nem, akkor viszont kezdődjék a telepítés (*6. kép*)! Az OpenOffice-t





böngészésre is használhatjuk, ez esetben minden bizonnyal Java- és JavaScript-támogatásra lesz szükségünk. A telepítő több-kevesebb sikerrel képes felderíteni a rendszerünkön telepített Java-környezetet. Ha nincs vagy nem lesz rá szükségünk, a lépést hagyjuk ki – egyébként telepítsük

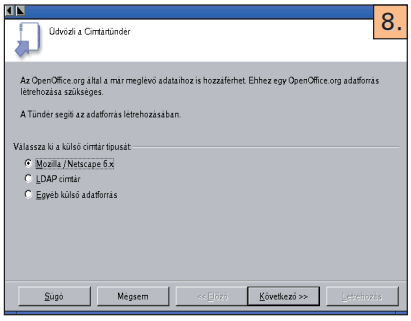


7.

vagy állítsuk be (*5. kép*)! Ezekután nincs más hátra, mint megvárni a *7. képen* látható képernyő megjelenését. A telepítés a végére ért, a ***Befejezés*** gombra kattintva elhagyhatjuk a telepítőt.

A használatba vétel

Miután a telepítést sikeresen végrehajtottuk, bizonyára használni szeretnénk a programot. Ezt a `~/OpenOffice.org641/soffice` parancs kiadásával tehetjük meg, de erre a feladatra természetesen bármilyen fájlkezelő programot is (például Konqueror) használhatunk. Első indításkor egy „tündérrel” találkozunk – na, semmilyen mesére ne gondoljunk: a ***Címtár tündér*** segítségével a Mozilla/Netscape 6.x, esetleg az LDAP- vagy külső adatforrásból

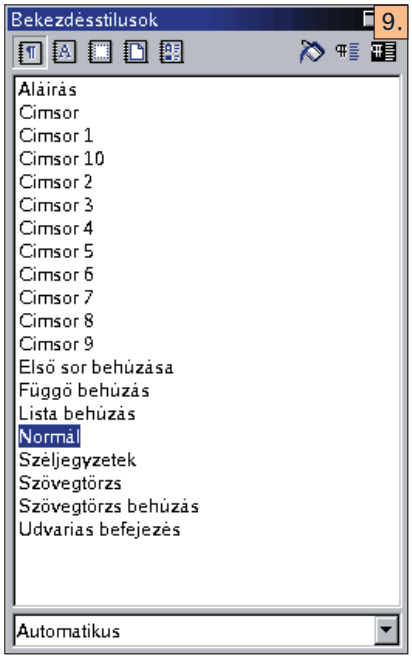


8.

vehetjük át a címtárunkat (8. kép). Miután címtárunk feldolgozásával végeztünk, a szövegszerkesztőbe azonnal el is kezdhetünk írni. A menük teljes mértékben magyarítottak, a rendszernek egy magyar ***ispell*** alap helyesírás-ellenőrző rendszer is a része. A súgórész azonban sajnos csak angol nyelven érhető el, de ne feledjük, ennek lefordítása hatalmas munka!

Utolsó simítások a megfelelő működéshez

A fentebb leírtak egy egyszerű felhasználói telepítést mutattak be. Ha az adott gép több felhasználót is kiszolgál,



9.

a telepítést rendszergazdai jogosultsággal érdemes végezni, és a `setup` parancsot a `/net` kapcsolóval jó futtatni. Ekkor ugyanis a teljes programot egy mindenki által elérhető könyvtárba telepítjük (mondjuk a ***/opt/OpenOffice.org641***), majd a felhasználók mindegyike a `/opt/OpenOffice.org641/setup` programot futtatva telepítheti saját magának. Ilyenkor figyelnünk kell a helyesírás fájljaira, mivel nem önműködően kerülnek a felhasználó könyvtárába! A gyógyír: a ***/opt/OpenOffice.org641/user/wordbook*** könyvtárból az összes fájlt át kell másolni a ***/home/FELHASZNÁLÓNÉV/OpenOffice.org641/user/wordbook/*** könyvtárba. Mindenkinek eredményes magyar nyelvű linuxos irodai élményeket kívánok, és köszönettel tartozunk mindazoknak, akik mindezt elérhetővé tették a számunkra!
A magyarítók névsora és további érdekes adatok a ➲ http://office.fsn.hu címen érhetők el.

Mozilla

Ismét megjelent egy frissítés Mozilla böngészőből. A 28. CD Mozilla könyvtárában található mindenki által telepíthető tar.gz illetve rpm formátumban.

PostgreSQL 7.2

Elkészült ennek a remek SQL-adatbázisnak a 7.2-es változata, korongunkra forrás- és rpm formátumban is felkerült. Az rpm-ek Red Hat 6.2, illetve 7.2 alá is egyszerűen telepíthetők.

Rendszermag

A fejlesztői rendszermag 2.5.4-es változatát adjuk most közre. Mindenki csak a saját felelősségére használja!

29. korong

A LokiGames feladta linuxos játékfejlesztését és bezárta kapuit. Ennek ellenére az általuk készített játékok mindenkinek megmutatták, hogy a linux igenis alkalmas és megfelelő játékfelület bármilyen játékhoz, és hogy a játékok nemcsak egyszerű kártyajátékok lehetnek, hanem az igen fejlett grafikus felületet, az egyre jobban támogatott 3D-gyorsítókat kihasználva bármilyen grafikával tervezett alkalmazások is (gondoljunk csak a Quake-játékokra). A grafikus felület fejlődésével a Linux mind jobban erőre kap, úgyhogy igazán remek játékgépet állíthatunk össze. A cég játékai között mindenféle volt, így mindenki kiválaszthatja, mire is szeretné az idejét „elpazarolni”. Olyan nagy nevű játékok szerepeltek a kínálatukban, mint a Quake III Arena, a Soldier Of Fortune, Descent3, a Railroad Tycoon II, a SimCity 3000, az Eric's Ultimate Solitarie, a Heretic II, a Heavy Gear II, a Rune: Halls of Valhalla, a Tribes 2, a Sid Meier's Alpha Centauri with Planetary Pack, a Heavy Metal: F.A.K.K.2, a MindRover, a Deus Ex, az Unreal Tournament, a Heroes III, a Myth II: Soulblighter és a Civilization: Call to Power – mint a listából is látszik, mindenkire gondoltak. A stratégiai játékoktól a kártyajátékokon keresztül a vérengzős akcióig minden megtalálható. Sajnos úgy látszik, korai volt a piacot a jobbnál-jobb játékokkal elárasztani. 29. korongunkon a LokiGames különféle játékokhoz kiadott javításait tesszük közzé, amit a következő mellékleteken folytatunk.

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

Linux és Playstation

Már csak májusig kell várniuk azoknak, akik a Playstationbe és a Linuxba egyaránt beleszerettek.

A ➲ http://www.us.playstation.com című honlapról ugyanis ekkortól rendelhetik meg azt a kiegészítő készletet, amellyel a különféle linuxos alkalmazások immár a népszerű játékkonzolon is futtathatók lesznek.
A 199 dolláros csomag tartalma várhatóan a következő:

- 40 GB-os merevlemez,
- 100 Mb sebességű ethernetcsatoló,
- a Linux for Playstation2 operációs rendszer 1.0-s változata DVD-lemezen; 2.2.1-es rendszermaggal, Xfree86-környezettel, fejlesztői eszközökkel,
- számítógépes képernyő csatlakozását lehetővé tévő csatlakozó,
- USB-csatolós billentyűzet és egér.

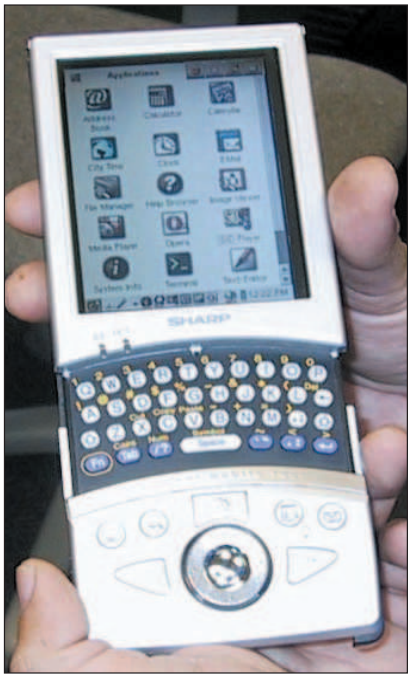
A kiegészítő csomagot amerikai Playstation2 konzolokhoz készítették.
➲ http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/01-30-2002/0001658223&EDATE=

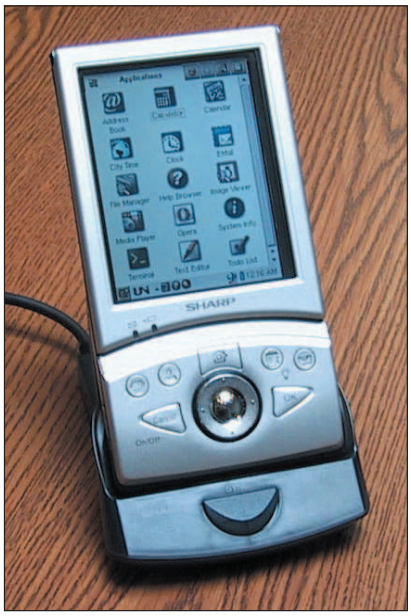
Váltani készül az Oracle

Larry Ellison szerint az Oracle Linuxot futtató Intel processzoros kiszolgálókkal készül leváltani azt a három unixos gépet, amelyek a cég üzleti alkalmazásainak túlnyomó részéhez a hátteret biztosítják. Ellison szerint a nagygépek órái meg vannak számlálva, mivel az Intel-alapú gépek könnyebben karbantarthatók, gyorsabban cserélhetők – a kijelentést a Sun, az Oracle egyik fontos szövetségese bizonyára értékelte.
A Linuxra sem véletlenül esett a cég választása, hiszen legfőbb vezetője szerint a nyílt operációs rendszer internetkapcsolattal rendelkező rendszerek esetén megbízhatóbb, mint a Microsoft programjai.
Ellison szavainak hitelét némileg csökkenti, hogy egy pénzügyi elemzők számára tartott, az Oracle új fürtözési megoldásának előnyeit ecsetelő bemutatón hangzottak el. Az olcsó, Intel-alapú, fürtözött gépekre telepített alkalmazások nagyobb megbízhatósággal és alacsonyabb költségekkel futtathatók – véli Ellison.
Larry Ellison az utóbbi időben örömmel tör borsot a nagy számítástechnikai cégek vezetőinek orra alá, és Oracle részvényeinek értéke révén ***Bill Gates*** első helyét komolyan fenyegette a világ leggazdagabb embereinek rangsorában.
➲ http://www.computerworld.com/storyba/0,4125,NAV47_STO67867,00.html

Hamarosan boltokba kerül a Zaurus

A Sharp Zaurus nevű, Linux-alapú zsebtitkáráról már a ***Linuxvilág*** hasábjain is szó esett. Valószínűleg nem sokáig kell már várni az újdonságra, megjelenése márciusban esedékes – minden bizonnyal először a japán, majd az amerikai boltokban.





A lelkes fejlesztők már hónapok óta kezükben tarthatják a Zaurus kisebb tudású változatát. A végleges Zaurus SL-5500 típusban 64 MB memória és 16 MB flashmemória áll a 206 MHz-es Intel StrongARM processzor rendelkezésére, a megjelenítést 240×320 képpontos, színes TFT érintőképernyő végzi, az adatbevitelt pedig egy ötletesen beépített billentyűzet is segíteni fogja.
➲ http://www.linuxdevices.com/news/NS6655019514.html

Bemutatkozott a MontaVista Linux 2.1

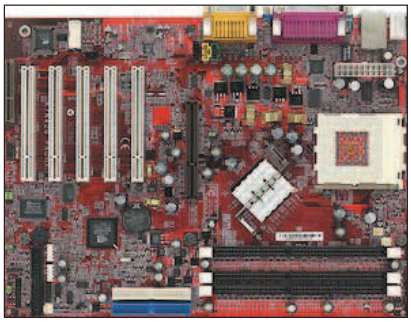
A MontaVista Software bemutatta MontaVista beágyazott Linuxának 2.1-es változatát. A MontaVista Linux érdekessége, hogy hat gyártó több mint 20 processzorát, géptípusát támogatja, közülük a legismertebbek az IA-32, a PowerPC, a StrongARM, a MIPS vagy az SH. A terjesztés a legújabb 2.4-es rendszermagsorozat tagjait használja, amelyeket a cég a célterület igényeihez további valós idejű működést segítő fejlesztésekkel igazított.
A MontaVista Linux számos fejlesztési környezetet támogat, és a legújabb vezeték nélküli kapcsolatok használatára is alkalmas.
➲ http://www.mvista.com
➲ http://www.linuxnews.com/

CCOffice: irodai csomag beágyazott Linuxra

A CCOffice az első olyan kínai fejlesztésű beágyazott irodai csomag, amely képes a Microsoft Office formátumainak kezelésére. A beágyazott irodai csomag kiváló megoldás zsebtitkárokra, vékony ügyfelekre és egyéb beágyazott eszközökre.
A sajnos nem ingyenes csomag három fő részből: a szövegszerkesztőből, a táblázatkezelőből és egy pdf-megjelenítőből áll, az együttes teljes mérete 1,5 MB. Egyaránt támogatja a Microwindows-, a Tiny-X- és a Qt/Embedded-környezetet. Nekünk, magyaroknak talán azért is érdemes lehet odafigyelnünk rá, mert ha képes a többféle módon is bevihető kínai karakterek megjelenítésére és felismerésére, akkor egyszer talán az ékezetes magyar betűk kezelésére is alkalmasnak mutatkozhat.
➲ http://www.linuxnews.com/stories.php?story=02/02/03/2162692

nVidia-lapkakészletes alaplapok az üzletekben

Sokat és régóta hallani arról, hogy a grafikus lapkáiról ismert nVidia is belép az alaplapi lapkakészletet fejlesztő cégek sorába. A sokat emlegetett lapkára épülő alaplapok azonban már nemcsak a fényképeken vagy a hírekben bukkannak fel, hanem a hazai üzletekbe is megérkeztek. Az egyik ilyen típus az MSI K7N420 Pro alaplapja.



Az AMD processzorokhoz készült alaplapon a lapkakészletbe építve egy 3D-vezérlőt találunk, amely nagyjából egy GeForce2 MX400-as kártya teljesítményét nyújtja, valamint egy 10/100 Mb/s sebességű ethernetcsatolóra és egy hatcsatornás hangkártyára bukkanthatunk. Az alaplapon öt PCI- és egy osztott CNR-foglalat lelhető fel, a memóriákat három aljzat várja, és a tervezők szerencsére a további bővíthetőségről sem feledkeztek el, és az AGP-foglalatot sem hagyták el. Az alaplap ára ugyan egy kicsit borsos, körülbelül 50 000 forint plusz áfa, ám ha összeadjuk egy jobb AMD-s alaplap, egy MX400-as kártya, egy ethernet és egy hangkártya árát, az összeg már nem is tűnik olyan nagynak.

➲ http://www.msi.com.tw

Debian-BSD?

Érdekes fejlesztési vonal kezdi bontogatni a szárnyait az alábbi címen. Mivel a Debian-projekt nem felület- és operációsrendszerfüggő, tehát nemcsak Linux-felület adhat neki otthont, a fejlesztők egy BSD-alapokon nyugvó, de szintén Debian rendszert készíthetnek. Ennek a rendszernek az alapja a NetBSD, ugyanis ez a legtöbb számítógépfajtán futó Unix-alapú rendszer. Ugyanilyen választható megoldást jelenthet majd a jövőben a Hurd mag köré épített Debian rendszer, ez azonban még eléggé kezdetleges állapotban van, így komoly munkára nem alkalmas.

➲ http://www.srcf.ucam.org/~mjg59/debian-netbsd/debian-netbsd.tar.gz

➲ http://debian-bsd.sourceforge.net

Share360: csoportmunka-kiszolgáló Linux alá

A japán Cybozu Corporation fejlesztő-cég megjelentette Share360 termékének angol nyelvű változatát is. Bár nem viseli az 5-ös számot, a kétféle változatban kapható programcsomag a Cybozu Office 4 utóda. Segítségével tennivalóink listáját, naptárunkat, címtárunkat, feljegyzéseinket központi helyen tárolhatjuk, és munkatársainkkal többek között elektronikus hirdetőtáblát és vitafórumot üzemeltethetünk és oszthatunk meg. A kiszolgáló tartalma webes felületen keresztül érhető el, akár az irodán kívülről is.

A Share360 próbaváltozatát bárki megtekintheti a gyártó honlapján, a

➲ http://www.share360.com címen, valamint a háromhónapos működő változatot is letöltheti. A csomag egyetlen kellemetlen jellemzője, hogy nem ingyenes, sőt a felhasználók számától és a változattól függő 900 dollártól kezdődő ár elég magasnak mondható.

➲ http://linuxpr.com/releases/4489.html

Norvég szakemberek nyerték a 2001. évi Turing-díjat

Két norvég szakember ***Ole-Johan Dahl*** és ***Kristen Nygaard*** nyerte a 2001. évi Turing-díjat. A valós számítógépeket modellező Turing-gép atyjáról, ***Alan Mathison Turing***-ról elnevezett díjat a számítástechnika Nobel-díjaként is szokták emlegetni.

A két norvég fejlesztő az objektumközpontú programfejlesztési szemlélet kialakításában végzett munkájáért kapta az elismerést.

Dahl és Nygaard még a hatvanas években Oslóban fejlesztették ki a Simula 1 és Simula 67 programozási nyelveket. Munkásságuk során alakult ki az az újfajta szemlélet, melyben a programozók munkájukat elvonatkoztatási rétegenként végzik, az egyes rétegeket az alacsonyabb szintű rétegek szolgáltatásaira építve. A kidolgozott eljárások nemcsak a programozásban, hanem például az üzleti élet tervezési folyamataiban is alkalmazhatók.

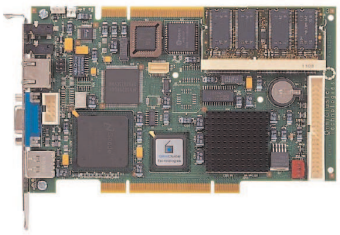
Ugyancsak az ő munkájuk eredménye az objektumorientált programfejlesztési elgondolás, valamint a Beta, egy korai, általános célú objektumorientált programozási nyelv kifejlesztése is.

➲ http://slashdot.org/articles/02/02/07/034201.shtml

Az OmniCluster nyerte a LinuxWorld „Best Productivity" kategóriadíját

Az OmniCluster olyan kiszolgálókat fejleszt, amelyek egyetlen PCI-foglalatú kártyán helyezkednek el. A kártyán kap helyet a központi processzor, a memória, illetve minden, a merevlemezekkel vagy a külvilággal történő kapcsolattartáshoz szükséges vezérlő.

A kiszolgálók x86-os processzorokat, változattól függően National Geode vagy Intel Pentium III lapkákat hordoznak. Ennek köszönhetően igény szerint



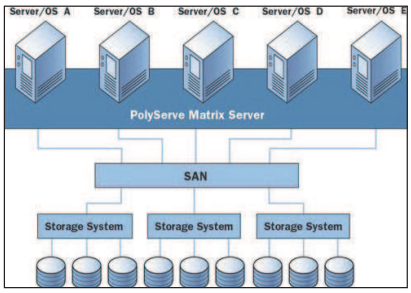
Linux, FreeBSD vagy Windows operációs rendszer futhat rajtuk. A kártyák saját merevlemezeket kaphatnak, vagy osztozhatnak a gazdagép merevlemezein. A kiszolgálók fürtözhetők egymással, így akár egyetlen számítógépházban egész kiszolgálótelep hozható létre – az alkalmazási lehetőségeknek csak a vásárlók képzelete szab határt. A kártyák önálló 10/100 Mb/s sebességű ethernetcsatolóval rendelkeznek, illetve változattól függően SVGA-, USB-, hang- és további ethernetcsatlakozókat is kaphatnak. A gazdagéppel és egymással gigabit sebességű, a gazdagép PCI buszán keresztül létesített kapcsolat segítségével tartják a kapcsolatot.

➲ http://www.omnicluster.com/

Mátrixos feldolgozás linuxos gépekkel

A PolyServe Matrix Server nevű programja lehetővé teszi, hogy Intel processzoros, Linux operációs rendszert futtató számítógépek segítségével nagyméretű, közös tárolóeszközzel rendelkező kiszolgálófarmot hozzunk létre.

A nem csak a PolyServe által mátrixos feldolgozásnak nevezett elképzelés szerint a viszonylag kis teljesítményű csomópontok nagysebességű, intelligens



hálózati kapcsolatok révén csatlakoznak egymáshoz. Megfelelő programmal – például az Oracle9i Real Application Clusters névre hallgató fürtözött adatbázis-kezelő megoldásával – rendkívül rugalmas, ugyanakkor a nagygépek teljesítményét elérő számítási képességű rendszer hozható létre. Természetesen itt is a költségeken van a hangsúly, a fejlesztők elképzelései szerint az így kiépített rendszer teljes birtoklási költsége alacsonyabb, mint egyetlen nagygépé.
➲ http://www.polyserve.com/

160 GB egyetlen merevlemezen

Csak választani kell, hogy 120 vagy 160 GB legyen a mérete annak a merevlemeznek, amely a Maxtor külső, USB-

csatlakozós egységében bújik meg. A 160 GB-os Personal Storage 3000XT a talán kevésbé elterjedt IEEE 1394-es buszt használja, a 120 GB-os 3000LE viszont USB 2.0 csatlakozóval rendelkezik. Míg a 300 és 400 dolláros külső merevlemezekért aligha lesz hazánkban tolongás, figyelemre méltó, hogy a Maxtor 120 GB-os belső merevlemeze már a hazai árlistákon is feltűnt: körülbelül 60 000 forint plusz áfa áron szerezhető be. A hatalmas, ATA/133 felületre csatlakozó merevlemez korongjai 5400-as fordulatszámmal pörögnek, átlagos elérési ideje 12 ms alatti.
➲ http://www.maxtor.com

IBM-nagygépek Linuxszal

Az IBM bejelentette, hogy olyan nagygépeket dob piacra, amelyeket kifejezetten Linux operációs rendszerrel szállítanak. Az IBM gépei eddig is támogatták a Linuxot, ám ahogy – a gazdasági helyzet kedvezőtlen alakulása miatt – a vásárló cégek egyre szorosabbra húzzák a nadrágszíjat, egyre inkább az ingyenes operációs rendszer kerül előtérbe.

Az IBM részéről úgy vélik, hogy a nagygépek piaca újra feltámad. A 60-as és 70-es években a nagygépek uralták a számítógépek piacát, ám az olcsó PC-s kiszolgálók később háttérbe szorították őket. A kék óriás részéről úgy látják, hogy olcsóbb és egyszerűbb egyetlen nagygépen összevonni a vállalat kiszolgálóit, mint esetleg több száz gépet telepíteni és felügyelni. Az IBM eServer zSeries sorozata az iparág legnagyobb növekedésnek örvendő típusa volt, az eladások az elmúlt öt negyedévben töretlenül növekedtek.

Az iSeries sorozatot a kisebb, a zSeries sorozatot a nagyobb szervezeteknek ajánlják, utóbbival akár több száz meglévő kiszolgáló is kiváltható.
➲ http://www.ibm.com/

Megjelentek a GeForce4-alapú 3D-kártyák

Megjelentek az nVidia GeForce4 grafikus lapkái. A gyártók sem késlekedtek sokáig, többek közt a Creative, az Asus és a Leadtek is rendelkezik már az új lapkákra épülő termékekkel, amelyek minden bizonnyal rövidesen a hazai üzletekben is feltűnnek – egyetlen aggályunk csak az, hogy vajon mennyit kell majd értük fizetnünk.

Az új lapkák a cég állítása szerint elődjeik teljesítményének négyszeresét nyújtják. A zűrzavar fokozása érdekében az GeForce4 esetében sem egyetlen lapkáról van szó, hanem teljes sorozatról beszélhetünk. A GeForce4 Ti 4600-as, 4400-as és 4200-as típusok nyújtják a legnagyobb teljesítményt, az olcsóbb MX 460-as, 440-es és 420-as inkább az „érték” kategóriában számíthatnak figyelemre, míg a 440 Go és a 420 Go a hordozható gépek piacára készül.
➲ http://www.nvidia.com

A Linux nem veheti fel a verseny a Windowszal?

Bob Young, a Red Hat elnöke legalábbis így véli. A Red Hatnél a fejlesztések hangsúlya egyre inkább a nagyvállalati megoldások, valamint a beágyazott termékek felé tolódnak. Szó sincs róla azonban, hogy a Red Hat bármit is feladna – egyszerűen csak túllép a ma nehézségein és a jövőre helyezi a hangsúlyt.

A Microsoft operációs rendszerének több mint húsz évvel ezelőtti sikeres elterjedését annak köszönhette, hogy egyfajta műszaki korszakváltás következett be: a nagygépek mellett egyre több személyi számítógép jelent meg. Hasonló korszakváltásra számíthatunk most is, amikor az Internet lassan otthonunk minden szegletébe elér, és egyre kisebb szerep jut a hagyományos személyi számítógépeknek; helyüket különféle intelligens, beágyazott, többek közt a hálózat használatát is lehetővé tévő eszközök veszik át. Éppen ez az a terület, ahol a Linux is megtalálhatja a maga helyét – persze nem szabad elfeledkezni arról sem, hogy a kihívásokra .NET névvel a nagy vetélytárs is megfogalmazta a maga válaszát.
➲ http://zdnet.com.com/2100-1104-828802.html

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

Asus 8600

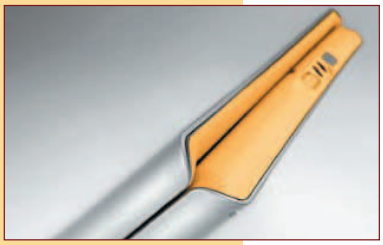
Miért választ az ember hordozható számítógépet a kényelmes PC helyett? Mert sokat utazik, vagy mert nem fér el más otthon az asztalon, esetleg mert a munkahelyén kapta, és így megtakaríthatja egy otthoni számítógép megvásárlását. A „miért” szinte mellékes is, mert egy ilyen gép megvásárlásakor a legfontosabb szempont általában az ár, a méret, a felszereltség, a kényelem és a programok elérhetősége az adott modellre. Szinte pontosan megegyező feltételek, mint egy új gépkocsi vásárlásakor, bár számítógépet nem 5–10 évre vásárol az ember, mint autót. Ezt ugyanúgy érdemes jól megfontolni, ugyanis ha túl nehéz vagy túl gyenge gépet veszünk, a lassú és rozoga kocsihoz hasonlóan nem fogjuk tudni használni. Sokkal nehezebb helyzetben van azonban egy átlagos Linux-felhasználó, hiszen a legtöbb gyártó nem tünteti fel, hogy eszköze milyen mértékben működik együtt a Linuxszal. Ezért amikor notebookot szeretnénk vásárolni, nagyon körültekintően kell szétnézünk a piacon, és meg kell kísérelnünk felmérni, hogy az adott kiépítés linuxos szemmel nézve is rendben van-e. Egy másik lehetőség, hogy széjjelnézünk olyan ismerőseink háza táján, akik notebookjukat Linuxszal használják. Az igazi gond azonban ott kezdődik, ha új típusra vagy modellre vágyunk. Ilyenkor igazán kockázatos, hogy a megközelítőleg ötszázezer forintért vásárolt gép vajon minden eleme és szolgáltatása működik-e Linuxszal. E kérdéskört illetően igyekszem felvilágosítással szolgálni, hiszen akaratos természetemnek és újságíró mivoltomnak köszönhetően vásárlás előtt a kiszemelt Asus 8600-as Notebookot megkaptam egy hét tesztelésre. Fontos kérdés és kevésbé reklám, hogy miért pont az Asust kértem. Gondos munkával összeírtam egy papírra, hogy milyen igényeim vannak egy hordozható számítógéppel kapcsolatban. Íme legfontosabb szempontjaim:

- ne kelljen MS-termékeket vennem hozzá,
- igazán kicsi legyen, ugyanakkor kellemes legyen dolgozni rajta,
- lehessen rajta DVD-t, divixet és egyéb filmeket nézni,
- minden egyes eszköze működjön Linux alatt,
- utoljára, de nem utolsó sorban nettó ára négyszázezer forint körül legyen.

Ezután következett az árlisták összegyűjtése 21. századi recept szerint: végy egy ➲ http://www.google.net-et, adj hozzá reguláris kifejezéseket és kulcsszavakat, mindezt lassú tűzön böngészd! Az eredményt újabb szűkítések és kulcsszavak hozzáadásával tedd ropogósabbá végül a kész listát ízlés szerint tálald. A rostán fennakadt modellek: Ibm, Asus, Gericom, Portocom, Sony. Az ár–teljesítmény viszony gondos elemzése után sajnos máris csupán kétversenyzőssé szűkült a döntő, innen pedig egyenes út vezetett az 1,7 kg össztömegű Asus 8600-as modellhez. Ezekután már csak a forgalmazó felkutatása és meggyőzése volt hátra, amely nem bizonyult nehéz feladatnak, mert a Sowah Kft. munkatársai az első pillanatban elismerték, hogy nem bővelkednek tapasztalatban a Linux–Asus párosítás kapcsán, de szeretnék, ha a helyzet megváltozna. Így egy frissen csomagolt géppel távoztam a helyszínről, abban a boldog tudatban, hogy most körülbelül egy hétig nem fogok aludni, hiszen az ember addig nem nyugszik, amíg „jövendőbelijén”minden rendesen nem működik. A profi felhasználó pontosan ugyanannyira utál kézikönyvet (értsd „Hogyan üzemeljük be gépünket” leírást) olvasni, mint amennyire a kézikönyvírók utálnak kézikönyvet írni, amiből természetesen az következik, hogy körülbelül húsz perccel előbb jöttem volna rá, hogyan kell a külső CD segítségével indítani (boot) a gépet, ha a hozzá adott füzetet mégiscsak elolvasom. Egy Debian Woody legfrissebb változatát telepítettem a gépre különösebb zökkenő nélkül, bár ez valószínűleg annak köszönhető, hogy az elején nem kellett a PCMCIA-kártyák támogatásával szórakozni, hiszen Rtl-8139-megfelelő alaplapi hálózati kártya van a gépben. Első lelkesedésemben a 2.2.20-as rendszermaggal (a legfrissebb üzembiztos mag a 2.2-es rendszermagfában) kezdtem neki a kipróbálásnak. Gyakorlatilag két-három rendezgetés, fordítás, szöszmötölés árán több-kevesebb sikerrel csaknem minden eszközt

sikerült beüzemelnem. Ekkor döntöttem úgy, hogy bár a 2.2.20-as az utolsó, a kiszolgálókhoz is melegen ajánlott rendszermag, én a 2.4.17-es ugyancsak megbízható, új rendszermagot fogom előnyben részesíteni, hiszen minden újdonságot tartalmaz. Mindenkinek tudom ajánlani, hiszen rengeteg olyan új eszközmeghajtó van az új rendszermagban, amely a 2.2.x-es sorozatnak nem része, így például a hangkártyát 2.2 alatt csak külön ALSA-meghajtó segítségével lehetett volna használni, míg a 2.4 alatt már támogatott.
A rendszermagfordításra nem érdemes több szót vesztegetni, hiszen a CD-mellékleten megtalálható 28. CD Magazin/Asus könyvtárában szerepel a mentett *.config* állomány. Inkább nézzük meg az X-kiszolgáló támogatottságát! A Woody rendszerben lévő 4.1-es kiszolgáló teljes mértékben támogatja a beépített Silicon Motion, Inc. SM720 Lynx3DM videokártyát. Legalábbis elsőre gyönyörűen ment az alap xserver-svga csomaggal. A gondok akkor kezdődtek, amikor Divix/DVD-filmeket szerettem volna rajta nézni. Kis méretben zökkenőmentesen ment, de amint teljes képernyősre nagyítottam, elkezdett szaggatni mind a kép, mind a hang.

Sok időt eltöltöttem az MPplayer-lejátszó különböző alkatrészeinek és programjainak testreszabásával, de a megoldást az xserver-xfree86 csomag telepítése adta. Bár az xserver-svga is támogatta a kártyát, az alkatrészes gyorsítórészét nem. Természetesen a *XF86Config-4* beállítófájl is a korongra került (28. CD Magazin/Asus), hogy ne kelljen a keresgélésre időt pazarolni. Ezekután még a DVD-lejátszás is gyönyörűen működött, pedig amikor nekikezdtem az egésznek, a Weben azt olvastam egy



Asus-oldalon, hogy az adott géppel Linux alatt nem fog menni a DVD-lejátszás. Erre örömmel cáfolok rá. Éppen akkor, amikor már azt hittem, hogy ennél szebb és jobb nem lehet semmi, rájöttem, hogy soros kapu bizony nincs a géphez, csak két USB. Ugyan az Asusnál lehet körülbelül 30 ezer forintért kapusokszorozót kapni, amely 1 párhuzamos, 2 soros és 2 ps/2-es kaput tartalmaz, de a gyári készletben volt párhuzamos USB-átalakító, soros azonban sajnos nem. A gyári harmincezres kapusokszorozó további hibája, hogy meglehetősen nagy doboz, amelyet az ember nem szívesen hord egy ilyen kis gép mellett. Lázasan kutattam a Google oldalain a megoldás után, mígnem eszembe jutott a ➲ http://www.alphasonic.hu címen található nagykereskedés meglátogatása, és egy kicsi USB soros átalakító megvásárlása, bár ennek ára is tízezer forint körül mozog. A kereskedésben figyelmeztettek, hogy ez a kapu Windows alatt csak modemmel és egyéb hasonló soros eszközzel tud jól együttműködni, az egérrel nem. A bevásárlást lázas rendszermagfordítás követte, és teljes eredménytelenséggel járt – már éppen vittem volna vissza a kereskedésbe a szerkezetet, amikor is a nagy Manitu a kezemet ismét a Google felé vezette. Nem tettem mást, mint a termék dobozára írt típusjellemzőket szóról szóra begépeltem. Az eredmény egy, azaz egyetlen német nyelven írt levél volt. Miután leápoltam ínhüvelygyulladásomat, amelyet a német-magyar szótár pörgetésétől kaptam, már tudtam is, hogy az `USB_SERIAL_PL2303` modulra van szükségem. Innentől egyenes út vezetett Nokia telefonom soros kapura kötéséig és a szolgáltatóhoz való betárcsázásig, vagy akár egy másik Linux `mgetty`-vel kiajánlott konzoljára való csatlakozásáig. Egy szó, mint száz, a kapu azóta is teljes értékű soros kapuként üzemel.

Amikor már azt hittem, hogy nincs előttem lehetetlen, még a 2.4.17 rendszermag hibáival kellett szembesülnöm. Az első nehézséget az okozta, hogy az óra folyamatosan késett, hiába hangoltam össze naponta többször is az atomórával. A hiba a rendszermag 2.4-es részében van, méghozzá a CMOS és az RTC (real time clock) körül. Erre jelent megoldást az adjtimex csomag, amely a rendszer késésre hajlamos óráját CMOS-órához igazítja. A másik gond valószínűleg nem túl ismert, de annál érdekesebb. Nagyon sokan használnak CF-kártyás eszközt, például fényképezőgépet. Ha CF-memóriánk tartalmát menteni szeretnénk, megtehetjük például a soros USB-kapun keresztül a kamerához adott kábel és program segítségével, ez azonban lassú. Erre nagyon jól bevált eszköz a PCMCIA CF-adapter, amely körülbelül 2400–5000 forintos áron bárhol kapható. A kártyát az adapterbe helyezve adatainkat az IDE sín sebességével menthetjük vagy írhatjuk ki. Én az SSH- (távoli felügyelet lehetősége) kulcsaimat tárolom CF-kártyán. Visszatérve fentebb vázolt nehézségünkhöz, a 2.4-es rendszermagban egy elírás okozza a hibát, amely az ehhez tartozó modult helytelenül `ide_cs.o`-nak hívja `ide-cs.o` helyett.

Bár egyszerűnek hangzik, a hibára rábukkanni több napba telt. A megoldást a következő programocska lefuttatása jelenti:

```
#!/bin/sh
"cd /etc/pcmcia
cp config config.bak
sed -e 's/ide_cs/
➥ide-cs/' <config.bak >
➥config
```

A rendszer így máris */dev/hdc1*-nek látta a kártyát.

Mindezek után bárkinek merem ajánlani az Asus 8600 sorozatát, mert kb. 2–3 órás munkával nagyon kényelmesen használható linuxos eszközt varázsolhat belőle.

Az Asus 8600 adatai

Tömeg: 1,7 kg
Processzor: Pentium III 850MHz
Memória: 192 MB RAM
Merevlemez: 20 GB
Meghajtó: DVD/CD-ROM külső

Kapcsolódó címek

Asus–Linux ➲ http://www.linux-on-laptops.com/asus.html
Mplayer ➲ http://www.mplayerhq.hu
Sowah ➲ http://www.sowah.hu
Alphasonic ➲ http://www.alphasonic.hu

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

Találkozás Costa Rica tudományos és technológiai miniszterével

Beszélgetés a távközlésről, a Linuxról és a nyílt forrású fejlesztések Costa Rica-i jövőjéről.

Nemrég az a megtiszteltetés ért, hogy *Guy F. de Téramond*-nal, Costa Rica tudományos és technológiai miniszterével találkozhattam. Don Guy – ahogyan közeli munkatársai hívják – a Linux és a nyílt forrású fejlesztések elkötelezett híve. Mivel a minisztereket az elnök nevezi ki, elmondhatjuk, hogy az ottani kormányzatban egy igen komoly Linux-hívő is helyet kapott.

Guy F. de Téramond

Először 2001 júniusában találkoztunk a Costa Rica-i Linux-felhasználók Egyesületének (GULCR) találkozóján. Akkor igencsak elkápráztatott engem azzal, hogy milyen komolyan kiáll a Linux és az állampolgárok számára biztosítandó jó minőségű internetes kapcsolatok mellett. E találkozónk jó alkalmat kínált arra, hogy érdekeivel és szakmai hitvallásával közelebbről is megismerkedhessem.
Minisztériumi irodájában találkoztunk. Megérkezésemkor éppen egy számítógépes probléma megoldásán töprengett: új rendszermagot fordított Mandrake-alapú laptopjához, és még mielőtt meggyőződött volna róla, hogy az új működőképes lesz, törölte a régit. Ezt is egy „szükséges rossz tapasztalatként" könyvelte el, és beszélgetésünk után ismét belevetette magát a Mandrake 8.1 CD-ről történő frissítésébe.
Guy eredetileg kutatóorvosként szerzett diplomát Párizsban, ahol 1968 és 1977 között tanult. Egy évet töltött a Harvardon, majd szintén egy évet a Stanfordon. Guggenheim-ösztöndíjas (1986), elnyerte a Fulbright kutatási díját (1983), a Clodomiro Picado Twight nemzeti kitüntetést (1979) és az űrkutatók szövetségének díját is (1997). A Costa Rica-i Egyetem fizikusprofesszora volt 1982-től egészen 2000-ig, amikor is felkérték a miniszteri posztra. Tagja az Amerikai Fizikustársaságnak és alapító tagja a Costa Rica-i Nemzeti Tudományos Akadémiának. Folytathatnám a sort, de azt hiszem, Olvasóim előtt ebből a felsorolásból is kirajzolódik a megfelelő kép.
Ő felügyelte Costa Rica csatlakozását a BITNET-hez (1990), majd az Internethez is (1993). Amikor azt kérdeztem tőle, hogy először miért a BITNET-re szavazott, azt felelte, hogy az Internet akkoriban leginkább az Egyesült Államok számítógépes szakembereinek hálózata volt, míg a BITNET más tudományos és szakmai területek szakemberei számára is biztosította az együttműködést.
Amikor visszatért Costa Ricába, meggyőződött róla, hogy a távközlési kapcsolatok óriási lehetőségeket biztosíthatnak az országa számára, így megszervezte a CRNet kiépítését, mely a Costa Rica-i felsőoktatási és kutatóintézmények között létrehozott, üvegszálas kábelekből és útvonalválasztókból álló hálózat. Küldetéstudata szó szerint nem ismert határokat, hiszen részt vett Nicaragua, Panama, Jamaica, Honduras és Guatemala Internethez történő csatlakoztatásában is.
Legkedvesebb szenvedélyeiről is kérdeztem. Első helyen az elméleti fizikát, második helyen a számítógépeket és az Intenetet említette. A dobogó harmadik fokát firtatva azonban csak mosolygott, és kitért a válasz elől…
A CRNet kiépítése számára azt jelentette, hogy olyan hálózati kapcsolat jöhetett létre, amelynek köszönhetően visszatérhetett elméleti fizikai kutatásaihoz. A számítástechnika területén felhalmozott tudása és tapasztalatai azonban továbbra is sokkal szorosabban kötötték ehhez a területhez, mint azt korábban gondolta volna. Amit eddig tett, valamint amit a jövőben szándékozik tenni, hatalmas előrelépést jelentenek Costa Rica számára.

Guy és a Unix

Az első Costa Rica-i Unix-rendszer egy IBM RISC gép volt, melyen 1991-ben Guy az elsők között gyakorolhatott. Csapattársa volt *Mario Guerra* is, aki azóta Unix-szakemberré vált, és a ma is a MICIT-nál dolgozik.
Guy nemsokára bekapcsolódott a közép-amerikai országok internetes csatlakozásának kialakításába. Ez sok utazással járt a környező országokba, Nicaraguába, Jamaicába, ahol a kezdeti felkészülésben segédkezett. Mindig az volt a cél, hogy egy helyi szakembergárda felkészítésével saját szárnyaira bocsássa az adott ország távközlésének fejlődését. „Együtt dolgoztunk az emberekkel, mert ha csak odamegyünk és felállítjuk a rendszert, akkor senki nem tanul semmit, és ha baj van, minket keresnek. Nekünk nem ez volt a célunk, ilyet legfeljebb egy magánvállalkozás tenne."
„A CRNethez 24 egyetemet és tíz kormányzati hivatalt csatlakoztattunk" – mondta. „Mindig az volt a tervünk, hogy számítógépes szakemberekkel dolgozzunk együtt, mert ők gyorsabban tanulnak. Így, bár a CRNetnél nagyon kevesen voltunk, rengeteg munkát sikerült elvégeznünk."
Ez a munka folytatódott, s a további célok között a Costa Rica-i sávszélesség növelése is szerepelt. Ezáltal egy PanAmSat műholdas kapcsolat és egy kizárólag letöltésre használt vonal jött létre. Mario feladata az volt, hogy proxik felállításával csökkentse a bejövő adatok sávszélességigényét.

Jelenlegi feladatok

Miniszteri kinevezése után Guy a megkezdett munkát nemzeti szinten kívánta folytatni – terve az volt, hogy a nagy sávszélességű otthoni internetezést minden állampolgár számára elérhetővé teszi. Costa Rica 1,2 millió dolláros adományt kapott, melyet egy DSL-próbahálózat kialakítására használhatott fel. Ennek eredményeképpen Costa Rica 240 telefonkörzetéből ötben a DSL szolgáltatás elérhetővé vált. „A meglévő infrastruktúrát alkalmazzuk, a logikai elemeket csatlakoztatjuk az üvegszálra, majd a rézhuzalokat használjuk” – mondta. „A DSL fantasztikus technológia, mert nincs szükség a vonalak átépítésére – az ember azt használja, ami van.”

„Most a második lépcsőfok következik” – tette hozzá. „Ennek két fontos része van. Százezer DSL-vonalat építünk ki az ország mind a 240 telefonkörzetében.”

Az egy főre jutó nagy sávszélességű vonalak alapján a világ első országa Dél-Korea, a második Kanada, a harmadik az Egyesült Államok. A százezer vonallal, mely az összes rézvezetékes végpont 10 százalékát (és a teljes lakosság 2,5 százalékát) jelenti, a feljesztés eredményeképpen Costa Rica a harmadik helyre lép föl.

A vállalkozás teljes költsége 60 000 000 dollár. Ez a befektetés óriási összeg Costa Rica számára, ennek ellenére kedvező, hiszen a DSL a meglévő üvegszálas és rézhuzalos hálózatot használja. A DSL bevezetése a helyi telefonközpontok terhelését is csökkenti, hiszen az általában hosszú ideig tartó internetes hívásokat a DSL veszi át. „Nagyon figyelünk a jól méretezhető technológiákra, nem a frame relay-re, az ATM-re és társaira” – mondta Guy.

Phil Hughes, Guy F. de Téramond és Mario Guerra a kamerák előtt

Rámutatott, hogy ez az általános bekapcsolódás azért mehet végbe, mert a kormányzat a távközlésben monopóliummal rendelkezik. Az ICE telefonszolgáltató tulajdonképpen kormányzati hivatal, így érthető, hogy a nagy sávszélességű internetezést miért a DSL segítségével oldják meg. Ha ezt a tervet egy magáncég hajtaná végre, nem lenne ennyire átfogó, hiszen a kis népsűrűségű területeken a vállalkozás nem lenne kifizetődő.

Ez ahhoz az esethez hasonlít, ami az Egyesült Államokban történt a REA-val, mely az elektromosságot juttatta el szinte mindenkihez. Ma a Costa Rica-i háztartások 97 százalékában van elektromos áram, melyet egyébként teljes mértékben természetes erőforrásokból állítanak elő. A „Villanyt minden háztartásba!” jelmondat mai megfelelője pedig mi más lehetne, ha nem a „Nagy sávszélességű internetelérést mindenkinek!”.

Ahogy Guy mondta: „Sokféle munkát végezhetünk el otthoni irodánkból is, így több időt tölthetünk a családunkkal, és ehhez természetesen jó internetes kapcsolatra van szükség.” Mindemellett egyre több Costa Rica-i költözik a nagyvárosokba, ami az ilyen helyek (például San José) közlekedését terheli, és nem kis mértékű környezetszennyezést is okoz. Ezáltal az ország távközlésének fejlesztése még fontosabbá válik.

Az üvegszálas és rézhuzalos hálózat mellett az elektromos vezetékeken történő adattovábbítással is kísérleteznek. Az ICE az ország fő áramszolgáltatója is, így például a meglévő optikai hálózat kábelei mentén futó, a hálózat felügyeletéhez használatos üvegszálas kábeleket is felhasználhatják az optikai rendszer kisegítő hálózataként.

A belső hálózatnak természetesen jó minőségű internetes kapcsolattal is kell rendelkeznie, és ez az igény vezetett az Arcos megszületéséhez. Az Arcos a karibi térség nemzetközi gyűrűs hálózata, melyhez Mexikóban, Közép-Amerika minden országában, a karib-tengeri szigeteken és Miamiban leágazásokat építettek. Sávszélessége nagyjából 1 terabájt/másodperc, és a rendszer minden útvonalválasztója a szárazföldön található, ami azt eredményezi, hogy a hibák sokkal olcsóbban javíthatók, mint a tengeren épített útvonalválasztó állomások esetében.

Amikor Olvasóink e cikket a kezükben tartják, az Arcos már elkészült. Kérdésemre, hogy miért az Arcos mellett döntöttek, Guy azt válaszolta, hogy ez a módszer sokkal olcsóbb sávszélességet biztosít, mintha műholdas kapcsolatot építettek volna ki. Azt is hangsúlyozta, hogy a fejlesztés célja az Internetet mindenkihez eljuttatni: magánszemélyekhez, iskolákhoz, kórházakhoz, de a bankokhoz és az üzleti központokhoz is.

A rendszer szolgáltatásainak árairól máris viták folynak, és mindenki a további csökkentéssel ért egyet. Az eredeti tervekben 30 dolláros havidíj szerepelt a 64 kbites

DSL, 40 dollár/hó a 128 kbites vonalhoz stb. Az árban a vonal és a szolgáltatás költsége egyaránt szerepel. Mivel az ICE nem profitközpontú vállalkozás, az árak várhatóan nem sokkal lesznek magasabbak, mint amennyit a rendszer fenntartása indokol.
A terv mérnöki része elkészült. A rendszer átméretezhető és a biztonsági szolgáltatások is egyszerűen megvalósíthatók rajta. A tervezéskor gondosan ügyeltek az ITU-előírások betartására, így ezzel biztosan nem lesz probléma. A további szükséges berendezésekre az árajánlatkérés már megtörtént, a kezdeti költségeket pedig a kormányzat máris magára vállalta. A cél az, hogy 2002 januárjában megkezdhessék az első végpontok kiépítését.

Mi a helyzet az ingyenes programokkal?

Tudván, hogy Guy a Linux szerelmese, a nyílt forrású programok Costa Rica-i jövőjéről kérdeztem. Az alábbiakban ezt a beszélgetést olvashatják.

Phil: *Mi a szerepe ebben az egészben a Linuxnak? Az új internetes fejlesztések egy queposi lakos (ez a város a tengerparton fekszik, távol a fővárostól) számára lehetővé teszik, hogy számítógépes tanácsadóvá váljon. Van még valamilyen terve arra nézvést, hogyan hozhatná közelebb a Linuxot ehhez a közösséghez?*

Guy: Az egyetemen a különböző technológiák egész tárháza érhető el mindenki számára. A felhasználók maguk dönthetnek arról, hogy milyen operációs rendszert használnak, a legtöbben mégis a Windows, esetleg a Mac rendszert választották, melyekhez felhasználói szinten értenek. A Linuxra csupán a matematikai és a mérnöki tanszékek néhány megszállottja szavazott.
A központban körülbelül 100 kiszolgálónk található. Mindegyiken Linux fut, kivéve az adatbázisokat, melyek Solaris-alapú SPARC-gépeken futnak.
Úgy gondolom, hogy a Linux az utóbbi két-három évben nagyon sokat mozdult el az egyszerű otthoni, irodai felhasználók igényei felé. Ha ismét rétegekben gondolkodunk, láthatjuk, hogy legalul főleg a nyílt forrású nagyrendszerek, legfelül pedig a főként nem ingyenes programokat, operációs rendszereket alkalmazó felhasználók állnak.
A következő néhány évben derül ki, hogy ez a határvonal feljebb vagy inkább lejjebb húzódik. Ez nagymértékben attól függ, hogy a Linux és a nyílt forrású programok közössége elegendő energiát fektet-e be a végfelhasználó számára szükséges alkalmazások kifejlesztésébe.

Phil: *Ezzel egyetértek.*

Guy: Ez tehát mozgó határvonal. Természetesen mára csaknem minden nyílt forrásúvá vált. Ott vannak például a TCP/IP-protokoll kifejlesztői vagy *Tim Berners-Lee*, a HTML megteremtője. Ők mindannyian hatalmas eredményeket értek el az emberiség számára.
És egyébként ki ma a nyílt forrású fejlesztések legnagyobb felhasználója? A Microsoft, például a TCP/IP-vel kapcsolatos húzásaik miatt. Hiába adták ki a Microsoft Networköt, mindenki tudja, hogy ez valójában a TCP/IP volt.

Phil: *És ha nem is így lenne, akkor sem beszélhetünk egy túlságosan használható valamiről.*

Guy: Pontosan. 1995-ben a Microsoft protokollja még a NetBEUI és a NetBIOS volt, és egyiket sem lehetett útvonalazni. Ezek LAN-protokollok. Aztán egyszer csak hoztak egy fontos döntést és megpróbálták beolvasztani a TCP/IP-t. Egyet kell értenünk azzal, hogy könnyű a nyílt forrású fejlesztéseket sátániak kikiáltani, de ők maguk is ezeket használják föl.
Adott tehát ez a mozgó határvonal. Ha egy kicsit feljebb is szétnézünk, ott a webkiszolgálók, az Apache–IIS háború található. Az Apache nemcsak a piac 60 százalékát birtokolja, de ingyenes és megbízható is. És ne hagyjuk figyelmen kívül a Gartner Group biztonsági jelentéseit sem, amelyeket nem szabad viccnek tekintenünk.
Nagyjából ezzel válaszolnék a kérdésére. Csak egy példa: itt a minisztériumban minden kiszolgáló Linuxon fut. Ezt a továbbiakban a kormányzati munka még több elemére ki kívánjuk terjeszteni. Jómagam Linuxot használok a mindennapi munkámhoz, de a legtöbb mérnök is így van ezzel. A rengeteg vírus miatt meg sem fordul a fejemben, hogy visszatérjek a Windowshoz.
Az éremnek azonban két oldala van. Én módosított rendszermagot használok, mert tényleg szükségem van a Microsoft Office-ra. Őszintének kell lennünk: a Word és az Excel szabványnak számít. A nyílt forrású közösségben ezt a szintet még nem értük el.

Phil: *Még „Linux Torvalds" is Microsoft-alkalmazásokat használ, például PowerPointot. Nem tudnak igazán használható és működőképes operációs rendszert írni, viszont kitűnő programokat készítenek.*

Guy: Pontosan. A módosított rendszermagom tökéletesen működik, s így a PowerPoint, Excel és Word programokat gördülékenyen használhatom, ugyanis nem lenne rá időm, hogy a különféle formátumok között alakítgassak. De az egész Windows csupán egy program az operációs rendszeremen, így nem kell amiatt aggódnom, hogy vírusos-e vagy sem.
Mindkét „világ" legjobb elemeit gyűjtöttem össze és ezeket használom. Ezek kiegészítik egymást és szerintem ez így egészséges.
Országunk igen magas szintre jutott el a programfejlesztés területén – ez az egyik legfontosabb iparágunk, ezért természetesen támogatnunk kell.
Térjünk rá a második kérdésre, egészen pontosan arra, hogy mit tervezünk e területen. Itt a minisztériumban a mérnökök, jómagam és a Linux-hívők biztosan nem fognak visszatérni a kereskedelmi operációs rendszerek világába. Tudományos munkáimat LaTeX-ben készítem, abban minden megtalálható.
Jelenleg nem hozhatunk olyan döntést, hogy az egész kormányzati számítógépes rendszer működését nyílt forrású programokra bízzuk. Rengeteg titkárnő érezné úgy, hogy hátráltatják a munkájában, és nem tudnának olyan gyorsan dolgozni.
A jövőben a helyzet meg fog változni, de ez egy folyamat. Jelenleg a levelező, a webkiszolgálók és az adatbá-

zisok Linux alatt futnak. Az egyik mérnökünk képes távolról felügyelni az összes kiszolgálót. A rendszer hatékony, biztonságos, így elmondhatjuk, hogy a világ egyik legjobb számítógépes környezete van a birtokunkban. Ma még nem működik minden minisztériumban Linux, idáig még nem jutottunk el, viszont eljön az a nap, amikor a szabványos irodai alkalmazások teljesen beleolvadnak a linuxos asztali környezetbe. Erre azonban még várnunk kell.

Phil: *Nos, azt hiszem, ezzel meg is válaszolta a következő kérdésemet, ami az lett volna, hogy vajon az a helyes út, ha mi, a Linux-közösség egyre több és jobb minőségű eszközöket készítünk az átlagos felhasználók számára, és az áttörés egyszer bekövetkezik?*

Guy: Mindenképpen. Van ennek egy másik oldala is: az oktatás, az iskolák, a nyílt forrású fejlesztések pedagógiai haszna óriási.

A tanulóknak természetesen el kell sajátítaniuk a Word, az Excel és a hasonló kereskedelmi programok használatát, de a felsőbb osztályokban meg kell értetni a diákokkal, hogy a nyílt forrású fejlesztések egy távolabbi horizont felé mutatnak: itt a kód az első sortól az utolsóig mindenki számára hozzáférhető, a program ezáltal teljes egészében kiismerhető és továbbfejleszthető, és ez az, ami csodálatos.

A program sorról sorra nyomon követhető, megváltoztatható, bővíthető, egyszerűsíthető. Iparunknak ebből óriási haszna származhat, hiszen rengeteg kiváló programozónk van. Ezért mondtam, hogy a nyílt forrású fejlesztések nagy hatással lehetnek az oktatás módszertanára is. Ez nagyon közel áll a tudományos modellhez, ahol az eredmények nyilvánosak és szabadon megvitathatók. Csak a legjobb kód „éli túl" a fejlesztéseket.

Szeretném, ha a nyílt forrású fejlesztések a középiskolás diákokhoz is eljutnának. Ezt szeretném elősegíteni, de törekvéseim sikere az oktatási tárcától is függ. Szeretném látni, hogy éles eszű gyerekek ülnek le egy gép elé, változtatnak a programsorokon (közben persze hibákat is elkövetnek – én is így kezdtem), és valami újat hoznak létre.

Phil: *Igen, én is azt tapasztaltam, hogy ha a hozzáértő emberek a programok mellett azok forráskódját is tanulmányozhatják, rögvest sokkal nagyobb kedvük lesz elmélyülni, majd részt venni a fejlesztésben. Innen hogyan tovább?*

Guy: A kérdés nem az, hogy mit csináljunk, hanem hogy mikor. Ahogy már említettem, az álmom az volt, hogy a szakemberek, bárhol legyenek is, részt vehessenek a legmagasabb szintű kutatásokban. Most már természetesen nagy örömmel figyelem a szélesebb körű hálózat fejlődését is. Ez jól kiegészíti saját terveimet a fizikai kutatások területén. Számomra jó terepet jelent ez a fejlesztés.

Hadd említsek egy példát annak alátámasztására, hogy milyen fontos az oktatás ezen a területen. A fejlesztés kezdetekor viták folytak arról, hogy a rendszer gerincét vezeték nélküli vagy üvegszálas hálózat képezze-e. Aki jártas a fizikában, az ismeri a mágikus 1015-ös számot, mely az üvegszálas hálózat frekvenciájának együtthatója. Így az üvegszálas hálózaton történő adattovábbítás jóval szélesebb sávon történhet, mint a vezeték nélküli megoldásoknál, ahol ez a szám 106 és 108 között van.

Phil: *Igen tiszteletre méltó az Ön és a Costa Rica-i kormányzat elkötelezettsége az ügy érdekében. Az Egyesült Államokban rengeteg vita folyik arról, hogy a kormányzatnak mit kellene támogatnia. Vegyük például az egészségügyi ellátást: az egészséges emberek termékenyebbek a munkában. Ugyanígy, ha a kormányzat támogat egy ilyen kiterjedt fejlesztést, azzal rengeteg ember boldogulását segítheti elő.*

Guy: Azt hiszem, az egészségügy valóban jó példa. Nálunk a védelem mindenkire kiterjed, mi nem hagyunk senkit sem az utcán meghalni.

Phil: *Volt még egy kérdés a tarsolyomban arról, hogy a használt programok hány százaléka nyílt forrású, de úgy tűnik, hogy Ön szerint a linuxos közösség erőfeszítéseinek hatására előbb vagy utóbb bekövetkezik az áttörés, mégpedig önmagától.*

Guy: Így van. Az utóbbi néhány évben elért eredmények magukért beszélnek, de természetesen nincs megállás, folytatnunk kell a munkát.

Összegzés

Nagyon élveztem a Don Guyjal folytatott beszélgetést. Őszinte lelkesedésével és aktív hozzáállásával lenyűgözött. Elkötelezettségének ékes bizonyítéka országában az olcsó, gyors internetes kapcsolat és az oktatás fejlődése. Abban, hogy ez egyáltalán elindulhatott, az egész Costa Rica-i kormányzatnak is igen nagy érdemei vannak. Említettem neki, hogy az Embedded Linux Journal első játékának nyertesei számára Costa Rica-i utazást sorsoltunk ki, mire azt válaszolta, hogy reméli, majd valamilyen „hivatalos" segítséget is tud nekik adni. Azzal zárnám tehát e sorokat, hogy a Linux és a nyílt forrású fejlesztés él és virul Costa Ricában, és úgy tűnik, befolyása könnyen kiterjedhet a környező országokra is. Remélem, Costa Rica példaként szolgál a Linux közép-amerikai terjesztéséhez.

Phil Hughes (phil@ssc.com) a Linux Journal kiadója, elkötelezett Linux-őrült. 1993 óta rengeteg embert nyert meg a Linux számára.

Kapcsolódó címek

A Costa Rica-i Linux-felhasználók csoportja
➲ http://www.linux.or.cr
A DSL-fejlesztés honlapja
➲ http://www.micit.go.cr/redia/1_etapa.html
A kész terv ➲ http://www.micit.go.cr/redia/usuario.html
La Nacion ➲ http://www.nacion.com
A jelenlegi állapot
➲ http://www.micit.go.cr/redia/cobertura.html
Az ország helyezése a kapcsolatok száma alapján
➲ http://www.micit.go.cr/redia/ranking.html

Lapzárta után: DMCA – szólásszabadság: 2:0, de van még remény

A 2001. decemberi lapzártát követő pár napban súlyos jogi csapásokat szenvedett a szólásszabadság, a Linux-társadalom szívügye, amely nélkül valószínűleg nem lenne sem Linux, sem Internet. Az egyetlen reménysugarat az FTC (Amerikai Szövetségi Kereskedelmi Bizottság) jelenti.

Mindkét ügyben a DMCA (Digital Millennium Copyright Act – digitális szerzői jogi törvény) győzedelmeskedett. A DMCA, amelyet 1998 októberében iktattak törvénybe, kiterjeszti a szerzői jog hatáskörét, s egyebek közt a szerzői jogi szabályok áthágását *bűntettnek* minősíti.

A *Felten kontra RIAA* ügyben professzor *Edward Felten* (aki jelenleg nem érhető el a Stanfordon) és mások pert indítottak a Recording Industry Association of America (Amerikai Kiadók Szövetsége) ellen, azt állítva, hogy a RIAA perbe fogással fenyegette a professzort abban az esetben, ha a kutatásait egy tudományos konferencián bemutatja. Felten arról tartott volna előadást, hogy kutatótársaival együtt hogyan sikerült megkerülniük a digitális vízjeleket, de visszalépett, miután a RIAA ügyvédjétől fenyegető levelet kapott. A RIAA tisztviselői később azt nyilatkozták, hogy nem állt szándékukban pert indítani. A kerületi bíróság bírája, *Garrett E. Brown* a keresetet azzal az indoklással utasította el, hogy Felten panasza nem volt jogos.

Az ügy előzménye: *Dmitry Sklyarov* orosz akadémikust bebörtönözték és a mai napig a büntetőtárgyalásra vár. A vád ellene, hogy megszegte a DMCA-t, amikor egy konferencián részletesen bemutatta az Adobe eBook technológiai programjának gyenge pontjait. Az Adobe kezdeményezésére 2001. július 16-án Sklyarovot a hotelszobájában, Las Vegasban letartóztatták, amint éppen haza készült utazni, Oroszországba. Három hetet töltött börtönben, és a tárgyalásra azóta sem került sor, noha az Adobe időközben megvonta támogatását az ügytől.

Cary Sherman, a RIAA rangidős ügyvezető alelnöke így nyilatkozott: „Örömmel tapasztaljuk, hogy a bíróság felismerte, amit kezdettől fogva hangoztattunk: szó sincs jogvitáról. Amint azt többször is elmondtuk, Felten professzor szabadon publikálhatja a kutatási eredményeit”.

A per lezárását követő sajtónyilatkozatában *Cindy Cohn*, az Electronic Frontier Foundation (EFF, Elektronikus Határokért Alapítvány) jogi igazgatója ezt mondta: „Miután a kormány és az iparág a DMCA szabályozásait eltérően értelmezi, nem meglepő, ha a tudósok és a kutatók összezavarodnak, és inkább elállnak kutatásaik közlésétől attól való félelmükben, hogy a DMCA miatt pert akasztanak a nyakukba…”

Függetlenül bizonyos a kormány, illetve az iparág részéről történt korábbi fenyegetésektől, a tudósoknak nem kellene újból és újból megtapasztalniuk ennek a pontatlan szerzői jogi törvénynek a dermesztő hatását.

A *Universal kontra Reimerdes* perben az USA Másodfokú Fellebbviteli Bírósága jóváhagyta a kerületi bíróságnak az alperest elmarasztaló ítéletét. A felperesek a Universal Studios, a Tri-Star, a Disney, a 20th Century Fox, a Paramount, a Columbia Pictures és az MGM voltak. Az alperesek pedig *Eric Corley* és a 2600 magazin, amely kiadta a DeCSS nevű DVD-dekódoló programot – ami egyébként a világhálón széles körben elterjedt; tulajdonképpen a 2600 is a letöltést biztosította.

A DeCSS segítségével a Linux-felhasználók is nézhetnek DVD-t. Corley ügyvédje azzal érvelt, hogy a DeCSS a szólásszabadság védelme alá esik. A stúdiók viszont azt bizonygatták, hogy az iparágnak okozott kár többet nyom a latban, mint a szólásszabadság nyújtotta védelem. A bíróság a panaszosoknak adott igazat. Cindy Cohn (EFF), aki segített Corley-t képviselni az ügyben, így kommentálta a döntést: „Véleményem szerint ez a szólásszabadság csorbítása. Úgy tűnik, a bíróság támogatja a folyóirat internetes cenzúrázását”.

Az sem mellékes, hogy – *Eric S. Raymond*, az Open Source Initiative tagjának szavaival élve – a DMCA védi a lejátszó eszközök kartellizálását. *Jon Johansen* és a norvég MoRE (Masters of Reverse Engineering) alapvetően azzal a céllal fejlesztették ki a DeCSS-t, hogy linuxos eszközökön is lehessen DVD-t lejátszani. Ezzel a területtel a filmstúdiók és a fogyasztói elektronika piacán mozgó üzlettársaik addig nem foglalkoztak.

Az egyetlen jó hír, hogy a US Federal Trade Commission (Amerikai Szövetségi Kereskedelmi Bizottság) bejelentette, hogy 2002 januárjában vizsgálatot indít az alábbi címmel: „Verseny és a szellemi tulajdont érintő törvények és irányelvek a tudásalapú gazdaságban”. A Federal Register honlapja (➲ www.ftc.gov/os/2001/11/ciphearingsfrn.htm) további tájékoztatást nyújt, valamint vélemények írásbeli beadását is lehetővé teszi.

Doc Searls

Működj, görgő!

Az egér görgőjét a legtöbb gépen könnyedén munkára bírhatjuk, csupán olyan protokollt kell beállítani, amelyik támogatja a görgők kezelését.

Ilyen például az IntelliMouse-meghajtó.

Rendszergazdaként nyisd meg a */etc/X11/XF86Config-4* fájlt, és az egér részénél a protokollt állítsd be `imPS/2`-re. Fontos még, hogy a `ZaxisMapping` értéke is helyes legyen!

```
Section "InputDevice"
    Driver   "mouse"
    Option   "Protocol"       "imPS/2"
    Option   "ZAxisMapping"   "4 5"
```

Szy György

Lapzárta után: DMCA – szólásszabadság: 2:0, de van még remény

A 2001. decemberi lapzártát követő pár napban súlyos jogi csapásokat szenvedett a szólásszabadság, a Linux-társadalom szívügye, amely nélkül valószínűleg nem lenne sem Linux, sem Internet. Az egyetlen reménysugarat az FTC (Amerikai Szövetségi Kereskedelmi Bizottság) jelenti.

Mindkét ügyben a DMCA (Digital Millennium Copyright Act – digitális szerzői jogi törvény) győzedelmeskedett. A DMCA, amelyet 1998 októberében iktattak törvénybe, kiterjeszti a szerzői jog hatáskörét, s egyebek közt a szerzői jogi szabályok áthágását *bűntettnek* minősíti.

A *Felten kontra RIAA* ügyben professzor *Edward Felten* (aki jelenleg nem érhető el a Stanfordon) és mások pert indítottak a Recording Industry Association of America (Amerikai Kiadók Szövetsége) ellen, azt állítva, hogy a RIAA perbe fogással fenyegette a professzort abban az esetben, ha a kutatásait egy tudományos konferencián bemutatja. Felten arról tartott volna előadást, hogy kutatótársaival együtt hogyan sikerült megkerülniük a digitális vízjeleket, de visszalépett, miután a RIAA ügyvédjétől fenyegető levelet kapott. A RIAA tisztviselői később azt nyilatkozták, hogy nem állt szándékukban pert indítani. A kerületi bíróság bírája, *Garrett E. Brown* a keresetet azzal az indoklással utasította el, hogy Felten panasza nem volt jogos.

Az ügy előzménye: *Dmitry Sklyarov* orosz akadémikust bebörtönözték és a mai napig a büntetőtárgyalásra vár. A vád ellene, hogy megszegte a DMCA-t, amikor egy konferencián részletesen bemutatta az Adobe eBook technológiai programjának gyenge pontjait. Az Adobe kezdeményezésére 2001. július 16-án Sklyarovot a hotelszobájában, Las Vegasban letartóztatták, amint éppen haza készült utazni, Oroszországba. Három hetet töltött börtönben, és a tárgyalásra azóta sem került sor, noha az Adobe időközben megvonta támogatását az ügytől.

Cary Sherman, a RIAA rangidős ügyvezető alelnöke így nyilatkozott: „Örömmel tapasztaljuk, hogy a bíróság felismerte, amit kezdettől fogva hangoztattunk: szó sincs jogvitáról. Amint azt többször is elmondtuk, Felten professzor szabadon publikálhatja a kutatási eredményeit”.

A per lezárását követő sajtónyilatkozatában *Cindy Cohn*, az Electronic Frontier Foundation (EFF, Elektronikus Határokért Alapítvány) jogi igazgatója ezt mondta: „Miután a kormány és az iparág a DMCA szabályozásait eltérően értelmezi, nem meglepő, ha a tudósok és a kutatók összezavarodnak, és inkább elállnak kutatásaik közlésétől attól való félelmükben, hogy a DMCA miatt pert akasztanak a nyakukba…”

Függetlenül bizonyos a kormány, illetve az iparág részéről történt korábbi fenyegetésektől, a tudósoknak nem kellene újból és újból megtapasztalniuk ennek a pontatlan szerzői jogi törvénynek a dermesztő hatását.

A *Universal kontra Reimerdes* perben az USA Másodfokú Fellebbviteli Bírósága jóváhagyta a kerületi bíróságnak az alperest elmarasztaló ítéletét. A felperesek a Universal Studios, a Tri-Star, a Disney, a 20th Century Fox, a Paramount, a Columbia Pictures és az MGM voltak. Az alperesek pedig *Eric Corley* és a 2600 magazin, amely kiadta a DeCSS nevű DVD-dekódoló programot – ami egyébként a világhálón széles körben elterjedt; tulajdonképpen a 2600 is a letöltést biztosította.

A DeCSS segítségével a Linux-felhasználók is nézhetnek DVD-t. Corley ügyvédje azzal érvelt, hogy a DeCSS a szólásszabadság védelme alá esik. A stúdiók viszont azt bizonygatták, hogy az iparágnak okozott kár többet nyom a latban, mint a szólásszabadság nyújtotta védelem. A bíróság a panaszosoknak adott igazat. Cindy Cohn (EFF), aki segített Corley-t képviselni az ügyben, így kommentálta a döntést: „Véleményem szerint ez a szólásszabadság csorbítása. Úgy tűnik, a bíróság támogatja a folyóirat internetes cenzúrázását”.

Az sem mellékes, hogy – *Eric S. Raymond*, az Open Source Initiative tagjának szavaival élve – a DMCA védi a lejátszó eszközök kartellizálását. *Jon Johansen* és a norvég MoRE (Masters of Reverse Engineering) alapvetően azzal a céllal fejlesztették ki a DeCSS-t, hogy linuxos eszközökön is lehessen DVD-t lejátszani. Ezzel a területtel a filmstúdiók és a fogyasztói elektronika piacán mozgó üzlettársaik addig nem foglalkoztak.

Az egyetlen jó hír, hogy a US Federal Trade Commission (Amerikai Szövetségi Kereskedelmi Bizottság) bejelentette, hogy 2002 januárjában vizsgálatot indít az alábbi címmel: „Verseny és a szellemi tulajdont érintő törvények és irányelvek a tudásalapú gazdaságban”. A Federal Register honlapja (➲ www.ftc.gov/os/2001/11/ciphearingsfrn.htm) további tájékoztatást nyújt, valamint vélemények írásbeli beadását is lehetővé teszi.

Doc Searls

Működj, görgő!

Az egér görgőjét a legtöbb gépen könnyedén munkára bírhatjuk, csupán olyan protokollt kell beállítani, amelyik támogatja a görgők kezelését.

Ilyen például az IntelliMouse-meghajtó.

Rendszergazdaként nyisd meg a */etc/X11/XF86Config-4* fájlt, és az egér részénél a protokollt állítsd be `imPS/2`-re. Fontos még, hogy a `ZaxisMapping` értéke is helyes legyen!

```
Section "InputDevice"
    Driver   "mouse"
    Option   "Protocol"       "imPS/2"
    Option   "ZAxisMapping"   "4 5"
```

Szy György

Beágyazott Linux és Java – a jövő fuvallata?

Rick a közeljövőben várható Java-Linux párosítást használó intelligens eszközök robbanásszerű elterjedéséről elmélkedik.

Az intelligens eszközök világa gyökeresen átalakul. A minket körülvevő számítógépes eszközök egyre okosabbak, egyre többféle kapcsolatra képesek, egyre önállóbbak – és egyre többen lesznek. A történések pedig egyre sebesebben követik egymást… Idézhetnénk Moore törvényét is, de tény, hogy jelenleg különböző teljesítményű processzorok és változatos csatlakozási lehetőségek kerülnek gyakorlatilag mindenbe, ami árammal működik, legyen akár helyhez kötött, akár mobileszköz.

A HP Digital Entertainment Centere

Az irányzat alapját a magas fokon egybeépített, a teljes rendszert egyetlen lapkán tartalmazó processzorok jelentik, amelyekhez nagyméretű rendszermemória és lemezes tárterület társul, és amelyekhez vezetékes és vezeték nélküli adattovábbítási felületek – ethernet, IrDA, 802.11, Bluetooth – tartoznak.

További fontos jelenség, hogy mind a beágyazott feldolgozás, mind a kapcsolat egyre inkább „szétkenődik", az eszközök intelligenciája egyre kevésbé köthető helyhez. Az eszközökben futó programok pontos helyét egyre nehezebb meghatározni, és jobban meggondolva: tulajdonképpen minket is egyre kevésbé érdekel, hogy az adott program pontosan hol található. Talán a készülékben fut az alkalmazás? Esetleg valamilyen távoli kiszolgálót, például egy otthoni szolgáltatásokat nyújtó átjárót vesz igénybe? Lehet, hogy egy internetes alkalmazásszolgáltatónál érhető el? Talán mindhárom módszert felhasználja?

Nevezhetjük ezt elosztott intelligenciának vagy elosztott adatfeldolgozásnak. Lehet .NET a neve. Mondhatjuk, hogy ez a PC utáni korszak. Teljesen mindegy, hogy minek nevezzük, egyvalami biztos: az elkülönült, önálló, helyben található programokat futtató asztali számítógépek korszaka hamarosan letűnik, akár a jégkorszak mamutjaié.

Üdvözlünk a PC utáni korszakban!

Ahogyan a hagyományos számítógépes felfogás határai egyre inkább elmosódnak és újfajta, elosztott, összekapcsolt és mindenre kiterjedő világ bontakozik ki, az új korszak néhány jellegzetessége hamar megragadja a figyelmünket.

- Az intelligens eszközök száma – például a beágyazott operációs rendszerrel rendelkezőké – exponenciálisan fog nőni, hamarosan elérve az egymilliárdot.
- A processzorok kiválasztása sokkal inkább a költségektől, semmint a műszaki megoldástól és a felépítéstől függ majd.
- Gyakorlatilag minden eszköz rendelkezni fog valamilyen kapcsolattal – akár vezetékessel, akár vezeték nélkülivel.
- Új gépi szintű program (firmware) vagy új alkalmazások letöltésével a legtöbb eszköz esetében lehetőség lesz a távoli frissítésre vagy javításra.
- Az eszközök jellemzően egy-egy feladatra és nem általános célokra lesznek használhatók, így alkalmazásaikat is inkább a gyártó, és nem a felhasználó fogja kiválasztani.

Elmondható, hogy az új korszak eszközei jellemzően nem személyi számítógépek lesznek – sokkal inkább különféle képességű és jellemzőkkel bíró intelligens eszközök, amelyeket tájékozódásra, szórakozásra, felügyeletre és egyéb célokra fogunk használni. Elég csak az apró karórákra gondolni, amelyekbe mobiltelefont és személyi adatkezelő szolgáltatásokat építettek, vagy a legújabb mobiltelefon-zsebtitkár készülékekre, az egyesített hang- és mozgóképrendszerekre, a biztonsági rendszerekre, az autós kiegészítőkre, az egyre okosabb konyhai gépekre, esetleg a személyi számítógép jellegű asztali terminálokra. A lista egyre csak gyarapodik.

A Linux-mag termékeny földre hullik

A beágyazott eszközök számának és sokszínűségének robbanásszerű növekedésével párhuzamosan növekszik az igény az eszközök célraszabottsága és előállítási költségének csökkentése iránt. A beágyazott Linux ezért méretezhetőségének, testreszabhatóságának és kedvező árának köszönhetően egyre kedveltebbé fog válni.

Eddig viszonylag kevés figyelmet kapott, hiszen a Linux futtatásához szükséges memória és processzor magasabb költsége lényeges tényezőt jelentett, különösen költségérzékeny esetekben. Most azonban a beágyazott Linux használatához szükséges erőforrások – hozzávetőleg 2 MB flash- és 4 MB RAM memória, valamint egy átlagos teljesítményű processzor – Moore törvényének köszönhetően viszonylag olcsók lettek.

A Java

Az új kor – amelyben egyre több, egyre okosabb, egymással kapcsolatot teremtő eszköz vesz majd körül minket – úgyszintén fontos kihívása lesz, hogy le kell egyszerűsíteni és fel kell gyorsítani az alkalmazások fejlesztését, telepítését, valamint karbantartását. Valószínű, hogy a Java egyre nagyobb szerephez fog jutni.

A Java ugyan nem érte el fejlesztésének eredeti célját, amely, némi gúnnyal szólva, az intelligens eszközök operációs rendszereként való elterjedése lenne, de lehetővé tette, hogy alkalmazásokat egyszerűen mozgassunk a különböző eszközök között. Az eredményben a Web hihetetlen mértékű elterjedése nem kis szerepet játszott.

A Java jelenleg – beágyazott operációs rendszerként való gyenge szereplése ellenére – ígéretes választásnak

tűnik, ha olyan eszközfüggetlen felületet keresünk, amely beágyazott operációs rendszer felett fut. Ebben az esetben a Java ugyan nem operációs rendszerként szolgál, de képes az alsóbb rétegek egyedi jellegzetességeinek elfedésére, és jelentősen kibővítheti az operációs rendszer által nyújtott szolgáltatások körét.
Ha figyelembe vesszük az intelligens eszközök számának exponenciális mértékű emelkedését, a Java kézenfekvő megoldást nyújt, ha az egyes készüléktípusokhoz kötődő fejlesztési munkát a lehető legkisebbre akarjuk csökkenteni, és inkább a tervezet lényegi pontjaira szeretnénk összpontosítani. A Java segítségével grafikus felhasználói felületek, webböngészők, protokollvermek, kézírás- és beszédfelismerők, vezeték nélküli csatlakozások, multimédiás felületek, adatbázis-kezelők és távoli szolgáltatások széles körének kényelme válik elérhetővé.

Néhány szereplő az élvonalból

A beágyazott Linux és Java párosításra épülő termékek egyik érdekes példája az a szórakoztatóközpont, amelyet nemrég jelentett be a Hewlett-Packard. A HP Digital Entertainment Center alapvetően otthoni szórakoztató-elektronikai készülék, amely digitális zenéket és tudnivalókat szállít a nappalinkba széles sávú internetkapcsolat és az otthoni hálózat segítségével, személyi számítógép közreműködése nélkül. A rendszer a már meglévő otthoni hangrendszerhez hangösszetevőként csatlakozik, és saját CD-k írására alkalmas; tulajdonosa nagyméretű merevlemezén MP3-fájlokat készíthet, tárolhat és rendezhet, a zenéket digitális zenelejátszóra töltheti át, valamint internetes rádióadókat hallgathat vele.
„A HP a Linuxot nyílt forrású jellegének és széles körű támogatottságának köszönhetően karolta fel fogyasztói készülékeiben" – mondta el *William Woo*, a Hewlett-Packard Embedded Software Operation vezetője. Majd így folytatta:
„Igényeinknek megfelelően módosítottuk a Linuxot, hogy a HP beágyazott operációs rendszereként tudjuk használni, majd HP Chai megoldásunkkal házasítva beágyazott Java-alkalmazásokkal és webes kapcsolattal kiegészített megoldást készítettünk. A HP Chai támogatja a Java-alkalmazásokat, amelyekkel vásárlóink számára élvezetes elektronikus szolgáltatásokat biztosíthatunk."
Egy másik figyelemre méltó Linux-alapú eszköz, amely támogatja a Java-alkalmazásokat, természetesen a Sharp Zaurus névre hallgató zsebtitkára. „A Zaurus SL-5000D korlátlan lehetőségeket kínál a fejlesztők számára" – állítja *Steve Petix*, a Sharp Mobile & IT Solutions Group alelnöke. „Örömmel működünk együtt azokkal a Linux- és Java-fejlesztőkkel, akik a következő nemzedékbe tartozó mobilalkalmazásokat készítenek erre az új, nagyteljesítményű felületre."

A jövő fuvallata?

Az új, PC utáni korszakban egyre több és többféle intelligens, egymással kapcsolatot teremteni képes számítástechnikai eszköz vesz minket körül. A kifinomultabb rendszerfelületek és protokollok fejlesztésével, karbantartásával és támogatásával kapcsolatos kihívások azonban ugyanilyen ütemben bővülnek.

Kapcsolódó címek

Az alábbiakban néhány Java és Java-szerű virtuális gépet és fordítót soroltunk fel, melyeket egyre gyakrabban használnak a Linux-alapú intelligens eszközök és beágyazott rendszerek fejlesztésekor.
Chai Appliance Platform (HP) ➲ http://www.hp.com
GCJ-, GNU-fordító Java programozási nyelvhez (Red Hat)
➲ http://www.redhat.com
Embedded PERC 2.2, Javával történő használatra is alkalmas környezet (NewMonics) ➲ http://www.newmonics.com
intent Java Technology Edition (Tao) ➲ http://tao-group.com
Java 2 Platform, Micro Edition – CDC (Sun Microsystems)
➲ http://java.sun.com
Jbed Micro Editon CLDC JVM (készíti az esmertec)
➲ http://www.esmertec.com
Jeode-felület, Java-megoldás beágyazott Linuxra (Insignia)
➲ http://www.insignia.com
JRun, beágyazható J2EE-együttműködésre kész Java-alkalmazáskiszolgáló Macromedia ➲ http://www.allaire.com
JTime, valós idejű Java virtuális gép és API (TimeSys)
➲ http://www.timesys.com
Kaffe, nyílt forrású Java-megvalósítás (Transvirtual Technologies)
➲ http://www.kaffe.org
VisualAge Micro Edition (IBM Object Technology, Inc.)
➲ http://www.embedded.oti.com
Waba, nyílt forrású, Java jellegű felület kisméretű eszközökhöz (Wabasoft)
➲ http://waba.sourceforge.net
Wonka, nyílt forrású beágyazott JVM- és osztálykönyvtár (ACUNIA)
➲ http://wonka.acunia.com

Ebből a szempontból nézve a Java – mint kiváló támogatásnak örvendő alkalmazás- és szolgáltatásfejlesztési környezet – növekvő népszerűségével használatra kész programösszetevők elképesztő serege áll majd rendelkezésünkre, amelyeket beágyazott Linuxon azonnal munkára foghatunk, így a készülékek fejlesztése egyszerűsödhet, tudásuk pedig tovább bővülhet.

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.

Hogyan bánjunk el a programkód-szabadalmakkal?

A Nyílt Forráskód Közösségének számos tagja idegenkedik a programszabadalmaktól. Úgy vélik, hogy a szabadalmak hátráltatják a programozás művészetének terjedését, és így a nyílt és mindenki számára hozzáférhető kód áldásos hatásait ássák alá. A szerzői jog megsértésének veszélye sokkal kisebb, mint egy szabadalom megsértésének veszélye. A szerzői jog megsértését könnyen elkerülhetjük, ha tiszta lappal indulunk és a szerzői jog védelme alatt álló program helyett saját változatot írunk.

Ezzel szemben egy programszabadalom megakadályozhatja, hogy szabadalmazott találmányt állítsunk elő, használjunk vagy árusítsunk, még abban az esetben is, ha nem a feltaláló programját másoljuk. Ez azt jelenti, hogy nem feltétlenül leszünk képesek elkerülni a szabadalom megsértését, bármilyen óvatosan járunk is el. Jelentkezhet valaki, akiről még csak nem is hallottunk, és közölheti, hogy az ő szabadalmát használjuk. Hacsak nem tudjuk egy újabb találmánnyal kikerülni a szabadalommal védett találmányt, nyílt forráskódú fejlesztésünk ezen a ponton véget is ér.

Tetszik vagy sem, a szabadalmazott programok intézménye a valóság része. Az amerikai kongresszus és a bíróságok számos esetben áldásukat adták rá, csakúgy, mint sok más ország törvénykezése. Számolva ezekkel a tényekkel fontos megértenünk a szabadalommal védett programkódnak a felhasználási szerződésekre vonatkozó következményeit, hogy kiválaszthassuk a felfogásunknak és a céljainknak megfelelő szerződést.

A nyílt forráskódú programok terjesztési szerződésének szemszögéből háromfajta szabadalmat veszünk figyelembe:

1. a terjesztési szerződés kibocsátójának szabadalmait,
2. harmadik fél által birtokolt szabadalmakat és
3. a program felhasználási szerződését elfogadó fél, vagy a programmal kapcsolatos jogokat rajta keresztül megszerző felek tulajdonában lévő szabadalmakat.

A terjesztő szabadalmai: tegyük fel, hogy elfogadtuk egy nyílt forrású program felhasználási szerződését, és rájövünk, hogy a terjesztő a programmal kapcsolatban szabadalommal is rendelkezik, amelyre a szerződés nem terjed ki. Ennek hiányában nem készíthetünk, használhatunk és terjeszthetünk saját változatot. Valahányszor egy felhasználó felkérésére átnézek egy felhasználási szerződést, meggyőződöm róla, hogy van-e kifejezett engedély „a szerződés kibocsátójának tulajdonában jelenleg lévő, vagy a továbbiakban bejegyzett szabadalmakra való hivatkozás esetében is a programot vagy annak részeit elkészíteni, használni, árusítani, árusításra felkínálni, elkészíttetni, illetve egyéb módon rendelkezni felette". Sok nyílt forrású felhasználási szerződés, beleértve a BSD-szerződést is, nem tartalmaz ehhez hasonló rendelkezést. A szerződés ilyen esetben szintén magával vonhatja a szabadalom felhasználási jogának átadását, de nem javaslom, hogy ilyen hallgatólagos rendelkezésre hagyatkozzunk. Amikor csak lehetséges, törekedjünk rá, hogy kifejezett engedéllyel rendelkezzünk a programmal kapcsolatos szabadalmak felhasználására.

Harmadik fél által birtokolt szabadalmak: egy program terjesztőjének nem feltétlenül van tudomása minden, a programot érintő szabadalomról. Egy harmadik fél váratlanul bejelentheti, hogy olyan szabadalommal rendelkezik, amely a program bizonyos részeire terjed ki. Egy szabadalmi jogot sértő program felhasználójaként pedig lehet, hogy a felhasználási szerződés ellenére is kénytelenek leszünk felhagyni a program használatával. Ennek a helyzetnek a kezelésére a kereskedelmi terjesztésű programok szerződése gyakran egy jóvátételi záradékot tartalmaz, amelynek értelmében a terjesztő a szerződött felet biztosítja a harmadik fél által támasztott, szabadalmakkal kapcsolatos követelésekkel szemben. Ígérheti a jogdíj visszatérítését, a szabadalmi jogot nem sértő változat rendelkezésre bocsátását vagy a szabadalmi jog megvásárlását harmadik fél által birtokolt szabadalmak fölmerülése esetén. A nyílt forrású szerződések általában nem tartalmaznak jóvátételi záradékot, mivel a nyílt forráskódú programok terjesztői többnyire nem szednek jogdíjat, amelyből a jóvátétellel kapcsolatos költségeket fedezhetnék. A legtöbb nyílt forráskódú program terjesztése „úgy, ahogy van", szabadalmi jogok megsértésének elkerülésére vonatkozó garanciák nélkül történik. Nyíltforráskód-felhasználók, vigyázat! A harmadik fél szabadalmaival kapcsolatos kockázatot általában a felhasználó viseli.

A felhasználó szabadalmai: ez a kérdés sokkal kényesebb, mint a terjesztő és a harmadik felek szabadalmának esete. A terjesztők gyakran foglalnak a szerződésbe úgynevezett „szabadalomretorzió" záradékot, hogy megóvják magukat a felhasználó által ellenük fordított szabadalmaktól. Mivel ez a kérdéskör heves indulatokat vált ki, érdemes körültekintő megfontolás tárgyává tenni – ezért ennek a témának külön oldalt szentelek.

Bármi legyen is a felfogásunk a szabadalmazott programkóddal kapcsolatban, fontos megértenünk, hogy a szabadalmaknak milyen hatásuk lehet a programok terjesztési szerződéseire nézve. Nem elég csak annyit mondani, hogy „ki nem állhatom a programok szabadalmaztatását". Akár a felhasználási szerződés kibocsátójaként, akár annak elfogadójaként meg kell győződnünk róla, hogy a szerződés megfelelően fejezi-e ki a szabadalmakkal kapcsolatos felfogásunkat.

Lawrence Rosen

A szabadalmak elleni záradékok formáiról

A felhasználó szabadalmait bővebben fejtem ki, mert ez a tárgykör fogósabb, mint a többi. Itt a szerződés elfogadójának, vagy ami a leszármazott programok szempontjából talán még fontosabb, a nyílt forráskódú programmal kapcsolatos jogokat rajta keresztül megszerzők szabadalmairól van szó. A terjesztési szerződés kibocsátója a jogsértés vádja elleni védekezés céljából a szerződésben gyakran szabadalom elleni záradékot helyez el. Egy ilyen záradék lényegében azt mondja ki, hogy amennyiben a felhasználó a program terjesztőjét a szabadalmi jog megsértésének címén bepereli, a felhasználási szerződés azonnal érvényét veszti. A felhasználó nem perelheti a terjesztőt a szabadalom megsértéséért, miközben a programot is használja.

Egy nyílt forráskódú program terjesztője így igazolta a szabadalmak elleni záradék létjogosultságát: „Ha biztosítjuk, hogy a szabadalmakat a jogvédelem eszközeként használják, segíthet visszaszorítani az igazságtalanságokat".

A szabadalom elleni záradékok két fő típusa közül az egyik az úgynevezett gyenge változat, ami valahogy így hangzik: ha a program felhasználója programmal kapcsolatos szabadalmi jogot próbál velem szemben érvényesíteni, a felhasználási szerződés érvényét veszti. Ugyanez a biztosíték érvényes a leszármazott programváltozatok felhasználóira vonatkozóan is: ha egy felhasználó bepereli az eredeti program terjesztőjét, az ő szerződése is érvényét veszti.

Személy szerint a szabadalom elleni záradékok gyenge változatát támogatom, mivel tisztességes egyensúlyt teremt a terjesztő és a felhasználó érdekei között. Ezáltal a felhasználó és a programmal kapcsolatos jogokat rajta keresztül megszerző felek nem húzhatnak hasznot a nyílt forráskódú és szabad programból, miközben a terjesztőt jogdíj fizetésére kötelezik egy olyan szabadalom címén, amelyet éppen a szóban forgó programban használ fel.

A szabadalom elleni záradékok úgynevezett erős változata valami ilyesmit mond: ha a terjesztési szerződés elfogadója bármiféle szabadalmi jogot próbál velem szemben érvényesíteni, a felhasználási szerződés érvényét veszti. Többnyire ebben az esetben is ugyanez a feltétel érvényes a leszármazott programváltozatok felhasználóira.

Úgy gondolom, ez az erős változat többet árt a Nyílt Forráskód Közösségének, mint amennyit a terjesztők érdekének védelmével használ, mert a jelentős szabadalomcsomaggal rendelkező cégek nem fognak olyan nyílt forráskódú programokat használni, melyek terjesztési szerződése lényegében megengedi, hogy a felhasználó (vagy az általa készített leszármazott program felhasználói) szabadalmait a terjesztő rosszhiszeműen kijátssza. Egy nyílt forrású program szerződését elfogadva akár abban a helyzetben is találhatjuk magunkat, hogy egy szabadalom elleni erős biztosíték megakadályozza származtatott programunk széles körben való elterjedését. Programunk felhasználói visszautasíthatják egy olyan „vírus" elfogadását, amely nem teszi lehetővé, hogy a programmal összefüggésben nem lévő szabadalmaikat a program eredeti terjesztőjével szemben érvényesítsék.

Az Apple például olyan cég, amely ragaszkodik hozzá, hogy felhasználási szerződéseibe szabadalom elleni erős záradékokat helyezhessen el.

Adott Apple-program felhasználójaként a program használatára csak akkor van jogunk, ha semmilyen más szabadalommal kapcsolatban sem indítunk pert az Apple ellen. Adott esetben ügyfeleimet valószínűleg figyelmeztetném arra, hogy éppen ezen okból kerüljék el az Apple nyílt forrású programjainak használatát.

A gyenge és az erős szót használtam a szabadalom elleni záradékok két változatának leírására, de ezek a szavak hétköznapi értelemben használva félreérthetőek lehetnek. A gyenge szó alatt nem azt értem, hogy erőtlen, hatástalan vagy életképtelen. Az erős szót sem a fizikai erő, illetve a gazdasági vagy pénzügyi hatékonyság értelmében használom. Inkább arra szerettem volna utalni, hogy a terjesztő érdekei védelmében milyen mértékű nyomást gyakorolhat a felhasználóra. A szabadalom elleni záradék gyenge formája csak meghatározott körülmények esetén érvényesíthető, és a szabadalmak egy viszonylag szűk körére érvényes. Ezzel szemben a szabadalom elleni záradékok erős változata a felhasználóra a programhoz nem kötődő szabadalmak érvényesítésével szemben is nyomást gyakorolhat.

A program szerződésének kibocsátójaként vagy egy felhasználási szerződés elfogadójaként egyaránt meg kell győződnünk arról, hogy a szerződésben található szabadalom elleni záradékokkal együtt tudunk-e élni. Ha felhasználóként nincs bejegyzett szabadalmunk (és nem is áll szándékunkban szabadalmakat bejegyeztetni), akkor a szabadalom elleni záradékok mindkét formája elfogadható lehet a számunkra. Másrészt ha leszármazott programokat szeretnénk továbbadni, gondoljuk meg, hogy ügyfeleink milyen szabadalom elleni záradékokat tudnak elfogadni. Terjesztőként pedig fontoljuk meg a szabadalom elleni záradék alkalmazását a felhasználók szabadalmai elleni védekezésben.

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).

A hónap szakmai tanácsai

A /dev helyreállítása

Red Hat 5.0-t használok és egy hibás parancsfájl véletlenül letörölte a */dev* könyvtáramat. Leállítottam és újraindítottam a Linuxot, de még mielőtt helyreállítottam volna a */dev*-et a mentésből. Emiatt a helyreállító lemez használatával a */dev*-et nem tudom visszatölteni a szalagról, mert nincs többé */dev/st0*. Hogyan hozhatnék létre egy alap */dev* könyvtárat a helyreállítólemez segítségével? Milyen módon állíthatom vissza a */dev*-et a `tar`-ral?
Rene, rene.diependaele@ibbbs.be

Az alapvető eszközfájloknak a helyreállító lemezen is meg kell lenniük, így a pillanatnyilag üres */dev* könyvtárba tudod másolni őket a `cp -av /dev/* /mnt/dev/` paranccsal (ha a merevlemezt a */mnt* alá fűzted be). Ezután oly módon indítsd a rendszeredet, hogy a LILO parancssorába a `linux init=/bin/bash` utasítást írod, mert ez esetben csupán nagyon kevés eszköz használata szükséges.
Marc Merlin, marc_bts@valinux.com

Az st0 helyreállításához add ki az `mknod /mnt/dev/st0 9 0` parancsot.
Christopher Wingert, cwingert@qualcomm.com

A PCMCIA kártya nem megy a 2.4-sel

A Teles PCMCIA/S0 kártya és a 2.4 rendszermag nem hajlandó együttműködni. Hiába kerestem felvilágosítást a Világhálón a hírcsoportokban és egyebütt is, sehol sem jutottam eredményre. A hírcsoportokban csupán azt vettem észre, hogy nem állok egyedül gondommal. Talán ti tudtok segíteni.
Amikor a 2.2.x rendszermagokat használtam, a Teles PCMCIA/S0 kártya gond nélkül működött. Az új rendszermag fordításakor szükség volt a PCMCIA-kártyák támogatására, és egy foltra, amelyet a ➲ http://home.wtal.de/petig/ISDN címről lehetett letölteni. Ezután minden tökéletesen ment. A 2.4 rendszermagtól kezdve ez a folt sajnos nem használható, mert nem fordul le. A 2.4 rendszermag forrásának leírása megemlíti, hogy a Teles PCMCIA/S0 kártya támogatott. Akármivel próbálkoztam, működésképtelen. Állandóan arra panaszkodik, hogy nem megfelelő az I/O-cím, hiába használom ugyanazt az IRQ-t és I/O-t, mint a 2.2.x rendszermagoknál. A másik dolog, hogy a rendszermag a kártya befűzésekor a *teles_cs.o* modult akarja betölteni, amely megegyezik a ➲ http://home.wtal.de/petig/ISDN/ címről letölthető folttal.
Tudtok segíteni valamilyen módon nekem és azoknak az embereknek, akik ezt az ISDN-kártyát szeretnék használni? Jelenleg a Red Hat 7.1-est használom az összes folttal.
Andre Seesink, a.seesink@chello.nl

Ellenőrizd, hogy a megadott IRQ- és I/O-cím még mindig érvényes-e. Előfordul, hogy más operációs rendszerek indításakor az I/O-kapuk és az IRQ-k más értékeket kapnak az azonnal használható (Plug-and-Play) eszközöknél.
Christopher Wingert, cwingert@qualcomm.com

Nekem sem volt sok szerencsém a rendszermag PCMCIA-támogatásával. *David Hinds*, a `pcmcia-cs` szerzője azt javasolta, hogy tiltsam le a PCMCIA-támogatást a rendszermagban (ehhez újra kell fordítani a rendszermagot a PCMCIA-támogatás tiltásával), és használjam az önálló `pcmcia-cs` csomagot. Amikor megtettem, a nehézség azonnal megoldódott. Talán rajtad is ez segítene.
Marc Merlin, marc_bts@valinux.com

Két kérdés, egy válasz

Hogy helyezhetem át MP3-fájljaimat a FAT32-ről a Linux ext2 fájlrendszerére?
Cougar, cougaram@home.com

Amikor Mandrake-et használtam, felismerte és befűzte a windowsos meghajtóimat, amelyek egy másik merevlemezen foglaltak helyet. Most Red Hat 7.2-t használok, és a windowsos meghajtók nem jelennek meg, nem tudom őket befűzni. Hogyan lehet befűzni a */dev/hda*-t? Jelenleg a rendszer a */dev/hdb*-re van telepítve.
Glen Kingston, gkingstun@yahoo.com

A Windows lemezrészeit be kell fűzni a Linux alá. Úgy tudhatod meg, hogy milyen Windows-lemezrészeid vannak, hogy kiadod az `fdisk -l /dev/hda` parancsot. Figyeld a Windows-lemezrészeket (FAT32 vagy NTFS), és illeszd be őket. Ha a Windows-lemezrész például a */dev/hda1*, rendszergazdaként add ki a következő parancsokat:

```
mkdir /dos
mount /dev/hda1 /dos
```

A Windows lemezrésze a */dos* könyvtár alatt fog megjelenni.
Christopher Wingert, cwingert@qualcomm.com

Nem támogatott fájlrendszer

A gépemen felváltva használok Red Hat 7.1-et és Windows 2000-t. Az első lemezrész FAT16 (hdb1), a második lemezrész NTFS (hdb2). A Linux a második merevlemezen foglal helyet, ez a másodlagos lemez (HDD). Amikor az NTFS-lemezrészt be akarom fűzni a fájlrendszerbe, az alábbi hibaüzenetet kapom: `The kernel does not support the ntfs fs.` (A rendszermag az NTFS-fájlrendszert nem támogatja.)
A rendszermag változatszáma 2.4.2-2. Minden adatom az NTFS-lemezrészen van. Szeretném a Linuxot használni elsődleges operációs rendszerként, de a lemezrészek felosztását nem akarom megváltoztatni.
Nigel Pereira, pnigel1@hotmail.com

A lemezrész befűzéséhez és eléréséhez az NTFS-támogatást bele kell fordítanod a rendszermagba, mert alapértelmezés szerint nincs benne. A rendszermag

beállításának és lefordításának menetét megtalálod a *Kernel-HOWTO*-ban, amely rendszerint a terjesztések része, vagy a linuxdoc webhelyén is megtalálható (➲ http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html). Légy óvatos, és tartsd meg az utolsó működő változatot, hogyha valami rosszul sül el, visszatérhess a kiindulási helyzethez.
A legjobb az lenne, ha azon a lemezrészen átérnél a FAT32 használatára. Ha ragaszkodsz az NTFS-hez, a rendszermagot újra kell fordítanod. Engedélyezd a kísérleti kódot, hogy az NTFS-támogatás lehetősége megjelenjen. Semmiképpen ne engedélyezd az írás támogatását, mert biztosan tönkretenné a lemezrészt.
Ben Ford, ben@kalifornia.com

Nincs meg a FontTastic, de WordPerfectet kell használnom

Red Hat 7.0-t futtatok és telepítettem a WordPerfect Office 2000-t. Amikor megpróbálom futtatni valamelyik WordPerfect Office 2000 terméket, a következő hibaüzenetet kapom:

```
Unable to add FontTastic font server
➥to the font path.
The font server is probably
➥not installed or not running.
➥Correct the problem and
➥try again.
```

(Nem sikerült a FontTastic betűkészlet-kiszolgálót hozzáadni a betűkészletek elérési útjához. A betűkészlet-kiszolgáló nem fut vagy nincs telepítve. Javítsd ki a hibát, és próbáld újra!)
Voltam a Corel weboldalán, és írtam is nekik, de a helyzet még nem oldódott meg.
James H. Birdsong, jbirdsong@orbitworld.net

A Corel WordPerfect csapata valamiért úgy döntött, hogy saját betűkészlet-kiszolgálót használ. Ez azonban az XFree86 4.x-el együtt alkalmazva nehézségeket okoz. Ez már ismert hibaforrás, a Corel-hírcsoportok átböngészésével megtalálhatod a megoldást.
Ben Ford, ben@kalifornia.com

A sok RAM bizonytalanná teheti a rendszert?

A gépemben Soyo K7V Dragon alaplap működik 1400 MHz-es Athlon processzorral, és a BIOS az `"optimized defaults"` beállításokkal bír. A gépbe 1,5 GB Nanya PC2100 memóriát raktam és telepítettem a Red Hat 7.2-t. A rendszer elindul és fut, de véletlenszerűen lefagy. A rendszernaplóban nincs üzenet, a gép csak egyszerűen újraindul… Ha az 1,5 GB-ból 512 MB-ot elveszek, a rendszer szépen működik. Sokféle terjesztést, rendszermagot és BIOS-beállítást kipróbáltam, több különböző lemezelrendezéssel (RAID-del, RAID nélkül, csak a Promise lemezvezérlő használatával, csak a Via lemezvezérlő használatával), de egyik sem vált be. A terjesztéshez adott rendszermag nem ismerte fel a via8233 déli hidat (south bridge). Elvégeztem a szükséges módosításokat a *via82cxxx.c* fájlban, és újrafordítottam a rendszermagot. A déli híd meglett, de a rendszer továbbra is lefagyott.
Végül kicseréltem a memóriát, a nehézség azonban nem hárult el. Az 512 MB DDR ott hever az asztalomon, ami nagyon zavar, ugyanis igen drága volt (199 dollár darabja). Természetesen most már olcsóbb, de jó volna, ha ki tudnám használni a gépemben rejlő lehetőségeket.
Jim Peterman, jptr@msn.com

Az összes memóriamodult kicserélted vagy csak egyet? Ha csak egyet, próbáld meg a többit is kicserélni.
Christopher Wingert, cwingert@qualcomm.com

Elképzelhető, hogy az alaplap egyik memória-csatlakozóhelye hibás, vagy az alaplap kettőnél több memória-csatlakozóhelyet nem tud megbízhatóan árammal ellátni.
Marc Merlin, marc_bts@valinux.com

Nincs DNS-em, de használnom kell a „wvdial”-t

Külső terminéladapteren keresztül próbálok kapcsolódni az internetszolgáltatómhoz. Amikor a `wvdial` kapcsolódik, a következő hibaüzeneteket kapom:

```
--> warning, can·t find addressfor
➥·suse.de·
--> warning, address lookup does not
➥work
--> Nameserver (DNS) failure, the
➥connection may not work
```

Mitko, mitak@post.com

Ellenőrizd, hogy a */etc/resolv.conf*-ban szerepel-e a következő sor:

```
nameserver aaa.bbb.ccc.ddd
```

Az *aaa.bbb.ccc.ddd* a működő DNS-kiszolgáló IP-címe legyen. A */var/log*-ban megnézheted a PPP naplóját is, talán segít behatárolni, mi is a gond.
Felipe E. Barousse Boué, fbarousse@piensa.com

Üdvözöljük nyomtatni kívánó vendégeinket!

Van egy Linuxot futtató gépem, amely DHCP-kiszolgáló, tűzfal és átjáró egyben. Ügyfeleim között számos Windows-felhasználó található. Lehetséges volna-e a linuxos gépet nyomtatókiszolgálóként is használni, hogy a windowsos ügyfelek nyomtathassanak a linuxos nyomtatókiszolgálóra, de az ügyfeleken ne kelljen nyomtatómeghajtókat telepíteni?
Danny Patel, dharmesh@yahoo.com

Én létrehoznék egy Samba-megosztást a nyomtatónak, és egy másikat a Windowshoz szükséges nyomtató-meghajtóknak.
Christopher Wingert, cwingert@qualcomm.com

Vizsgáld meg a CUPS (Common UNIX Printing System) használatának lehetőségét. Ez támogatja az IPP, LPD, SMB (Windows) és AppSocket (JetDirect) protokollokat.

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.

Tűzfal kialakítása Zorp segítségével

Tűzfalszolgáltatásokat minden fő rendszermagváltozat nyújt a 2.0 óta, bár ez volt a rendszermag egyik legváltozékonyabb része. Tűzfalunk kialakításakor folyamatosan egyre több lehetőség áll a rendelkezésünkre, ám egyvalami semmi nem változott: az átvitt, illetve megszűrt forgalomra minden rendszermag csomagok sorozataként tekint.

Az egyszerű csomagszűrők (Linux 2.0: IP Fwadm, 2.2: IP Chains), illetve az állapottartó csomagszűrők (Linux 2.4: IP Tables) jellemzője, hogy elsősorban a csomagok fejlécét vizsgálják, míg az adatrésszel nem, vagy csak kis mértékben foglalkoznak.

Hogy megértsük, mit is jelent ez pontosan, tegyünk egy kis kitérőt: az IP-alapú hálózatok egy ötrétegű hálózati modellnek feleltethetők meg (szemben az ISO OSI által leírt hét réteggel), a rétegek a következők: fizikai, adatkapcsolati, hálózati, szállítási és alkalmazási. Az alsó három réteg együtt nyújtja az IP-szolgáltatást, a szállítási réteg felel a TCP-ért, illetve az UDP-ért, az alkalmazási rétegben pedig az adott feladatnak megfelelő protokoll kap helyet.

Alkalmazási protokoll lehet például a levelek letöltésére való POP3 vagy a webezéshez használt HTTP, de minden feladathoz különböző protokoll tervezhető. Az alkalmazási protokollok között léteznek nyílt szabványúak, amelyek jól meghatározottak (HTTP, FTP), illetve teljesen zárt, ismeretlen protokollok is (Oracle SQL*Net).

A csomagszűrők az IP-verem alsó négy rétegével foglalkoznak, az alkalmazási protokollal (tehát a TCP-csomagok adatrészével) egyáltalán nem. Tekintve, hogy az alsó négy réteg fejlécei a csomag méretének körülbelül három százalékát jelentik (ethernet esetén), az ellenőrizetlen adatmennyiségek száma meglehetősen nagy.

Az IP-protokollverem alsó négy rétegének megvalósítását (a fizikaitól a szállítási rétegig) általában az operációs rendszer tartalmazza. Az alkalmazási réteget különböző, az operációs rendszerre épülő programok valósítják meg. Az alsóbb rétegek megvalósítása alaposan kipróbált, működése megbízható, hiszen minden hálózati program épít rájuk. Az alkalmazások minősége viszont gyakran elmarad ettől, több-kevesebb hibát tartalmaznak, ami sok esetben biztonsági sérülékenységhez vezethet.

A betörők a hálózaton keresztül a programok biztonsági réseit próbálják meg kihasználni, és mivel az egyetlen kapcsolat a programmal maga az alkalmazás protokollja, a biztonsági hibák kihasználása gyakran megjelenik az alkalmazási protokollban – ha például túl hosszú felhasználónevet küldünk egy kiszolgálónak, tömbtúlcsordulás következhet be.

Csomagszűrésnél jóval nagyobb biztonságot érhetünk el, ha a csomagok fejrésze mellett az alkalmazási protokollt is elemezzük, mely a csomagok adatrészében helyezkedik el (a TCP-, illetve az UDP-adatrészében).

Az ellenőrzést az alkalmazástól függetlenül egy megbízható elem végzi, amely a nem kívánatos részeket képes visszautasítani.

Ilyen ellenőrzést képesek megvalósítani az úgynevezett alkalmazásszintű átjárók, avagy a proxytűzfalak. Ilyen program például a Gauntlet, az TIS fwtk, a T.REX, illetve a Zorp is.

Telepítés

A Zorp fejlesztése Debian-rendszeren történik, ezért ezen a Linux-változaton lesz a legkönnyebb dolgunk, de természetesen más terjesztések is alkalmasak a futtatására.

Itt jegyezném meg, hogy a tűzfalakra fokozottabban áll, hogy ne futtassunk rajtuk felesleges szolgáltatásokat, és minden szempontból tegyük őket biztonságosabbá (setuid bitek eltávolítása, programok `chroot`-tal történő futtatása stb.).

A Zorp jelenleg a 2.2-es rendszermaggal működik együtt a legjobban, bár megszabott szolgáltatásokkal a 2.4-es rendszermaggal is használható (transzparencia csak egycsatornás TCP-alapú protokollokkal érhető el).

A rendszermagon kívül a következő csomagokra lesz szükségünk:

- openssl 0.9.6, glib 1.3.1 (pontosan az 1.3.1 szükséges, mivel a változtatások miatt a későbbi glib-változatokkal nem működik együtt),
- libcap 1.10, python 1.5.2 (figyeljünk rá, hogy legyen benne száltámogatás(thread)),
- python-extclass 1.2 (ExtensionClass néven is ismert és mostanában a Zope részeként terjesztik).

Ha a fenti csomagokat és a hozzájuk tartozó fejlesztői csomagokat feltelepítettük, neki is állhatunk a Zorp fordításának:

```
# tar xfz zorp-1.4.0rc17.tar.gz
# cd zorp-1.4.0rc17
# ./configure && make && make install
```

Ha a fordítás sikeresen lefutott, a Zorp programfájljait a */usr/local* könyvtárban találjuk meg. Figyeljünk, hogy a */usr/local/lib* könyvtár szerepeljen a dinamikus szerkesztő beállításában (*/etc/ld.so.conf* fájl), majd futtassuk le az `ldconfig` parancsot.

Beállítás

A Zorp az átmenő adatfolyam alkalmazásszintű elemzéséért felelős, de természetesen csomagszintű szűrést is alkalmazunk, melyet a rendszermag által nyújtott csomagszűrő szolgáltatásokkal érünk el. A csomagszűrés célja kettős:

1. megtiltjuk az átmenő forgalmat, és gondoskodunk róla, hogy a csomagtovábbítás helyett a proxykhoz kerüljön,
2. szabályozzuk a tűzfalat elérő csomagokat.

Egy Zorp-alapú tűzfal beállítása legalább a csomagszűrő-, valamint a proxyk beállításából tevődik össze (a valóságban természetesen további programokra is szükségünk lehet).

Az IP Chains segítségével alapértelmezésként mindent tiltunk, az engedélyezett forgalmat pedig egy `REDIRECT`-szabály segítségével a proxyhoz irányítjuk.

Amennyiben az ábrán látható hálózati felépítéssel rendelkezünk, a következő szabály „fogja meg” és téríti el a belső hálózatról kifelé irányuló webes forgalmunkat:

```
# ipchains -A input -i eth0
# -p tcp -d 0/0 80 -j REDIRECT 50080
```

A fenti szabály feltételezi, hogy a tűzfalgép 50 080 kapu-

ján egy olyan proxy hallgatózik, amely a forgalom közvetítéséről gondoskodik. Figyelem, a `REDIRECT`-szabályok működéséhez az IP-csomagok továbbítását a */proc/sys/net/ipv4/ip_forward* fájlban be kell kapcsolnunk, ezért gondoskodnunk kell róla, hogy ennek ellenére valós továbbítás a következő szabállyal ne történjen:

```
# ipchains -A forward -j DENY -l
```

Az 1. listán (http://www.linuxvilag.hu/Zorp/1.html) egy teljes csomagszűrő-beállítás látható – az `ipchains-restore` programnak megfelelő formátumban.

A Zorp két állományból áll, amelyet, ha a programot a fenti módon telepítettük, a */usr/local/etc/zorp* könyvtárban találhatunk, csomagból való telepítés esetén pedig a */etc/zorp*-ban.

Az egyik állomány az *instances.conf*, amely azt írja le, hogy a tűzfalon milyen és hány Zorp-példány fusson. Érdemes minden zónának kijelölt példányt létrehozni, mely majd az adott zónából származó kéréseket szolgálja ki. Az *instances.conf* minden sora egy-egy példányt határoz meg: az első szó a példány nevét, a sor további része pedig a Zorp programnak átadandó parancssori kapcsolókat adja meg. A mintaállományt a 2. lista (http://www.linuxvilag.hu/Zorp/2.html)) tartalmazza.

A létrehozott példányokat a `zorpctl` parancs segítségével indíthatjuk el, illetve állíthatjuk le. A másik beállítási fájl az alkalmazandó szabályrendszert tartalmazza, neve az *instances.conf*-ban egy kapcsolóval megadható. A Zorp esetében a szabályrendszert Python segítségével írjuk le, így ez a fájl egy Python-modul lesz, alapértelmezett helye pedig a *$prefix/etc/zorp/policy.py*. Mivel a fájl Python-modul, egy dolgot szem előtt kell tartanunk: a blokkok kezdetét és hosszát beljebbkezdés (indent) jelzi. A beljebbkezdésre vonatkozó egyetlen megkötés az, hogy egy blokkon belül azonos méretű kell legyen. Szóközt és tabulátort egyaránt használhatunk, az utóbbi mérete viszont az értelmező szerint 8 szóköz, ha tehát tabulátort használunk, a szövegszerkesztőnkben is ezt az értéket állítsuk be. Ha nem ügyelünk erre a szabályra, szintaktikai hibát kapunk.

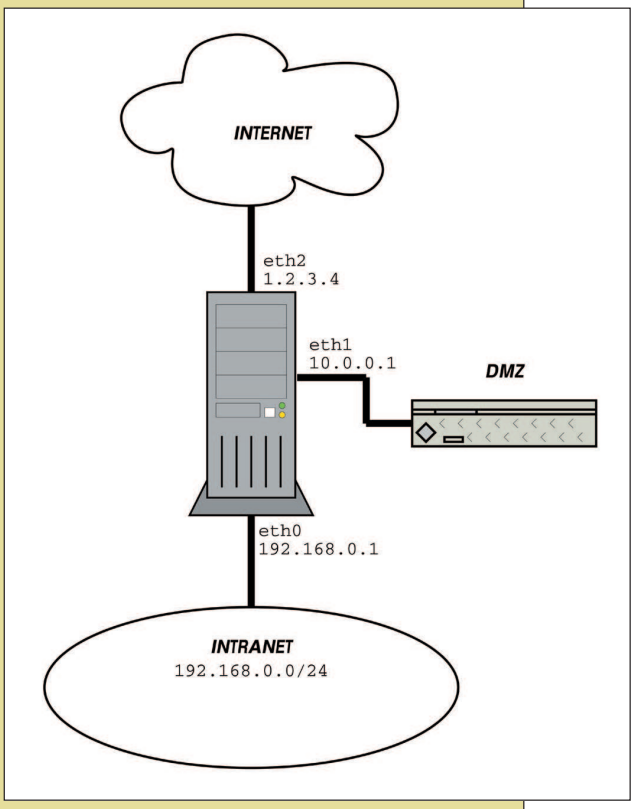
Nézzük, hogyan is épül fel egy beállítási fájl:

1. importálási rész, ami a Zorp által nyújtott szolgáltatásokat teszi elérhetővé,
2. zónameghatározási rész, mely a Zorp számára a környező hálózat felépítését írja le,
3. a proxy beállítása,
4. példánymeghatározások

Az egyes részeket a 3. listán (http://www.linuxvilag.hu/Zorp/3.html) láthatjuk.

Egy szolgáltatást a következő módon vehetünk fel: kiválasztjuk a használni kívánt Zorp-proxyt, majd eldöntjük, hogy a saját igényeink szerint kívánjuk-e testreszabni. A Zorpban minden proxynak, illetve a kapcsolódó beállításoknak egy-egy Python-osztály feleltethető meg.

A testreszabás egy kész osztályból való származtatást, illetve a megfelelő értékek (tulajdonságok – attributes) beállítását jelentik. Az egyszerűség kedvéért új szolgáltatásunk legyen egy átlátszó (transparent) webszolgáltatás a belső hálózatról az Internetre, melyet a Zorp alap HttpProxy-osztályával továbbítunk. Adjunk nevet ennek a szolgáltatásnak, legyen a továbbiakban „intra_HTTP”. A példányleíró részben minden Zorp-példányt egy Python-függvény jelképez, amelynek neve megegyezik a Zorp-példány nevével. Ez a függvény felelős az adott



példány betöltéséért. Új szolgáltatásunk létrehozásáért a példánynak megfelelő függvénybe vegyük fel a következő két sort:

```
Service("intra_HTTP", HttpProxy)
Listener(SockAddrInet
➥("192.168.1.1", 50080),
➥"intra_HTTP")
```

A fenti két sor létrehoz egy intra_HTTP nevű szolgáltatást, majd egy hallgatózó foglalathoz (socket) köti, ami az 50 080-as kapun várja a kéréseket.

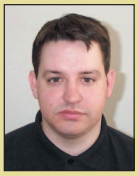
Ezekután a szolgáltatást engedélyeznünk kell: a beállítófájlban a zónameghatározásoknál a belső hálózatot leíró részhez az engedélyezett kifelé tartó szolgáltatásokhoz (outbound_services tömb) adjuk hozzá. Ehhez hasonlóan a szolgáltatás céljaként szereplő zónáknál ugyanezt az inbound_services halmazba is vegyük fel. Láthatjuk, hogy a Zorp ugyanazon a kapun hallgatózik, ahová az előző példában szereplő IP Chains-szabály a kimenő http-forgalmat eltérítette. Tehát a teljes kimenő http-forgalom a Zorp http-proxyn keresztül megy át, mely az adatfolyamot alkalmazásszinten elemzi.

Szolgáltatás létrehozásakor további lehetőségeink is adódnak, például eldönthetjük, hogy a célkiszolgáló címe

milyen módon dőljön el. Ha nem adunk meg semmit (mint fent), akkor alapértelmezésként a Transparent-Router nevű osztályt használjuk, ami kideríti, hogy a `REDIRECT`-szabály alkalmazásakor mi volt a kapcsolat eredeti célja. Olyan esetek is előfordulnak azonban, amikor nem `REDIRECT` téríti el az adatfolyamot, hanem a kéréseket egy előre meghatározott címre kell továbbítani (például DMZ-ben lévő webkiszolgálóra, amely nem rendelkezik útválasztóra küldhető címmel), ilyenkor válhat hasznunkra a DirectedRouter, melynek alkalmazása a következő:

```
Service("inter_HTTP_dmz",
➥HttpProxy,router=
➥DirectedRouter(SockAddrInet
➥("10.0.0.2", 80)))
Listener(SockAddrInet
➥("1.2.3.4", 80),
➥"inter_HTTP_dmz")
```

Természetesen számos további lehetőségünk van, próbálkozhatunk a proxyosztályok testreszabásával (ehhez tájékoztatást a Zorp Python-moduljainak forrásában találhatunk), végezhetünk címfordításokat az `SNAT` és `DNAT` segítségével, de a megfelelő osztályok módosításával a teljes hozzáférés-vezérlést is lecserélhetjük.
A fejlesztők a zorp-hu, illetve a zorp levelezési listákon bármilyen észrevételt szívesen vesznek, a listára a ➲ http://lists.balabit.hu/mailman/listinfo címen lehet feliratkozni.



Scheidler Balázs
(bazsi@balabit.hu) A kezdetek óta Linuxozik, jelenleg a BalaBit IT Kft. ügyvezetője és a Zorp vezető fejlesztője.

Cégcsokor (6. rész)

Sorozatunkban olyan cégeket gyűjtünk csokorba, amelyek régebb óta számos területen Linuxot alkalmaznak. Ezen írást a sorozat lezárásul szánom.

Forsz'98 Kft. – Szolnok

Készítettek egy rendszert, amit Hálózatos Pénztárgép Rendszernek kereszteltek el. Mit is tud? Alapkiépítésben egy PC-s kiszolgálóra egy (vagy több) pénztárgép csatlakozik, ami a következő lehetőségekkel bővíthető:

- A telephelyen több számítógép is lehet, ezekről (a kiszolgálóhoz hasonlóan) lehet például árut felvinni vagy számlázni.
- A boltban egy (vagy több) árlekérdezésre szolgáló (minimális kiépítésű) számítógép helyezhető el, amellyel a vevők az árucikkek árát ellenőrizhetik.
- A rendszer távolról, például telefonvonalon keresztül felügyelhető.

Maga a programrendszer ügyfél-kiszolgáló felépítésű. A központi helyen az adatbázis-kiszolgáló program áll, amellyel a készletnyilvántartó és számlázó, valamint a pénztárgépes program tart kapcsolatot. Alapkiépítésben a PC-s kiszolgáló futtatja ezeket az alkalmazásokat, a pénztárgépekkel online kapcsolat áll fenn. A bővítésekkel a programrendszer egy része a többi (ügyfél) számítógépre „szétszórható". Így előfordulhat, hogy vannak olyan gépek, amelyek csak árlekérdező programot futtatnak, de olyanok is, amelyek csak a készletnyilvántartó és a számlázó programot futtatják. Az adatokat a „munkaállomások" minden esetben a kiszolgálótól kapják.

A rendszer gépigénye a kiszolgálóhoz

- legalább 400 MHz-es Celeron osztályú processzor;
- legalább 64 MB memória;
- 4 GB-os merevlemez;
- CD-ROM-meghajtó;
- háromgombos egér, magyar billentyűzet;
- szünetmentes tápegység (kötelező);
- hálózati kártya (csak abban az esetben, ha további számítógépek is a rendszerbe vannak illesztve).

A választható ügyfélgépek alkatrészigénye ennél jóval szerényebb.

Nézzük a rendszer programigényeit!

- Linux operációs rendszer;
- PostgreSQL adatbázis-kiszolgáló;
- Gnome grafikus felület.

Mire is képes ez a rendszer?

- Egy soros vonalon legfőljebb 15 pénztárgép kezelhető (külön kábelezéssel ennek a többszöröse is megoldható).
- Több számítógép összekapcsolása hálózatban, amelyeken a munkavégzés (számlázás, leltározás stb.) párhuzamosan folyhat.
- Alapvetően vonalkódra épülő készletnyilvántartás.
- Különböző akciók kezelése két dátum között.
- Partnerek (törzsvásárlók) kezelése.
- Négyféle bizonylat kezelése.
- Vonalkód nyomtatása bármilyen kereskedelemben beszerezhető nyomtatón.
- Vonalkód alapján történő árlekérdezés.
- Összesítő és részletes kimutatások készítése.

A program magyar nyelven „beszél" és teljes leírást mellékelnek hozzá, amely szintén magyar nyelvű.
E teljes rendszer bemutatásával búcsúzom olvasóinktól.

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

Kitekintés a garázsomból

A legfontosabb tömegtájékoztatási eszközök még mindig teljesen félreértelmezik a történteket: Doc a sajtóban megjelent Linux-ellenes megnyilvánulásokra felel.

A böngészőmben szerepel a CNET egyik cikke, címe: „Van már Linuxod? Sok társaság nemmel válaszol erre a kérdésre". A cikk szerzője ***Sergio G. Non***. Az alcímben ez olvasható: „Az újságok elemzése: a Linux-pingvinek egyre hangosabban rikácsolnak, de a társaságok nem tervezik, hogy a közeljövőben túl sokat átvennének közülük." A bevezetés a tényeket összekeveri a feltételezésekkel, majd a cikkíró előáll az érveivel: „…mivel a gazdaság még mindig nagyon gyengélkedik, számos technológiai megfigyelő úgy vélekedik, hogy a Linux és „ingyenes" feliratú árcédulája számos üzleti vállalkozást vonzani fog, amelyek a költségeiket csökkenteni szeretnék, legalábbis elméletben. Ám a gyakorlat egészen mást mutat."

A „gyakorlat" csakugyan egészen más a ***Goldman Sachs*** felmérése alapján, amelyet száz technológiai cég igazgatójának segítségével készítettek. Felkérték őket, hogy nevezzék meg a terveikben szereplő legfontosabb és legkevésbé fontos kiadásaikat. A legsarkallatosabb kiadásnak a Windows új változatára történő átállás bizonyult, ezt követték a biztonsági programok és a Unix-kiszolgálók költségei. A legkevésbé fontos kiadások között a központi számítógépek, az ellátási láncot kezelő programok és a Linux-kiszolgálók beszerzését említették. Megismétlem, ez a kiadások fontossági sorrendje volt, nem a gyakorlat. Szó sem esett arról, hogy a Linux remek lehetőség a költségek megtakarítására. Végül a cikk közepe felé megtalálhatjuk a tulajdonképpeni gyakorlatot, amikor a szerző arról számolt be, hogy az Amazon.com éppen most „takarított meg milliókat azáltal, hogy számos üzleti területen Unixról átállt Linuxra". Utána egy informatikai tanácsadó szavait idézi, aki ezt mondja: „Számos ügyfelünk valós költségmegtakarítási lehetőségnek tekinti a Linuxot", majd pedig: „Ez csakugyan nyereséges befektetés." Ugyanez a tanácsadó később hozzáteszi: „Egyetértek azzal, hogy sok cég jelentősen visszavesz a jövőre vonatkozó nagy terveiből… Az ügyfeleinktől kapott visszajelzések azonban arra utalnak, hogy a Linux nagyon érdekli őket."

A helyzet az, hogy a cikkben felsorolt bizonyítékok egésze valójában a Linuxnak kedvez. De a CNET írójának nem ez volt a szándéka. Jelenleg az üzleti sajtóban Linux-ellenes hadjárat zajlik, és ez a fickó csak a munkáját végzi. Az igazság az, hogy most mindent ostoroznak, ami a dot-com buborékkal kapcsolatos: a kockázati tőke befektetőitől és a spekulánsoktól kezdve egészen a számtalan őrült ötletig, amelyek a spekulatív jellegű befektetéseket annyira vonzották.

Úgy gondolom azonban, hogy ez a hadjárat több okból is elsősorban a Linuxot érintette. Az egyik ok az, hogy nem sok tisztán linuxos társaság maradt fenn. A The Business Tux népessége annyira megfogyatkozott, hogy veszélyeztetett fajnak látszik, míg számos olyan nagy cég, mint az IBM, fennen hirdeti a Linux iránti szeretetét.

A múlt héten átválogattam az elmúlt évek során összegyűjtött névjegykártyáimat, és néhány kivételével az összes a szemétkosárban kötött ki. A kidobott névjegykártyákat elsősorban olyan társaságoktól kaptam, amelyek időközben csődbe mentek, beolvadtak más társaságokba, vagy azon fáradoznak, hogy új helyet találjanak maguknak a piacon. A másik gondot az a rövid, de látványos pillanat jelentette, amikor a spekulációs láz nagy csúcspontján a Linux a Wall Streeten a Tux mellett reflektorfénybe került. 1999 végén a Linux tűnt a legjobb befektetésnek, annak ellenére, hogy csak alig néhány befektető tudta, mit is tett a Linux azon kívül, hogy színre lépésével veszélybe sodorta a Microsoft magabiztos helyzetét.

A Linux nem technológia, nem is üzlet, és ez elvezet bennünket harmadik gondunkhoz, vagyis ahhoz, hogy a lényegét kell megértetnünk és elmagyaráznunk az igazi üzleti vállalkozások számára. A CNET említett cikke által idézett Goldman Sachs-tanulmány jól szemlélteti ezt a galibát. A Linux a maga tiszta állapotában nem értékesítésre való. Népszerűségét nem lehet ennek fényében vizsgálni.

Ami még ennél is rosszabb, hogy az előrejelzések egyébként is fabatkát sem érnek. A garázsomban kutakodva nemrég néhány gyöngyszemet találtam. Például a Sports Illustrated 1981. november 30-i számát, amelynek címlaptörténete az előszezon kérdéseivel foglalkozott, és az észak-karolinai bajnokcsapat, a Tar Heels négy csillagáról: ***James Worthy***-ről, ***Matt Doherty***-ről, ***Jimmy Black***-ről és ***Sam Perkins***-ről írt. Szóba sem jött, hogy hamarosan feltűnik nemcsak a csapat legnagyobb csillaga, hanem a kosárlabdása történetének egyik legjobb játékosa, ***Michael Jordan***.

Később megtaláltam a Popular Electronics 1976. júliusi számát. „Szenzáció! Most felépíthet egy kiváló minőségű intelligens terminált!"– írta. Belelapoztam. Nagyon sok hirdetés szólt CB-rádiókról és sztereokészülékekről, elektronikai folyamatokról és antennákról. A címlap közelében pedig egy kétoldalas ív szerepelt, amelynek felcíme így szólt: „Képzeljen el egy mikroszámítógépet". Alatta egy doboz látszott, amelynek a külsején LED-ek és kapcsolók voltak, benne pedig az összes szükséges csatlakozópanel és az áramellátás. A cikk így folytatódott:

„…Képzeljen el egy mikroszámítógépet, amely a legjobb mikroszámítógépek minden tudásával, kezdetlegességével és fejlettségével fel van szerelve. Képzeljen el egy mikroszámítógépet, amelyet az illesztőegységek, memóriák és processzorok tucatnyi lehetősége támogat. Amelyhez sok-sok külső eszközt lehet csatolni…"

A kép egy MITS Altair 8800-b számítógépet ábrázolt. Ez a készülék ugyan nem lett sikeres, de a jövőkép igencsak ismerősen hangzik, nemde?

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

MPlayer for Linux

Videolejátszás Linux alatt magyar módra: a mindentudó MPlayer.

Kevés olyan embert tudok magam elé képzelni, aki az alcímet olvasva nem mosolyodik el egy picit. Tény, hogy az elmúlt időszakban számos bírálat érte a Linuxot a multimédia területén mutatott gyenge szereplése miatt, melyben akad némi igazság. Figyelembe kell azonban vennünk, hogy az alapvetően a megbízhatóságot és a hordozhatóságot szem előtt tartó rendszer nem jegyzi elsődleges feladatai között a digitális mozgókép visszaadását. Ennek ellenére elmondhatom, hogy ez az irányzat jelentősen javulni látszik mind a grafikus felületek által nyújtott támogatások, mind a támogatást kihasználó programok terén. Ezen túlmenően különösen büszkék lehetünk arra, hogy a fent említett területen egy magyar fejlesztésű program jeleskedik. Fontos kiemelnem, hogy a lejátszó teljes értékű, tehát nem vethető össze az elődök sebességével és gyenge képminőségével, igaz, a grafikus felület is nagyobb támogatást nyújt. Az alábbiakban megpróbálom lépésenként, az alapanyagok fokozatos összegyúrásával bemutatni, hogy a fenti program segítségével a semmiből hogyan juthatunk el a legtöbb ismert formátum magas szinten történő lejátszásához.

A programról

Az MPlayer egy Linuxon működő, alapjaiban magyar fejlesztésű videolejátszó, amely azonban sok más Unixon és akár a nem x86 processzorokon is fut. Alapvetően parancssoros lejátszónak indult, tehát ez a része adja igazi erejét, ám az utóbbi változatokban már grafikus felület használatára is lehetőség nyílik. Szinte az összes videofájltípust képes lejátszani. Ismeri a legtöbb MPEG-, VOB-, AVI-, VIVO-, ASF/WMV-, QT/MOV-, FLI-fájlt, amelyek mellé jó néhány beépített XAnim és Win32 kodeket sorakoztat fel. Nem találtam olyan hibákkal teli, sérült indexszel rendelkező fájlokat, amelyeket ne tudott volna mindenféle fennakadások nélkül megjeleníteni. VideoCD, SVCD-ket és DVD-ket ugyancsak le tud játszani. A másik figyelemreméltó tulajdonsága a megjelenítési módok széles választéka. Támogatja az X11-, Xv-, DGA-, OpenGL-, SVGAlib-, fbdev-, aalib-, DirectFb-módokat, amelyeket akár SDL-en vagy GGI-n keresztül is el tud érni – mindeközben a réteg nyújtotta szolgáltatásokat is kihasználja. Ezen felül az MPlayer támogatja az MPEG-kártyákkal történő dekódolást és megjelenítést. Támogatja az OSD-t is, melynek segítségével a filmekre simított, árnyékolt feliratokat helyezhetünk. Ismer jó néhány feliratformátumot (köztük egy saját fejlesztésűt) is. Nem szabad arról az aprócska tényről sem elfeledkeznünk, hogy az MPlayer csomaggal kódolhatunk is! Erről azonban sorozatunk következő részében szólunk.

Az MPlayer 0.6-os változatának forráskódja letölthető a ➲ http://www.mplayerhq.hu címről, ahogyan a szükséges tartozékok nagy része is. Mindezek nagy része a 28. CD Magazin/MPlayer könyvtárában is megtalálható.

Telepítés

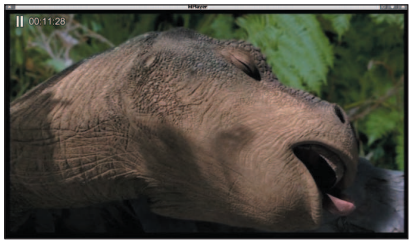
Mindenekelőtt nem árt, ha gyors masinánk van – lehetőleg valamilyen 2.4-es rendszermaggal és egy minél újabb fejlesztésű X-felülettel (lehetőségeink a változatszám növekedésével egyenes arányban bővülnek, célszerű tehát a legújabb kiadást használni). Jelen esetben grafikus kártyánk típusa meghatározó lehet.

A leírásban részletes eligazítást kapunk a használható videokártyák tekintetében is. Régi, elavult eszközökön nem érdemes próbálkozni, mert a kártyás gyorsítás hiánya a filmnézést élvezhetetlenné teheti – annak ellenére, hogy a program lehetőséget biztosít a lejátszásra. Nem kell azonban elrettennünk, egy középkategóriás gép is megteszi. A program az X-kiszolgálón keresztül minden olyan kártyát támogat, amely hardveres YUV-gyorsítást és átméretezést tartalmaz (ez a legtöbb mai kártyára igaz). A cikk írása közben történő próba során egy Celeron II 1000 (@1120) MHz-es processzorból, S3 Savage4 videokártyából, és egy Sound Blaster 16 Vibra hangkártyából összeállított gép állt a rendelkezésemre, és a processzor kihasználtsága a legnagyobb felbontású film lejátszása közben sem ment 50–60 százalék fölé.

Mivel a program csak forráskód formájában áll rendelkezésre, magunknak kell lefordítanunk. Az ehhez szükséges csomagok.

- binutils – ez a program felelős az MMX, 3DNow!, SSE stb. utasítások kezeléséért, ezért igen fontos.
- Xfree86 – mint már említettem, mindig a legfrissebb változat használata javasolt. Tapasztalatom szerint 4.0.2-es változat alatt nem érdemes próbálkozni, mivel a program az újabb videokártyákban elérhető hardveres YUV-gyorsítást ettől a kiadástól kezdve támogatja. A fordításhoz a fejlesztői csomag is szükséges! (A program enélkül is lefordul, csak az X által nyújtott szolgáltatások nem érhetőek el.)

Ezen túl a fordításhoz természetesen a szokásos kellékekre van szükségünk (`gcc`, `make`). Egyelőre azonban ennyi is elég a lejátszó beindításához. A későbbiekben még esik szó róla, hogy az egyes



szolgáltatások kihasználásához milyen egyedi kellékeket kell beszereznünk.
A fenti összetevők megléte lehetővé teszi, hogy a programot lefordítsuk. Megjegyzendő, hogy a videók kódolását és dekódolását végső úgynevezett kodekere is szükségünk van. Ezek mindegyike külön modult alkot, ezáltal maga a lejátszóprogram nem ismeri a formátumokat, hanem a videók tartalmát e kodekek segítségével nyeri ki.
Az alapvető, igen elterjedt és gyakori formátumok lejátszásához szükséges dekódoló eljárások a forráskódban megtalálhatók. A fordítás során ezek is lefordulnak és a helyükre kerülnek, különösebb odafigyelést nem igényelnek. Fordítsuk le a programot! Elárulom, ezt a lépést végre fogjuk még egy párszor hajtani – a program felépítése ugyanis olyan, hogy a rendszer tulajdonságai a fordítás során kerülnek a lejátszóba, tehát új alkatrész beszerzése esetén (például DVD-lejátszó), vagy ha új kodek kerül a birtokunkba, a programot újra kell fordítanunk.
Ha a program lefordult, a ***codecs.conf*** fájlt a ***$HOME/.mplayer*** könyvtárba át kell másolnunk (ez tartalmazza a fordítás során feltérképezett kodekek listáját és adataikat a program számára).
Innentől a lejátszónk működőképes. Akár használhatjuk is, a tudása azonban még szegényes. Ennek ellenére már teljes értékű program, ám ahhoz, hogy a cikk elején beharangozott profi lejátszóhoz is hozzájussunk, e ponton még nem állhatunk meg, akad némi teendőnk. Előbb azonban mérjük fel a jelenlegi összeállítás képességeit!

Kapcsolók

A lejátszót parancssorból és grafikus felületről (lásd később) egyaránt vezérelhetjük. A grafikus felület még kezdetleges, tehát a különleges lehetőségeket inkább csak parancssorból, kapcsolók segítségével állíthatjuk be. A leírásban az összes kapcsoló használatára vonatkozóan találunk példát, de a teljesség igénye nélkül be szeretnék mutatni közülük néhányat. Az első és egyik legfontosabb lehetőség a megjelenítési mód (milyen leképezési eszközt használjon) és a hanglejátszó eszköz megadása. Ezt a `-vo <megjelen ti eszk z>` és a `-ao <hangkezeli eszk z>` kapcsolókkal tehetjük meg. Arról, hogy milyen elérhető eszközökkel rendelkezhetünk, a `-vo help` és a `-ao help` kapcsolók segítségével tájékozódhatunk. Az élvezetes filmnézéshez hozzátartozik, hogy a kép a képernyő egészét betöltse, ami a `-fs` kapcsolóval érhető el. Attól függően, hogy milyen változatú és fajtájú ablakkezelő rendszert használunk, másmás teljes képernyős módok szükségesek a program számára, ezt a `-fsmode <ØrtØk>` kapcsoló állítja be. A filmek tömörítése veszteséges, ami torzulást eredményezhet, de ezen a képpontok elsimításával javíthatunk. Az eljárás neve utófeldolgozás (postprocessing). Lényege, hogy a már dekódolt kép tulajdonságait különböző szűrőkkel feljavítjuk.
E szűrők fajtáit, tulajdonságait adhatjuk meg a `-pp <hexadecimÆlis szÆm>` kapcsolóval. A hexadecimális számsor megadásáról a leírásban részletesen is olvashatunk. A számsor egyes tagjai (sőt a számok egyes helyi értékei) szabják meg az elsimító eljárás bizonyos értékeit (színekre, képpontokra vonatkozó beállítások). Ez a lehetőség kodekfüggő és nem mindegyik fájltípus esetében működik, de a legelterjedtebbeket (MPEG1-2,DivX,DivX, OpenDivX) támogatja. A `-vfm <ØrtØk>`-mel állíthatjuk be, hogy lejátszónk milyen kodekcsaládot használjon. Célszerű a beépített `libavcodec` családot használni, mert a leírások szerint ez a leggyorsabb, és a program is ezt támogatja leginkább (`-vfm 5`).
További gyöngyszemek:

- `-aspect <ØrtØk>` – képarány beállítása (4:3;16:9) – régi MPEG-eknél hasznos lehet.
- `-vc <ØrtØk>` – a kényszerített videokodek neve (`-vc help` – kodekek listája)
- `-ac <ØrtØk>` – a kényszerített hangkodek neve (`-ac help` – kodekek listája)
- `-framedrop` – lassú gépeken engedélyezi a képkockák dobását, így a film nem lassul le, hanem részletek maradnak ki belőle (nem az összes képkockát rajzolja ki).
- `-zoom` – programból megvalósított átméretezés. Nem a videokártya végzi, hanem a program (rendkívül processzorigényes művelet).

A részletesen leírásban az összes kapcsoló szerepel.
Ennyi kapcsolót folyamatosan begépelni természetesen elég fárasztó lenne, ezért beállításainkat a ***$HOME/.mplayer/config*** nevű fájlban készíthetjük el – csakúgy, mintha a kapcsolókat használnánk.
Lássunk egy példát a fájl tartalmából:

```
vo=sdl:xv     # xv-meghajt
              # hasznÆlata
              # SDL-en kereszt l
ao=oss:/dev/dsp1     #oss
```

```
használata, azon belül is
a 2. hangeszköz
fs=1          #teljes képernyős
megjelenítés
fsmode=1      #KDE-nél jól
működik a teljes képernyős mód
pp=0x2007f #postprocessing
örtékek
```

Így nem kell folyton megadnunk őket, elegendő, ha a lejátszani kívánt fájl nevét írjuk be. Amennyiben mégis úgy adódik, hogy ideiglenesen más lehetőséget szeretnénk használni, kapcsolóként megtehetjük, mert felülbírálja a beállítási fájlban megadott értéket.

Billentyűk futásidőben

f* vagy *g* hátra-**, illetve előretekerés 10 másodperccel; ***up* vagy *down* hátra-**, illetve előretekerés 1 perccel; ***pgup* vagy *pgdown* hátra-**, illetve előretekerés 10 perccel; ***p* vagy *Space pillanat–állj; ***q* vagy *Esc*** kilépés; **+ *vagy* -** audiokésleltetés megváltoztatása 0,1 ms-mal; **/ *vagy* *** hangerő csökkentése/növelése; ***o*** OSD-választás: nincs/folyamatsáv/folyamatsáv+időbélyeg; ***m*** master/pcm hangerőállítás között váltás; ***z*** vagy ***x*** felirat késleltetésének változtatása 0,1 másodperccel.

SDL

A hatékonyabb és kényelmesebb lejátszás érdekében ajánlatos telepítenünk az SDL-t (Simple DirectMedia Layer). Ez olyan réteg a Linux X-felületében, amely annak kép- és hangkezelő részére közvetlenül ráépülve egységes felületet biztosítva az őt használó programok számára értéknövelt szolgáltatásokat nyújt. Az MPlayer képes arra, hogy az SDL-rétegen keresztül érje el az X-felület által kínált alkatrész-támogatást, valamint a hangkártyát. Ez nemcsak további lehetőségeket kínál a számunkra, de néhány különleges esetben csak az SDL használatával érhetjük el, hogy lejátszóprogramunk helyesen működjön.
Az SDL használatát a `-vo sdl:<megjelenítési mód>` kapcsolóval érhetjük el. Ekkor lejátszás közben is lehetőségünk nyílik váltani az ablakos és a teljes képernyős nézetek között, továbbá teljes képernyős módban a C billentyű megnyomásával a képernyő felbontását is megváltoztathatjuk, ami sokszor jelentős processzoridő-megtakarítást jelenthet. Alapértelmezetten a lejátszó a teljes nézetre váltáskor az X-ben beállított lehetőségek közül a videó méretéhez legközelebb eső felbontásba állítja a képernyőt. Ennek kiküszöbölésére nincs lehetőségünk, annyit tehetünk, hogy a C betűvel a kívánt képernyőmódig evezünk, vagy `-fs` kapcsolóval indítjuk a lejátszót, így azonnal teljes képernyős módba áll, és nem változtat a felbontáson (ezután az F billentyűvel állhatunk vissza ablakos módba).

Az SDL a Linux-környezet része, telepítésével kapcsolatban nem tudok egységes útmutatást adni, mivel terjesztésfüggő. Ha az SDL eddig nem volt a rendszerünkre telepítve, a lejátszót most újra kell fordítanunk (a forráscsomag beállítóprogramjának kimenetéből megtudhatjuk, hogy van-e már SDL a gépen; ezenkívül az összes, a lejátszó szempontjából lényeges adatot is kilistázza).

Kodekek – a világ nem elég?

Ha már SDL-lel is rendelkezünk, elmondhatjuk, hogy egész kis „fegyvertárat" tartunk a kezünkben. Bátran elsütögethetjük és gyönyörködhetünk az eredményben. Mi azonban a gyönyörködés helyett inkább a többi „vetélytárs" fölé szeretnénk kerekedni… például azzal, hogy a mi lejátszónk – kis túlzással – az összes létező fájltípust képes megjeleníteni. A lejátszó számos kodekcsalád használatát támogatja, nekünk csak annyi dolgunk marad, hogy telepítsük őket.
Win32-es kodekek: a Windowsban használatanak számos fájltípus lejátszásához nyújt hathatós segítséget. Telepítésükhöz csupán annyit kell tennünk, hogy az MPlayer honlapjáról letöltjük a kodekgyűjteményt, majd a ***/usr/lib/win32*** könyvtárba kicsomagoljuk (szerzői jogvédelmi okok miatt ez a CD-mellékletre nem került fel).
Az egyes kodekcsaládok között természetesen jelentős átfedések vannak. Ez a könyvtár is tartalmazza például a DivX-ek dekódolásához szükséges modulokat. A lejátszáskor kiválaszthatjuk, hogy pontosan mely alkalmas kodeket szeretnénk használni, ám ha lehetőség nyílik rá, mindig a ***libvacodec***-készlet tagjait célszerű alkalmazni.
DivX4: igazából csak akkor van szükségünk rá, ha a `mencoder` segítségével kódolni is szeretnénk, mivel a ***libvacodec***-készlet hatékonyabban képes a lejátszásra. Ez a kodek a ➲ http://www.divx.com címről tölthető le, telepítése a csomagban leírt módon történik (lásd még 28. CD Magazin/MPlayer/Codecs).
XAnim kodekek: a régi videók (például az Indeóval vagy a Cinepakkal tömörített filmek) lejátszására alkalmasak.
Ha a kívánt kodekeket telepítettük, a programot újra kell fordítanunk.
A kodekek jelenlétéről a fordítás során kiadott `configure` parancs kimenetéből győződhetünk meg.

OSD – mi is ez?

Most már létezik egy olyan lejátszónk, amely gyakorlatilag az összes manapság használatos fájltípus lejátszását lehetővé teszi – itt az ideje tehát, hogy programunkat „extrákkal" ruházzuk fel.
Az OSD- (On Sreen Display) támogatás lehetővé teszi, hogy a filmre különböző adatokat írjunk ki, mintha szokásos szöveges kimeneti eszköz lenne. Ily módon időbélyeg (timer) és egyéb állapotok – mint például a hangerő mértéke vagy a filmbeli folyamatsáv – kijelzésére is lehetőség nyílik. Képes továbbá egy filmhez tartozó ismert formátumú szöveges feliratfájlból annak tartalmát a képernyőre írni (például angol nyelvű filmhez a magyar feliratot).
Ezek a lehetőségek természetesen benne vannak a programban, ám ahhoz, hogy megjelenjenek, betűtípusokra van szükségünk, melléjük pedig egy ***fonts.desc*** nevű fájlra, amely a unikód betűtípus-pozíciókat hozzárendeli a felirat tényleges kódlapjához. Léteznek azonban előre elkészített betűtípusok, amelyek szintén letölthetők az MPlayer honlapjáról (ezek is felkerültek a 28. CD Magazin/MPlayer/OSD Fonts könyvtárába). Számunkra jelenleg az ISO-8559-1 és 2 kódlapok támogatása a fontos, ám szerencsénkre az elkészített csomagok között találunk ennek megfelelőt. Telepítése szintén másolgatós jellegű: a letöltött csomag tartalmából az egyes könyvtárak közül ki kell választanunk a megfelelő méretű és fajtájú betűtípust tartalmazót, majd a tartalmát a ***$HOME/.mplayer/font*** könyvtárba kell másolnunk. Ez esetben a programot nem kell újrafordítanunk. Ezt követően lejátszás közben az O billentyű nyomogatásával válthatunk a különböző OSD-módok közül (idő, idő+folyamatsáv, csak folyamatsáv).

DVD – a korong

A fájlformátumokkal még csak-csak rendben volnánk, de a videofájlok nem-

csak merevlemezen vagy adat CD-n találhatók meg, hanem sok esetben más médián. Ilyen például a jó öreg VCD (VideoCD) vagy az újabb keletű SVCD (Super VideoCD). Mivel hordozójuk a szokásos CD, különösebb felkészítésre nincs szükség – az MPlayer le tudja őket játszani, csupán annyi a dolgunk, hogy közöljük a programmal: most egy ilyen lemezt szeretnénk vele megnézni.
Miután a CD-t betettük a meghajtóba, a `-VCD <sÆv sorszÆma> <meghajt eszk z>` kapcsolóval (például `mplayer -vcd 2 /dev/cdrom`) tudjuk lejátszani.
Fontos, hogy a meghajtót nem szabad befűzni (`mount`) a fájlrendszerhez!
VideoCD-n az első sáv általában egy adatsáv, maga a film a második sávtól kezdődik. Az eszköz alapértelmezés szerint */dev/cdrom*, tehát nem szükséges megadnunk. Ez a lehetőség akkor használatos, ha például a másodlagos CD-ROM-mal szeretnénk lejátszani.
Érdekes tény azonban, hogy bekapcsolt SCSI-emuláció mellett a */dev/scd0*, */dev/scd1* eszközöket nem hajlandó kezelni, a */dev/cdrom* mutatót át kellett irányítanom a */dev/scd0*-ra, és csak azután tudtam a videókat lejátszani.
SVCD formátumú lemez lejátszása ugyanígy történik.
A VideoCD-k tökéletes lejátszása nem boldogít bennünket, ha DVD-meghajtónk van, és mi egy ilyen korongot szeretnénk lejátszani vele. Ne ijedjünk meg, erre is lehetőségünk nyílik. Ehhez le kell töltenünk a ***libdvdcss-***, valamint a ***libdvdread***-forrásokat, és ugyanebben a sorrendben le kell őket fordítanunk (ezek a Linux-rendszer részei, az egyik a dvd formátum olvasásához, a másik a kódolt DVD-k dekódolásához szükséges). Most az MPlayert újra kell fordítanunk, majd a VideoCD-knél hasonló módon lejátszhatjuk a filmet.

- `mplayer -dvd 1` és az alábbi kapcsolók (ismét a teljesség igénye nélkül)

- `-slang <felirat azonos t ja>`; ezt írja a képernyőre,
- `-alang <orszÆg k d1, orszÆgk d2>`; ha van ilyen nyelvű hangsáv (mindkettőt, vagyis a nyelvek és a feliratok listáját a `-v` kapcsolóval kérhetjük le),
- `-chapter <jelenet szÆma>`; ettől a jelenettől indul,
- `-dvdangle <kamerasz g azonos t >`; a kameraszög kiválasztása.

Az összes kapcsoló megtalálható a részletesen leírásban.
A DVD-ken használatos 4 vagy 6 (például Dolby Digital 5.1) csatornás AC3-kimenet lejátszására is lehetőségünk nyílik, igaz, egyelőre csak SB Live! hangkártyákon, ez a „lista” azonban reménhetőleg bővülni fog. Meg kell említenem, hogy a lejátszás sajnos nem minden hangkártyán működik jól. Az oka nem feltétlenül a lejátszóban keresendő, ugyanis a kép és hang összehangolásához szabványos OSS-utasításokat használ, és az újabb, olcsóbb hangkártyák linuxos vezérlői sajnálatos módon nem szabványosak. Ilyenek például az alaplapra integrált hangkártyák, ugyanakkor ilyen a Christal 4281 PCI is – ezzel szemben egy ezeréves Sound Blaster 16-tal tökéletes eredmény érhető el.

Grafikus felhasználói felület

Ekkorra már egy igazán mindentudó lejátszóval rendelkezünk. Legyünk rá büszkék! Ami még hab lehet a tortán, az a változtatható kinézetű grafikus felhasználói felület. Ha mi is szeretnénk ilyet, fordításkor `configure` parancsot `--enable-gui` kapcsolóval kell kiadnunk. Mivel a felület GTK-t használ, ennek szerepelnie kell a rendszerünkön, és mivel fordításról beszélünk, a fejlesztői csomagokkal ugyanez a helyzet.
Újrafordítás után a `-gui` kapcsolóval (vagy az `gmplayer` paranccsal) kérhetjük a grafikus felület használatát. Előtte azonban az MPlayer honlapjáról le kell töltenünk néhány bőrt (skin), a lejátszó ugyanis alapvetően parancssoros, még az alapkinézet sem része a forrásnak.
A letöltött bőrt bontsuk ki a ***$HOME/mplayer/Skin/<skin neve>*** könyvtárba.
A program az alapkinézetet kapcsolók nélkül a ***$HOME/.mplayer/Skin/default*** könyvtárban keresi. Ha az alapértelmezettől eltérő kinézetet szeretnénk használni, a `-skin <skin neve>` kapcsolóval tehetjük meg.
Ha a lejátszó grafikus felülete elindult, programunkat a szokott módon egerezve vezérelhetjük. Mivel a grafikus felület még gyerekcipőben jár, a beállításokat kevés kivétellel csak parancssorból adhatjuk meg. Fontos még, hogy a grafikus felülettel történő filmlejátszás SDL-lel nem jól működik együtt, a kettőt egyszerre ne használjuk! Számos bőr található a 28. CD Magazin/MPlayer/Skins könyvtárában.

Dőljünk hátra…

… és élvezzük munkánk gyümölcsét. Azt hiszem, nem szükséges ecsetelnem, mekkora eredmény, hogy egyetlen programmal ennyiféle fájl- és médiatípust sikerült összefognunk, mindezt oly módon, hogy egy igen hatékony, bármely más programmal versenyképes, élvezhető minőséget nyújtó eszközt kaptunk eredményül. A változatszámból is látszik, hogy a program még csak most teszi meg az első lépéseit, ám soksok váratlan szolgáltatásával máris elkápráztat bennünket. Viszont még korántsem értünk a végére! A program fejlesztése ezekben a pillanatokban is folyik, hamarosan újabb kiadás megjelenése várható, amely többek között már grafikus telepítővel is rendelkezik.
Írásunkban sajnos most csak ennek az általános leírásnak jutott hely. A program annyira összetett, hogy felépítésének bemutatása, mélyebb elemzése csak nagyobb terjedelemben lehetséges. Sorozatunk következő részében a beépített libva kodekcsaládról, valamint a fájltípusok kezeléséről és felépítéséről szólunk.

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

Nemlineáris videoszerkesztő programok

Robin hat nemlineáris videoszerkesztő (NLE) programcsomagot vesz górcső alá: a Broadcast 2000-et, a Crow-t, a Kinót, az LVE-t, a MainActort és a Trinityt.

A MainActor kivételével azok a szerkesztőprogramok, amelyekről most szó lesz, mind nyílt forrásúak. A két legérdekesebb: a Kino, ez a nagy elismertségnek örvendő DV-szerkesztő (DV: digital video), melynek épp most jelent meg az új változata, valamint az LVE, ez a kevésbé ismert német származású MPEG-szerkesztő.

Broadcast 2000

A Broadcast 2000 még mindig szerepel a Linux Media Arts (LMA) videorendszer-integrátor cég kínálatában, bár abból, hogy a cég weblapjának főoldaláról eltávolították, arra lehet következtetni, hogy nem fejlesztik tovább. Az LMA elnöke, ***Mike Collins*** cáfolja a hírt: „Folytatni fogjuk a Broadcast 2000 fejlesztését mind nyílt forráskódú formában, mind pedig kereskedelmi programcsomagként, amelyhez megfelelő termék-támogatást is nyújtunk.” Collins elmondja még, hogy a Broadcast 2000-et a ➲ http://SourceForge.net-re is próbálják feljuttatni, de addig is közvetlenül tőle bárki letöltheti, rpm- vagy tar-csomag formájában érhető el.

A Broadcast 2000-rel a Linuxvilág 3. számának 30. oldalán foglalkoztunk részletesen. A mi telepítésünk folyamata mivel nem Red Hatet, hanem Debiant használunk, egy kicsit eltér. Először alakítsuk át a Red Hat-féle rpm-csomagot a Debian deb formátumára (*1. lista*).

Az Aliennel az rpm-csomagok deb formátumba való átalakítása remekül megoldható, a program azonban sajnos nem találja meg azokat a könyvtárakat, amelyek nem szabványos elhelyezkedésűek. Jegyezd meg az Alien üzenetét, miszerint nem találja a ***libbcbase.so***-t, és majd a deb `dpkg`-vel való telepítése után térj vissza a kérdésre. A Broadcast 2000 telepítésének befejeztével keresd meg az elkallódott DLL-t, és elérési útvonalát add az ***ld.so.conf***-hoz. Ezután az ***ldconfig*** futtatásával frissítsd a futásidejű program-összeszerkesztő (linker) beállításait. Enélkül a program nem fog futni, mivel megosztott könyvtárait nem lesz képes megtalálni.

A Broadcast 2000 formátumai közt szerepel a wav, pcm, QuickTime és jpeg képsorozat. A támogatott Quick Time (MOV) típusok a jpeg-kép, mozgó jpeg, png, png alfa, tömörítetlen RGB, tömörítetlen RGBA, YUV 4:2:0 planar és a YUV 4:2:2 packed. Elég soknak tűnik, de ezek a formátumok kevésbé férnek össze más operációs rendszerekkel és eszközökkel.

1. lista A Red Hat-féle rpm átalakítása a Debian deb formátumára

```
alien -k bcast-2000c-1.i386.rpm
  dpkg-shlibdeps: warning: format of
  ➥libbcbase.so
                  not recognized
  bcast_2000c-1_i386.deb generated

dpkg -i bcast_2000c-1_i386.deb

find /usr -name libbcbase.so -print
  /usr/local/bcast/libbcbase.so

vi /etc/ld.so.conf
/usr/X11R6/lib/Xaw3d
/usr/X11R6/lib
/usr/aw/maya4.0/lib
/usr/local/bcast
/usr/local/lib

ldconfig

/usr/local/bcast/bcast2000.sh
```

Crow

A Crow az ausztrál ***Eric Fry*** fejlesztése. Ugyan még csak alfa-változatú, mégis kíváncsiak voltunk rá. Csak a CVS-en (Concurrent Versions System – párhuzamos változatkezelő rendszer) keresztül tölthető le. Linuxos gépünk windowsos hálózaton lóg egy Windows-tűzfal mögött, kábelmodemmel csatlakozva. Tűzfalunkat úgy kellett beállítani, hogy engedélyezze a CVS-t. A Telnettel ellenőriztük, hogy a ➲ http://SourceForge.net 2401-es CVS-kapuja elérhető-e kézi beállítással. Ezután létrehoztunk egy TCP-burkolót a WinGateben a (***gap*** nevű) tűzfalgépen. A TCP-burkolót úgy állítottuk be, hogy a ***cvs.Crow.sourceforge.net***-re mutasson.

A linuxos gépről megkísérelt CVS-belépésünk kudarcot vallott: a ***.cvspass*** valamelyik beállításával lehet gond a távoli gépen – tippeltünk. Mindenesetre arra képesek voltunk, hogy CVS-ellenőrzést hajtsunk végre, s igazából csak erre volt szükségünk.

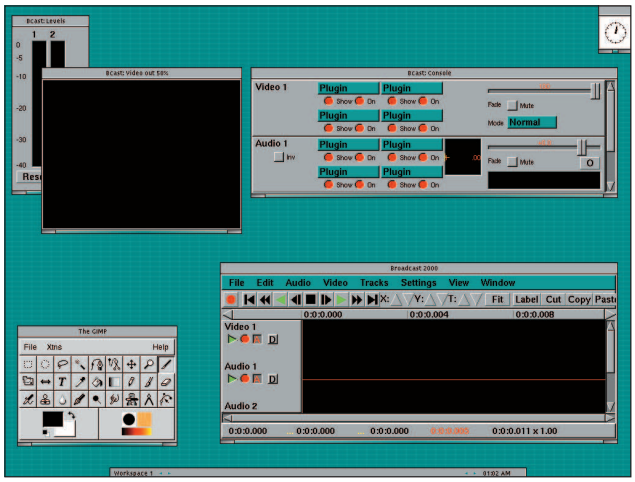
```
cvs -z3 -d:pserver:anonymous@gap:
➥/cvsroot/crow co crow
```

Ha tűzfalunkat nem fúrtuk volna meg, a ***gap*** helyett a CVS parancsban a ***cvs.Crow.sourceforge.net*** lett volna megadva.

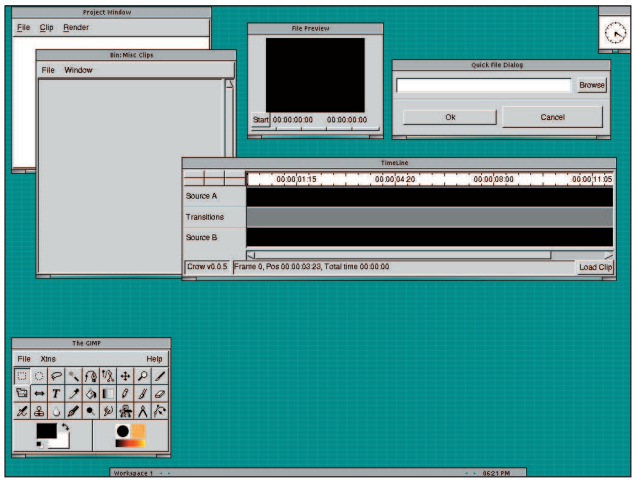
A Crow jelezte, hogy a libtool ***ltconfig***-változata nem megfelelő, ezt a `libtoolize` segítségével hoztuk rendbe:

```
libtoolize --force
aclocal
./configure
make
```

Újabb gond: a `make` parancs nem futott le, mert az egyik beépítő állomány nem találta a ***gtk/gtl.h***-t. A `Makefiles` nem találja az útvonalat, ami eredendően a GTK++ GUI-fejállományra mutat – a mi GTK++ állományaink más könyvtárba lettek telepítve. Ez némi beavatkozást igényelt. A `GNOME_INIT` parancsot hozzáadtuk a ***configure.in***-hez, amely feltölti a `GTK_CFLAGS` és `GTK_LIBS` változókat. Ezután ezeket a



1. kép
A Broadcast 2000 olyan a QuickTime-on alapuló videoszerkesztő, amely számos fejlett tulajdonsággal rendelkezik, például a Gimp által használt hatásokkal



2. kép
A Crow fejlesztője szerint az alkalmazás még nem kész a használatra, telepítése során mindenesetre sok mindent megtanultunk az alfaprojektek fordításáról és beállításáról

változókat – például `$(GTK_CFLAGS)` – helyettesítettük be a Crow által használt 14 különböző ***Makefile.am*** állományban. A beállítóállomány helyreállítása után a Crow-t sikerült összeszerkeszteni. Ezt megcsíptük!

```
aclocal
autoconf
./configure
make
ulimit -c unlimited
./app/crow
```

Nem próbálkoztunk a `make install`-lal, mert a Crow INSTALL-állománya óva intett ettől. Beállítottuk az `ulimit`-et, hogy összeomlás esetén a ***core***-állományt elő tudjuk állítani. És ahogy a Crow-t elindítottuk, egy szegmentálási hibával azonnal össze is omlott. Keressük meg a hibát a `gdb` hibakereső segítségével:

```
gdb app/.libs/lt-crow core
> bt
```

Mivel a ***./app/crow*** parancsállomány, ki kellett találnunk a futtatott állomány valódi nevét, hogy a `gdb`-be tölthessük. A nyomkövető (backtrace – bt) futtatásával vizsgálva az utasításverem tartalmát most nem érnénk célt, mert fordításkor nem helyeztünk el hibakereső adatokat. Először tehát a hibakeresés engedélyezésével a programot újra kell fordítanunk:

```
make clean
make -e ?CC=gcc -g?
```

Most ismét betölthetjük a `gdb`-t, és megvizsgálhatjuk a címverem tartalmát, valamint a forráskódot. Azonnal nyilvánvalóvá vált, hogy a lefagyást egy nullértékű mutató (a `dir`) okozta az ***app/plug_in.c*** állomány 275. sorában. Egyszerűen beszúrtunk egy sort, amely a változó 0 értéke esetén a kiléptet a függvényből, mielőtt még a változó használatára sor kerülhetne:

```
if(!dir) return 0;
```

A Crow még nagyon kezdetleges állapotú a használathoz. Fry elmondása szerint a ➲ http://SourceForge.net-en egy régebbi változat található és az újraírását tervezi. Szeretne olyanokkal beszélni, akik érdeklődnek egy teljesen új linuxos szerkesztőprogram írása iránt. „Egy olyan teljes tudással felvértezett szerkesztőprogramot szeretnék, amely képes olyan szerkesztőműveletekre, mint a 3:2 átalakítás (pull-down), az EDL (Edit Decision List) kimeneti adatbázis, a klipkezelés és a HD-visszajátszás” – mondja Fry.

Kino

A Kino egyszerű, csak vágásra alkalmas DV-szerkesztő. A programnak nemrégiben jelent meg a 0.50-es változata, ami komoly újításokat tartalmaz. A Kino telepítéséhez először egy sor illesztőprogramot kell feltelepíteni. Ez és a rengeteg elérhető DV-alkalmazás közül néhány feltelepítése a folyamatot meglehetősen bonyolulttá teszi.
A leírás a minél jobb IEEE 1394 FireWire DV-támogatás elérése érdekében a legújabb rendszermag használatát ajánlja. A `make xconfig` használatakor elsőként azon lepődtünk meg, hogy az IEEE 1394 beállításai a menüben szürkén jelentek meg, nem lehetett választani. 2.4.14-es rendszermagunk beállításakor a 1394-menüpont engedélyezéséhez először a ***Code maturity*** lehetőséget kell kiválasztani. Ezt az alábbi meghajtók követték: libavc1394-0.3.1.tar.gz, libdc1394-0.8.3.tar.gz, libraw1394_0.9.0.tar.gz, libdv-devel-0.9-1.i386.rpm and libdv-0.9-1.i386.rpm.
Tar-állományok esetén a szokásos eljárást követhetjük:

```
tar xvfz libdc1394-0.8.3.tar.gz
cd libdc1394-0.8.3
./configure
make
su
make install
```

Az rpmként érkező állományok telepítéséhez az Alient használtuk. A `libdv` tar-állományból történő telepítése után először a `pkg-config`-ot kell telepíteni ahhoz, hogy a beállítás sikeresen végrehajtható legyen.
A könyvtárak néhány egyszerű eszközt is tartalmaznak.

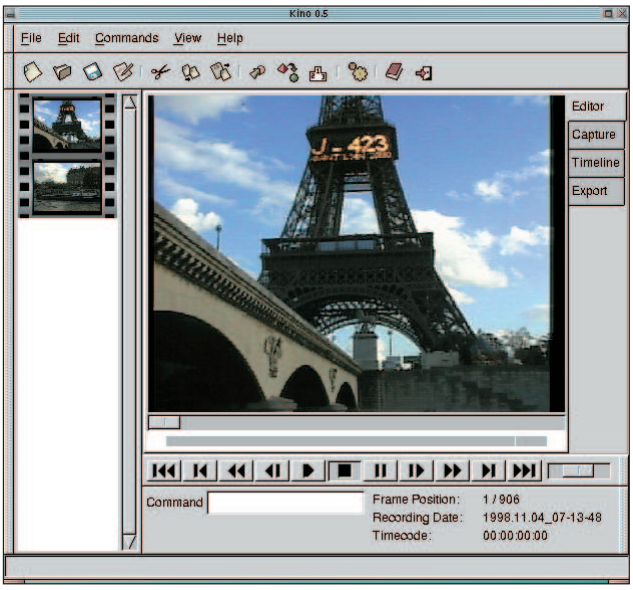
2. lista A modprobe

```
./test/dvcont help
  Couldn't get 1394 handle: No such device
  Is ieee1394, driver, and raw1394 loaded?

modprobe video1394

modprobe raw1394

lsmod
  Module      Size   Used by      Tainted: P
  raw1394     6672     0
  video1394   14756    0    (unused)
  ohci1394    16608    2    [video1394]
  ieee1394    24200    0    [raw1394
                    ➥video1394 ohci1394]
.
.
.

./test/dvcont stop
```

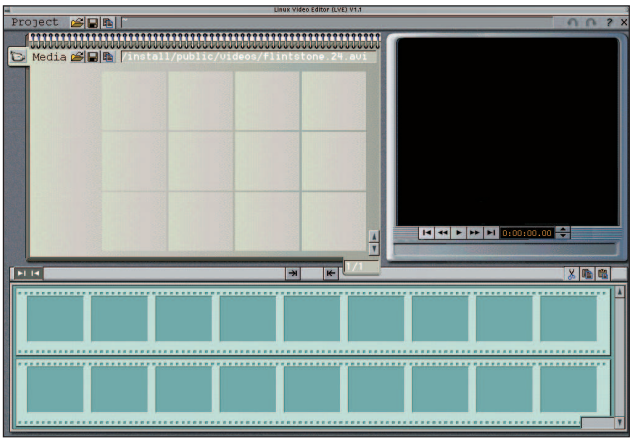
A libraw1394 például a testlibraw-t tartalmazza. Indítsunk ezzel, hogy kideríljön, megtalálja-e FireWire-kártyáinkat. Athlonnal szerelt gépünk két FireWire PCI-kártyával bír: egy PYRO gyártmányúval és egy névtelen (no-name) kártyával, amit a FireWire lapolvasóhoz kaptunk. Mindkettő OHCP-megfelelő (Open Hardware Certification Program – OHCP), és a próba helyesen ismerte fel. A `chmod` segítségével kellett lehetővé tennünk, hogy az eszközhöz közönséges felhasználóként is hozzá lehessen férni:

```
chmod 666 /dev/raw1394
./src/testlibraw
  successfully got handle
  current generation number: 3
  2 card(s) found
    nodes on bus:  1, card name: ohci1394
    nodes on bus:  1, card name: ohci1394
  using first card found: 1 nodes on bus,
    local ID is 0, IRM is 63
  ,
  ,
  ,
```

A libavc1394 memóriaellenőrző program a csatlakoztatott FireWire-eszközök ROM-adatait részletezi. Hiba nélkül ismerte fel Sony TRV8 DV kameránkat is.
A libavc1394-el együtt kapjuk a dvcont nevű, a képfelvevő távirányítását lehetővé tévő programot. Első használatakor érdemes a `help` kapcsolóval indítani, így mindjárt a parancsaival is megismerkedhetünk. Amennyiben a meghajtó programjait nem találja meg, azonnal kilép, még a súgót sem jeleníti meg. Szükség lehet a `modprobe video1394 és raw1394` futtatására. Kapcsoló megadása nélkül a `dvcont` semmit nem csinál. A videokamera-lejátszás üzemmódjában kiadott `dvcont stop` megállítja a lejátszást. Hasonlóan kapcsolhatjuk a képfelvevő többi üzemmódját is.
A ***libdv*** tartalmazza a ***playdv*** és ***encodedv*** programokat, de ezek szegmentálási hibával szálltak el. Feltételezem, ezek lennének



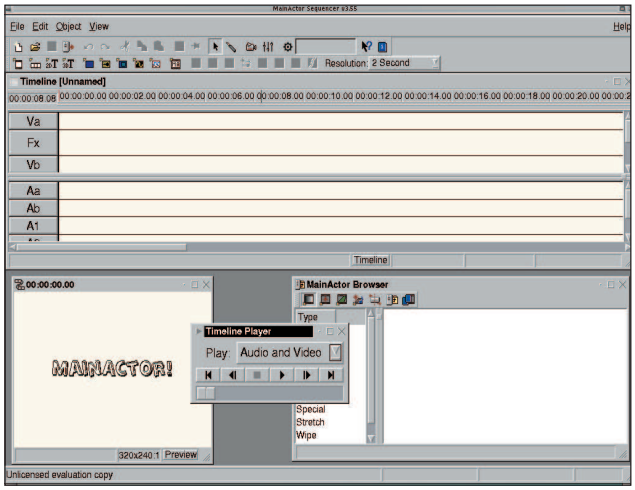
3. kép
A népszerű DV-szerkesztő, a Kino új változata nemrég látott napvilágot



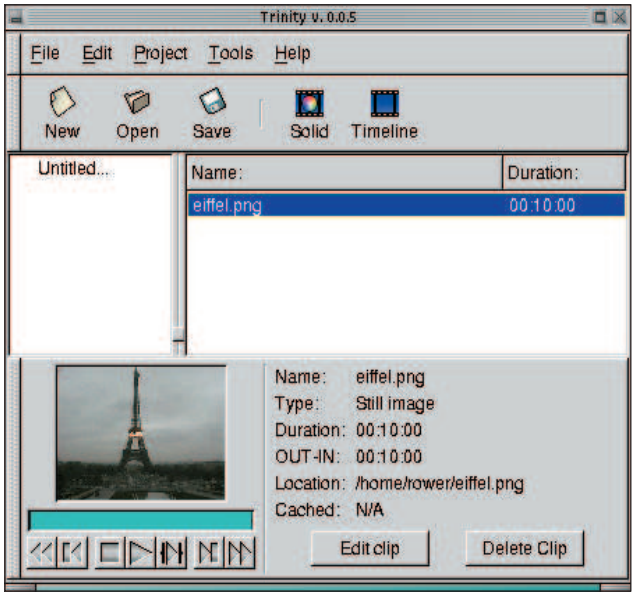
4. kép
Az LVE egy (S)VCD-k készítésére írt MPEG-szerkesztő Németországból

alkalmasak a ***dv***-állományok lejátszására, illetve kódolására. Néhány hasznos DV-alkalmazás: dvgrab-1.01.tar.gz, gscanbus-0.6.tgz and gstreamer-0.2.1.tar.gz.
A coriandert és a gscanbust nem sikerült lefordítani. A gstreamer – bár igen hosszú idő alatt – lefordult, de szegmentálási hibával leállt. Az eszköz, amire leginkább fentük a fogunkat ebből az összeállításból, a `dvgrab` volt, amellyel konzolon próbálhattuk ki a DV-digitalizálást. Mivel a Kino használatát nem gátolta, nem álltunk meg vizsgálni, hogy egyes DV-eszközök vajon miért nem működnek.
A `dvgrab` segédprogram a kamerában lejátszott videót másolja a számítógépre. Az eszközt nem ellenőrzi, azaz amennyiben a kamera nem lejátszás üzemmódban van, a dvgrab várakozik. Ha nem észlel FireWire DV-forgalmat, nem csinál semmit.

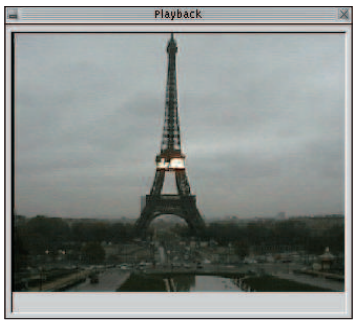
```
dvgrab --frames 30 test
ls -l test*
  -rw-r--r--     1 rower
  ➥rower 2471424 Nov 17 17:27 test001.avi
```



5. kép
A MainActor egy feltételekhez kötötten ingyenes szerkesztő Linux és Windows alá



6. kép A Trinity csak a fejlesztőre vár, aki új nevet és jövőt ad majd neki



7. kép Lejátszó

Próbaképpen egy egymásodperces videót olvastunk be. A `dvgrab` segítségével DV-folyamként mentettük, AVI protokollba csomagolva. A `dvgrab` és a Kino fejlesztését ***Arne Schirmacher*** irányítja. Először a Kino 0.46-os változatát telepítettük (kino-0.46.tar.gz), de próbálgatásaink alatt a 0.5 is napvilágot látott (kino-0.5.tar.gz). A fordítással eleinte nehézségeink adódtak, mert a `configure` a ***gnome-config*** állományt nem találta. Ez kicsit furcsának tűnt, hiszen rendelkeztünk telepített Gnome-mal (Debian Sid lévén a gépen). A dpgk-nak a libgnome-dev csomagban kellene megtalálnia (mint alább látható), de ott nem került elő.

```
dpkg -S gnome-config
  libgnome-dev: /usr/include/libgnome/
  ➥gnome-config.h
  libgnome-dev: /usr/bin/gnome-config
  libgnome-dev: /usr/share/man/man1/
  ➥gnome-config.1.gz
```

Dan Dennedy, a Kino egyik fejlesztője a Ximian Gnome-ot ajánlotta telepítésre, mivel ő is ezt használja. Hozzáírtunk hát még egy hivatkozást a ***sources.list***-ünkhöz és telepítettük:

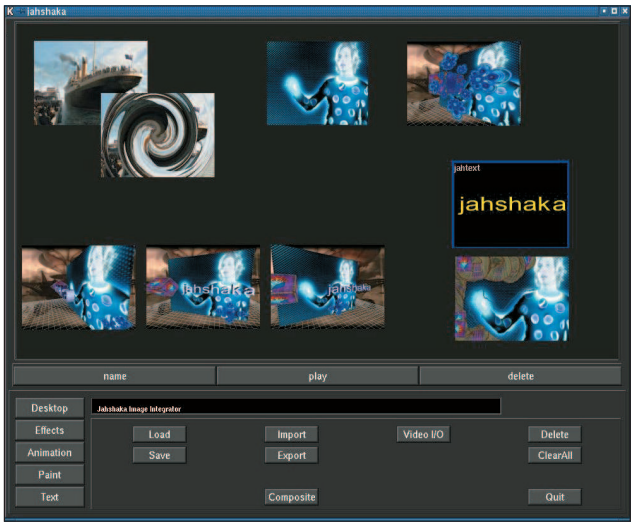
```
vi /etc/apt/sources.list
deb http://red-carpet.ximian.com/debian
➥stable main
.
.
.

apt-get update

apt-get install task-ximian-gnome
```

A Ximian telepítése súlyos ütközéseket okozott a már telepített Gnome-összetevőkkel. Számos csomagot, amelyet az `apt-get` sérültnek jelzett, újra kellett telepíteni. Az itt közölt példa csak szemlélteti a folyamatot, a végrehajtás valójában sokkal több fáradságba került. Meglepetésünkre a Gnome GTK dokumentumok tűntek a telepítést leginkább zavaró tényezőknek. Végül is sikerült a Ximiant telepíteni, s ezzel együtt a Ximian Gnome-fejlesztőkészlet csomagjait is. A Kinót gond nélkül sikerült lefordítani.

```
dpkg -r libgnome-dev
apt-get build-dep libgtk1.2-dev
dpkg -r libgtk1.2-doc
apt-get -f install
apt-get install libxml2
```

A Kino használatának alapjait a `vi` parancsai képezik. A sorokra, szavakra és betűkre vonatkozó parancsok tárgya itt a film, a vágások és a képek. A `d0` a bemenő pont, a `d$` pedig a kimenő pont jelölésére használatos.
„A Kino új változata az átdolgozott felhasználói felülettel jelentősen eltér az előzőtől, nincsenek többszörösen úszó ablakok, és a tervezőasztal-nézet jobban hasonlít az iMovie-éra. Észre fogjátok venni, hogy a felület egységesebbé vált”– mondja Dennedy. Arról is tájékoztat, hogy a Kino fejlesztőcsapata új taggal bővült, ***Charles Yates***-szel – neki volt a legnagyobb befolyása a 0.5-ös változatra. Dennedy hozzáteszi még, hogy a Kino EDL-je XML SMIL formátumú, hangtámogatása pedig az OSS-en (Open Sound System) alapul. Mivel a legtöbb hangkártya a többszörös megnyitást nem támogatja, az esound (Enlightened Sound Daemon) csak az SBLive! kártyákkal van támogatva. A Contour ShuttlePro USB-vezérlő (125 dollár) kényelmes eszközt biztosít a Kino vezérléséhez, kikerülve a `vi`-alapú felületet. A Kino az előnézet javítása érdekében támogatja az XVideót, de nekünk ezt a szolgáltatást ki kellett kapcsolnunk, mert asztalunkon az előnézet ablaka állandóan leragadt. Úgy tűnt, ennek oka leginkább az, hogy az ati.2-meghajtó rossz változatát telepítettük, és

nem az XVideóval van gond általánosságban.
A Kino képes a DV1 vagy DV2 AVI-formátumú állományokat kezelni, de nem kezeli a strukturálatlan DV-állományokat (*.dv*). A választott formátum a többi eszközzel való együttműködésre hatással van. „A Windows Media Player (WMP) és a linuxos Avifile lejátsza a DV2 AVI formátumú állományokat, amennyiben a microsoftos DLL telepítve van. Ennek neve ***qdve.dll***.
Az MPlayer szintén képes ezek lejátszására, de ehhez a beállítóállományt egy sorral ki kell egészíteni" – magyarázza Dennedy. Hozzáteszi még, hogy a WMP szintén kezeli a DV1 AVI-t, továbbá a MainActor elboldogul a DV-vel, de a DV-kamerákhoz nem rendelkezik csatlakozással.

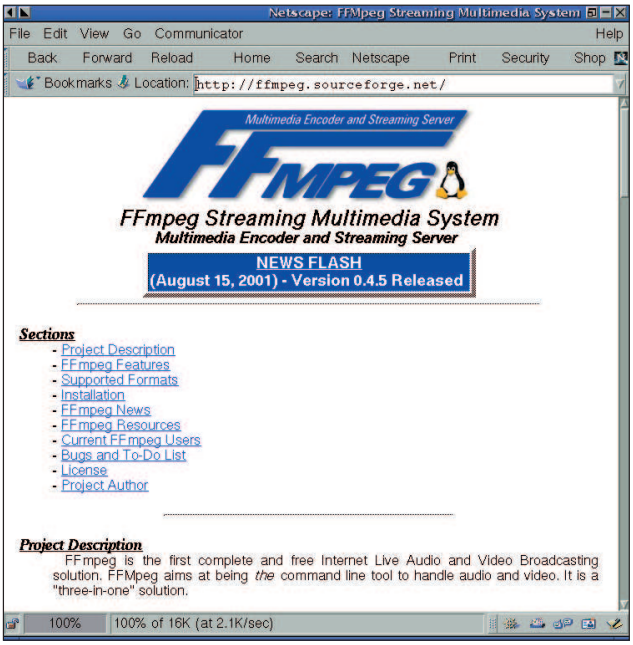


8. kép
A Jashaka különleges hatásokat előállító OpenGL-alapú alkalmazás Linux és Windows alá

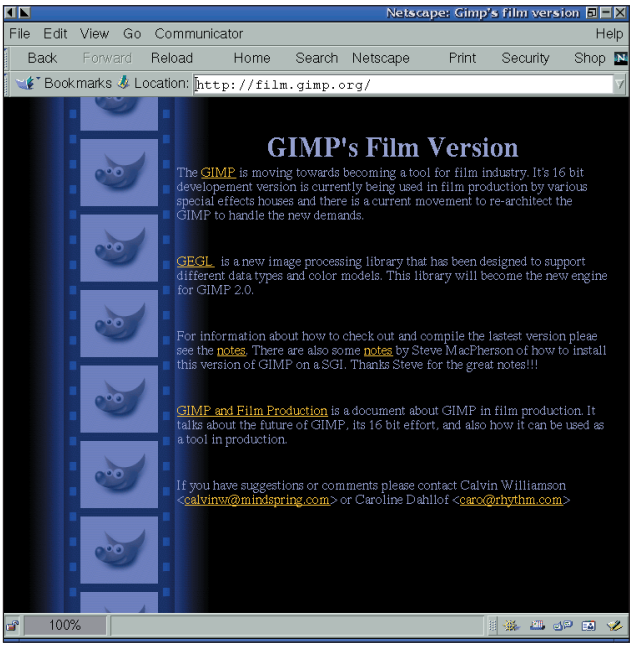
Az olyan eszközök, mint például a dv2jpeg, segíthetnek a formátumok közötti átjárásban. Dennedy hozzáteszi: „Az mjpegtools olvassa a DV AVI-t VCD készítéséhez, és támogatással bír az MPEG-1 VCD és MPEG-2 SVCD formátumokhoz. A Kino nemsokára exportálni tud majd MPEG-1 és MPEG-4 formátumban az FFmpegre támaszkodva, de az MPEG-2-támogatása az FFmpegből pillanatnyilag hiányzik." Az mjpegtools DV-hang leválasztására is képes. A transcode nevű eszköz a DV AVI-t OpenDiv X-formátumúvá konvertálja.
Dennedy lelkesen tudatja, hogy a Kino a felvevő felé a DV-exportot is támogatja. Mint mondja, ezzel a Kino az egyetlen DV-alkalmazás, amely a `libdv` konzolos alkalmazás mellett erre képes. A Kino a naplózó üzemmódot nem támogatja, Dennedy szerint a közeljövőben nem is fogja. Szerinte ez a szolgáltatás nem túl érdekes azok számára, akik a filmhez különböző hatásokat kevernek, és már fejlesztés alatt áll az új dv1394-meghajtó, amely TV-monitoron keresztül biztosít előnézetet. Hozzáteszi még, hogy a Kino MPEG-exportja is úton van, és hogy nem tervezik a Film Gimppel való egybeolvasztást.

LVE

„A programmal célunk a MPEG-állományok vágása és (S)VCD-formátumba való kódolása volt" – tájékoztat ***Gerhard Monzel***, a Linux Video Editor (LVE) fejlesztője. Monzel az SAP-nál dolgozik rendszergazdaként a németországi St Ingbertben.
Az LVE teljes leírása német nyelvű. „Az LVE néhány ingyenes csomagra épül: a libmpeg3 a keresést és az MPEG dekódolását végzi, a libsdl pedig a GUI alapja" – tájékoztat Monzel.
„A többi saját munka." Elmondja még, hogy sokféle MPEG-formátum támogatott (MPEG-1, MPEG-2), beleértve a VOB-ot és IFO-t is, de a DVB-t nem. A felhasználói felületre a Pinnacle Studio MP10 volt nagy hatással.
Az LVE csak vágásra képes, sem hatások (effects), sem feliratkozási lehetőség nincs. Megjegyzendő a forrásanyag akár PAL, akár NTSC, a kimenet mindig PAL marad. Az NTSC-forrású anyagokat a képsebesség javítása érdekében a hangmagasságot változtatva a soxszal kell feldolgozni.
Az LVE-t nem kell telepíteni, csak a tar-csomagot kibontani a saját könyvtárban:



➲ http://www.ffmpeg.sourceforge.net



➲ http://www.film.gimp.org

Kapcsolódó címek

Broadcast 2000 ➲ http://www.heroinewarrior.com/bcast2000.php3
Coriander ➲ http://www.sourceforge.net/projects/coriander
Crow ➲ http://www.crow.atu.com.au/main.php3
dumpmpeg ➲ http://www.sourceforge.net/projects/dumpmpeg
dv2jpg ➲ http://www.sourceforge.net/projects/dv2jpg
dvbackup ➲ http://www.sourceforge.net/projects/dvbackup
dvgrab ➲ http://www.schirmacher.de/arne/dvgrab/index_e.html
Film Gimp ➲ http://www.film.gimp.org
FFmpeg ➲ http://www.ffmpeg.sourceforge.net
GAnSO ➲ http://www.gpul.org/proyectos/ganso
Gnonlin ➲ http://www.sourceforge.net/projects/gnonlin
Gscanbus ➲ http://www.gscanbus.berlios.de
GStreamer ➲ http://www.sourceforge.net/projects/gstreamer
Jahshaka ➲ http://www.sourceforge.net/projects/jahshakafx
Kino ➲ http://www.sourceforge.net/projects/kino
LVE ➲ http://www.de.groups.yahoo.com/group/liunux_mpeg_world
libavc1394 ➲ http://www.sourceforge.net/projects/libavc1394
libdc1394 ➲ http://www.sourceforge.net/projects/libdc1394
libdv ➲ http://www.libdv.sourceforge.net
Libraw1394 ➲ http://www.sourceforge.net/projects/libraw1394
Linux 1394 ➲ http://www.sourceforge.net/projects/linux1394
Linux 1394 FAQ ➲ http://www.linux1394.sourceforge.net/faq.html
Linux DV ➲ http://www.schirmacher.de/arne/dvgrab/index_e.html
Linux Media Arts ➲ http://www.linuxmediaarts.com
Linux Video Studio ➲ http://www.ronald.bitfreak.net
MainActor ➲ http://www.mainconcept.com
Matterial ➲ http://www.matterial.sourceforge.net
mjpegtools ➲ http://www.sourceforge.net/projects/mjpeg
PYRO 1394 PCI card ➲ http://www.adstech.com
RAYZ ➲ http://www.silicongrail.com
Shake ➲ http://www.nothingreal.com
Trinity ➲ http://www.sourceforge.net/projects/trinitytv

```
cd /

tar xvfz /install/public/nle/lve/
➥lve_bin-31-10-01.tar.gz

ls /usr/local/lve/bin
  bbainfo     bbinfo bbvinfo       ffmpeg_lve
  gensmart    lmp    mplex  toolame
  bbdmux      bbmplex       encode genmpg
  gensvcd     lve    qdir

chmod 666 /usr/local/lve/lib/SystemFont.bmp

./lve
```

MainActor

Az általunk vizsgált alkalmazások közül egyedül a MainActor zárt forráskódú. RPM-ből telepítettük az Alien segítségével szinte ugyanúgy, mint a Broadcast 2000-et.
A MainActorral (2D-s és 3D-s szövegekkel egyaránt) feliratozhatunk, és a mozgófilmet szerkeszthetjük. Dolgozhatunk képátmenetekkel és különféle hanghatásokkal. A próbaváltozat mindaddig a „MainActor” szöveget írja a filmünkre, amíg a bejegyzett példányt meg nem vesszük. A MainActor a következő alkalmazásokat tartalmazza: `maseq` (NLE), `mave` (átalakító), `macap` (V4L MJPEG-felvevő) és `lmatool` (karakteres videoállomány-átalakító). A leírás a */usr/share/doc/Packages/MainActor* útvonalon érhető el.

Trinity

Chris Hardy, a Trinity felelőse szerint az alkalmazás kezdetleges MPEG- és audiotámogatással rendelkezik és képsorozatokat is kezel. „A forráskód két éve érintetlen. A fejlesztőről egy ideje semmi hír, a projekt függőben van” – teszi hozzá. A felhasználói felület az egyik dolog, amiért Hardy szereti a Trinityt.
„A fordítás közben akadt egy-két kisebb hiba, amit a 0.5-ös változatban kijavítottunk, de előfordult egy komoly is. A javítást elküldtük Hardynak. Egy kereskedelmi programmal való ütközés miatt a Trinityt át kell keresztelni. Hardy örülne, ha egy NLE-ben érdekelt fejlesztő csatlakozna a projekthez, és folytatni lehetne a munkát.”

Összegzés

A számítógépek nagy előnye a filmgyártásban, hogy a segítségükkel könnyedén változtathatjuk a jelenetek sorrendjét, elhagyhatunk belőlük vagy újakat szúrhatunk be – ezért is nevezik nemlineáris szerkesztésnek. A nemlineáris szerkesztők képsorozatok szerkesztésére, végső soron televíziós vagy mozifilmek készítésére használatosak.
A Broadcast 2000, a Crow, a Kino, az LVE, a MainActor és a Trinity vizsgálatával a Linuxon elérhető videoeszközök közé pillantottunk be, természetesen a teljesség igénye nélkül. Az FFmpeg, a GAnSO, a Gnonlin, a Jahshaka, a Linux Video Studio, a matterial, az mpgtx, az mpegcut és a SAMPEG-2 további választási lehetőséget jelenthetnek.
Két kereskedelmi (és drága) eszköz vizsgálatát tervezzük a jövőben: a Nothing Real Shake-ét és a Silicon Grail RAYZ-ét. Számos hollywoodi mozifilm különleges hatásai készültek a közreműködésükkel. Az egyetlen nyílt forrású eszköz, amelyet nagyobb mozifilmek készítéséhez is használnak, a Film Gimp, amelyet a Harry Potter és a bölcsek köve, a Kutyák és macskák, A gyűrűk ura és egyéb filmeknél is alkalmaztak. A következő alkalommal erre is vetünk egy pillantást.

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

Átállás (2. rész)

Folytatódjék hát a kezdő és a leendő Linux-felhasználóknak szánt sorozatunk!

Sorozatunk első részben főként a Linux és a hozzá hasonló szabad felhasználású és ingyenes Unix-rendszerek előnyeit és hátrányait taglaltuk, mostani cikkünkben pedig a Windows és a Linux közötti legfontosabb különbségekre térünk ki. Ahogy már előző számunkban is tisztáztuk, a Linux Unix-alapú operációs rendszer. A két rendszer használatában ez a legfontosabb különbség, tehát ez az írásunk lényegében a Unix- és a Windows-rendszerek közötti eltérésekkel foglalkozik. Akinek már akadt dolga valamilyen Unix-rendszerrel, annak rengeteg dolog ismerős lesz a Linuxszal való ismerkedés során. A most leírtak tehát inkább azoknak szólnak, akik ezeket a rendszereket csupán hírből ismerik.

Az első, amit meg kell jegyeznünk, hogy a Linux egy valódi többfeladatos 32-bites operációs rendszer. A már őskövületnek számító Windows 3.1 például jóindulattal sem nevezhető operációs rendszernek, mivel tulajdonképpen nem más, mint grafikus héj a szöveges DOS-környezeten. Ugyan lehetővé tette, hogy „egyszerre" több alkalmazást is futtathassunk, de minden futó folyamat (process) maga dönthette el, hogy meddig kíván futni, és mikor adja át a futási lehetőséget a többi folyamat számára. Ez nem valami hatékony megoldás, mert ha egy alkalmazás lefagy, az egész rendszert magával rántja.

A Windows 95 megjelenésével javult valamit a helyzet, a folyamatok ütemezését már maga a rendszer végezte (habár messze elmaradt a Windows NT-től), és büszkén hirdette magáról 32-bites mivoltát. Ám ez utóbbi nagyon csalóka, mivel a régi alkalmazásokkal való megfelelősége (compatibility) következtében rengeteg 16-bites kódrészt hagytak a rendszerben, sőt a rendszer a 32-bites védett módból a 16-bites valós módba ide-oda kapcsolgatott. Ez pedig a rendszer üzembiztonságára nézve nagy veszélyt jelentett. A Windows 98/ME forgalomba kerülésével további fejlődést tapasztalhattunk: amit lehetett, átírtak 32-bites környezetbe, továbbá egyéb biztonsági szolgáltatásokkal is kiegészítették.

A Windows NT/2000/XP azonban teljesen más, ezek – a Linuxhoz hasonlóan – valódi többfeladatos és 32-bites operációs rendszerek. Minden alkalmazás egymástól teljesen elkülönített „térben" fut, és az egyik nem tud a másik dolgába „belerondítani".

A Linux magas szintű üzembiztosságát nem annak köszönheti, hogy kevesebb hibát tartalmaz, mint például a Windows. Az igazat megvallva, tele van hibával, hiszen ezt is emberek alkották. A különbség csak annyi, hogy egy alkalmazás lefagyása vagy helytelen működése korántsem okozhat akkora galibát, mint egy Windows 95/98/ME rendszerben.

Ez annak köszönhető, hogy egy folyamat nem férhet hozzá közvetlenül gépünk erőforrásaihoz, például rendszerünk memóriájához. Csak abba a tartományba írhat, ahová a rendszermag számára engedélyezi. Hiba esetén pedig a rendszermagnak jogában áll az alkalmazás futásának beszüntetése és a memóriából való „kitakarítása".

Tehát Linux alatt is szép számmal fagynak az alkalmazások, de ez csak a legritkább esetben okoz fennakadást a rendszer és a többi folyamat működésében. A Linux csak akkor dőlhet össze, ha vagy a rendszermagban vagy a gép szintjén történik lefagyás (ezek az esetek szerencsére elég ritkák), viszont egy alkalmazás egyedül nem képes erre.

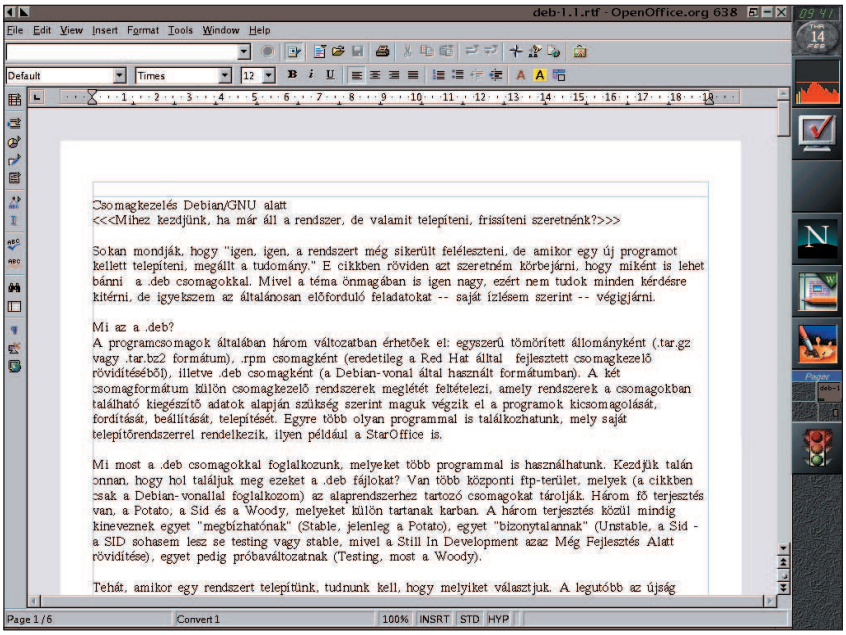
Most térjünk a tárgyra, azaz a Linux- és a Windows-alapú rendszerek különbségeinek bemutatására!

A Linux-rendszerek magja modularizált. Ez azt jelenti, hogy a rendszermag bizonyos részei különálló egységekben (modulokban) találhatók, amelyeket kedvünkre ki- és betölthetünk. Modulokban leggyakrabban a különböző eszközök támogatását szokás elhelyezni. Az ilyen modularizált rendszermagok előnye, hogy ezek a modulok csak akkor töltődnek be, amikor szükség van rájuk, egyébként nem foglalnak helyet a memóriában.

Azt hogy a rendszermag mely részei kerüljenek modulba és melyek ne, a rendszermag újrafordításakor mi is meghatározhatjuk. Előző számunkban említettük, hogy a rendszermag fordítása a Linux világában hétköznapi műveletnek számít, ugyanis számtalan előnye van, ha saját magunk készítette rendszermaggal dolgozunk. Egyrészt a mi processzorunkra lesz testreszabva (ez nagyobb futási sebességet eredményezhet), másrészt csak azok az elemek kerülnek közvetlenül a rendszermagba, illetve modulba, amelyekre szükségünk van (ezáltal a rendszermag mérete csökken).

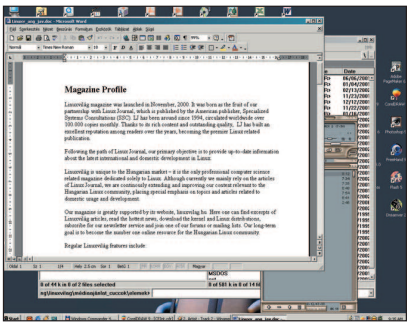
A Windowshoz szokott felhasználók ezt első látásra furcsa és bonyolult műveletnek tarthatják, pedig semmiféle programozási előképzettség nem szükséges hozzá.

Előző számunkban szintén megemlítettünk, hogy a Unixok elsősorban hálózati rendszerek. Az ilyeneket általában tehát nem egy, hanem több felhasználó használja, ezért bizonyos biztonsági szolgáltatásokra van szükség. A rendszernek például nem szabad engednie, hogy bármely felhasználó ész nélkül törölgethessen benne, írhasson bele stb. Ezért a Windows NT-khez hasonlóan, a munka megkezdése előtt itt is egy felhasználónév-jelszó párossal kell bejelentkeznünk. A `root`-felhasználó, vagyis a rendszergazda bír a legtöbb jogosultsággal. Csak ő képes például a különböző rendszerbeállításokat megváltoztatni, további felhasználókat létrehozni, illetve törölni stb. A rendszergazdára (kevés kivétellel) semmilyen korlátozás nem vonatkozik, bármit letörölhet – szabadon „garázdálkodhat" a rendszerben. Ezért a Unix világában nagyon fontos, hogy csak akkor lépjünk be rendszergazdaként, ha valami olyasmit szeretnénk elvégezni, amelyhez ezek a jogosultságok szükségesek. Egyébként hozzunk létre magunknak egy saját felhasználót, amivel a „hétköznapi" munkálatokat végezzük. A felhasználókat egyébként csoportokba is szervezhetjük. Ez azért hasznos, mert az adott csoportra meghatározott jogosultságok a csoportban lévő összes



Linuxos képernyő

felhasználóra vonatkoznak.
A Linux fájlrendszere az ext2. Itt minden állományhoz egy úgynevezett fájlleíró (i-node) tartozik, amely az adott fájl tulajdonságait (például méret, tulajdonos, lemezblokkok címei stb.) tartalmazza. Egyes Linux-terjesztések egyébként azt is lehetővé teszik, hogy a rendszert akár FAT lemezterületre is telepíthessük. Az újabb változatokban már az ext2 fájlrendszert annak továbbfejlesztett változatára, az ext3 fájlrendszerre cserélhetjük le. Ez már egy úgynevezett naplózó (journaling) fájlrendszer. A naplózó fájlrendszerek lényege, hogy a merevlemezünkön lévő adatszerkezetből naplót szerveznek, és az összes függőben lévő írásműveletet a napló végéhez csatolják. Így egy szegmensben fájlleírókat és adatblokkokat vegyesen találhatunk. A naplózó fájlrendszerek működése ennél azért sokkal összetettebb. A lényege az, hogy sok kisebb írásművelet esetén a naplózó fájlrendszerek használata gyorsabb működést eredményez, és például kevésbé érzékeny az áramszünet okozta fennakadásokra.
A linuxos fájlrendszereken kívül a rendszermag egyéb fájlrendszereket is támogat, például a FAT16-ot és a FAT32-őt, továbbá az OS/2 HPFS-ét és a Windows NT NTFS-ét.
A Unix-rendszerekben a különböző lemezegységek fájlrendszerei nincsenek különválasztva egymástól, mint a Windows-rendszerekben, hanem mindegyikük egységes, összefüggő könyvtárszer-



Windowsos képernyő

kezetet alkot. Ha például a gépünkbe helyezett CD-n található állományokhoz szeretnénk hozzáférni, akkor azt előtte be kell fűznünk ehhez a fájlrendszerhez. Ha megtörtént, CD-nk tartalmát egy megadott könyvtár alatt (általában a */cdrom*) találjuk. Ezt a műveletet befűzésnek (`mount`) hívjuk, sorozatunk későbbi számaiban e témára részletesen ki fogunk térni.
A Unix-rendszerek másik sajátossága, hogy szinte az összes rendszerbeállítást szöveges állományok tömkelegében tárolják. Ezek az állományok a */etc* könyvtárban találhatóak, és általában csak a rendszergazda módosíthatja őket. Azzal, hogy a különböző beállítások mely állományokban vannak, és azokat miként módosítatjuk, sorozatunk további részeiben foglalkozunk.
A rendszer bináris állományai (azaz maguk a programok) a */bin*, illetve a */sbin* könyvtár alatt lelhetők fel. Ezekben a könyvtárakban csak a legalapvetőbb felhasználói és rendszerfelügyeleti programok találhatók, a többiek a */usr/bin*, illetve a */usr/sbin* könyvtárban helyezkednek el.
A */home*-ban a felhasználók úgynevezett saját könyvtárait találhatjuk. A saját könyvtár szintén fontos fogalom a Unix világában. Ez nem más, mint a felhasználó egyéni könyvtára, ahol a magánjellegű dolgait tárolhatja. A saját könyvtár neve általában a felhasználó azonosítójával egyezik meg.
Mivel a Unix többfelhasználós rendszer, alapvető fontosságú meghatároznunk, hogy ki milyen állományhoz férhet hozzá. Minden fájlnak van egy tulajdonosa és egy csoportja. Az állományra vonatkozó jogokat a tulajdonos, illetve a rendszergazda határozhatja meg. Megmondhatja, hogy milyen jogok vonatkozzanak a tulajdonosra, a csoportra és mindenki másra. A Unixban alapvetően háromféle jogosultság létezik: olvasás (read), írás (write) és végrehajtás (execute). Az utóbbi értelemszerűen csak a futtatható állományoknál érdekes. Ezenkívül létezik még néhány úgynevezett módosító bit is, ezekről a későbbiek folyamán fogunk szót ejteni.
Egy másik unixos „szokás”, hogy az ilyen rendszerek alatti különböző eszközöket úgynevezett eszközállományok (*device files*) segítségével érhetjük el, amelyek a */dev* könyvtár alatt találhatók. Ha például a második soros kapunkra akarunk hivatkozni, ami a DOS-ban a COM2 névre hallgat, ezt Unix alatt a megfelelő eszközfájl nevének megadásával tehetjük meg (jelen esetben */dev/ttyS1*). A merevlemezeknek és a gépünkben lévő többi meghajtónak (drive) is megvan a maga megfelelő eszközfájlja. Például az elsődleges ATAPI merevlemeznek a */dev/hda*, az azon található első lemezterületnek pedig */dev/hda1* a neve.
Sorozatunk következő részében már a gyakorlati oldalról vizsgáljuk meg a Linuxot: megismerkedünk néhány alapvető utasítással, és bemutatunk néhány egyszerű rendszerfelügyeleti műveletet.

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 (3. rész)

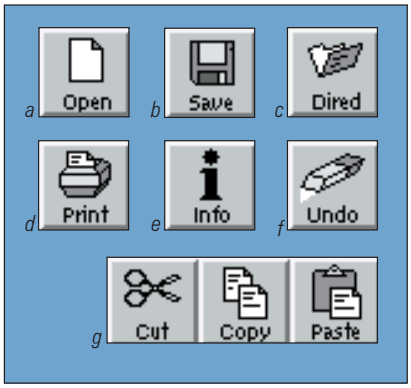
Sorozatunk e részében az Open párbeszédablak kínálta lehetőségekben mélyedünk el.

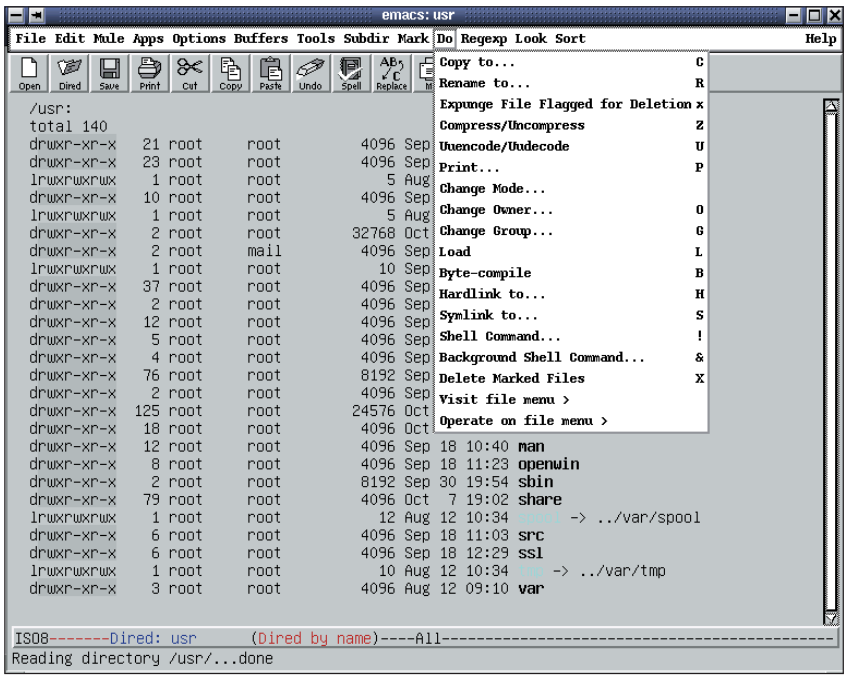
Térjünk vissza a sorozatunk előző részében taglalt fájlbetöltés példájához, hasonlítsuk össze a unixos életérzést tükröző billentyűs állomány megnyitási módját a Windowsban megszokott párbeszéddobozzal! Ehhez az Open ikont kell megnyomnunk (*1a kép*).

Az XEmacs természetesen nem lenne Emacs-változat, ha szokványos ***Fájl, Megnyitás*** párbeszédablakot kapnánk. Ha megnézzük a legfelső sort, ott a „Possible Completions are:" szöveget olvashatjuk. Tehát elvileg nem állományokat látunk magunk előtt, hanem lehetséges kiegészítéseket. A kiegészítéseket a parancssorról ismerhetjük, ahol a TAB nyomogatásával az általunk beírt parancsok mindig kiegészülnek, vagy – ha a helyzet nem egyértelmű – egy listát kapunk. A mini átmeneti tároló is ott van legalul, hogy még akkor is gépelhessünk, amikor erre látszólag semmi szükség. De a kiegészítések egyben szűrőként is működnek, hiszen amint beírjuk ide az első betűket és nyomogatjuk a TAB billentyűt, úgy látunk egyre kevesebb állományt az ablakban, míg végül megkapjuk az általunk keresettet. Ne felejtsük el, hogy Unixban a fájlkiterjesztéseknek nincs olyan meghatározó szerepük, mint Windowsban, ezért Unix-állományok esetében a *.* típusú szűrés nem mindig célravezető!

Ahogy az ablakban navigálunk, a visszhangterületen mindig láthatjuk, hogy éppen hol járunk. A jobb egérrel vagy a NYÍL billentyűkkel kiválaszthatjuk a kívánt állományt, majd ENTER-t nyomva beolvashatjuk. A fentiek ismeretében talán már nem is tartjuk olyan hasznosnak az XEmacs ***Fájl Megnyitás*** párbeszédablakát, különösen akkor, ha tudjuk, hogy a mini átmeneti tárolóba begépelt C-x C-f után a TAB kétszeri megnyomásával kiegészítéslistát kapunk, ahonnan szintén egérkattintással választhatunk vagy beírhatjuk a kívánt állomány nevét. A kiegészítések nemcsak állományok meglátogatásakor működnek, hanem parancsok bevitelekor is. A TAB mellett a szóköz billentyű is használható kiegészítésre, de ez csak egy szóig egészíti ki a beírt szöveget. Ha egy nem teljesen beírt parancsnál az ENTER-t nyomunk, a parancs kiegészül, és ha lehetséges, végrehajtódik. Ha nem, listát kapunk. A program itt is elkészíti a lehetséges kiegészítések listáját, ha a ? karaktert gépeljük be. A mini átmeneti tárolóban a le és a fel nyilakkal választhatunk a korábban behívott fájlok közül, majd az ENTER megnyomásával döntésünket jóváhagyhatjuk.

Az Emacs-szakzsargon szerint nem megnyitjuk a kívánt állományt, hanem ***meglátogatjuk***. Ez azt jelenti, hogy a program átmeneti tárolót hoz létre, majd bemásolja oda a megnyitott állományt,





2. kép

és megjeleníti egy ablakban. A továbbiakban tehát az átmeneti tároló tartalmát szerkesztjük, nem a megnyitott fájlt. Változtatásaink csak akkor fognak a merevlemezen vagy más adathordozón tárolt állományban megjelenni, ha a C-x C-s teljes billentyűvel mentjük.

Mentés

Az Emacs bizonyos időközönként menti a fájlokat, és az ilyen ideiglenes állományokat elöl és hátul # karakterrel jelöli meg, például ***#mule.txt#***. Az önműködő mentés nem az eltelt időtől függ, hanem a begépelt karakterek számától, ami alapértelmezetten 300 leütés. Ezt az értéket az `auto-save-interval` változó tárolja, aminek az értékét az ***Options, Customize, Emacs, Data, Auto Save, Interval*** menüpontban változtathatjuk meg. Mi magunk is végezhetünk biztonsági mentést az ***M-x do-auto-save*** paranccsal. Esetleges áramszünet vagy más gond esetén ezeket a külön mentett

3. kép

állományokat használhatjuk a baj előtti állapot visszaállítására az ***M-x recover-file RET fájl neve RET*** paranccsal. Az Emacs az önműködő mentést ilyenkor kikapcsolja, mert jogosan feltételezi, hogy a visszaállított fájlban értékes adatok vannak. Amikor végeztünk a helyreállítással, a C-x C-s paranccsal nekünk magunknak kell mentenünk, majd az ***M-x auto-save-mode RET*** begépelésével az önműködő mentést újra be kell kapcsolnunk. Az XEmacsban a ***Save*** ikonnal is menthetünk. (*1b kép*)
Az Emacs a biztonsági mentések mellett biztonsági másolatokat is készít, amelyeket a ~ jellel jelöl meg, például ***mule.txt~***.

A Dired

A Dired a ***Directory Editor*** (Könyvtárszerkesztő) szó összevonásából ered, segítségével könnyen mozoghatunk a könyvtárakban, és szerkeszthetjük őket. (*1c kép*)
Az ikon megnyomása helyett gépelhetjük a C-x d teljes billentyűt vagy az ***M-x dired*** parancsot, hogy a kisméretű átmeneti tárolóba beírhassuk a kiválasztandó könyvtár nevét. Döntésünket az Enter megnyomásával hagyhatjuk jóvá. Próbáljuk ki a fentihez hasonló C-x C-d parancsot is, ami egy megadott könyvtárat listáz ki. Látjuk, hogy a C-x d a Könyvtárszerkesztő megnyitását végző és a C-x C-d listázó parancsok billentyűkombinációi csak egy helyen térnek el egymástól. A különbség annyi, hogy az első esetben a Ctrl billentyűt fel kell engedni, mielőtt a D billentyűt nyomnánk meg, a második esetben az X és a d billentyűk megnyomásakor a Ctrl-t nyomva tarthatjuk, ha akarjuk, de akár fel is engedhetjük az X után, hogy utána ismét lenyomhassuk. A C-x 4 d parancs hatására a Könyvtárszerkesztő új ablakban fog megjelenni. Ha már belekerültünk a Könyvtárszerkesztőbe, vigyáznunk kell, mert a közönséges billentyűk most csúnya dolgokat művelhetnek! Ha y állomány fölött állunk, akkor a D vagy a C-d törlésre jelöl ki, amit a sor elején megjelenő D betű és az állomány elszíneződése mutat. A # és ~ billentyűkkel a könyvtárban lévő összes biztonsági mentés és biztonsági másolat egy lépésben kijelölhető törlésre. Az U megnyomása eltávolítja a kijelöléseket, de ha az X-et nyomjuk meg, a kijelölt fájlok valóban törlődnek! Szerencsére törlés előtt rákérdez, hogy valóban végrehajtsa-e ezt az immár visszavonhatatlan műveletet. Ha sok állományt jelöltünk ki egyszerre törlésre, az U fentről lefelé érvényteleníti a törlést, a Del (vagy Delete) alulról felfelé. Még egyszer hangsúlyozom, a kijelölés még nem jelent törlést, ahhoz az X billentyűt kell megnyomni! A Könyvtárszerkesztőben a szerkesztőutasítások közt szerepel az átnevezés,

A legfontosabb állománykezelő teljes billentyűk és parancsok

C-x C-f	meglátogat egy állományt;
C-x 4 f	egy másik ablakban látogatja meg az állományt;
C-X 5 F	új keretben látogatja meg az állományt;
C-x C-r	csak olvasásra látogatja meg az állományt;
C-x C-v	új állományt hív be a mostani helyébe;
C-X I	egy másik fájl tartalmát szúrja be az átmeneti tárolóba a kurzornál;
C-x C-q	egy eredetileg csak olvasásra megnyitott állományt írhatóvá tesz az átmeneti tárolóban, szerkesztés után más néven kell menteni;
C-x C-w	más néven menti az állományt;
C-x C-s	az állományt kimenti a lemezre;
C-x s	az összes állományt lemezre menti;
M-~	ha egy fájt semmiképpen sem akarunk kimenteni, még ha meg is változtattuk, ezzel a billentyűvel megakadályozhatjuk a mentést, mivel úgy teszünk, mintha az átmeneti tár tartalma nem változott volna;
M-x view-file RET fájlnév RET	egy állományt úgy olvas be, hogy mindig egy képernyőnyit mutat belőle;
M-x delete-file RET fájlnév RET	törli a megnevezett fájlt;
M-x copy-file RET régi fájlnév RET új fájlnév RET	a régi fájlnevű állományt új fájlnevűbe másolja;
C-u C-x C-s	a most mentett állomány a következő mentésnél biztonsági másolattá válik;
C-x C-d könyvtárútvonal RET	röviden kilistázza egy könyvtár tartalmát;
C-u C-x C-d könyvtárútvonal RET	szószátyár listát ad a könyvtár tartalmáról;
M-x make-directory RET könyvtárnév RET	létrehozza a megadott nevű könyvtárat;
M-x delete-directory RET könyvtárnév RET	törli a megadott nevű könyvtárat;
M-x rename-file RET régi fájlnév RET új fájlnév RET	a régi fájlnevet új fájlnévre nevezi át.

Néhány átmeneti tárat kezelő parancs

M-x rename-buffer RET tároló név RET	megváltoztatja a működő átmeneti tároló nevét;
M-x rename-uniquely	egyedien nevezi át a működő átmeneti tárat, úgy, hogy az addigi tárolónévhez egy számot ad, például az *emacs.rtf* névből *emacs.rtf<2>* lesz;
M-x revert-buffer	az átmeneti tár tartalmát állítja vissza ismételten meglátogatva a lemezen lévő eredeti állományt;
M-x view-buffer RET átmeneti tároló neve RET-olvasásra	beolvas egy már meglátogatott állományt.

Az átmeneti tárolók tartalma

MR Buffer	Size	Mode	File
. % cbx.png	0	Image	/usr/share/xemacs/21.1.14/etc/cbx.png
*RMain.pas	53165	Pascal	/home/ratio/ratiosoft/emacs/src/RMain.pas
*map.c	7586	C	/home/ratio/ratiosoft/emacs/src/map.c
gnats.el	77675	Emacs-Lisp	/home/ratio/ratiosoft/emacs/src/gnats.el
gnuslogo.ps	64315	PostScript	/home/ratio/ratiosoft/emacs/gnuslogo.ps
% calccard.tex	20266	LaTeX	/usr/share/xemacs/21.1.14/etc/calccard.tex
% *info*	21971	Info	/usr/share/xemacs/info/dir
software.html	68428	HTML	/home/ratio/ratiosoft/software.html
scratch	197	Lisp Interaction	
% *Completions*	424	Completion List	
* "*Buffer List*"	751	Buffer Menu	

amihez elegendő a nagy R betűt (SHIFT-R) megnyomnunk, a nagy C betű átmásolja, az f pedig megnyitja az állományt, az o úgy viselkedik, mint az f, de az állományt új ablakban nyitja meg. Ha a v betűt nyomjuk meg, akkor olvasásra nyitjuk meg a fájlt (ugyanaz, mint a ***File, View Fájl…*** menüpont vagy az ***M-x view-file*** parancs). Ha egy fájlt csak olvasásra nyitottunk meg, akkor nem tudjuk szerkeszteni, hacsak be nem ütjük a C-X C-Q teljes billentyűt, ami az átmeneti tár tartalmát szerkeszthetővé teszi, de nem magát az eredeti állományt! Ilyenkor a munka befejeztével a fájlt más néven kell menteni. A ***File, Save some buffers*** vagy a C-X S az összes fájlt kimenti. Ha rossz fájlt töltöttünk be egy ablakba, a C-X C-V paranccsal kicserélhetjük. A ***File, Insert File…*** vagy C-X I egy teljes fájlt nyit meg és szúr be a már szerkesztett állományba a kurzor helyén. Nem meglepő, hogy az új állománynevek beírásához a fenti utasítások jó része a kisméretű átmeneti tárolót használja.
Figyeljük meg azt is, hogy az XEmacsban időnként megváltozhat a menü vagy az ikonsor. Amikor belépünk a Könyvtárszerkesztőbe, új menüt kapunk (lásd a *2. képet* a 40. oldalon).
Ha kapcsolatunk van a Világhálóval, a beépített böngészőt az ***Apps, Browse the Web*** menüvel indíthatjuk el. Most nemcsak a menü, hanem az ikonok is megváltoznak (lásd a *3. képet*).

Nyomtatás

Egy jól beállított Linux-rendszeren a nyomtatást nem kellene külön megemlítenem, hiszen csak a ***Print*** ikont szükséges hozzá megnyomnunk, de az XEmacsban mindjárt kétféle nyomtatási mód közül választhatunk (lásd az *1d képet* a 40. oldalon).
A ***File, Print Buffer FájlNév*** menüpontban a pillanatnyi fájlt egyszerűen nyomtathatjuk ki, míg a ***File, Pretty-Print Buffer FájlNév*** szebben nyomtat: csinos fejlécet tesz a lap tetejére – kiírva az oldalszámokat és a dátumot, valamint a fájl nevét és az elérési útját. Természetesen ezeket a szolgáltatásokat a kisméretű átmeneti tárolóból is indíthatjuk, de az ***M-x print-buffer*** parancs nem nyomtat annyira csinosan, mint a ***Pretty-Print Buffer***, de egyszerű dátumos és oldalszámos fejlécet azért tesz a lap tetejére. Az ***M-x lpr-buffer*** parancs nem nyomtat fejlécet. Az ***M-x print-region*** fejléccel, az ***M-x lpr-region*** fejléc nélkül nyomtatja ki a kijelölt területet, de nem az egész fájlt.

Átmeneti tárak

Ahogy már korábban említettem, az Emacs a megnyitott állományokat átmeneti tárolókban (buffer) tárolja. Sok átmeneti tároló létezhet egy időben, de közülük mindig csak egy lehet működő. Mindegyik átmeneti tárolónak külön neve van, amit az ***M-x rename-buffer*** paranccsal változtathatunk meg. Átnevezéskor vigyázzunk arra, hogy ne hozzunk létre névazonosságokat, és arra is, hogy a kis- és nagybetűk különböznek. Az átmeneti tárolók egyedi azonosítói segítenek abban, hogy kiválaszthassuk őket a ***Buffer, List All Buffers*** menüben. Az összes átmeneti tárolót a C-X C-B billentyűkkel is kilistázhatjuk, majd a kapott felsorolásból a középső egérgombbal vagy a jobb egérgomb két kattintásával, esetleg az ENTER használatával választhatunk:

Az ***Átmeneti tároló listában*** olvashatjuk az átmeneti tárolók nevét, méretét, a megnyitás módját és az állomány abszolút elérési útját. Amelyik átmeneti tároló előtt egy * van, azt már módosítottuk. A % (százalékjel) mutatja, hogy az állomány csak olvasható, nem írható. A . (pont) jelzi, hogy melyik a működő átmeneti tároló. Ha túl sok átmeneti tároló van nyitva, választhatunk ebből a listából, és a ***C-x k TárolóNév*** paranccsal kiirthatjuk őket. Ha nem adunk meg nevet, hanem a parancs kiadása után egyszerűen megnyomjuk az ENTER billentyűt, akkor a jelenlegi átmeneti tárolót irtjuk ki. Ilyenkor a legutóbb használt átmeneti tároló válik kiválasztottá. Ha az átmeneti tárolóban nem mentett adatok vannak, az Emacs felajánlja a mentést.
Az ***M-x kill-some-buffers*** parancs egyenként végigkérdezi, hogy melyik átmeneti tárolót akarjuk törölni. A ***yes*** begépelése töröl, a ***no*** a következő átmeneti tárolóhoz ugrik tovább.
A ***C-x b TárolóNév*** paranccsal is válthatunk az átmeneti tárolók között. Ha ilyenkor az átmeneti tároló nevének begépelése helyett egyszerűen ENTER-t ütjük le, az előzőleg szerkesztett átmeneti tárolóba jutunk. Itt is igaz az, amit az állományok megnyitásakor mondtam: ha az adott nevű átmeneti tároló nem létezik, az Emacs létrehoz egyet a számunkra. Az ilyen, fájlokhoz nem kötődő átmeneti tárolókat jegyzetelésre használhatjuk, de ha akarjuk, a végén menthetjük is őket. A C-X B teljes billentyűhöz az ***M-x switch-to-buffer TárolóNév*** parancs van kötve.
A ***C-x 4 b PufferNév*** új ablakban mutatja meg a kiválasztott átmeneti tárolót, és azonos az ***M-x switch-to-buffer-other-window PufferNév*** paranccsal. Az ***M-x switch-to-buffer-other-frame*** a megnevezett átmeneti tárolt új keretbe hívja be. Ez utóbbi parancs a C-X 5 B billentyűkombinációhoz van kötve.
Sorozatunk következő részében az ablak- és keretkezeléssel foglalkozunk.

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

Csomagkezelés Debian/GNU alatt

Mihez kezdjünk, ha már áll a rendszer, de valamit telepíteni, frissíteni szeretnénk?

Sokan mondják, hogy „igen, igen, a rendszert még sikerült felélesztени, de amikor új programot kellett volna telepíteni, megállt a tudomány.” E cikkben azt szeretném röviden körbejárni, miként is lehet csomagokkal bánni. Mivel a téma önmagában is igen nagy falat, nem tudok minden kérdésre kitérni, de igyekszem az általánosan előforduló feladatokat – saját ízlésem szerint – sorra járni.

Mi az a .deb?

A programcsomagok általában három változatban érhetők el: egyszerű tömörített állományként (***.tar.gz*** vagy ***.tar.bz2*** formátum), ***.rpm*** csomagként (eredetileg a Red Hat által fejlesztett csomagkezelő rövidítéséből), illetve ***.deb*** csomagként (a Debian-vonal által használt formátumban). A két csomagformátum külön csomagkezelő rendszerek meglétét feltételezi, amelyek a csomagokban található kiegészítő adatok alapján szükség szerint maguk végzik el a programok kicsomagolását, fordítását, beállítását és telepítését. Egyre több olyan programmal találkozhatunk, amely saját telepítőrendszerrel rendelkezik, ilyen például a StarOffice is.

Most a ***.deb*** csomagokkal foglalkozunk, melyeket több programmal is használhatunk. Kezdjük talán onnan, hogy hol találjuk meg ezeket a ***.deb*** fájlokat? Több központi ftp-terület is létezik, melyek (a cikkben csak a Debian-vonallal foglalkozom) az alaprendszerhez tartozó csomagokat tárolják. Három fő terjesztés létezik: a Potato, a Woody és a Sid, melyek karbantartása külön zajlik. A három terjesztés közül egyet mindig „megbízhatónak” neveznek ki (Stable, jelenleg ez a Potato), egyet „bizonytalannak” (Unstable), egyet pedig a próbaváltozatnak (Testing, most éppen a Woody); a Sid pedig sohasem lesz se testing, se stable, mivel Still In Development, azaz „még fejlesztés alatt” a rövidítése.

Amikor rendszert telepítünk, tudnunk kell, hogy melyiket válasszuk. Lapunk lemezmellékletére legutóbb a Woody került fel. Ha a saját terjesztésünkhöz tartozó anyagokat akarjuk megtalálni, nem mindegy, hogy melyikben keresgélünk. Igaz, az esetek túlnyomó részében a más terjesztésből származó csomagok is működnek, illetve számos csomagot külön helyekről kell beszerezni, erről azonban egy kicsit később szólok.

APT, a kényelmes megoldás

Most egy kicsit rendbontó leszek, és előreveszem az `apt-get` programot, ezzel ugyanis nagyon sok mindent meg lehet oldani. Röviden írok róla, hiszen már foglalkoztunk vele.

Az apt-get felkeresi a ***/etc/apt/sources.list*** fájlban megadott helyeket, és az ott található csomagokkal dolgozik. Itt megadhatunk helyi területet (***file:*** vagy ***cdrom:***), webes elérést (***ftp:*** vagy ***http:***), illetve titkosított kapcsolatot (***ssh:***). Nagyon jó leírásra bukkanhatunk a `man sources.list` parancs kiadásával.

Amennyiben az apt által elérhető csomagok listáját frissíteni szeretnénk, adjuk ki az `apt-get update` parancsot; ha mondjuk a Gimpet kívánjuk telepíteni, akkor az `apt-get install gimp` parancsot akkor, ha pedig azt szeretnénk, hogy a kezelő a telepítés óta frissült összes csomagot önműködően frissítse, az `apt-get upgrade` parancsot használjuk.

1. lista Hibaüzenet

```
#apt-get install joe
Reading Package Lists... Done
Building Dependency Tree... Done
You might want to run 'apt-get -f install'
➥to correct these:
Sorry, but the following packages have unmet
➥dependencies:
  joe: Depends: libncurses5 (>= 5.2.20010310-1)
  ➥but 5.0-6 is to be installed
  libc6-dev: Depends: libc6 (= 2.1.3-10)
  ➥but 2.2.4-1 is to be installed
              Conflicts: libc-dev
              Conflicts: libdl1-dev but it
              ➥is not installable
              Conflicts: libdb1-dev
              Conflicts: libgdbm1-dev but it
              ➥is not installable
              Conflicts: libpthread0-dev but
              ➥it is not installable
  locales:E: Unmet dependencies. Try
  ➥'apt-get -f install' with no packages
  ➥(or specify a solution).
```

No, itt kezdődnek a turpisságok. Mert mi történik, ha az *1. listán* látható üzenetet kapunk?

Ilyenkor a szerencsétlen áldozat megpróbálkozik a hibaüzenetben javasolt `-f` kapcsolóval. Ez többnyire segít. Na, nem mindig. Előfordulhat például, hogy egy-egy csomag nem található meg a forrásterületen, pedig ott kellene lennie. Velem esett meg, hogy amikor a `ktimer*` csomagot kerestem, amelyből a lista szerint a 2.2.2-8-asnak kellett volna a tükrön lennie, már csak a 2.2.2-8.1-esre bukkantam. Ez az a pont, amikor kézzel kell megoldanunk a helyzetet.

Egy szinttel lejjebb: dpkg

Az `apt-get` a háttérben valójában a `dpkg` programot hívogatja. A fenti példánál maradva, ha a kezelőrendszer nem talál meg egy csomagot, magunknak kell a letöltési területre eltéblábolnunk, majd letöltés után a `dpkg` segítségével telepíteni:

```
# lynx --source ftp://...akarhol-is-van/
# ktimer_2.2.2-8.1_i386.deb > /opt/source/
# ktimer_2.2.2-8.1_i386.deb
# dpkg -i /opt/source/ktimer_2.2.2-8.1_i386.deb
```

Természetesen egy idő után a tükör is helyesen tartalmazza a csomagot, semmi baj, az `upgrade` képes ezt a helyzetet kezelni. Egyébként érdemes egyszer végignézni, mi mindenre képes ez a csuda ügyes program, majd komolyan megfontolni

2. lista Az atp használata

```
ferike:~# apt-get install ace-of-pen*
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  ace-of-penguins
The following NEW packages will be installed:
  ace-of-penguins
0 packages upgraded, 1 newly installed,
➥0 to remove and 0  not upgraded.
Need to get 333kB of archives. After
➥unpacking 595kB will be used.
Do you want to continue? [Y/n]
Get:1 http://dexter sid/main ace-of-penguins
➥1.1-2.1 [333kB]
Fetched 333kB in 0s (435kB/s)
Selecting previously deselected package
➥ace-of-penguins.
(Reading database ... 38202 files and
➥directories currently installed.)
Unpacking ace-of-penguins (from
➥..../ace-of-penguins_1.1-2.1_i386.deb) ...
Setting up ace-of-penguins (1.1-2.1) ....
```

3. lista Részlet a /etc/apt-proxy/apt-proxy.conf fájlból

```
add_backend /debian/                          \
   /mnt/store/apt-proxy/debian/               \
   ftp.linuxvilag.hu::mirrors/debian/         \
   ftp.fsn.hu::linux/debian/                  \
   ftp.de.debian.org::debian/                 \
   ftp2.de.debian.org::debian/
```

a `dpkg -h` által kiírt utolsó sort: „Felhasználóbarát csomagkezeléshez a dselect programot használjuk”.

Csili-vili csomagkezelés: dselect és aptitude

Bár először mindenkinek ezt a programot ajánlják, nem használom túl sokat. Sőt, hogy teljesen őszinte legyek, egyetlen izgalmas szolgáltatását alkalmazom: ha olyan csomagot keresek, amelynek nem tudom pontosan a nevét, elindítom a programot, belépek a ***Select*** területre (S, ENTER, majd SZÓKÖZ), és itt szövegesen keresek rá (/ (perjel), majd a keresett szövegrész). Ha például arra vagyok kíváncsi, hogy melyik csomagban is vannak a más rendszerekből már ismert pasziánszjátékok, rákeresek a `Solitaire` szóra, mire a rendszer az ***ace-of-penguins*** sorra ugrik. Ha telepíteni szeretném, ütök egy + (pluszjelet), majd az ENTER segítségével elhagyom a terepet. Ekkor szinte biztos, hogy a `dselect` hosszas magyarázkodásba kezd, hogy mit kell még telepítenie. Miután itt megadtuk a függőségeket (kibogoztuk a szálakat), a rendszer a kért csomagokat telepíti.

A másik megoldás, hogy a csomag megtalálása után kilépünk a `dselect`-ből (CTRL-C), majd `apt-get install` *csomagnØv*. Megjegyzem, a csomagnév megadásakor helyettesítő karaktert is használhatunk.

Ha valaki kedveli a menüvezérelt csomagkezelőket, érdemes egy pillanatra elkalandoznia az `aptitude` világába. Ez a

4. lista Példa a /etc/rsyncd.conf fájlra

```
log file = /var/log/rsync.log
pid file = /var/run/rsync.pid

[debian]
 comment = A Debian tukor
 path = /var/ftp/debian
 max connections = 20
 read only = yes
```

kezelő is csupán egy „felület”, a háttérben szorgalmasan hívogatja a többieket. Az `aptitude` megjelenésekor elég szegényesnek látszik, ugyanis csupán faszerkezetet mutat. Aki viszont hozzászokik a használatához, döbbenetes sebességgel lesz képes a szükséges feladatokat megoldani. Nagy segítség, hogy a fában külön találjuk a telepített és a nem telepített csomagokat, a változatszámukkal együtt.

Használata hasonlít a többi hasonszőrű programéra. A / gombbal tudunk keresni, a + jelöli ki a csomagokat telepítésre.

A válogatás után a program a *g* hatására új listára vált át, melyben a telepítendő csomagokat láthatjuk, a *g* újbóli megnyomására pedig elindítja a letöltést. A programot most részletesen nem ismertetem, szerencsére eléri az egyik ismerősöm megadta „felhasználóbarát alapkövetelményt”: kiírja, hogy a *?* segítségével kapjuk meg a használathoz szükséges alapvető parancsok leírását.

Ha már itt tartunk, gyorsan szembe kell néznünk a csomagkezelők nagy rákfenéjével: „Jó, jó, de mi történik akkor, ha csak egy rusnya modemes kapcsolatunk van? Hogyan frissítsük vagy gyűjtsük be a szükséges csomagokat, és miként végezzük a karbantartásukat?”

Nézzük gyorsan végig, hol és miként érdemes beszerezni és tárolni a csomagokat. A legeslegjobb, ha folyamatos és terhelhető internetkapcsolattal rendelkezünk. Ekkor elég, ha a */etc/apt/sources.list*-ben megadjuk egy megbízható tükör címét, és a gép minden munkát a címről végez. De mi történik, ha nincs folyamatos kapcsolatunk? Ekkor több lehetőségünk kínálkozik: gyűjtögetjük a lemezeket, egyesével beolvastatjuk őket (`apt-cdrom add`); egyesével letöltögetjük a telepítendő csomagokat; elkészítjük a letöltendő csomagok listáját, majd a szomszédban található jó kapcsolaton keresztül töltjük le; helyi tükröt hozunk létre; vagy letöltőügyfelet üzemelünk be.

Az első két lehetőség magáért beszél, nézzük a többit.

Letöltési listák: apt-zip

Azok számára, akik nem a célgépen kívánják letölteni a csomagokat, ez a két programocska nagyon hasznos lehet. Az első, az `apt-zip-list` elkészít egy héjprogramot, amelyet egy távoli gépre magunkkal vihetünk (diákotthonos ismerősök előnyben), a távoli gépen a héjprogram a kívánt csomagokat letölti egy kellően nagy hordozható tárolóra (ziplemez, hordozható merevlemez stb.), és a lemezt a célgépre visszahozva az `apt-zip-inst` egyszerűen futtathatjuk programot, mely elvégzi a telepítéseket.

Nézzük meg, hogy hogyan működik mindez a gyakorlatban! Ha valaki például hordozható merevlemezt használ, és maga szeretné a befűzéseket kezelni, használhatja a `--medium` *k nyvtÆrnØv* `--skip-mount` kapcsolókat, így a két elkészítendő fájl a megadott könyvtárba kerül, a program pedig nem kísérletezik a kérdéses könyvtár befűzésével (alapértelmezésben

5. lista Forrás megadása

```
$ rsync ftp.fsn.hu::
Welcome to the ftp.fsn.hu anonymous rsync
➥service.

Please email suggestions and questions to
➥bra@fsn.hu

You can access the archive using 'rsync'
➥program like
rsync -av
ftp.fsn.hu::[Module]/path/to/the/files

Module          Content
-------------------------------------

ftp              FTP area
linux            Linux distributions
cdimages         CD-ROM Images
$
```

a */ZIP*-et befűzi, oda dolgozik, majd a művelet végén ki is fűzi). Az `apt-zip-list` két fájlt hoz létre. Az első a `fetch-script-wget-gØpnØv`, mely a távoli gépen futtatandó héjprogram. Ha belenézünk, a fájl végén láthatjuk a ténylegesen letöltendő csomagok listáját. Ezt a listát a program magától készíti el, az apt-vel a telepítésre kijelölt, de még le nem töltött fájlok listájából. Fontos még megjegyezni, hogy a program jelenleg a `wget`-et használja – figyeljünk rá, hogy a letöltést végző környezetben rendelkezésre álljon.
A munka vége még egyszerűbb. Miután a lemezt visszahoztuk, igény szerint befűztük, kiadjuk az `apt-zip-inst` parancsot (az eszköz kezelésére vonatkozóan lásd a fenti megjegyzéseket), ami az újonnan hozott csomagokkal a kért műveleteket elvégzi.

Tükrök használata

A tükrök felállítása külön feladat, melyre most nem térek ki, egy-egy ilyen területet számos módon ki lehet alakítani, teljes tükröt viszont csak akkor érdemes felállítani, ha a helyileg elérhető tükör megéri a hálózati terhelést. Szintén érdemes áthozni egy „alapot”, amennyiben nem rendelkezünk nagyon erős kapcsolattal, egy tükör ugyanis akár húszgigás területet is elfoglalhat (amiben a teljes fa minden ága nincs is benne). Mindenképpen megéri viszont tükröt üzemeltetni, ha a hálózatban legalább öt-hat debianos gép van, ugyanis mind a frissítés, mind a teljes telepítés nagyságrendekkel felgyorsulhat. Azonkívül ötvözni is lehet a módszereket, például a gépeken forrásként a helyi tükröt és a letöltőügyfelet is megadhatjuk. Amit szem előtt kell tartani, az, hogy a meglévő sávszélességet ne terheljük feleslegesen.

Letöltőügyfél kis hálózatoknak: apt-proxy

A végére hagytam a nekem legjobban tetsző megoldást. Ez a kis programocska kifejezetten olyan felhasználócsoport számára készült, akik ugyanazokat a fájlokat valószínűleg többször is felhasználják, de nem szeretnék az egész Debian-fát folyamatosan tükrözni. A program helyi gyorstárat üzemeltet és a karbantartását is elvégzi. Ha valaki gyorstárazott fájlt kér tőle, HTTP-protokollon keresztül szolgálja ki. Ha a keresett fájl nincs meg a gyorstárban, a program a háttérben az `rsync` segítségével letölti, majd továbbítja. Ez a felhasználó számára elméletileg „áttetsző”, azaz nem okoz fennakadást.
Természetesen a gyorstárterület egyúttal tükör is lehet, kis sávszélesség esetén például hasznos megoldás a nagyobb frissítéseket lemezekről bemásolni (hétvégenként tükrözni stb.), amire pedig napközben van szükség, azt az `apt-proxy` lehúzza. Nagy előnye még, hogy egyszerre több forrásból is képes dolgozni (csak rsync-protokollon keresztül), ezért több forrást is meg tudunk adni neki, amelyeket a keresett fájlért sorrendben végigjár.

Az apt-proxy beállítása

A program telepítése után nagyon jó súgóoldal (`man apt-proxy`) segít a telepítésben. A fejlesztő ígérete szerint a telepítési lépéseket szép lassan beépíti a csomagba, így nem kell sok további lépést megtennünk. Addig is röviden: a */etc/inetd.conf*-ban engedélyezni kell a 9999-et, létre kell hozni egy felhasználót, ami alatt a program dolgozik (alapértelmezés: apt-user), majd a */etc/apt-proxy/apt-proxy.conf* fájlban be kell állítani a gyorstár helyét és a tükröket.
Itt az `add_backend` sorok az érdekesek. Három részből állnak: az első a kérelem előtagja szerinti illesztés (ahogy a felhasználó által kért fájl kezdődik), a második a gyorstár a helyi lemezen, a többi pedig a használható források címe (`rsync`-formátumban).

Röviden az rsyncről

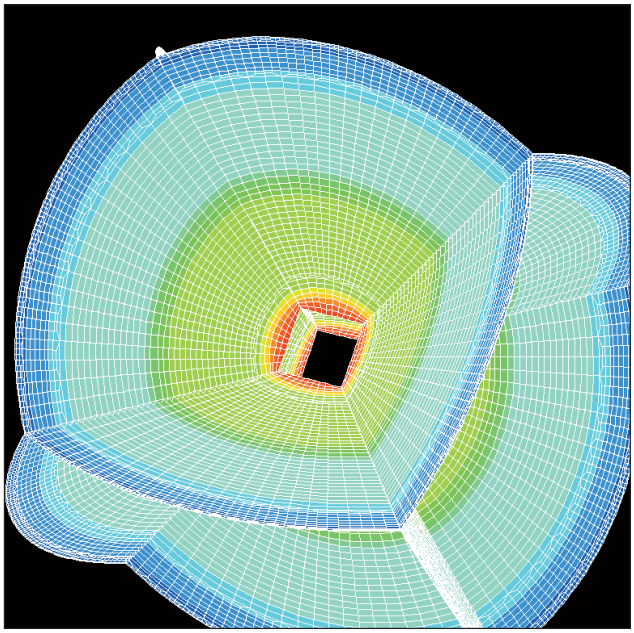
Az `rsync` hálózati fájlátviteli program, mely egy gyors ellenőrzés és hasonlítás segítségével csak a megváltozott részeket továbbítja. Más szóval: ha például egy negyven megás fájl végéhez hozzáfűzünk egy új sort, az `rsync` nem tölti le újból az egész fájlt, csupán azt a pár kilobájtot. Na jó, ennél egy kicsit bonyolultabban működik, de a lényege ennyi.
Egy terület megosztása `rsync` protokollon keresztül nem túl nehéz, csupán az `rsync` telepítése után olvassuk el a súgóoldalt, készítsük el a */etc/rsyncd.conf* fájlt, engedélyezzük az `rsync`-et a */etc/inetd.conf*-ban, majd indítsuk újra az `inet` démont. A beállítófájl igen hasonló a *samba.conf* szerkezetéhez. Szerencsére például beállíthatjuk, hogy egyszerre legfeljebb hányan csatlakozhatnak a szolgáltatáshoz.
A felhasználó oldaláról az `rsync` nagyon hasonlóan működik, mint például az `rsh` vagy az `ssh`. Hogy az újabb protokoll ne okozzon további kavarodást, az rsync-területek címzésénél két kettőspontot használjunk (vagy az `rsync://` formátumot). Ha kettőspontot használunk, a program egyszerű távoli héjmásolást végez. Ha pedig az elérhető megosztások és fájlok listáját akarjuk megkapni, akkor csak a forrást kell megadni (lásd az *5. listát*).
Ha valakit a program felhasználhatósága részletesebben érdekel, melegen ajánlom az idevágó súgóoldalakat (`man rsync` és `man rsyncd.conf`). A leírásokban teljes példákat is találhatunk a biztonsági másolatok és tükrök készítéséhez, illetve fenntartásához.
Röviden igyekeztem összefoglalni, miként érdemes elindulni a Debian-csomagok kezelésének útvesztőjében, és mint ahogy az elején írtam, a paletta sokkal szélesebb a most bemutatott programoknál. Érdemes minél többet megismerni, majd a nekünk legjobban tetszőket kiválasztani.

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

Földrengéshullámok terjedésének modellezése

A Kaliforniai Műszaki Tudományok Intézete
irdatlan méretű, 156 linuxos gépből álló, ikerprocesszoros fürtöt épített.

A hatalmas földrengések a sűrűn lakott területeken végzetesek lehetnek, és az érintett területek gazdasága számára óriási károkat okozhatnak. Az utóbbi nagy földrengések – El Salvadorban 2001. január 13-án 7,7 erősségű, Indiában 2001. január 26-án 7,6 erősségű és Seattle-ben 2001. február 28-án 6,8-as erősségű földmozgás – miatt nagy szükségünk van rá, hogy minél jobban megértsük a földrengések fizikáját, és a földrengések előrejelzésére és a veszélyeztetett épületek, valamint a háttérágazat meghatározására tett kísérletek is sürgetőekké váltak.
A földrengések során tapasztalható erős földmozgásokat a mozgáskiegyenlítődés alakítja, amely három hullámfajtánál jelentkezik: nyomásnál, illetve hanghullámnál, nyíróhullámnál (shear) és a felszíni hullámoknál. A földrengéshullámok terjedését numerikus módszerek használatával háromdimenziós összetett modelleken le lehet vezetni. A földrengéstudományban a nehézségek két fő csoportba sorolhatók: egyrészt a térségi folyamatok utánzása, úgymint a földrengések terjedése a sűrűn lakott, földrengésre hajlamos üledékes medencékben, amilyenre Los Angeles vagy Mexikóváros példa, másrészt a földrengéshullámok terjedése a Föld teljes tömegében.
Az összes földrengés folyamán a földgolyó több pontján mindössze néhány rengésjelző laboratórium rögzíti ezeket a Föld belső szerkezetéről árulkodó rengéshullámokat.



1. kép A Föld hálószemekre bontva

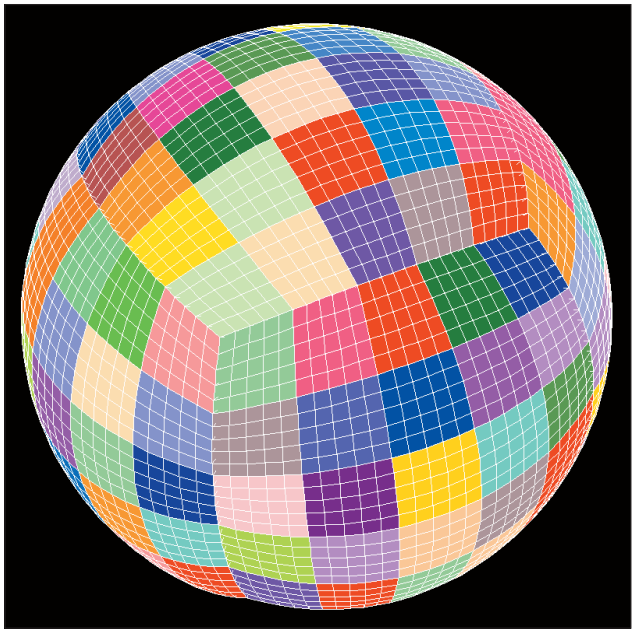
A numerikus módszer

A Kaliforniai Műszaki Tudományok Intézetének (California Institute of Technology) Földrengésjelző Laboratóriumában a kutatók nagy pontosságú numerikus módszert fejlesztettek ki: a spektrumelemes módszert a földrengéshullámok terjedésének háromdimenziós modellezésére. Az eljárás a mérnöki tervezésben már általánosan elterjedt véges elemes módszeren alapul. Minden elem néhány száz pontot tartalmaz, és a földrengéshullámok lefutását helyi hálószemekre képezi le, a számítási eredményeket pedig a szomszédos hálószemeknek közvetíti. A Földön terjedő földrengéshullámok modellezéséhez az egész földgömböt hálószemekre, majd azokat további nagyszámú szeletre bontottuk (*1–2. kép*). Minden egyes szelet nagyszámú elemet tartalmaz, általában több tízezret. A cél az, hogy a számítási műveleteket párhuzamos gépen végezzük, minthogy a feladat mérete nem teszi lehetővé, hogy programunkat osztott táras gépen vagy munkaállomáson futtassuk.
A fent leírtak miatt a módszer fürtözött gépeken tökéletesen kivitelezhető, vagyis olyan elrendezésben, amikor egy-egy gép a hálószemhez tartozó részhalmaz összes elemét kezeli. A számítások eredményének számítógépek közötti közvetítésére a hálózaton belül üzenetküldő módszereket alkalmazunk.
A Linux-rendszeren történő párhuzamos feldolgozás alapgondolata a tudományos társadalomban gyorsan kifejlődött (bővebb adatok a megadott irodalomjegyzékből nyerhetők: ***M. Konchady*** és ***R. A. Sevenich*** cikkei).
A nagy gépfürtök tudományos célú felhasználásának kutatása 1994-ben a NASA (Az Egyesült Államok Űrkutatási Hivatala) Beowulf névre keresztelt vállalkozásával kezdődött (➲ http://www.beowulf.org), amelyet a Caltechnél a Hyglac- és Los Alamosban a Loki-program követett (lásd ***Tom Sterling*** és munkatársainak ***Hogyan építsünk Beowulf rendszert?*** című könyvét és a ➲ http://www.cacr.caltech.edu/resources/naegling honlapot a Világhálón.) Az Egyesült Államokban ***Hans-Peter Bunge*** a Princeton Egyetemről az elsők között használt geofizikai kérdések megoldására a most bemutatásra kerülőhöz hasonló fürtözött rendszert, Európából pedig ***Emmanuel Chaljub*** a párizsi Földtudományi Intézetnél (Institut de Physique du Globe) vezette be az üzenetküldés ötletét a földrengéshullámok tovaterjedésének tanulmányozására. A fürtözést jelenleg az ipar és a tudomány számos területén használják.

A rendszer alapját képező gépi berendezés

Elhatároztuk, hogy a feladat megoldására fürtöt újonnan építünk, szabványos számítógép-alkatrészek felhasználásával. A COTS (Commodity Off The Shelf) mozaikszó a kiskereskedelmi forgalomban kapható alkatrészekből való összeszerelést jelöli, s ezt az elnevezést mi magunk is gyakran használtuk módszerünk leírására. A legerősebb kényszerítő tényező az volt, hogy sok gépre volt szükségünk, ezenkívül rengeteg tárterületet igényeltünk modellünkhöz az általunk használni kívánt háló nagysága miatt. A ki- és bemenet, valamint a gépközi adatcsere számunkra nem volt fontos, mivel a gépek működési idejének legjavát a számítási tevékenység teszi ki, ugyanakkor a gépek közötti adatforgalom a fenti jellemzőhöz mérten állandóan kicsi. Ezen okból kifolyólag a mi hálózatunk

nem lenne képes különösebben kihasználni az olyan nagy teljesítményű hálózatokat, amilyen a Gigabit ethernet- vagy a Myrinet-hálózatok. Ehelyett szabványos, 100 Mb/s sebességű Fast ethernethálózatot építettünk. A szükséges processzorok nagy száma miatt – összesen 312 – ikerprocesszoros alaplapokat használtunk, hogy a számítógépházak számát 156-ra szorítsuk le és egyúttal a helyigényt is csökkentsük. Ez az elrendezés nagy lendületet ad a működéshez, mert a két processzor osztozik a közös memóriasínen, amely a sín telítettségéhez vezet,



2. kép A processzorokhoz rendelt szeletek

de egy füst alatt a gép költségeit is csökkenti, hiszen a két processzorhoz csak egyetlen számítógépház, alaplap, merevlemez stb. szükséges. A gépek keretbe szerelését – lényegében a költségcsökkentés érdekében – kizártuk, és végül a polcokra állított szabványos közepes toronyházak felhasználása mellett döntöttünk – amint az a *3. képen* látható. Ezt az elrendezést időnként a LOBOS mozaikszóval jelölik, utalva rá, hogy a polcokon sok gépet helyeztek el (Lots of Boxes on Shelves). A polcrendszert és a 156 gépes telepet olyan gépterembe helyezték, amelyben már nagy teljesítményű szellőztető berendezés működött.

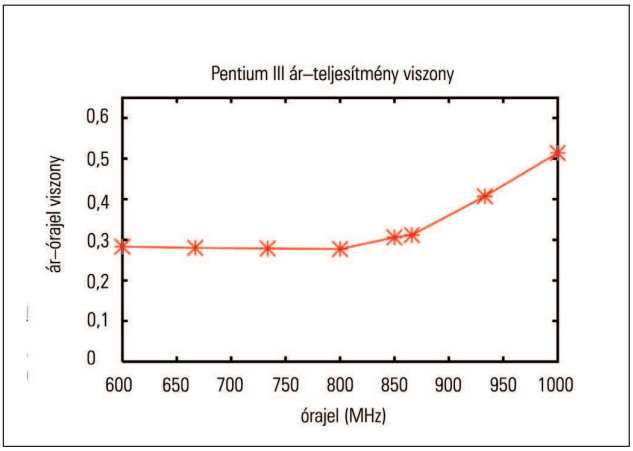
Úgy hírlik, az Athlon gyorsabb a lebegőpontos számításoknál, amely a legtöbb tudományos számításban a fő művelettípus, beleértve a miénket is. A telep építésének idején semmilyen ikerprocesszoros, az Athlon központi egység fogadására alkalmas alaplap sem volt kapható. Mint már fentebb említettük, a közönséges csomópontok építése növelte volna a fürtépítés költségeit, éppen emiatt a Pentium III-ra esett a választásunk. A gépek összeszerelésekor nagy a kísértés, hogy a legújabb műszaki megoldásokat használjuk fel, viszont az új processzorok a fél éve piacon levő eszközöknél jóval drágábbak, és csak csekély teljesítménybeli növekedést nyújtanak. A 3–6 hónapos processzorok ár, illetve teljesítményaránya a legkedvezőbb. Amikor 2000 nyarán összeszereltük a rendszert, a gépekbe 733 MHz-es órajelű processzorokat építettünk.

A *4. kép* a Pentium III processzor ár-teljesítmény arányát mutatja. A feltüntetett árak az egyesült államokbeli kiskereskedőknél jellemző árak. Mint látható, a régi processzorok olcsók, ezzel szemben viszonylag lassúk. Az új processzorok gyorsabbak, viszont sokkalta drágábbak. A megfelelő ár-teljesítmény arányt a két véglet között kellett megtalálni. Úgy döntöttünk, hogy az alaplapokon minden tárfoglalatot kitöltünk, vagyis gépenként 1 GB memóriát zsúfoltunk a foglalatokba, amely a teljes fürtre nézve 156 GB tárterületet képezett. Minden gépre csomópontként vagy inkább számítási csomópontként (node) hivatkoztunk. Fontos megjegyezni, hogy a memóriaköltségek a teljes telep költségének több mint felét teszik ki. A többi számítógép-alkatrész már szinte teljesen átlagos: minden gép 20 GB-os merevlemezzel, Fast ethernet hálózati kártyával és 1 MB-os PCI-sínes videokártyával rendelkezik, utóbbi ahhoz kell, hogy a gép az operációs rendszert helyesen be tudja tölteni, és amikor éppen szükséges, a gép működését figyelemmel lehessen kísérni. Jó minőségű, középmagas, golyós tengelycsapágyas házventilátorral ellátott toronyházakat használtunk, mert a fürt mechanikus alkatrészei – mint például a tápegységek és a ventilátorok – hajlamosak a leginkább a meghibásodásra. Érdemes figyelmet fordítani a fürt hihetetlenül nagy lemezkapacitására, amely 156×20 GB, azaz 3×120 GB = 3 TB (terabájt). A fürtépítés költségeinek további csökkentése és a felhasznált alkatrészek minőségének teljes körű ellenőrzése érdekében az előreszerelt számítógépek megrendelése helyett az alkatrészek különböző kereskedőktől való beszerzése és a gépek saját összeszerelése mellett döntöttünk. A teljes fürt összeállítása három ember hozzávetőlegesen egyheti munkáját vette igénybe. Az egyik gép, a felhasználói felület szerepét betöltő vezetőgép a fürtön belüli különleges szereppel bírt: ez tartalmazta az egyes felhasználók saját állományrendszerét, a SCSI-meghajtókat, az önműködő befűző (automounter) által a többi csomóponton elérhetővé tett NFS-t, a fordítóprogramokat, az üzenetküldő könyvtárakat és sok minden mást is. A modellezési folyamatokat erről a gépről indítottuk, majd innen követtük nyomon. A vezetőgép a csomóponti gépeken karbantartási céllal naplót vezet. A csomópontokat a Cisco cég által gyártott 192-kapus Catalyst 4006-os hálózati kapcsolóval kötöttük össze, amelynek a hátlapi sávszélessége 24 Gb/s.

A rendszer beállítása és a programkódfejlesztés

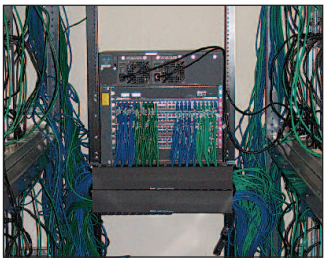
A fürtön belüli összes gép Red Hat 6.2-es Linux-változattal üzemel. A Linux tökéletesen megfelel az alkalmazásokkal szemben támasztott elvárásainknak: nagyfokú rendelkezésre állást követelünk meg, mert a gépet tudósok használják, és számukra az a legfontosabb, hogy úgy végezhessék a munkájukat, hogy ne kelljen a gépek összeomlása miatt aggódniuk. Kilenc hónapja, vagyis a rendszer telepítése óta egyetlen rendszerösszeomlásunk sem volt. Az operációs rendszert a legnagyobb teljesítmény eléréséhez a berendezés jellemzőinek megfelelően kell hangolni; a nyílt forrás alapelveit követve a rendszermag újrafordítását el tudtuk végezni – a mi gépünkre jellemző néhány alapvető beállítással kiegészítve. A közelmúltban telepítettük a 2.4.1-es rendszermagot, amely a kétprocesszoros SMP-gépeket sokkal jobban támogatja, mint a 2.2. rendszermag. A gép csodálatosan működik: a 2.2-es rendszermagról a 2.4.-re való átállás után a felhasználói programok központi-egység-lekötési ideje 25 százalékkal csökkent. Hálózati szempontból a fürt az 192.168.1.x címre van befűzve. Biztonsági okok miatt a telep nincs a külvilággal összekötve; a teljes üzenetfeldolgozás és elemzés a vezérlőgépen, helyben történik. A cron táblában beállítjuk, hogy az `rdate` paranccsal a csomóponti gépek óráit napi egyszeri alkalommal a vezérlőgép órájához igazítsuk. A számítógéptelep használatáért a legsúlyosabb árat a meglévő soros programkód üzenetküldési elven alapuló

3. kép
A 156 ikerprocesszoros fürt. A gépek 100 Mb/s sebességű Fast ethernethálózatra vannak fűzve (zöld és kék kábelek). A háttérben a 192-kapus Cisco hálózati kapcsoló látható. A polcrendszer elérte a 2 m 64 cm-es magasságot



2. ábra Az ár-teljesítmény arány a Pentium III processzorra vonatkozóan

párhuzamos kóddá alakításáért kellett fizetnünk. Esetünkben ez az ár tetemes volt, hiszen csapatunk kutatókból áll, nem pedig hivatásos programozókból. Ez a helyzet számunkra azt jelentette, hogy a több tízezer sornyi soros programkód átírására rá kellett szánnunk néhány hónapot. A fő nehézséget



4. kép
Cisco 4006 típusú hálózati kapcsoló

az üzenetközvetítő elv jelentette, vagyis hogy a vezérlőgépnek avagy a főgépnek a munkát szükségszerűen egyenlő mértékben kell elosztania a többi elem, vagyis számítási pont között. Minthogy a csomópontoknak a munkát minden időlépésben össze kell hangolniuk, egy-egy gépnek a számítási műveleteket nagyjából azonos idő alatt kell befejeznie. Amennyiben a terheléselosztás egyenlőtlen, azaz rossz a munkaelosztás, a gépek a leglassabb csomóponthoz fognak igazodni, ez pedig a legkedvezőtlenebb forgatókönyv szerinti működéshez vezet.

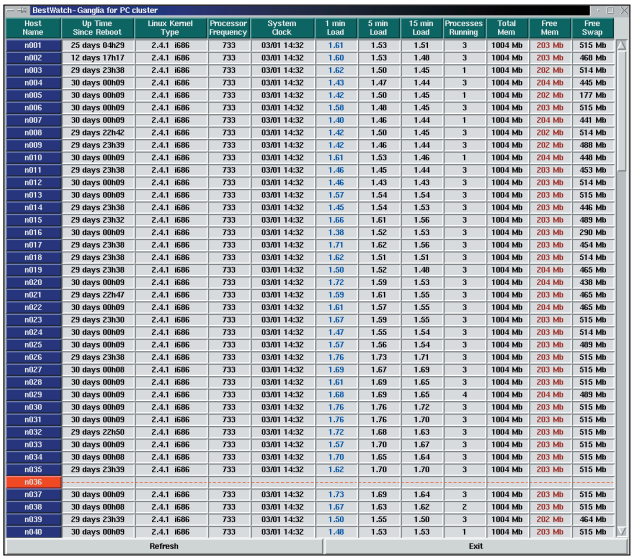
Akadályt jelenthet az az eset, amely ellehetetleníti a gépek közötti adatcserét. Jó példa erre, amikor A gép adatot vár B géptől, de ugyanekkor B gép is adatot vár A-tól. Az ilyen holtpontok elkerülésére használandó a mester, illetve szolga programozási módszer.

Az üzenettovábbításra a beszédes nevű Message Passing Interface (MPI) programkönyvtárat használjuk, nevezetesen az Argonne National Laboratoryban kifejlesztett MPICH megvalósítást telepítettük (lásd ***W. Gropp*** és munkatársainak 1996-ban megjelent cikkét, amely a ➲ http://www-unix.mcs.anl.gov/mpi/mpich címen olvasható). Ez a csomag Linux-rendszeren rendkívül megbízhatónak bizonyult. Az MPI a párhuzamosprogramozási-közösségen belül egyre inkább szabvánnyá válik. Az MPI számos szolgáltatása a korábbi, párhuzamos feldolgozást megvalósító PVM (Paralell Virtual Machine) programkönyvtárhoz hasonlít, amelyet 1998-ban ***R. A. Sevenich*** mutatott be a Linux Journalban.

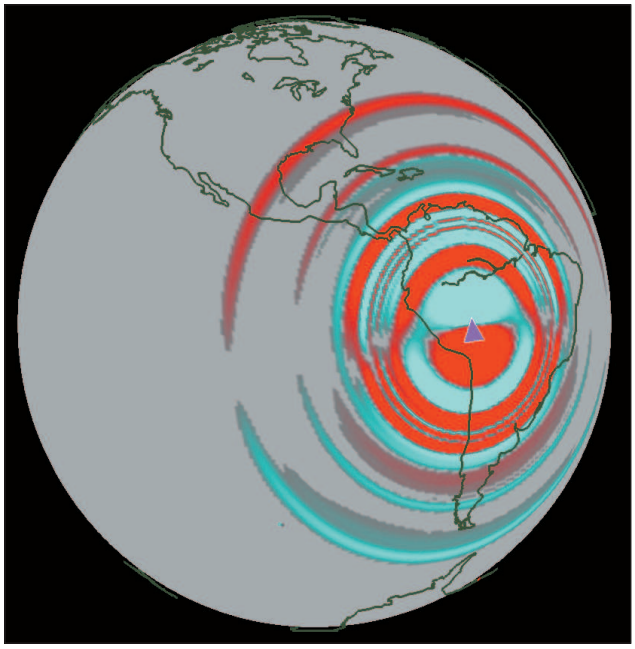
Nehézséget jelentett feladatunk megoldásánál az örökségül ránk maradt régi programkód, amellyel mindenképp foglalkoznunk kellett. Sok korszerű programkód alapul olyan programkönyvtárakon, amelyekben még 1970-es, 1980-as évekbeli régi programkódok lelhetők fel. Szinte mindegyik programkódkönyvtár Fortran77-ben íródott, tehát az akkori idők közkedvelt nyelvén; a C nyelv pedig az idő tájt még nem terjedt el. Olyan döntést hoztunk, hogy a több mint 40 000 sornyi forráskódot nem írjuk át C nyelvre, inkább a fordítóprogramot frissítjük Fortran77-ről Fortran90-re. Az új változatban már létezik dinamikus tárfoglalás, mutatók stb., ráadásul visszafelé együttműködik a Fortran77-tel. Készítettünk egy Perl-héjprogramot, hogy a kódátalakítást önműködővé tegyük: néhány részletet kézzel kiegészítve, a tárfoglalást pedig statikusról dinamikusra változtattuk. Ismereteink szerint ingyenes Fortran90-es program Linux-rendszerre sajnos még nem érhető el. A GNU g77 és f2c csomagok szerényen csak a Fortran77 fordítóprogramot támogatták. Így jobb híján kereskedelmi terméket, a pgf90-et kellett megvásárolnunk a Portland Grouptól (➲ http://www.pgroup.com). Ez lett számítógéptelepünk egyetlen nem nyílt forrású eleme.

A géptelepek korlátja a rendszerfelügyelet és -karbantartás. A gépek százszámra való használatával együtt növekszik a gép- és programhiba lehetősége. Géphiba felmerülése esetén a gépekre nézve kifejezetten előnyös, hogy alkatrészeik csereszabatosak, órákon belül beszerezhetők és kicserélhetők. Ezért a karbantartással kapcsolatos költségek alacsonyak azokhoz a karbantartási szerződésekhez képest, amelyeket a kutatók a szupergépek üzemben tartására kötnek.

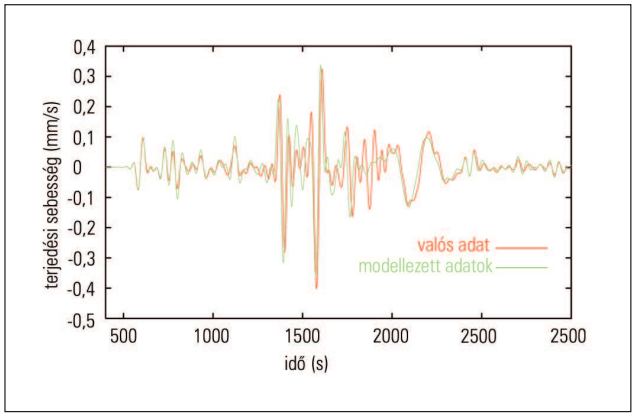
Sokkal izgatóbb kérdés a programkarbantartás: egy 156 gépes



5. kép A programrendszer működésének megfigyelése a Ganglia-programmal



6. kép A 8,2 magnitudójú bolíviai földmozgás rengéshullámai



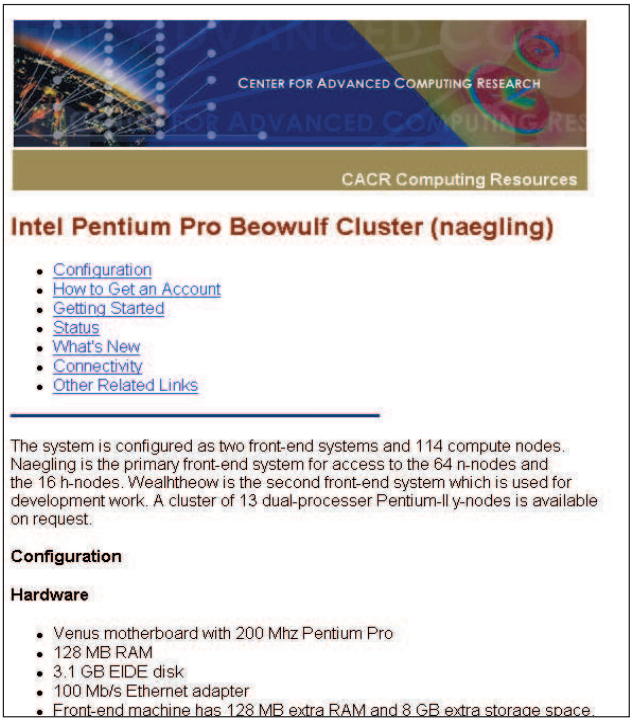
2. ábra A függőleges sebesség – ahogyan Pasadenában érzékelték

➲ http://www.pgroup.com

rendszernél hogyan ellenőrizhető, hogy minden gép megfelelően dolgozik-e? Miként telepíthetünk új programokat? Hogyan kísérhetjük figyelemmel egy futó munka végrehajtását? A fürt telepítésekor héjprogramot készítettünk, amely `rsh` parancsok küldésével adatokat gyűjtött az egyes gépekről. Azóta az olyan egyetemek, mint például a Berkeley, valamint vállalatok, mint például a VA Linux, hatékony programcsomagokat fejlesztettek ki a telep működésének megfigyelésére, és végül ezeket nyílt forrású programokká nyilvánították. A VA Linux SystemImager nevű gépklónozó csomagját használjuk a programfrissítések elvégzésére. Ezzel a csomaggal csak a vezetőgép frissítését kell kézileg elvégeznünk. Ezt követően a csomag a `rsync` és `tftp` parancsok alkalmazásával a változásokat a további 155 gépre átmásolja, vagy ha úgy tetszik, klónozza. A telep működésének és a futó programok állapotának figyelésére ***Matt Massie***, a Berkeley Egyetem munkatársa által készített Ganglia-csomagot használjuk. Ez egy olyan gyors és könnyen kezelhető program, amely a csomóponti gépekről a vezérgépre történő adatküldéshez rendszerdémonokat használ. A kapott adatokat a fő gép összegyűjti és megjeleníti. A *6. képen* a Ganglia-csomaghoz kapcsolódó Tcl/Tk-felületet mutatjuk be. Az általunk használt grafikus felület egy további nyílt forrású csomagon, a ***Jacek Radajewski*** által készített bWatchon alapszik (➲ http://www.sci.usq.edu.au/staff/jacek/bWatch). A szabványos `rsh` parancsok helyett a Ganglia programot használjuk, a forráskódot ennek megfelelően módosítottuk. A VA Linux is megjelentette VACM (VA Cluster Management) elnevezésű fürtkezelő csomagját, ezt azonban eddig még nem telepítettük.

A bolíviai földrengések és földmozgások

1994. június 9-én irtózatos, a nyílt végű Richter-skála szerinti 8,2 erősségű rengés pattant ki a földfelszín alatti 641 km-es, azaz 400 mérföldes mélységben. A legtöbb földrengés a föld-



➲ http://www.cacr.caltech.edu/resources/naegling/

felszínhez sokkal közelebb zajlik le, rendszerint 30 km-nél sekélyebb tartományban jön létre. Ez utóbbi bolíviai rengés a valaha feljegyzett legnagyobb mélységi rengések egyike volt. Szokatlan jellemzői miatt e földrengést a földrengéstudományi szakmai közösségen belül máris nagyon sokan választották írásuk témájául. Ezekután számítógéptelepünkön mi is megpróbáltuk utánozni az eseményt.

A 7. *kép* egy ilyen földlökést mutat be – a föld elmozdulását adott helyen, amely a földrengés által keltett hullámok miatt létrejött elmozdulást ábrázolja. A bolíviai rengés epicentrumát a képen lila háromszög jelöli. A földrengések a felszín alatt és mentén haladnak. Látható például az is, amint az Egyesült Államok területén szétterjednek. Az állandó elmozdulás Bolívia körül a Föld felszínén is látható, északon egészen az Amazonasig terjed. Ezt a jelenséget statikus eltolódásnak nevezik és több bolíviai földrengésjelző állomás is rögzített effélét. A földrengés olyan erősségű volt, hogy a földet állandó jelleggel több milliméteres kitéréssel mozgatta meg. A függőleges elmozdulás déli irányban elérte a 7 mm-t. Mindezt programkódunkkal teljes pontossággal képesek voltunk megismételni. Ez annak köszönhető, hogy a rengéshullámok a Föld teljes tömegében szétterjednek, a bolíviai földmozgásokat a többi országbeli földrengésjelző állomások is érzékelni tudták.

A 2. *ábra* a pasadenai (Kalifornia) állomás által rögzített adatokat mutatja, mellette pedig a modellező módszerünk létrehozta idevonatkozó adatokat. Ismételten megállapíthatjuk, hogy az egyezés kielégítő. Ez a modellezési folyamat minden egyes időlépésben egy 500 millió ismeretlenes egyenletrendszer – az ismeretleneket a rendszer szabadságfokaként szokás emlegetni – megoldását követelte meg. A rengéshullámok terjedésének modellezéséhez a fürt 48 órás működésére volt szükség – a csomópontok felének felhasználása mellett (150 processzor). Talán nem is szükséges megemlíteni, hogy a most bemutatott kutatásunk mekkora teljesítményt és megbízhatóságot merített a Linuxból és a nyílt forrás filozófiájából. Nagyméretű géptelep használatával tudtuk utánozni a pusztító földrengések nyomán fellépő rengéshullámokat, és modellezésükre példa nélkül álló megoldást találtunk.

Köszönetnyilvánítások

Luis Rivera a vállalkozáshoz felbecsülhetetlenül fontos adatokkal és a segítségével járult hozzá. Köszönetet mondunk ***Jan Lindheim***-nek, ***Tom Sterling***-nek, ***Chip Coldwell***-nak, ***Ken Ou***-nak, ***Jay Nickpour***-nak és ***Genevieve Moguilny***-nek a fürt szerkezetét érintő viták során tett építő megjegyzéseikért. ***Matt Massie*** több szolgáltatással egészítette ki egyébként is kitűnő Ganglia-csomagját, hogy gépfürtünk minél több jellemzőjének megfigyelését tegye lehetővé a számunkra. ***Rusty Lusk*** pedig hasznos ötleteket adott a nagy fürtön való MPICH-működtetést illetően.

Dr. Dimitri Komatitsch
a Kaliforniai Műszaki Tudományok Intézete Föld- és Bolygótudományi Osztályának vezető kutatója. Érdeklődési területei az alkalmazott matematika, a numerikus analízis és a számítógéptudomány alkalmazása geofizikai és szeizmológiai feladatok megoldásánál.

Dr. Jeroen Tromp
a Kaliforniai Műszaki Tudományok Intézete Föld- és Bolygótudományi Osztályának professzora. Érdeklődési területe az elméleti, különösen az egész Földre kiterjedő szeizmológia. Az utóbbi időkben figyelmét a földrengéshullámok terjedésének numerikus modellezésére összpontosította.

Irodalomjegyzék

T. J. R. Hughes: The Finite Element Method, (A véges elemes módszer), Prentice-Hall, 1987.

W. Gropp, E. Lusk, N. Doss és A. Skjellum: A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard (Az MPI üzenetküldő rendszer nagy rendelkezésre állású, hordozható megvalósítása), Parallel Computing, 22. évf., 1996. szeptember.

T. L. Sterling, J. Salmon, D. J. Becker és D. F. Savarese: How to Build a Beowulf (Hogyan építsünk Beowulf rendszert?), MIT Press, 1999.

D. Komatitsch és J. Tromp: Introduction to the Spectral-Element Method for Three-Dimensional Seismic Wave Propagation (Bevezetés a spektrumelem-módszer alkalmazásába a háromdimenziós földrengéshullám-terjedés elemzésnél),Geophysical Journal International, 139 évf., 1999.

M. Konchady: Parallel Computing Using Linux (Párhuzamos programozás Linux-rendszerben), Linux Journal, 45. szám, 1998. január.

by R. A. Sevenich : Parallel Processing Using PVM, (Párhuzamos feldolgozás PVM alkalmazásával), Linux Journal, 45. szám 1998. január

M. Lucas: Remote Sensing with Linux (Távérzékelés Linux-rendszerrel), Linux Journal, 82. szám, 2001. február

Válasszunk Webmailt!

Minden rendszergazda életében eljön az a pillanat, amikor a felhasználók úgy döntenek: Webmailre van szükségünk! De melyiket válasszuk?

Mire jutottunk az elmúlt lapszámokban? Be tudunk állítani egy levelezőkiszolgálót, amely fogadni és küldeni képes a leveleket, valamint az IMAP- vagy a POP3-kiszolgálón keresztül a felhasználóink is hozzáférnek (lásd Linuxvilág 9. szám 53. oldal). Majd a felhasználók vagy a vezetőség úgy dönt, hogy szükség van egy Webmailre. Melyek az általános követelmények egy Webmail programmal szemben? Legyen könnyen kezelhető és többnyelvű – azaz magyarul is értsen, felhasználónként legyen testreszabható, tudjon valamilyen címtárszolgáltatást kezelni, mondjuk az LDAP-ot. Most lássuk az általam kiszemelt versenyzőket!

Squirrelmail

Személyes kedvencem a Squirrelmail, ha bővíthetőségről, a szolgáltatások számáról és testreszabhatóságról van szó. A sebességével azonban már nem vagyok annyira megelégedve, és valószínűleg az írók sem, ugyanis a levelezőlistán állandó téma, hogy a `core` (mag) részt teljesen újraírják, így téve hatékonyabbá a sebességnövelést. Mégis miért ez lett a kedvencem? Mert olyannak találták ki, hogy az alapcsomag a lehető legkevesebb lehetőséget biztosítsa: levélírás és olvasás, valamint egyéni beállítások. A kiegészítőket bővítmények (plug-in) formájában adhatjuk hozzá a programhoz, ezek fogják végül teljessé tenni. Ezek között olyan bővítményeket is találunk, melyek leveleinket a távoli POP3-fiókról töltik le, valamint olyanok is, amellyel címlistánkat emelhetjük ki vagy tölthetjük be az Outlook Express levelezőügyfélbe. Ezenkívül rengeteg olyan hasznos szolgáltatással is bír, amellyel a vetélytárs levelezők nem. 100–150 felhasználóig tökéletes megoldás, ha a PHP-t kordában tartjuk, mert a programban néha bizony találnak biztonsági hibát. Egy jól beállított kiszolgálóval ez azonban nem jelenthet gondot, és a kód auditálása folyamatos, míg a többi programról ez általában nem mondható el. Előnyei közé sorolom, hogy IMAP-kiszolgálót igényel. Miért előny ez? Mert nem szeretem, ha egy alkalmazás közvetlenül nyúl a fájlrendszerhez. Azonkívül, ha egyszer úgy döntök, hogy az alkalmazást lecserélem, minden további izgalom nélkül megtehetem, mert a leveleket az IMAP válogatta és tárolta, nem pedig valamilyen egyedi megoldás szerint zajlott, azaz a rendszer egésze így sokkal jobban megőrzi együttműködő képességét. A Squirrelmail itt is kiemelkedő teljesítményt nyújt, mert a legelterjedtebb IMAP-kiszolgálókhoz már „gyári" beállításokkal rendelkezik. Másik előnye, hogy minden olyan webkiszolgálón futtatni tudjuk, amelyik PHP-futtatási modullal rendelkezik.

➲ http://www.squirrelmail.org

Imho

Valamikor emellett a Webmail mellett tettem volna le voksomat, de megváltozott az álláspontom. Még ma is kitűnő levelezőügyfél, de csak Roxen webkiszolgáló, illetve Caudium alatt hajlandó futni (a Caudium alkotói az új Roxen fejlesztési irányvonalával nem értenek egyet, és a Roxen a 1.3-t a saját elképzeléseik szerint fejlesztik tovább). A megbízhatósága és a sebessége kitűnő, azaz gyorsabb, mint a Squirrelmail. A hordozhatóság hiánya miatt került le a kedvenceim listájáról. Vannak azonban előnyei is: jól kezeli a különböző nyelveket, könnyen testreszabható, és a külcsín is könnyen megváltoztatható. Az IMAP-kiszolgálót sem terheli le annyira, nem nyit minden kéréshez új kapcsolatot, ami akkor különösen kellemetlen, ha egy felhasználó sok levéllel rendelkezik.

➲ http://www.lysator.liu.se/~stewa/IMHO/

Camas

Fentebb már említett levelező, amely az előző ügyféltől nem sokban különbözik – bár a Camas-hívők biztosan tiltakoznának a kijelentésem ellen. Leginkább frissítések találhatók benne. Nagy tudású levelező, de ismét csak egyetlen kiszolgálón fut, ezért nem használom.

➲ http://camas.caudium.net

NeoMail

Amikor ezt programot ajánlom, ellentmondok önmagamnak. Ez ugyanis Perl nyelven írt levelező, ami a felhasználó levelesládáját közvetlenül éri el, POP3- vagy IMAP-kiszolgáló nélkül. A fejlesztése sajnos befejeződött, mert a programíróval a cége egy olyan papírt íratott alá, amelynek alapján csak az adott cégnek fejleszthet. Mondhatnánk, hogy ennyi volt, de egykor ez volt a kedvenc levelezőügyfelem. Remélem, valaki folytatja majd a fejlesztését.

➲ http://sourceforge.net/

Összegzés

A fentebb felsorolt webes levelezőügyfelek rendkívül hasznos alkalmazások: akad olyan cég, ahol a telepítés után a felhasználók már nem is akarnak mást használni. Természetesen az itt felsoroltakon kívül rendkívül sok program létezik még, de mint említettem, ezek a személyes kedvenceim, és a velük szerzett tapasztalataim alapján választottam ki őket. Választás előtt mindig vizsgáljuk meg, hogy az adott környezetnek mi felel meg a leginkább. Hiába bíráltam például az Imhót, amennyiben a kiszolgáló, amire Webmailt telepítenénk, Roxen webkiszolgálóval rendelkezik, döntsünk mellette, hiszen sokkal megbízhatóbban fog vele működni, mint a Squirrelmail a PHP-modullal.
Következő írásomban a Squirrelmail telepítését követjük végig Apache kiszolgáló alatt, valamint néhány érdekességgel kezdhetünk meg a levelezés világából.

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.

PoV-Ray ismeretek (8. rész)

Sorozatunk utolsó részében a függvények ismeretéből fakadóan már az animációk elkészítésének rejtelmeibe pillanthatunk bele.

Tekintettel arra, hogy ezt a programot igazából a sugárkövetéses képszámítás használatára készítették, nem találkozhatunk benne olyan megoldásokkal, melyeket az általánosan használt modellezőkben megszokhattunk. Itt nincs mód hierarchikus tárgyfelépítésre, ha tehát valaki például inverz kinematikát szeretne használni, és nem riad vissza a nehézségektől sem, még akkor is sokáig eltart, amíg az animáció elkészültekor minden tárgy a helyére kerül. Ennyi bevezetés után lássunk végre munkához!
Egyszerű animációt fogunk elkészíteni, amelynek egy képét a sorozat előző részében már felvázoltam. Most a malom lapátkerekét, a vizet és a felhőket fogjuk mozgásra bírni, miután példákat mutatok az előzőekben alkalmazott dinamikus tárgylétrehozásra. A jelenet a következő tárgyakra bontható:

- a malom épülete,
- mellette a malomkő,
- a táj,
- az égbolt,
- a víz – felette pára,
- a kerítés.

Természetesen a malom épülete nem csupán egy objektumból áll, hanem egy nagyobb `box` objektumból kivontam egy kisebbet, majd ebből az ajtó és az ablakok helyén szintén kivonással készítettem el a nyílásokat. Az ablakkeretek külön tárgyak, ezeket végül az épület alsó részével egyesítettem. A tetőszerkezetet háromszögalapú `prism` objektumokból képeztem, majd a homlokzatot egy külön `prism` objektum alkotja. Itt el lehetett volna hagyni a homlokzatot alkotó harmadik tárgyat, de a szemléltetés kedvéért megtartottam. Így a tető létrehozása után például rácsos homlokzatot is készíthetünk, amely jobban szellőzik.
Most részletesebben kitérek az ajtó elkészítésének módjára, bár nem olyan nagy ördöngösség. Az ajtóléceket egy ciklusutasítás (`#while`) segítségével egy keskeny téglatest eltolásával hozom létre. A ciklusszámláló változásakor a számlálót használom a pillanatnyi eltolás kiszámításához. Itt minden ajtólécet a léc szélességével és egy tetszőleges értékkel kell eltolni, amely az ajtólécek közötti rést határozza meg. Az ajtóléceket egymás mellé helyező kódrészletet az *1. listán* láthatjuk.
Hasonló módon készült a kerítés is, ez esetben viszont a közbeeső vastagabb oszlopok létrehozására a maradékos osztás műveletét is felhasználtam. Amikor a ciklusváltozó maradék nélkül osztható öttel, az oszlopot nem kell annyira lekicsinyíteni, így a kerítésben oszlopok és keskenyebb tartólécek is lesznek. A kerítés két részből áll, mindkettő ezzel a módszerrel készült, de a ház előtt lévő rész rövidebb, kevesebb tartóoszlopból áll, és a létrehozás után még el kellett fordítani, hogy a másik lécsorra merőlegesen álljon.
A malom lapátkerekét szintén több részből készítettem, CSG-műveletek felhasználásával. A lapátokat tartó abroncs két henger különbségeként alakult ki. A kerékagyat és a küllőket

1. lista

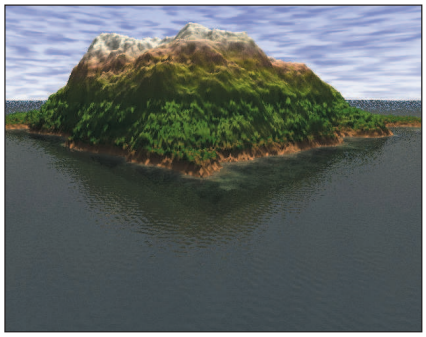
```
// itt jon az ajto
#declare ajtolec = 0
#declare Ajto=union {
  #while ( ajtolec < 10 )
    box {
      <0, 0.07, 0>
      <0.07, 1.48, -0.05>
      translate <-0.08*ajtolec, 0, 0>
    }
    #declare ajtolec = ajtolec + 1
  #end
}
```

2. lista

```
// Max. ennyire lehet nyitva az ajt
#declare max_ajto=40

// Ennyire van kinyilva az ajto
#if (clock<0.5)
  #declare ajto_nyilasszoge =
           max_ajto*clock
#else
  #declare ajto_nyilasszoge =
           max_ajto-(max_ajto*clock)
#end

// Az ajt  Øs forgatÆsa
object {
  Ajto
  material {
    AjtoMat
  }
  rotate -y*ajto_nyilasszoge
  translate -0.16*x
  translate -0.95*z
}
```

egyszerű hengerek alkotják, a küllőket oldalirányból egy kicsit összenyomtam. E helyütt a lapátokat és a küllőket szintén ciklusban hoztam létre. A lapátoknál még ki kellett számítani az egyes lapátok új helyzetét és az elfordulásukat. Ezeket a koordinátákat szinusz és koszinusz függvényekkel lehet kiszámítani. Az x koordinátát a következő képlet alapján: x=R*cos(rad(szög))+dx, míg az y koordinátát az y=R*sin(rad(szög))+dy képlet segítségével. Az `R` annak a körnek a sugara, amelynek mentén a tárgyat el szeretnénk

helyezni, míg a dx és a dy értékek e kör a középpontját határozzák meg. A képletekben szereplő sz g természetesen az elforgatás szöge fokban meghatározva. Az egyes darabok elkészítése után a részeket egyesítettem, így a további eltolások és forgatások már az egész kerékre vonatkoznak.
A malomkő esetében a legnehezebb feladat a pontos elhelyezés és dőlés beállítása volt – maga a tárgy két hengerből készült, szintén kivonással. Az elhelyezését kísérleti úton oldottam meg, hiszen így tűnt a legegyszerűbbnek.



A hangyák szigete

Techno-malom

A táj létrehozásakor a jól ismert Gimp nevű képfeldolgozó és festőprogramot hívtam segítségül, és két képet készítettem vele. Az egyik egy szürkeárnyalatos kép, melyet először középszürke (R: 128; G: 128; B:128) színűre festettem be, utána pedig sötétebb színnel a mélyebb területeket (például a patakmeder és a tó (ez az animációban nem látszik)) festettem rá, majd világosabb színekkel kialakítottam a dombokat és hegyeket. A táj színeinek és mintázatának meghatározásához később ezt a képet használtam fel. Első lépésként 256 színűvé alakítottam egy zöldárnyalatokból álló paletta használatával, majd visszaalakítottam 24-bites színmélységűvé. Így már rendelkezésemre állt egy zöld, sárga és szürkeárnyalatokból álló tájkép felülnézete. Erre külön rétegekben ráfestettem a patakokat, majd az egyéb árnyaló színeket, hogy a kép ne legyen annyira egyszínű. A harmadik rétegen festettem meg az utat, ennek elkészítése vette igénybe a legtöbb időt. Minden próbálkozás után kiszámítottam egy képet, és ha a ház nem volt teljesen az út leágazó részének a végén, az utat arrébb kellett festenem, mert a patak miatt az épületet nem lehetett elmozdítani a helyéről. Végül a részletek jobb láthatóságáért a képet a kétszeresére nagyítottam és újabb változatos mintákat festettem a fűre.
Az előbbi tájkészítési mód bonyolultnak tűnhet, ezért el kell mondanom, hogy csak azért volt szükség a kézi munkára, hogy a tárgyakat pontosan el tudjam helyezni. Mivel nagyméretű tárgyakról van szó, amelyeket nem lehet pontatlanul elhelyezni, teljesen sík felületet kellett létrehoznom, ahol a síkalapú épületet el tudtam helyezni. A CD-mellékleten található néhány segédprogram, köztük a terraform, amely kifejezetten felülnézeti domborzati képek készítésére használható. Korábbi DOS-os és más operációs rendszer alatt futó megfelelője VistaPro névre hallgat, bár a CD-n található program tudásában kissé elmarad a nem linuxos változattól.
A program a létrehozott domborzatból PoV-Ray forrást is képes készíteni, amit a későbbiek folyamán saját céljainkra használhatunk fel. A fenti képen ezzel a programmal készítettem egy tájat, amit szintén a PoV-Ray számított ki.
A CD-mellékleten (28. CD Magazin/Pov-Ray) találhatunk még egy átalakító programot, amellyel különféle 3D-formátumokat tudunk konvertálni, többek között *.pov* forrásállománnyá, valamint egy kifejezetten a PoV-Rayhez készülő modellező (TrueVision) programot is, mely ugyan még fejlesztésének kezdeti szakaszában van, de bonyolultabb jeleneteink megalapozásához így is alkalmazható. A TrueVisionnal készítettem el a példában szereplő jelenet anyagait, majd a beállításokat a hagyományos módon (szövegszerkesztővel) finomítottam.
Mindezek után rátérhetünk a jelenet mozgó részeinek meghatározására. A PoV-Rayben egy animáció elkészítésekor a program a bemeneti forrásállományt újra és újra beolvassa és feldolgozza, majd a memóriában létrehozza a tárgyakat, az anyagokat, a fényforrásokat stb. Ez azt jelenti, hogyha valamilyen változást szeretnénk előállítani, akkor a két képkocka előállítása közben kell a forrásban módosításokat végezni. Például megtehetnénk ezt úgy is, hogy egy külső program minden képkockához előállít egy forrásállományt, amelyek csak néhány sorban és vektorösszetevőben különböznek egymástól, majd a PoV-Ray segítségével kiszámítjuk őket. Ez nem túl kényelmes megoldás, ezért a programozók egy belső változót bocsátanak a rendelkezésünkre, a clock-ot, melynek értékét a PoV minden képkocka kiszámítása előtt növelni fogja. Hogy mennyivel fog változni képről képre, a következők alapján számíthatjuk ki: a PoV-Ray számára megadhatunk egy kezdőképszámot, egy másik értékben pedig a képek legnagyobb számát. A képkockák kezdeti indexét a Initial_Frame, az utolsó kocka számát a Final_Frame értékkel határozzuk meg a beállításokat tartalmazó fájlban. Ugyanígy határozzuk meg az időszámláló kezdő- és végértékét a Initial_Clock és Final_Clock kulcsszavakkal. A clock minden képkocka között a következő képlettel meghatározható értékkel fog változni:

```
Delta_clock=(Final_Clock-Initial_Clock)/
➥(Final_Frame-Initial_Frame)
```

Ezt a változást használjuk fel a jelenetleíró állományban, hiszen mindig azonos értékkel fog módosulni, és a segítségével tetszőleges tárgyat is nagyon egyszerűen elforgathatunk. Erre az elforgatásra mutat egyszerű példát az alábbi lista:

```
cylinder {
    0*y,
    3*y,
    0.2
    rotate z*clock*360
}
```

Itt jegyezném meg, hogy a clock-változó kezdő- és végértékét célszerű 0-ra és 1-re állítani, mert így könnyebben számolhatunk vele. Erre a beállításra is láthatunk példát a CD-mellékleten (28. CD Magazin/Pov-Ray), de az érthetőség kedvéért itt is megemlítem:

```
Input_file_name=malom.pov
Initial_Frame=0
```

```
Final_Frame=120
Initial_Clock=0
Final_Clock=1
```

A fentiek szemléltetésére lássunk egy bonyolultabb példát is (a 2. *lista* tartalmazza), melyet a cikk fő témáját adó jelenetből vettem.
Itt a feltételes utasítás-végrehajtást a mozgást szolgáló `clock`-változóval együtt alkalmaztam, így az animáció első felében az ajtó negatív irányba fordul el, a második felében pedig az ellenkező irányba, vagyis kinyílik és becsukódik. Az `Ajto`-t mint tárgyat egymás mellé helyezett lécek egyesítésével már korábban meghatároztam.
Ezekkel az ismeretekkel bárki nyugodtan elindulhat a PoV-Rayjel készített animációk világának felfedezésére. Az alábbiakban egy képet mutatok be, ehhez hasonló látható az előző oldalakon is. A különbség csak annyi, hogy ez esetben már nem egy ember görbe vonalai alkotják a ház körvonalait, hanem gépies pontosságú egyenesek. A számítógéppel előállított kép számos esetben kisebb hatást gyakorol az emberi lélekre, mint egy hagyományos módon festett kép, hiszen a festményeknél nem csupán a tintapettyek halmaza hat ránk, hanem egy bizonyos szinten a festő érzéseit, gondolatait is átélhetjük. A számítógéppel előállított képek esetében ezt a közvetítő szerepet a megfelelően megalkotott jelenettel kell helyettesíteni, ezért úgy gondolom, számítógéppel nehezebb hatásos képet alkotni, mint ecsettel és festékkel. A CD-mellékleten (28. CD Magazin/Pov-Ray) a befejező részhez tartozó animáció MPEG-2 formátumban megtalálható, és egy segédprogram is, amivel az egyes képeket animációvá fűztem és tömörítettem. Mivel a PoV-Rayt bemutató sorozat a végéhez érkezett, már csak egy hibajavítást szeretnék végrehajtani.
A sorozat hatodik részében a `halo` anyagtípusról adtam leírást, de a PoV-Ray újabb változataiban ez a típus már nem szerepel, használata hibát okoz, és a számolás már az előfeldolgozás során leáll. Helyette a `media` kulcsszó alkalmazható, ami több beállítási lehetőséggel rendelkezik, és gyakorlatilag a programhoz adott példákon keresztül, kísérletezéssel lehet leginkább megismerni. Az írásunkhoz készült példában is alkalmazom a patak fölött lévő pára megvalósításához.
Ennek kiindulási alapja a CD-mellékletre is felkerült ***hollow2.pov*** forrásállomány volt. Annak ellenére, hogy a PoV-Ray telepítőcsomagja ezeket a példákat tartalmazza, mégis felkerültek a CD-re, mert így könnyebben megtalálhatók és tanulmányozhatók.
Köszönöm az érdeklődést, és további kellemes perceket kívánok a PoV-Rayjel végzett munkához!

Fábián Zoltán
(dzooli@freemail.hu, dzooli@yahoo.com)
24 éves, jelenleg programozóként dolgozik. Szabadidejében szívesen kirándul, túrázik. Emellett szeret rajzolni, érdekli a 3D grafika és a Linuxszal kapcsolatban minden olyan program és programnyelv, amit még nem ismer vagy nem próbált ki.

Ablakkezelők (2. rész)

Az ablakkezelőkről szóló cikksorozat második részéhez az előzőeknél összetettebb ablakkezelőket válogattam össze.

Cikkünkben a következő ablakkezelő programokkal foglalkozom részletesebben: fvwm1, fvwm2 (fvwm95), blackbox és XFce.

Az fvwm

A név után tudatosan nem írtam semmilyen számot, ugyanis e két-három ablakkezelő tulajdonképpen egyetlen közös alapon nyugszik, a kezdetük azonos. Az fvwm maga meglehetősen egyszerű eszközkészlettel rendelkező ablakkezelő, és első ránézésre nem igazán nyeri meg a mai grafikus környezetekhez szokott felhasználók többségét. Ez az egyik legkorábbi Linuxon megtalálható grafikus felület. Természetesen nemcsak Linuxon, hanem más Unix-rendszereken is használható, például SunOS-en és Solarison. Ennek eredményeképpen rendkívül megbízható hátteret nyújt munkánkhoz.
Mi az alapvető különbség az fvwm1 és fvwm2 között?

- Az fvwm2 sok fvwm1 alaphiba javítását tartalmazza.
- Az rc-beállításokat és a vezérlőelemeket tartalmazó fájlok felépítését módosították.
- Könnyebben be lehet állítani.
- Több és jobb modulokat tartalmaz.
- Az fvwm2 memóriahasználata nagyobb az 1-éhez viszonyítva, de még így is nevetségesen kicsi a mai ablakkezelőkéhez képest (ezért használják előszeretettel az egy hajlékonylemezes Linux-terjesztésekben).

Az fvwm1

Miként neve is mutatja, ez a korábbi változat, és ennek megfelelően egyszerűbb is: csupán néhány alapszolgáltatást tartalmazó menü és eszköz áll a rendelkezésünkre. Az fvwm-re jellemző, hogy a különböző eszközöket modulokként tudjuk elindítani. Főbb elérhető eszközei a következők:

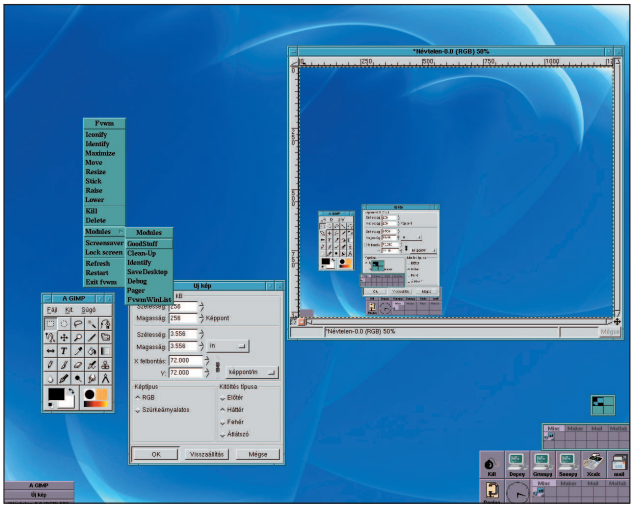
- ablaklista,
- többféle lapozóeszköz,
- indítópanel.

Az fvwm jellemzőinek egyike, hogy forrásból puszta felhasználóként is telepíthetjük a saját könyvtárunkba, indítása az ***.Xclients*** és a ***.xinitrc*** fájlokból történhet. Továbbá a Gnome alapfelületeként is alkalmazható. A szükséges beállításokat az ***.fvwmrc*** fájl tartalmazza. Saját menüvel bír, de amennyiben Linuxunk úgy lett telepítve, hogy az ablakkezelő átveszi a rendszermenüt, erre is hajlandó (Debian esetén ez a ***Menu*** program). A felület legfrissebb változata az ➲ ftp://ftp.fvwm.org/pub/fvwm/ címről tölthető le. Hivatalos honlapjuk a ➲ http://www.fvwm.org/ címen érhető el.

fvwm2 (fvwm95)

Elődjéhez képest az évek során jóval több eszközzel szerelték fel. A következő beállítóeszközöket találhatjuk meg benne:

- Screen setup: a képernyő beállításához.
- FvwmForm-Form: az ablakok ablakkezelőhöz tartozó részeinek beállításához.
- Base Config: az ablakok jellemzőinek beállítását végzi el.
- Bell Setup: a számítógép hangszórójának beállítására szolgál.
- Pointer Setup: beállítóeszköz az egérhez.
- Keyboard Setup: beállítóeszköz a billentyűzethez.



Az fvwm1–2 alapfelülete

Ezenkívül sok olyan további beállítási lehetőséget tartalmaz, amelyek a tulajdonságok kiválasztását menüből teszik lehetővé. Több betölthető modul is használható, ezek közül lássuk az érdekesebbeket:

- Buttons Bar: az fvwm1-ből már ismert gombsort jeleníti meg.
- Mini Button bar: az asztal felső szélén egy kis ikonokból álló indítópanelt jelenít meg.
- WMaker buttons: a WindowMaker ablakkezelő indítópanelét a saját beállításaival indítja el.
- Iconbox: az elindított ablakok ikonjait jeleníti meg, feladata a tálcáéhoz hasonló.
- Talk to Fvwm: grafikus parancssort jelenít meg, innen lehet a modulokat elindítani.

A régebbi változatok – amikor még a rendszermenü kezeléséhez a KDE és a Gnome nem volt használatos – a menüpontokat a ***/etc/X11/wmconfig/menu*** könyvtárban létrehozott fájlokban tárolták. Az alábbi példa az ***elm*** nevű fájl beállításait szemlélteti:

```
elm name "Elm"
elm description "E-mail olvas"
elm icon mail2.xpm
```

```
elm mini-icon mini-mail.xpm
elm exec "xterm -e elm"
elm group Alkalmazsok/Levl
```

Amennyiben a megjelenő felület nem nyeri meg a tetszésünket, a fvwm-re jellemző módon le is cserélhetjük: általában a leírás mellett előre elkészített *fvwmrc*-fájlokat találunk, amelyekkel a grafikus felület jellemzőit nagyon könnyen át tudjuk alakítani. Az említett fájlokat *.fvwmrc* néven másoljuk be */home* könyvtárunk főkönyvtárába. Ezután az újrainduló grafikus felület már az új beállításokat jeleníti meg.

Blackbox

A Blackbox is egyszerű felépítésű ablakkezelő, talán az egyik legfiatalabb. Az első változatot 1998-ban készítették el. Ma már nagyon sok Unix-alapú rendszeren használható.
Beállítását grafikus eszköz is segíti, ez a toolbox. Tulajdonképpen a beállításokat tartalmazó témafájlok szerkesztését teszi lehetővé. A számunkra leginkább megfelelő témát a menüben tudjuk kiválasztani.
A Blackbox néhány saját eszközzel is rendelkezik:

- `bbmail`: a beérkező levelek (*/var/mail/*) figyelésére szolgál. A levelek helye beállítható.
- `bbsload`: a rendszerterheltség figyelését végzi.
- `bbppp`: a telefonos hálózati kapcsolat vezérlését és figyelését végzi. Az alapértelmezett `ppp`-kapcsolatot használja.
- `bbdate`: dátumkijelző, csak a megjelenítés módosítható.
- `bbtime`: pontosidő-kijelző, csak a megjelenítés módosítható.
- `bblaunch`: az alkalmazások indítására szolgáló eszköz.

Ezek sajnos nem rendelkeznek grafikus beállítóeszközzel, de beállításaik a */usr/share/bbtools* könyvtárban lévő beállításokat tartalmazó fájlokban könnyen módosíthatók.

XFce

Az XFce az egyik legkedvesebb ablakkezelőm, több hónapig dolgoztam vele. Az eddig bemutatott ablakkezelők közül ez a legnagyobb tudású. Számos saját beállítóeszközzel rendelkezik, az eszközök maguk egyetlen kezelőpanelbe vannak összevonva (az alábbi rész az eredeti XFce-leírás magyar nyelvű fordítása).

A kezelőpanel

A panel főbb szolgáltatásai a következők:

- az alkalmazások indítása,
- a listamenük megnyitása,
- váltás a virtuális képernyők között,
- hozzáférés a felületbeállítási eszközökhöz.

Míg az óra nem bír különleges tulajdonságokkal, és csak a dátumot mutatja meg eszköztippként, addig a többi ikont alkalmazások indítására használhatod. Választhatunk analóg vagy digitális óra között, csak a bal (nálam a jobb) egérgombbal rá kell kattintanunk az órára. A kezelőpanelen lévő ikon módosításához vagy valamely ikonhoz való parancs hozzárendeléséhez a jobb egérgombbal kattintsunk az ikonra. A virtuális képernyők számát a felhasználó is megadhatja, 2-től 10-ig terjedően. Minden egyes ikon tetején egy pici nyíl található. E gombok mindegyike menülistát nyit meg, például itt lehetséges néhány szöveges módú alkalmazás – például a `vi` – számára szükséges terminál megnyitása. A terminál megnyitásához a parancs elindítása előtt a `term` kulcsszót kell használnunk.
Parancssor: `term vi`

Listamenük

A menülisták használata az alkalmazások XFce-panelből történő indításának egy további módját képezik. A húzd és ejtsd (drag and drop) módszer a különböző fájlkezelőkből (természetesen XFTree szükséges) is alkalmazható. A menü tetején egy bejegyzést (***Ikon hozzáadása...***) találhatunk, amellyel új menübejegyzéseket adhatunk meg. A vékony kicsi gomb a menü tetején csak akkor látszik, ha a ***Legördülő menük használata*** engedélyezett. Erre a gombra kattintva ezt a menüt kifűzhetjük, és a képernyőn bárhova mozgatható válik, sőt akkor is nyitva marad, ha bejegyzést választunk ki belőle, így használata nagyon kényelmes.

A beállítások és értékeik

Minden lehetséges beállítás elvégezhető grafikus felületen. Ennek megfelelően az XFce beállító párbeszédablakot is tartalmaz – négy belső füllel, amelyek a következők: ***Paletta***, ***XFce***, ***Ablakok*** és ***Automatikus indítás***.

A Paletta fül

Az XFce a népszerű GTK+ eszközkészleten alapul, amely a *gtkrc*-t (a felhasználó könyvtárában elrejtett stílusbeállító fájlt) használja ahhoz, hogy minden GTK+-alkalmazás azonos színsémájú legyen. A *gtkrc*-t általában egy témából másolják vagy kézzel írják, az XFce ugyanakkor a felhasználó által a palettabeállító képernyőn kiválasztott színértékekből és betűtípusból is képes *gtkrc* fájlt készíteni.
Az XFce meg tudja határozni, hogy a *gtkrc* fájlt a felhasználó testreszabja-e vagy sem. Amennyiben saját *gtkrc*-stílust szeretnénk felépíteni, vagy ha a Gnome beállítóközpont (*gnomecc*) stílusaiból kívánunk egyet használni, az XFce nem írja felül (ezért ha jelenleg *.gtkrc* fájllal rendelkezünk a könyvtárunkban és azt szeretnénk, hogy az XFce kezelje a stílusainkat, akkor vagy törölnünk kell azt, vagy a jelenlegi *~/.gtkrc* fájlukat át kell neveznünk). Ám ha nincs *.gtkrc* fájl a felhasználó saját könyvtárában vagy az üres, az XFce egy *gtkrc* fájlt készít, hogy az összes GTK+-os alkalmazás – a Gnome-alkalmazásokat is beleértve – teljesen azonos megjelenésű legyenek. Az Xfce szempontjából a paletta nyolc színből és egy betűtípus nevéből áll. Mindegyik szín az Xfce-felület egy-egy különleges elemét jelöli, mint például egérmutató, alapértelmezett ablakháttér, szöveges értékek stb. A palettához így kiválasztott színek az összes XFce-alkalmazásban használatosak lesznek (XFTree, XFClock, XFwm stb.).

Az XFce fül

Ez a párbeszédablak tartalmazza az összes XFce-tulajdonságot:

- Beállítja a várakozási időt az eszközsúgó megjelenése előtt.
- Az XFce a fő ablakot újrarajzoltatja a képernyőgombok színével.
- Egyszerű színek helyett színátmenetet használ (csak akkor érhető el, ha az előző lehetőséget kiválasztottuk, ez azonban színfelhasználás csökkentésére 256 színnél kevesebb színt használó terminálokon nem használható).
- Használhatóvá teszi a ***tear-off*** menüt, amivel a menüpontot ki lehet kapcsolni.
- Az XFce-panelben elérhető virtuális képernyők száma kettőtől tíz képernyőig engedélyezett.

- A listamenük száma: az XFce-panelben 0-tól 12 listamenüig választhatunk.
- Az ikonok méretének megadása a panelen. Ha az általunk megadott ikonok nagyobbak, mint a megadott érték, az ikon önműködően a gomb méretére méreteződik át.
- Az ikonok mérete a listamenükben. Ugyanúgy működik, mint az előbbi lehetőség.
- Az XFce-alkalmazásokban használt betűkészletek módosítása (mint a paletta esetén: a betűkészlet minden XFce-alkalmazásra kiterjedő lesz).

Az Ablakok fül

Ez a fül kezeli az XFwm-et, vagyis az ablakkezelőt és a szolgáltatásait. Az ablakok jellemzőinek megadására a következő lehetőségek használhatók:

- ***Click to focus windows***: ha ezt a lehetőséget engedélyezed, az ablakon belülre kell kattintanod, hogy az ablak átvegye a billentyűzetfókuszt. Máskülönben a billentyűzetfókusz az ablakokban az egérmutatót fogja követni. Ha mozgatod az egeret, a fókusz követni fogja.
- Amennyiben az ***Autoraise window*** lehetőség engedélyezett (nem választható azonban, ha a ***Click to focus*** lehetőség is ki van választva), a billentyűzetfókuszt tartalmazó ablakok önműködően előugranak, vagyis az összes többi ablak elé kerülnek.
- A ***Show contents of windows during move/resize*** lehetőség adja meg, hogy vagy az ablak tartalma, vagy a kerete mozgatás, illetve átméretezés közben látható legyen. A keretárnyék (ha ez a lehetőség nincs kiválasztva) további tájékoztatást ad az ablak helyzetéről és méretéről. Figyeljünk arra, hogyha a nem átlátszó mozgatást és átméretezést használjuk, rendszerünket vagy – távoli terminál használata esetén – hálózatunkat nagyon leterhelhetjük.
- Beállíthatjuk az ablakrács méretét. A ***Windows snapping*** nagyon jól használható eszköz, ha az ablakot úgy mozgatjuk, hogy egy másik ablakot vagy a szélét megérintjük – ilyenkor az ablak egy pillanatra megáll ebben a helyzetben.
- A működő ablak fejléce egyféle vagy átmeneti színekkel rajzoltatható ki.
- Az XFwm az ikonizált ablakokat önműködően rendezi a képernyőn. Kiválaszthatjuk, hogy az ikonok elhelyezésére a képernyő teteje (top), a bal oldala (left), az alja (bottom) vagy a jobb oldala (right) szolgáljon-e.
- A három utolsó beállítás az ablakkezelő által használt betűtípusok fajtáit változtatja meg.
  1. A fejléc által használt betűtípus.
  2. A menük által használt betűtípus.
  3. Az ikonok által használt betűtípus.

A Startup fül

Az e képernyőn található lehetőségek használatával megadhatjuk, hogy az XFce indításkor mely alapmodulokat (démonokat, illetve szolgáltatásokat) olvassa be.

Menük

A saját (***Main***) menüt az egér jobb (nálam a bal) gombjával a fő ablakra kattintva kaphatjuk meg, vagy az ALT+F2 lenyomásával juthatunk hozzá. Az ablakmenü a fő ablakra kattintva a középső egérgombbal érhető el (amennyiben kétgombos egered van, a bal és a jobb gomb egyidejű használatával), vagy az ALT+F1 lenyomásával.

A fájlkezelő XFTree

Az XFce saját fájlkezelője, az XFTree nagyon egyszerű fájlkezelő eszköz. Az XFTree önmagában is alkalmazza a húzd és ejtsd eszközt, de természetesen más GTK+-os alkalmazásokkal (XFce panel) egyaránt képes alkalmazni.

Egyéb beállítóeszközök

Az ***XFMouse*** egérbeállító eszköz. A XFMouse-nak köszönhetően az egérgombok működését és tulajdonságait meg tudjuk változtatni.
Az ***XFSound*** feladata egy hang lejátszása, ha valamilyen esemény történik az ablakkezelőben. Indításkor önműködően, modulként indul el.
Az ***XFPager*** kicsinyke eszköz, feladata az összes XFwm-felületet kicsiben mutatni. Az XFPagert modulként kell indítani.
Az ***XFbd*** a háttérbeállításokért felelős eszköz. Segítségével a háttérképhez tudunk képfájlokat beállítani. Az XFbd meglehetősen sok fájlformátumot képes használni: png, gif, jpeg, bmp és tiff.

Xfgnome

Az xfgnome modult használjuk a Gnome-val való együttműködésre. Ha az xfwm-et ***Gnome-érzékeny Window manager***-ként indítjuk el, és a ***Gnome applet***-ek alkalmazását engedélyezzük, olyan lesz, mint a ***Gnome pager***. Az xfgnome modul az XFce 3.2.0-es változatában jelent meg.

Telepítés

Az XFce-ben a könnyű telepítést, illetve a beállítást (`/usr/bin/xfcesetup`) két kényelmes parancsfájl segíti. Ezek a fájlok az xfwm-et és az XFce-t annyiszor indítják el, ahányszor a felhasználó az X-folyamatot elindítja (`startx`-szel vagy `xdm`-mel). Az előző beállítófájlok a rejtett ***.xfce_bckp/*** könyvtárba fognak kerülni. Ezután az előző fájlok visszaállítására az ***xfce_remove***-t kell használni. A könyvtárbeállítás a későbbiekben az `XFCE_DATA` környezeti változó egy másik könyvtárra történő változtatásával módosítható.

```
bash: export XFCE_DATA=/usr/local/share
csh: setenv XFCE_DATA /usr/local/share
```

Mivel az XFce a nyelvi támogatást (Native Language Support-sot – NLS) is kihasználja, győződjünk meg róla, hogy a `LANG` környezeti változó az országunknak megfelelően van-e beállítva.

```
bash: export LANG="hu"
csh: setenv LANG "hu"
```

Az eddigieket visszaolvasva úgy tűnhet, kicsit részrehajló vagyok az XFce felé. Ennek talán az is oka, hogy több magyar leírás állt róla a rendelkezésemre (a FAQ-t annak idején én fordítottam magyarra), illetve az is, hogy ennél az eszköznél a szükséges beállításokat mind grafikus felületen tudtam elvégezni. Ettől eltekintve úgy gondolom, a most bemutatott ablakkezelők sokkal barátságosabbak, mint az előző részben bemutatott szerényebb tudással rendelkezők. Az ablakkezelőkről szóló sorozatunk következő részében többek közt az icewm-ről és az enlightenmentről lesz szó.

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.

A Perl és a hálózat

Mint más nyelvekben, a Perlben is számos lehetőség nyílik a folyamatok közötti kapcsolattartás (IPC) megvalósítására.

Lássuk, miként beszélgethet egymással két különböző számítógépen csücsülő folyamat! Megoldásunkhoz használjuk az egyik legelterjedtebb programozási nyelvet, a Perlt.

Az IPC (interprocess communication)

Ugyanazon a számítógépen futó két folyamat rengeteg módon cserélhet adatot. Használhatnak szignálokat, fifókat, csővezetéket, és még számtalan más eszköz áll a programozó rendelkezésére (lásd man 1 perlipc). Minden megoldásnak megvan a maga előnye és hátránya. Az egyik természetesen hatékonyabban láthat el egyes feladatokat a másiknál, és bizonyos helyzetekben nem is mind használható. Létezik viszont egy olyan kialakítás, amelyet én még minden gond esetén használni tudtam, továbbá egyszerű és megbízható: ez pedig a foglalatok (socketek) használata. Foglalatokat nemcsak a helyi gépen futó folyamatok közötti adatcserére, hanem távoli számítógépek közötti kapcsolattartásra is fel lehet használni. Nekem azért tetszett meg nagyon, mert általa a Perlben objektumközpontú felületet kapunk, és nem (feltétlenül) kell az alacsony szintű függvényekkel bajlódnunk.

Foglalatok – miképpen épül föl egy kapcsolat?

Amikor két folyamat beszélgetni akar, először el kell dönteniük, hogy milyen tartományban találhatók. Ha a Unix-tartományt választjuk, csak a helyi számítógépen futó folyamatok kapcsolódhatnak egymáshoz. Az INET-tartomány használata teszi lehetővé a távoli gépek közötti adatcserét. Ezután mindkét folyamatnak létre kell hoznia egy-egy foglalatot. Ennek a foglalatnak be kell állítani a tulajdonságait, és ezután kaput kell neki foglalni. Az ügyfélnél ez a lépés elhagyható, a rendszermag megteszi helyettünk. A kiszolgálónál azért kötelező, mert tudnunk kell a folyamat kapuszámát, máskülönben nem tudjuk elérni. Ezt követően az ügyfél a megadott kapun már kapcsolódhat is a kiszolgálóhoz, és megkezdődhet a „beszélgetés”. Ez valóban beszélgetés, ugyanis ezt követően az ügyfél foglalata állományleíróként viselkedik, és ugyanúgy lehet bele írni és olvasni, mint egy szabályos fájlba. A beszélgetés továbbá kétirányú: a kiszolgáló ír valamit, amit az ügyfél olvas; ezután az ügyfél ír és a kiszolgáló olvas stb. Arra azonban vigyázni kell, hogy ez a foglalat egy olyan „állomány” leírója, amelynek nincsen vége. Pontosabban akkor van vége, amikor a kapcsolat lezárult. Ezért a kétirányú kapcsolattartáshoz ki kell találni egy „vége” jelet. Hasonlóan az indiánfőnök és a kisindán beszélgetéséhez, itt is, ha az egyik folyamat befejezte a mondanivalóját, uff-ot mond. Ebből tudja a másik, hogy ő következik, olvasás helyett írni kezd a foglalatra.

Perl – ahol mindez egyszerűbb

A Perl – mint fentebb már említettem – objektumközpontú felülettel segíti a programozók munkáját. Természetesen itt is léteznek azok az alacsonyszintű függvények, mint C-ben, tehát ha valaki ragaszkodik hozzá, használhatja őket.
A `Socket` modul alkalmazásával viszont mindez sokkal egyszerűbb. Amennyiben még esetleg nem foglalkoztál objektumközpontú programozással, akkor se ijedj meg. Az objektumközpontú programozás filozófiája csak egy, a strukturált nyelveknél az emberi gondolkodáshoz közelebb álló elképzelés. Eszerint először általánosítasz (osztályok), aztán leírod, hogy az eset miben különbözik az átlagostól (objektumok). A Perl-kézikönyvoldalak nagyon jó bevezetőt tartalmaznak az objektumközpontú programozáshoz. Ha valamennyire ismered már a Perlt, de nem tudod, mi az objektumközpontú programozás, feltétlenül olvasd el a perlboot oldalt (1. fejezet).

És ahogyan mindez a gyakorlatban fest

Ha Unix-tartományban akarsz dolgozni, a `Socket::UNIX`-osztályt kell használnod (man 3perl `IO::Socket::UNIX`), INET-tartomány esetén pedig – ki gondolta volna – a `Socket::INET`-osztályt (man 3perl `IO::Socket::INET`). Az osztályból készítesz egy objektumot és már kész is a foglalat. Tudnod kell, hogy Perlben nincs külön kinevezett objektum-függvénylétrehozó (constructor). Egy osztály bármely eljárása lehet függvénylétrehozó, ha meghívja a `bless()` függvényt, csupán megfelelően le kell írnia az osztályt. Természetesen mindenki az egységes felület kialakítására törekszik, ezért a függvénylétrehozót többnyire `new()`-nak hívják. Ebben az esetben nem kell az `IO::Socket::INET->new()` formához ragaszkodnod, használhatod a jobban olvasható `new IO::Socket::INET` kifejezést. A függvény egy asszociatív tömböt vár. Mivel ennek elemeit a programban sehol máshol nem használjuk, nyugodtan lehet anonim. A szükséges kulcsok egy kiszolgáló esetén a következők lehetnek:

LocalPort

Annak a kapunak a száma, amelyen a szolgáltatást indítani akarod. Ne felejtsd el, hogy az 1024 alatti kapuk használatához rendszergazdai jogok szükségesek!

Proto

A használni kívánt protokoll. Ez TCP vagy UDP lehet attól függően, hogy kapcsolat- vagy datagramközpontú protokollal akarsz dolgozni. Ha nem tudod, hogy mik ezek, használj TCP-t. A TCP teszi lehetővé a programozók számára, hogy minden csomag megérkezzen, és abban a sorrendben tegyék, ahogy elküldte őket.

Listen

Ez a várakozási sor hossza. Amíg egy ügyféllel foglalkozol, nem tudsz többet kiszolgálni (kivéve, ha a kiszolgálót többszálasra készíted). A többiek ezalatt a várakozási sorba kerülnek, és a kapcsolatra várnak. Ha a várakozási sor is betelik, a további

1. lista A kiszolgáló

```
#!/usr/bin/perl -w

use strict;
use IO::Socket;

my $kiszolgalo = new IO::Socket::INET (
             LocalPort => '1111 ,
             Proto =>  tcp ,
             Listen => '1 ,
             ReuseAddr => '1'
             );

my ($ugyfel,$most);

while ($ugyfel=$kiszolgalo->accept()) {
    $most = localtime(time());
    print $ugyfel $most."\n ;
    close($ugyfel);
}

close($kiszolgalo);
```

2. lista Az ügyfél

```
#!/usr/bin/perl -w

use strict;
use IO::Socket;

my $ugyfel = new IO::Socket::INET (
             PeerAddr =>  localhost:1111 ,
               Proto =>  tcp
             );

while (<$ugyfel>) {
    print;
}
```

ügyfelek „A kiszolgáló nem elérhető” ICMP-üzenetet kapják. A SOMAXCONN állandó tartalmazza az operációsrendszer-függő legnagyobb értéket.

ReuseAddr

Ez már nem kötelező, én viszont ajánlom a használatát. Ha ugyanis egy program megfogott egy kaput, a rendszermag a felszabadítása után azt egy kis ideig nem engedi át új folyamatnak. Ha a `ReuseAddr`-nek igaz értéket adsz, akkor kapufoglalás előtt a rendszermagot minden szabad kapu elengedésére utasítja.
Ügyfél esetén ez még egyszerűbb:

`PeerAddr`

A távoli folyamat címe `<IP c m>:<kapu szÆma>` formában.
`Proto` (lásd fentebb)
Ezután kiszolgáló esetén a foglalatobjektum `accept()` elemfüggvényét kell használnod. Ez mindaddig nem tér vissza, amíg nem történt sikeres kapcsolódás. Ekkor a kapcsolódott ügyfél foglalatát visszaadja. Erre a foglalatra lehet írni, illetve olvasni. Ezzel létrejött a párbeszéd.
Minden foglalat létrehozása után érdemes meghívni annak `autoflush()` elemfüggvényét. Az 1.18 előtti `IO::Socket` alapértelmezés szerint a foglalatra írt adatot nem küldi el rögvest, hanem átmeneti tárba helyezi. Így az ügyfél az adatot nem azonnal kapja meg, csak amikor az átmeneti tár telítődik, majd szükségszerűen ürül. Ezt bírálhatod felül, ha az `autoflush()` függvénynek igaz értéket adsz, ami az értelmezőt az adatok azonnali küldésére kényszeríti, például `:$kliens->autoflush(1);`.
Az 1.18-as és az azt követő változatoknál azonban ez az alapértelmezett, így ezt a függvényt pusztán együttműködési okokból szokás használni.
A példa egy nagyon egyszerű kiszolgáló-ügyfélpárt mutat be. A kiszolgáló folyamatosan várakozik, és minden kapcsolódó ügyfélnek kiírja a pillanatnyi időt, majd zárja a kapcsolatot. Az igazat megvallva a kiszolgáló nem igazán „korrekt” abból a szempontból, hogy az utolsó sor (`close $server;`) sosem fut le, a programot CTRL-C-vel kell leállítani. Ez a rendszer működése szempontjából semmilyen zavart nem okoz, a Perl-értelmező a programból történő kilépéskor minden nyitott állományt (a foglalatokat is beleértve) bezár.

Végezetül

A CD-mellékleten (28 CD Magazin/Perl) egy kidolgozott kiszolgáló-ügyfélpárt találsz, ami már elég összetett. Különösebb haszna ennek sincs, de jó példa, és szerintem sokat lehet belőle tanulni. Ha ilyen vagy hasonló programot fejlesztesz, van számodra egy tanácsom. Ha egy mód van rá, kezdd a kiszolgálóval, és ügyfélként a Telnetet használd. A példában a `telnet localhost 1111` ugyanazt az eredményt adja, mint a saját kézzel írt ügyfelünk.
Remélem, hogy e módszer bemutatásával sok fáradságos munkától tudtalak megkímélni. Használd sokat a Perlt! Nemcsak azért, mert gyorsan lehet benne alkalmazásokat készíteni, hanem mert felületfüggetlen. Példánk bármelyik példánya futhatott volna Windows, Machintos vagy akár Solaris alatt is.

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

Leáll, kikapcs!

Amennyiben azt szeretnénk, hogy gépünk leállításkor ki is kapcsoljon, akkor használjuk a `halt` parancsot a `-p` kapcsolóval vagy a `poweroff` utasítást.
Ez a legtöbb rendszerben alapértelmezett eljárás, és a */etc/init.d/halt* héjprogramban ellenőrizhetjük. Emellett az is fontos, hogy a rendszermag támogassa az ilyen szolgáltatásokat (azaz a *General* rész alatt az *Advanced Power Management BIOS Support* be legyen fordítva), illetve a BIOS-ban is engedélyezzük.

Szy György

A méretezhető próbarendszer

A programok kipróbálása nem móka, hanem igen fontos feladat; az Open Source Development Lab and Scalable Test Platform ebben szeretne segíteni.

A nyílt forráskódú fejlesztői laboratórium és méretezhető próbarendszer (Open Source Development Lab – OSDL) nem nyereségközpontú cég. A Linux méretezhetőségének és távközlési felhasználásának tökéletesítésén dolgozik. Az OSDL támogatói (➲ http://www.osdlab.org/sponsors) egy teljes teszt- és fejlesztői labort rendeztek be több terabájtos tárolókkal, kettőtől tizenhat processzorig felszerelt SMP-kiszolgálóparkkal. A laborban elhelyezett nagyvállalati kategóriába tartozó gépeket a fejlesztők távolról is elérhetik.
A tesztek létrehozásán és futtatásán közösen dolgozunk a fejlesztőkkel. A folyamat során rájöttünk, hogy bizonyos dolgokat minden, a laborba érkező próba során újra és újra el kell végeznünk. Feljegyeztük egy átlagos próbafuttatás lépéseit és kiderült, hogy számos közülük önműködővé tehető. A Scalable Test Platform (STP) a feladatok automatizálását tűzte ki célul a kérés leadásától a végső jelentés visszaadásáig.

Gondok a próbamódszerrel

Már maga a teljesítményvizsgálat eleve olyan gondokat vet fel, amelyek túlmutatnak cikkünk és az STP keretein. A jelenlegi próbaeljárások esetén is akadtak azonban megoldható kérdések, a laboratórium tehát ezekre próbál összpontosítani. Nem szabad megfeledkezni arról, hogy az általunk végzett teljesítménymérés teljesen különbözik azoktól az eljárásoktól, amelyekkel a piac számára készítenek összehasonlításokat.
A próbakörnyezet beállításait ritkán írják le, a leírások többnyire azokra a szempontokra korlátozódnak, amelyek a kitűzött célok tükrében a próbát végző számára fontosnak tűnnek. A részletek hiánya azonban a későbbiek folyamán gondokhoz vezet, amikor a jelentést más elemzők próbálják értelmezni. Nem ritka, hogy az elemzőnek az egész próbát meg kell ismételnie, hogy a hiányzó adatokhoz hozzáférjen. Az is gyakran előfordul, hogy a próbafolyamatnak csak egy része önműködő, az emberi beavatkozást igénylő lépésekről pedig nincs kielégítő leírás, így a próba megismételhetetlen.
A teljesítményvizsgálat nagyon erőforrás-igényes – mind idő, mind eszközök tekintetében. Hány nyílt forrású fejlesztést végző programozó fér hozzá egy ötven ügyfélből álló kétutas kiszolgáló-ügyfél rendszerhez, amelyet gigabites hálózat köt össze, hogy kipróbálhasson több 8 és 16 processzoros gépből álló kiszolgálóparkot? Kevés cég engedheti meg magának, hogy ilyen eszközöket vásároljon, és ha mégis, jó néhány vezetőt kell meggyőzni a kiadásokból származó jövőbeli eredmények megtérüléséről. Amennyiben egy fejlesztőnek jó ötlete támad, de a fenti okok miatt nem tudja kipróbálni, az elképzelés jó eséllyel feledésbe merül.
Jelenleg nem létezik olyan központi hely, ahol a jól leírt teljesítmény-, üzembiztonsági és szabványossági vizsgálatok eredményeit elérhetnénk. A kutatók kénytelenek saját maguk próbafuttatásokat végezni, vagy az elérhető kétes megbízhatóságú teszteredményekre hagyatkozni, és a legjobb esetben is csak találgathatnak. A rendszerfelügyelők nem fordulhatnak egy központi nyilvántartáshoz, amelyből kiderülne, hogy a tervezett terheléshez melyik rendszermag-terjesztés-vas összeállítás illik a legjobban. Az ilyen természetű adatok hiánya miatt gyakran zavaros a kép a milliónyi Linux-megoldás teljesítményét és megbízhatóságát illetően.

Linux-rendszermagfejlesztés

A Linux-rendszermag fejlesztőinek nem áll módjukban hosszadalmas teljesítmény- és üzembiztonsági vizsgálatokat futtatni a hibajavításokon. Még abban az esetben sem, ha a fejlesztő történetesen hajlandó időt áldozni a kipróbálásra – a rendszer telepítéséhez és beállításához is gyakran szaktudás és különböző vasak, alkatrészek szükségesek. Alkalmanként ez ahhoz vezethet, hogy a rendszermag fejlesztői és üzembiztos változatában egyaránt nemkívánatos hibák jelennek meg. Az is előfordulhat, hogy az új hibajavítás egy korábban már kijavított hibát okoz, de ezt a regressziós tesztelés hiánya miatt senki sem veszi észre.
A Linux-rendszermaggal foglalkozó levelezési listán többen felvetették, hogy az új hibajavításokat szabványosított próbáknak kellene kitenni. Számos felhasználó és fejlesztő egyetért abban, hogy egy olyan egyszerű eljárás, amely a teljesítmény- és üzembiztosságfelmérést foglalná magában, a szabványokhoz való igazodást nagyító alá venné és tartalmazná a regressziós tesztelést, igen hasznos lehetne a Linux-rendszermag fejlesztése során. Nem írhatunk próbát minden létező hibára, de a típushibákat ellenőrizhetjük. Általában nem túl bonyolult dolog a hiba felfedezése és javítása után hozzáadni egy regressziós tesztesetet az eljáráshoz. A gond nem a próbák létrehozásával van. A legtöbb fejlesztő elismeri, hogy a próbák elvégzése szükséges: a nehézség abból származik, hogy míg a kódolás izgalmas feladat, a tesztelés többnyire unalmas. Ha a fejlesztő mással próbáltathatná ki a kódját, és amíg az a valaki a piszkos munkát elvégzi, ő folytathatná a kódolást, sokkal többen hajlanának a hibajavítások kipróbálására.

A méretezhető próbarendszer

Az STP (Scalable Test Platform) a kipróbáláshoz összeállított gépparkot és a rajta futó programok összességét takarja. A háttérfeladatok elvégzésére kifejlesztettük a Brimstone-t, ami a kötegelt feladatokat és a próbákat irányítja. A kéréseket az Eidetic-felületen – felhasználóbarát webes felület – keresztül lehet feladni, esetleg a brim-gate elektronikus leveleket feldolgozó program is felhasználható erre. Az egyszerű felületek segítségével kevesebb mint két perc alatt egy egész tesztsorozatot össze lehet állítani.
Az STP használata azzal kezdődik, hogy a fejlesztő a hibajavítást elküldi a rendszermagot tartalmazó CVS-fába, majd kér egy próbasorozatot a módosított rendszermaggal. A teszt végeztével a rendszer a részletes eredményeket elektronikus levélben visszaküldi a fejlesztőnek. Az eljárás még tovább egyszerűsíthető, hiszen a fejlesztő írhat egy rövid parancsfájlt, amely kevesebb mint öt sorban elvégzi a CVS-műveleteket és a kérés feladását. Így a hibajavítás ellenőrzése akár egyetlen paranccsal is megoldható. Minden, ami a teljes próbafuttatás-

hoz szükséges, egy másodperc alatt el lenne intézve.
Az STP számára fenntartott eszközök között található egy 1,8 TB-os tárolórendszer, amely több üvegszálon keresztül csatlakozik minden kiszolgálóhoz (ezek mindegyike négy vagy több processzorral rendelkezik). A kiszolgálók között található két-, négy-, illetve nyolcprocesszoros gép (mindegyikből kettő), továbbá egy 16-processzoros gép is. Egy másik 16 Itanium processzort tartalmazó NEC AzusA kiszolgálóra már leadtuk a rendelést. Emellett több mint ötven ügyfélgéppel is rendelkezünk, ezeket az STP számára egy gombnyomással elérhetővé tehetjük. Jelenleg azt fontolgatjuk, hogy a meglévő gépek mellé néhány egyprocesszoros gépet is beállítunk, így a fejlesztők biztosak lehetnek abban, hogy módosításuk a legtöbbek által használt telepítéseket nem befolyásolja.
Az Eidetic, a Brimstone és az elektronikus levélátjáró programok mind GPL-esek, tehát aki saját próbafuttásaihoz keres környezetet, használhatja.

Próbafuttatás igénylése

Az első teendő a jelentkezés: ingyenes és a segítségével OSDL Lab Associate lehetsz (lásd ➲ http://www.osdlab.org). Ezekután meg kell adnod egy tesztfuttatást, a processzorokkal kapcsolatos részleteket, a gépek felépítésére vonatkozó korlátozásokat (ez választható), és egy nem kötelező LILO parancssort (az elérhető fizikai memória méretét korlátozhatjuk vele). Az adatok elküldése után egy kevéske időt kell szentelned a beállításokat tartalmazó oldal kitöltésére, majd feladhatod a kérést. A próba lefutásának idejétől függően akár 25 perc alatt megkaphatod az eredményt. A környezet teljes leírását és az eredményeket megőrizzük a webkiszolgálónkon – ennyire egyszerű az egész. A rövid próbák is legalább 15 percig tartanak, mert minden alkalommal frissen telepített operációs rendszerből indulunk ki. Mivel a folyamat teljesen önműködő, a rendszert munkaidőn kívül is használhatod. Ez azt jelenti, hogy minden fejlesztő számára elérhetők vagyunk – a földrajzi elhelyezkedésre való tekintet nélkül.

Részletek

Mivel az STP-t csak most kezdtük igazán használni, egyelőre kevés a futtatható próba. E cikk írásakor a következők állnak a felhasználók rendelkezésére: ***Juan Quitela*** „memtest”-je (a VM-et próbára tevő program), a dbench (Samba) fájlrendszer próbája, a mindig népszerű rendszermagfordítás, a CVS-igénybevétel próbája és az lmbench.
Tervezzük számos több kiszolgálót és alkalmazást igénybevevő próba elkészítését is, ezek közül az Apache- és a MySQL-alapú próbákat emelném ki. Természetesen nyílt forrású kezdeményezés révén bárki segítségét szívesen fogadjuk, főleg a próbák önműködővé tétele terén.
A Linux Test Project (LTP) keretein belül együttműködünk a SGI és az IBM fejlesztőivel. Célunk, hogy az LTP-t célzott és általános terheléspróbákkal bővítsük ki. Az LTP rendszermag-tulajdonság próbája önműködővé tételéhez csaknem készen áll. Ezzel nagy regressziós tesztgyűjteményt kapunk, valamint biztos alapot a megbízhatósági próbákhoz. A LTP jövőbeli tervei között néhány önálló próba fejlesztése szerepel, amely remek cél az STP számára.

Nathan Dabney (smurf@osdlab.org)
a Slackware óta (1994) használ Linuxot. Szereti megdönteni a rossz elméleteket, és szívesen sétál a menyasszonyával esőben.

PHP-vel biztonságosan (1. rész)

Mire figyeljünk, ha nem akarjuk, hogy féltett adataink egy rosszul megírt PHP-program révén illetéktelenek kezébe kerüljenek.

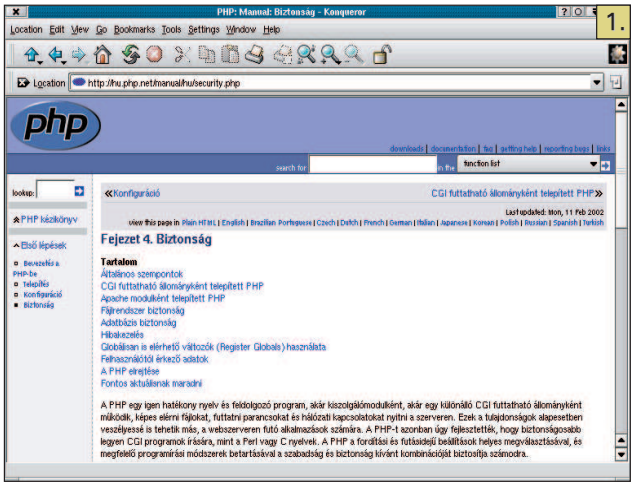
Félreértés ne essék, teljesen tökéletes rendszer nem létezik. Olvasóink e sorok végigolvasása után sem lesznek képesek törhetetlen webes rendszereket írni és fenntartani, de a célom nem is ez, hanem csupán annyi, hogy fellebbentsem a fátylat azokról az elkövethető hibákról, amelyek a leggyakrabban ütik fel fejüket a gombamód szaporodó PHP-ben írt webes alkalmazásokban.
Cikkem egy GNU/Linux-környezetben működő Apache webkiszolgálót és egy 4.x változatszámú PHP-t használó rendszert vesz alapul.

Kinek a feladata?

Egy PHP-projekt körül a biztonsági teendők alapvetően két nagy csoportra bonthatók. A telepített webkiszolgáló és PHP-értelmező beállításait a biztonsági követelményeknek megfelelően finomítani kell. Ezek a rendszergazda hatáskörébe tartozó feladatok.
A rendszergazda mindent megtehet a PHP-rendszer védelme érdekében, de erőfeszítései hiábavalóak, ha maguk a PHP-ben írt programok nem felelnek meg azoknak az alapvető feltételeknek, amelyek egy nagyközönség számára szánt programtól elvárhatók. Ezen elvárások röviden összefoglalva: a programmal a felhasználó ne tehessen olyat, amit mi nem akartunk.

A forráskód elrejtése lenne a megoldás?

Egyes elképzelések szerint a legjobb védekezési módszer a kívülállók sötét tudatlanságban tartása, azaz ha semmit nem árulunk el programunk belső felépítéséről, vagyis eltitkoljuk a forráskódját. Ez nem igazán célravezető megoldás. A trükkös, játékos kedvű idegenek, akik webalkalmazásunk meggyötrésére hivatottak, a sok kis apró adatból nagyon sok információt ki képesek gyűjteni. Inkább arra érdemes törekedni, hogy PHP-s rendszereinket úgy írjuk meg, hogy egyetlen rosszindulatú felhasználó se tudjon károkat okozni, még a teljes forrás ismeretében sem. Ennek már csak egészségügyi szempontból is hatalmas előnyei vannak, hiszen nem kell amiatt rettegnünk, hogy féltett forrásunk illetéktelenek kezébe kerül.
Működő rendszerünk fájljai között azonban olyanok is akadhatnak, melyeket joggal féltünk a külvilágtól. Teszem azt, egy beállításokat tartalmazó ***.inc*** fájlban lehetnek olyan érzékeny adatok, amelyeket nem a nagyközönségnek szántunk, például egy adatbázis-hozzáférés jelszava. Átlagos kiszolgálóbeállítást alapul véve az ilyen ***.inc*** fájlok – ha a nyilvános webdokumentumok könyvtárában helyezkednek el – elérhetővé válhatnak. Hibás lépés az ilyen érzékeny adatokat tartalmazó fájlokat itt tárolni. Az első, amit megtehetünk, hogy a fájlrendszer nyilvánosan el nem érhető pontján tároljuk őket. Ha nem tennénk, a támadási lehetőségeket felmérő kívülállónak elég lenne csupán a fájl nevét megismernie, és máris olyat láthatna, amit nem szabad. Ennek oka, hogy a webkiszolgáló csak olyan fájlkiterjesztésekre ereszti rá a PHP-értelmezőt, amit annak beállításaiban sorolhatunk fel. Általában a következő kiterjesztéseket vizsgálja: a ***.php .php3 .phtml***. Ezek közé a ***.inc*** is



A PHP-leírás ide vonatkozó része már magyarul is elérhető

felvehető, viszont senki sem szeretné, ha a ***.inc***-állományokat önállóan is futtatni lehetne. Ezért kell a webkiszolgáló dokumentum saját könyvtárán kívül helyezni azt, amit nem akarunk, hogy közvetlenül elérjenek, vagy akárcsak megcímezzenek. Ezt rendszerint felhasználóként is megtehetjük. Ha mégsincs rá lehetőség, más megoldás is létezik: ha beillesztendő fájljainkat szintén ***.php*** kiterjesztéssel láthatjuk el. Ekkor viszont arra kell figyelni, hogy ezek a nem önálló futásra tervezett programrészletek önmagukban is meghívhatók, és ezáltal újabb biztonsági rések keletkeztek. Ennek legkönnyebb kivédése, ha beillesztett fájljaink elején leellenőrizzük, nem önállóan lettek-e meghívva. Ezt a következő módon érhetjük el – már feltéve, hogy a védeni kívánt fájl a ***szamlalo.php*** (egy ***szamlalo.inc*** nevű állományt nevezve át) névre hallgat. A beillesztendő fájlt az *1. listán* látható módon kezdjük. Ezáltal a közvetlen lefuttatást meghiúsítottuk, ilyen próbálkozások esetén a beillesztésre hivatott ***szamlalo.php*** weboldalunk nyitólapjára irányít át. A fenti programrészletet eseménynaplózó rendszerrel ki lehet egészíteni , amely rögzíti számunkra az ilyen gyanús eseményeket.

Ne bízzunk a bejövő adatokban sem!

A webkiszolgálón futó programokban a legkönnyebben elkövethető hiba, ha az oldalról oldalra átadott adatokban vakon megbízunk. Egy egyszerű címsorban továbbított adatsort bárki módosíthat, és ezzel rossz esetben a program működése nagyon eltérhet attól, mint amire szántuk.
Minden egyes megkapott adatot érdemes alaposan megvizsgálni, hogy megfelel-e az elvárt formai és tartalmi követelményeknek. Ha nem tesszük, jó esetben is előfordulhat, hogy a nem kívánt tartalmú adatot is elfogadjuk, például a távoli jövőbe mutató születési dátumot – és ez még csak a legkisebb rossz, ami megtörténhet…

1. lista Számláló

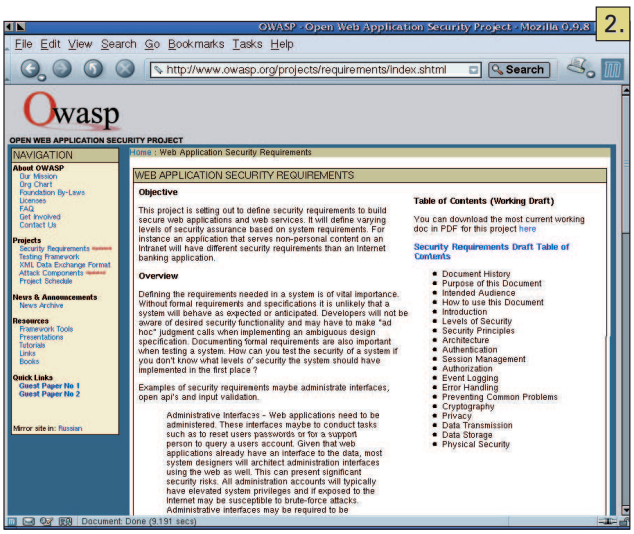
```
<?

if (basename($PHP_SELF) == "szamlalo.php") {

     header("Location: /");
     die();

}

?>
```

2. lista A letölthető anyagok

```
<HTML>
<HEAD>
     <TITLE>Let ltheti anyagok</TITLE>
</HEAD>
<BODY>

<FORM ACTION="letoltes.php" METHOD="POST">

     <SELECT NAME="mit">
       <OPTION VALUE="dokumentacio.pdf"
       >dokumentacio.pdf
         ➥<OPTION VALUE="terkep.pdf">
         ➥terkep.pdf
       <OPTION VALUE="tech_parameterek.pdf">
       ➥tech_parameterek.pdf
     </SELECT><BR>

     <I
NPUT TYPE=submit NAME="indit"
➥VALUE="Let ltØs ind tÆsa">

</FORM>

</BODY>
</HTML>
```

3. lista A letoltes.php

```
<?

// szÆmlÆl  f ggvØny behœzÆsa
include "szamlalo.inc";

$download_dir = "/home/endre/downloads/";
$letoltendo_fajl = $download_dir.$mit;

if  (file_exists($letoltendo_fajl)) {

     header("Content-Type: application/pdf");
     readfile($letoltendo_fajl);

     // ez a f ggvØny n veli a szÆmlÆl t
     szamlalo_novel($mit);

} else {

     echo "HIBA: A fÆjl nem lØtezik!";

}
```



2.

Tegyük fel, weboldalunkon letölthető pdf formátumú dokumentumokat szeretnénk felkínálni a nyájas közönség számára. Mindezt ráadásul úgy, hogy a letöltéseket számon tarthassuk. Kézenfekvő megoldásnak tűnik a 2. és *3. listán* látható PHP-támogatott letöltő programocska megírása.
Senkinek nem ajánlom, hogy ilyen letöltőrendszert használjon! A gond abban lakozik, hogy a *letoltes.php* feltételezi, hogy az őt meghívó űrlapba senki sem fog belekotorászni. Pedig megteheti. Bárki mentheti az űrlapot tartalmazó oldalt a saját gépére, és kedvére módosíthatja. Ráadásul nincs, aki megakadályozza abban, hogy mondjuk a `mit` nevű bejövő adat értékét erre változtassa: ***../../../../etc/passwd*** (lásd *4. lista*). Letöltendő fájlunk ekkor a ***/home/endre/downloads/../../../../etc/passwd*** lesz. Ez érvényes elérési útvonal, és megegyezik a sima ***/etc/passwd*** hivatkozással. Tegyük fel, hogy jelszavainkat itt, és nem a shadow állományban tároljuk, és a baj már meg is történt. Az alattomos gonosztevő a próbálgató módszerrel máris nekikezdhet a kódolt jelszavak visszafejtésének. De mégha a shadow telepítve is van, e baki által a kiszolgáló felhasználólistája akkor is illetéktelenek kezébe került. Ilyen esetekben a védekezésnek több módja is akad:

- Elfogadható megoldás: a `basename()` függvény segítségével a `$mit` változó elejéről a könyvtárhivatkozásokat – csak a fájlnév megtartásával – levághatjuk.
- Kicsit jobb megoldás: használhatjuk az `ereg()` függvényt, hogy kiszűrjük a `../` részleteket, és a rendszergazdát a letöltés megtagadása mellett levélben értesítsük. A PHP-leírás alapján ilyen `ereg`-szűrésre példa a következő: `ereg('^[^./][^/]*$', $mit)` Ha e kifejezés értéke hamisnak bizonyul, az rettentően gyanús.
- Az elegáns megoldás: az űrlapban nem közvetlenül a fájlnevekkel dolgozunk, hanem minden letölthető fájlhoz azonosítószámot rendelve az azonosítót kérjük be. Ezzel a letölthető anyagok halmazát egyértelműen egy adott fájlcsoportra szűkíthetjük.

4. lista A módosított űrlap

```
<SELECT NAME="mit">
  <OPTION VALUE="dokumentacio.pdf">
  ➥dokumentacio.pdf
    <OPTION VALUE="terkep.pdf">
    ➥terkep.pdf
  <OPTION VALUE="tech_parameterek.pdf">
  ➥tech_parameterek.pdf
  <OPTION VALUE="../../../etc/passwd">
  ➥/etc/passwd
</SELECT><BR>
```

5. lista Űrlap némi hibával

```
<FORM ACTION="change_pw.php" METHOD="POST">

    <B>Jelsz vÆltoztatÆs:</B>
    ➥<?echo$username?><BR>
    <INPUT TYPE="hidden" NAME="user"
    ➥VALUE="<?echo$username?>"><BR>

     j jelsz :<BR>
    <INPUT TYPE="password"
NAME="password1"><BR>

    Jelsz  mØgegyszer:<BR>
    <INPUT TYPE="password"
    ➥NAME="password2"><BR>

    <INPUT TYPE="submit" VALUE="OK">

</FORM>
```

Mindemellett hasznos lehet az űrlapot feldolgozó oldalon a `$HTTP_REFERER` Apache környezeti változót is vizsgálni. Ha ez nem a mi webes űrlapunk címe, a feldolgozást meg lehet tagadni.

Biztonságos adatkezelés

Tegyük fel, hogy egy weboldalon bizonyos szolgáltatásokhoz csak felhasználónév és jelszó bekérése után akarjuk megengedni a látogatók hozzáférését, mindezt ráadásul testreszabott jogosultságokkal. Egy ilyen webes rendszer gyenge pontja a felhasználói adatokat módosító almodul lehet. Tegyük fel, egy „geza” néven belépett felhasználó jelszómódosítás kérése esetén az *5. listán* látható űrlapot kapja.
Itt a „user” nevű rejtett űrlapelemben tárolódik, hogy kinek is a jelszavát kell megváltoztatni. Egy ilyen űrlappal bármelyik bejegyzett felhasználó átveheti egy másik belépőkódját, csak a rejtett mezőben át kell írnia a felhasználónevet. Ez nyilvánvalóan tervezési hiba, de sajnos találkozni lehet vele a weben.
Hogy is kellene ehelyett eljárnunk? Ha http-azonosítást használunk: a `$PHP_AUTH_USER` változóban megtalálható, ki is a belépett felhasználó.
Egy másik elfogadható megoldás, ha munkamenet- (session) kezeléssel egyszerű űrlapon megvalósított beléptetéssel oldjuk meg az azonosítást. A munkamenet-kezelés nemcsak hasznos módszer, de a biztonságos adatkezelésben is hasznos

társ. Ekkor ugyanis elég lapról lapra egyetlen, a kapcsolatunkat meghatározó egyedi azonosítót küldözgetni. Egy ilyen azonosító olyan hosszú és bonyolult karaktersorozat lehet, aminek az ellophatósági (vagyis a címsorban átadott vagy rejtett űrlapelemként beépített azonosítót módosítgatva valaki más kapcsolatazonosítóját kapjuk meg) valószínűsége a nullával egyenlő. Ha egy ilyen „elköthetetlen” kapcsolatazonosítás adott, minden hozzá tartozó adatot – például azt, hogy ki lépett be ezen a kapcsolaton – a kiszolgálón tárolhatunk.
Ekkor bármilyen adatmódosító űrlap esetében nem az űrlapon keresztül kell küldeni, hogy kinek az adatait módosítjuk, hanem az a kapcsolatkiszolgálón tárolt adataiból kiderül.
Jelen bevezető cikkben csak a legalapvetőbb hibák és kivédésük módozatai kerültek terítékre. Soron következő írásomban kicsit mélyebbre ások a PHP és a biztonság kapcsolatában, és a rendszergazdai feladatkörbe tartozó tennivalókra is sor kerül.

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

Kapcsolódó címek

A PHP-leírás vonatkozó része immár magyarul
➲ http://hu.php.net/security

Összefoglaló a webalkalmazások biztonsági követelményeiről
➲ http://www.owasp.org/projects/requirements/index.shtml (2. kép)

Ismertetés a telepített PHP biztonságossá tételéről
➲ http://www.onlamp.com/pub/a/php/2001/03/29/php_admin.html (3. kép)

3D-programozás Pythonban

Jason néhány Python-modul segítségével PyOpenGL programozási módszereket mutat be nekünk, épp csak érintve az OpenGL felszínét.

A grafikaprogramozás olykor fárasztó, igen fárasztó tevékenység. Ha programjainkat behemót 3D-függvénykönyvtárakhoz csatoljuk, a fordítási idő megnövekszik. Mivel gyakran megesik, hogy rengeteg finomhangolás szükséges ahhoz, hogy minden tökéletesen nézzen ki, általánosnak mondható, hogy az apróbb változtatások kipróbálását és a hosszú fordítgatást igénylő programokat összecsapják. Ezek a hosszadalmas hibakeresési ciklusok elég okot szolgáltatnak arra, hogy 3D-alkalmazásaink prototípusát olyan magas szintű nyelv segítségével készítsük el, mint amilyen a Python is.

Rengeteg 3D grafikai API-hoz a Pythonban kiegészítés is található. IRIX-rendszereken a Pythont olyan modullal csomagolják, amely hozzáférést nyújt az SGI IRIX GL könyvtárához. Egy Python-programból a JPython segítségével akár a Java3D API-t is használhatjuk, mely egy olyan Python-változat, ami Java virtuális gépből fut. Ez a cikk elsősorban az OpenGL-re összpontosít elterjedtsége és a hozzá biztosított, mind a Pythonhoz, mind a Linux alá megvalósított nagyszerű támogatás miatt.

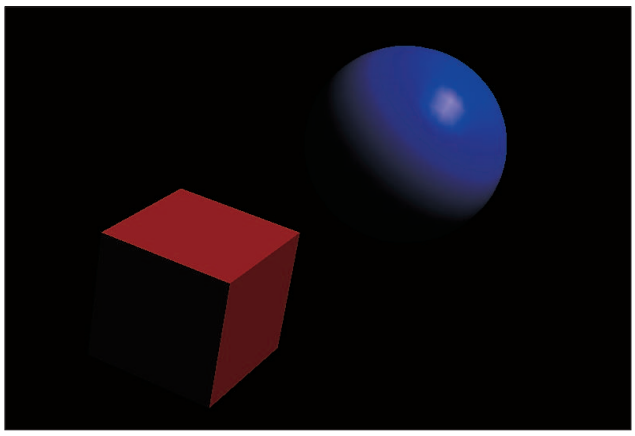
A PyOpenGL letöltése és telepítése

A PyOpenGL Python-modulokból álló készlet, mely hozzáférést ad az OpenGL-hez, csakúgy, akár egy sor más segédeszközhöz és OpenGL-kiegészítéshez, mellyel az OpenGL alacsony szintű felületét egészíthetjük ki. Az OpenGL-t eredetileg ***James Hugunin***, ***Thomas Schwaller*** és ***David Ascher*** fejlesztette ki. A program fejlesztését mostanában ***Tarn Burton*** vette át, a csomagot pedig ***Rene Liebscher*** és ***Michael Flatcher*** tartja karban.

Mivel az PyOpenGL szolgáltatásait nagyrészt OpenGL-hívások alkotják, ahhoz, hogy programozni tudj vele, az OpenGL alapszintű ismeretére lesz szükséged. Az OpenGL elsajátításához rengeteg ismertető és referencia létezik, ajánlásokat a cikk végén találsz hozzájuk.

A legelső követelmény a PyOpenGL-hez maga az OpenGL. Ha a gépeden még nincs működőképes OpenGL-változat, nézz utána, hogy GNU/Linux-terjesztésedben létezik-e hozzá csomag, vagy töltsd le a Mesa 3D grafikus könyvtárat a ➲ http://www.mesa.org címről. Hogy a PyOpenGL teljes gőzzel működhessen, szükség van még egy „Numerical Python" nevű könyvtárra is. A Numeric és a PyOpenGL forráskódja a ➲ http://numpy.sourceforge.net címről tölthető le. Telepítésük ***Greg Ward*** `distutils`-moduljának köszönhetően viszonylag egyszerű, ezt az 1.6-os változattól kezdődően megtaláljuk a Python-csomagban. Ha a kicsomagolt forráskód saját könyvtárából lefuttatjuk a `python setup.py install` parancsot, a parancs a modulokat összeállítja és telepíti. Mielőtt elindulnál beszerezni a csomagokat, győződj meg, hátha az általad használt GNU/Linux terjesztés is tartalmazza ezeket a programokat. Debianban például mindkét csomag megtalálható. Figyelem, a cikk megírásakor a PyOpenGL 1.5.7-es változatát használtam, de azóta már a 2.0-s változat is megjelent.

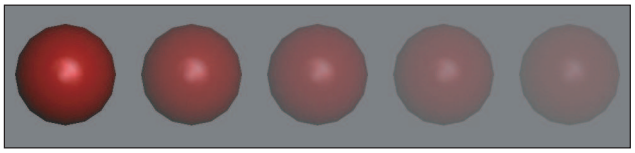
1. kép A 3. lista eredménye



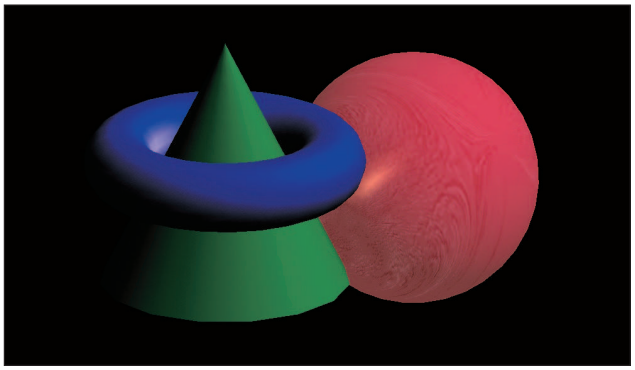
2. kép A 4. lista eredménye

Egy egyszerű Python OpenGL-alkalmazás

Az OpenGL megvalósítása nem határozza meg, hogy az OpenGL hogyan működjön együtt az ablakkezelő rendszerekkel. Ennek következtében az OpenGL-t használó programokban valamilyen külső GUI-eszköztárra is szükség van. Az első listában található program a GLUT-ot használja fel, mely az OpenGL-hez több felületen is elérhető. Hacsak nem valamilyen kereskedelmi OpenGL-változatot használsz, a GLUT minden bizonnyal rendelkezésre áll a rendszereden. Ez a kód egy ablakot nyit meg, beállítja a világítást és megjelenít egy teáskannát (1. lista ➲ http://www.linuxvilag.hu/Python). Eltekintve a Python írásmódjának egyszerűségétől, ez a program szinte ugyanúgy néz ki, mint C-beli megfelelője. Apró különbség a „display" szolgáltatás visszahívó függvényének beállítása. Ez C-ből vagy C++-ból a `glutDisplayFunc(display)` függvényhívással oldható meg. A visszahívó függvény beállítása az PyOpenGL-ből két lépésben történik: először meghívjuk a `glutSetDisplayFunc()` függvényt, ezt követően a `glutDisplayFunc()`-ot. Ez a sajátosság egyéb visszahívó függvényekre is jellemző, például a `glutMouseFunc()`-ra, vagy a `glutReshapeFunc()`-ra.



3. kép A fog.py a PyOpenGL része

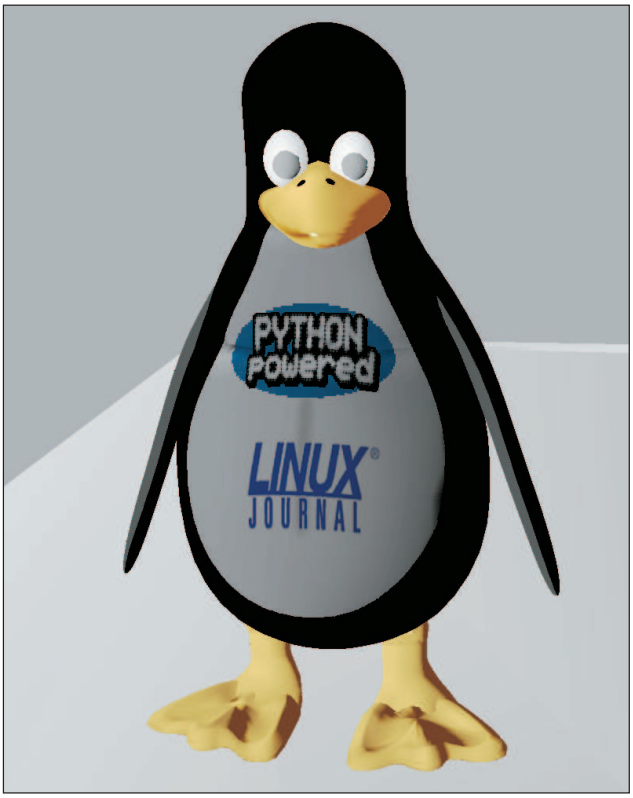


4. kép Egyszerű jelenet fénnyel és textúrával

A GLUT kisebb alkalmazásoknál nagyszerűen használható, ellenben meglehetősen sok munka szükséges ahhoz, hogy programjainkhoz alapvető és elvárt szolgáltatásokat adjunk hozzá, mint például egérrel vezérelt ráközelítések, kamerakövetés vagy forgatás. A Togl egy olyan Tkinter-összetevő, amely ezeket biztosítja, és emellett alapvető megvilágítást is beállít. A 2. lista (➲ http://www.linuxvilag.hu/Python) az előző programnak Togl felhasználásával készült változata.
Ez a program lényegesen rövidebb kódból áll, szolgáltatásaiban ugyanakkor felülmúlja az előzőt, de ennek ára a rugalmasság. Ha a Togl által biztosított fények és a felhasználófelület nem felel meg az igényeidnek, a Togl segítségével újratervezheted őket. A Togl prototípuskészítésre is tökéletes eszköz, mivel a melegítőlap megvilágítását és a navigálás megvalósítását is elvégzi helyettünk.
A PyOpenGL más 3D-s GUI-eszközkészletekkel is jól összevonható. Létezik kapcsolat wxWindowshoz, FLTK-hoz, FOX-hoz és GTK-hoz is.

Textúraleképezés használata

Textúraleképezésnek hívjuk azt a módszert, amellyel képadatot, mondjuk egy fényképet ragasztunk valamilyen sokszögre. Ha textúrákat szeretnénk használni, a képet először valamilyen OpenGL által támogatott alacsonyszintű formátumra át kell alakítani. Bár a Pythonban írt kód lényegében megegyezik az OpenGL alkalmazása során használt normál kóddal, a Python mégis nagymértékben leegyszerűsíti a textúrának használt képek betöltési és alkalmazási folyamatát.
A Python alapértelmezett könyvtára tartalmaz egy `rbgimg` nevű modult, mellyel az SGI ***imglib*** (***.rgb***) állományait olvashatjuk be. Bár elég homályos formátumnak tűnik, egyszerűségének köszönhetően könnyedén az alapkönyvtár része lehet, mivel nem sok helyet foglal. A Gimp vagy az ImageMagick segítségével bármilyen png, jpeg vagy tiff formátumú képet átalakíthatsz az SGI ***imglib*** formátumába. Az `rgblib.longimagedata(fÆjlnØv)` paranccsal egy teljes SGI imglib formátumú fájlt olvashatsz be, míg a függvény egy négybájtos RGBA képpontokat tartalmazó binárist karaktersorozat formátumba alakít át. Ez az adattípus már a `GL_RGBA` formátumkapcsoló és a `GL_UNSIGNED_BYTE` adattípus felhasználásával átadható az OpenGL függvényeinek



5. kép 3D Studio modell PyOpenGL-lel

(mint amilyen a `glTextImage2D` is).
Egy másik nagyon hasznos modul a Python alapkönyvtárából az ***imageop***. Ez a modul a képek vágásához, átméretezéséhez és szürkeárnyalatúra alakításához tartalmaz függvényeket. Az `imageop` függvények ugyanazon az `GL_RGBA/GL_UNSIGNED_BYTE` adattípuson dolgoznak, amellyel az `rgblib.longimagedata()` függvénye tér vissza.

A PyOpenGL kisegítő függvényei

Amellett, hogy a PyOpenGL tartalmazza az OpenGL függvényeit, még egy sor más, magasabb szintű függvényt is biztosít. Volt már szükséged arra, hogy egy OpenGL-jelenetből egyetlen állóképet merevíts ki magadnak? Természetesen bármikor lehetőséged van rá, hogy teljes képernyőképet (screenshot) készíts a képernyőről, mondjuk az xv-vel vagy a Gimppel, de mi történik olyankor, ha az OpenGL-jelenet egy olyan képkockájára van szükséged, amelyik csak bizonyos idő eltelte után jelenik meg, és csak egy pillanatra? A képernyőkép készítése akkor sem megoldás, ha olyan programról kell képet készítened, mely felhasználói beavatkozás nélkül fut, mint mondjuk egy CGI-parancsfájl. Az sem igazán kivitelezhető, hogy egy videó elkészítéséhez minden másodpercben egy sor képet készíts. Emiatt a PyOpenGL a képek kimerevítéséhez beépített függvényeket tartalmaz, amelyek azután többféle formátumban felhasználhatók. A *3. listában* található program annyival módosult az előző teáskannás programunkhoz képest, hogy a kép megjelenése után egy PostScript-pillanatképet készít róla, majd a lemezre mentve azonnal kilép a programból. Az *1. képen* magad is láthatod, programunk milyen képet készített.
A jelenetet a következő paranccsal mentettük:

```
openglutil.glSaveEPS( ex3.eps , 240, 240)
```

Az első érték annak a fájlnak a nevét adja meg, ahová menteni szeretnénk. A második és harmadik pedig rendre a szélesség és magasság értéket tartalmazza. A `glSavePPM()` függvényt ugyanezekkel az értékekkel meghívva ugyanúgy egy képet készítene, csak EPS helyett úgynevezett „portable pixmap" fájlformátumban. Ennek megfelelően ha a PyOpenGL tartalmaz tiff-támogatást, a képet a `glSaveTIFF` paranccsal tiff-be menthetnénk.
A PyOpenGL az OpenGL objektumkijelelő módszeréhez is kényelmes kezelőfelületet biztosít. Ezzel a kijelölő móddal egyértelműen megállapítható, hogy a kép egy adott régiójában mely objektumok találhatók. Ez annyit jelent, hogy a képen bármely pillanatban megjelölhető egy pont, ahol mondjuk az egérrel kattintottunk, és az OpenGL megmondja, hogy abban a pontban éppen melyik objektum található.
A PyOpenGL kijelölő művelete a következő módon állítható be:

1. Készíts függvényt vagy eljárást, amellyel a jelenet objektumait fogod kirajzolni, mindezt az OpenGL `glPushName()` és `glPopName()` függvényei között elhelyezve.
   A `glPushName()`-nek minden objektum kirajzolása előtt egy integer típusú számot adj meg, amellyel az OpenGL később hivatkozni tud az objektumaidra.
2. Állítsd be az egérkattintás eseménykezelőjét. Ennek beállítása attól függ, hogy milyen grafikus eszközkészletet használsz. Tkinter esetén például a `bind()` eljárásra van szükséged, ennek kell átadnod annak a függvénynek vagy eljárásnak a referenciáját, amelyet szeretnél, ha kattintáskor meghívódna.
3. Amint az egérrel kattintasz, az x és y koordinátákat az 1. pontban készített referenciával együtt add át az OpenGL `GL.glSelectWithCallback()` függvényének. Ez a függvény egy „tuple" típusú (közelség, távolság, nevek) elemeket tartalmazó listával tér vissza, melyből az első két érték az objektum mélységét jelöli, a harmadik pedig azon értékek a listája, melyet a `glPushName()` függvénnyel az első pontban megadtunk. Amennyiben a kérdéses helyen semmilyen objektum nincs, a függvény egy üres „tuple"-lal tér vissza.

A *4. listában* erre a kijelölő módszerre található példa. Próbáld ki, milyen hatást érsz el, ha a CONTROL gomb nyomva tartása mellett a bal egérgombot különböző helyeken nyomod le!
Ennek a feladatnak az elvégzésére a kijelölőmechanizmus a legegyszerűbb mód, de természetesen nem az egyetlen. Egy másik technika, ha az objektumokat egy háttér átmeneti tárban állítjuk össze, melyet a képernyőn nem jelenítünk meg. Ennek elvégzéséhez minden objektumot más színnel mentsünk, és a jelenetet kétdimenziós háttér átmeneti tárban állítsuk össze. A szín lekérdezésével megállapíthatjuk, milyen objektum található a képernyő adott részén. Ezt a háttér átmeneti tárat az OpenGL adagoló átmeneti tárának hívjuk.
A harmadik lehetőség az objektumok kijelölésére, ha a metszéspontok megadásával kézzel számítjuk ki, hogy egy pontban milyen objektum lehet. Ez annyit tesz, hogy el kell indítanod egy sugarat a képernyő valamely pontjáról, és meg kell határoznod, hogy a sugár mely objektumokat metszi. Amint megkaptad a metszett objektumok listáját, a mélységi tulajdonságokat figyelembe véve megállapíthatod, hogy melyik objektum található a legközelebb a képernyőhöz. Bár ez a megoldás jelenti a folyamat feletti legnagyobb irányítást, ennek a megoldásnak a kivitelezése egyúttal a legbonyolultabb is, és Pythont használva nem is a leghatékonyabb.

Teljesítménynövelés

Most, hogy már tudod, hogyan írj egyszerű OpenGL-programokat, bizonyára érdekel, hogy a méretezhetőség szempontjából a Python mennyire felel meg a 3D-s grafikus objektumok támasztotta igényeknek. Alapesetben a Python grafikus objektumainak sebessége természetesen csak kullog a C és a C++ nyomában, de megfelelő módszereket alkalmazva egészen szép eredményeket érhetünk el.
A legfőbb stratégia a sebesség növelésére, hogy minél jobban csökkentsük azt az időt, amelyet a kód a Python értelmezőjében tölt el, mindezt úgy, hogy az időigényes műveleteket natív kódba helyezzük át. Ennek elvégzéséhez minden lomha műveletet Python helyett C-ben vagy C++-ban írunk meg, így a lefordított kódnak nem kell a Python értelmezőjével vesződnie, viszont az eredmény a Pythonból is elérhető és felhasználható. Bár ezzel a megoldással jelentős sebességnövekedést érhetünk el, elveszítjük a Python használatából adódó egyszerűséget. Továbbá az is szükséges, hogy tudjuk a módját, hogyan írhatunk egy másik nyelven Pythonhoz kapcsolódó modulokat. Ugyanakkor ha az egészet C-ben írnád, nem kellene Python-bővítmények használatával vesződnöd.
Az OpenGL megjelenítési listái lehetővé teszik, hogy a sebességigényes dolgokat natív kódban írd meg, mindezt az előző megoldással járó számos gond elkerülésével. A megjelenítési listákkal egész OpenGL-műveletek helyezhetők az OpenGL gyorstárába, ahonnan az adatok egyenesen az OpenGL objektum-összeállító csővezetékébe haladnak tovább. Bizonyos körülmények között az OpenGL arra is képes, hogy megjelenítési listáit magán a grafikus kártyán tárolja, messze a Python gépezetétől.
A `glGenLists()` függvénnyel üres megjelenítési listákból álló tömböt hozhatunk létre. A függvény egyetlen értéket vár, annak számát, hogy hány megjelenítési listát hozzon létre. Visszatérési értéke a sikeresen létrehozott listák száma.
Az OpenGL `glNewList()` és `glEndList()` függvényeivel határolhatjuk be az elkészítendő listákat. Miután ezzel megvagyunk, az egyes listákat a `glCallList()` függvénnyel hívhatjuk újra elő. A PyOpenGL megjelenítési listák kezelésére szolgáló API-ja mondhatni teljesen megegyezik a hasonló feladatot ellátó C nyelvű kiterjesztéssel.

További lépések

Az eddigiekkel még csak épp hogy megérintettük a lehetőségek felszínét, melyet a Python és OpenGL eszköztára nyújt. További adatokért a PyOpenGL leírásához fordulj, melyet
➲ http://pyopengl.sourceforge.net/documentation/index.html
címen érhetsz el.

Jason Petrone
(jpetrone@acm.org) a CNRI (Nemzeti Kutatási Kezdeményezés) technikai munkatársa.
A Python és OpenGL erejét még előző munkahelyén ismerte fel, miközben egy CAVE virtuális valóság projekten dolgozott az NCSA-nál.

Könyvek

A Pythonban való OpenGL-programozásról egyelőre még nem jelent meg könyv, de az OpenGL általános programozásához3 az Addison-Wesley már megjelentetett egy írást, címe „The OpenGL Programming Guide". Mindemellett az OpenGL kézikönyv szintén nagyon jó könyv, melyet nem árt, ha magadnál tartasz.

Entitásbabok

Megtanuljuk, hogyan írjunk entitásbabokat, miképpen csatlakoztassuk őket adatbázishoz, illetve hogyan érjük el őket unokatestvéreiken, a sessionbabokon keresztül.

Az elmúlt hónapban bepillantottunk a Sun kiszolgálóoldali webalkalmazásokhoz szánt szabványának (J2EE – Java 2 Enterprise Edition) legfontosabb alkalmazásába, az Enterprise JavaBeans (EJB) világába. Bár sem a Java nyelv, sem a J2EE-szabvány nem nyílt, mégis egyre növekvő számban akadnak Linux-barátok, akik ezek segítségével készítenek kiszolgálóoldali webalkalmazásokat. Az igazság az, hogy úgy tűnik, a J2EE lesz az egyetlen használható lehetőség Microsoft's .NET keretrendszerrel szemben, ami sokak szemében minden bizonnyal még kívánatosabbá teszi.
Az Enterprise JavaBeans két alapvetően fontos – sessionbabnak és entitásbabnak nevezett – elemet tartalmaz. A sessionbabok többnyire folyamatokat modelleznek és nincs állapotuk (lack any state), és lehetővé teszik, hogy az üzleti logikát a JavaServer Pages (azaz JSP) helyett az EJB-be vigyük be. A múlt hónapban tervezett számológép, amely két számot tudott összeszorozni, az egy eljárással rendelkező sessionbabok nagyon egyszerű példája volt. Az entitásbabok ezzel szemben állapottal rendelkeznek, néha akár meglehetősen összetett állapottal. Ez az állapot többnyire egy relációs adatbázis-kezelő táblázatának egyik sorát jelenti, ahol vagy a bab, vagy a saját beépített objektumrelációs átalakító megoldását használjuk (bean-managed persistence, azaz BMP), vagy pedig az EJB-re hagyjuk ennek a feladatnak a megoldását (container-managed persistence – CMP). Az EJB-tároló tranzakciókat is támogat, lehetőséget adva objektumainkon, illetve a nekik megfelelő adatbázison a mindent vagy semmit típusú műveletekre.
Ebben a hónapban egyszerű entitásbabot fogunk írni, adatbázishoz csatlakoztatjuk, majd Java-alkalmazásból sessionbabokon keresztül megpróbáljuk elérni. Ehhez a nyílt forráskódú JBoss EJB-tárolót fogjuk felhasználni (ez a GNU Lesser General Public License, azaz LGPL szabadalom alá tartozik), de kódunk kis módosításokkal bármely EJB-t támogató J2EE-kiszolgálón futni fog.

Készítsünk entitásbabokat!

Amint a múlt hónapban is láthattuk, a sessionbab írása tulajdonképpen három Java-osztály létrehozását jelenti:

- a tényleges feladatot elvégző babosztályét,
- a távoli csatolófelületét, amely a babosztály eljárásainak megfelelő tagfüggvényeket tartalmaz, és tulajdonképpen átjárót (proxy) képez a babunkhoz,
- annak a saját csatolófelületnek létrehozását, amelynek segítségével a babból új példányokat hozhatunk létre, illetve különféle feltételeknek megfelelő babokat kereshetünk ki.

Az entitásbabok esetében ugyanígy mindhárom osztályt létre kell hoznunk. Ezenkívül azonban gyakran egy negyedik, úgynevezett „elsődleges kulcs" osztályt is el kell készítenünk. Bár e havi példánkban nem lesz ilyen elsődlegeskulcs-osztályra szükségünk, az érdekesség és a teljesség kedvéért létre fogjuk hozni.
A legtöbb EJB-alkalmazás legalább egy entitásbabot (az adatmodellezéshez) és legalább egy sessionbabot tartalmaz (az üzleti logika megvalósításához). Minthogy az objektumközpontú programozás alapötlete épp az, hogy az adatokat és a kódot egyetlen csomagban kezeli, az entitás- és a sessionbabok ilyen módon történő szétválasztása kicsit furcsának tűnhet. Ennek ellenére ez a módszer mégis működőképes, és ha egyszer megállapodtunk az alapelvekben, meglehetősen leegyszerűsíti a munka megosztását.

Munka a JAWS-zal

A J2EE egy szabvány, melynek tényleges megvalósítása attól függ, hogy ténylegesen ki írta a kiszolgálót. A J2EE alkalmazáskiszolgáló leglényegesebb részeinek egyike az objektum-relációs átalakító, amely a Java-osztályokat átlátszó módon alakítja át relációs adatbázistábla-sorokká (és fordítva). Ennek az objektumrelációs átalakítónak a lehető legrejtettebbnek kell maradnia, hiszen csak így lehetséges, hogy a háttértárat például Oracle-ről a Java-kód megváltoztatása nélkül egyszerűen MySQL-re váltsuk.
A JBoss objektumrelációs átalakítórendszer (JAWS) általában alig igényel beállítást. Ennek ellenére hasznos lehet a JAWS beállításfájlját végignézni (***standardjaws.xml*** a JBoss ***conf/default*** könyvtárban), hogy tisztában legyünk vele, tulajdonképpen mi zajlik a háttérben.
A ***standardjaws.xml*** elején található meghatározások a teljes JBoss-kiszolgálóra vonatkozó értékeket állítanak be. Itt adhatjuk meg, hogy melyik adatbázis-kezelőt szeretnénk használni. A HyperSonic adatbázis-kezelő a JBosszal együtt érkezik, és a következő példákban mi is ezt fogjuk használni.
A ***standardjaws.xml*** magját többszörös `<type-mapping>` (egytagú) részek alkotják, amelyek az adatbázisok mindegyike esetében az összes `<java-type>`-elemet egy `<jdbc-type>`- és `<sql-type>`-elemhez kapcsolják. Mivel EJB-nk nem készít táblákat és közvetlenül SQL-t sem ír, nagyon fontos, hogy ezeket az értékeket pontosan állítsuk be. Beállításukkal növelhetjük a programunk hatékonyságát és rugalmasságát.
Ne feledjük azonban, ha a JAWS-t azután módosítjuk, miután már adatokat tettünk az adatbázisba, az könnyen zavarhoz, károsodáshoz vagy adatvesztéshez vezethet.
Ha az EJB-t egyszerűen csak ki akarjuk próbálni, akkor egyáltalán nem kell a ***standardjaws.xml***-t módosítanunk. Inkább írjuk át a ***jboss.jcml***-t, ugyanis ez az XML-fájl adja meg azokat a kezelőbabokat (managed beans avagy MBeans), amelyeket a JBoss a rendszer beállítására és irányítására használ.
A ***jboss.jcml*** fájl HyperSonic- és InstantDB-támogatást is tartalmaz. Ahhoz viszont, hogy HyperSonickal is működjön,

előbb az összes InstantDB-vel kapcsolatos bejegyzést el kellett távolítanom a ***jboss.jcml***-ből. Ezt a ***jboss.jcml*** „JDBC” szakaszának módosításával tehetjük meg: töröljük a `JdbcProvider` Mbeanhez tartozó ***Drivers*** részből az `org.enhydra.instantdb.jdbc.idbDriver` bejegyzést, majd a teljes `XADataSourceLoader <mbean>` szakaszt, illetve minden `XADataSource` típusú vagy InstantDB nevű szolgáltatást.
Ha a ***jboss.jcml***-ből már minden InstantDB-vel kapcsolatos sort töröltünk, indítsuk el a JBosst:

```
cd $JBOSS_DIST/bin
sh run.sh
```

Entitásbabunk megírása

Soron következő entitásbabunk egy egyszerű könyvesboltot fog modellezni, ahol minden egyes könyv címmel, szerzővel, kiadóval és árral rendelkezik. Az egyszerűség és a tömörség kedvéért most eltekintünk attól, hogy egy könyvnek több szerzője és kiadója is lehet. Továbbá az adatokat normalizálni sem fogjuk, ami egyébként azt jelentené, hogy olyan példányváltozóink lennének, amelyek maguk is entitásbabok.
A `BookBean`-osztály megírásával kezdünk, amit az *1. listá*-ban is olvashatunk, illetve a 28. CD Magazin/Beans könyvtárában is meglelhetünk. A BookBean jó példa az egyszerű tárolókezelésű babosztály meghatározására; minden egyes nyomon követendő adatbázisoszlophoz megad egy-egy mezőt, ideértve az `id` egész típusú mezőt is, amely elsődleges kulcsként szolgál.
Meg kell adnunk egy `ejbCreate()` tagfüggvényt, ami a saját csatolófelület `create()` tagfüggvényének felel meg. Valahányszor csak meghívjuk a saját csatolófelületünk `create()` tagfüggvényét, az EJB-tároló az `ejbCreate()`-et adott értékekkel meghívja a babosztályunkra. Tulajdonképpen az `ejbCreate()`-ben megy végbe a valódi létrehozatal; a CMP entitásbabnak egyáltalán nem kell az objektumrelációs átalakítással törődnie, viszont megfelelő értékűre kell beállítania a példányváltozókat.
Az ejbCreate()-en kívül már csak a „leszedő” és „feltöltő” tagfüggvényeket kell minden egyes mezőhöz létrehoznunk, hogy objektumaink a mezők tartalmát le tudják kérdezni, illetve módosítani tudják. Példánkban természetesen minden tagfüggvény meglehetősen egyszerű, mindössze módosítja vagy visszaadja egy példányváltozó értékét.
A távoli csatolófelület (amit a 2. *listában* találunk meg) neve *Book.java*, és felülete csaknem teljesen megegyezik a babosztályával. Az alkalmazások többnyire a távoli csatolófelülettel beszélgetnek, és ha valami gond merül fel, egy `RemoteException`-t dobnak.
Meg kell adnunk egy `create()` tagfüggvényt tartalmazó saját csatolófelületet is (ezt a 3. *listában* találjuk), amely a `Book` objektumpéldányok létrehozására szolgál (illetve közvetett módon egyúttal adatbázistáblánkban is új sorokat hoz létre), ha a könyv összes adatát átadjuk neki. Amennyiben elég lelkesek lennénk, többfajta `create()` függvényt is felajánlhatnunk, egyet-egyet minden értékszámhoz.
Figyeljük meg, hogy `create()` tagfüggvényünknek az elsődleges kulcsot közvetlenül kell megadnunk. A tapasztalt adatbázis-programozók tudják, hogy az elsődleges kulcsokat lehetőség szerint nem szabad szem előtt hagyni, és ennek önműködővé tételére a legtöbb adatbázis-kezelő lehetőséget is ad; a PostgreSQL ***SERIAL***-típusa, a MySQL ***AUTO INCREMENT***, és az Oracle-szekvenciák a szokásos megoldások erre a feladatra. Sajnos nincs rá igazán egyszerű lehetőség, hogy ezeket az önműködően előállított elsődleges kulcsokat EJB-ből kihasználjuk. Így aztán vagy közvetlenül meg kell adnunk őket (ahogyan az e havi példában is tettük), vagy egy külső értéket kell használnunk, például az ISBN-számot, amely karaktersorozat is lehet. E szolgáltatás hiánya döbbentett meg engem a leginkább az EJB-ben, és remélem, a szabvány következő változatai már orvosolni fogják a hibát.
A `findBy...()` tagfüggvény segítségével megkereshetjük és lekérhetjük a `Book` egyes példányait.
A `findByPrimaryKey(5)` például azt a `Book`-objektumot fogja visszaadni, amelynek elsődlegeskulcs-értéke 5. Az összes `findBy...()` tagfüggvény az EJB-tárolóban már meg van határozva, így nekünk már nem kell megtennünk.
A `findAll()` tagfüggvény az összes ilyen típusú objektum egy csoportját adja vissza (azaz az adatbázistábla összes sorát), így végiglépkedhetünk rajtuk.
A beépített `findAll...()` tagfüggvény sajnálatos módon

2. lista Távoli csatolófelületünk, a Book.java forráskódja – egyszerűen csak tükrözi a BookBean-osztályunkban meghatározott tagfüggvényeket

```
package il.co.lerner.book;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;

/* A Book entitÆsbab tÆvoli csatol fel lete.
   A BookBean minden tagf ggvØnyØhez
   ➥megadunk egy megegyezı jelzØsß
   ➥tagf ggvØnyt . */

public interface Book extends EJBObject
{
   public int getId() throws RemoteException;
   public void setId(int newId) throws
      ➥RemoteException;

   public String getTitle() throws
   ➥RemoteException;
   public void setTitle(String newTitle)
      ➥throws RemoteException;

   public String getAuthor() throws
      ➥RemoteException;
   public void setAuthor(String newAuthor)
      ➥throws RemoteException;

   public String getPublisher() throws
      ➥RemoteException;
   public void setPublisher(String
   ➥newPublisher)
      ➥throws RemoteException;

   public double getUsDollarPrice()
      ➥throws RemoteException;
   public void setUsDollarPrice
   ➥(double newDollarPrice)
      throws RemoteException;
}
```

3. lista Saját csatolófelület megadása

```
package il.co.lerner.book;

import javax.ejb.EJBHome;
import javax.ejb.CreateException;
import javax.ejb.FinderException;
import java.rmi.RemoteException;
import java.util.Collection;

/* Ez az osztÆly hatÆrozza meg Book
/* entitÆsbabunk
➥sajÆt csatol fel letØt. */

public interface BookHome extends EJBHome
{
    /* j Book-pØldÆny kØsz tØse */
    public Book create(int newId, String
    ➥newTitle,
                  String newAuthor, String
                  newPublisher,
                  double newUsDollarPrice)
      throws RemoteException, CreateException;

    /* Keress k meg az adott azonos t jœ
    /* Book-pØldÆnyt. Az EJB-
    tÆrol  ezt a tagf ggvØnyt  nmßk dıen
    ➥hozza lØtre helyett nk. */

    public Book findByPrimaryKey (int id)
      throws RemoteException, FinderException;

    /* Egy k nyvgyßjtemØny visszaadÆsa,
    /* ahol
    a szerzı neve megegyezik. Ezt a
    ➥tagf ggvØnyt
az EJB-tÆrol   nmßk dıen hozza lØtre. */

    public Collection findByAuthor
    ➥(String authorName)
      throws RemoteException,
      ➥FinderException;

    /* Az adatbÆzis  sszes
    /* Book-objektumÆnak visszaadÆsa */
    public Collection findAll()
        throws RemoteException,
FinderException;
}
```

egyszerű egyenlőségvizsgálatot végez. Nem tudunk például szabványos kifejezések szerint keresni vagy más módszereket alkalmazni, például kikeresni az összes O betűvel kezdődő könyvcímet, vagy azokat a könyveket, amelyeknél a kiadó neve A betűvel kezdődik és M-re végződik. Ehelyett a teljes készletet kénytelenek vagyunk a `findAll()` használatával végigpörgetni, és a szükséges szűréseket magunk elvégezni (azaz pontosan azt veszítjük el, amiért érdemes adatbázis-kezelőt használni ezzel az erővel akár egy fájlban is lehetne – a ford.).
Végül a *4. listában* található elsődlegeskulcs-osztályunk (***BookPK.java***) egyetlen példányváltozót vezetett be (`id`),

4. lista BookPK.java, az entitásbabunkhoz tartozó elsődlegeskulcs-osztály

```
package il.co.lerner.book;

import java.io.Serializable;

public class BookPK implements
java.io.Serializable
{
    public int id;
    public BookPK () { }

    public BookPK(int value)
    {
        id = value;
    }

    public boolean equals(Object obj)
    {
        if (obj == null ||
        ➥!(obj instanceof BookPK))
            return false;

        if (((BookPK)obj).id == id)
            return true;

        return false;
    }

    public int hashCode ()
    {
        return id;
    }


    public String toString()
    {
        return String.valueOf(id);
    }
}
```

amelyet elsődleges kulcsként használunk. Az `equals()` tagfüggvény azt mutatja meg, hogy két `BookPK`-példány azonosnak tekinthető-e, ezáltal a rendszer képes lesz összehasonlítani két `Book`-példányt. A `hashCode()` tagfüggvénynek minden példányhoz egyedi azonosítót kell visszaadnia, amelyet aztán az adott esetben kulcsként használhatunk fel.
A `toString()` tagfüggvény feladata, hogy az elődleges kulcs karakteres értékét, azaz egyszerűen az osztályunk `String.valueOf(id)` értékét adja vissza.
Minthogy mind a négy fent említett osztály az ***il.co.lerner.book*** csomagban található, mind a négy forrásfájlt (***Book.java***, ***BookBean.java***, ***BookHome.java*** és ***BookPK.java***) a ***$BOOK/il/co/lerner/book*** könyvtárba helyeztem, ahol a BOOK projekt a saját könyvtárat jelenti.

Telepítésleíró

Most, hogy az entitásbabot elkészítettük, itt az ideje, hogy az EJB-tároló számára le is írjuk. Telepítésleírónk egy ***ejb-jar.xml*** nevű fájl – ez látható az *5. listában*. Az ***ejb-jar.xml***-t a

5. lista Az ejb-jar.xml – az entitásbabunkhoz tartozó telepítésleíró

```
<?xml version="1.0" encoding="UTF-8"?>

<ejb-jar>
  <description>ATF Book Bean</description>
  <display-name>ATF Book</display-name>
  <enterprise-beans>
     <entity>
        <description>Models a book</description>
        <ejb-name>Book</ejb-name>
        <home>il.co.lerner.book.BookHome</home>
        <remote>il.co.lerner.book.Book</remote>
        <ejb-class>il.co.lerner.book.BookBean
             </ejb-class>

        <persistence-type>Container</persistence-type>
        <reentrant>False</reentrant>

        <prim-key-class>il.co.lerner.book.BookPK
             </prim-key-class>

        <cmp-field><field-name>id</field-name>
             </cmp-field>
        <cmp-field><field-name>author</field-name>
             </cmp-field>
        <cmp-field><field-name>title</field-name>
             </cmp-field>
        <cmp-field><field-name>publisher</field-name>
             </cmp-field>
        <cmp-field><field-name>usDollarPrice
             </field-name></cmp-field>
     </entity>
  </enterprise-beans>
</ejb-jar>
```

$BOOK/il/co/lerner/book/ könyvtárba helyezzük, azok mellé a Java-osztályok mellé, amelyek majd az entitásbabunkat fogják alkotni. Amikor a babot az Anttal felépítjük, a ***META-INF*** alkönyvtárba fog kerülni.

Az ***ejb-jar.xml*** entitásbabokra vonatkozó legérdekesebb része a `<persistence-type>` szakasz (CMP esetén ***Container***-re kell állítani), a `<prim-key-field>`- és `<prim-key-class>` szakaszok (ahol elsődleges kulcsaink osztályát nevezhetjük meg), és a `<cmp-field>` szakasz (amely a tárolóvezérelt mezőket írja le a JBossnak).

A telepítésleíró az EJB szabványos része, és minden EJB-kiszolgálóval és- tárolóval működnie kell. Nem adja meg azonban az összes futásidejű beállítási lehetőséget, ezért aztán a JBoss helyes működéséhez egy ***jboss.xml*** nevű fájlt is be kell fűznünk, amely a kiszolgálónak megmondja, hogy a babokat a hálózaton hol találjuk meg. A ***jboss.xml*** másolata, amit az ***ejb-jar.xml*** mellé a ***$BOOK/il/co/lerner/book/***-ba kell helyeznünk, mely folyamat az alábbi listán látható:

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss>
  <enterprise-beans>
    <entity>
      <ejb-name>Book</ejb-name>
      <jndi-name>Book</jndi-name>
    </entity>
  </enterprise-beans>
</jboss>
```

Alkalmazás

Egyszerű próbaalkalmazásunk (***UseBook.java***) forráskódja a 6. listában látható a 28. CD Magazin/Beans könyvtárában. Szépen bizonyítja, hogy mégoly kevés kóddal is milyen sokat el lehet érni. Csak a `main()` tagfüggvényt határozza meg, és máris az EJB-kel kezd dolgozni. Először elfogja a JNDI-környezetet és felkeresi az általunk megadott `Book` babot. Ezáltal lehetővé válik egy `BookHome`-példány létrehozása, s következő lépésben már új `Book` babot hozhatunk létre:

```
Book book =
home.create(testPrimaryKey,
   ➥"Book title",
                    ➥"AuthorFirst
AuthorLast",

➥"PublisherName", 10.50);*
```

Amint láthatjuk, az adatbázishoz adandó értékeket beégettük a kódba, az elsődleges kulcsot is beleértve. Ez egy valós alkalmazás esetében nyilvánvalóan elfogadhatatlan lenne. Egy valódi alkalmazás az elsődleges kulcsot (és más értékeket) egy fájlból, a parancssorból, a környezeti változóból, egy webes HTML-űrlapból venné, vagy éppen önműködően készítene egyet.

Figyeljük meg, hogy egyszer sem kellett SQL-lekérdezésket vagy adatbázis-lekérdezéseket beszúrnunk. Háttértárolónk feltételezhetően relációs adatbázis, de Java ügyfélprogramunk erről nem tud, és nem is számít neki.

Miután beszúrt egy új sort a táblába, az UseBook megváltoztatja néhány értékét (a példányváltozók értékeinek az átírásával), majd az adatbázisban található összes `Book`-példányt lekérdezi (a `findAll()` segítségével). A teljes futás alatt állapotüzeneteket ír a ***System.out***-ra, amelyek a konzolra íródnak.

Az Ant beállításfájl

Programunk fordításához és telepítéséhez az Antot használjuk, a hagyományos make program Java-megfelelőjét.

A 7. lista (elérhető a 28. CD Mgazin/Beans könyvtárában) a felhasznált ***build.xml*** fájlunkat mutatja be, amely a ***$BOOK/il/co/lerner/book*** könyvtárban az összes ***.java*** forrásfájlt lefordítja, majd a telepítésleíróval és a JBoss futásidejű beállításfájljával egyetemben Java ***.jar*** fájllá alakítja őket, végül a fájlokat a JBOSS könyvtárba telepíti. Ha az Antot az ***ANT***-ba telepítettük, a következő paranccsal az összes fájlt lefordíthatjuk, majd a JBoss telepítési könyvtárába telepíthetjük, és elindíthatjuk az `UseBook.main()`-t:

```
$ANT/bin/ant use-book-ejb
```

Amint a JBoss felfigyel az új (vagy frissített) ***.jar*** fájlra, a kime-

netet a kiszolgáló tevékenységét mutató képernyőn már láthatjuk is. Az Ant kimenete ezzel szemben azokat az elemeket fogja mutatni, amelyeket a `main()`-ből küldtünk a `System.out`-ra – a 6. listában olvasható állapotüzenetek mutatják, mit is csináltunk. Valahányszor csak a `main()`-t új `ID` értékkel fordítjuk le és futtatjuk, mindig egy-egy új sor szúródik be az adatbázistáblába.

Az EJB a jövő?

A Sun azt szeretné, ha azt hinnénk, hogy az EJB minden kiszolgálóoldali alkalmazás jövője. A Microsoft természetesen mindent megtesz, hogy azt bizonyítsa be: a .NET a valódi jövő. Hogy igazodjék el egy magányos fejlesztő az óriások csatájában, és ugyan mit tehetnek a szabad programok hívei? A jó hír az, hogy a J2EE kitűnő szerkezettel és filozófiával rendelkezik. Nem éppen a legelegánsabb és nem is a legrugalmasabb, sőt, egyetlen nyelvhez láncol, általánosságban azonban jó benyomást tett rám. A J2EE láthatóan fontos mérföldkő a webalkalmazások programozása terén.

Sajnálatos módon számos olyan gond van a J2EE-ben, amelyekkel mindig szembekerülök, valahányszor csak megpróbálok benne alkalmazást írni. Az első, hogy sem a Java, sem a J2EE nem nyílt forrású, annak ellenére sem, hogy a Java ingyenes és a JBoss LGPL felhasználási szerződésű. A Sun általában becsületesen játszik, de mégiscsak a haszonelv alapján működő vállalat, amelynek megvannak a saját érdekei. A nyílt forrás híveit könnyen érheti az a meglepetés, hogy a Sun egyszercsak korlátozni kezdi a kód vagy a szabvány használatát.

Továbbá úgy tűnik, hogy az Enhydra Enterprise-részleg végleg eltűnik, és a JBoss marad az egyetlen nyílt forrású J2EE-alkalmazáskiszolgáló a piacon. A JBoss nem rendelkezik a hivatalos J2EE-alkalmas minősítéssel, igaz, inkább csak azért, mert a Jboss-csapat nem tudja a hivatalos minősítést megfizetni. Ez rámutat, milyen rövidlátó is a Sun; nyugodtan felajánlhatnának egy olcsó vagy ingyenes minősítést a GPL vagy LGPL felhasználási szerződésű kiszolgálóknak, hiszen (a Berkeley felhasználási szerződéssel szemben) a (L)GPL szavatolja, hogy a minősített kiszolgálót semmilyen cég sem alakíthatja kereskedelmi programmá. A hivatalos J2EE-minősítés nekem ugyan nem sokat számít, de számos döntéshozó számára fontos dolog, és ez azt jelenti, hogy a Jbosst gyakran érdemtelenül hagyják figyelmen kívül.

A Java és az EJB elég összetett, és bizony időbe telik, míg egy programozó megtanulja használni őket. De tapasztalataim szerint a Java- és az EJB-programozás eltörpül ama munkamennyiség mellett, amit a hihetetlen mennyiségű beállításfájl, az új meghatározások, a környezeti változók és más, csak a Javára jellemző dolgok megtanulása jelent. Egy jobb leírás nyilván segítene, ami azonban igazán hiányzik nekem, az a CPAN javás megfelelője, amely a kiszolgálóoldali Java-beállításokat könnyíthetné meg, és így lehetővé válna, hogy a programozók rendszerkarbantartási feladatok helyett inkább a programozásra összpontosítsanak.

A .NET egyetlen olyan jellemzője, amit tényleg érdekesnek találok, az viszonylag nagy nyitottsága a különféle programozási nyelvek irányában. Alapértelmezés szerint a J2EE mindenképpen Javát igényel, ami gyakran valóban a helyes választás, de nem kellene, hogy az egyetlen választás legyen. A SOAP és az XML-RPC lehetővé teheti a nyelvek közötti szakadék áthidalását – de a kellemes tranzakciókezelés és objektumrelációs modell nélkül, amit az EJB hozott meg számunkra. Jelenleg úgy tűnik, az egyetlen lehetőség a Python és az EJB közti párbeszédre a SOAP vagy XML-RPC (avagy Jython), de szívesen látnék újabb lehetőségeket is a közeljövőben.

Összefoglalás

Az EJB lenyűgöző módszer, és sokkal többre képes, az olyan egyszerű objektumrelációs átalakítóknál, mint az Alzabo vagy a DODS. Tapasztalataim szerint az EJB-vel való munka sokkal inkább szervezési és logisztikai feladat, semmint technikai. Az EJB elsajátítása mindenképpen jó ötlet minden webalkalmazás-fejlesztő számára; nyilvánvaló, hogy ez a szabvány jelentős mértékben terjedni fog az iparban, és számos komoly alkalmazást EJB-ben fognak írni a jövőben. A minősített nyílt forrású megoldások a programozóknak még könnyebbé tennék az EJB kipróbálását, és javasolnám a Sunnak, hogy amint lehetséges, induljon el ebben az irányban.

A következő hónapban sebességet váltunk és a Zope-ot kezdjük vizsgálni, ezt a nagyrészt Pythonban íródott és az eddigiektől merőben eltérő webalkalmazás-keretrendszer. A Zope az utóbbi pár évben meglehetős népszerűségre tett szert, és gyakran ez az alkalmazás az, amely a Pythont a programnyelvek élvonalába emeli. Érdemes tehát egy pillantást vetnünk a Zope-ra és megvizsgálnunk, hogy mi módon használhatnánk fel saját alkalmazások írására.

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 JBoss hivatalos lapja a ➲ http://www.jboss.org. A honlapon fórumokat, levelezési listákat, letöltési lapokat és leírást is találhatunk. A Jboss-leírás néhol kicsit zavaros, nem mindig részletezi a dolgokat, és elég komoly EJB-hez kapcsolódó tudásbázist tételez fel. Ugyanakkor majdnem mindent tartalmaz, amire csak szükségünk lehet, és kellemesen egyszerű nyelven íródott, illetve néhány jól használható példakódot is tartalmaz.

Kitűnő bemutatkozó írás EJB-témában *Richard Monson-Haefel* könyve, az Enterprise JavaBeans, amely az O'Reilly kiadásában jelent meg. A ➲ http://www.jboss.org példáival és útmutatójával együtt ez a könyv lehetővé teszi, hogy néhány egyszerű EJB-t viszonylag rövid időn belül el tudjunk készíteni.

Az Ant – a Java válasza a `make`-re és a `Makefile`-okra – az Apache Software Foundation Jakarta Project része. Letölthető a ➲ http://jakarta.apache.org/ant címről.

A Sun honlapja a ➲ http://java.sun.com, igen sok adatot tartalmaz a J2EE-szabványról, az Enterprise JavaBeansről, illetve a kiszolgálóoldali Javáról.

Kiszolgálók teljesítményének növelése

Hogyan növeld kiszolgálód teljesítményét intelligens hálózati kártyára Linux-alapú kiszolgálóról a TCP/IP-verem áthárításával.

Növelni szeretnéd felsőosztályú Linux-kiszolgálód teljesítményét? Rendszered nagy sebességű hálózathoz kapcsolódik? Kiszolgálóidnak túl sok erőforrását emészti fel a TCP/IP-verem és az ethernetkeretek kezelése? Ezek napjaink legáltalánosabb hálózattal összefüggő gondjai, melyet a TCP/IP-forgalomnak az elmúlt évtizedben történt drámai növekedése okoz mind az Interneten, mind a nagyvállalati hálózatokon, és ez a forgalom nagy valószínűséggel csak emelkedni fog. Az Internet világméretű növekedése és az új hálózati adattároló módszerek (például iSCSI) következtében a forgalom és a sebesség még inkább nőni fog. Bár a processzorok elképesztő iramban fejlődnek, a hálózat fejlődése még ezt is túlszárnyalja, rákényszerítve a processzorokat, hogy erőforrásaikat elsődleges feladataik helyett a hálózati csomagok kezelésére pazarolják. Az Intel egy olyan intelligens hálózati kártya (network interface card – NIC) prototípust fejlesztett ki, amellyel a TCP/IP-verem a Linux-kiszolgálóról teljes egészében erre az iNIC-re irányítható.

Az iNIC felépítése

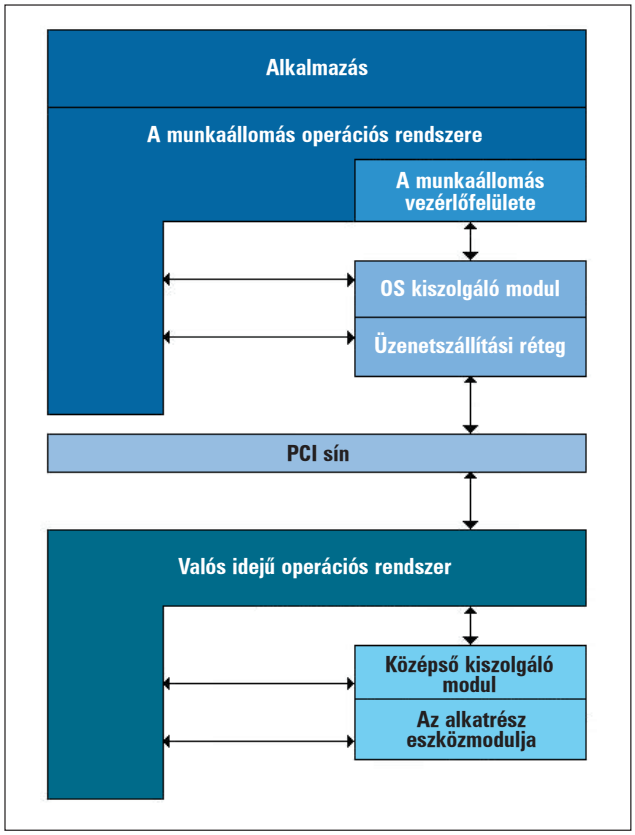
A hálózati kártyán valós idejű operációs rendszer (Real-Time OS – RTOS) fut teljes 4-es változatú TCP/IP-támogatással. Az iNIC-en a hálózati csomagokat egy I/O processzor (IOP) dolgozza fel a gazdaprocesszor tehermentesítését biztosítva. A felosztás elvégzéséhez a gazdagép részéről a legcsekélyebb logika szükségeltetik, hogy a hálózati csomagokat átirányítsa az iNIC-re.

Foglalatok áthárítása

Ez a módszer az intelligens I/O (I2O) szerkezetre épül, amelyet a Linux 2.4-es változata már támogat. Az *1. ábra* az I2O alapfelépítését tartalmazza. Az I2O-megvalósítás üzenetközvetítő módszer a gazda operációs rendszer (OS) és az IOP-on található I/O-eszközök között. Az IOP-on egy intelligens RTOS (IRTOS) fut, mely a kapcsolódó eszközök eszközmeghajtó moduljait (DDM-ek) tartalmazza. A hordozhatóság végett az I2O osztott eszközmeghajtó modellt használ. A megvalósítás minden eszközcsoporthoz alapvető üzenetrendszert határoz meg (pl: helyi hálózat, szalag, lemez). A gazda operációs rendszer is egy előre meghatározott üzenetrendszer alapján tartja a kapcsolatot az IOP-on található eszközmeghajtókkal. Ezeket az I2O-üzeneteket az eszközmeghajtó fordítja le alkatrészjellemző parancsokra. A gazdagépnek egy I2O-eszközzel való kapcsolattartásához eszközmeghajtóval kell rendelkeznie, amely a gazda operációs rendszer eszközparancsait I2O osztályszintű parancsokra fordítja le. Ezt a modult a gazdagépen operációsrendszer-modulnak (OSM) nevezik.

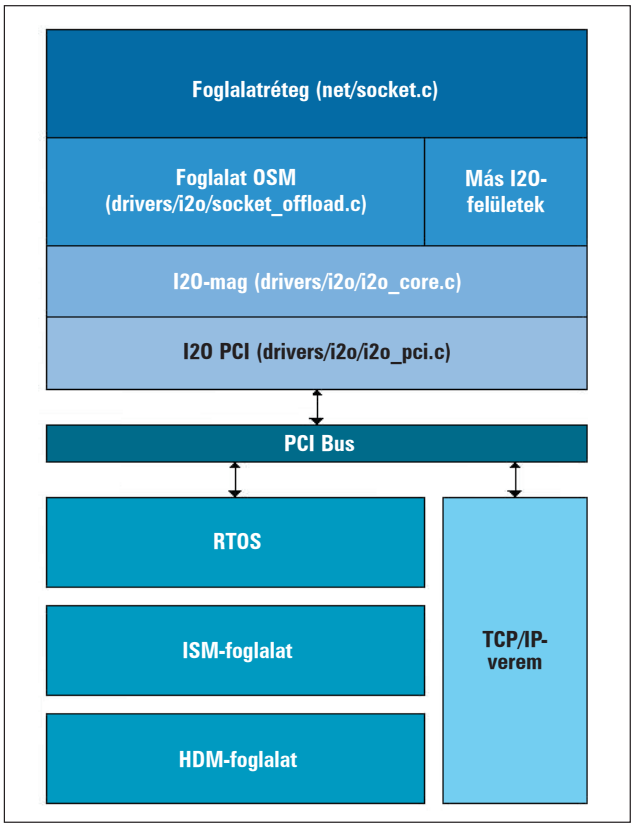
A megoldás az architektúra egy kiterjesztésére épül egy foglalati (socket) osztály létrehozásával. A teljesítménynövelés és a késés elkerülése érdekében – melyet többnyire az osztott eszközmeghajtó modell okoz – az I2O felépítésén változásokat hajtottak végre.

A foglalati osztály határozza meg a gazda operációs rendszer és az eszközmeghajtó kapcsolattartásához szükséges üzeneteket. A két meghajtó, az OSM és a DDM, azaz az operációsrendszer-modul és az eszközmeghajtó modul kétszintű üzenetközvetítő csatornán keresztül tartja a kapcsolatot. Az üzeneti réteg indítja el a kapcsolattartási folyamatot, a szállítási réteg pedig meghatározza, hogy az üzenetek hová menjenek. A DDM két almodulból épül fel: a középfokú szolgáltatás modulból (ISM) és a

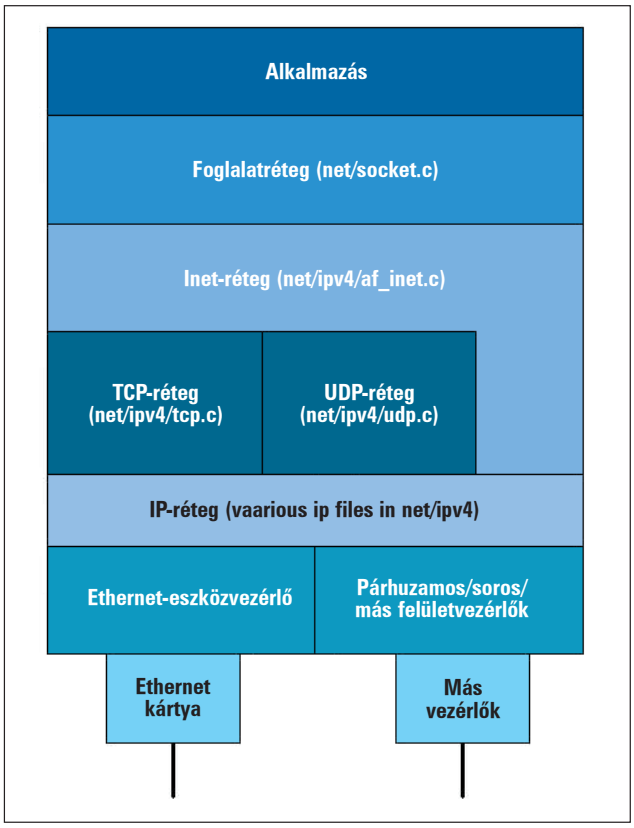


1. ábra I2O-szerkezet

gépi eszközmodulból (HDM). A 4-es változatú TCP/IP-vermet teljes egészében az ISM szolgáltatja. A HDM az iNIC eszközmeghajtója. A foglalati OSM semmilyen más Linux-eszközmeghajtóhoz nem hasonlítható. A normál hálózati kártyák meghajtói protokollfüggetlenek és a Linux-rendszermaggal a hálózati alkalmazások programozói felületén keresztül tartják a kapcsolatot. A foglalati OSM ezzel ellentétben közvetlenül a programozói felület alatti szinttel tartja a kapcsolatot – ezáltal válik lehetővé, hogy a szükséges foglalati szolgáltatások az IOP-hoz kerüljenek, amelyen a foglalati szintű ISM fut. A foglalati OSM lecseréli TCP/IP-verem által a rendszermagnak nyújtott szolgáltatásokat, így biztosítva a Linux-rendszermaghoz a szükséges felületeket, valamint a foglalati kérelmeket; az adatokat az iNIC-en futó TCP/IP-veremnek továbbítja.



2. ábra Foglalati áthárító szerkezet



3. ábra A Linux TCP/IP-verme

A foglalati OSM

Az OSM a következő alrendszerekre van felosztva: felhasználói felület, üzenetközvetítő felület, rendszermagfelület és memóriakezelő alrendszer.

A felhasználói felület lecseréli a rendszermag `af_inet` foglalati rétegét. Ez az új réteg pontosan olyan felületet biztosít, mint az eredeti. Az üzenetközvetítő réteg felel a foglalati áthárító rendszer működéséért, biztosítja az elindítását és az irányítását, és a felhasználói foglalati kéréseket foglalati üzenetekre fordítja le.

Az OSM számára a rendszermag felülete biztosít rendszermag-szolgáltatásokat. Ez az az OSM-csatlakozási pont, ahonnan az OSM TCP/IP-veremszolgáltatásokat biztosít a rendszermagnak. Ezt az alrendszert úgy alakították ki, hogy a Linux-rendszermagon csak a legcsekélyebb változtatásokat kelljen elvégezni.

A memóriakezelő felel a felhasználószintű alkalmazások kapcsolattartáshoz szükséges tárhelyeinek a kialakításáért. A memóriakezelő alrendszert úgy tervezték, hogy

1. minél kevesebb DMA-kérésre és -visszaigazolásra legyen szükség,
2. a gazdagépen minél kevesebb megszakításkérelemre legyen szükség,
3. elkerülje a költséges fizikai-virtuális címátalakító folyamatokat,
4. elkerülje a futásközbeni dinamikus memóriatúltelítődéseket.

Az OSM a DMA-adatok tárolására két különböző adatterületet tart fenn. A küldésre szánt adatok adatterületét az iNIC felé, és a fogadott adatok adatterületét – ezeket a foglalati eszköz küldi a rendszermag felé.

Ahogyan a *3. ábrán* látható, a Linux hálózati összetevői rétegelt szerkezetbe vannak szervezve. A felhasználói területen programozók a hálózati szolgáltatásokat foglalatokon keresztül érik el, felhasználva a Linux által biztosított foglalati rétegszolgáltatásokat. A ***linux/net.h*** fejlécfájlban kialakított foglalati szerkezet képezi a foglalati kapcsolati réteg alapját. A felhasználói szint alatt található az INET foglalati réteg, amely az IP-alapú protokollok (mint például TCP vagy UDP) közti kapcsolati csatlakozópontokat képezi. A réteg felépítését a ***net/sock.h*** fejlécfájlban található adatszerkezet-foglalat határozza meg.

Az INET foglalati réteg alatti réteget a foglalat típusa szabja meg, esetleg a TCP- vagy UDP-réteg, vagy közvetlenül az IP-réteg. Az IP-réteg alatt találhatók a hálózati eszközök, amelyek a csomagokat közvetlenül az IP-rétegtől kapják.

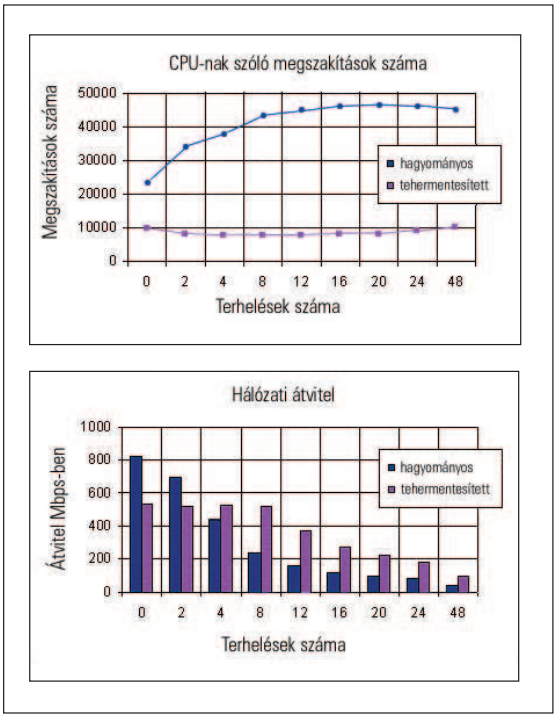
A foglalati OSM cseréli le az INET foglalati réteget. A foglalatok felől érkező összes foglalatokkal kapcsolatos üzenet I2O-üzenetekké alakítódik át, melyek végül az IOP-on található ISM felé továbbítódnak.

A beágyazott célpont

A beágyazott rendszerprogram a következő rétegekből épül fel: üzenetközvetítő réteg, TCP/IP-verem, eszközmeghajtó és RTOS. A program az üzenetközvetítő rétegnek azt a részét képezi, mely az OSM felől fogadja az üzeneteket, majd értelmezi őket, végül a kéréseket a TCP/IP-verem felé továbbítja. Ez a réteg felelős az üzenetek fogadásáért is, és ez küldi vissza a megfelelő válaszokat az OSM-nek. A teljesítménynöveléséért, valamint az örökölt osztott eszközmeghajtó rendszer miatt keletkező késések következményeit csökkentendő az üzenetközvetítő réteg az üzeneteket kötegekbe rendezi és csővezetéken keresztül továbbítja, egyúttal a válaszadásért is felel.

A TCP/IP-vermet teljes egészében a BSD 4.2 veremnek megfelelően építették fel, biztosítva a hálózati verem funkcionalitását

az üzenetközvetítő réteg felé. Mint az összes IOP-on futó egyéb program, a verem is az Intel 80310-es I/O processzor lapkakészletére van testreszabva, mely az Intel Xscale mikroszerkezetére épül (mely már fel van készítve az ARM-szerkezetre). A fejlesztés során a TCP/IP-vermet sokszor állították próbák elé, hogy a legkülönbözőbb mértékű hálózati forgalmat is a lehető leghatékonyabban kezelje.
A HDM-et úgy alakították ki, hogy a NIC-kel való áthárításból kifolyólag a lehető legtöbb haszonra tehessen szert. Ez magában foglalja a TCP- és IP-csomagok ellenőrzőösszegeinek ellenőrzését, az 1500 bájtnál nagyobb TCP-csomagok feldarabolását, és a lapka által támogatott megszakítások kötegelését. A következő NIC-lapkákhoz létezik támogatás: Intel 82550 Fast Ethernet Multifunction PCI/CardBus vezérlő, és Intel 82543GC Gigabit ethernetvezérlő.
Az RTOS olyan szabadalmaztatott operációs rendszer, melyet a bonyolult I/O-műveletek igényének megfelelően alakítottak ki. Ez az operációs rendszer teljes egészében I2O-képes. Ez részben azért alakult így, mert a tervezők egy prototípust is létre akartak hozni.
Mint már említettük, az alkalmazási réteg által kiadott foglalati hívások először átalakítódnak, majd pedig a PCI-sínen (bus) keresztül az I/O processzor felé továbbítódnak. Ez a beágyazott rendszer egy teljes értékű, I/O-átvitelek kezelésére kialakított számítógép, mely egy processzorból, memóriából, az RTOS-ból, valamint a PCI-sínből áll. Mivel elsősorban I/O-műveletekre hegyezték ki, nagyban lecsökkenti az üzenetváltások következményeit. Amint egy üzenet az IOP-hoz kerül, azonnal értelmezi. Az alkalmazás által kért foglalati hívás menten a beágyazott hálózati verem felé továbbítódik, és ha a foglalati művelet véget ér, az OSM rögvest értesítést kap.



40. ábra A próbák eredményei

A próbák eredménye

A prototípuson elvégzett teljesítménypróbák egyértelműen azt mutatják, hogy a TCP/IP-verem áthárításával mind a processzor terhelése, mind a gazdagép felé irányuló megszakításkérelmek száma csökkent. Egy hálózati szempontból teljesen leterhelt tesztgépen az áthárított verem képes volt a teljes forgalmat kezelni, a CPU pedig minden idejét az elsődleges alkalmazásoknak szentelhette. Egy áthárított veremmel nem rendelkező alaprendszeren a CPU a tevékenységét kénytelen volt rendszeresen megszakítani, de még így sem tudta a hálózati forgalmat teljes sebességgel ellátni, ami természetesen a hálózati forgalom lelassulásában mutatkozott meg.

Elképzelések a jövőt illetően

Az iSCSI (adattárolás IP-protokollon keresztül SCSI-csomagok TCP/IP-be csomagolásával) elterjedésével egyre nagyobb igény mutatkozik a hálózati forgalom növekedéséből adódó terhelés csökkentésére. A TCP/IP-verem IOP-ra való átültetésével az iSCSI-adapterek teljes értékű TCP/IP-verem támogatással fognak rendelkezni. Ha az iSCSI-t a normál SCSI programozói felület részévé tennék, az nagyban csökkentené a Linux-felületre kifejtett hatását. Ahhoz, hogy az iSCSI felvehesse a versenyt a Fibre Channel módszerrel, legalább hasonló teljesítményt kell tudnia felmutatni.
Ugyancsak a jövőre vonatkozó terv az is, hogy RTOS-ként Linuxot használjanak. A prototípus elkészítésekor egy Intel i960 RM/RN processzort használtak, de ekkor még nem állt rendelkezésükre beágyazott Linux. Azóta bemutatták az Intel Xscale mikroszerkezetet, amely a Linux alkalmazását a StrongARM-magon lehetővé tette. A Linux-alapú StrongARM Linux portolása az Xscale mikroarchitektúrára ez év végére befejeződik.
E mögött a prototípuskészítési terv mögött számos más cél is meghúzódott:

1. megmutatni, hogy a hálózati feladatoktól mentesített gazdagépprocesszor a hálózati forgalom elemzésére jóval kevesebb órajelciklust pazarol el;
2. bebizonyítani, hogy egy egyedi programmal felvértezett iNIC ugyanazon hálózati tevékenységek elvégzése mellett a hálózat teljesítményét maximálisan szinten képes tartani;
3. engedélyezni olyan I/O-processzorok használatát, amelyek a hálózati forgalom kezelésében képesek együttműködni a gazdagép processzorával, így maximalizálva csekély költséggel a Linux-kiszolgáló teljesítményét.

A TCP/IP-verem hálózati programokból álló környezetre történő áthárításának módszere beágyazott processzorok segítségével a teljesítménynövelés egyik leghatékonyabb módja. A nagysebességű hálózatok fejlődésében és a hálózati adattárolás elterjedésében a TCP/IP elkerülhetetlenül igen fontos szerepet fog játszani.

Technikai szakértők:
Dave Jiang, Dan Thompson, Jeff Curry, Sharon Bartmans, Don Harbin and Scott Goble.

Chen Chen
a fejlett I/O-alkalmazások terén végez kutatásokat az Intelnél. Elérhető a chen.chen@intel.com címen.

David Griego
több mint három éve lelkes Linux-rajongó. Szintén az Intelnél dolgozik, a tervezéstechnikai fejlesztési részlegnél. Elérhető a david.a.friego@intel.com címen

A megfigyelések megfigyelése

Ebben a hónapban Marcel a Gqcam a xawtv-t és a MASH alkalmazását mutatja be, amelyekkel webkameránkat kezelhetjük.

Bonjour, mes amis! Isten hozott benneteket Chez Marcel kitűnő Linux-konyhájában! Bocsássátok meg, ha egy kicsit letörtnek látszom, de ez alkalommal meggondolatlan voltam. Tudjátok, amikor François elmondta, hogy mai beszélgetésünk témája a SOHO lesz, arra gondoltam, ez fantasztikus! Olyan téma, ami naprendszerünk csodáit mutatja be. Képzelhetitek a meglepetésemet, amikor kiderült, hogy a SOHO ebben az esetben Small Office/Home Office (kicsi/otthoni iroda) és nem Solar and Heliospheric Observatory (Szoláris és helioszférikus obszervatórium – egy napmegfigyelő űrszonda ➲ http://www.soho.? ➲ http://www.nascom.nasa.gov/ – a ford.). Ekkor egy kicsit elkezdtem játszani a szavakkal. Az obszervatóriumok, vagyis a csillagvizsgálók arra valók, hogy az ember megfigyelje a csillagokat, mint az a távcső is, amit monsieur Hubble-ról neveztek el. És akkor az jutott az eszembe, hogy a mi előkelő vendégeink az igazi csillagok, vagyis Ti, kedves barátaim.

Innen származik mai esti menünk. Hogy a következő recepteket a legmagasabb fokon készíthessük el, melegen ajánlom a legutóbbi rendszermag beszerzését. Különösen azért, mert a legtöbb olcsó webkamera, amit majd a helyi számítástechnikai üzletben találtok, USB-csatlakozóval rendelkezik. Minél frissebb a rendszermag, annál valószínűbb, hogy támogatja az eszközt. A receptek ízlelgetésének-tesztelésének folyamán én a 2.4.9-es rendszermagot és két kamerát használtam, amelyek közül az egyik a CPiA lapkakészletre épült, a másik pedig az ov511-es egységet használta.

A világhálóra kötött kamerák túlnyomó része az adott honlap üzemeltetőjének életébe enged bepillantást. Néhány webhelyen a kamera azt a célt szolgálja, hogy biztosítsák a látogatót, a munka valóban folyik, máshol a szülők figyelhetik meg gyermekeik játékát a nappali felügyelet közben. Az alkalmazást, mely ezeket a képeket a számítógépre továbbítja, képlopónak (frame grabber) is szokták nevezni.

François, hol a csudában vagy? Á, úgy látom gyönyörű és titokzatos hölgy ül a 22-es asztalnál… François ábrándozik. Hol is tartottam? Mais oui, a képlopóknál. A Gqcam épp egy ilyen csomag. Alkalmas a kamera üzembe helyezésére, bármit szemmel tarthatunk vele és tetszés szerint rögzíthetünk képeket. Akár egy folyamatos esemény tetszőleges időtartamáról készült képsorozat önműködő rögzítésére is beállítható.

A Gqcam-mel való foglalkozást Cory Lueninghoener honlapján (➲ http://cse.unl.edu/~cluening/gqcam) a legfrissebb forráskód letöltésével kezdd, majd csomagold ki és fordítsd le:

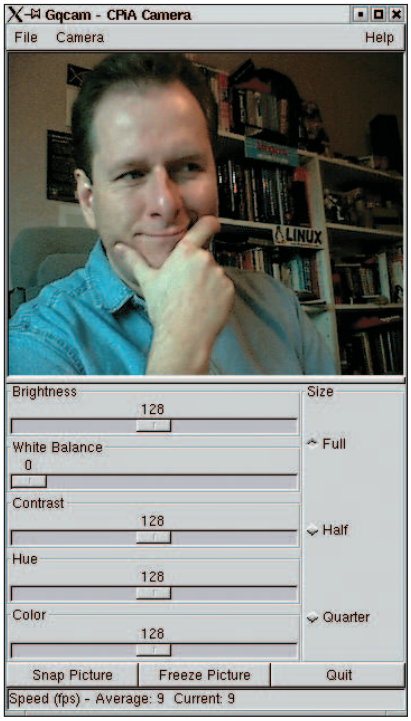
```
tar -xzvf gqcam-0.9.tar.gz
cd gqcam-0.9
make
```

Valami mást vártatok, non? Ebben az esetben nincs szükség `make install`-ra, mes amis. A végső bináris állományt a */usr/local/bin* könyvtárba másolhatjátok vagy ahová a futtatható állományt szeretnétek menteni. Az alkalmazást a `./gqcam &` begépelésével közvetlenül a fordítás könyvtárából is futtathatjátok, ha úgy tetszik. Észreveszitek majd – ahogyan én is –, hogy a program alapértelmezésben a /dev/video útvonalat figyeli. Az én USB felületű webkamerám ténylegesen a */dev/video0*-ként volt elérhető, a */dev/video* egy könyvtárra mutatott. Ezt a `-v` paraméterrel lehet megoldani:

```
gqcam -v /dev/video0
```

Hogy mit figyeltetsz a Gqcam segítségével, teljesen rajtad múlik, de arra vonatkozóan, hogy magával az alkalmazással mit kezdhetsz, engedj meg néhány javaslatot. Az *1. képre* nézve egy pillanatképet láthatsz a Gqcam működéséről. Futtatása közben érdekes dolgokat tehetsz meg vele, például a ***Snap Picture*** gombra kattintva az adott képet bármikor rögzítheted, amelyet aztán tetszésed szerint menthetsz png, ppm vagy jpg formátumban. Ha gyors kattintásokkal több is képet rögzítesz, egyfajta filmmé is összerendezheted őket.

Természetesen folyamatosan képeket kattintgatni unalmas, ezért a Gqcam időzítőt bocsát a rendelkezésünkre. Kattints a ***Camera*** menüpontra és válaszd



1. kép A Gqcam egy alkalmas pillanatban lefotózza francia főszakácsunkat

a ***Set timer*** szolgáltatást. Kiválaszthatod a könyvtárat és a megfelelő állománynevet a kép rögzítéséhez, például */somedir/mypic.jpg*. Állítsd be az időzítőt egy másodpercre, és indítsd el. Az eredmény a /somedir könyvtárban keletkező, egymást követő képsorozat lesz, melyeket a program mypic kezdetű, a dátum és idő hozzáírásával keletkező állománynévvel és jpg kiterjesztéssel lát el.

A legszórakoztatóbb az, hogy egyetlen parancs segítségével a kis mozit akár futtathatod is, s ez a parancs az ImageMagick csomag része (amely valószínűleg már a gépeden van):

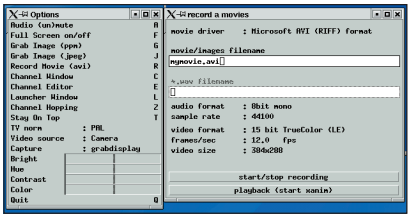
```
animate mypic.*.jpg
```

A Gqam közvetlenül parancssorból is futtatható, egyszerűvé téve ezáltal, hogy olyan `cron`-feladatot hozzunk létre, amely kiválasztott időközönként rögzíti a képeket:

```
gqcam -v /dev/video0 -d
/wwwdir/filename.png
```

A `-d` kapcsoló határozza meg a

rögzített képek lerakásának a helyét. A képek alapértelmezett formátuma a PNG, a `-t` kapcsolót pedig ennek a beállításnak JPEG-gel való felülbírálatára használhatod. Ha most valahol webkiszolgálót futtatok, és éppen kedvem tartja rögzíteni az aktuális pillanatot, voilá! Mindannyian az Internet sztárjai lehetünk, non? Már csak egy egyszerű



2. kép AVI-felvétel beállítása az xawtv-ben

weboldalra van szükségünk, amellyel a képet megjeleníthetjük – valami ilyesmire gondolok:

```
html> <head>     <title>Cam Chez
Marcel</title> </head> <body
bgcolor="White"> <center>
<h1>Bienvenue! You are watching
Cam Chez Marcel</h1>     <img
src="camsnap.png"> </center>
</body> </html>
```

Egy másik program is létezik, amely rendelkezik a Gqcam néhány szolgáltatásával: ***Gerd Knorr*** xawtv-je. Ahogyan neve is sejteti, a program szándéka az, hogy eszközöket biztosítson tévéprogramok nézéséhez (a megfelelő alkatrészekkel), de emellett a webkamerákról érkező jelet is kezelni tudja. Amikor úgy érzed, hogy valami olyasmit csináltál, ami fokozott figyelmet igényel (videós árubemutató például), ízlésed szerint egy videoklippel is feldobhatod az alkotásodat. A xawtv megadja az ehhez szükséges képességet. Látogasd meg a ➲ http://bytesex.org/xawtv honlapot, és töltsd le a legutóbbi változatot. A munkakönyvtárban csomagold ki a forrást és fordítsd le:

```
tar -xzvf xawtv_3.64.tar.gz
cd xawtv-3.64
./configure
make
make install
```

A program a `xawtv &` parancssorba történő begépelésével indítható. Ez esetben a program kamerám útvonalát felismerte, nem volt szükség további beállításokra. Jobb egérkattintásra egy beállítómenü jelenik meg, ahol olyan képtulajdonságokat állíthatsz be, mint a szín és aa fényerő. A Gqcamhez hasonlóan itt is lehetőség nyílik kép rögzítésére, és ppm vagy jpeg formátumban való mentésére. Beszélgetésünk szempontjából a következő menüpont az érdekesebb. Az R megnyomása után egy felvételablak jelenik meg. Az alapbeállítás a többszörös képrögzítés, a képek összeillesztéséhez pedig külső programot használhatsz. Emellett rákattinthatsz a ***Movie driver*** feliratra és kiválaszthatod a ***Strukturálatlan videoadat***-ot vagy az ***AVI formátum***-ot. Válaszd az AVI-t, írj be egy állománynevet, amibe a mentést kéred, majd kattints a ***Start recording*** fülre. Amikor

3. kép Elle est une dame mystérieuse

megadtad a kívánt hosszt, kattints még egyszer a fülre (ez ekkor már ***Start/stop*** feliratú). Ha a xanim csomag telepítve van, a közvetlenül alatta lévő fülre kattintva a felvett anyag visszajátszása is kiválasztható. Vess egy pillantást a *2. képre*, vagyis a felvétel párbeszédablakának egy lehetséges állapotára. Gyertek közelebb, mes amis! Figyeljétek csak François-t, kicsit nyugtalannak tűnik. Meglehetősen el van ragadtatva a 22-es asztal titokzatos hölgyétől. Szegény fiú túl félénk, hogy odamenjen, és személyesen beszéljen vele. Mai utolsó receptemet az én hűséges pincérem és a hozzá hasonlók számára mutatom be. Ha szemtől szemben szeretnél lenni valakivel, de a távolság (vagy egyéb körülmények) ezt bonyolulttá teszik, a megoldás a videokonferencia. Itt jut szerephez a MASH. A MASH valójában az Open MASH konzorcium által létrehozott eszközök gyűjteménye (***Larry Rowe*** és ***Steve McCanne*** alkotásai). Céljuk egy szabadon felhasználható eszközkészlet létrehozása volt együttműködő fejlesztés és adatfolyam-alkalmazások (streaming) támogatására. A MASH az olyan multimédiás környezeteket képviseli, amelyek heterogén környezeteket kapcsolnak össze. Nem viccelek!
Ha kedved van, a forrást letöltheted és a csomagot az alapjaitól kezdve lefordíthatod:

```
tar -xzvf mash-src-
➥5.1.5.tar.gz
cd mash-code
./build
```

A legkönnyebb eljárás az előfordított bináris csomag letöltése az Open MASH honlapjáról (➲ http://www.openmash.org). Ha kész, a csomagot csak ki kell bontani és a telepítő parancsállományt le kell futtatni. Figyeljetek, mes amis:

```
tar -xzvf mash-bin-5.1.5
➥-linux-gnu.tar.gz
cd mash-5.1.5
./setup.sh
```

Olyan egyszerű, non? Ahogy említettem, ez eszközgyűjtemény, nem egy különálló program. A legérdekesebb, különösen François számára, a videokonferencia-alkalmazás (a vic), amely egyszerre több felhasználónak teszi lehetővé a videokommunikációban való részvételt.

```
vic hostname/port_no
```

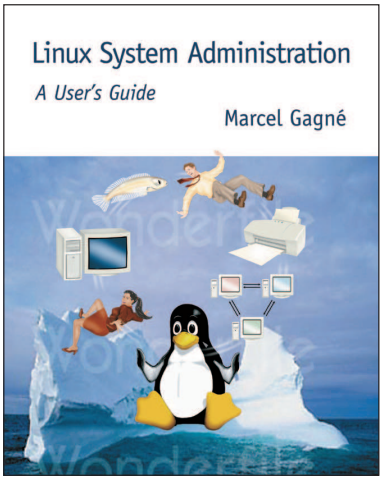
Most egy kis előugró ablaknak kell megjelennie a következő egyszerű üzenettel a közepén: „Waiting for video…”. A kép, amire várunk, természetesen az ifjú hölgy a 22-es asztalnál. A fiatal hölgyeknek pedig valószínűleg François. Ki tudja? Amíg a másik oldal csatlakozására várunk, kattintsunk a ***Menu*** gombra. Egy vezérlőablak jelenik meg, ahol az aktuális folyamat jellemzőit vezérelhetjük. Kattints az ablak tetején lévő ***Transmit*** gombra. Egy kis képet fogsz látni magadról, ahogy a kamerád előtt ülsz, egyelőre a másik oldal kapcsolata nélkül. Most kattints a kis képre, és egy nagyobb kép jelenik meg. Amikor a vonal másik végén valaki bekapcsolódik, megjelenik a képe, és ugyanezzel az eljárással még étvágygerjesztőbb képet érhetsz el. A *3. kép* François-t mutatja, miközben a vic segítségével a titokzatos hölggyel beszél. Amint a képen is látható, még így sem meri a saját képét mutatni, helyette más megtestesülést választott. Szegény François.
Ahogy a többi alkalmazásnál is láthattuk ma, a vic videovezérlő menüi számos beállítást megváltoztatását lehetővé teszik, például módosíthatod a fényerőt, a kontrasztot vagy a színeket. Még a nézőkék mérete is megváltoztatható a ***Size*** gombra kattintva.
A záróra megint elérkezett. Rendes körülmények között megkérném a pincéremet, hogy töltsön egy utolsó pohárral, de attól félek, ma este az én feladatom lesz. François kissé szórakozottnak tűnik.
Oh, l'amour.

Marcel Gagné

Linux-rendszerfelügyelet

Kóstoló francia konyhafőnökünk nemrég megjelent könyvéből.

Legújabb könyvem, a Linux rendszerfelügyelet – felhasználói kézikönyv még az idén felkerül a hazai könyvesboltok polcaira. Mind a Linux Journal, mind a Linuxvilág munkatársai olyan fenemód jószívűek voltak, hogy helyet biztosítottak arra, hogy ebben a számban a könyvből egy kis ízelítőt adjak. Vért izzadtam, mire kitaláltam, mit tudnék ilyen kis helyen bemutatni belőle, hiszen annyi mindent választhatnék. Lapolvasó telepítése? Vagy inkább CD-íróé? A hálózat megvédése a betolakodóktól? Nyomtatás? Mentések? Levelezési fogások? Melyiket szeressem? Végül is úgy döntöttem, hogy egy olyan témával foglalkozom, amelyik mai internetes világunkban nem kap elég nagy nyilvánosságot, s ez nem más, mit a jó öreg felhasználói biztonság. A könyv hetedik, Felhasználók és felhasználói csoportok című fejezetéből fogok szemezgetni. Nem az egész fejezet olvasható itt és nem is részletek sorozata. Már hallom a kifogást: „A szakács túl sokat kortyolgat a saját borából!”. Valójában mintát, vagy ha úgy tetszik, kóstolót szerettem volna adni mindabból, amivel az olvasó az új könyvem oldalain találkozhat. Mintha csak csipegetnénk egy büféasztalról. Ahogy Chef Marcel mondaná: Bon appétit!



Adatok

Cím: Linux-rendszerfelügyelet
Szerző: Marcel Gagné
Kiadó: Addison-Wesley
ISBN: 0-201-71934-7

Élet egy többfelhasználós világban

A Linux többfelhasználós operációs rendszer, ami azt jelenti, hogy egy időben egynél több felhasználó is dolgozhat rajta. Minden felhasználóra a felhasználói nevével hivatkozhatunk, s ezek mindegyikéhez egy felhasználói azonosító (UID) tartozik, amely egy vagy több felhasználói csoporttal társítható. A felhasználói nevekhez hasonlóan ezek a csoportok is egy-egy numerikus azonosítóval rendelkeznek, amit ebben az esetben csoportazonosítónak (GID) hívunk. Mindkét azonosítófajta egyedi értékekkel rendelkezik.
Az állományok és könyvtárak biztonsága a Linuxban olyan engedélyek révén valósítható meg, amelyek a felhasználói azonosítóval közvetlen kapcsolatban állnak. A felhasználók lehetnek közönséges és felügyeleti jogokkal rendelkező felhasználók. A legfőbb felügyeleti feladatkört ellátó felhasználó neve: root. A felhasználói azonosítótól függ, hogy az adott személy milyen parancsokat futtathat, milyen állományokba tud betekinteni vagy milyeneket tud módosítani. Minden felhasználói azonosítóhoz jelszó tartozik, amelyek megváltoztatása csak szabályozott keretek között történhet(ne).

Mikor ne használjuk a rendszergazdai jogosultságot?

A kérdésre a legrövidebb válasz: soha ne használd, csak ha nagyon muszáj. A veszély abban rejlik, hogy a rendszergazda gyakorlatilag a rendszer teljhatalmú ura, s egy pici hiba is komoly következményekkel járhat, vagyis akár az egész rendszert tönkreteheti. Ha egy mód van rá, közönséges felhasználói azonosítóval dolgozz. További indokok is felsorolhatók.
Az első a biztonság kérdése: mivel a rendszergazda mindenhez hozzáfér, csak azzal érdemes megosztani a jelszót, akinek valóban szüksége van rá. Minél kevesebben ismerik, annál jobb. Miért érdemes ilyen féltékenyen őrizni ezt a jogosultságot? Mert sokkal egyszerűbb a biztonságkezelés és csökken annak a veszélye, hogy a hibák az egész rendszert befolyásolják. Valóban egy közönséges felhasználó is komoly kárt okozhat egy rendszerben, de ennek kockázata sokkal-sokkal kisebb.

A jelszóállomány ellenőrzése

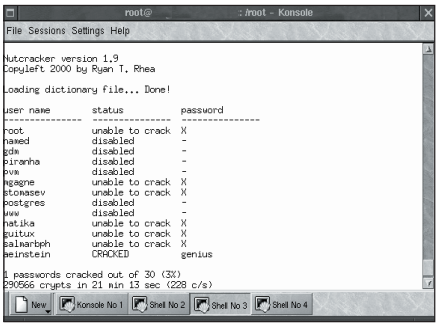
Ha még mindig nem lenne elég feladatod, itt a következő munka: szabályos rendszerességgel ki kell gyűjtened a használaton kívüli felhasználói azonosítókat. A könyv egy korábbi fejezete bemutatja a `finger` parancsot, amely az azonosítók használatának legutóbbi időpontjáról jelenít meg adatokat. Próbáld ki, hogyan lehet a parancs beírásával az összes felhasználói azonosítót és az utolsó belépés időpontját kiíratni. A parancsban lévő (az aposztróf a `sort` parancs és a pipa előtt) a ~ (tildejel) billentyűjén találod meg (az angol billentyűzeten, a magyaron pedig az 1-es a számbillentyűzeten – a szerk.).

```
finger ·sort /etc/passwd | cut -f1 -d":"·
      ➥| grep -i log | more
```

A parancs kimenete valahogy így néz ki:

```
Login: aeinstein          Name: A. Einstein
Never logged in.
Login: guitux             Name: Tux the Penguin
Last login Mon Jan  8 14:54 (EST) on tty2
Login: halt               Name: halt
Never logged in.
Login: lp                 Name: lp
Never logged in.
Login: mail               Name: mail
Never logged in.
Login: mgag               Name: Marcel Gagne
Last login Wed Mar  7 17:29 (EST) on 1 from
➥website
Login: named              Name: Named
Never logged in.
Login: natika             Name: Natika the Cat
```

Figyelmeztetésként fogadd meg a jótanácsomat: használd a józan ítélőképességedet (rendszergazdák számára alapvető követelmény). A felhasználói azonosítók egy része – például a `sync` vagy az `lp` – a rendszer része. Megszokott esetben ezekkel soha senki nem léphet be a rendszerbe. Másrészt Mr. Einstein (a lista tetején) sem lépett még be soha, és ez természetesen nem is rendszerfelhasználói név. Az is megeshet, hogy ez új azonosító (most az), vagy akad egy felhasználó, aki még soha nem lépett be ezen a néven. A második esetben a legjobb minél előbb megszabadulni az azonosítótól. Az előbbi példa parancssori jártasságodat volt hivatott növelni, de meg kell mondanom, járhatóbb út is létezik. Linux-rendszereddel egy `lastlog` nevű ügyes kis parancsot kapsz, amely éppen ezt csinálja.



Amiért ne szótár szavakat használjunk jelszóként…

```
[root@scigate /root]# lastlog | more
Username    Port    From     Latest
root        tty1       Wed Mar 7 17:18:40
                       -0500 2001
bin                    **Never logged in**
daemon                 **Never logged in**
adm                    **Never logged in**
lp                     **Never logged in**
sync                   **Never logged in**
shutdown               **Never logged in**
mgagne      1scigate Wed Mar 7 17:29:55
                       -0500 2001
postgres               **Never logged in**
www                    **Never logged in**
natika      8localhost.locald      Thu Dec 7
                                   ➥14:30:15
                       -0500 2000
guitux      tty2       Mon Jan  8 14:54:55
                       -0500 2001
```

A `lastlog` parancs az adatokat a */var/log/lastlog* nevű állományból veszi, amely kézzel nem módosítható.
Maradt még egy teendőd, amit rendszeres időközönként el kell végezned: ez a `pwck` futtatása. Alapértelmezésben a program végiglépked a */etc/passwd* és a */etc/shadow* állományokon és végrehajt egy ellenőrzést annak vizsgálatára, hogy a megfelelő számú mező szerepel-e, és minden név egyértelműen azonosítható-e. A csoportazonosító vizsgálatára a `grpck` parancs használható.

Miként törik fel jelszavainkat a kalózok?

Arról, hogy miért érdemes gondosan megválasztani a jelszót, a könyv korábbi részében olvashattál, a jelszóállomány leírásánál – kifejezetten jelszómezőről a nem árnyékállományokban (nonshadow). Egy gyors emlékeztető:

```
root:2IsjW45pb4L56:0:0:root:/root:/bin/bash
```

A jelszó mezője (a második) egy nyalábolóindexelés-algoritmus (hashing) szerint kódolt állapotban látható. Ha nagyon alapos vagy, és érdekel a részletes leírás, csak gépeld be a `man crypt` parancsot, és itt mindent megtalálhatsz, amit a jelszavak kódolásáról valaha is tudni szerettél volna. Röviden összefoglalva: a felbukkanó furcsa jelszó valójában jelszavadnak egy véletlenszerűen előállított kétkarakteres szó (a ***salt***) segítségével kódolt formája. Ezt a saltot átadva a kódoló-nyalábolóindexelési eljárásnak előáll a végleges karaktercsoport. A nyalábolóindexelés kifejezés olyan eljárást takar, melynek során az adat gyors visszanyerésére egy karaktersor (például egy személy vezetékneve) segítségével (eszményi esetben) egyedi kulcs áll elő. Amit teszünk, az a szöveg kódolása rövidebb, rendszerint numerikus megfelelőjére.
A jelszótörők ezt a saltot használják fel jelszólétrehozásra a szótárban szereplő minden egyes szót kipróbálva. Bár ez kicsit bonyolultnak tűnik, mégsem az. Egy egyszerű program meghívja a titkosító eljárást, lefuttatja egy szón, és összehasonlítja a */etc/passwd* állomány jelszóbejegyzésével. Ha egyezik: bingó! Megszerezték a jelszavadat. Ha nem, veszik a következő szót. Egy elég gyors rendszeren nem tart túl sokáig, hogy a kalózok minden jelszóhoz megtalálják az utat.
Nem hiszel nekem? Csak vess egy pillantást a képen látható, a szabad felhasználású Nutcracker által előállított listára. Ez a programocska épp az imént felvázolt „nyers erő” (brute-force) jelszófeltörő módszer egy fajtáját alkalmazza.
Ahogy a *képen* is látszik, a jól megjegyezhető szavak – mivel túl gyakoriak – jelszóként rossz választásnak bizonyulhatnak.

Honnan jelentkeztem be?

Nézzük, mi történik, amikor bejelentkezünk egy gépre. Minden rendben lévőnek tűnik. Megvan a felhasználói nevem, kéri a jelszavamat, beírom és voilá, bent vagyok.

```
login: mgagne
Password:
Last login: Mon Jan   8  16:00:39 from energize
```

De várjunk csak! Mi az a kis üzenet, ami a jelszóbeírás után megjelent? Mi a fene az az ***energize***? Az ***energize*** nyilván annak a gépnek a neve, amelyikről a legutóbb bejelentkeztem. Kivéve ha nincs ilyen nevű gépem. Sőt, tételezzük fel, hogy nem is ismerek ilyen nevű gépet és mindig ugyanonnan jelentkezem be. Az egyetlen magyarázat, hogy egy ***energize*** nevű gépről valaki az én nevemmel és jelszavammal lépett be a rendszerbe. Ez csak kitalált eset, de jól szemléltet egy szokást, amire esetleg érdemes a felhasználóinkat megtanítani. Ha nap mint nap ugyanarról a számítógépről jelentkeznek be, ez az üzenet nem változhat. Amennyiben a gép nevét az üzenetben nem ismerik fel, bölcsen teszik, ha értesítenek.
A biztonság kérdése nem kizárólag a rendszergazdára tartozik, bőven akad tennivalója enélkül is. Minden segítség elkél, tehát a felhasználók bevonására is szükségünk van. Hadd tudják és érezzék, hogy a rendszer biztonságának kérdése legalább annyira az ő ügyük, mint a tiéd.

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/

FORDUL A LINUXVILÁG

Kedves Olvasóink!

Tájékoztatni szeretnénk Önöket a lapunk életében történt változásokról.

Keressék az újságárusoknál!

Előfizetés

Idén 11 számmal jelenünk meg, mivel a januári és a februári számot összevontuk, terveink szerint most utoljára. Önök továbbra is nyugodtak lehetnek: előfizetésük a jövőben is lapszámokra vonatkozik majd: az egyéves előfizetés 12, a féléves pedig 6 számra vonatkozik (hiszen összevont számunk egynek számít).

Felemeltük előfizetőink kedvezményeit 8, illetve 20%-ra:
féléves (6 lapszám): 10 930 Ft
egyéves (12 lapszám): 19 008 Ft

A pénz beszél...

Ebben az évben belső köreinkből kiléptünk ország-világ színe elé.
A Linuxvilág szakmai írásaival kívánja meghódítani Olvasói szívét, ezért már a kezdetektől fogva a minőségre helyeztük a hangsúlyt és továbbra sem kívánunk csupán „hirdetőoszloppá" válni. A január–februári számtól kezdődően magazinunkat az újságárusoknál is megtalálhatják.

Éves előfizetéssel olcsóbb, számonként csak 1584 Ft.

Biztos megérkezés

Előfizetőink számára választási lehetőséget kínálunk: vagy ajánlott küldeményként juthatnak hozzá lapunkhoz (vállalva ennek költségeit), vagy pedig az eddig megszokott módon egyszerű postai küldeményként kapják meg.

Már ajánlott küldeményként is megrendelhető (+90 Ft/küldemény).

Akció!

Új Olvasóink számára legelső négy számunkat

- 2000. november,
- 2000. december,
- 2001. január,
- 2001. február–március

óriási, 50%-os kedvezménnyel kínáljuk.

További lehetőségek: a 2002 január előtt megjelent régi számaink (az első négy szám kivételével) ára változatlanul 1484 forint marad.
Az áremelés bejelentése előtt beérkezett előfizetéseket természetesen még a régi áron fogadtuk, aki tehát megrendelését már előzőleg meghosszabbította, még a régi áron juthatott magazinunkhoz.

Terjesztés: telefon: (06-1) 303-9119
fax: (06-1) 303-1619
e-mail: terjesztes@linuxvilag.hu

Ha bármilyen kérdésük, gondjuk támad, kérjük, írják meg az **info@linuxvilag.hu** címre.

További kellemes és hasznos olvasást kívánunk!

A Linuxvilág munkatársai