Nyelvészkedjünk egy kicsit!

Itt az ideje, hogy figyeljünk azokra a szavakra is, amelyeket nap mint nap használunk.

Köszönöm a magyarításokkal kapcsolatban beküldött véleményeket, ötleteket. Örömmel tapasztaltam, hogy a szakma egy jelentős hányada érzi és érti, hogy miért fontos, miért hasznos az idegen szakkifejezéseket lefordítani. Egyúttal be kell valljam, hogy sok olyan szóval találkoztunk, melyekhez nagyon nehezen találunk csak fordításokat, sőt, van néhány, melyhez eleddig nem is sikerült megfelelő fordítást találnunk. Egy-két kérdéses szó kapcsán összegyűjtöttem a gondolatokat, kísérleteket, ismét kérek mindenkit, hogy ha van ideje, gondolkozzon el ezeken a szavakon, és írja meg nekünk, ha van jó ötlete.

Masquerading

A fogalom egy szolgáltatást takar, ennek lényege, hogy tetszőleges számú gépet egy kiszolgáló mögé rakunk (a kiszolgáló lesz az adott hálózati szakasz átjárója), de amikor a gépek a külső hálózattal akarnak beszélni, a kiszolgáló kezeli a kapcsolatot helyettük. A külső hálózat számára a kiszolgáló válaszol, ő alakítja át és közvetíti az adatokat.
Több fordítási kísérlet volt, az első vonal, maradjunk a legalább hasonló hangzású szavaknál, megbukott, a maszkolás értelmi ütközés miatt esett ki (a hálózat témakörében ugyanis már van maszkolás: *masking*), a maszkírozás és a maskarádézás egyéb okok miatt (értelem, magyarság stb.).
Jelenleg az értelem szerinti fordítások tűnnek befutónak, a rejtés, a bújtatás, a takarás és a képviselés. Ezek közül - remélhetőleg hamarosan - egy marad fenn, közös megegyezéssel. Végig kell gondolni azt is, hogy a szakszövegnél hogyan fogalmazzuk meg, hogy például a *Sanci* gépnek Mormogi nyújtja ezt a szolgáltatást:

- Mormogi elrejti Sancit, Sanci számára Mormogi nyújt rejtegetést...
- Mormogi bújtatja Sancit, Sanci számára Mormogi nyújt bújtatást...
- Mormogi takarja Sancit, Sanci számára Mormogi nyújt takarást...
- Mormogi képviseli Sancit, Sanci számára Mormogi nyújt képviselést..

Természetesen a példamondat is változtatható, hisz az utolsó részmondat helyett mondhatjuk, hogy Sancit Mormogi képviseli (a külső hálózattal szemben) stb. Várom mindenki véleményét, kinek melyik változat tetszik legjobban (esetleg egy új kifejezés jutott eszébe)!
➲www.linuxvilag.hu/nyelv/

Compatible

Ezt a szót a legtöbben leírjuk magyarul. Igen ám, de a rendkívül kényelmes megoldás nagy veszélyeket hord magában. Ha azt a nem jelentéktelen tényt félretesszük, hogy így nem magyarítást végzünk, még mindig ott marad a gond, hogy az angol annyira könnyed, pontatlan nyelv, hogy egy-egy szót sok értelemben használhat. Néhány helyzet, ahol a compatible szót az angol szaknyelvben használjuk:

- egy alkatrész egy bizonyos típusú gépbe beszerelhető
- két különböző típusú gép (néhány) alkatrésze egyikből a másikba áthelyezhető
- egy gép megfelel valamilyen előírásnak, szabványnak
- egy gép helyettesíteni tudja egy másik gép bizonyos szolgáltatásait
- egy gép saját sorozatának egyéb változatain futó programokat képes futtatni.

A példák között vannak hasonlók, és a legtöbb átültethető a programok és a programmodulok világába. Gondolhatjuk tehát, hogy egy huszárvágással elintézzük, mostantól mindegyikre leírjuk magyarul a compatible szót (hogy rövid vagy hosszú í-vel, az már egy másik történet), de vajon nem vesztünk-e ezzel? Nemcsak a magyar nyelv színességéből, de a szaknyelv pontosságából is. Nézzük, milyen kísérletek voltak idáig, és hogy (szerintem) melyiket hol érdemes használni.
A csereszabatos volt az első nagy port felkeverő magyarítás. Ezt olyan helyzetekben ajánlom, amikor egy gép vagy egy program egy másik szolgáltatását ki tudja váltani, a két elem egymással kicserélhető. Például a PC-DOS és az MS-DOS (többnyire) csereszabatosak egymással. Elnézést, keresek egy linuxos példát is. A KDE és a Gnome egymással csereszabatosak az X szemszögéből. Az egy másik kérdés, hogy a külön az egyik vagy a másik környezetre írt programok néha nehezen kelthetők életre a másik környezetben...
Később rájöttünk, hogy meddő dolog egy szóval helyettesíteni ezt a sok helyen használt fogalmat, és megjelentek a csak egy-két értelemben használatos szavak. Ilyen például a megfelelő. Ez akkor használható, ha egy gép vagy program eleget tesz valamilyen szabványban leírt követelményeknek, illetve ha például egy program egy másik program összes szolgáltatását képes nyújtani (itt csak egy irányban kell feltételezni a képességet). A Linux erősen törekszik, hogy POSIX-megfelelő legyen, a MESA megfelel az OpenGL szabványnak, tehát OpenGL-megfelelő.
Még egy kemény dió maradt, amikor két program vagy alkatrész képes egymással kapcsolatot tartani, együtt dolgozni. Erre a változatra az együttműködő (együttműködésre alkalmas), illetve az összeférő szavakat ajánlom. A két program képes az együttműködésre, mindkét program képes egyszerre ugyanazon szabályok szerint kezelni egy eszközt, egymással összeférnek. Egy SCSI lemez és egy SCSI lapolvasó (jó esetben) összeférnek.
Gondoljuk csak át ezt a példát! A SCSI kártya, a lemez és a lapolvasó is SCSI-megfelelő, az eszközök egymással képesek együttműködni, a kártyára kötött két eszköz összefér. Ezek nem egyértelmű tények, hiszen a három közül bármelyik lehet hamis, míg a másik kettő igaz.

Hardware, Software

Idáig nehézségekről beszéltem, ez már komoly kihívás. Nemcsak azért, mert iszonyúan általános értelmű szavakról beszélünk, és nemcsak azért, mert már mindenki ismeri és gyakran használja

ezeket a szavakat, hanem azért is, mert a szóbokor mellé az angol nyelvterületeken nap mint nap hozzácsapnak még egy-két szót. Ilyenek például a *freeware* (ingyenes), a shareware (próbaváltozat), a *bookware* (a program használatának feltétele egy bizonyos könyv megvásárlása) és általában bármi, ami feltétele az adott program vagy eszköz hivatalos használatának, megtoldva a -ware végződéssel.
Még egyszer térjünk vissza arra, hogy miért is fontos ezek helyett magyar szavakat használni. Akit még mindig nem győztem meg, kérem, forduljon egy ismerőséhez, és mondja a következőt: „Te figyelj! Hogyan határoznád meg pontosan a *hardware* szó jelentését?" Ha rendszergazda, viccesen azt válaszolja majd: „A *hardware* az, amibe bele lehet rúgni..." Tehát bármi, ami tárgy. Esetleg feltétel még, hogy kapcsolatban legyen a számítástechnikával, de ez már csak honi pontosítás. A software szóra még pontatlanabb meghatározást kapunk: „Ami nem hardware". Ezel leegyszerűsítettük a világot, és kiváltottuk a ketyere és az akármi szavakat két angol szóval.
Ebben az esetben sem hoztak megváltást az általános értelmű magyar szavak. Az egyik bökkenő az volt, hogy a *hardware* jelenthet egy gépet, egy vállalatnál megtalálható minden számítástechnikai eszközt, de akár egy önmagában használhatatlan monitorkártyát vagy memóriamodult is. Ugyanez a széles értelmezés igaz a *software*-re is.
De mit is tudunk használni? Alkatrész, eszköz, gép, géppark, hálózati elem, vagy megadhatjuk a szóban forgó elemet: kártya, alaplap, modul. Ha meg akarjuk adni, hogy legalább milyen gép kell egy program futtatásához milyen gép szükséges: igényelt/szükséges géptípus, gépkövetelmények, legkisebb kiépítés, gépigény stb. Ha általásnosan akarunk beszélni, használhatjuk a vas szót hibás alkatrészekre pedig hivatkozhatunk vadállatok nevével (például szarvas).
És a „minden más"? Program, alkalmazás, rendszer, modul, csomag, elem. Ha esetleg valakinek vannak további ötletei, kérem, írja meg nekem. Kíváncsi vagyok, találunk-e még szép magyar kifejezéseket?

License

Hoppá, még egy darázsfészek! Ezzel nem bírtunk megküzdeni, leginkább azért nem, mert nem találtunk rá olyan szót, ami értelmileg fedné a jelentését. A fő gond itt is az, hogy ugyanazt a szót használja az angol több fogalomra. Egyszerűsít, például a *license arrangement* és az *end user license* is gyakran egyszerűen csak *license*.
Itt is az látszik a leghasználhatóbb útnak, ha az egyes előfordulásokat egyesével, értelem szerint fordítjuk. A *license* valahol a szerződés és az engedély között van (sőt, van, ahol licencszerződésről beszélnek...), és a tartalmától függően lehet felhasználói, felhasználási, üzemeltetési, karbantartási stb. De melyiket is használjuk?
Ez ismét olyan kérdés, amit csak az idő old meg. Én addig is örömmel pártolom a felhasználási szerződést.

Cracker, Hacker, Geek

Többen kifogásolták hackerre adott fordítási javaslatomat. Igaz, többeknek viszont tetszett. Volt, aki azt írta, hogy a legjobb lenne, ha az egész fogalomkört fognánk és nagy ívben elfelejtenénk, keresnénk találó szavakat, és mondandónkhoz illesztve mindig a megfelelőt használnánk. Szomorúan intenék búcsút a betyárnak, a csibészt is hasonló sorsra szánták, de mielőtt végleg feladom, kérek mindenkit, ha jó ötlete van, ossza meg velem! Álljon itt a fogalmakról egy-egy rövid értelmezés (köszönöm a geek pontos felkutatását *Mészáros Gergelynek*):

- Cracker: Rendszerek, programok stb. feltörésére, védelmi vonalának hatástalanítására, áttörésére szakosodott ember, általában javak tisztességtelen úton történő szerzésének reményében tevékenykedik.
- Hacker: Általában szakmailag magasan képzett ember, többnyire rendszergazda, hálózati felügyelő vagy programozó, aki szabadidejében biztonsági kérdésekkel foglalkozik, rendszerek biztonsági réseinek feltárásában, a hibák kiküszöbölésében segít. Többnyire szívességből, nem anyagi megfontolásból és nem pusztító célzattal tevékenykedik.
- Geek: A számítástechnika világában élő, egy viszonylag zárt baráti körben mozgó, elsősorban az informatika világára figyelő ember. Lényegében bárki, akinek a hobbija, sőt, az élete a számítástechnika, és bármikor órákat el tud beszélgetni a legújabb fejlesztésekről. Már kaptam egy-két javaslatot: kockafejű, bitagyú, gépőrült, fanatikus. A hírek szerint a *geekest geek* Jon „maddog" Hall, a Linux egyik fő személyisége.
- Script kiddie: Egy új keletű kifejezés, azokra a személyekre mondják, akik unalmukban különböző rendszerbiztonsági hézagokat feltáró vagy kihasználó programokkal rendszereket akarnak feltörni. A név hangulatát erősíti, hogy általában gyermeteg lelkületű és céltalan (vagánykodó) emberekről van szó, akik a számítástechnikához nem is értenek olyan szinten, hogy komoly hasznot húzhassanak a betörésekből. Egy vicc is kering, ami állítólag az első script kiddie nyilvános megszólalása volt az egyik levelezési listán: „Vagesz, feltörtem a Pentagon gépét ezzel a progival, ami a listán lejött tegnap! De mondjátok csak, hogyan lehet lemásolni fájlokat a feltört gépről?"

Természetesen sok olyan fordítás van ebben a lapban is, ami számunkra is új. Mivel nem vagyunk tévedhetetlenek, könnyen lehet, hogy bizonyos szavakat tévesen vagy nem érthetően fordítunk. Sok olyan fordítás is előfordul, melyekről még nem kaptunk visszajelzést (ilyen szavak tucatjával vannak a 25. oldalon kezdődő *SDL gyorstalpaló* című cikkben, vagy a 40. oldalon kezdőző *Linux és videó*ban). Kíváncsi vagyok, hogy aki a boncolgatott témákhoz ért, a magyarított szavakat megérti-e, illetve aki nem ismerős a területen, rá tud-e jönni, hogy melyik fogalom mit takar?
Tényleg, valakinek nincs egy jó ötlete a set-top box-ra? Ez az eszköz egy „a tévé tetejére helyezhető" doboz, ami lehet videó, játékgép, zenedoboz, vagy bármilyen, általában a multimédia témakörébe tartozó szolgáltatást biztosít.

Mindenkinek kellemes és hasznos linuxozást kívánok ebben az új évezredben is!

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

Kapcsolódó címek

A Linuxvilág honlapja:
➲ http://www.linuxvilag.hu/

A Magyar Linux Projekt honlapja:
➲ http://magyar.linux.hu/

Programvadászat

Szemezgetés CD-mellékletünk tartalmából.

Kedves Olvasóink, immár mindkét CD-nken válogatásokat adunk közre a linuxos világ fontos és érdekes anyagaiból. Az első CD-n találhatók az újságban megjelenő cikkekhez szorosabban kapcsolódó programok és források, míg a második korongra a nagyobb anyagaink kerültek fel.

Hancom Office 1.0

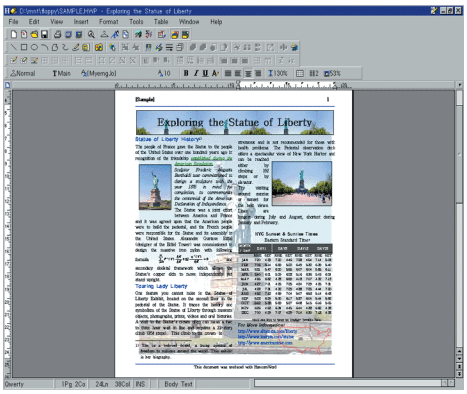
Az első korongon a legnagyobb terjedelmű anyag egy irodai programcsomag Koreából, a Hancom Office 1.0. Tartalma:

- Hancom Word R5 szövegszerkesztőt
- Hancom Sheet táblázatkezelőt
- HancomPainter rajzolóprogramot
- HancomPresenter bemutatókészítő.

Nagyon könnyű telepíteni, ha valakinek RPM-alapú rendszere van, mivel a telepítőcsomagban RPM formátumú fájlok találhatók. Debian alatt egy egyszerű trükkel szintén telepíthetjük.

Az `apt-get install rpm` parancs kiadásával rendszerünk feltelepíti az RPM csomagok kezeléséhez szükséges programokat és könyvtárakat. Az install héjprogramot grafikus felületen rendszergazdaként elindítva, a felhasználói engedély elolvasása után kiválaszthatjuk a számunkra szükséges összetevőket. A telepítés befejezésekor mi két különböző Linuxon is hibaüzenetet kaptunk, ennek ellenére a telepítés tökéletesen lezajlott. RedHat 7.0 alatt a program a Gnome menüjébe automatikusan beépítette magát.

➲ www.hancom.com



Manstyle

Gyönyörű megjelenésű html-segédleteket készíthetünk e programmal. Óriási segítség az automatikus oldalszámozás, a fejezetek szerkezetének felépítése és az automatikus tartalomjegyzék-készítés. Mindenkinek ajánlható, aki programjához, termékéhez vagy bármihez tökéletes leírást szeretne készíteni html formátumban bőséges támogatás felhasználásával.

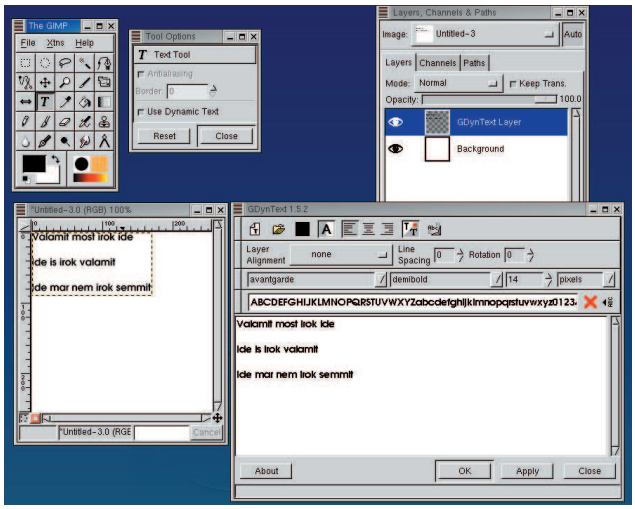
➲ http://manstyle.sourceforge.net/

Gimp 1.2

A GIMP a unixos világ Photoshopja, nem véletlenül érdemelte ki ezt a megtisztelő címet. Ingyenes és mesés tudású program. Több mint egyéves fejlesztőmunka eredménye az új 1.2-es változat. Számos újdonsággal bővült a program kínálata. Talán az egyik leghasznosabb új tulajdonsága a dinamikus szövegkezelés, dolgozhatunk a szöveggel, forgathatjuk, méretezhetjük, hozzáírhatunk, elvehetünk belőle, tehát teljesen szabadon szövegként kezelve munkálkodhatunk vele. Mindenképpen ajánlom mindenkinek a kipróbálását. Korongunkra mind forrásként mind pedig RPM formátumban felkerült.

➲ www.gimp.hu

➲ www.gimp.org



Xfree86 4.0.2

Az Xfree a Linux alapértelmezett grafikus felülete. Hálózati átjárhatóságának köszönhetően segítségével egy régi kiselejtezett számítógépből is hasznos internetes munkaállomást készíthetünk. A 3-as sorozathoz képest teljesen modulárissá tették a meghajtók elhelyezkedését a rendszerben, így már sokkal könnyebb új meghajtókkal bővíteni. A gyártók közül először az nVidia és a Matrox készített kártyáihoz ilyen modulokat. Ez a változat már támogatja a többképernyős rendszereket, amit xineramának hívnak. A Matrox oldaláról letölthető meghajtóprogrammal akár a duplafejes G400, G450-es kártyáinkat is használhatjuk két monitorral. Sajnos, még csak bétaváltozat, de tapasztalataim szerint így is megbízhatóan működik.

➲ www.xfree86.org

Ted 2.8

A Ted szövegszerkesztő egy X11 alá írt teljes szolgáltatású irodai alkalmazás. Nem kezel ugyan ttf (True Type Font) betűtípusokat, de az X11 betűit elfogadja, így egyszerűen elintézhetjük készletünk bővítését.
➲ http://www.nllgg.nl/Ted/

Multimédiás válogatás

- Broadcast 2000

Nagy tudású film- és hangfeldolgozó programcsomag. Linux alatt megbízható felületet ad a filmjeink gigabájtokban mérhető mennyiségű mozgókép- és hagfolyamainak feldolgozásához. A 28. oldalon átfogó cikket olvashatunk róla.

- Xmovie

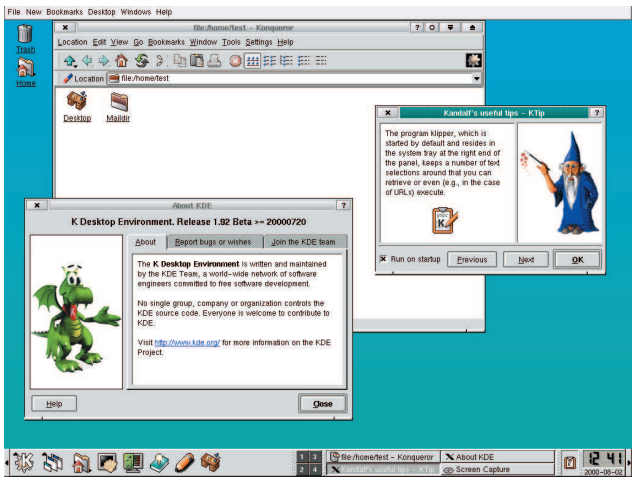
Quicktime-lejátszó Linuxhoz. Az RPM mellett a forrást is mellékeltük.

- Xmps

Gtk-s felületű, bőröket támogató MPEG1-lejátszó.

KDE 2.0.1

A mindenki által ismert és ünnepelt, esetleg szidalmazott ablakkezelő rendszer, a KDE legfrissebb változata, RedHat 7.0, SuSE 7.0, Mandrake 7.2 RPM formátumban és forrásként is. Sok nyelvi kiegészítéssel látták el, így mindenki a kívánt nyelven használhatja számítógépén. A KDE egy beépített irodai csomaggal is rendelkezik, a KOffice-szal, ennek részei például a KWord szövegszerkesztő, a KSpread táblázatkezelő, valamint a KPresenter bemutatókészítő.
➲ www.kde.hu
➲ www.kde.org
➲ www.koffice.org

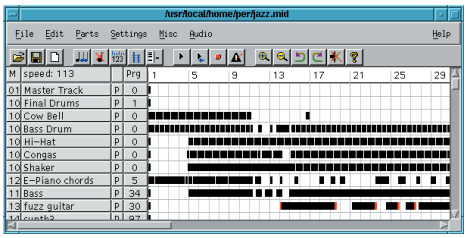


A multimédia szerelmeseinek

A Jazz++ széles körben használt és a zenészek között kedvelt MIDI sorozatszerkesztő (sequencer). Windows és Linux alatt egyaránt fut. Számos olyan tulajdonsága van, ami a megjelenéséig csupán a drága programok vásárlói számára volt elérhető.
Ilyenek például:

- a GM, GS és XG támogatás,
- a gyors és hatékony eseményszerkesztés,
- az audio/MIDI egyidejű feldolgozása,
- az audio mintaszerkesztő/feldolgozó,
- az igényes dobkészítő,
- a harmóniakereső/készítő,
- a dallamkészítő,
- az Arpeggio készítő,
- a keverő beállításainak elmentése a MIDI fájlba,
- a külső szinkronjel kezelése, valamint
- az eseménylistaszerkesztés.

Rendszerkövetelmények
Operációs rendszer: ELF Linux,
Lejátszáshoz és felvételhez szükséges OSS/Linux vagy ALSA hangkártya meghajtó és az általuk támogatott hangkártya.
(Az ALSA-rendszer alatt egyelőre csak a MIDI rendszer működik.)
Minimum 486-os processzor és 32 MB memória az ajánlott (csak MIDI-hez).



Lyx

A Tex/Latex rendszer széles körben elterjedt megoldás, tökéletes dokumentumok létrehozásához. Segítségével készíthetünk könyvet, matematikai képletgyűjteményt vagy akár kottát is. A szövegfájlok szerkesztését kevesen tanulják meg szívesen, így ehhez a rendszerhez számukra elkészítettek egy könnyen használható előtétprogramot. Ennek segítségével úgy használhatjuk, mint egy szövegszerkesztőt. A rendszer teljes tudását azonban nem érhetjük el ebből, mivel az alkotóknak annyira sokoldalú programot kellett volna létrehozniuk, ami szinte lehetetlen.
➲ www.lyx.org

Rendszermag

Hosszas fejlesztés és ellenőrzés után az új évezredet új rendszermaggal kezdi a linuxos társadalom. A 2.4.0 megbízható változat megtalálható a lemezen. Újdonságaiból: többprocesszoros gépek támogatása 32 processzorig, USB és új alkatrészek támogatása.
Természetesen felkerült még a 2.2-es sorozat 2.2.18-as friss kiadása is. Ebbe bekerült egy USB átültetés a 2.4-es sorozatból. Ezzel mindenki használhatja egyszerűbb USB-s eszközeit.
➲ www.kernel.org

Opera

Az Opera böngésző nem igazán régi motoros a Netscape és Internet Explorer mellett a piacon. Azonban lendületes fejlődésének köszönhetően egyre népszerűbbé válik, mert több operációs rendszerre is átültették és fejlesztik. Nem elhanyagolható tulajdonsága a másik két programmal szemben a mérete, ami jóval a versenytársaké alatt marad, de ez nem megy a tudása rovására. Kezeli a HTML 3.2, HTML 4.0 oldalakat, CSS 1 és 2 kiegészítéseket, FTP-helyeket, segítségével feltölthetjük fájljainkat, támogatja az XML és WAP oldalakat is. Telepítése nagyon egyszerű, mindenki kiválaszthatja a rendszerének megfelelő telepítőfájlt és az `rpm -i opera-xxx` vagy a `dpkg -i opera-xxx` parancs kiadása után már használhatja is a böngészőt.
➲ www.opera.com

Csontos Gyula
(Csontos.Gyula@linuxvilag.hu) a Linuxvilág hír- és CD-szerkesztője, valamint a www.linuxvilag.hu tartalomfelelőse.

„Olcsón Linuxot”

Linux-változatokat vehetünk olcsón a Cheeplinux cégtől. A hatlemezes Debian Gnu/Linux 2.2 csak tíz dollárba kerül. Különböző program-összeállítások is megrendelhetők, illetve megvásárolhatók:

- Linux Games a játékok szerelmeseinek (Quake, Robo Rally stb.)
- Linux Themes, témák minden mennyiségben a KDE, az Enlightenment, valamint a Windowmaker ablakkezelőkhöz,
- Linux Utilities, kiegészítő programok grafikus és karakteres felülethez,
- Linux Console Applications, karakteres felületű programok gyűjteménye,
- X/ KDE/ Gnome Applications, X, KDE és Gnome programok,
- Linux Megapack, a legutóbbi és legnépszerűbb Linux-változatokat tartalmazza 11 CD-n.

➲ http://www.cheeplinux.com

KDE

Knetfilter

A Knetfilter program segítségével a 2.4-es sorozatú rendszermag `netfilter` részének működését tudjuk felügyelni.

➲ http://expansa.sns.it:8080/knetfilter
➲ http://www.kde.hu
➲ http://www.kde.org
➲ http://www.koffice.org

Darwin

A Mac OS X a Darwin rendszermagot használja, ez FreeBSD-n és a Mach 3.0 eljáráson alapul. Jelenleg a Darwin PowerPC-alapú

Macintosh számítógépeken fut, folyamatban van az Intel-alapú PC-kre történő átültetése is.

➲ http://www.freeos.org
➲ http://www.apple.com
➲ http://www.opensource.apple.com

Játék

A Lokigames cég tervei között szerepel a Heavy Metal és a Rune játékok átültetése Linuxra. Így a Linux mint játékfelület egyre népszerűbbé válhat. A Lokigames oldalán több mint tíz játék érhető el, megrendelhető vagy letölthető a bemutatója bármelyik programnak. A híresebb játékok közül néhány:

- Quake III Arena,
- Descent 3,
- SimCity 3000
- Railroad Tycoon II,
- Heretic II.

➲ http://www.lokigames.com
➲ http://www.runegame.com
➲ http://fakk2.godgames.com
➲ http://www.quakearena.com

Újoncoknak

A linuxnewbie.com-on minden kezdő linuxos – különös tekintettel a Windowsról Linuxra váltó felhasználó – találhat megfelelő leírásokat.

➲ http://www.linuxnewbie.org

Kezdők segítésére magyar nyelvű források is találhatók az Interneten. Néhány HOGYAN (HOWTO) fordítása már elérhető, emellett a Linux-kezdő levelezőlistára való feliratkozással, vagy csak a levelezőlisták levéltárainak olvasásával is választ kaphatunk kérdéseinkre.

➲ http://magyar.linux.hu
➲ http://www.kde.hu
➲ http://www.gnome.hu
➲ http://www.debian.hu
➲ http://www.gimp.hu
➲ http://suselinux.ini.hu

Csomagkeresés

Ha Linux-rendszerünkhöz bármilyen csomagra van szükségünk, akkor hosszas keresgélés helyett használhatjuk az *rpmfind* szolgáltatását. Bármit szeretnénk elérni, ha rendszerünk az RPM csomagkezelőt használja, itt valószínűleg megtaláljuk. Hatalmas adatbázisa több mint százezer csomagot tartalmaz, amelyből másodpercek alatt kinyerhetjük a számunkra fontos adatot.

A szolgáltatást üzemeltető gazdagép egy „sima” PC, csak éppen több merevlemez van benne, mint egy átlagos gépben. Egy nyolc csatolós 3Ware IDE RAID kártyát tartalmaz, merevlemezein ext3 naplózó fájlrendszert használnak.

A Debian csomagjainak a kereséséhez is kapunk segítséget a Debian Linux honlapján. Kiválaszthatjuk például, hogy csak a csomagnevekben vagy a leírásokban is keressen, azt is megadhatjuk, hogy melyik változathoz keresünk csomagot – fejlesztői, ellenőrző, megbízható vagy mind.

Az ftp-kiszolgálókon történő keresgéléssel szemben mindkét hely legkedvezőbb tulajdonsága a gyorsaság.

Ha valamit megtaláltak a keresők, azt azonnal le is tölthetjük.

➲ http://rpmfind.net
➲ http://www.debian.org/distrib/packages

Földönkívüliek Linuxra

Ha szeretnénk segíteni a kutatókat a földönkívüli értelmes lények felkutatásában, akkor letöltve egy programot a mi számítógépünk is részese lesz a világűrből érkező rádióhullámok feldolgozásának. Természetesen gépünknek folyamatos internetkapcsolatra van szüksége, hogy az adatokat folyamatosan kapja és visszaküldhesse feldolgozva. A program csak akkor foglalja le a processzoridőnket, ha mi már nem dolgozunk a gépen. Így a felhasználó számára észrevétlen marad a program. Szinte minden operációs rendszer alá elkészítették már, Linux, BSD, Windows stb. alatt is segíthetünk számolni.

➲ http://seti.index.hu
➲ http://www.seti.org
➲ http://setiathome.ssl.berkeley.edu/

Mobil Linux

Több cég is fejleszt linuxos kisméretű PDA-gépeket, ilyenek például a Samsung Yopy, VTech Helio, Agenda VR3, Acer Slimmate, Compaq Ipaq.
Mindegyikük közös jellemzője, hogy valamilyen formában Linux operációs rendszer fut rajtuk, alkatrészeikben azonban a legkülönfélébb összeállításokat találhatjuk meg. Van közöttük 206 MHz-es StrongARM processzoros, 66 MHz-es MIPS processzoros gép is, memóriaméretük pedig a 8 MB-tól egészen 48 MB terjed. Ekkora különbségek mellett természetesen a rajtuk futtatható programok is eltérnek egymástól.
A következő oldalon bőséges segítséget kaphatnak azok, akik laptopjukat Linuxszal szeretnék használni: ➲ http://www.cs.utexas.edu/users/kharker/linux-laptop
A Tuxtops cégtől pedig előre telepített Linuxszal vehetjük meg hordozható gépünket, vagy ha már birtokunkban van ilyen eszköz, akkor megvásárolhatjuk a kifejezetten hordozható gépekhez hangolt Linux-változatot, legyen az RedHat vagy akár Debian.
➲ http://www.mobilelinux.com/linuxpdas.html
➲ http://www.mobilelinux.com/linuxnotebooks.html
➲ http://www.tuxtops.com

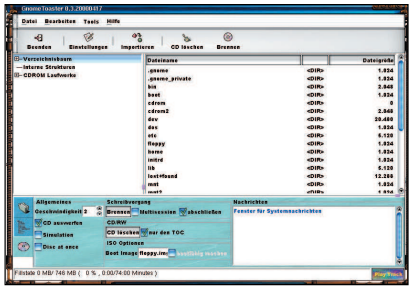
Gnome

gmmusic

Grafikus felület PostgreSQL 7 adatbázishoz. Segítségével a teljes zenei gyűjteményünket nyilvántarthatjuk, legyen szó CD-ről, lemezről, kazettáról vagy akár MP3-ról.

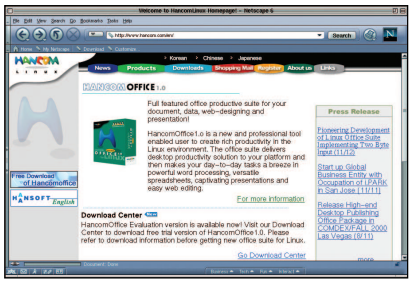
Gnome Toaster

Grafikus felület a cdrecord, cdrdao, cdda2wav, mkisofs CD-író programokhoz. Segítségével mindenféle CD-t készíthetünk, például írhatunk MP3 fájlokból audioCD-t.
➲ http://www.gnome.hu
➲ http://www.gnome.org



Irodai csomag Távolkeletről

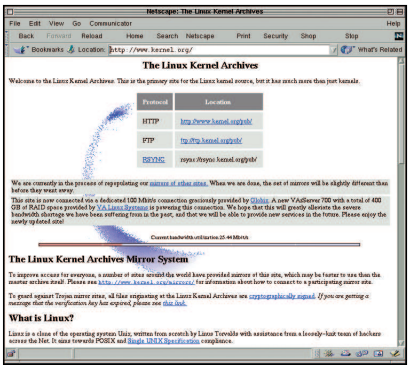
A HancomLinux cég a koreai Haansoft, Inc. leányvállalata. A HancomLinux cég készítette az első kínai nyelvű szövegszerkesztőt Linuxhoz. Az általuk készített irodai programcsomag kevésbé ismert, ami valószínűleg annak „köszönhető”, hogy kereskedelmi termék, tehát fizetni kell érte. A cég honlapjáról letölthető a 60 napig működő bemutatóváltozat. Több nyelvi változata is létezik:



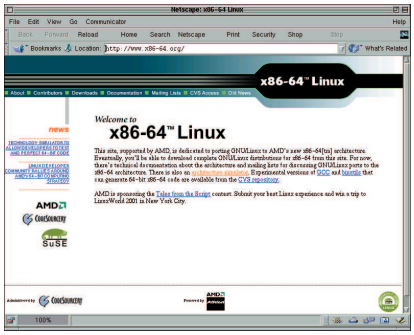
koreai, japán, kínai és angol. Letölthetjük a Hancom Word R5 szövegszerkesztőt, HancomSheet 1.0 táblázatkezelőt vagy akár az egész HancomOffice 1.0 programcsomagot.
➲ http://www.hancom.com

Rendszermag

Hosszú várakozás és ellenőrzés után megjelent a 2.4-es sorozatú rendszermag. Újdonságai közé tartozik, a többprocesszoros rendszerek támogatása akár 32 processzorig. Ezenkívül az Intel hatvannégy bites Itanium processzorát is támogatja, valamint teljesebb



lett az USB-s eszközkezelés. Rengeteg új meghajtó került bele, és hamarosan várható e rendszermaggal készült Linuxkiadások megjelenése. A RedHat máris bejelentette, hogy tavasszal jelentetik meg a 2.4-es rendszermagon alapuló Florence kódnevű operációs rendszerüket.
➲ http://www.kernel.org
Alighogy megjelent a 2.4-es rendszermag, a fejlesztők máris elkezdték összeállítani a 2.5 sorozat „kívánságlistáját”. Ezek között szerepel a MOSIX, mely fürtözéses eljárással egészíti ki a rendszermagot.
➲ http://www.zdnet.com/zdnn/stories/news/0,4586,2669858,00.html



64 bites Linux

Az AMD által fenntartott internetes oldalon, az új 64-bites AMD processzor és a GNU/Linux kapcsolatáról kaphatunk tájékoztatást. Letölthetünk egy szimulátort is, ezen kipróbálhatjuk milyen 64 bites környezetben dolgozni. Megfigyelhetjük a 32 bites programok futását, a rendszermag hibaüzeneteit, ha engedélyezett a 64 bites mód. Ez a csomag csak bináris RPM csomagként érhető el SuSE és RedHat Linuxhoz, mivel tartalmaz néhány olyan kódrészletet is, amit az AMD nem ad ki. Továbbá letölthető még a gcc-cross-x86_64, 64 bites GNU C fordító is.
➲ http://www.x86-64.org

BSD-körkép

A közelmúltban a BSD operációs rendszereket használó szakemberek és érdeklődők megalapították a Magyar BSD Egyesületet. Írásunk ennek kapcsán megkísérli bemutatni a honi helyzetet.

Az egyesület célkitűzései

A BSD, illetve a BSD-alapú Unix-szerű operációs rendszerek és az alkalmazásaival kapcsolatos ismeretek széles körű terjesztése.
– A programok, leírások átültetése magyar nyelvre.
– A BSD-alapú programok, rendszerek fejlesztése, illetve kutatás.
– Kapcsolattartás külföldi szervezetekkel. Minden érdeklődőt és jelentkezőt szívesen fogadunk az egyesületünk honlapján ➲http://www.bsd.hu

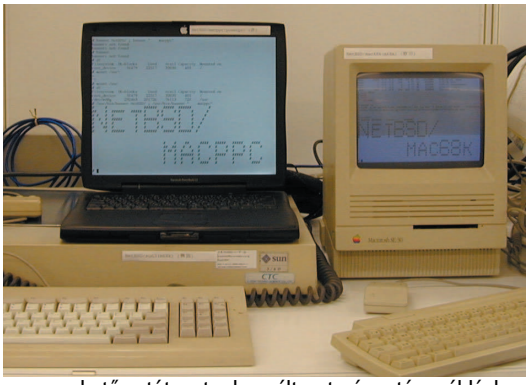
A BSD rövid története

A mai BSD-rendszerek a Berkeley Egyetem Computer Systems Research Group (CSRG) által fejlesztett rendszer utódjának tekinthetők. Ez az AT&T-féle Unix rendszermagjának és programjainak továbbfejlesztése, illetve kiegészítése. Az első BSD-kiadások főleg csak felhasználói programokból álltak, de az események döntő fordulatot vettek, amikor a CSRG megbízást kapott a DARPA-tól (Defense Advanced Research Projects Agency) az ARPANET hálózatuk kapcsolattartó protokolljainak a továbbfejlesztésére. Az új protokollt egyszerűen Internet Protokollnak hívták, ez a későbbiek során TCP/IP néven vált ismertté és terjedt el. Ma a legszélesebb körben ezt a protokollt használják a hálózatokban. Az első, széles körben terjesztett IP-megvalósítás a 4.2BSD része volt, mely 1982-ben jelent meg. Az IP kifejlesztése hatalmas lépés volt mind a BSD, mind az ARPANET számára, és ez lett az Internet kezdete. A mai operációs rendszerekben a TCP/IP kezelését – szinte kivétel nélkül a BSD-ből vették át.
A nyolcvanas évek közepén több új számítástechnikai cég tűnt fel. Sokan közülük inkább a Unix alapkód szerződéses felhasználását részesítették előnyben, mint hogy saját operációs rendszer készítésébe fogjanak. (Például a Sun Microsystems egy 4.2 BSD-változat alapján készítette el a SunOS-t.) Az AT&T cég kereskedelmi forgalomba hozta a Unixot, először a System III-mal, ezt pedig rövid időn belül követte a System V. A System V alapkódja nem tartalmazott hálózatkezelést, így minden változat BSD-részeket tartalmazott; beleértve a TCP/IP hálózatkezelést, valamint az olyan segédeszközöket is, mint a vi szerkesztő vagy a csh shell (ezeket a bővítéseket összességükben Berkeley Extensionsnak hívták). A BSD szalagok AT&T kódrészleteket tartalmaztak, így szükség lett volna (AT&T) Unix felhasználási szerződésre. 1990-ben a CSRG anyagi gondokkal küzdött, a BSD projektet a leállítás, megszüntetés veszélye fenyegette. Így a csoport néhány tagja elhatározta, hogy kiadja a BSD kódokat (ez ugyanis nyílt forrású) az AT&T kódrészletek nélkül. Ez végül a Networking Tape 2-vel történt meg (Net/2). A Net/2 nem volt teljes operációs rendszer, ugyanis a rendszermag körülbelül húsz százaléka hiányzott. A CSRG egyik tagja, *William F. Jolitz* megírta a hiányzó részeket, és 1992 elején kiadta 386/BSD néven. Ugyanebben az időben néhány másik volt CSRG-tag kereskedelmi céget alapított Berkeley Software Design, Inc. néven (BSDI), és kiadta egy BSD386 nevű operációs rendszer bétaváltozatát (Figyelem, a 386BSD és a BSD/386 tehát nem ugyanaz!). A BSD/386 név azóta BSD/OS-re változott.
A 386/BSD soha nem vált stabil operációs rendszerré. Ehelyett két projekt folytatta a fejlesztést: a NetBSD és a FreeBSD. A FreeBSD 1.0 1993 novemberében, a NetBSD 1.0 pedig 1994 novemberében jelent meg. Szó volt ugyan a két projekt egyesítéséről, de ez nem vált valóra. 1995-ben egy újabb projekt indult: a NetBSD-ről levált OpenBSD.

BSD-kiadások

Minden BSD-projekt kiadások (release) formájában teszi elérhetővé a rendszerét. Először is minden kiadásnak van saját változatszáma, például: 1.4.1 vagy 4.1.
a változatszámot utótag is követi:

1. A rendszer fejlesztői változata a Current nevet viseli. A FreeBSD Current ága változatszámot is kap, például: FreeBSD 5.0-Current. A NetBSD egy



egybetűs utótagot rak a változatszám után, például: 1.4.3G. Az OpenBSD nem rendel hozzá változatszámot: OpenBSD-Current. Minden új fejlesztés ebbe a fejlesztői ágba kerül.

2. Bizonyos időközönként, negyed- vagy félévente megjelenik egy kiadás, amiből megvásárolható/letölthető CD-ROM, tehát kiadás lesz. Például: OpenBSD 2.6-Release, FreeBSD 4.1-Release. A NetBSD *javító* kiadásokat (release-eket) is megjelentet, melyeket egy harmadik számmal lát el, például: 1.4.2.
3. Amikor a release ágban fellelt hibákat kijavítják és megjegyzéseket fűznek hozzá a CVS-fába, a végeredmény neve FreeBSD és OpenBSD esetén stable-ág (NetBSD-nél továbbra is release) lesz. Ebből következik, hogy a FreeBSD-nél és OpenBSD-nél a release egy pillanatnyi állapota a forrásfának, míg a stable egy fejlesztési ág jelölése. Például a FreeBSD 4.1-Release-ből 4.1-Stable lesz, mely tartalmazza a 4.1-Release óta kiadott hibajavításokat, illetve néhány olyan új lehetőséget, ami esetleg bekerült a stable-ágba. Bármikor a percre időszerű stable-ra frissíthetjük rendszerünket, nem kell a kiadásokra várnunk! A kiadás előtt a stable ágat átnevezik betára, ebből lesz aztán az ellenőrzés után a következő release. OpenBSD-nél a stable ágat szokás még foltágnak is hívni (patch branch).

A BSD projektek a teljes operációs rendszert karbantartják, ennek az a látható eredménye, hogy a változatok közötti frissítések nagyon gördülékenyen zajlanak, gyakorlatilag a könyvtárak hibátlanul működnek (a Ports Collectionnek köszönhetően még az operációs rendszeren kívül eső külső fejlesztésű programoknál sem jelentkeznek hibák). A nagyon régi alkalmazások számára úgynevezett együttműködési (compatibility) könyvtárakat telepít a rendszer, így gond nélkül futnak többéves alkalmazások is, melyeket több nemzedékkel korábbi könyvtárakkal fordítottak.

A BSD felhasználási szerződésről

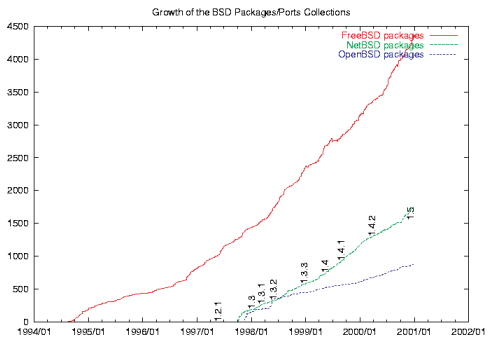
A BSD és a BSD-alapú rendszerek egyik legnagyobb előnye: a programok széles körű, szabad felhasználásának lehetősége. Láthatjuk, hogy a BSD kódokat rendkívül széles körben használják akár üzleti, akár pénzintézeti, akár beépített rendszerekben. Fontos megérteni a nyílt forráskódú programok (Open Source) készítésekor az egyik legtöbbet használt BSD felhasználási szerződés lényegét.
A BSD felhasználási szerződésű program szabadon felhasználható és terjeszthető, viszont a program forráskódját nem kell közreadni, mindössze meg kell említeni, hogy milyen kód alapján dolgoztunk (Ez a program Free BSD kódrészleteket tartalmaz.).
Miért jó ez? Képzeljük el, ha a TCP/IP-t használó operációs rendszereknek közzé kellett volna tenniük a forráskódot? Mennyivel egyszerűbb volt egy működő programot átvenni. Hosszan lehetne sorolni azokat az alkalmazásokat, hálózati berendezéseket, melyek mind-mind vagy BSD alapokra épülnek, vagy egyszerűen BSD operációs rendszert használnak.

A BSD operációs rendszerek fejlesztési modellje

A BSD operációs rendszerek esetében a fejlesztési irányok és célok meghatározását, valamint a munka irányítását az úgynevezett Core Team (belső mag) végzi. Ők a legtermékenyebb fejlesztők közül kerülnek ki.
A fejlesztés CVS-alapú forrásfában történik. Az egyszerű közreműködőknek, fejlesztőknek nincs írási joguk.
A forrásfába csak az írási joggal rendelkező fejlesztők (comitters) tudják a módosításokat beépíteni. Mielőtt ezeket beépítik, általában más fejlesztők ellenőrzik. Minden módosítás és frissítés könnyen követhető.
A rendszeralkalmazások frissítésére, hibajavításokra is a CVS szolgál. Így elmondható, hogy rendkívül hatékony a rendszer fejlesztésének követése.

A mai BSD-k

A mai BSD-rendszerek megegyeznek az átgondolt tervezésben, az erőteljes, rendkívüli terhelhetőségben,



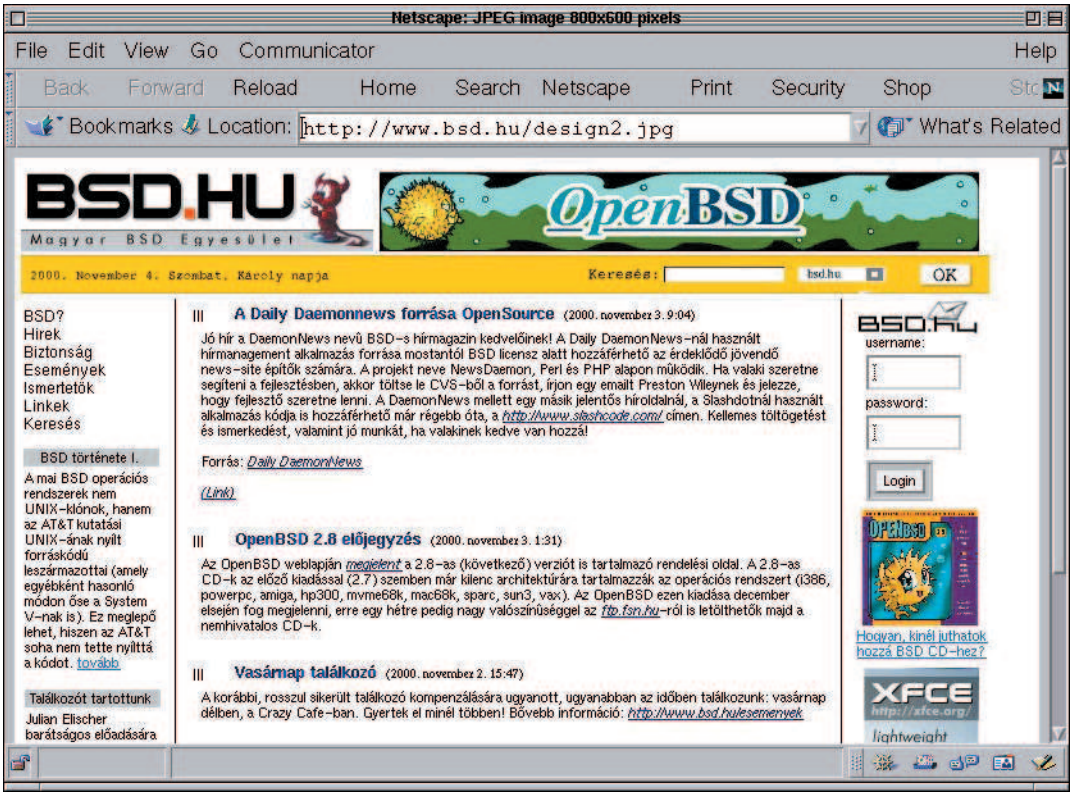
a legkorszerűbb hálózati szolgáltatásokban (Ipv6, IpSec), titkosításban (OpenSSH, Kerberos). Futtatható rajtuk BSD, BSD/OS, SunOS, Linux, SCO binárisok, és több ezer alkalmazás vált elérhetővé BSD-re.

FreeBSD

A mai BSD-rendszerek legismertebb tagja a FreeBSD. Célkitűzései az egyszerű kezelhetőség, a legnagyobb megbízhatóság elérése, valamint a gépből kihozható legnagyobb teljesítmény kihasználása. Jelenleg x86 és Alpha gépeken érhető el, ez utóbbin teljes 64 bites rendszerként viselkedik. Más géptípusokra is folyik az átültetése. Jelenleg az IA64 felépítés támogatása is előrehaladott állapotban van. Minden jel arra mutat, hogy a FreeBSD egy szervezett módon fejlesztett, egységes, megbízható operációs rendszer, mely megállja a helyét a legváltozatosabb terhelési és felhasználási körülmények között is. Előszeretettel használják internetszolgáltatók és tartalomszolgáltatók webkiszolgálóként, de más célra is alkalmazható. A legnagyobb ftp- és webkiszolgálókon találkozhatunk a FreeBSD-vel, azonban nem csak e szerepkörben állja meg a helyét, hanem a memóriakezelő rendszer előnyei az otthoni felhasználásnál is megmutatkoznak, hiszen a grafikus felületen futó programok is a sok memóriát igénylő alkalmazások közé tartoznak.
A programkészítők szívesen átültetik alkalmazásaikat FreeBSD-re (ahogyan más BSD-kre is), ugyanis az operációs rendszer több mint húszéves unixos múltjának köszönhetően könnyű átírni rá a programokat. Alkalmazásának példái jól mutatják előnyös tulajdonságainak kiaknázását, a legnagyobb forgalmat bonyolító és a leghosszabb üzemidejű internetes kiszolgálók terén (➲ftp.cdrom.com, yahoo.com).
A ftp.freesoftware.com kiszolgálóról egy nap például két terabájt mennyiségű adatot töltöttek le. A FreeBSD-hez kapcsolódik a TrustedBSD, mely az operációs rendszer biztonsági, titkosítási bővítéseit szolgálja. A FreeBSD jelenlegi stabil változata: 4.2. A következő változatban (5.0) újraírják az SMP támogatást, ennek kódrészleteit a BSD/OS-ből vették át.

NetBSD

A NetBSD a FreeBSD-vel csaknem egy időben alakult ki. A fejlesztés fő célja egy géptípusfüggetlen, átgondoltan megtervezett és kiérlelt megoldásokon építkező operációs



rendszer létrehozása, amely szabadon felhasználható, ezért a rendszermag csak BSD felhasználási szerződés alá eső részeket tartalmazhat. „NetBSD a legkisebb Windows CE-s kéziszámítógépektől a régi Amiga és VAX gépeken át, a Sega Dreamcaston keresztül az AlphaServerekig nagyon sokféle géptípuson fut, az egységes felhasználási szerződés pedig lehetővé teszi, hogy szervezetek, cégek saját eszközeikben felhasználják, például beágyazott rendszerekben, anélkül, hogy kérésre a saját hozzátett forráskódjukat ki kellene adniuk. Ennek köszönhetően például a NASA is szívesen használja bizonyos (zárt) célokra.
Több cég is alkalmazza hálózati eszközökben, útválasztókban is ➲http://www.errnet.com, ezenkívül jól megállja a helyét webkiszolgálóként is (➲http://www.saab.com). Ajánlható a régi számítógépek használatához is, mivel a rendszer az erőforrásokkal rendkívül gazdaságosan bánik, emellett megtalálható benne minden olyan eszköz, ami a mai követelményeknek megfelel.
A NetBSD jelenlegi stabil változatszáma 1.5, jelenleg 30 felületre érhető el.

OpenBSD

Az OpenBSD a NetBSD projektről vált le, célja egy biztonságos kiszolgáló operációs rendszer megteremtése. Az ő művük az OpenSSH is, de sok más hibajavításukat is átveszik különböző operációs rendszerek.
Újabban az Apache webkiszolgálót is beillesztették, azaz az OpenBSD forrásfában fejlesztik tovább. Az OpenBSD-t leginkább tűzfalként használják.
A jelenlegi stabil változat a 2.8, és 11 felületre érhető el. Használja többek között az Amnesty International VPN-megoldásaihoz és az Adobe programóriásnál tűzfalként.

BSD/OS

A BSD/OS a BSDI cég kereskedelmi terméke, a fejlesztés leginkább a FreeBSD-hez áll közel. 2000 első félévében összeolvadt a BSD/OS-t fejlesztő BSDi és a FreeBSD-t fejlesztő Walnut Creek CDROM, ennek eredménye a két operációs rendszer közötti kódcsere és a közös fejlesztés.

Naptár

Multics	1965.
Unix	1969 nyara, DEC PDP-7,
1BSD	1977 vége
	1978. 03. 09., PDP-11, Pascal, ex, 30 szabad másolatot kiküldtek, valamint 35 szalagot eladtak 50 dollárért,
4.2BSD	1983. szeptember
4.3BSD	1986. június
4.3BSD NET/1	1988. november
4.3BSD NET/2	1991. június
386BSD 0.0	1992. február
4.4BSD	1993. június 1.
FreeBSD 1.0	1993. november
NetBSD 1.0	1994. október 26.
386BSD 1.0	1994. november
OpenBSD 2.0	1996. október 18.
FreeBSD 4.2	2000. november 21.
OpenBSD 2.8	2000. december 1.
NetBSD 1.5	2000. december 6.
OpenBSD 2.8	2000. december 1.
FreeBSD 4.2	2000. november 21.

Az új cég továbbra is a BSDi nevet viseli. A BSD/OS jelenlegi stabil változatszáma aktuális verziószáma 4.2.

Darvin

A BSD-család legifjabb tagja a Darwin, ismertebb nevén a Mac OS X (az Aqua, azaz a grafikus felület nélküli Aqua). A FreeBSD alapjaira épül, így minden esély megvan rá, hogy egy nagyszerű PowerPC-s BSD rendszer jöjjön létre.

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

Pontrendszert indítunk!

Ha valaki el akar kezdeni linuxozni, az első nagy gond, hogy ha nem elég jó angolból (esetleg németből), akkor annyi esélye van, mint egy muslicának a sör-virsli versenyen. Ezért aztán fontos, hogy minél hamarabb, minél színvonalasabb hazai linuxos leírások, kézikönyvek, súgók jelenjenek meg. Mivel ezeket a fordításokat idáig szintén lelkesedésből, magánszorgalomból végezte mindenki, nagyon sok fehér folt van még. Az egyik fontos terület a HOGYAN-ok fordítása, melyek tömören egy-egy hiba elhárításával, egy-egy gond megoldásával foglalkoznak. A Linuxvilág szeretné a maga eszközeivel ösztönözni a magyar anyagok készítését, ezért egy pontrendszert hirdet meg. A pontok később ajándékokra, könyvekre, előfizetésre vagy akár pénzre válthatók. A tervezet megtekinthető és véleményezhető a Linuxvilág honlapján. ➲http://www.linuxvilag.hu/

„Az alkatrészek, a programok és a sávszélesség költségei nullára fognak csökkenni. Ha ez bekövetkezik, a beágyazott alkalmazások számára megszűnik minden határ. Semmi okunk nincs tehát arra, hogy készülékeinket ne ruházzuk fel értelemmel. A program már adott hozzá, Linuxnak hívják."
– Michael Tiemann, RedHat

Linux-index

1. Az XML felhasználásának növekedése a Linux-fejlesztők között, 2000. szeptemberhez képest: 75%
2. A megkérdezettek között az XML-t is használó Linux-fejlesztők aránya: 28%
3. A megkérdezettek között munkájuk legalább feléhez XML-t használó Linux-fejlesztők aránya: 27%
4. A megkérdezettek között a Javát használó Linux-fejlesztők aránya: 54%
5. Az elmúlt 25 évet nézve kevesebb klubösszejövetelre járók aránya: 58%
6. Az elmúlt 25 évet nézve a családi vacsorák csökkenésének aránya: 33%
7. A baráti összejövetelek csökkenésének száma az elmúlt 25 évet nézve: 45%
8. Akik szerint csökken a halálozás esélye, ha belép egy klubba: 50%
9. Az az év, amikorra a nyílt forráskód várhatóan teljesen megváltoztatja a programipart: 2004
10. Kapcsolatban nem lévő Zelerate (eredetileg Open Sales) alapítók száma, akiknek vezetékneve Ferber: 2
11. Az Egyesült Államok vállalkozói pénzalapjából származó nyereség, az 1999. év utolsó negyedévében: 59,4%
12. Az Egyesült Államok vállalkozói pénzalapjából származó nyereség, a 2000. év első negyedévében: 23,1%
13. Az Egyesült Államok vállalkozói pénzalapjából származó nyereség, az 2000. év második negyedévében: 3,9%
14. Vállalkozói tőkések által teremtett új pénzalap, 2000 október közepén: 64 000 000 000 dollár
15. Vállalkozói pénzalapból származó nyereség tervezett százaléka 20 év alatt 2000. 06. 30-tól: 19,9%
16. A Priceline részvények árfolyama, ahogy azt Henry Blodget-nek *Merrill Lynch* internetelemző jósolta: 150 dollár
17. A Priceline részvények árfolyama 2000. április 22-én: 76,38 dollár
18. A Priceline részvények árfolyama 2000. április 30-án: 165 dollár
19. A Priceline részvények előre jelzett árfolyama 2000 májusában *Jamie Kiggen* szerint, a DLJ-t követően: 190 dollár
20. A Priceline részvények árfolyama 2000. október 17-én: 5 dollár

Források:

- 1–4: Evans Data Corporation (2000. szeptember 26.)
- 5–8: BowlingAlone.com
- 9: Wired, idézet a Forester Researchtől
- 10: Zelerate
- 11–13: Venture Economics
- 14: San Jose Mercury News, idézet a Venture Economicstól
- 15–20: CBS.MarketWatch.com

LJ-történelem

A Linux Journalban hat évvel ezelőtt megjelent a Beszélgetés Linus Torvaldsszal című cikk, melyben Linus elmondta, hogy Canberrában megharapta egy pingvin, szereti az ír sört, és a Freakix helyett inkább Linuxra keresztelte az új operációs rendszer 1.2-es változatát. Linus a beszélgetésben felfedte a hosszú távú tervét is: „Világuralom. Gyorsan."

Ők mondták

„A kibertér legborzasztóbb szavai: tönkrement a kapcsolatom." *(Bob Metcalfe)*

„A programokra már nem vonatkozik a Moore törvény." *(Ellen Ullman)*

„A Double-Click Inc. reklámóriás és a hordozható eszközök tartalomszolgáltatója, az OmniSky kellemetlen szoruláskénti jelentkezett a vezeték nélküli izékre reklámokat küldözgető cégek egyre erősödő tünetegyütteséhez." *(Tom Matrullo)*

„Az olaj végnapjai elkezdődtek." *(Mike Bowlin, az ARCO vezérigazgatója)*

„Minden bővítés csonkolással is jár." *(Marshall McLuhan)*

„A tehetség mindig tudatában van saját nélkülözhetetlenségének, és ebből következően semmit sem hajlandó megosztani másokkal." *(A. Szolzsenyicin)*

„Ha az autóipar az utóbbi ötven évben a számítástechnikával azonos ütemben fejlődött volna, az autók ma két fillérbe sem kerülnének és fénysebességnél is gyorsabban száguldanának." *(Ray Kurzweil)*

„De ki akar olyan autóba ülni, amelyikbe egyedi üzemanyag kell, és használat közben naponta tízszer is lefagy?" *(Egy amerikai autógyár szóvivője)*

„A Linux a General Motors számára is fontos, nem csak a gm.com-nak." *(Larry Augustin, VA Linux)*

„Ahol káosz van, ott több lehetőség is van." *(Bryan Sparks, Lyneo)*

„Káosz nélkül nem lenne móka és kacagás." *(Jonah Kinata)*

„Egyre több új készülék jelenik meg, és egyre nagyobb az esély is arra, hogy ezek az eszközök kapcsolatba is lépjenek egymással." *(Michael Tiemann, RedHat)*

Munkaerő-piaci folyamatok

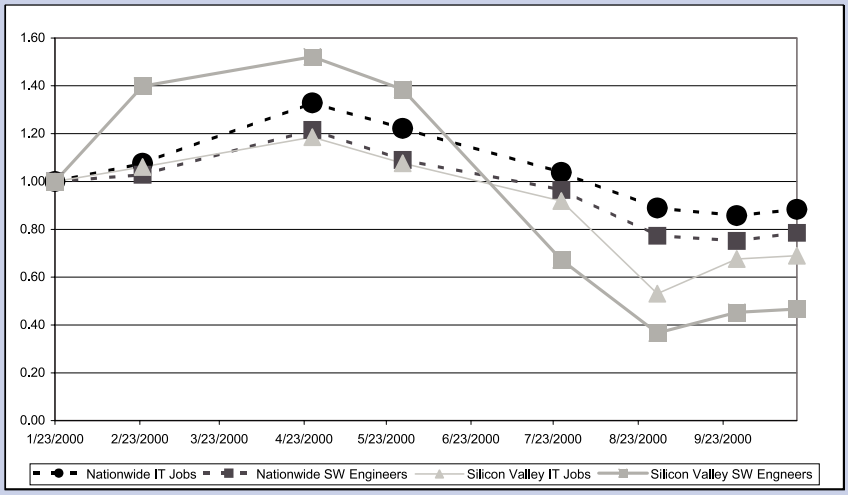
Múlt hónapban a felkínált programmérnöki állások száma enyhén emelkedett, különösen itt a Szilícium-völgyben. Ez a hónap is bizonyítja, hogy ez nem csak egy szerencsés véletlen volt.

Munkalehetőségek

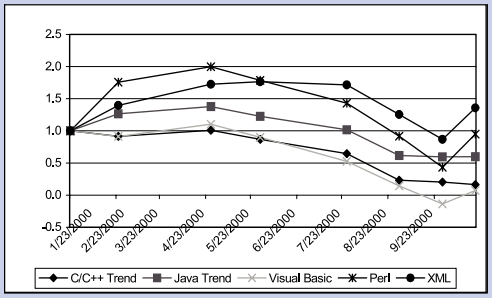
Még mindig folyamatos igény van általában az IT munkatársakra, és főként a programmérnökökre. Bár úgy tűnik, hogy ez a kereslet kezd kiegyenlítődni. A programmérnökök iránti kereslet az IT állások növekedésével arányosan növekszik. (Megjegyzés: az *1. ábra* a 2001. januári állások számához van igazítva. Ez az érték 2000 januárjában 1-nek tekinthető.)

A programnyelvtől függő kereslet irányvonalai

2001 elejétől kezdve az összes jelentősebb programnyelv iránti kereslet érdekesen alakul. A *2. ábra* a napjaink legnépszerűbb programnyelvei iránti keresletet mutatja. A kereslet tavaly április és május között érte el a csúcspontját. Azóta a kereslet alábbhagyott, de a múlt hónapban a dolgok érdekes fordulatot vettek. Hirtelen megnőtt az internetes nyelvek iránti kereslet, és a Java, C/C++ iránt csökkent. Véleményem szerint a változás mögött a különböző rendszerek összekötésének igénye áll, melyet hatékonyan meg lehet valósítani a megosztott adatszabványok és az általános célú parancsnyelvek (például a Perl) használatával.



1. ábra



2. ábra

A Visual Basic kakukktojás, a kereslet nőtt, valószínűleg az új prototípusok elkészültének köszönhetően.
➲www.accu-usa.org

Reginald Charney az Association of C and C++ Users amerikai tagozatának vezetője.

IBM

Legutóbbi beszélgetésünkkor *Steve Solazzo*, az IBM Global Linux Strategy alelnöke elmondta, hogy vállalata eladott a Lawson Inc. nevű japán gyárnak több mint 15 000 RedHat Linuxot futtató eServert. Solazzo szerint ez volt minden idők legnagyobb Linux-értékesítése. Solazzo az IBM győzelmeként értékelte a tényt, és hangsúlyozta az esemény jelentőségét a Linux közösség számára, mondván, a japánoknak nagyon fontos a minőség és a megbízhatóság. A tény, hogy egy ilyen méretű vállalat a vásárlói oldalra is ezt a rendszert választotta, bizonyíték arra, hogy a Linux „felnőtt a nagyvállalati szintre”, és ezzel igazolta, amit a linuxos közösség már régóta hangoztat.

Új termékek

SuperServer 6010 és 5010 kiszolgálók

A Supermicro bemutatta a kétprocesszoros 6010 és az egyprocesszoros 5010 1U kiszolgálóit. Mindkét típus két gyors, alaplapra épített Ethernet hálózati kártyát tartalmaz, amely hibatűrő, valamint alkalmazkodó terheléselosztásra képes széles sávú hálózati környezetben. A SuperServer 6010H két Intel 370 FCPGA Pentium III processzort képes fogadni 1 GHz-ig, a ServerWorks ServerSet III HE-SL lapkakészlet legkevesebb 4 GB kétutas PC133/100 SDRAM memóriát támogat, van benne 64/32 bites 66/33 MHz-es PCI sín, Adaptec Ultra 160 kétcsatornás SCSI csatoló és két Intel 82559 hálózati csatoló.
Kapcsolat: Supermicro
e-mail: marketing@supermicro.com,
➲ http://www.supermicro.com/.

Infinistore 1.2 változat

A Grau Data Storage Inc. bejelentette az Infinistore Virtual Disk (IVD) hálózati tárolókiszolgálójának 1.2-es változatát, ez támogatja a baleset utáni helyreállítást és távolról felügyelhető. Az eszköz egyszerre alkalmazza a RAID merevlemezes tárolást és a rendkívül nagy kapacitású szalagos tárolást, kiegészítve egy beépített tároláskezelő programmal. Több IVD is kezelhető egy helyen vagy szétszórva.
Kapcsolat: Grau Data Storage Inc.
e-mail: info@GrauData.com,
➲ http://www.GrauData.com/.

Az SANavigator ingyenes próbaváltozata

A Connex SANavigator nevű, összetett tárolótelepek kezelésére szolgáló programjának 30 napos próbaváltozata letölthető a
➲ http://www.sananivator.com/
címről. A SANavigator könnyen használható grafikus felületű program, amit a vállalati tároláskezelés igényeihez fejlesztettek ki, és képes valós időben megjeleníteni az egész tárolótelepet. Letöltés előtt a felhasználóknak ki kell tölteniük egy bejelentkezési űrlapot.
Kapcsolat: Connex, Inc.
e-mail: sales@connex.com,
➲ http://www.sanavigator.com/.

Ascendence-fürtök

Az Atipa Corporation bemutatta a Beowulf számítógépfürtök (telepek) egy új változatát, az Ascendence sorozatot. A telepek négy vagy nyolc csomópontos kiszerelésben kaphatók, Intel, AMD és Alpha processzorokkal. A vevő választhat három nagy sebességű összekapcsolási lehetőség közül, ezek a Wulfkit, a Myrinet és a 100 MB Ethernet. A beszereléshez választható 1U, 2U és 3U állványba szerelhető ház, vagy teljes magasságú toronyház, működés közben cserélhető tápegységekkel.
Kapcsolat: Atipa Corporation
e-mail: info@atipa.com,
➲ http://www.atipa.com/.

A Synopsys tervezőeszköz-csomagja

A Synopsys megjelentette magas szintű tervezést segítő eszközkészletét, amelyet a lapkatervezésben és más DFT eljárásokban jelentkező egyre növekvő igény kielégítésére szánnak. A csomag tartalmazza a Design Compilert, a PrimeTime statikus időzítéselemző eszközt, a Sirocco nagy teljesítményű VHDL-szimulátort és a Module Compilert. Az eszközkészlet mind a tervezéshez, mind az ellenőrzéshez tartalmaz segédprogramokat. Az eszközök működését áttekintő grafikus bemutató megtekinthető a Synopsys honlapján.
Kapcsolat: Synopsys, Inc.
➲ http://www.synopsys.com/.

Debian-alapú 2U kiszolgálók

A VA Linux most 2.2-es Debian GNU/Linux programmal kínálja a 2200-as sorozatba tartozó 2U internetkiszolgálóit, amihez támogatás és szolgáltatások, valamint ingyenes frissítés jár az Interneten keresztül. A VA Linux az első nagyobb kiszolgálóépítő, amely telepíti és támogatja a Debiant. Az automatikus programfrissítés az apt-get használatával valósul meg a cég által fenntartott webhelyen keresztül, amely gondoskodik a biztonsági frissítésekről, a programok függőségeiről és továbbfejlesztéseiről.
Kapcsolat: VA Linux Systems, Inc.
e-mail: info@valinux.com,
➲ http://www.valinux.com/.

AIT LibraryPro

Az Enhanced Software Technologies-szel (EST) együttműködésben az Overland Data bejelentette, hogy az AIT LibraryPro igazoltan Linux-megfelelő. Az AIT LibraryPro teljes kiépítésben (kilenc modul) 8,55 terabájt tárolókapacitással bír (22,2 TB tömörítéssel), és az átviteli sebessége 388 gigabájt másodpercenként (1,0 TB tömörítéssel) Sony AIT-2 meghajtók használata esetén. Az EST működteti a linuxos szalagos meghajtók megfelelőségét igazoló programot.
Kapcsolat: Overland Data, Inc.
e-mail: srichardson@overlanddata.com
➲ http://www.overlanddata.com/.

X Window System az Intel IA-64-hez

A Metro Link Inc. bejelentette, hogy elkészítette az X Window System IA-64 Itaniummal (Intel) használható változatát. Az üzleti X-kiszolgálót, a Motifot és az OpenGL-t a nagy teljesítményű munkaállomásokhoz (például az animáció- és épülettervezésre, illetve a tudományos számítások elvégzésére használt gépek) készítették. Az X Window System tartalmazza a Metro-X-et – ez egy bővített X-kiszolgáló –, valamint a Metro Motif Complete-et, melybe három Motif-változat is tartozik. Ezenkívül a mozgókép-lejátszásra használható Metro Mediát, valamint az OpenGL monitorkártyára is támaszkodó változatát, a Metro Extreme 3D-t.
Kapcsolat: Metro Link, Inc.
e-mail: sales@metrolink.com,
➲ http://www.metrolink.com/.

Kérjük, küldjön tájékoztatást Linuxszal kapcsolatos termékek megjelenéséről az ujtermekek@linuxvilag.hu vagy a Kiskapu Kft. Linuxvilág szerkesztősége, 1081 Budapest, Népszínház u. 29. címre, vagy faxolja el a 303-1619 számra. A beküldött anyagot szükség esetén átszerkesztjük vagy rövidítjük.

Best Linux 2000

Már kapható a SOT Finnish Software Engineering Ltd. által készített Best Linux 2000 nevű Linux-változat.

Az otthoni felhasználásra szánt csomag négy CD-jén több mint 2000 alkalmazás található, többek között a StarOffice, a GIMP, CD-író program, játékok, multimédiás alkalmazások, adatbázisok és még sorolhatnánk. A telepítést grafikus segédprogram könnyíti meg, és több operációs rendszert is képes indítani egy grafikus indítómenüből.
Kapcsolat: Best Linux/SOT
➲ http://www.bestlinux.net/.

Epitera Desktop és Kandu Genie

Az Epitera nevű, izraeli központú vállalat új munkakörnyezetet dobott piacra, ezt főleg az internetes eszközök és PC-k használóinak ajánlják. Segítségével önműködő letöltéseket állíthatunk be, és a terméktámogatás is elérhető az Interneten keresztül. A csomag háromszintű súgórendszerében egy Kandu nevű „szellem" segíti a felhasználókat.

Kapcsolat: Epitera USA
e-mail: jgundell@epitera.com,
➲ http://www.epitera.com/.

Oracle 9i

Az Oracle 9i relációs adatbázisában és alkalmazás-kiszolgálójában a hagyományosan különválasztott üzleti eljárásokat egymás mellett találhatjuk meg. A csomag többféle helyről (weboldalak, telefonos eszközök stb.) érkező adatok feldolgozására képes egy rendszeren belül. A teljesítmény és a méretezhetőség is sokat javult, hiszen a feldolgozás és az adattovábbítás az adatbázison belül történik. A program gyorsabb, olcsóbb lett és kezelni is könnyebb, mert kevesebb összetevőre van szükség.
Az Application Server már megvásárolható, a Database, Warehouse Builder és a BI Beans csomagok pedig tavaszra várhatók.
Kapcsolat: Oracle Corporation
➲ http://www.oracle.com/.

qX25 for Linux és PCI-os ARTIC kártyák

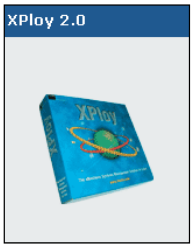
A Quadron Corporation bejelentette a qX25 piacra dobását, mely a kiszolgálókba és munkaállomásokba szánt PCI-os ARTIC kártyákhoz készült X.25-ös fejlesztőeszköz. Segítségével a felhasználók az egyéni igényekhez illeszkedő adatátviteli környezetet építhetnek fel. Minden ARTIC kártyán saját processzor van, és a gépen belüli különálló, programozható adatátviteli rendszernek felel meg. Jelenleg három PCI-os ARTIC kártya kapható: az ARTIC186 8-Port PCI Adapter, az ARTIC 186 Model II ISA/PCI Adapter és az ARTIC186 X.25 ISA/PCI Adapter.
Kapcsolat: Quadron Corporation
e-mail: info@quadron.com,
➲ http://www.quadron.com/.

E-Disk SCW35

A Bitmicro Networks kifejlesztett egy menet közben cserélhető háttértártípust, az E-Disk SCW35-öt. Az új lemeztípus célja, hogy egy megbízható, mégis gyors cserélhető eszközzé váljon. Az E-Disk a kiszolgálóból, a RAID-rendszerből vagy a JBOD környezetből a gép kikapcsolása nélkül is kivehető és visszahelyezhető. Az SCW35 tökéletesen helyettesítheti a szabványos 3,5"-os merevlemezeket, flashlemezeket, és a 80 tűs SCA-2 csatlakozós meghajtókat.
Kapcsolat: Bitmicro Networks, Inc.
e-mail: info@bitmicro.com,
➲ http://www.bitmicro.com/.

XPloy 2.0

A Trustix AS e-üzleti rendszerkezelő eszköze most már távolról és biztonságos módon is képes felügyelni a



kiszolgáló erőforrásait. A legfrissebb változat újdonságai: a futáskörnyezetbe elég csak a használatban lévő modulokat betölteni, valamint a rendszerben helyet kapott egy terheltségkiegyenlítő modul és egy rendszerművelet-figyelő. Az XPloy 2.0 előfizetéses szolgáltatás, havi díja 39 dollár kiszolgálónként (3 éves szerződés esetén).
Kapcsolat: Trustix AS
➲ http://www.trustix.com/.

Communicado Fax

A Communicado egy faxcsomag, ennek segítségével bármilyen rendszer bármelyik alkalmazásából faxolhatunk. A főbb tulajdonságok: együttműködés rengeteg géptípussal és alkalmazással, beépített címtár, faxmegjelenítő (előképekkel és címoldalakkal), továbbítás, jeletéskészítés, számlázás. A program több bejövő és kimenő vonalat is támogat.
Kapcsolat: Merlin Software Technologies International, Inc.
e-mail: info@merlinsoftech.com,
➲ http://www.merlinsoftech.com/.

A hónap szakmai tanácsai

Nem tudok felhasználót létrehozni a COAS-ban

Van egy COAS-szal kapcsolatos gondom. Nem tudom elindítani a felhasználó létrehozására szolgáló segédprogramot. Töröltem az összes zárfájlt, de ezután sem sikerült elindítani. Mi lehet az oka?
Anil Nair, anicmair@usa.net

Ez egy jól ismert nehézség, a gyakori kérdéseknél sokszor tárgyalják. Röviden összefoglalva a lényeget, el kell távolítanod a `/etc/shadow-` és a `/etc/ptmp` fájlokat.
A részletekhez keress rá a COAS kulcsszóra a ➲ http://support.calderasystems.com/ honlapon.
Andy Bradford, andyb@calderasystems.com

Wordperfect telepítése CD-ROM-ról

Van egy Wordperfect CD-m, és szeretném telepíteni. Készítettem egy Wordperfect alkönyvtárat, és kiadtam a következő parancsot a telepítés megkezdéséhez:

```
darkstar:~/wordperfect#      mount/dev/
       cdrom/cdrom
```

A következő hibaüzenetet kaptam:

```
bash: mount/dev/cdrom/cdrom:
       No such file or directory
```

Beléptem a CD-ROM alkönyvtárába, és ott is próbálkoztam, hasonló eredménnyel:

```
darkstar:/cdrom#    mount/dev/cdrom/cdrom
```

Larry Carnahan, lcarn876@aol.com

Biztos vagy benne, hogy a `/dev/cdrom` egy valódi CD-ROM blokkeszközre (pl.: `/dev/hdc`) mutató közvetett hivatkozás? Próbáld ki:

```
ls -l /dev/cdrom
```

Pierre Ficheux, pficeux@wanadoo.fr

Ahelyett, hogy a `mount/dev/cdrom/cdrom` parancsot használnád, próbáld ki a `mount /dev/cdrom /cdrom` parancsot! A `mount` a parancs neve, a `/dev/cdrom` az eszköz, és a `/cdrom` a könyvtár. E három dolog közé szóközt kell írni.
Paulo Wollny, paulo@wollny.com.br

A ServerName beállítása Apache-nál

Telepítettem az Apache 1.3-at egy Debian csomagból. Kiadtam az `apachectl start` parancsot, ami a következő üzenetet küldte:

```
Cannot determine local host name. Use
the ServerName directive to set it
manually.
```

Mi az a ServerName beállítás? Találtam egy ServerName sort a httpd.conf-ban, erről beszél az üzenet? Próbáltam a ServerName-et beállítani, először azt írtam be, hogy `root`, aztán `127.0.0.1`, aztán `localhost`, végül `daniel54` (a gépem neve), de egyik sem működött. Az Apache nem ad használható hibaüzenetet, egyszerűen közli, hogy a httpd nem indult el. Van valami egyszerű módszer az Ethernet kártya lekapcsolására a fizikai eltávolításon kívül?
Daniel Meilleur, daniel_meilleur@hotmail.com

Jó nyomon vagy, a ServerName tényleg a httpd.conf-ban található. Olyan nevet kell itt megadni, ami egyezik egy DNS-bejegyzéssel. A legegyszerűbb módja ennek az, hogy a /etc/resolv.conf fájlban beállítod, hogy a névfeloldás először a `hosts` fájl alapján történjen meg: `order hosts,bind`. Ezután a `/etc/hosts` fájlba írd be ezt a sort: `127.0.0.1 localhost daniel54`.
A hálózati kártya bármikor lekapcsolható az `ifconfig eth0 down` paranccsal. Az eth0 helyett a megfelelő értéket kell írni, ha az eth1-et, vagy magasabb számon nyilvántartott hálózati kártyát szeretnéd lekapcsolni.
Chad R. Robinson, crobinson@rfgonline.com

A windowsos ügyfelek nem találják az SMTP-kiszolgálót

A levelezésre használt kiszolgálón Linux fut. Amikor a felhasználóim levelet szeretnének küldeni vagy fogadni, minden felhasználó az alábbi hibaüzenetet kapja: `The server could not be found (Account "sendmail" SMTP Server; mail, Error number 0x800ccc0d).`
Walter Minja, waminja@yahoo.com

A hibaüzenetedből arra következtetek, hogy a felhasználóid windowsos levelezőprogramokkal dolgoznak. Be kell állítanod helyesen a dolgokat. A kimenő levelekhez minden számítógépen állítsd be az SMTP-kiszolgáló IP-címét a Linux kiszolgáló IP-címére, amelyet a helyi hálózat felé használ.
A bejövő levelekhez a POP3 IP-címét kell beállítani ugyanarra, mint az előbb. Ne feledd, hogy a POP3 szolgáltatás már telepítve is legyen, és engedélyezni kell (a „pop-3” sor ne legyen megjegyzésbe téve a `/etc/inetd.conf`-ban).
Felipe E. Barousse Boué, fbarousse@piensa.com

Kettős rendszer Windows 98-cal

Jelenleg Linux van a 4 gigabájtos merevlemezemen. Hogyan telepíthetek erre Windows 98-at úgy, hogy mindkét rendszer indítható legyen?
Navin Maahdkar, nrm@mindsping.com

A könnyebb út az, hogy elindítod a rendszert a Linux vagy Windows CD-jéről (vagy indítólemezéről), és az fdisk programmal létrehozol egy FAT32-es lemezrészt a Windowsnak és ext2 és swap lemezrészt a Linuxnak. Ezután olyan sorrendben telepíted őket, ahogy akarod. Egy figyelmeztetést azért szívlelj meg: ha a Linux után tervezed a Window 98 telepítését, ne felejts el linuxos indítólemezt készíteni, hogy újratelepíthesd a LILO-t az MBR-be, a Windows telepítője ugyanis törli a LILO-t az MBR-ből.
A „nehezebb” út a Partition Magic használata, ezzel a programmal átszervezheted a lemezrészeket a Linux újratelepítése nélkül.
Bármelyik utat választod, előbb mentsd az adataidat!
Mario Neto, mneto@argo.com.br

Szükséged lesz valamennyi üres helyre a merevlemezen, hogy a Win98-at az első lemezterületre telepíthesd. Ha nincs, akkor egy másik merevlemezre kell telepíteni a

Win98-at. Ennek kell a mesternek lenni (különben újratelepíthetsz mindent!). Ezután be kell állítanod a LILO-t, hogy mindkét rendszert indítani tudd. Valami ilyesmit kell a `/etc/lilo.conf`-ba írni:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
linear
default=linux

image=/boot/vmlinuz-2.2.14-5.0
        label=linux
        initrd=/boot/initrd-2.2.14-5.0.img
        read-only
        root/dev/hdb1

other=/dev/hda1
        label=dos
```

További adatokat a LILO mini-howtoban találsz.
Pierre Ficheux, pficeux@wanadoo.fr

Linux kiszolgáló otthoni hálózathoz

Szeretném RHL 6.1 rendszeremet úgy beállítani, hogy kiszolgáló legyen az otthoni hálózatomban (eth0), és egyben behívó ügyfele legyen az internetszolgáltatómnak dhcp használatával (ppp0). Milyen beállításokat használjak?
Tom Dolan, tdolan@erols.com

Több dologra is szükséged lesz. Terjedelmi okokból nem írunk le itt mindent, csak azokat az adatokat adjuk meg, amelyek világosan elmagyarázzák az összes lépést.
1. Helyi hálózat beállítása:
➲ http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Networking-Overview-HOWTO.html
➲ http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Net-HOWTO.html
2. Csatlakozás az internetszolgáltatóhoz:
➲ http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/ISP-Hookup-HOWTO.html
3. A SAMBA beállítása (ha van Windowst futtató számítógép a helyi hálózatban):
➲ http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/SMB-HOWTO.html
➲ http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Windows-LAN-Server-HOWTO.html
4. Egy kevés biztonsági ismeret sem árthat:
➲ http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Security-HOWTO.html
Felipe E. Barousse Boué, fbarousse@piensa.com

Linux használata kinevezett Citrix ügyfélként

Van-e módszer arra, hogy a Linuxot úgy telepítsük, hogy buta terminálként viselkedjen, azaz a számítógép töltse be a Linuxot, automatikusan lépjen be, és indítson el egy előre beállított alkalmazást?

Vékony ügyfeleket szeretnék létrehozni, a Linux lenne az operációs rendszer és a Citrix ICA az egyetlen alkalmazás. Legjobb lenne, ha a számítógép betöltené a Linuxot, bejelentkezne egy tetszőleges felhasználói néven, majd elindítaná a Citrix ICA ügyfélprogramot.
Stefan Ostadal, sostadal@gate.co.uk

Az elképzelésed megvalósítható, ehhez csak az INIT működését kell megértened. Amikor a rendszer elindul, beolvassa a `/etc/inittab` tartalmát, amely néhány parancsfájl futtatását írja elő. Azt is megmondja a rendszernek, hogy indítson el egy pár getty folyamatot, amelyek a beléptetést végzik. Csak annyit kell tenned, hogy hozzáadsz egy bejegyzést, ami elindítja az alkalmazásodat a getty helyett. Megjegyzendő, hogy az INIT eleve a rendszergazda nevében fut, ebben az esetben nem kell még egyszer bejelentkezned. Ne feledd azonban, hogy ez a módszer nem biztonságos! Csak olyan esetekben javasolható, mint a tied, azaz egy vékony ügyfél létrehozásához, ahol a helyi biztonság érdektelen.
Chad R. Robinson, crobinson@rfgonline.com

További LILO-gondok megoldása

Nemrég telepítettem a RedHat 6.2-t, és remekül működött. Rendszerindításkor megjelent a LILO, és betöltődött a RedHat Linux. Megváltoztattam a LILO-t, hogy DOS-t indítson. Most, ha elindítom a rendszert, egy C:> parancssort kapok. Hogy kaphatom vissza a RedHatet?
Pablo, sonicmaster@teleweb.pt

Egy linuxos indítólemezt kell használnod ahhoz, hogy elindítsa a Linuxot. Azután állítsd be a lilo.conf fájlt úgy, hogy a LILO képes legyen mindkét rendszert elindítani, és érvényesítsd a változásokat a `lilo` parancs kiadásával (rendszergazdaként). Például:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
linear
default=linux

image=/boot/vmlinuz-2.2.14-5.0
        label=linux
        initrd=/boot/initrd-2.2.14-5.0.img
        read-only
        root/dev/hdb1

other=/dev/hda1
        label=dos
```

Pierre Ficheux, pficeux@wanadoo.fr

Magyar HOGYAN-ok lelőhelyei:
➲ http://linuxvilag.hu
➲ http://magyar.linux.hu/frameset.php3?melyik=hogyan
➲ http://suselinux.ini.hu

Beágyazott Linux

Az emberek gyakran azt kérdezik: „Ez az egész beágyazott Linux történet jól hangzik, de vajon használják-e a gyakorlatban a vállalatok, jelenik-e meg valaha is kézzelfogható termék? És mikor kerülnek ezek a piacra?"
A válasz: „Fogadjunk, hogy már most is kapható beágyazott Linuxot futtató készülék, nem is egy!". Miközben ezeknek a termékeknek már zajlik az árusítása, és néhány nagy tételben is kapható, a legtöbb termék még különböző fejlesztési állapotban van. Ne felejtsük el, hogy a legtöbb új termék fejlesztése általában kilenc–tizenkét hónapig tart, miközben a beágyazott Linux-rendszereket már az elmúlt évben is nagy érdeklődés kísérte. Ha néhány gyors számítást végzünk, könnyen megjósolható a Linux-alapú beágyazott rendszerek sikere.
A Linuxvilág és más kiadványok már számos beágyazott Linux-alapú eszközt bemutattak, kipróbáltak, de érthető okokból a legtöbb projektet még titok övezi. A beágyazott Linuxot használó eszközök gyártói azt mondogatják: „Nagyon szeretnénk bemutatni számos izgalmas terméket, de sajnos titoktartási kötelezettségünk van, ezért nem beszélhetünk róluk."
A következőkben röviden bemutatunk néhány Linux-alapú eszközt, melyek mindeddig a nagyközönség elől rejtve maradtak. Ne felejtsük el, ez még csak a kezdet.

Marokgépek

Samsung Yopy PDA
A PDA-khoz illő méretű doboz böngészőt, MP3-lejátszót és CompactFlasht tartalmaz. Az eszközben 3,9"-os színes háttérvilágításos LCD kijelző, ARM processzor, RS232 és USB csatlakozó van. Ezenkívül minden hagyományos PDA-alkalmazás megtalálható benne.
➲http://www.linuxdevices.com/news/NS6600527506.html

Agenda VR3 PDA
A VR3 egy teljes értékű PDA, 160x240 képpontos (2,25x325" látható képméret) háttérvilágításos LCD-vel. Egy 66 MHz-es, 32 bites NEC VR4181 processzor hajtja, van még benne 8 MB rendszermemória, valamint legfeljebb 8 MB flashmemóriát tehetünk bele. A beépített flashmemória az elemlemerülés okozta adatvesztés elkerülésére szolgál. Soros RS232, IrDA és egy egyedi nagy sebességű csatolófelület található rajta.
Az operációs rendszer Linux-VR.
➲http://www.linuxdevices.com/articles/AT4992223978.html

Compaq iPAQ PDA
Nem kimondottan linuxos eszköz, de számos tervezet van az iPaq linuxosítására, egy csoport például a Compaq által támogatott ➲handhelds.org oldalon is megtalálható.
Az iPAQ 240x320 képpontos háttérvilágításos színes kijelzővel, 206 MHz Intel StrongArm processzorral, 32 MB RAM-mal és 16 MB flashmemóriával rendelkezik. A külső bővítési lehetőségei: IrDA, soros (sync/async), USB, PCMCIA.
➲http://www.handhelds.org/Compaq/iPAQH3600_H3600.html

Telefonok különféle méretben és formában

Aplio/PRO Internet Phone
Ennek a csöppnyi kihangosítható internettelefonnak a lelke egy Aplio/TRIO processzor, ez pedig beágyazott Linuxot futtat. A belső memória 4 megabájt RAM-ból, plusz 2 megabájt flashmemóriából áll. Az Internetre a beépített modem, vagy a hálózati kártya (modelltől függően) segítségével kapcsolódhatunk. Operációs rendszere a uClinux.
➲http://www.linuxdevices.com/articles/AT9173372049.html és a „VoIP és a beágyazott Linux" című írásban (Linuxvilág 2000. november, 78. oldal).

Ericsson Cordless Screen Phone
Az eszköz tulajdonképpen egy vezeték nélküli webpad, beépített telefonnal, Bluetooth módszerrel, otthoni használatra. Lehet vele internetezni, levelet olvasni, hangüzeneteket küldeni és telefonálni. A beágyazott számítógép egy National Geode processzor, ami RedHat Linuxot futtat. A GUI (a grafikus felület) a Trolltech Qt/Embedded GUI eszközkészlet segítségével készült, a böngészője pedig az Opera.
➲http://www.linuxdevices.com/articles/AT4268573160.html

SK Telecom IMT2000 WebPhone
Ez az eszköz egy mobiltelefon és egy PDA ötvözete, egy 4"-os LCD kijelzővel és beépített kamerával felszerelve. Úgy néz ki, mint egy PDA, de van benne egy telefonegység is, valamint StrongARM SA1110 206 MHz processzor, 32 MB RAM és legfeljebb 32 MB flashmemória, az operációs rendszer PalmPalm Tynux Qt/Embedded for GUI támogatással, és egy Opera böngésző.
➲http://www.linuxdevices.com/articles/AT3334419107.html

TV Set-Top eszközök

Nokia Media Terminal Set-Top Box
Ez a készülék rengeteg szolgáltatással egészíti ki a hagyományos tévékészülékeket. Lehetővé válik a digitális kép és hang, a digitális tévé, a felhasználó által kiválasztott program (Personal TV), a tárolt televíziós programok, az Internet, az e-mail és csevegés, és játék,

valamint számos webalapú alkalmazás használata.
A Media Terminal programja Linux-alapú, a böngésző a szabad forráskódú Mozilla, az X-Window System és a különlegesen felhasználóbarát „Nokia Navi Bar" felhasználói felület. A beépített számítógép 336 MHz Intel Celeron processzor, Intel 810 lapkakészlettel, 32 MB SDRAM-mal, és legalább 20 GB merevlemezzel.
➲http://www.linuxdevices.com/articles/AT4370516520

Indrema Set-Top Gaming System
Az Indrema Entertainment System (IES) egy olyan set-top egység, amely egy felsőkategóriás játékgéppé varázsolja a tévékészüléket. Az eszköz egy lapos, de kiemelkedő minőséget sugalló dobozba került. A Világhálóra telefonnal vagy más széles sávú eszközzel is csatlakozhatunk (a beépített 10/100 hálózati csatlakozó segítségével), és jár hozzá játékvezérlő is. A beágyazott számítógép egy „X86" processzor, 64 megabájttal és legfeljebb 50 GB merevlemezzel. A nagy sebességű grafikáról egy bővítménnyel fejleszthető nVidia GPU gondoskodik. Az operációs rendszer a nyílt forráskódú, játékosoknak készült DV Linux.
➲http://www.linuxdevices.com/articles/AT2772260294.html

TiVO
A TiVO „személyes videofelvevő", az egyik első és legismertebb beágyazott Linux-rendszer. Az eszközben 54 MHz-es PowerPC 403GCX processzor, 16 MB RAM és akár harmincórányi tévéműsor tárolására alkalmas merevlemez található. Meglepő módon a videó nem számítógépszerűen viselkedik, pedig a működése egy grafikus processzoron alapul. Ráadásul az eszköz tartalmaz egy teljes értékű beágyazott Linux-felületet, bár megjegyzem, nem minden felhasználónak van szüksége grafikus felhasználói felületre.
➲http://www.linuxdevices.com/news/NS8858229837.html

Webpadok

FrontPath ProGear Wireless Webpad
Elsősorban lapgépek piacát célozták meg e vezetéknélküli, Linux-alapú hordozható eszközzel. A készülék számos médiaformátumot támogat, és 10,4"-os TFT kijelzőjét többek között beviteli eszközként is használhatjuk: például virtuális billentyűzetként vagy kézírás felismerésre.
➲http://www.linuxdevices.com/articles/AT5771747599.html

Screen Media FreePad
Könnyen használható, teljes értékű webpad, amely az összes kapcsolattartási és számítógépes felhasználási lehetőséget támogatja, beleértve a webböngészést, az e-mailt, a telefont és az üzenetrögzítőt, valamint az intelligens kártyaolvasót (SmartCard). Az eszköz nagy (10,4") LCD kijelzőt, érintőpanelt, beépített DECT vezetéknélküli módszert, és USB-csatlakozót is tartalmaz. Processzora 166 MHz-es MediaGX, 32 MB RAM-mal, és 16 MB belső flashmemóriával. A beágyazott operációs rendszer Linux, az ablakkezelő rendszer a Nano-X Microwindows projektjétől, a böngésző pedig az Operától származik.
➲http://www.linuxdevices.com/articles/AT265123453.html

Szórakoztató hangdobozok

Kerbango Internet Radio
Az Internetre telefonvonalon, helyi hálózaton vagy USB hálózati csatlakozón keresztül léphetünk be. A Kerbango Internet Radio számítógépét egy 80 MHz-es Motorola PowePC processzor hajtja 8 MB RAM-mal és 8 MB flashmemóriával felvértezve.
A beágyazott Linux operációs rendszer a MonaVista Hard Hat Linux, a különleges Kerbango Audio Operating Systemmel (KAOS) kiegészítve, amely lehetővé tette a könnyen kezelhető felhasználói felület létrehozását. Az internetes rádióadások „vétele" mellett, a beépített FM rádióvevővel és antennával a hagyományos rádióadók is foghatók.
➲http://music.gamespot.com/features/kerbango

Diamond Rio Audio Receiver
Ez a házi vevőkészülék, amely elsőként nyerte el a ZDNet „Szakmai divatdiktátor" díját, lehetővé teszi a felhasználó számára, hogy a számítógépéről MP3 zenét sugározzon a ház bármelyik helységébe, akár a központi hifiberendezésre, ezt eddig csak hordozható digitális lejátszókkal lehetett megoldani.
A beépített számítógépben a Cirrus Maverick processzor beágyazott Linuxot futtat.
➲http://www.linuxdevices.com/news/NS7845657816

PhatNoise PhatBox MP3 System
A PhatBox autóhangrendszer, amely elnyerte a „Legjobb termék" címet a Third Annual MP3 Summit fórumon.
E termék lehetővé teszi az MP3 felvételek hallgatását az autóban is. A beépített számítógép egy 74 MHz-es Cirrus Logic EP7212 processzor, ami beágyazott Linuxot futtat.
➲http://www.linuxdevices.com/news/NS7845657816.html

Rick Lehrbaum

A Morlock piac

A grafikus felület jó, mégis a parancssor az igazi.

A Unix nem egy termék, hanem egy monda, a megszállott programozók kultúrájának különös gondossággal összeállított mondája. A mi Gilgames eposzunk. Sok programozó ismeri, szereti és érti, s ők azt is tudják, ha szükséges, egészen az alapoktól újra felépíthetik. Ezt nagyon nehezen értik meg azok, akik képzeletében az operációs rendszer egy cég megvehető termékeként él.

Szívesen olvasom *Neal Stephenson* esszéit. Ismertebb művei a *Snow Crash* és a *Cryptonomicon. Az In theBeginning Was the Command Line* teljes terjedelémben megtalálható a Neten. A mű a parancssor használatának valós világára derít fényt: a gyakorlatias eredetiségére, a felmutatott eredményeken alapuló hagyományaira, a „legjobb eszköz az adott feladatra” hozzáállásra és arra, hogy a grafikus felületet használó többség mennyire nem érti meg a számítógép használatának ezt a jobb, elemibb módját (használatról van szó és nem csak közvetlen kapcsolatról).

Ez az írás a jövőről is szól. Stephenson történetének vége visszakanyarodik a kezdetekhez, a parancssorhoz. A parancssor használata nem egyszerű, írja, de autót javítani, vagy házat építeni sem az. „Az élet nagyon nehéz és bonyolult dolog” – foglalja össze –, „nincs olyan felület, ami ezt elrejthetné, aki nem hiszi, pórul jár. Ha nem szereted, hogy mások döntenek helyetted, el kellene kezdened saját döntéseidet meghozni.”

Az elmúlt tizenöt évben a számítógép-felhasználók többsége úgy határozott, hogy a Microsoftra bízza a döntést. Személyes véleményem az, hogy a Microsoft túlontúl kevés elismerést kap a kedvező döntéseit illetően. A tény, hogy a rendkívül bonyolult eljárásokat és feladatokat kevésbé összetett, de rendkívül mutatós termékekké alakították, önmagában egy hatalmas piaci siker. Manapság elképzelhetetlen Microsoftprogramok nélkül az üzleti élet, olyan lenne mint az áruszállítás a belső égésű motorok nélkül. A programok gyakori hibáival szemben mindenütt méla belenyugvás figyelhető meg. „Képzeld el, mi lenne, ha minden csütörtökön, amikor teljesen szokványos módon bekötöd a cipőd, az felrobbanna.” – mondja *Jeff Rankin* – „A számítógépekkel kapcsolatban ez mindennapos, mégsem jut eszébe senkinek panaszkodni.”

Stephenson két okot is megjelöl. Az emberek szeretik a közvetett tapasztalatokat. „Nézd csak meg a Disney sikerét: jobban kihasználja a közvetett tapasztalatokat, mint bárki más” – írja Stephenson. „Ha a Disney tudná, hogy mi az az operációs rendszer, és hogy az emberek miért használják, egy-két éven belül eltaposhatná a Microsoftot.” A másik ok, hogy „manapság túl elfoglaltak vagyunk ahhoz, hogy minden részletet megértsünk. Jobb valamit körülbelül megérteni, mint egyáltalán nem.”

A hangsúly a „minden”-en van. A személyi számítástechnika olyan becsvággyal született, ami bőven túlmutatott a lehetőségein. Mivel *lehetősége volt* arra, hogy szinte bármit megtegyen, meg is *kellett* tennie mindent. És meglepő módon, több mint elégséges volt a kereslet a „szinte bármi”-re ahhoz, hogy vonzza a kockázati befektetőket, akik a programkészítő cégek sokaságát pénzelik. De hosszú távon (ami azért nem is volt olyan hosszú táv), úgy tűnik, csak egy cég értette meg, hogy pontosan mennyit tud a „*szinte bármi*”-ből egy PC megvalósítani, és hogy hogyan lehet az ebből fakadó bonyodalmakat a lehető legtöbb ember számára mérsékelni. Bármennyire is szörnyű a Microsoft más tekintetben, ez a felismerés a damaszkuszi fordulathoz hasonlítható.

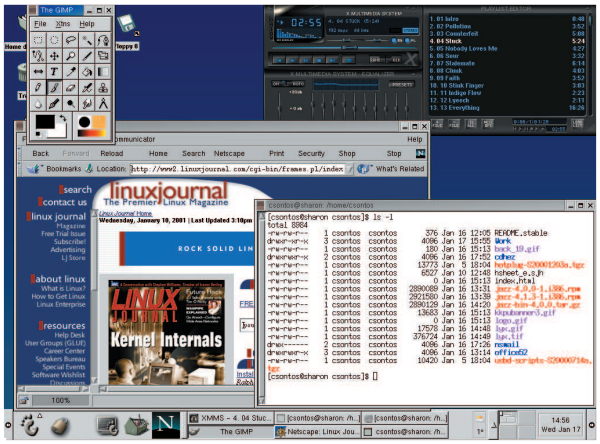
Az eredmény: „kétrétegű rendszer, mint a Morlockok és az Eloik világa *H. G. Wells Időgép* című regényében, azzal a különbséggel, hogy itt minden fordítva van”.

Íme a magyarázat: A regényben az Eloik képviselik az elsatnyult felső réteget, a föld alatt élő Morlockok pedig– akik kiszolgálják őket – forgásban tartják a tudomány fogaskerekeit. A mi világunkban ez fordítva van. A Morlockok vannak kevesebben, és ők parancsolnak, mert értik, hogy mi hogyan működik. Az Eloik, akik sokkal többen vannak, mindent az őket átitató elektronikus tömegtájékoztatási eszközökből tanulnak, amit a könyveket olvasó Morlockok irányítanak és ellenőriznek. Ennyi tudatlan ember veszélyes lehet, ha rossz útra tér; ezért kialakítottunk egy olyan népszerű kultúrát, ami hihetetlenül fertőző, és kiheréli azokat, akiket megfertőz: nem akarnak döntéseket hozni, és képtelenek állást foglalni.

A Morlock, aki bonyolult témakörök szakértője, s aki elég művelt és tetterős ahhoz, hogy felfogja a részleteket, Disney-típusú érzékelésre alapozott felületeket gyárt, azért, hogy az Eloi megkapja a lényeget anélkül, hogy megerőltetné az agyát, vagy unatkoznia kellene.

Hogy igazságos legyen, Stephenson elismeri a közvetett tapasztalatok kedvező társadalmi hatásait. Az a politika, amit az olyan szavazók kénye és kedve irányít, akik azt hiszik, hogy a pankrációban az ütések valóságosak, jogosan kétségbeejtő azok számára, akik nem így látják. De az olyan világok sem túl népszerű helyek, amelyeket parancssorból irányítanak, azaz keményvonalas értelmiségiek – legyenek világiak vagy egyháziak – kormányoznak. A kulturális különbségek érdekesek, de az

igazat megvallva nem fontosak. Az igazi lényeges különbség a gyártók és a felhasználók között van. Osszuk a számítástechnika világát három osztályra és nézzük, hogyan vonhatunk párhuzamot a Morlockok és az Eloik világával: a PC-s munkafelület az Eloik birodalma. Az Apple megálmodta, a Microsoft birtokolja, és a legtöbbünk azt népesíti be. A kiszolgáló egy kicsit keverék. Ez már a Morlockok világa, de sok benne az Eloi-vonás. Ezért nőnek a Linux és a Windows NT/2000 számadatai, abszolút értékben és piaci részesedésben egyaránt (semmiképpen nem akarom megsérteni azt a számtalan Morlockot, aki becsülettel dolgozik windowsos rendszerrel, programozza azt). A beágyazott világ egyedül a Morlockoké. Ez mindig így volt és mindig is így lesz.



A beágyazott világban, ahol minden a sajátos szakterületekre tagolódásról szól, nevetséges elképzelés az olyan eszköz, ami mindet tud. A beágyazott világnak nem kellenek csillogó-villogó szóképek, mert senki sem akarja, hogy egy gomb vagy egy számlap „bármit” megtegyen, az alkalmazás függvényében. A beágyazott világban nincs „bármi”. Ha egy rádióállomást keresel, vagy egy szelepet szabályozol, céleszközt használsz, ami másra nemigen jó.

Kiderül, hogy a Linux, mivel a mérete kicsi, a felépítése moduláris, a rendszer jól ismert és nyílt forráskódú, eszményi végtelen sajátos, egyedi feladat megoldására. Mivel ez rengeteg dolgot foglal magába, várható, hogy a Morlockok népességének száma, felforgató tevékenysége és hatalma gyorsan fog nőni. Az eredmény egy sokkal mélyebbre ható és fontosabb forradalom lesz, mint a személyi számítástechnika.

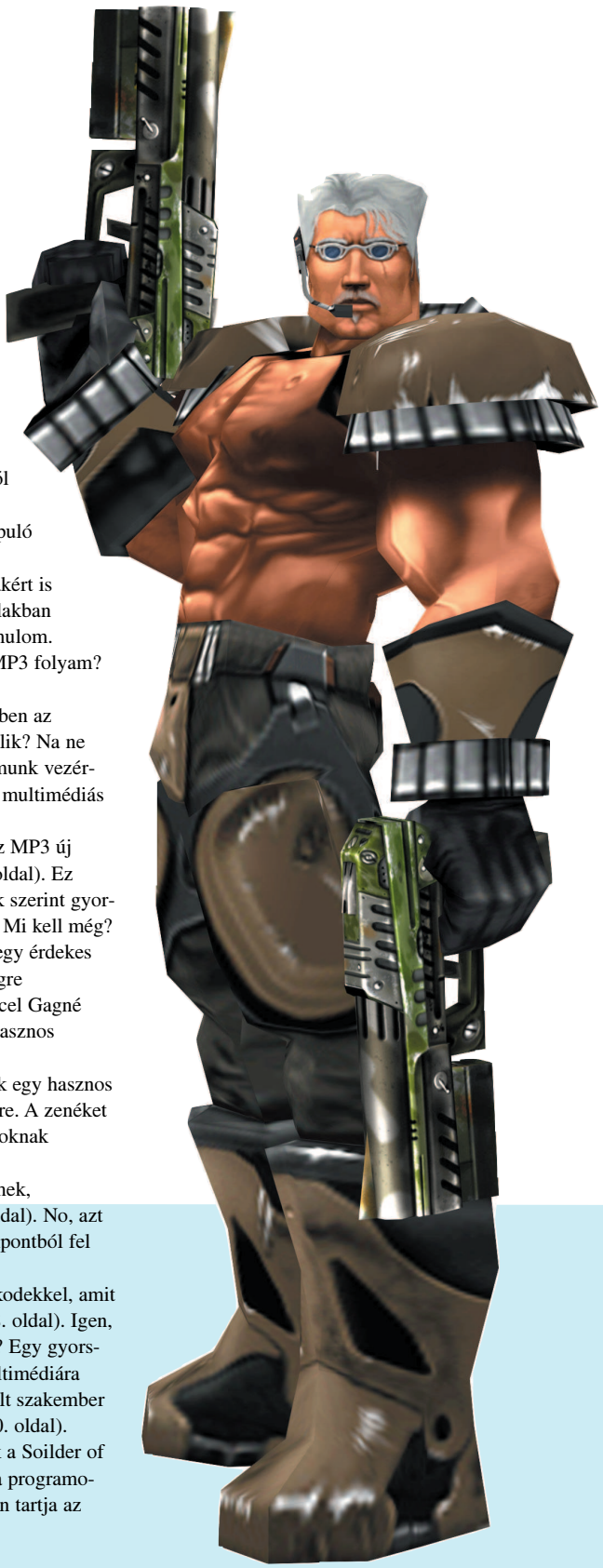
Az üzletemberek számára a legfontosabb kérdés ez lehet: *mennyi idő van még hátra addig, amíg a mindennapi egyedi feladatok ellátására megjelenő Linux elavulttá teszi a mai értelemben vett személyi számítástechnikát?* Másképpen megfogalmazva, mikor lesz majd könnyebb és gyorsabb összebarkácsolni (vagy venni, esetleg venni és összebarkácsolni) egy Linuxon futó összevont számlázási és készletnyilvántartó rendszert, mint megbirkózni egy harmadik fél által gyártott programcsomaggal, aminek a nyűgös, öreg Windows 98-on kell futnia? Esetleg egy könyvviteli rendszert, ami könyvel és TCP/IP-n kapcsolódik a világ többi részéhez és általános, megbízható gépen fut?

Nézzük másfelől a kérdést. *Mennyi időnek kell még eltelnie addig, amíg nem rúgnak ki valakit azért, mert Linuxot használ, amikor a vezető üzletemberek többsége Morlock?*

Lássunk egy jó kiindulópontot: az IBM-nél már ez a helyzet.

Doc Searls
(doc@ssc.com)
a Linux Journal
szerkesztője.

Multimédia



A Raging Search (➲http://ragingsearch.altavista.com/) szerint a „my cat” szópáros jelenleg 50 665-szor fordul elő az Interneten, a streaming media” viszont 57 615 honlapon szerepel. Ez a web már nem olyan, mint amilyennek megismertük. Az emberek az „ez itt az érettségi tablóképem” típusú szegényes látvány helyett egyre inkább tetszetős animációkat, multimédiás fájlokat helyeznek el oldalaikon, és ebből bizonyára mindenki azt a következtetést vonja le, hogy jól megjegyeztük a GIF fájlokkal kapcsolatos leckét: az emberek közötti kapcsolatot ne bízd többet egy jogdíjas szerződésen alapuló formátumra.

De az emberek nem tanultak ebből, pedig nemsokára az MP3-akért is pénzt kell fizetnie mindenkinek. Ahogy ezt írom, egy másik ablakban a ➲http://www.mp3licensing.com/royalty/broadcast oldalt bámulom. Internetes rádióállomást szeretnél indítani, melynek magja az MP3 folyam? Akkor 2001. január 1-től vedd elő szépen a pénztárcádat.

Akkor mi is lesz most? Valami európai nagyvállalat szép csendben az „Internetes Rádiózás Információellenőrző Minisztériumává” válik? Na ne már! Mi itt a Linuxvilágnál imádjuk a szabadságot. E havi számunk vezérfonalában kizárólag ingyenes, nyílt forrású, nagy teljesítményű multimédiás alkalmazások területén nézünk körbe.

De mit is kínálunk a körképben? Először is megismerkedünk az MP3 új vetélytársával, a teljesen ingyenes Ogg Vorbis csomaggal (34. oldal). Ez a könnyedén bővíthető és alkalmazható módszer a szakemberek szerint gyorsan leváltja majd az MP3-mat. Okosabb, hatékonyabb, szabad... Mi kell még? Aki maga szeretne zenét írni, az előző számunkban találhatott egy érdekes cikket a MOD formátumról és a modulszerkesztőkről. Most végre hozzájuthat mindenki a cikk második részéhez (54. oldal). Marcel Gagné most is körbevisz minket egy érdekes sétára, bemutat egy-két hasznos zenével kapcsolatos linuxos programot (73. oldal).

A hangok világához kapcsolódik még egy cikkünk. Bemutatunk egy hasznos kiszolgálót, amivel központi vállalati zenedobozt hozhatunk létre. A zenéket rápakoljuk, majd szavazunk a számokra. A program a kívánságoknak megfelelően állítja össze a műsort (MSERV, 42. oldal).

A másik fontos téma a mozgókép. Megismerkedhetünk egy remek, filmvágásra alkalmas programmal, a Broadcast 2000-rel (30. oldal). No, azt nem mondom, hogy nem kell hozzá erős gép, viszont sok szempontból fel tudja venni a versenyt a fizetős programokkal!

Gondoltunk a programozókra is! Megismerkedünk egy remek kodekkel, amit végre fel tudunk használni saját programjainkban (FIASCO, 38. oldal). Igen, igen, de milyen nyelven is érdemes multimédia-programot írni? Egy gyorstalpaló erejéig belekóstolunk az SDL környezetbe, ami egy multimédiára kiélezett köztes réteget képez (25. oldal), valamint egy tapasztalt szakember is leírja tapasztalatait, ötleteit, tanácsait (A Linux és a videó, 40. oldal).

És hogy lazítsunk is egy kicsit, ontunk egy kis vért, bemutatjuk a Soilder of Fortune linuxos változatát. Végülis, ha valaki éppen nem akarja programozással tölteni az éjszakáját, akkor csak kell valami, ami magasan tartja az adrenalinszintet, nem igaz?

Hasznos olvasást és kellemes kikapcsolódást kívánok!

SDL gyorstalpaló

A babók szeretik a játékokat. Mi is...

A linuxos játékok egyre terjednek. Részben a mostanában beindult linuxos multimédia-fejlesztéseknek köszönhetően, részben pedig egyszerűen azért, mert a babók szeretik a játékokat. Az utóbbi néhány évben számos kitűnő linuxos multimédiás eszköz látott napvilágot. Ilyen például a GGI grafikus csatoló vagy az ALSA hangrendszer. Az utóbbi időben az SDL programkönyvtár is fejlődésnek indult.

Az SDL egy általános célú multimédia-programozási könyvtár, ami gyors és hordozható elérést szolgáltat a grafikához, a hanghoz, a bemeneti eszközökhöz, a szálkezeléshez és az OpenGL megjelenítéshez. Az SDL könyvtár magja, több Unix-változat mellett átvihető BeOS, MacOS és Win32 rendszerekre is. Ezáltal kitűnő választás lehet azok számára, akik a hatékonyság feláldozása nélkül szeretnének felületfüggetlen játékokat készíteni.

Más multimédia eszköztárakkal szemben, az SDL nem közvetlenül a géppel tart kapcsolatot, inkább egy réteget képez az alkalmazás és az alatta lévő rendszer között. Például az SDL grafikus rendszere képkockatárat (frame buffer) vagy az X11-et használ Linux alatt, viszont DirectDraw-t Windows alatt. Az SDL API-felülete egyik esetben sem változik, az alkalmazásnak nem kell törődnie azzal, ami a háttérben zajlik. Egy gondosan felépített SDL-alkalmazást egyetlen gyors újrafordítással át lehet vinni más felületre.

Ebben a cikkben az SDL video-API világában teszünk egy körutat, egészen az alapoktól kezdve. Azt is bemutatjuk, miképpen kell adatokat fogadni a billentyűzetről. E cikk legnagyobb része kivonat a *John Hill* Programming Linux Games című, készülő művéből (No Starch Press and Loki Entertainment Software, 2001 elején várható).

Az SDL beszerzése

Az SDL ingyenes csomag (az LGPL alá tartozik), és letölthető a csoport honlapjáról ➲http://www.linsdl.org. A jelenlegi SDL könyvtáron kívül, ez a honlap számtalan példaforráskódot, bemutatópéldányt, játékot és kiegészítést is tartalmaz. Az SDL-t könnyű telepíteni, de ennek megkönnyítésére a honlap futtatható állományokat is kínál a legismertebb felületekhez.

Az SDL tervezési alapgondolata

Ha valaki dolgozott már a Microsoft DirectX eszköztárával, észrevehette, hogy ahhoz képest az SDL egy aprócska könyvtár.

A rendszermaghoz tartozó könyvtár (kernel library; magkönyvtár) forráskódja hat megabájt alatt van, ebbe pedig már sok olyan kódot is belevettünk, amelyek Linux alatt soha nem kerülnek fordításra. De ez ne tévesszen meg senkit. Ez a hat megabájt jól kihasznált, és a SDL magkönyvtár szinte mindent elérhetővé tesz számunkra, ami csak egy kiemelkedő minőségű linuxos játékhoz vagy médialejátszóhoz kellhet. Továbbá, a honlap otthont ad számos kiegészítő könyvtárnak is, amelyek olyan további szolgáltatásokat nyújtanak, mint például a képek betöltése és a fejlett hangkeverés. Ezeket a lehetőségeket a magkönyvtártól elkülönítve tartják, ennek köszönhetően az SDL kicsi és könnyen megismerhető marad. Az SDL könyvtár több rész-API-t tartalmaz, ami által felületfüggetlen támogatást nyújt a videó-, hang-, bemenet-, szálkezelés, OpenGL-leképezés és számos olyan további képesség eléréséhez, amit a játékírók igazán értékelni tudnak. Sajnos, nincs elég helyünk, hogy mindent bemutassunk, ezért most csak a videoprogramozásra és a bemenetkezelésre szorítkozunk. Ez az a két dolog, amire leginkább szükség lehet, hogy az első lépéseket megtegyük az SDL-ben.

Az SDL video-API

Az SDL video-API egyetlen célja, hogy megfelelő videoeszközt találjon, és beállítsa a program számára. Miután előkészítette a megjelenítőt (készített egy ablakot vagy átkapcsolta a videokártyát a megadott módba), az SDL félreáll az útból, mindössze néhány alapvető függvényt nyújt a képponttömbök mozgatásához. Az SDL nem rajzoló eszközkészlet: az már nem az SDL dolga, hogy az előkészítés után mit teszünk a megjelenítőeszközzel.

Az SDL úgynevezett felületeket (SDL_Surface típusú szerkezeteket) használ a grafikus adatok megjelenítéséhez. A felület egy egyszerű memóriatömb, ami téglalap alakú, képpontokból álló terület tárolására használható. Minden felületnek van szélessége, magassága és egy adott képpontformátuma (erről később bővebben is szó lesz). Az SDL a képállományokat közvetlenül felületekbe tölti, és a képernyő is felületnek tekinthető (jóllehet meglehetősen kivételes felületnek).

A felületek legfontosabb tulajdonsága, hogy nagyon gyorsan lehet őket egymásra másolni. Az egyik felület képpontjait át lehet helyezni egy másik felület azonos méretű téglalap alakú területére. Ezt a műveletet blitnek (block image transfer; képrészletátvitel), vagy részletátvitelnek nevezik. A részletátvitel igen fontos eszköze a játékprogramozásnak, mivel lehetővé teszi, hogy teljes képeket előre rajzoljanak meg (ezt gyakorta egy művész készíti el valamilyen rajzolóprogram segítségével). Mivel a képernyő ugyanolyan felület, mint bármely másik, egész képeket lehet a képernyőre küldeni egyetlen részletátvitel segítségével.

Az SDL általános függvényeket szolgáltat a felületek közti gyors részletátvitelhez, sőt, a különböző formátumot használó felületek közötti átalakítást is futás közben végzi el.

A megjelenítés beállítása

Mielőtt tömböket dobálhatnánk a képernyőre, előbb be kell állítanunk az SDL könyvtár alapértékeit, és át kell váltanunk a megfelelő módba. Vessünk egy pillantást az első listára, ami a „Szia Világ!” SDL-es megfelelője.

Ez a program behívja az `SDL.h` fejlécfájlt, ami az SDL mesterfejléce. Minden SDL-alkalmazásnak hivatkoznia kell erre a

fejlécre. A program ezek után beolvas még két szabványos fejlécfájlt a printf és az atexit függvények eléréshez. Az `SDL_Init` meghívásával kezdünk, amivel az SDL-t alaphelyzetbe állítjuk *(lásd 1. listát)*. Ez a szolgáltatás értékként egy `VAGY` művelettel összeállított értéklistát vár, ami megmutatja számára, hogy mely alrendszereket kell üzembe állítania. Mivel bennünket jelenleg csak a videoalrendszer érdekel, az `SDL_INIT_VIDEO` értéket adjuk át (ha például hangot is szeretnénk, a szolgáltatást ezzel az értékkel kellene meghívni: `SDL_INIT_VIDEO | SDL_INIT_AUDIO`). Hacsak valamilyen végzetes hiba nem történik, ez a függvény nullával tér vissza. A C atexit rendszerét használjuk arra, hogy kilépéskor meghívjuk az `SDL_Quit` függvényt. Ez adja meg az esélyt arra, hogy az SDL megfelelően leálljon (ami különösen akkor fontos, ha egy teljes képernyőn futó alkalmazás omlana össze).

Ezután a `SDL_SetVideoMode` függvényt használjuk arra, hogy a megjelenítő tudomására hozzuk az alkalmazandó felbontást (ez jelen esetben 640×480 képpont) és színmélységet (16 bites csomagolt képpontok). Azonban van itt egy kis csalafintaság: az SDL megpróbálja ugyan beállítani a kért videomódot, de előfordul, hogy nem jár sikerrel. Ilyenkor az SDL nem szól semmit, hanem saját maga emulálja a kért módot. Ez többnyire elfogadható, hiszen az emulációs kód viszonylag gyors, és általában nekünk sem feladatunk a különféle videomódokkal foglalkozni. Az `SDL_SetVideoMode` egy felületmutatót ad vissza, ami a képernyőnek felel meg. Ha valami hiba történt, a szolgáltatás `NULL`-t ad vissza.

Utolsó lépésként hírt adunk a sikerről, és kilépünk. A C könyvtár automatikusan meghívja a `SDL_Quit` függvényt (mivel már bejegyeztük az atexit segítségével), és az SDL az eredeti helyzetbe állítja vissza a videomegjelenítőt. (Közvetlenül is meghívhatjuk az `SDL_Quit` függvényt, amennyiben az alkalmazásból történő kilépés előtt szeretnénk lezárni a rendszert. Nem okoz gondot, ha egynél többször hívjuk meg.) Elkészítettünk egy SDL-alkalmazást, most már csak le kell fordítanunk. Helyes telepítést feltételezve, az SDL-alkalmazásokat könnyű felépíteni, mindössze néhány jelzőre (flag) és könyvtárra van szükségünk. Az alap SDL-változat tartalmaz egy sdl-config nevű programot (hasonlóan a gtk-config vagy a glib-config programokhoz, amelyek a GTK+ eszközkészlettel érkeznek), ezzel tudjuk előállítani a gcc-hez szükséges megfelelő parancssori kapcsolókat. Az `sdl-config -cflags` egy kapcsolólistát készít, amelyet aztán átadhatunk a fordítónak. Az `sdl-config --libs` pedig a használandó könyvtárak listáját állítja elő. A parancssorba helyezéshez akár fordított aposztrófos behelyettesítést (`` `parancs` ``) is alkalmazhatunk. Amennyiben az SDL már telepítve van rendszerünkön, a példát a következő parancssorral fordíthatjuk le:

```
$ gcc sdltest.c -o sdltest `sdl-config --cflags
--libs`
```

Képpontok közvetlen kirajzolása

Egy SDL-felületre adatokat tenni igen könnyű feladat. Minden `SDL_Surface` szerkezet tartalmaz egy `pixel` tagot. Ez egy void mutató a nyers, grafikus képre, ahová akár közvetlenül is írhatunk, amennyiben tudjuk, milyen típusú képpontok alkotják a felületet. Mielőtt hozzányúlnánk az adatokhoz, először meg kell hívnunk az `SDL_LockSurface` függvényt (hiszen néhány felület különleges memóriaterületeken helyezkedik el, és éppen ezért különleges elbánást igényel). Amikor végeztünk a felület módosításával, a felszabadításához meg kell hívnunk az `SDL_UnlockSurface` függvényt. A kép szélességét és magasságát a szerkezet `w` és `h` eleme adja meg, a képpontformátumot pedig az `SDL_PixelFormat` típusú elem. Az SDL gyakorta emulál nem szabványos képernyőfelbontásokat magasabb felbontású módban, az `SDL_PixelFormat` szerkezet `pitch` eleme viszont mindig a képkockatár (frame buffer) valódi szélességét adja meg. Az eltolások számításához minden esetben a `pitch` elemet használjuk a `w` elem helyett, különben a program nem fog helyesen működni bizonyos megjelenítőkön.

A *2. listán* bemutatott program az `SDL_PixelFormat` adatot használja arra, hogy önálló képpontokat rajzoljon a képernyőre. A bemutató céljára tizenhat bites (hicolor) módot választottunk, de más képernyőmódokat is éppoly könnyű lenne programozni.
A programot a megjegyzések magától értetődővé teszik, de néhány dolog talán nem annyira nyilvánvaló. Ez a program egy nagyon általános eljárást alkalmaz az SDL által felismerhető hicolor képpontok előállítására. Természetesen írhattunk volna külön eljárást minden egyes hicolor adatformátumhoz (ami nyilván gyorsabb lenne), csakhogy ez sok különmunkával járna, ezzel szemben csak kis mértékben növelné a hatékonyságot. Ugyan a hicolor 565 (5 vörös bit, 6 zöld bit, 5 kék bit) a legszélesebb körben használt képpontformátum, és így ésszerű lenne erre kiélezni a kódot, mindazonáltal az 556 és az 555 formátum sem ritka. Ráadásul semmi sem biztosítja azt, hogy a bitmezők vörös, zöld, kék sorrendben következnek. A CreateHicolorPixel eljárásunk azonban az `SDL_PixelFormat` szerkezetben található adatoknak megfelelően kezeli ezt a kérdést. Például az eljárás a szerkezet `Rloss` elemét használja annak megállapítására, hány bitet kell eldobni a 8 bites vörös összetevőből, majd az `Rshift` elem segítségével állapítja meg, hol helyezkednek el a vörös bitek a 16 bites képpontértéken belül.
Egy másik fontos dolog, amiről beszélni kell, az `SDL_UpdateRect` függvény. Amint azt már korábban említettük, az SDL néha emulálja a videomódokat, ha a videokártya nem képes azokat megjeleníteni. Tegyük fel, a videokártya nem támogatja a kért 24 bites módot, akkor az SDL például 16 bites módba vált, és visszaad egy hamis 24 bites képkockatár-beállítást. Ez lehetővé teszi számunkra, hogy gond nélkül folytassuk a programot, az SDL pedig futásidőben fogja 24 bitről 16 bitre átültetni a képpontokat (ami némi teljesítménycsökkenést okoz).
Az `SDL_UpdateRect` függvény arról tájékoztatja az SDL-t, hogy az adott képernyőterület tartalma megváltozott, és végre kell hajtania a megfelelő átalakításokat az adott terület megjelenítéséhez. Ha a programban nem használjuk ezt a függvényt, még megvan az esély rá, hogy a működni fog. Jobb azonban biztosra menni, és meghívni ezt a függvényt, valahányszor a képkockatárnak használt terület megváltozik.
Végül, ha lefuttatjuk a programot, feltűnhet, hogy ablakban fut, ahelyett, hogy a teljes képernyőterületet birtokba venné. Ha ezt szeretnénk elérni, az `SDL_SetVideoMode` hívásban a nulla helyére az `SDL_FULLSCREEN` állandót kell írnunk. Mindazonáltal legyünk óvatosak. A teljes képernyős alkalmazásokban előforduló hibákat nehezebb nyomon követni, és többnyire csúnyán összekavarnak mindent, amikor összeomlanak.

Rajzolás részletátvitel segítségével

Láthattuk, miképpen lehet képpontokat közvetlenül a képernyőre rajzolni, és nincs is semmi ok, ami miatt ne lehetne ilyen módon akár egy teljes játékot megalkotni. Mégis, nagy adatmennyiség képernyőre küldésére létezik egy sokkal jobb módszer is. A következő példánk egy teljes felületet tölt be egy fájlból, majd egyetlen SDL-függvénnyel a képernyőre rajzolja azt. A program a *3. listán* látható.
Amint láthattad, az `SDL_LoadBMP` függvénnyel töltöttük a memóriába a bitképet. A függvény vagy egy, a képet tartalmazó `SDL_Surface` típusú mutatót ad vissza, vagy NULL-t, amennyiben a képet nem sikerült betölteni. A fájl sikeres betöltését követően, a bitkép egyszerű SDL-felületként kezelhető, amit a program képernyőre vagy bármilyen más felületre rajzolhat. A bitképek dinamikusan foglalt memóriát használnak, amit fel kell szabadítani, ha a továbbiakban már nincs rá szükség. Az `SDL_FreeSurface` függvény felszabadítja a bitkép által lefoglalt memóriaterületet.
Az `SDL_BlitSurface` függvény feladata az egyik felület másikra vitele, miközben képpont-átalakítást hajt végre az egyes formátumok között, amennyiben szükséges. Ez a függvény négy bemenő értéket

1. lista: A megjelenítő felkészítése

```
#include "SDL.h"
#include <stdio.h>
#include <stdlib.h>
int main()
{
    SDL_Surface *screen;
    /* Az SDL videorendszerének üzembe
       helyezése, majd a hibák lekérdezése */
    if (SDL_Init(SDL_INIT_VIDEO) != 0) {
                printf("Az SDL üzembehelyezése
                        nem sikerült: %s\n",
                        SDL_GetError());
                return 1;
    };
    /* Bizonyosodjunk meg, hogy az SQL_Quit
    meghívásra kerül a program futása végén! */
    atexit(SDL_Quit);
    /* kisérlet a 640x480-as módba váltásra */
    screen =
SDL_SetVideoMode(640,480,16,SDL_FULLSCREEN);
    if (screen == NULL) {
        printf("Nem tudunk a megfelelő módba
                váltani: %s\n",
                SDL_GetError());
        return 1;
    };
    /* Ha eddig eljutottunk, akkor minden
    működött */
    printf("Siker!\n");
    return 0;
}
```

vár: a forrásfelületet (ahonnan a képet másolni kell), egy `SDL_Rect` szerkezetet, ami a forrásból ténylegesen átvitelre kerülő téglalap alakú területet határozza meg, a célfelületet (ahová a kép kerül), és egy másik `SDL_Rect` szerkezetet, amely a célterületet adja meg. Ennek a két területnek azonos szélességűnek és magasságúnak kell lennie (mivel az SDL jelenleg még nem támogatja a torzítást), a területek kezdő x és y koordinátái azonban különbözhetnek.

Színkulcsok és átlátszóság

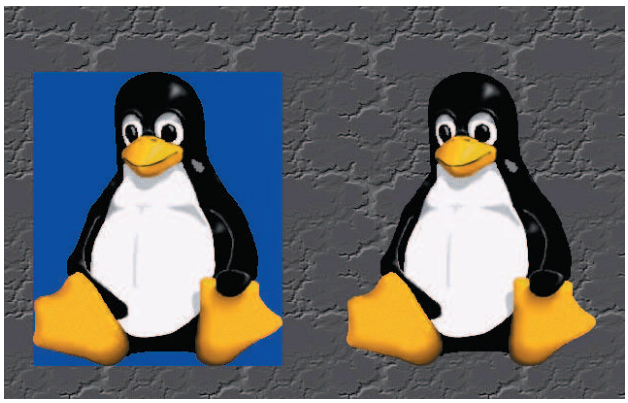
A játékokban gyakran van szükség az átlátszóság látszatára. Például van egy játékfigurát ábrázoló bitképünk valamilyen egyszínű háttér előtt, és szeretnénk kirajzolni a játék pályájára. Elég szörnyen nézne ki, ha a hátteret is kirajzolnánk, és a figura egyszínű dobozba zárva jelenne meg. Sokkal jobb lenne, ha csak azokat a képpontokat rajzolnánk ki, amelyek valóban a figura részei, nem pedig a háttéré. Ezt színkulcsos részátvitellel tehetjük meg. Az SDL ezt is támogatja, sőt, még a színkulcs tömörítéses gyorsítását (run-length colorkey acceleration) is lehetővé teszi számunkra (ez egy ügyes trükk a rajzolás meggyorsítására). Az RLE gyorsítás óriási sebességnövekedéshez vezethet színkulcsos munkáknál.
Alkalmazásuk azonban csak olyan bitképek esetében célszerű, amelyek a futás alatt nem változnak (mivel az RLE szerint kódolt kép megváltoztatásához előbb ki kell bontani, majd a változtatás után újra be kell csomagolni).
A színkulcs egy olyan képpontérték, amit a program átlátszó színnek határoz meg. Az SDL-ben ezt a SDL_SetColorKey függvénnyel lehet megadni. Azok a képpontok, amik megegyeznek a színkulccsal, nem másolódnak át a kép átvitelekor. A mi játékfigurás példánkban, a tömör hátteret kell színkulcsként megadnunk, így az nem jelenik meg. A színkulcsok révén egyszerűvé válik a téglalap alakú területen tárolt, de nem téglalap formájú tárgyak használata. A következő példánkban színkulcsos átvitelt fogunk használni arra, hogy Tux képét egy másik képre rajzoljuk rá (a lista az ➲ftp.scc.com/pub/lj/listings/issue81/ címről tölthető le). Tuxot egyöntetű kék szín előtt tároltuk, így a kék színt (RGB 0, 0, 255) fogjuk színkulcsként használni. Az összehasonlítás kedvéért színkulcs nélkül is kirajzoljuk ugyanazt a képet.

Egyszerű billentyűzetkezelés

Az SDL a billentyűzeten található minden billentyűhöz „virtual keysym” kódokat rendel. Ezeket a számokat (valós egészeket) az operációs rendszer saját billentyűkódjaira alakítja (amelyek a billentyűzet által küldött kódokra hivatkoznak), az SDL azonban a színfalak mögött felügyel ezekre az átalakításokra. Minden virtuális keysymhez egy előfeldolgozó-állandót (preprocessor symbol) rendel. Például az ESCAPE billentyűnek az `SDLK_ESCAPE` állandó felel meg. Az érvényes állandók listája megtalálható az SDL leírásában.
Ezeket a kódokat használjuk, valahányszor csak közvetlenül szeretnénk lekérdezni valamely gomb állapotát (lenyomott, vagy felengedett), illetve az SDL ezeket adja át, amikor billentyűzeteseményt küld. A virtuális keysymeket az SDLKey adattípus határozza meg.
Mivel az eseménykezeléssel most nemigen foglalkozunk, közvetlenül kell lekérdeznünk a billentyűzet állapotát, valahányszor kíváncsiak vagyunk valamelyik billentyűre. A program pillanatfelvételt kérhet a teljes billentyűzetről egy vektor formájában. Az `SDL_GetKeyState` függvény egy mutatót ad vissza az SDL belső billentyűzet-állapot tömbjére, amit az `SDLK _keysym` állandók segítségével címezhetünk meg. Ezt a függvényt csak egyetlen egyszer kell meghívni, a mutató a teljes futásidő alatt érvényes marad. A vektor minden egyes eleme egy egyszerű Uint8-típusú jel, amely azt mutatja meg, hogy az adott billentyű lenyomott állapotban van-e. Időnként meghívjuk az `SDL_PumpEvents` függvényt, ezzel frissítve a vektor értékeit.

Még, még, még!

Az induláshoz ennyi is elég az SDL-ről. Sok mindenen átsiklottunk, többek közt az animáción, az alfacsatorna használatán, valamint a hangkezelésen. Ha többet akarsz tudni ennek a könyvtárnak a programozásáról, a legjobb kiindulási hely az SDL Documentation



2. lista: Egyedi képpontok rajzolása a képernyőre

```
#include "SDL.h"
#include <stdio.h>
#include <stdlib.h>
Uint16 CreateHicolorPixel(SDL_PixelFormat *fmt,
                          Uint8 red, Uint8
                          green, Uint8 blue)
{
    Uint16 value;
    /* Az alábbi eltolások segítségével kiszá-
       mítjuk a szükséges 8 bites vörös, zöld és
       kék értékeket a 16 bites értékekből
    */
    value = ((red > fmt->Rloss) << fmt->Rshift) +
        ((green > fmt->Gloss) << fmt->Gshift) +
        ((blue > fmt->Bloss) << fmt->Bshift);
    return value;
}

int main()
{
  SDL_Surface *screen;
  Uint16 *raw_pixels;
  int x,y;

    /* A felkészítő kód jön. Létrehozunk egy
       256x256-os, 16 bites felületet, majd
       elmentjük a screen mutatóba. Lásd az előző
       példát. */
    /* A screen "zárolása", így közvetlenül raj-
       zolhatunk rá. */
     SDL_LockSurface(screen);
     /* Készítünk egy mutatót a videofelület
        i-memóriájához. */
     raw_pixels = (Uint16 *)screen->pixels;

    /* Most már nyugodtan írhatunk a videofelület-
       re. Egy szép átmenetes mintát rajzolunk, a
       vörös és a kék összetevők változtatásával.
    */
    for (x = 0; x < 256; x++) {
       for (y = 0; y < 256; y++) {
       Uint16 pixel_color;
       int offset;
       pixel_color =
         CreateHicolorPixel(screen->format, x,0,y);
       /* A módosítandó képpont memóriaeltolásának
          számítása. */
       offset = (screen->pitch/2 * y + x);
       raw_pixels[offset] = pixel_color;
       };
    };
   /* Készen vagyunk, úgyhogy megszüntetjük a
      zárolást. */
   SDL:_UnlockSurface(screen);
   /* értesítjük az SDL-t, hogy a képernyő tartal-
      ma megváltozott. Ez szükséges, ugyanis az
      SDL felülete nem a tényleges megjelenítő,
      hanem egy köztes réteg.
    */
    SDL_UpdateRect(screen,0,0,0,0);

    /* Pár másodpercig várunk, hogy a nézőnek
       legyen ideje ámuldozni. */
    SDL_Delay(3000);
    return 0;
}
```

3. lista: Nagy mennyiségű adat rajzolása a képernyőre

```
#include <SDL/SDL.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
   SDL_Surface *screen;
   SDL_Surface *image;
   SDL_Rect src,dest;
   /* Kezdeti kód, az előző példával azonos
      módon. */
   /* A bitkép beolvasása. Az SDL_LoadBMP egy
      mutatóval tér vissza, mely a képet tartal-
      mazó új felületre mutat. */
   image = SDL_LoadBMP("tux.bmp");
   if (image == NULL) {
       printf("Nem tudom betölteni a képet.\n");
       return 1;
   };
   /* Az SDL képátvitel (blitting) számára pon-
      tosan meg kell adni az átvinni kívánt adat
      mennyiségét. Ezt az src és dest  SDL_Rect
      szerkezetekkel adjuk meg. A két területnek
      azonos méretűnek kell lennie. Az SDL jelen-
      leg nem kezel torzításokat. */
   src.x = 0; src.y = 0;
   src.w = image->w; /* Az egész kép másolása */
   src.h = image->h;
   dest.x = 0; dest.y = 0;
   dest.w = image->w;
   dest.h = image->h;
   /* A bitkép kirajzolása a képernyőre. Hicolor
      módban vagyunk, ezért nem kell foglalkoz-
      unk a színpalettákkal. Képátvitelnél
      (blitting) nem kell zárolnunk a képernyőt,
      az SDL elintézi a zárolást. */
   SDL_BlitSurface(image,&src,screen,&dest);
   /* Utasítjuk az SDL-t, hogy frissítse az egész
      képernyőt. */
    SDL_UpdateRect(screen,0,0,0,0);
   /* Pár másodpercig várunk, hogy a nézőnek
      legyen ideje ámuldozni. */
    SDL_Delay(3000);
    /* A lefoglalt memória felszabadítása.. */
    SDL_FreeSurface(image);
    return 0;
}
```

Project honlapja a ➲http://www.libsdl.org. Esetleg benézhetsz a #sdl csatornára az ➲irc.openprojects.net-en, ahol valószínűleg szép számmal találsz majd több-kevesebb tapasztalattal rendelkező SDL-rajongókat.
Jó játékot és jó kódolást!

John Hall (overcode@lokigames.com) a Georgia Tech cégnél vezető kutatóinformatikus. Mindenféle linuxos játék érdekli. Amikor éppen nem önkívületben ül a billentyűzet előtt, gyakran lehet látni az egyetem környékén, amint pókszabású kedvenceit sétáltatja.

Filmvágás Linuxon? Azt megnéznénk...

A Broadcast 2000 készítői a filmkészítés művészetét a Linux erősségeivel társították.

Sok-sok évvel ezelőtt ott voltak a nagy HPUX, IRIX meg AIX gépek az egyetemi laborok sarkaiban, de azokat – vérlázító módon – csak adattárolásra és jellegzetes, központosított feladatok elvégzésére használták, illetve csak néhanapján dolgoztatták meg őket valami látványosabb célért. Aki szórakozni akart, annak egy 75 MHz-en repesztő windowsos PC-t indíthatott újra naponta tizenkilencszer.

A unixos gépek gyorsasága és hatékonysága azonban sokakat elgondolkodtatott: mi lenne, ha ilyen rendszerek segítségével végezhetnénk el a filmkészítéssel kapcsolatos utómunkákat? Sajnos, akkoriban egyetlen hasonló program sem létezett, így hiába a nagy teljesítmény és a tökéletes eszközmeghajtók – az ezeket összefogó alkalmazás híján az álom nem teljesedhetett be. Pedig egy olyan C program, mely gigabájtokban mérhető mennyiségű mozgókép- és hagfolyamokkal képes dolgozni, mindenki számára a tökéletesebb megoldást jelentette volna.

A Windows- és Mac-hirdetések hatására egyre többen változtatják otthoni számítógépüket grafikai munkaállomássá – mindeközben a Linuxra a különböző rendszerek közötti „ragasztóanyagként” tekintenek. Az Interneten barangolva egyre több olyan vállalkozó szellemű számítógép-tulajdonosra bukkanhatunk, akik elhatározták: márpedig ők csakazértis a Linux segítségével fognak filmeket vágni és feldolgozni.

A Broadcast 2000 Linux alatt tagadhatatlanul az év legnagyobb dobása ebben a témakörben. A program egy iMac tudásával vetekedő munkaállomást varázsol gépünkből: a mozgóképek rögzítése, szerkesztése és leképezése soha nem volt ilyen egyszerű, mint most.

A Broadcast 2000 minden Linux-felhasználó számára megnyitja azokat a helyeket, ahol eddig csak a Windows NT újraindításával foglalatoskodó „szakemberek” járhattak.

A csomag segítségével többórányi, nagyfelbontású képet és DVD-minőségű hangot tartalmazó adatfolyamot is megszelídíthetünk, ugyanis a két gigabájt körüli fájlméretek meg sem kottyannak neki. A Linux által nyújtott lehetőségeket a végsőkig kihasználja, és ami a legfontosabb: a nagyteljesítményű rendszerek megbízhatóságát egyesíti az alkotómunkára való képességgel.

A Linux lehetőségeiről nem könnyű meggyőzni az olyan programozókat, akik a Linuxot elsősorban a megbízhatóságot igénylő számítások elvégzésére alkalmas rendszerként ismerték és kedvelték meg. Emellett a programfejlesztő cégek hozzáállása sem túl kedvező: általában inkább más felületek támogatására szavaznak. Pedig az ehhez hasonló méretű vállalkozásokhoz hatalmas összefogásra, valamint nem kevés tőkére és rengeteg programozással eltöltött munkaórára is szükség van.

Ahogy a PC-k egyre gyorsabb ütemben fejlődnek, a Linux is egyre inkább alkalmassá válik komolyabb multimédiás felhasználásokra: reklámfilmek készítésére, CD-k utómunkáinak elkészítésére, filmek vágására és látványhatások alkalmazására. Csak a megfelelő programot kell telepítenünk hozzá...

Az egyszerű telepítés

A Broadcast 2000 telepítésének legegyszerűbb módja, ha az RPM csomagok letöltése után kiadjuk az alábbi parancsot:

```
rpm -U --nodeps --force <fájlnév>
```

Mindössze a két alábbi változatszámra kell figyelnünk:

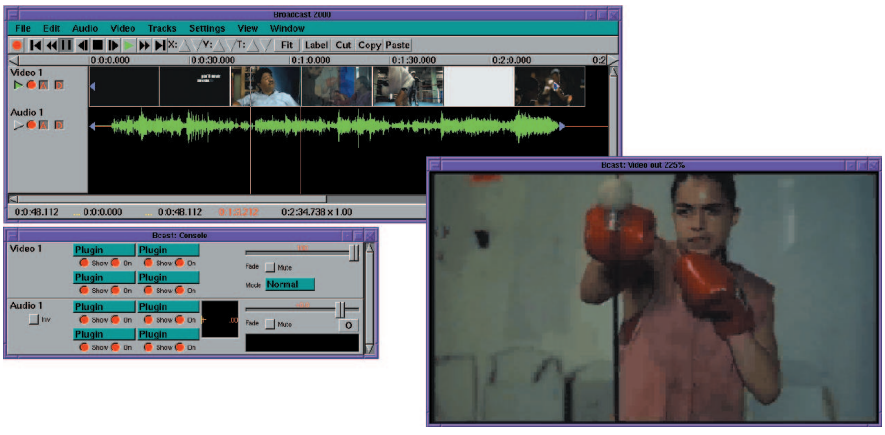
```
XFree86 4.0.1
Linux Kernel 2.2.17
```

A legtöbb gond az RPM parancs használatából adódik. A három érték kötelező, hiszen így az RPM nem foglalkozik a programhoz szükséges más csomagok jelenlétével.

A Broadcast 2000 készítőihez intézett kérdések nagy része az RPM-mel, vagy a rendszergazdai jogok nélküli telepítéssel foglalkozik. Az rpm2cpio nevű ódivatú segédprogrammal is helyettesíthetjük az RPM-et:

```
rpm2cpio <rpm fájlnév> | cpio -i --make- \
       directories
```

Ekkor a munkakönyvtárban létrejön egy, a teljes telepítést tartalmazó



Egy internetes reklámfilm, visszafelé lejátszva. Az 1998 előtt megjelent animációkat az xanim különleges kiadásának segítségével tömörítés nélküli Quicktime formátumra alakíthatjuk. Itt csak a mozgókép játszható le, mivel csak ahhoz a sávhoz van lejátszógomb. Minden sávhoz tartozik viszont felvétel gomb, így a kijelölések az összes sávot érintik.

könyvtárszerkezet, tartalmát az `ls -lR` paranccsal listázhatjuk ki.
Rendszergazdaként a /usr-be telepítsük a programot. Rendszergazdai jogok nélkül komoly kihívás a program telepítése, jól kell ismernünk a programkönyvtárak világát is. A legegyszerűbb, ha saját gépünkön kísérletezünk.
A /usr/local/bcast/bcast2000.sh nevű parancsfájl tartalmazza a könyvtárfájlok megfelelő útvonalait. Ha a /usr/local/bcast könyvtár helyett mást adtunk meg, akkor a BCASTDIR környezeti változóban be kell állítanunk az új helyet.

A bonyolultabb módszer

A Broadcast 2000-et a ➲ heroinewarrior.com honlapról letöltők egyharmada inkább a forráskódot tölti le a bináris fájlok helyett. Ez egyébként több okból is célszerű: a fordító különböző kapcsolóival nagymértékben gépünk képességeihez igazíthatjuk a programot, és a folyamat „kézi vezérlésének” köszönhetően a program használata során jóval nagyobb függetlenséget élvezhetünk.
Felhívom mindenki figyelmét, hogy ez a módszer kifejezetten fárasztó és időigényes. A futtatható változatban megtalálható könyvtárfájlok többségét a Makefile statikus könyvtárfájlokként építi fel, ezzel is egyszerűsítve a fordítást. Az utóbbi években megjelent kiadások azonban még tovább bonyolították az amúgy sem egyszerű helyzetet. A fordításhoz az alábbi csomagokhoz tartozó fejlécfájlokra lesz szükségünk:

```
XFree86 4.0.1
Linux Kernel 2.2.17
```

A telepítéshez a következő parancsokat kell kiadnunk:

```
./configure
make
make_install
```

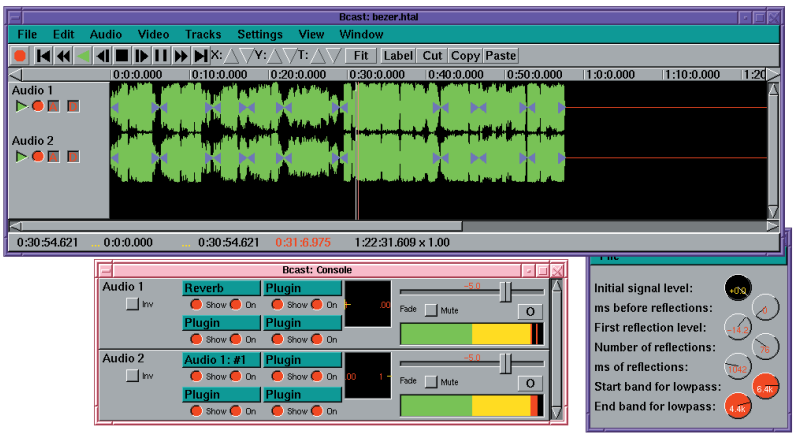
Biztos, hogy sok gond lesz a rendszer újabb változataival, és lehet, hogy némi programozásra is szükség lesz. A Broadcast 2000 indításához a `/usr/local/bcast/bcast2000` parancsot kell kiadnunk.

A Broadcast 2000 használata

A telepítés után a Broadcast 2000 több száz műveletet képes végezni az adatokkal, amelyek között a hang- és képelfogás (capture) is helyet kapott. A rendszer az egyszerű hangsávoktól a nagy sávszélességet igénylő mozgóképekig bármivel használható, de ez a felhasználótól meglehetősen sok állítgatást követel meg. A beállításhoz használható értékek leírását a /usr/local/bcast/docs könyvtárban találhatjuk. Amíg a készítők ki nem találnak valami egyszerűbb módszert, addig a beállítások kiismerésének egyetlen módja a próbálgatás.
Nézzük meg most a program használatának három jellegzetes lépését: a nyersanyag beolvasását, a szerkesztést és a mentést.

A nyersanyag beolvasása

A terméktámogatásra gyakran érkeznek levelek a sikertelen fájlbetöltéssel kapcsolatban. A legtöbben tömörített mozgóképeket szeretnének beolvasni, de 1998-ban a linuxosoknak szembesülniük kellett a rideg valósággal: az új tömörítési formátumok egyikét sem készítik el kedvenc rendszerünkre. Windows alatt már a szabványok három



A Broadcast 2000 egy 57 perces WAV fájlt játszik le egyszerű visszhanghatással. Minden „plugin” gomb egy-egy jelfeldolgozó egységet, illetve egy másik célsávot tartalmazhat. Itt a 2. sávot az 1. sáv visszhangján vezettük át; mindkét csatorna dinamikája 5 dB.

nemzedéke váltotta egymást a videofeldolgozás területén, ezek közül azonban egyik sem használható Linuxon. A gond nélkül lejátszható három formátum az MPEG-1, a tömörítés nélküli Quicktime és a DV (digitális video). A Quicktime nem is igazán önálló formátum, csupán egy közös felület több formátum számára. Néhányat ezek közül profi stúdiókban is használnak, és ezeket Linux alatt is elérhetjük. A nagy felbontású képek betöltését és feldolgozását is megoldották.
Azt azért ne felejtsük el, hogy egy magára valamit is adó stúdióban nem az Internetről leszedett animációkkal dolgoznak, és a tömörített formátumok használatát is igyekeznek mellőzni legalábbis, szerkesztés közben a végeredménynél természetesen már szóba jöhetnek ezek is. A fő forrást a bemeneteken keresztül a kameráról vagy szalagokról beolvasott adatfolyamok jelentik. A végfelhasználó által a film megtekintésére használt tömörített formátumokkal a szerkesztési feladatok közben ne foglalkozzunk. Mindent tömörítés nélküli formátumban tároljunk és olvassunk be.
A Broadcast 2000 a Video4Linux, Video4Linux 2, Firewire, LML33 csatolókat és a képernyőlopást támogatja, ezek mindegyikét használhatjuk a félig duplex hangfelvételtől kezdve a teljesen duplex mozgóképrögzítésig. Ezek a géptípus és a telepítés összetettségének függvényében használhatók. A mozgóképelfogást a rendszermag hibáinak kijavításával, önálló kódrészletek írásával valósították meg.

Az anyag szerkesztése

A nyersanyag beolvasása után a szerkesztő kezelőfelületével kell megismerkednünk. A felhasználói felületek tervezésében legújabban alkalmazott szokásoktól eltérően a Broadcast 2000 a kivágás–beillesztés típusú eljárással dolgozik. (Ez a módszer hosszú évekig megfelelő volt a képanyagok szerkesztéséhez, mára azonban kissé eljárt felette az idő.)
A mozgóképek kijelölésének legegyszerűbb módja az ide-oda görgetés és a jelzők elhelyezése. Ezt követően az időcsúszkára duplán kattintva jelölhetjük ki a jelzők közötti szakaszt. A hangsávok kijelöléséhez egyszerűen a megfelelő hullámformát kell kiválasztanunk.
A Broadcast 2000 sikeres használatának kulcsa a jelzők szerepének megértése. A régi motorosok talán még emlékeznek a „Soundedit 16” nevű programra, amely hasonló elméleten alapult.
Gyakoriak a „nem tudom különválasztani a sávokat a szerkesztéshez” típusú kérdések. A régi típusú 24 sávos hangberendezéseken 24

bemeneti/repro kapcsoló kapott helyet, ezek lehetővé tették, hogy bizonyos sávokat lejátsszunk, és ezzel egy időben néhány másik sávra felvételt készítsünk. A Broadcast 2000-ben hasonló módon jelölhetjük ki szerkesztésre (felvételre), illetve lejátszásra a sávokat.
Minden sávhoz tartozik egy-egy felvétel és lejátszás gomb. Ezekkel elérhetjük, hogy a kijelölés csak a sávok egy általunk meghatározott csoportjára vonatkozzon. Így a szerkesztés során semmi nem köti a kezünket: a hang- és mozgóképsávokat tetszés szerint keverhetjük egymással.
A legjobb az egészben, hogy szerkesztés közben a forrásfájlok nem változnak. Csak így lehetséges az, hogy több órányi felvett anyagot szempillantásnyi idő alatt átmásolhatunk egyik helyről a másikra, illetve hogy akár ötszáz szint mélységben is visszaléphetünk a végrehajtott műveletekben.

A kész anyag mentése

Linuxos körökben az egyik leginkább mellőzött témakör a már lemezen lévő forrásanyagokhoz tartozó jelzőkészletek mentése. A Broadcast 2000 két módszert használ ennek megvalósítására: a jelzők tárolását és a leképezést. A „Save As” menüponttal egy fájlneveket és a jelzők helyzetét tartalmazó szöveges fájlt menthetünk, ezt profi stúdiókban csak „szerkesztési listaként” emlegetik.
A lista éppen annyi adatot tartalmaz, amelyek segítségével tökéletesen visszaállítható a szerkesztés mentés előtti állapota, de az egyes gépek között ezen fájlokat nem csereberélhetjük. Ha valamit szeretnénk átvinni egyik gépről a másikra, ahhoz előbb le kell képeznünk az anyagot. A leképezés során egy bárhol lejátszható bináris fájl jön létre. Ha a leképezéssel végeztünk, akkor már csak az internetes felhasználást is lehetővé tevő tömörítés van hátra. A Weben fellelhető klipek nagy többségének készítése során először egy tömörítetlen RGB adatfolyamot gyártanak le, majd ennek méretét valamelyik kedvelt tömörítőeljárással csökkentik. Linuxon ezt az utolsó lépést a RealVideo, az Mpeg2Movie vagy a LAME segítségével végezhetjük el. Bár ezek egyike sem olyan népszerű, mint az eredeti tömörítők, azért ezekkel is igen jó minőséget érhetünk el.

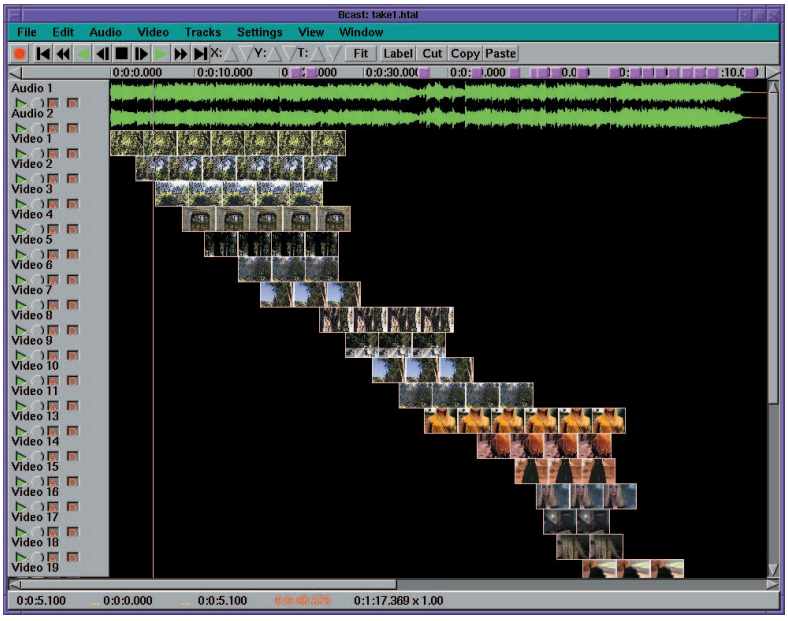
Ajánlott gépkörnyezet

Szóval most szereztük meg informatikus oklevelünket, jóval többet keresünk, mint egykori pajtásaink, és a lehetetlent akarjuk. Olyan linuxos gépet szeretnénk összehozni, amely képes adásminőségű videós munkára.
A tökéletes gép képes a tömörítés nélküli, nagy felbontású adatfolyam zökkenőmentes lejátszására. Egy szerényebb lehetőségekkel bíró munkaállomáson összeállított anyagról még tömörítés után is megállapítható, hogy kicsivel többet is költhettek volna a vasra.

A mozgókép-elfogás változatai

Típus	Eszköz	Ajánlott meghajtó
Tömörítetlen analóg	haupauge WinTV ANalog	Video4Linux 2
Firewire	Generic (általános)	ieee1394.sourceforge.net
Tömörített analóg	LML33	linuxmedialabs.com

A merevlemeznek a 45 MB másodpercenkénti olvasási sebességet is bírnia kell; nem árt, ha a videokártya másodpercenként legalább 50 megabájtot képes átpasszírozni az AGP csatolón. Ehhez, mondjuk, összekapcsolhatunk néhány 7200-as fordulatszámú merevlemezt, RAID vezérlővel. Mi az irodában a 75 MB másodpercenkénti sebességet is elértük már, mégpedig két 10 000-en pörgő SCSI merevlemezzel. A Promise RAID vezérlőjének ismertetőjében például 70 MB/mp-es sebességet ígérnek, akár IDE lemezekkel is.
Az adás hosszúságától függően úgy 80–200 GB tárolóhelyre van szükség. Egy 200 gigabájtos IDE RAID fürt 107 percnyi tömörítetlen adat tárolására képes, és ezt még megúszhatjuk 700 dollárból (kétszáz-kétszázötvenezer forint). A tehetősebbeknek azért inkább a Micronnál dolgozó ismerőseim példáját említeném, akik gondoltak egyet és felállítottak egy külön videokiszolgálót, 1 gigabites hálózati kapcsolattal. Így nem csoda, hogy a tárolás költségeit hamar le tudták faragni.
A Broadcast 2000 saját eljárásaival számolja ki a hatásokat. Óránként százezernyi állókép színeinek átállításához és az anyag szerkesztéséhez hatalmas teljesítményre van szükség. A linuxos programokat egyre inkább az általuk igényelt processzorteljesítmény és képernyőmód alapján ítélik meg. A videók világában tényleg



A hagyományos vágási módszer: minden egyes jelenethez több fényképet úsztathatunk egymásba

érdemes minél erősebb rendszerekre támaszkodni. A program használatához nem árt két 700 MHz-es processzor, hiszen a Broadcast 2000 önmaga teljesen lefoglalja az egyiket, és aki látta már kétprocesszoros gépen szárnyalni, az igen nehezen fogja megszokni az egy processzorral járó kellemetlenségeket.
Ezen igen komoly követelményekhez képest furcsának tűnhet, hogy az ajánlásban csupán 128 MB 133 MHz-es memória szerepel. A RAM ugyanis csak a mozgókép rögzítésénél válik igazán fontos tényezővé. Felvételkor a program képes arra, hogy tetszőleges számú képkockát a gyorstárba helyezve kiküszöbölje a merevlemez késése miatt jelentkező gondokat. A tömörítés nélküli képanyag lejátszása meg-megakad a 768 megabájtos próbagépünkön, amikor a program egy hisztériás pillanatában úgy hétszáz megabájt adatot söpör ki a lemezre. Természetesen a virtuális memória használatát is felejtsük el. A mozgókép elfogására többféle módszer is létezik (*lásd táblázat*). A videomagnóra az LML33 nevű eszköz segítségével küldhetjük ki a filmet.
Minden meghajtónak vannak hátrányai is. Tapasztalni fogjuk, hogy a Firewire meghajtó csak akkor képes érzékelni a kamerát, ha a modulkészletet néhányszor újra betöltjük. A Video4Linux 1 meghajtónak olykor szokása eldobni egy-egy képkockát. A Video4Linux 2 néha elveszti a szinkront, az LML33 pedig felcseréli a mezők sorrendjét. Ennek ellenére a fenti elemek bármelyikének felhasználásával építhetünk olyan linuxos gépet, amely képes tökéletes minőségű videós munkára.

A jövőről

A legtöbb analóg, videoelfogást vezérlő meghajtó kifejlesztéséhez vezető utat rengeteg fejtörés, türelem és sok elhalasztott egyetemi vizsga kövezi. Valaha azt gondoltuk, talán egyszer majd észhez térnek a gyártók és elkezdik a saját termékeikhez tartozó Linux-meghajtók megjelentetését, vagy legalább közölnek némi leírást a gépekben alkalmazott részegységekről. Nos, azóta történt ilyesmi.
Ha a grafikus gépek borsos árát tekintjük, nem is kell csodálkoznunk azon, hogy a Broadcast 2000 egyelőre a legtöbb video I/O kártyával nem működik.
A jó hír viszont az, hogy az újabb digitális tévészabványoknak köszönhetően a szakma fokozatosan megszabadul a kártyák és eljárások közti különbségekből eredő nehézségektől. A jelenleg támogatott meghajtók saját kódolási eljárást használnak, így háromhavi kísérletezés és munka után születhet meg egy-egy eszköz megbízható támogatása. Ezen meghajtók a DMA-tól a FIFO-n át az stdio primitívekig mindenféle eljárást használnak, és mindezt azért, mert az analóg kép minősége minden kártya esetében más és más. A digitális világban viszont a programozók a legjobb minőségű mozgóképpel dolgozhatnak, remélhetőleg egyetlen kódolási rendszer használatával.
A Broadcast 2000 fejlesztésének következő állomásaként a kezelői felületet alakítjuk át az újabb divatoknak megfelelően, átírjuk az egyes szolgáltatások kódját az újabb rendszermagok és C szabványok megjelenésekor, és remélhetőleg jut időnk új lehetőségek beépítésére is.
A telepítés azonban nem lesz egyszerűbb. A nehézségek itt a rengeteg változatban elterjedt könyvtárfájlok miatt jelentkeznek. Bár az erőszakos beállítófájlok és az #ifdef irányelvek átmenetileg enyhíthetik a szétesést, az igazi megoldást az összes szükséges elem (könyvtárfájlok, rendszermag-tulajdonságok stb.) egységesítése jelentené. 2000 októberében már jó néhány cégtől vásárolhattunk előre telepített, videós munkára szánt Linux-alapú gépet (ezek egyike volt a Linux Media Arts). Remélhetőleg egyre többen fedezik fel majd a Linux eleddig rejtett képességeit is.
A legújabb irányvonal szellemében a Linux felületének durvább, nehezebben felfogható részleteit elrejtjük a felhasználó elől, és az ezekhez kapcsolódó feladatok ellátását mikrovezérlőkre, kisebb-nagyobb készülékekre bízzuk. A Broadcast 2000 fejlesztésében arra a pontra érkeztünk, ahol már a rendszer szerkezetének újragondolása következhet. Olyan ez, mint amikor a filmeket egy webkiszolgálóról játsszuk le, a visszhangot meg egy átjáró képezi le. Egy munkatársam szellemes megjegyzése a linuxos gépekhez tervezett 3D webkamerával kapcsolatban:
„Ez olyan, amire csak a Microsoft lenne képes.”

Kapcsolódó címek

Broadcast 2000
➲ http://heroinewarrior.com/bcast2000.html

XAnim Exporting Edition
➲ http://heroinewarrior.com/toys.html

Mpeg2Movie
➲ http://heroinewarrior.com/mpeg2movie.html

XFree86
➲ http://www.xfree86.org/

Firewire
➲ http://linux1394.sourceforge.net/

Video4Linux 2
➲ http://www.thedirks.org/v4l2/

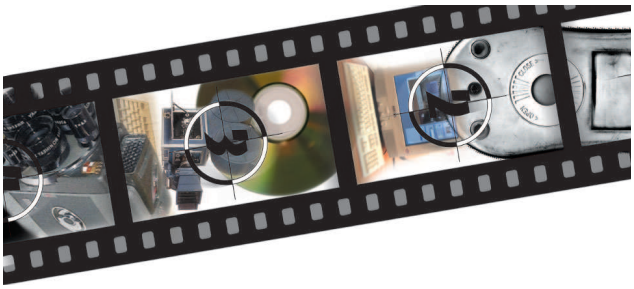
LML33
➲ http://www.linuxmedialabs.com/

Haupauge
➲ http://www.haupauge.com/

RealVideo kódoló
➲ http://www.realnetworks.com/products/

LAME
➲ http://lame.sourceforge.net/

Linux Media Arts
➲ http://linuxmediaarts.com/



Adam Williams
a Broadcast 2000 fejlesztője; jelenleg egy Silicon Valley-i cégnél tesz meg mindent a Linux képességeinek bővítéséért.

Ogg Vorbis - Hallasd a hangod!

A Nyílt Forráskód Közössége számára érdekes lehetőséget jelenthet az Ogg Vorbis az MP3-mal szemben.

A hálózat egyik legelterjedtebb felhasználása a zenelejátszás. Az átfogó hálózatokban rejlő terjesztési lehetőséget látván biztosak lehetünk abban, hogy a zeneipar mély és tartós átalakulásnak néz elébe.

A jelenlegi, szerzői jogokkal és a zeneszámok felhasználói engedélyeivel kapcsolatos pereskedéseket az Interneten robbanásszerűen terjedő hanggal és zenével kapcsolatos alkalmazások és fájlok váltották ki. A lemezipar csak most jön rá arra, amit a legelső felhasználók az első hangfájl lejátszása után már azonnal tudtak: egy új világ született a művészek, a zenehallgatók és a lemezgyárak számára egyaránt.

A felfordulás mozgatórugói azok az új módszerek, amelyek mindezt lehetővé teszik, valamint egy új eljárás, az Ogg Vorbis, amely jó eséllyel újabb lökést adhat a zenei forradalomnak.

Az Ogg Vorbis egy nyílt forráskódú, minden szabadalomtól mentes hangkodek, amelyet több más multimédiás projekttel párhuzamosan (ilyen például a cdparanoia és az Icecast) a Xiphophorus fejleszt. Összefogják a nyílt forráskódú, multimédiával kapcsolatos projektet és programozókat, akik azért dolgoznak, hogy az internetes multimédiaszabványok nyíltak maradjanak. Az Ogg Vorbison folyó munkálatokat jelenleg az iCAST – a CMGI szórakoztatóiparral foglalkozó részlege – pénzeli. Az Ogg Vorbis nyílt szabvány, és ez több szempontból is fontos, ugyanis kevés igazán nyílt szabvány létezik a digitális hangfeldolgozás birodalmában. Gondoljunk csak a Windows Mediára, a Quicktime-ra vagy a RealAudióra. Ezek a szabványok mind zártak és szabadalmak által védettek, ezért nehézkes az együttműködésük (ha egyáltalán lehet az együttműködésnek valamilyen fokáról beszélni az esetükben). Mikor tudsz lejátszani Quicktime 4-et a RealPlayerrel, vagy fordítva? Mikor lesz a Linuxnak Quicktime 4 vagy Windows Media támogatása?

A Linux és az Internet nyílt szabványokon alapulnak, és mivel a linuxos és az internetes multimédia gyorsan fejlődik, megnő az igény az olyan multimédiás alkalmazások létrehozására és fejlesztésére, mint az Ogg Vorbis. Az Ogg Vorbis két részből áll: az Oggból és a Vorbisból. Az Ogg egy héjformátum, hasonlíthatjuk például az Apple Quicktime-hoz vagy a Microsoft „Active Streaming Format”-jához. Segít összefogni az egybetartozó dolgokat. Például, ha van egy Ogg filmfájlod, az tartalmazhat egy Vorbis-folyamot egy más típusú másmilyen kódolású videofolyam mellett. Egy másik lehetőség, ha az Ogg filmfájl tíz Vorbis folyamot tartalmaz tíz különböző nyelvhez.

A Vorbis egy kodek, amit az Ogg keretrendszerben írtak. Általános felhasználásra szánt audiokodek, a legtöbb hangforrás tömörítésére jó eredménnyel alkalmazható. Nem használ alsávokat (subbanding), mint egyes más kodekek, de hasonlóan másokhoz használ vektoros kvantálást (vector quantization).

A Vorbis az egyetlen kodek, amit fejlesztettünk, de nem ez az egyetlen, amelynek az írását tervezzük, ugyanis körvonalazódik már a Squish és a Tarkin.

A Squish egy veszteségmentes audiohangkodek, ez azt jelenti, hogy a dekódolt folyam visszafejtésekor bájtról bájtra az eredetit kapjuk vissza. Ezt használhatjuk például a tőpéldányok archiválására, másolatok készítésére vagy távoli mentésre is.

A Tarkin pedig a születendő videokodekünk. A munka még javában folyik, de annyit már elárulhatunk, hogy hullámdaraboláson (waveleteken) alapul és nem MDCT-n, mint a legtöbb modern kodek (például az MPEG-4 és a JPEG). Szóval még semmi sem végleges, de a Tarkin biztató fejlesztés.

Kodekeket nehéz fejleszteni, hiszen írásuk mély matematikai ismereteket és rengeteg időt igényel. A fejlesztés befejeztével időt kell szakítani a finomhangolásra, a hibák kijavítására és új csúcsteljesítményű eszközök hozzáadására. Ezért összpontosít az Ogg Vorbis elsősorban a Vorbisra és az Ogg keretrendszerre.

Mi a baj az MP3-mal?

Sok olvasó fejében megfordulhat, hogy miért bajlódunk mi az Ogg Vorbis fejlesztésével, amikor ott az MP3, amit széles körben használnak. Mi a baj az MP3-mal? Az is nyílt formátum, nem? Hát nem!

Vajon miért nincs több MP3-kódoló program, annak ellenére, hogy mennyire elterjedt az MP3 formátum? Egy kezemen meg tudom számolni őket. Néhányan biztosan emlékeznek még a Frauenhofer intézet 1997-es levelére. Ebben az intézet kérte, hogy minden nyílt forráskódú és szabad MP3 tömörítőprogramot író csapat szüntesse be a tevékenységét, vagy fizessen jogdíjat. Körülbelül 12 szabadalom béklyózza meg az MP3 eljárásait (algoritmusait), és a Frauenhofer igyekszik mindet betartatni.

A szerzői jogdíjak beszedésének több kedvezőtlen hatása is van. A jogdíjak miatt majdnem lehetetlen ingyenes MP3 tömörítőt a piacra dobni, ugyanis a tömörítő eljárás használatának a díja letöltésenként 2,50 dollár (ha az eredeti Frauenhofer kódot használjuk, öt dollár). A legtöbb ingyenes tömörítő eltűnt, mivel nem tudták ezt az összeget kifizetni. Még a MusicMatch – a népszerű

windowsos lejátszóprogram fejlesztője – is eladta részvényei jelentős részét a Frauenhofernek a korlátlan felhasználói engedélyért. Ráadásul a Frauenhofer bármikor megváltoztathatja a játékszabályokat. 1997 előtt az MP3 tömörítők terjesztése megengedett volt. Jelenleg már csak az MP3 formátumban történő műsorszórás engedélyezett, de a Frauenhofer bejelentette, hogy ez év végétől erre is jogdíjat kíván szedni.
A RIAA (az Amerikai Lemezgyártók Szövetsége) általában zeneszámonként egyharmad vagy fél pennyt tesz zsebre letöltésenként. Ez még egészen tisztességesnek tűnik, ha azt vesszük figyelembe, hogy a Frauenhofer a bevétel egy százalékát tervezi elkérni, de számonként legalább egy pennyt (ezek az én becsléseim a jelenleg érvényben lévő vonatkozó jogdíjakból következtetve). Az MP3 formátum tényleg háromszor többet ér, mint a zene, amit hordoz? Fél dollárba kerül egy MP3 lejátszó felhasználói engedélyének díja. Ezenkívül vannak egyéb költségek is az MP3-mal kapcsolatban, bár igazából egy részük csak az én becslésem (reméljük, hogy a műsorszórásra kiszabott díj alacsonyabb lesz), de nem szabad elfelejteni, hogy a szabadalom tulajdonosa bármikor tetszése szerint módosíthatja a felhasználói engedélyek díjait. Ezt bizonyítja a műsorszórással kapcsolatban életbe lépő változás is. Nem az a lényeg, hogy a fizetendő összeg 15 000 vagy 5 dollár. A lényeg az, hogy joguk van az árat önkényesen megváltoztatni.
Az MP3 tömörítés régi eljárás. A zenebarátok és a programozók régóta együtt dolgoznak azon, hogy a tömörítésből a legjobbat hozzák ki, de az eljárás már nem változik, nem újul meg. Még a LAME-nek, az egyik legjobb MP3 tömörítőnek is vannak olyan beállításai, amelyek megsértik az MP3 szabványt, hogy jobb minőségű zenét lehessen kifacsarni a fájlokból. A szabványba egyszerűen már nem fér bele több apró trükk, nincs hely a fejlődésre.
A többi lehetőség sem túl biztató. Az Advanced Audio Coding (AAC), az MPEG-4 része, még több szabadalom foglya, mint az MP3. A legtöbb jelenleg használt eljárás fejlesztésében azonban még több cég vett részt, ez nehézkesebbé teszi a felhasználói engedélyek használatát. A VQF formátumot az NTT és a Yamaha szabadalmaztatta és féltékenyen őrzik. A RealNetwork és a Microsoft sem a nyílt szabványaikról híresek. Több leszármaztatott kodek (mint például az MP+) alkalmazása azért nehézkes, mert ugyanazok a korlátozások vonatkoznak rá, mint az eredeti MP3 tömörítőkre.
Az MP3-mal kapcsolatban jelentkező gondok megoldására azonban égető szükség van, hiszen az internetes hangátvitel állandóan fejlődik. Nem meglepő, hogy a megoldást a Nyílt Forráskód Közössége szolgáltatja.

Ogg Vorbis vagy MP3?

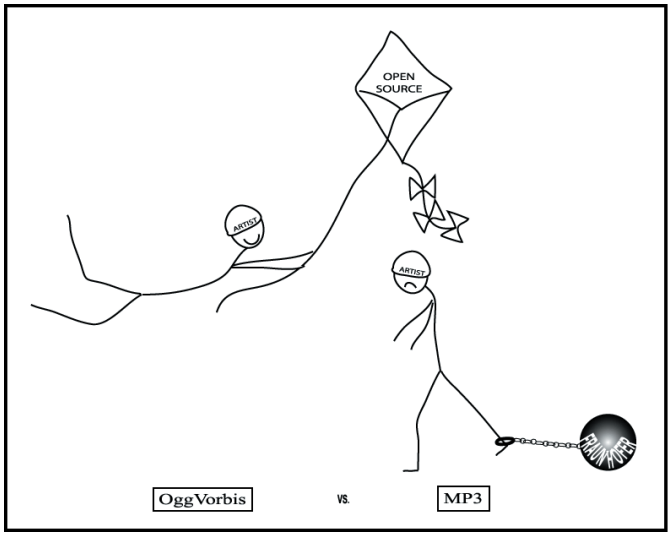
Az Ogg Vorbis szabadalmaktól mentes, és ezt így is tervezték a kezdetek kezdetétől. A használatához, legyen az magáncélú vagy kereskedelmi, nem kapcsolódik jogdíj. A forráskódja nyílt, az LGPL felhasználói engedély vonatkozik rá, tehát még a forrás is elérhető cégek, valamint érdeklődő programozók számára.
Természetesen csak a szabadság önmagában nem elég. A Vorbis jobb minőségű hanggal szolgál, ez el is várható egy következő nemzedékbeli hangkodektől. A bővíthető formátumnak köszönhetően a Vorbis minősége még éveken keresztül javulhat anélkül, hogy ez befolyásolná a régebbi kodekek működését. A Vorbis hangja már most is jó, de nem hasonlítható a fél év múlva elérhető Vorbiséhoz.
A minőség nem az egyetlen előnye Vorbisnak, ugyanis a formátum több egyedi technikai megoldással áll elő: ilyenek a bővíthető megjegyzések, a változtatható bitsebesség (bitrate peeling) és a nyers kodekcsomagok elérésének lehetősége. A megjegyzések elférnek

magában a formátumban, nincs többé szükség a ronda és korlátos ID3-as bűvészkedésekre. A Vorbis a megjegyzéseket *név = érték* formátumban tárolja. Léteznek előre létrehozott szabványos mezők, ezeket a lejátszóknak illik támogatniuk, de lehetőség nyílik saját egyedi mezők létrehozására is, ha erre van szükségünk.
A változtatható bitsebesség lehetővé teszi, hogy a folyam bitsebességét lejátszás közben csökkentsük, nem kell újrakódolni a folyamot alacsonyabb sebességhez. Ezt úgy érhetjük el, hogy a legfontosabb adatokat a csomagok elejére tesszük. A folyamot bármikor csökkenthetjük, egyszerűen levágjuk a csomagok végét, mielőtt kiküldenénk ezeket. Képzeljük el, hogy úgy hallgatunk rádiót, hogy az adás minősége az elérhető sávszélesség függvényében változik. Ha kimaradnak csomagok, a kiszolgáló kisebb folyamot küld, amikor befejeződik a letöltés, ismételten több adatot kapunk.
A műsorszóráshoz (multicast megoldásokhoz) vagy más különleges alkalmazásokhoz fontos lehet, hogy elérhetők a nyers Vorbis csomagok. Nem kell kiegyeznünk a kezdő és befejező szünetekkel a számok elején és végén. A Vorbisban van mintadarabolás a számokban történő mozgáskor és visszafejtés közben. Emlékszel azokra a szünetekre a számok között a kedvenc Trance CD-den? A Vorbisszal eltüntethetők. Meg kell keresned pontosan a 303054. mintát?
A Vorbisszal nem gond. Ezzel alkalmasabbá válik az ipari felhasználásra, mint amilyen az MP3 valaha volt.
A felhasználók és a fejlesztők egyaránt értékelni fogják a jó minőségű példakönyvtárakat. Ezek segítségével nem kell mindenkinek, aki lejátszót akar írni, egyben saját visszafejtőt is írnia. A fejlesztőknek kevesebbet kell a fájlformátummal foglalkozniuk, több időt fordíthatnak más hasznos dolgokra, ennek köszönhetően hatékonyabb és hasznosabb programok születhetnek.

Helyzetkép

Két és fél év Vorbis-fejlesztés után 1999 júniusában (a Vorbis mindig csak mellékes feladat volt) megszületett az Ogg Vorbis béta1. Csak egy bitsebességet támogatott, de a legtöbb lejátszóhoz voltak bővítményei és több felülethez is elkészült.
Az Ogg Vorbis béta2-t augusztusban mutatták be a LinuxWorld Expón San Joséban. Ez már öt bitsebességet ismert a 128 kb/mp és 350 kb/mp közötti tartományban. Jelenleg közeleg a béta3-as kiadás, amelyben jelentős minőségi javulást figyelhetünk meg. Ezt számos

zeneértő fül és szakavatott programozó hozzáértő segítségének köszönhetjük. A kódot átszerveztük egy állandó API köré, és több új eszközzel bővítettük a kiadást.
Sokat javítottunk a kódon, így a visszafejtő kétszer olyan gyors, mint volt. Felkészítettük a kódot arra az eshetőségre, ha valaki a Vorbist egész számos aritmetikával akarja megvalósítani. Ez megkönnyíti a gépi lejátszást, mely főleg a beágyazott rendszereknél fontos, könnyebben tudják ezentúl támogatni az Ogg Vorbis-lejátszást.
A három hónap alatt több mint százezren töltötték le az Ogg Vorbist, más termékek is egyszerűen csodálatosan támogatják.
Az Xmms, a Freeamp és a Kmpg már tartalmaznak Vorbis-lejátszást (az olyan elterjedt windowsos lejátszók is, mint a Winamp és a Sonique). A LAME elő tud állítani Vorbis és MP3 fájlokat is, valamint egy lépésben át tudja kódolni az MP3 fájlokat Vorbis formátumba. Többen jártak sikerrel a Grip nevű CD-összeállító programmal, és naponta jelennek meg új alkalmazások.
Néhány újdonságok iránt érdeklődő tartalomszolgáltató szintén felkarolta a formátumot.
A ➲Vorbisonic.com és az ➲eFolkmusic.com oldalakról letölthetők Vorbis fájlok.
Ezenkívül még több Vorbisszal foglalkozó cím található a ➲http://www.vorbis.com weblapon.
Röviddel a bétaváltozat után rákerestünk a weben a „vorbis” szót tartalmazó tartománynevekre, ennek alapján elmondhatjuk, hogy sokan jegyeztek be ilyen tartományokat. Több Vorbisszal kapcsolatos portál is megjelent, ilyen a ➲govorbis.com vagy a ➲vorbiszone.com.



Merre tovább?

A kód finomítását még csak most kezdtük el. Ha a visszafejtés oldaláról nézzük a folyamatot, az Ogg Vorbis majdnem olyan gyors, mint az MP3-lejátszók, és a közeljövőben utol fogja érni ezeket. Többen jelezték, hogy megbízhatóan működik a lejátszás 120 MHz-es Pentium processzorral. Ha a kódolást nézzük, a fájlok előállíthatók egy gyors Pentium II-es vagy Pentium III-as gépen. Most, hogy az API állandósul, kialakulnak a sajátságai, egyre többen foglalkoznak a sebesség kérdésével.
A Vorbist az MP3-hoz hasonlítani majdnem igazságtalan, hiszen a Vorbisban nincsen csatornakapcsolás (channel coupling). Itt is trükkös szabadalmi gondok állják utunkat, de a fejlesztők megtalálták a megoldást: szemet vetettek az Ambisonicsra. Az Ambisonics szerzői jogvédelem alatt állt, de a szabadalma már elévült.
Az Ambisonics cége csődbe ment versenytársának, a Dolbynak „köszönhetően”. Az Ambisonics igazi 3D-s térhangzást adhatna a Vorbisnak, amit tetszőleges számú hangfalra szét lehetne osztani, mindezt négy csatorna segítségével (egyes és kettes a sztereó hangnak, hármas a vízszintes térhatás hangoknak, négyes a függőlegesen helyezhető (spherical) hangoknak). A csatornakapcsolás használata könnyedén, akár 40 százalékkal is csökkentheti a bitsebességet.
A folyamok kezelése szintén előkelő helyet foglal a listán.
A műsorszórás jelenleg az ellenőrzésnél tart, a tervek szerint néhány adóállomást hónapokon belül üzembe helyezünk. Röviddel azután az Icecastba beépítjük a Vorbis támogatást, ez lesz a termék által támogatott elsődleges formátum. Ez jobb minőségű vételt tesz lehetővé az internetes rádiózás szerelmeseinek, és menekvést jelenthet a műsorsugárzók számára a 2001 végén életbe lépő felhasználói engedélyek díjai elől.
A folyamok szórásához az alacsony bitsebességek kulcsfontosságúak. Jelenleg a legalacsonyabb bitsebesség, ami a mintacsomagban szereplő kódolóval elérhető: 128 kb/mp. A folyamok adásához általában 24-64 kb/mp-re van szükség, ezért a közeljövőben arra fogunk összpontosítani, hogy alacsony bitsebességek mellett is elképesztően jól szóljon a Vorbis.
Az alacsonyabb mintavételezéssel is foglalkozni szeretnénk a közeljövőben.
Mint mindig, természetesen tovább foglalkozunk azzal, hogy a hallható zene minősége egyre tökéletesebb legyen, tesszük ezt új minőségjavító sajátosságok hozzáadásával, valamint a hallható hibák kijavításával.
A cikk megjelenésekor az Ogg Vorbis 1.0 a fent említett tulajdonságokkal megjelent.

Térnyerés az MP3-mal szemben

Sokan kérdezik, hogy hogyan tervezzük az MP3-mal szembeni térnyerést. Egyesek úgy gondolják, hogy ez nem sikerülhet. Én úgy gondolom, hogy igen. Nem hasonlíthatjuk igazán a Vorbist más hangkodekekhez. Bár ezek szintén azt próbálták elérni, amit mi, de a többi kodek sem szabadabb és nyíltabb az MP3-nál. Ez utóbbi azért válhatott ilyen népszerűvé, mert sok program támogatta, hiszen a kód elérhető volt, ott hevert az Interneten a leírásokkal együtt. Ez arról szólt, hogyan használhatjuk, vagy hogyan írhatunk sajátot, bárki számára elérhető módon. Egyesek az MP3 kontra Vorbis viadalt a Betamax kontra VHS küzdelemhez hasonlítják, ahol a Beta bár műszakilag jobb volt, mégis alulmaradt. Akik ezt gondolják, nem veszik észre, hogy a VHS azért tudott szélesebb körben elterjedni, mert műszakilag nyíltabb volt.
Két célcsoportunk van: az előadók és a fejlesztők. Az előadóknak és a tartalomszolgáltatóknak azért van szükségük a Vorbisra, hogy elkerüljék súlyos százalékok kifizetését egy német iparvállalatnak. Ezek az emberek általában abban is érdekeltek, hogy zenéjük a lehető legjobb minőségben legyen meghallgatható. Az emberek nem azért fognak az MP3 vagy a Vorbis mellett dönteni, mert egyik vagy másik formátum több műszaki újdonságot tartalmaz. Az emberek

azokat az előadókat akarják hallgatni, akiket szeretnek, abban a témakörben, ami érdekli őket, továbbá azt akarják, hogy a zene elérhető, átvihető és könnyen kezelhető legyen.
A fejlesztők multimédiával szeretnék gazdagítani az alkalmazásaikat. Itt nem csak a visszafejtésről és a lejátszásról van szó, a multimédiás tartalmat létrehozó alkalmazások fejlesztése a nyílt forráskód világában csak akkor lehetséges, ha nyíltak a multimédiás szabványok és szabadalmaktól mentesek az eljárások, mint az Ogg Vorbis esetében. A Vorbishoz egyszerű programot írni, a programozók rövid idő alatt össze tudnak dobni egy Vorbis-lejátszó bővítményt, még akkor is, ha nem ismerik régóta a Vorbist és az API-ját.
Ha a tartalmat Vorbis formátumban állítjuk elő, és minden lejátszó támogatja a Vorbist, a felhasználók számára észrevétlen lehet a váltás.
A felhasználó dolgának mgkönnyítése szempontjából a kulcs az átlátszóság. Lehet, hogy még évek múlva is „MP3” néven emlegetjük a hálózatról letölthető zenét, ahogyan akad, aki a mai napig „frigidaire”-nek hívja a hűtőszekrényt, pedig a jelenleg alkalmazott megoldások több gyártótól származnak.

Kapcsolódó címek

Az Ogg Vorbis honlapja:
➲http://www.vorbis.com
A Xiphophorus fejlesztői csapat honlapja:
➲http://www.xiph.org

Hogyan segíthetsz?

Mint minden nyílt forráskódú projekt, a Vorbis is akkor aknázza ki a legjobban a rendelkezésre álló lehetőségeket, ha megkapja a közösség támogatását. Ehhez szüksége van programozókra, zenebarátokra és zenészekre egyaránt.
Tömörítsük a zenét a Vorbis segítségével, hallgassunk Vorbis fájlokat, és mindenki szóljon, ha valami miatt nem hangzanak jól, ugyanis a hibák kijavítása általában könnyen megy, ha sikerül megtalálni őket.
Ha olyan projekttel foglalkozik valaki, amibe belefér a hanglejátszás, próbálja ki a Vorbis kodeket. Nem csak a Vorbist használók tábora gyarapszik, de a felhasználók élvezhetik a Vorbis által nyújtott lehetőségeket is.
Ha zenét készítenek, terjesszék Vorbis formátumban, ne MP3-ban. Vorbis fájlok használatával megkerülhetők a szabadalmak tulajdonosai által állított korlátok és növelhető a Vorbis iránti kereslet. A nyílt szabványok – mint amilyen a Vorbis – népszerűsítésével eltöltött idő hasznos.
A Vorbis még nagyon fiatal projekt, őszintén örülünk minden segítségnek!

Összefoglalás

A nyílt szabványok használata az internetes multimédiában megvalósítható célkitűzés és érdemes is törekedni rá, főleg egy ilyen jó minőségű, nyílt forráskódú hangkodekkel, mint a Vorbis.
Ahogy a HTTP, az FTP, a TCP/IP vagy más nyílt szabványok megváltoztatták a hálózat világát, úgy a mi célkitűzésünk, hogy átalakítsuk a multimédiáét a jobban szóló, a szebb, valamint a zárt forráskódú vagy szabadalmakkal sújtott változatokénál jobban együttműködő eszközökkel.
Valószínűleg olyan operációs rendszert használnak olvasóink is, ami a legbelső magjától kezdve nyílt szabványokra és nyílt forráskódra épül, miért ne várhatnánk el ugyanezt a használatunkban lévő multimédiás alkalmazásoktól?

Jack Moffitt hatéves kora óta programoz, írt már a játékprogramtól a hangkártyameghajtón és a kapcsolattartó programon keresztül a titkosító eljárásokig mindenfélét. 2000 januárja óta az CMGI cég iCAST nevű szórakoztatóiparral foglakozó részlegének műszaki igazgatóhelyettese és az Icecast, az Ogg Vorbis és a Tarkin nyílt forráskódú fejlesztésével foglalkozó kutatócsoport vezetője. Célkitűzéseik között megtalálható egy nyílt, internetes multimédia-keretrendszer létrehozása, melynek legfontosabb tulajdonságai közé tartozik a jó minőség és a megbízhatóság, valamint a jó együttműködési képességek.

FIASCO – egy nyílt forráskódú fraktálképkodek

A FIASCO a mai igényeknek megfelelő kép- és mozgókép-tömörítést kínál.

Egy kép többet mond ezer szónál – hangzik el gyakran érvként, ha a digitális képfeldolgozás létjogosultsága a kérdés. És valóban, nem is hinnénk, hogy életünk milyen nagy részét meghatározzák a digitális képek. Például a weboldalak nemcsak képeket, de egyre gyakrabban kisebb-nagyobb animációkat is tartalmaznak, melyek mind-mind a szépérzékünknél fogva igyekeznek az adott honlap közelébe csábítani bennünket. A digitális képek használatának egyetlen nagy hátránya, hogy minden kérelem alkalmával nagy mennyiségű adat átvitelére van szükség. Például egy tömörítetlen HDVT (1280×720 képpont, 24 bites színmélység) videofolyam egyetlen képkockájának tárolása 2 megabájtot emészt fel. Másodpercenként 60 képkockával számolva egy másodpercnyi mozgókép már 165 MB-ot foglal el, ez azt jelenti, hogy egy kétórás filmet ebben a formátumban 2000 CD-n tárolhatnánk! Ilyen irdatlan mennyiségű adat letöltése a jelenlegi körülmények között egyszerűen lehetetlen, és ezen a nagy sebességű internetkapcsolatok elterjedése (kábeltévé, ADSL stb.) sem segít. A FIASCO-hoz hasonló fraktálkép- és filmkodekek használata elengedhetetlen ekkora mennyiségű adat esetén.

A kép és a mozgókép tömörítése

A kép adatainak összepréselésére rengeteg megoldás született már, de vannak, akik a képkockaszám vagy a felbontás csökkentésétől sem riadnak vissza. Azonban az így elért nyereség még mindig édeskevés, ráadásul legtöbbször a minőség rovására megy. Aki mozgóképet szeretne tömöríteni, az általában az alábbi három irány valamelyike felől „támadja” a hatalmas memóriaigényű tárolást:

- Az egymás közelében fekvő *képpontok* ismétlődése, hasonlósága.
- A három *alapszín* (vörös, zöld, kék) előfordulásának és keveredésének aránya, s az itt jelentkező ismétlések.
- Az egymást követő *képkockák* hasonlósága.

Minden képtömörítő eljárás legfőbb célja, hogy ezen ismétlődéseket felismerje és kihasználja. A két legáltalánosabban használt tömörítési módszer a következő:

- *Veszteség nélküli*, azaz visszafordítható. A kibontott kép pontról pontra megegyezik az eredetivel (a fájlméret általában a fele). Ez a módszer akkor hasznos, ha a tömörített képpel további feladatokat kívánunk végezni.
- *Veszteséggel járó*, tehát visszafordíthatatlan. A tömörített kép néhol (kisebb-nagyobb mértékben) zavaros lehet, de cserébe az eredeti fájlméret kevesebb, mint egytizedére tömöríthetjük a képet. Ez a módszer kifejezetten alkalmas a kis sávszélességű felhasználásra (például a Webre).

A fentiekből következik, hogy a legtöbb helyen sajnos ma még az elég alacsony sávszélességgel (16–64 kb/mp) működő Internet-kapcsolatban ez utóbbi módszert részesítjük előnyben. Az elmúlt tíz évben több minőségromlással járó képformátum terjedt el széles körben, elég csak a mindenki által ismert JPEG, MPEG vagy H.263 fájlokra gondolnunk. Ezek legtöbbjét szerencsére már továbbfejlesztették, és így találkozhatunk a JPEG2000, az MPEG-4 és a H.263+ típusokkal is (lásd a Kapcsolódó címeket). Ezek mellett jó néhány olyan eljárás vált ismertté, amelyek még nem épültek bele egyik szabványba sem, ennek ellenére ígéretesnek tűnnek. Jelen pillanatban a hullámdarabolás-alapú képtömörítő rendszerek jelentik tudásunk csúcsát. Azonban ezen fájlok visszaalakításához (azaz megtekintéséhez) túl hosszú időre van szükség, ezáltal a program-alapú, valós idejű visszaalakítás majdnem lehetetlen. Emellett a tulajdonosok a legtöbb eljárást szigorú előírásokkal, kétségbe vonható értelmű és értékű szabványokkal igyekeznek megóvni az avatatlan kezektől, így a nyílt forrású fejlesztés is csak egy helyben topog.

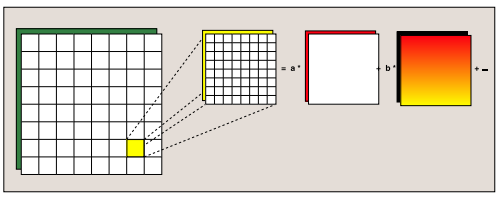
Képtömörítő eljárások

Az alacsony sávszélességű környezetekhez tervezett FIASCO (Fractal Image And Sequence Codec) a JPEG és MPEG formátumok helyettesítőjeként léphet színre. A rendszer három legfontosabb tulajdonsága:

- A mai igényeknek megfelelő kép- és mozgókép-tömörítés (egy alkalmazásban egyesítve).
- Valós idejű, programalapú kibontás.
- Nyílt forráskód.

A FIASCO által tömörített képek sokkal kisebbek, mint a hasonló JPEG formátumú változataik, a képminőségük azonban kielégítő. A FIASCO a tömörítési eljárás során tulajdonképpen a JPEG szabályait használja föl, de nagymértékben egységesíti és egyszerűsíti is azokat. A JPEG algoritmus a képet 8×8 képpontból álló négyzetekre bontja, majd ezek mindegyikét a *2. ábrán* látható módon 64 alapkép lineáris kombinációjával tömöríti (koszinuszalap, lásd a kapcsolódó címeknél). Ezen blokk-közelítések együtthatóit a tömörítő kiegyensúlyozza, majd egy fájlban tárolja.
Az ábrán látható kép 8×8-as tömbökből áll, ezek mindegyikét a koszinuszalap lineáris kombinációjával közelítjük meg. A közelítés együtthatóit (az ábrán a és b betűk jelölik ezeket) kiegyensúlyozzuk, majd a JPEG fájlban tároljuk. A FIASCO ezen lépéseket a következő módszerekkel egészíti ki:

1. A képet alkalmazkodó módon altömbökre (négyzetekre vagy téglalapokra) bontjuk. Az összetettebb részeken kisebb tömbmérettel dolgozunk.
2. Minden képtömböt egy dinamikus „szótár” képtömbjeivel közelítjük. A szótár nemcsak az alapképtömbökkel (például a JPEG esetében használt koszinusztömbökkel) dolgozik, hanem az összes addig tömörített tömböt is tárolja és hasznosítja. Ez a módszer az LZW-hez hasonló szövegtömörítési eljárásokra emlékeztet, amelyek szintén folyamatosan bővülő szótárat használnak.
3. A mozgóképek tömörítésekor csak az egymást követő képkockák különbsége tárolódik.



A JPEG tömörítési eljárás

A FIASCO az egyes képkockákon, illetve a mozgóképen belüli hasonlóságokat használja föl, és megkísérli kiszűrni a térbeli, színbeli és időbeli ismétlődéseket.

A FIASCO forráscsomag

A FIASCO a GNU GPL szabályozása alapján szabadon terjeszthető. A csomag parancssorból indítható alkalmazásokból, valamint a képek, a mozgóképek tömörítését, kibontását és megjelenítését végző könyvtárfájlból áll. Telepítése a hagyományos módon történik (./configure; make; make install). A program teljes egészében ANSI C nyelven íródott, és a legtöbb Unix-rendszeren működik.

Parancssori alkalmazások

A parancssorból indítható FIASCO tömörítő- és kibontóprogramok használata az IJPEG (Independent JPEG Group) csomagban található cjpeg és djpeg parancsokhoz hasonlít, melyek minden Linux-változatban megtalálhatók.

```
cfiasco --quality=10 --output=video.fco \
        frame0*.ppm
```

A fenti sor a megadott (PNM formátumú) videofájlokat egyetlen video.fco nevű FIASCO fájlba írja ki. Megjegyzendő, hogy a FIASCO és a JPEG minőségi értékei nem arányosak egymással. A FIASCO esetében ezen érték jellemző tartománya 1–100, mely egy 5-ösnél kisebb minőségi értékű JPEG fájllal vethető össze. A FIASCO alacsony sávszélességű környezetekhez készült, tehát egy 75-ös JPEG képpel azonos minőségű FIASCO fájlt már nem tudunk elkészíteni.
A dfiasco, a FIASCO kibontóprogramját a djpeg-hez hasonlóan indíthatjuk:

```
dfiasco --output=image.ppm image.fco
```

Ez a sor a megadott FIASCO fájlt PPM formátumra alakítja. A dfiasco azonban magában foglal egy megjelenítő modult is, mellyel a mozgóképet egy X11 ablakban tekinthetjük meg. Jelenleg csupán egy Xlib-alapú, egyszerű szolgáltatásokkal (lejátszás, leállítás, előre stb.) bíró változat létezik, de ez az egyszerű kibontóprogram és a FIASCO könyvtárfájl megmozgatja a fejlesztők képzelőerejét és elkezdik a FIASCO formátumot támogató modulok elkészítését kedvenc képfeldolgozó, képmegjelenítő programjaikhoz (Mozilla, GIMP stb.). A FIASCO honlapján (lásd Kapcsolódó címek) számos FIASCO képet és animációt találunk.

A FIASCO könyvtárfájl

A FIASCO tömörítő és kibontási képességei egy megosztott könyvtárfájlon keresztül is elérhetők. Az adattípusokat és a függvények mintapéldányait a fiasco.h nevű fejlécfájlban találhatjuk meg, mely a <telepítés_helye>/include könyvtárban található. Saját alkalmazásainkból a következő függvényhívással tömöríthetünk képeket vagy mozgóképeket:

```
fiasco_coder (image_names, fiasco_name, \
        quality, NULL)
```

Ez a hívás az image_names tömbben megadott fájlokat a quality-ban meghatározott minőségi értékkel tömöríti, és a fájlt a fiasco_name néven tárolja. Egy elhagyható paraméterobjektummal a művelet további tulajdonságait is beállíthatjuk. Ennek használatát a FIASCO leírása tartalmazza. Ha alkalmazásainkban FIASCO fájlokat szeretnénk kibontani, akkor a fiasco_decoder_t osztály fiasco_decoder_new (fiasco_name, NULL) függvényére lesz szükségünk. A fiasco_name a kibontandó fájl nevét határozza meg; a második (elhagyható) érték pedig a művelet további tulajdonságait tartalmazza.
Az egyes képkockákat ezután a fiasco_decoder_get_frame függvény többszöri meghívásával csomagolhatjuk ki. Ezek a képkockák ezután a FIASCO belső formátumában állnak rendelkezésünkre, amelyeket ezután az alkalmazás igényei szerint képezhetünk le. A könyvtárfájlról további adatokat a FIASCO leírásában olvashatunk.

Kapcsolódó címek

Department of Electrical Engineering, University of British Columbia, Canada
TMN H.263+ tömörítő/kibontó 3.1.2
➲ http://spmg.ece.ubc.ca/h263plus/h263.html

International Telecommunication Union (ITU), Genf, Svájc.
Video Coding for Low Bit Rate Communication, 1996. V. H.263 javaslat

JPEG, JBIG
➲ http://www.jpeg.org/

MPEG ISO Committee
➲ http://drogo.cselt.stet.it/mpeg/

M. Rabbani: Digital Image Compression Techniques, SPIE Optical Engineering Press, 1991.

Myers W. Carpenter et. al.: Ogg Vorbis: An Open-Source Audio Codec
➲ http://xiph.org/

T. Sikora: Mpeg Digital Video-Coding Standards, IEEE Signal Processing Magazine, 14(5):82-100, 1997. IX.

Ullrich Hafner: Fiasco: A Fractal Image and Sequence CODEC
➲ http://ulli.linuxave.net/fiasco/

Ullrich Hafner: Low Bit-Rate Image and Video Coding, Mensch und Buch Verlag, Berlin, 1999.

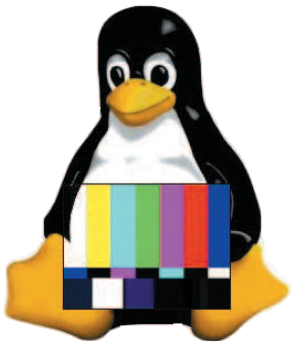
W. B. Pennebaker-J. L. Mitchell: JPEG Still Image Data Compression Standard, Van Nostrand Reinhold, 1993.

Összegzés

A FIASCO hatékony képtömörítő eljárás, mely alacsony sávszélességet igénylő környezetekben helyettesítheti a JPEG és MPEG formátumokat. A tömörítés valamivel hosszabb időt vesz igénybe, de ezt kárpótolja a gyors, programból történő kibontás és a rendkívül kicsi fájlméret. A FIASCO kifejezetten olyan alkalmazásokhoz készült, ahol tömörítésre csak egyszer, kibontásra viszont nagyon gyakran van szükség (például a weben). Ha a FIASCO-t egy nyílt forrású hangformátummal ötvöznénk (mint például a Vorbis – lásd Kapcsolódó címek), végeredményben egy teljesen nyílt forráskódú, ingyenes, bármilyen célra használható, hatékony videotömörítő rendszert kapnánk.

Dr. Ullrich Hafner
(hafner@bigfoot.de, ➲http://ulli.linuxave.net)
1990 óta szoftvermérnök Németországban. Doktori vizsgájára készítette a FIASCO-t.

A Linux és a videó



Robin Rowe megismertet minket a linuxos multimédiaprogramok fejlesztésének egy új és hatékony lehetőségével.

A Digital Domain, a film különleges hatásainak leképezéséhez több mint száz, DEC Alpha-alapú, Linuxot futtató gépet használt. Azonban ezek mellett 350 SGI IRIX, illetve száz Windows NT gépen végezték a gyártás művészeti részét. 1997-ben tehát a Linuxot még nem tartották megfelelőnek a videós munkához, csupán a megbízhatóságot igénylő számításokhoz. Nézzük most meg, mi változott e téren az utóbbi néhány évben!

Ahhoz, hogy a Linux komoly szerepet játszhasson a televíziós és filmes munkában, ahhoz tökéletes grafikai támogatással, a hatalmas fájlméretek miatt nagy teljesítményű lemezkezeléssel, az analóg és digitális eszközök összehangolásának képességével és természetesen a célra alkalmas programokkal kell bírnia. Mivel a Linux egyre inkább megfelel ezen követelményeknek, a videós alkalmazásokat fejlesztő cégek (jómagam is egy ilyennek dolgozom) máshogy kezdenek tekinteni a Linuxra.

A „GFX” a videós szaknyelvben (de a játékosok körében is) a grafika rövidítése. Én akkor hallottam először, amikor egy NBC tévétársaságnál szakmai igazgatóként dolgoztam. Rovatomban a Linuxszal kapcsolatos tapasztalataimat osztom meg olvasóimmal: multimédiás alkalmazásokat telepítek, kipróbálom azokat, és arról is szó esik majd, hogy milyen érzés Linuxra teljesen az alapoktól kezdve hasonló alkalmazásokat írni. Jómagam leginkább a C++ és Java nyelvű videóalkalmazás-fejlesztéshez értek, és ez az a terület, mely minden operációs rendszeren és gépi környezetben az egyik legkényesebb témakör. Ugyanis a videós alkalmazások támasztják a rendszerrel szemben a legkomolyabb követelményeket, hiszen nagymértékben leterhelik a processzort, a lemezkezelő alrendszert és egyéb alkatrészeket is.

A saját megélhetésükért programokat fejlesztő cégeknek azonban gyakorlati szempontok alapján kell dönteniük, amikor operációs rendszert választanak. A rendszer legyen népszerű, nagy teljesítményű, és könnyű legyen programokat fejleszteni alá. Néhány évente az általánosan használt rendszer feledésbe merül, és egy újabb lép a helyébe. Az utóbbi négy évben a legtöbb fejlesztést Windows NT alatt végeztük, és néhány kódrészletet Solaris, valamint Alpha-rendszerekre is átvittünk. A Windows használatára leginkább az olcsó alkatrészek óriási választéka ösztönzött bennünket (MPEG-1, MPEG-2 tömörítőkártyák stb.). A WinNT előtt Solarist futtató, Parallax Motion-JPEG kártyával ellátott Sparc20-ra fejlesztettünk, azelőtt pedig egy egyszerű SGI Indigóra, melyen IRIX operációs rendszer futott. A videós alkalmazások fejlesztői hamar megtanulják, hogy sosem szabad egy operációs rendszerben túlságosan megbízni. Sokáig a Macintosh volt az egyetlen nagy teljesítményű, de viszonylag kedvező árának köszönhetően mindenki számára elérhető videós rendszer, ezt a helyzetet azonban az újabb Windows-változatok előretörése villámgyorsan megváltoztatta. A Windows 98 és a Windows Millennium Edition a nagy teljesítményű grafikát használó játékok legfőbb operációs rendszere. Megbízhatósági és biztonsági okokból azonban a Win9x és ME használata a folyamatos működést igénylő környezetekben (videós utómunka, adássugárzás) nem javasolt. Ugyanezen okokból tiltották meg például az Egyesült Államok védelmi minisztériumában is e rendszerek használatát.

A minisztériumban egyébként számos grafikai rendszer folyamatos felállítása szükséges (WinNT és Solaris). A Windows NT/2000 is fontos szerepet játszik a videós alkalmazások területén. A Windows NT/2000 sokkal inkább hasonlít a Unixra, mint a DOS-alapú Windows-változatok, még a POSIX követelményeinek szempontjából is. Végül az SGI IRIX gépeket is megemlíthetjük, ezek ugyanis sokáig egyeduralkodók voltak a mozgóképek feldolgozásában.

A múlt év elején az SGI bejelentette, hogy elkezdenek komolyan foglalkozni a Linuxszal. A nagy újdonság a grafika, tehát az a terület, melynek az SGI a hírnevét köszönheti. Az nVidiával (grafikai processzorokat gyártó cég) közösen kifejlesztett, Intel-alapú, linuxos grafikai munkaállomások (a 230-as, 330-as és 550-es modellek) az SGI mérései szerint másodpercenként 17 millió háromszög és 540 millió képpont rajzolására képesek. A gépekben található, VPro névre keresztelt grafikai alrendszer 32 és 64 MB memóriával rendelkező, DDR AGP 4x típusú kártyák. Hozzám hasonlóan valószínűleg a legtöbben feleslegesnek érzik egy vadonatúj SGI munkaállomás megvásárlását, hiszen a grafikai fejlesztéseket nyílt forrásúvá tették, és az XFree86-ba is beépítették. A Linux grafikai képességei tehát ismét óriási mértékben fejlődtek.

A videós munka másik szakterülete – itt is rengeteget köszönhetünk az SGI-nek – a fájlrendszerek fejlesztése. Egy DV IEEE-1394 Firewire csatlakozó másodpercenként 30 MB-ot visz át (25-nek reklámozzák, de ebben nincs benne a hangfolyam). Ekkora sebesség mellett az Ext2 két gigabájtos fájlkorlátját röpke kilenc perc alatt elérjük. Egy 22 perces műsor egy ötgigás fájlban fér el. Egy százperces filmhez 23 GB-ot használunk el.

Az XFS egy olyan, naplózást is végző fájlrendszer, mely akár 9000 petabájtos (a tera utáni egység) fájlokat is képes kezelni. Egy ekkora fájlban 83700 évnyi DV-formátumú mozgókép és hang fér el – ennyi talán elég lesz egy darabig... Az XFS-t nemrég írták át SGI IRIX-ről Linuxra, a bétaváltozat 2000 szeptemberében jelent meg. A naplózó fájlrendszerek a lemezkezelést az adatbázis-kezeléshez hasonlóan végzik, és rendszerhiba esetén egy tetszőleges korábbi lemezállapothoz térhetünk vissza. Az fsck-ra tehát nincs szükség. Az XFS nem az egyetlen nagy teljesítményű linuxos fájlrendszer. Létezik még a JFS, a ReiserFS és az Ext3FS is. Ez utóbbi naplózó fájlrendszer, de a 2 GB-os fájlhatárt – elődjéhez hasonlóan – nem tudja túllépni, így videós munkákhoz is használhatatlan.

Az SGI GLX tulajdonképpen egy „ragasztó”, ez a gép szintjén működő OpenGL-t köti össze az X Window-rendszerrel. A GLX Client Library az alkalmazások számára elérhetővé teszi az OpenGL felületet, az X11 hálózati protokoll használatával közvetett leképezést végez, ha pedig a kiszolgáló és az ügyfél ugyanazon a gépen van, akkor közvetlen leképezésre is képes. A GLX Precision Insight-féle megvalósítása a DRI (Direct Rendering Infrastructure), mely a GLX-et, a Mesa 3.1-et, az XFree86 4.0-t és a Linux rendszermag módosításait használja. A GLX-et a Precision Insight vitte át Linuxra, a munkát az SGI és a RedHat anyagilag is támogatta.

Az OpenGL egy felületfüggetlen térleíró nyelv, amit a legtöbb nagy teljesítményű grafikus kártya, de még ezek 10–20 ezer forintos kistestvérei is támogatnak. A Microsoft Direct3D-jét és a 3dfx Glide-

ját felülmúlva az OpenGL a legszélesebb körben használt 3D programozási csomag, hiszen Linuxra, Macintoshra, Windowsra stb. egyaránt kiadták. Az OpenGL lehetővé teszi a 3D-s alkalmazások fejlesztői számára, hogy programjaik bármilyen támogatott grafikus kártyán fussanak anélkül, hogy minden egyes kártyához külön meghajtóprogramot kellene írniuk. Az OpenGL természetesen 2D programozói felületként is használható.
A játéktervező az OpenGL-lel írja le, hogy a grafika mely elemeit kell primitívekből (háromszögek, gúlák, kockák stb.), és melyeket mintázatokkal (képekkel) megjeleníteni. Mivel a munka javát, azaz a térbeli számításokat a gép végzi, ezért ez óriási teljesítménynövekedést tesz lehetővé a képzelt világokat megjelenítő játékokban ➲Quake: http://www.quake3arena.com/. Az SGI OpenGL-változatát a cég saját grafikus meghajtók tervezéséhez használja. A Mesa ➲http://www.mesa3d.org/ az egyik legfontosabb kivétel. A Mesát az eredetileg nem ingyenes OpenGL helyettesítésére találták ki, de az SGI azóta nyílttá tette az OpenGL, a GLX és az XFS forráskódját is.
Az SGI Open Inventor egy objektumközpontú eszköztár, melynek alapját egy 3D adatbázis képezi. Nemrég ez is nyílt forrásúvá vált: az Open Inventor az OpenGL-re és PostScript-nyomtatásra épül.
A további részletekért látogassunk el az SGi nyílt forrású fejlesztésekkel foglalkozó oldalaira ➲ http://www.sgi.com/developers/oss/.
Mivel a linuxos nagy teljesítményű grafikai támogatás még gyerekcipőben jár, a Linuxra írt grafikai alkalmazások még nem érték utol Windows NT vagy SGI IRIX társaikat. Ott van természetesen a GIMP, melyre sokan a Photoshop nyílt forrású megfelelőjeként tekintenek. De vajon hányan tudják azt, hogy a GIMP képes a Broadcast 2000-rel, a legjobb linuxos nemlineáris videovágó programmal együttműködni? (A Broadcast 2000-ről 30–33. oldalon olvashatnak – a szerk.) A Broadcast 2000 tömörítetlen, 720×480-as felbontású, Video4Linux-megfelelő kártyával (például a Hauppauge olcsó WinTV-jével, vagy a Linux Media Labs termékeivel) digitalizált mozgóképekkel is képes dolgozni. A hangfolyam 48 KHz-es sztereó sáv lehet, OSS-megfelelő hangkártya természetesen elkél hozzá. A 2.2-es rendszermagtól kezdve a Linux támogatja a DV IEEE-1394 Firewire protokollt is.
A GIMP mellett rengeteg grafikai program létezik Linuxra is.
A Blender nevű népszerű animátorcsomaggal lenyűgöző térbeli grafikákat készíthetünk. A nem ingyenes Houdini volt az első nagy teljesítményű linuxos 3D animátorcsomag. A Side Effects Software által írt, SGI IRIX-ről áthozott Houdinit rengeteget használják a Digital Domain és más hasonló stúdiókban különleges hatásokhoz és animációkhoz. A Titanic és az X-Men számos trükkjét a Houdini segítségével valósították meg. Több mint ötmillió forintos árával a Houdini sokakat elriaszthat, szerencsére egy 30 napig használható próbaváltozat is elérhető.
Terveink szerint a lap hasábjain a következő néhány hónapban a linuxos multimédia minden témakörét behatóan vizsgáljuk:

- Video4Linux, Hauppauge WinTV kártyával,
- 2D és 3D grafikai és mozgókép-programozás C++, OpenGL és Java felhasználásával,
- MPEG1 és MPEG2-lejátszók és -tömörítők,
- DV-lejátszók és -szerkesztők; IEEE-1394 Firewire,
- AVI fájlok,
- Quicktime fájlok,
- az MP3 és más hangformátumok (például az OGG),
- videofolyamok (Real Video, Quicktime),
- HDTV,
- a Houdini, a Titanic trükkfelelőse,
- animáció, rotoszkópia,
- a linuxos programok összehasonlítása más operációs rendszereken futó testvéreikkel.
- kodekek,

- webes témák (például az Amaya),
- játékgrafika, az SVGA használata
- gépek és alkatrészek (IEEE-1394 OHCI Firewire, WinTV, ATI All-in-Wonder),

Valószínűleg egy Linux-szakértő is képes lenne elemezni e témaköröket, de más szemszögből. Jómagam nem sok linuxos tapasztalattal rendelkezem, szóval Olvasóim nálam is számíthatnak a kezdők által gyakran elkövetett hibákra. De a dolog hatalmas előnye, hogy így együtt tanulhatunk. Az elkövetkező hónapokban programokat is fogunk írni. Egy Maxtor 20 GB-os, 7200-as fordulatszámú merevlemez néhány napja az asztalomon várja, hogy Windows 98-at, Windows 2000-et és Debian Linuxot telepítsek rá. Három rendszert telepítek, hiszen közben nem állhatok le az elkezdett windowsos fejlesztéseimmel sem. Emellett azt a meglehetősen egyszerű tervet is kiötlöttem, hogy az alkalmazásokat írjuk meg mindhárom rendszerre ugyanazon forráskódanyag felhasználásával. Ehhez természetesen mindenképpen trükközésre lesz szükség, ismerve az X és a Win32 grafikus felülete közti nem kevés különbséget.
A PC-be egy WinTV (ezzel nézem és digitalizálom a tévéadásokat) és egy PyroDV IEEE-1394 Firewire kártya kerül majd (ez utóbbira egy digitális videokamera lesz kötve). Ha ezek jól működnek, kipróbálunk egy ATI All-in-Wonder kártyát és talán egy Compaq iPAQ hordozható PC-t is, melyen szintén multimédiás programokat kívánunk fejleszteni. Mi is láttuk a San Diego-i USENIX-en a PocketLinuxot futtató iPAQ-ot, és azonnal beleszerettünk. Akit érdekel a dolog, látogasson el a ➲http://www.handhelds.org/ címre.
A Linux természetesen továbbra is tökéletes hálózati kiszolgáló marad. A BBC-nél weboldalakat, Real Media fájlokat és digitális szöveges szolgáltatásokat bíznak rá. A Victoria's Secrets műsorban a 2000-es cannes-i divatbemutató közvetítését is Linuxszal oldották meg. Ez volt az addigi legnagyobb méretű internetes közvetítés (kétmilliónál is többen nézték világszerte). Az SGI, az nVidia, a RedHat, a Side Effects Software és sok más cég támogatásával a Linux egyre inkább képes lesz átfesteni a róla kialakított képet („az Internet hátaslova") és a multimédiás felhasználásokban is igyekszik kitűnni. Jövő hónapban elindítjuk új rendszerünket és a Linux köré egy komoly multimédiás környezetet kezdünk el kiépíteni, a szükséges alkatrészekkel és programokkal együtt.

Robin Rowe (Robin.Rowe@MovieEditor.com) az internetes és videós alkalmazásokat készítő MovieEditor.com nevű cég egyik vezetője. Igazi nagyágyú: tervezett már kiszolgálóalapú videovágó rendszert a manhattani központú, folyamatosan sugárzó Time Warner New York One nevű hírcsatornának és honlapnak ➲http://www.ny1.com/, valamint egy önműködő tévéhírfigyelő rendszert a DARPA-nak és a Pentagonnak.

Bemutatkozik az MSERV

Drake elmondja, miképpen vethet véget az MSERV a zenei egyeduralomnak az irodákban.

Elgondolkoztál már valaha azon, hogy ki választja ki azt a zenét, amit az irodában és az üzletekben, a liftekben hallgatsz? Lefogadom, hogy volt olyan pillanat, amikor a liftben állva arra gondoltál: „Bárcsak lenyomhatnám azt a kis piros gombot.” Az ilyen gondolatokat a liftzene kelti. Így van ez mindig – hacsak nem egy haladó cégnél dolgozol –, örökké olyan művészek dalait kell hallgatnod, akik vagy régebben halottak, mint amióta te élsz, vagy a pénzüket a „dalok la liftmuzsika” feldolgozásokkal keresik.

Nyilvánvaló, hogy ez nem jó. A vezetőség két dolgot tehet: vagy mindenkit kis dobozokba ültet, ahol aztán barlangi medveként végzik munkájukat az emberek (de legalább a kedvenc muzsikájukat hallgathatják), vagy a vezetőség jelöli ki a zenét.

Az olyan személyes zenegépek megjelenése, mint amilyen a Creative Labs Nomadja (lásd a Kapcsolódó címeket), képes megoldani néhány gondot. Ezeknek a zenegépeknek hosszú játékidejük van, a Nomad például több mint 150 CD-nyi zenét tud tárolni, amire egy sétálómagnó soha sem lenne képes. A lejátszókkal kapcsolatos gondok forrását a következő szó testesíti meg: „személyes”. Nem oszthatod meg másokkal az általános zenei érdeklődésedet, ráadásul töméntelen pénzbe kerül a zenei CD-k beszerzése.

Az irodai hálózatok népszerűsége és az MP3 elburjánzása egy új lejátszótípus megjelenéséhez vezetett. Ezt a lejátszót MSERV-nek nevezik. Az MSERV olyan szabad program, amely lehetővé teszi a hivatal vezetése számára, hogy Linuxot vagy más nyílt forráskódú operációs rendszert futtató PC-t MP3 zenegéppé képes varázsolni. Sokkal inkább olyan, mint egy személyes zenegép, csak éppen központosított lehetőségeket teremt a teljes iroda számára.

Amikor először letöltöttem a MSERV-et, azt hittem, egy műsorszóró kiszolgáló (broadcast server). Nagyszerű ötletnek találtam, több mint 500 CD-nyi zene, egyetlen merevlemezen, amit én vagy bárki más könnyedén elérhet a helyi böngészőjével. Felállíthatnánk a saját zenelistáinkat, és mindannyian külön-külön azt hallgathatnánk, amit szeretnénk. Miután rájöttem, mi is az MSERV valójában, újra kellett gondolnom a dolgokat. Az MSERV zenehallgatás tekintetében igazi közmegegyezést jelentett irodánk dolgozói számára. Az MSERV műszaki meghatározása a következő lehetne: kiszolgáló-rendszerű, TCP/IP-alapú, központi zenesorkezelő és rangsoroló programrendszer. Röviden, olyan, mint a Web Board Poll MP3-hallgatóknak.

A végfelhasználó szempontjából az MSERV maga a valóra vált álom. Arra tervezték, hogy csak olyan zenét játsszon, amit az iroda dolgozói hallani szeretnének. Például, ha van tíz ember, aki kedveli az Eaglest, de csak egyvalaki akad, aki a Yannit, akkor annak az egynek, sajnos nincs szerencséje. Miért? Mert az MSERV egy rangsoroló rendszert használ annak eldöntésére, melyik zenét játssza le. A rangsoroló rendszert pedig teljes mértékben a felhasználók irányítják.

siránkoznál egész nap, egyszerűen csak elindítod a jó öreg böngésződet, és felkeresed a legközelebbi MSERV-kiszolgálót. Voilà, igazad volt, ez valóban olyan, mint amit a mamád szokott hallgatni. A Righteous Brothers az, és azt is látod, hogy a Dirty Dancing is többször van benn, mint ahányszor valaha is meg akartad hallgatni. De ne keseredj el! Az MSERV lehetővé teszi számodra, a végfelhasználó számára, hogy egyetlen kattintással feledésre kárhoztasd a nem kedvelt dalokat.

Természetesen, ha mindenki más kedveli őket, akkor visszahozhatják a rangsor elejére, és aztán óráról órára hallhatók. Ám legalább van némi esélyed arra, hogy befolyásold, mit hallgatsz egész nap.

Az MSERV telepítése nem bonyolult. A hon-

Az MSERV olyan szabad program, amely vagy nyílt forráskódú operációs rendszert

Képzeld el, hogy éppen az e heti feladatoddal birkózol, amikor, valami őrült okból, valami furcsa zaj kezd el szivárogni a hangszórókból. Nem vagy biztos benne, de pont úgy hangzik, mint amit a mamád szokott hallgatni. Azonban ahelyett, hogy morgolódnál és

lapon megtaláljuk a forrást, TAR és RPM-alapú változatként egyaránt. Ha egy RPM-alapú rendszert futtatsz, akkor rendszergazdaként csupán ennyit kell begépelned: `rpm -i mserv-0.33-1.i386.rpm`. Ez minden szükséges

összetevőt telepíteni fog.
Ha a program telepítve van, jelentkezz be egyszerű felhasználóként. A /usr/bin könyvtárba belépve, az MSERV paranccsal indíthatod el az MSERV-et. Az MSERV induláskor létrehoz egy .mserv könyvtárat a nem rendszergazda felhasználó saját könyvtárában.
Az MSERV-vel együtt érkezik egy webes ügyfél is. Az ügyfelet igen egyszerű telepíteni. Mindössze arra van szükség, hogy a Perl legyen telepítve a rendszeren, és az ExecCGI lehetőség be legyen kapcsolva abban a könyvtárban, ahova a rendszert telepíted. Egyszerűen másold a webes ügyfél állományait abba a könyvtárba, amin keresztül el szeretnéd érni. Például, ha a webkiszolgálód gyökérkönyvtára a /usr/local/apache/htdocs, akkor a fájlokat a /usr/local/apache/htdocs/mserv könyvtárba telepítsd. Ha már átszerkesztetted a httpd.conf fájlt az ExecCGI bekapcsolásához, akkor indítsd újra az Apache kiszolgálót.
Most, ha a böngésződet a megadott URL-re irányítod, az MSERV-nek kell feljönnie. Amennyiben az Apache rendben betölti az oldalakat, esetleg megváltoztathatsz néhány értéket a mserv.cgi programban. A telepítés szerinti elérési út könyvtárában találod. A leg-

lehetővé teszi iroda számára, hogy Linuxot futtató PC-t MP3-as zenegéppé varázsoljon.

fontosabb érték, amit érdemes megkeresned, a $host sor. Ez az a sor, amely megmondja az MSERV-nek, merre keresse a programokat.
Ha nem webes ügyfélre vágysz, az MSERV-nek van CLI és MS Windows ügyfélprogramja is. A CLI egy parancssoros kezelőfelületű (Command Line Interface) ügyfélprogram, amely alapjait tekintve a Telnet ügyfélhez hasonló módon működik. Az MS Windows ügyfél Delphiben íródott. Sajnos ezt a Delphi-változatot nem tudtuk kipróbálni, mivel mi nem használunk Windowst.
A Delphi-alapú ügyfél esetleg irodák számára lehet jó választás. Mivel mindannyian tudjuk, a hivatalok többsége még mindig MS Windowst használ, a Delphi ügyfélnek mindenképpen van létjogosultsága. Tekintsd ezt a terjedés lehetőségének. Először csak az MP3 kiszolgáló épül Linuxra, aztán a levelezőrendszer, majd a webkiszolgáló és így tovább. Nemsokára mindenki KDE2-t fog futtatni, és senki sem fogja észrevenni a különbséget.
Ha fejlesztő vagy, az MSERV tökéletesítési lehetőségek egész tárházát ajánlja fel neked. Természetesen ez nem jelenti azt, hogy az MSERV rossz termék lenne, éppen ellenkezőleg. Csakhogy mint minden nyílt forráskódú programnak, ennek is szüksége van némi csiszolásra. Elkelne például egy egyszerű telepítő és beállításegítő program, valami, ami jobb, mint az RPM és az újrafordítás. Szüksége lenne egy szebb, központosított kezelőfelületre, aminek emellett műsorszórási (broadcast) képességei lennének.
Az MSERV lehetővé teszi, hogy új termékek jelenjenek meg a piacon. Íme néhány példa: egy zenegép az autó hátuljában, amely közvetlen összeköttetésben áll a sztereórendszerrel – akárcsak a CD-alapú zenegépek, de tárhelye nem korlátozódik 10 CD-re kazettánként. Az egyetlen korlát a felhasznált merevlemez mérete. Márpedig ötvenezer forintért egy 82 gigabájtos merevlemezt is vásárolhatsz – 82 gigabájt pedig akár 2000 CD anyagának tárolásához is elegendő.
Másik alkalmazási példa lehetne a set-top box. Csak egy kis dobozt kellene csatlakoztatnod az otthoni sztereórendszeredhez, és máris minden CD-d tartalma azonnal elérhetővé válna. Akár úgy is be lehetne állítani, hogy amint új CD kerül a lejátszóba, ellenőrizze a CD-adatokat a CDDB-ből, és ha kell, kezdje el rögzíteni MP3 formátumban a számokat.
Ha ezek közül bármelyikhez még egy távkapcsolót is csatlakoztatsz, máris királynak érezheted magad a sok málészájú között.
Az MSERV jelenleg a 0.33-as változatnál tart, és nem számít üzembiztosnak. Mi, a Command Prompt, Inc. World Headquarters cégnél (a komoly név mindössze egy háromszemélyes vállalkozást takar) már néhány hete használjuk. Az egyhangú vélemény az, hogy az MSERV megbízható, és igen szeretjük. Ha van egy bennszülött gépőrült a háznál, adj egy esélyt az MSERV-nek! Nagyon mókás kis termék, és tulajdonképpen egy általános gondot old meg.
A program igazi ereje az, hogy nyílt forráskódú. Bárki továbbfejlesztheti, bárki bővítheti.

Joshua Drake e-kereskedelemi, illetve Linux-tanácsadó és a saját cégét, a Command Promptot vezeti ➲ http://www.commandprompt.com/. Majdnem kilenc éve használ Linuxot, és a Linux Documentation Project webmestere. Tevékenységi körébe tartozik még a LinuxPorts.com honlap fejlesztése és az OpenDocs kiadó cég irányítása.

Kapcsolódó címek

Nomad a Creative Labstól: ➲ http://www.nomadworld.com/
MSERV: ➲ http://www.mserv.org/

Könnyű álmok (3. rész)

Az Interneten használt fontosabb protokollok.

A korábbi számok bevezetőnek szánt elméleti fejtegetései után ez alkalommal ismét vessük magunkat újabb elméleti fejtegetésekbe. Ahhoz, hogy elég mélyen átlássuk a rendszer támadhatóságának okait, részletesebb adatátviteli és programozás-módszertani ismeretekre lesz szükségünk.
A Linux rendszermag egy olyan erős védelmi alrendszert tartalmaz, amit egyes cégek tűzfal néven el is adnak: a csomagszűrőt. Ez a rendszermag része, és alkalmas mind a rendszer elérhetőségének, mind a rajta áthaladó (ha van ilyen) forgalomnak a szabályozására. Helyes beállításához a hálózat működésének alapos ismeretére van szükség. Jelen cikk célja az, hogy a csomagszűrő beállításához szükséges adatokat megismertesse, és ha már ilyen mélyen beleástuk magunkat a hálózatok rejtelmeibe, akkor tovább megyünk, és a később ismertetésre kerülő hálózati támadások megértéséhez szükséges „okosságot” is szétosztjuk. Azoknak a bátraknak, akik már ma este nekikezdenek a rendszermag csomagszűrőjének beállításához, van egy jó tanácsunk. Biztonsági szempontból jelenleg a 2.2-es rendszermag-sorozat tekinthető megbízhatónak. A cikk megírásának időpontjában már megszületett ugyan a 2.4-es sorozat első változata, mégis sok ellenőrzésre van szükség ahhoz, hogy érdemes legyen telepíteni egy biztonsági rendszerre. A jó tanács: amíg az időszerű sorozat utolsó számjegye (sublevel) nem megy 5 fölé, addig kiszolgálón nem érdemes a rendszermagot használni, mert olyan hiányosságok derülhetnek ki, amelyek a folyamatos üzemelést megzavarhatják. A tanács természetesen fokozottan érvényes a biztonsági rendszerekre. Ezzel kapcsolatban Solar Designer – az ismert és elismert biztonsági varázsló – azt írja, hogy az általa fejlesztett biztonsági kiegészítés 2.4-es rendszerhez használható változata (ez a rendszermag biztonsági lehetőségeit terjeszti ki) nem is várható addig, amíg a rendszermag el nem éri a 2.4.10-es változatot.
A továbbiakban a TCP/IP protokollcsalád alapjaival fogunk megismerkedni. Sajnos, minden részletre kiterjedő ismertetésre nincs lehetőségünk, hiszen a témakör rendkívül széles területet ölel át. Akik mélyebben szeretnének megismerkedni a TCP/IP protokollal, azoknak az [1.] és [2.] szakirodalmat ajánljuk tanulmányozásra. Feltételezzük, hogy az olvasó alapszinten ismeri a számítógépes hálózatokat és az OSI-modellt.

Az Internet anyanyelve

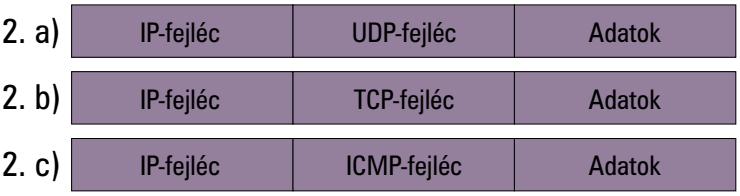
Az Internet adatátviteli rendszere a TCP/IP protokollcsaládra épül. A TCP/IP mind adatcsomag- (datagram) mind virtuális áramkör-szolgáltatást képes nyújtani a rá épülő alkalmazások számára. Az adatcsomag-szolgáltatás kapcsolatmentes, nem megbízható, viszont csekély felesleges adatátvitellel járó kapcsolatot nyújt. Előszeretettel alkalmazzák olyan esetekben, ahol a csomagok esetleges kiesése nem jelent komoly gondot, például üzenetszórással működő zenekiszolgálóknál. A virtuális áramkör-szolgáltatás kapcsolatalapú, nagy megbízhatóságú adatátvitelt tesz lehetővé. Ez teremti meg a lehetőségét annak, hogy a csomagok nem vesznek el, nem kettőződnek meg és sorrendjük sem cserélődik fel.
A protokollcsaládban a virtuális áramkör-szolgáltatást a TCP (Transmission Control Protocol), míg az adatcsomag-szolgáltatást az UDP (User Datagram Protocol) biztosítja. Az Internet legtöbb szolgáltatása TCP-re épül, mert az alkalmazásnak így nem kell kezelnie az adatátvitel közben fellépő hibák nagy részét. TCP-alapú protokoll például a webkiszolgálók és böngészők anyanyelve, a HTTP (Hypertext Transfer Protocol), vagy a leveleink megbízható továbbításáért felelős SMTP (Simple Mail Transfer Protocol).
Az UDP-re kevesebb, de nem kevésbé jelentős protokoll épül, például az Interneten használt nevek feloldásáért felelős DNS (Domain Name Service), vagy az órák összehangolását lehetővé tevő NTP (Network Time Protocol). Mind a TCP, mind az UDP egy közös rétegre, az IP-re (Internet Protocol) épül. Az IP mellett vezérlésre és hibajelzésre szolgál az ICMP (Internet Control Message Protocol).
A TCP/IP protokollcsaládnak még számos tagja van, de most csak a legáltalánosabban használt négy protokollal foglalkozunk.

0 3	4 7	8 11	12 15	16 19	20 23	24 27	28 31
Változat	FH	TOS		Csomaghossz			
Azonosító				Jelzők	Darabeltolás		
TTL		Protokoll		Fejlécellenőrző összeg			
Forráscím							
Célcím							
Kapcsolók						Kitöltés	
Itt kezdődik az adat...							

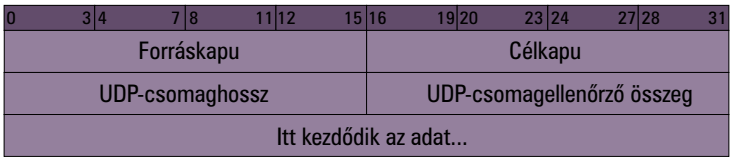
1. ábra Az IP fejlécének felépítése

IP protokoll

Az Internet Protocol feladata, hogy adatátvitelt tegyen lehetővé a hálózatba kapcsolt két számítógép között. Minden hálózatra kapcsolt rendszernek van egy egyedi azonosítója, ez az úgynevezett IP-cím, amelynek egyedinek kell lennie az egész Internetre nézve. Ez alól csak a kizárólag magánhálózatokban használható belső IP-címtartományok kivételek, ezeket viszont az Interneten nem lehet használni. Az IP-csomag egy fejlécből és a hozzá kapcsolt adat-



2. ábra Az Internet leggyakoribb csomagformátumai



3. ábra Az UDP fejléc felépítése

részből áll, ezek együttes hossza nem lehet több, mint 65535 bájt. Az IP-fejléc szerkezetét az *1. ábra* mutatja. Mivel a TCP/IP protokollcsaládot akár határozottan különböző jellemzőkkel bíró hálózatok összekapcsolására tervezték, ezért meg kellett oldani az esetleg nagyméretű IP-csomagok átjuttatását kisebb csomagméret továbbítását lehetővé tevő alhálózatokon. Ha az alhálózatba belépő csomag mérete nagyobb az adott hálózaton megengedettnél, az útválasztó a túlméretes csomagot feldarabolja. Ezt a folyamatot hívjuk darabolásnak (fragmentation). Az egyes darabok ezek után önálló csomagként utaznak a cél felé. A darabokat a célgép állítja össze az *Azonosító* az *MF jelző*, valamint a *Darabeltolás (Fragment offset)* segítségével. Szándékosan hibásan darabolt csomagokkal bizonyos támadások is kivitelezhetők, ha a célgép azok összeszerelésekor hibázik. Erről a későbbiekben lesz még szó.
A *Változat (Version)* az IP-csomag formátumára utal, jelenleg az IPv4 az általánosan elterjedt, de ez elvileg csak 2^{32} számítógép egyedi címzését teszi lehetővé (gyakorlatilag lényegesen kevesebbet). Mivel maholnap a fürdőszobamérlegnek is szüksége lesz IP-címre, így az Interneten új szabvány bevezetését tervezik: az IPv6-ot, ez nagyságrendekkel több rendszer címzését teszi lehetővé.
Ezt követi a *Fejléchossz – FH* mező, amely a fejléc hosszát adja meg 32 bites szavakban. Legkisebb értéke 5 (az ábrát áttanulmányozva jól látszik, hogy 20 bájtnál rövidebb IP-csomag nem létezik), legnagyobb értéke pedig 15, ami legfeljebb 60 bájt hosszú IP-fejléchez vezet. Így a kapcsolók (options) legfeljebb 40 bájtot foglalhatnak el. A *ToS* (Type of Service) mező az adatátvitel jellemzőit befolyásolja. Ennek ismertetésére most nem térünk ki [3.]. A *Csomaghossz* mező a teljes csomag hossza a fejlécet is beleértve, bájtokban megadva. Mivel ez 16 bit, így egy csomag legfeljebb 2^{16}-1, azaz 65535 bájt hosszúságú lehet. Az *Azonosító* mező célja, hogy lehetővé tegye a feldarabolt IP-csomag későbbi összeszerelését. Minden darab az eredeti IP-csomag azonosítóját tartalmazza. A *Jelzők (Flags)* mező két jelzőbitet tartalmaz. A *DF* (Dont Fragment) bit a csomag darabolását tiltja, míg az *MF* (More Fragments) bit mutatja, hogy a csomag darabokból áll, és nem ez az utolsó rész. A *Darabeltolás (Fragment offset)* mező az adott darab címe az eredeti IP-csomagban 8 bájtos osztásokkal. A *TTL* (Time to Live) mező a csomagélettartam korlátozását szolgálja. Értéke másodpercenként, de legalább útválasztón áthaladásonként eggyel csökken. Ha lenullázódik, a csomagot el kell dobni. Célja az, hogy az útválasztók hibájából hurokba kerülő csomag ne keringhessen örökké a hálózaton. A *Protokoll* mező jelzi, hogy milyen magasabb szintű protokollhoz tartozik az adott csomag, például TCP vagy UDP. A *Fejlécellenőrző összeg* a fejlécet hivatott védeni az adatátvitel közbeni hibáktól. A *Forráscím* a feladó, míg a *Célcím* a címzett(ek) azonosítására szolgál. Ezek után következnek az egyes *Kapcsolók*, amelyek a csomag továbbítását befolyásolhatják (ilyenek például a Record Route, Loose Source Routing, Strict Source Routing). Jelenleg ezekre sem térünk ki. A *Kitöltés* célja, hogy a fejléc után következő adatrész 4 bájttal osztható címre kerüljön, hiszen a *Fejléc hossz* mező alapegysége is ez.

UDP protokoll

A User Datagram Protocol célja, hogy adatcsomag-szolgáltatást (datagram service) nyújtson a felsőbb rétegek számára. Az adatcsomag-szolgáltatás kapcsolatmentes, így rövid üzenetek továbbítása során nem kell számolni a kapcsolat felépítéséből és lebontásából származó felesleges hálózati forgalommal és időveszteséggel.
Hátránya azonban, hogy az adatcsomag-szolgáltatás nem megbízható, azaz a csomag elveszhet, többszöröződhet vagy a csomagok sorrendje felcserélődhet. Ezért az adatcsomag-alapú protokollokat használó alkalmazásoknak fel kell készülniük e gondok kiküszöbölésére. Sokszor mégis előszeretettel használnak adatcsomag-szolgáltatást, mivel ez rendelkezik egy hasznos lehetőséggel a kapcsolatalapú protokollokkal szemben, ez pedig az úgynevezett üzenetszórás (broadcast) vagy csoportcímzés (multicast). E lehetőséggel több gép számára küldhetjük el ugyanazt az üzenetet, egyetlen csomag felhasználásával.
Egy UDP-csomag három részből áll *(2. a) ábra)*. Az első rész maga az IP-fejléc, melyet közvetlenül az UDP-fejléc követ. Ezek után jönnek a tényleges adatok. Az UDP-fejléc szerkezete a *3. ábrán* látható. Az UDP-nél az IP-hez képest egy új fogalom jelent meg, ez a kapu. A kapu célja, hogy a megcímzett gépen belül azonosítsa a küldő vagy a fogadó szolgáltatást. A küldőnek a címzett kapu címét ismernie kell. Erre két lehetőség is rendelkezésre áll:

- az alkalmazás szabványos (well-known) címet használ (pl.: DNS, NTP),
- dinamikusan foglal kaput, és a lefoglalt kapu címét egy szabványos címen levő brókerrel közli (pl.: RPC szolgáltatások, ahol a bróker a portmapper vagy rpcbind).

Az UDP fejléc első mezője a *Forráskapu*, ez a feladó kapucímét tartalmazza. A feladónak nem kell megnyitott kapuval rendelkeznie. A következő mező a *Célkapu*, amely a szolgáltatást azonosítja a célgépen belül. Ezt követi az *UDP csomaghossz* mező, mely megadja a csomag hosszát az IP-fejléc nélkül. Értéke 8-tól 65 515-ig terjedhet (hiszen bele kell férnie egy IP-csomagba, ahol az IP-fejléc legalább 20 bájt). Az *UDP-csomagellenőrző összeg* az UDP-fejléc, adatrész, valamint az IP-fejléc legfontosabb részeiből számolódik, és az átvitel közbeni esetlegesen fellépő sérülések észlelése a célja. Az UDP-fejléc után következik az adatrész, ennek hossza legfeljebb 65 507 bájt. UDP-csomagok esetében az IP-fejléc *Protokoll* mezője 17 értékű.
Mivel sem az IP, sem az UDP nem ad biztosítékot a feladó hitelességére vonatkozóan, így ezek a csomagok könnyen hamisíthatók. Ironikusan fogalmazva, UDP-csomag esetében csak egyvalamiben lehetünk biztosak: a címzettben (később látni fogjuk, hogy néha még ebben sem).

TCP protokoll

A Transmission Control Protocol az egész protokollcsalád legbonyolultabb tagja, így ismertetésében csak a legfontosabb tulajdonságaira térhetünk ki [5.]. Mint az előzőekben említettük, a TCP célja, hogy kapcsolatalapú, megbízható szolgáltatást nyújtson a magasabb protokollrétegek felé. A TCP a küldendő bájtokat szakaszokba (segment) csomagolja, amelyeket eljuttat a megcímzett gép megadott kapujára (a kapu fogalma itt is létezik az UDP-hez hasonlóan, de az UDP és TCP kapuknak nincs semmi köze egymáshoz). Adatátvitel előtt a kapcsolatot fel kell építeni, az átvitel végén pedig le kell bontani. A kapcsolattartó gépek TCP protokollrétegei teszik lehetővé, hogy az adatok megfelelő módon átvitelre kerüljenek. A forgalommentes időszakban a TCP képes folyamatosan ellenőrizni a kapcsolat működőképes állapotát (keepalive).

4. ábra A TCP fejléc felépítése

Az adatátvitel biztonságáért a vett adatot a fogadóoldalnak nyugtáznia kell a küldő felé. Ahhoz, hogy az átvitel csatornáját minél jobban kihasználhassa, az egyszerű *megáll és vár* protokoll helyett egyfajta csúszóablakos protokollt használ forgalomszabályozással [1.]. A forgalomszabályozás célja, hogy egy gyors adó ne áraszthassa el a lassabb vevőt. A csúszóablakos protokollok lényege, hogy az adás és a nyugtázás egymással átlapolva zajlik, így a kapcsolattartásra használható csatorna kihasználtsága nagy késleltetésű vonalak esetén is közelít az eszményihez.

A kapcsolat felépítésének két lehetséges módja van. Passzív felépítéskor a futó folyamat létrehoz egy kaput, amelyen képes bejövő kapcsolatokat fogadni. Egy bejövő kapcsolatépítési kérés elfogadásakor a kapcsolat a két végpont között létrejön. Aktív kapcsolatépítéskor a kezdeményező kapcsolatépítési kérést küld a túloldalnak. Ha a túloldal a kérést elfogadja, a kapcsolat felépül.

A TCP szakasz felépítése a *2.b) ábrán* látható. A csomag elején az IP-fejléc található. Ezt követi a TCP-fejléc, ez látható a *4. ábrán*, ezután következnek az adatok. Az első mező a *Forráskapu*, ezt követi a *Célkapu*. Utánuk a *Sorszám* jön, amely a szakasz legelső bájtjának sorszáma, amennyiben a *SYN* bit értéke 0. Minden bájt egyesével sorszámozott. Ha a *SYN* bit értéke 1, akkor a *Sorszám* a *Kezdő sorszámot* (ISN, Initial Sequence Number) jelenti, és a legelső adatbájt sorszáma a *Sorszám+1*. A következő mező a *Nyugtasorszám*. Ha a fejlécben az *ACK* bit 1 értékű, akkor ez a mező azt a *Sorszámot* tartalmazza, amelyet venni szeretne (az ez alatti bájtokat modulo 2^{32} már vette). Az *Adateltolás* (Data Offset) mező az adatok elhelyezkedését mutatja a TCP-fejléc elejétől 4 bájtos egységekben (ez egyébként a TCP-fejléc hossza). Mivel a TCP-fejléc legkisebb hossza 20 bájt, így e mező értéke legalább 5. Az egész TCP-fejléc hosszát szintén ez a mező korlátozza legfeljebb 60 bájtra.

A *Vezérlőbitek* mező a fejléc részeinek érvényességét biztosítja, illetve vezérlési feladatokat lát el. Az *URG* bit a *Sürgős adat mutató* érvényességét jelzi. Az *ACK* bit a *Nyugtasorszám* mező érvényességére utal. A *PSH* bit szerepe, hogy utasítsa a vevőoldalt az eddig fogadott adat eljuttatására a felsőbb rétegek felé. Az *RST* jelző a kapcsolat törlésére szolgál (pl.: a kapu nincs nyitva). A *SYN* jelző a kapcsolatfelépítési, míg a *FIN* a kapcsolatbontási kérelmet jelzi. Természetesen ezen jelzőknek csak bizonyos kombinációja használható, egyebek tiltottak (pl.: *SYN+FIN*). Az *Ablakméret* mező a *Nyugtasorszámtól* kezdve venni szándékozott bájtokat jelöli. Ennek célja, hogy az adó oldal ne küldjön több adatot, mint amennyi pufferrel a vevő rendelkezik e kapcsolat kiszolgálására. A következő mező a *TCP-ellenőrző összeg*, amely az UDP-csomagéhoz hasonlóan számítható. A *Sürgős adat mutató* csak az *URG* bittel együtt értelmes, szerepe az általános adatfolyamon kívüli fontos adat jelzése. Ezt követhetik a *TCP kapcsolók*, majd a *Kitöltés* mező. A *Kitöltés* mező célja megegyezik az IP-fejlécnél látottakkal, hiszen a fejléc méretét itt is 4 bájttal osztható méretben lehet csak megadni.

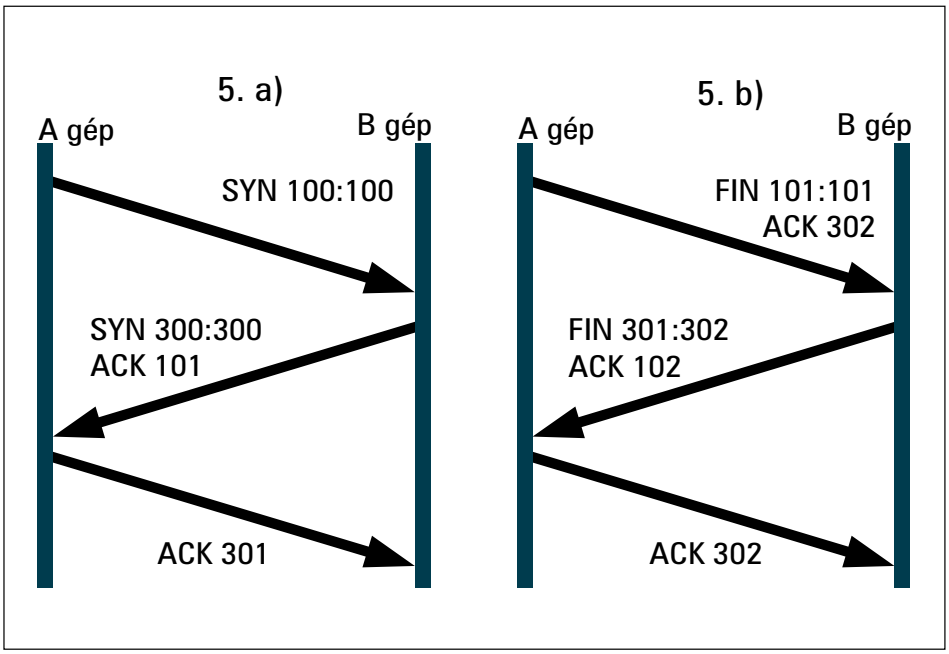
TCP-csomagok esetében az IP-fejléc *Protokoll* mezője 6 értékű.

A TCP a kapcsolat felépítésre háromszintű kézfogást használ, amint az az *5.a) ábrán* látható. A kapcsolatot kezdeményező fél egy *SYN*-es csomagot küld a túloldalnak, amelybe elhelyezi a saját *Kezdő sorszámát* (a kezdő sorszám választása mindkét oldalon roppant fontos, mint azt majd a következő részekben látni fogjuk). Amennyiben a

Az ICMP protokoll típusai

Típus	Leírás
0	echo reply
3	destination unreachable
4	source quench
5	redirect
8	echo request
9	router advertisement
10	router solicitation
11	time exceeded
12	parameter problem
13	timestamp request
14	timestamp reply
15	information request
16	information reply
17	address mask request
18	address mask reply

túloldal elfogadja a kapcsolatot, akkor erre egy *SYN+ACK* csomaggal válaszol, ahol a *Sorszám* a kapcsolatépítési kérést elfogadó gép *Kezdő sorszáma*, a *Nyugtasorszám* mező pedig a kezdeményező *Kezdő sorszám+1* értéket tartalmazza. A harmadik és a kapcsolatépítést befejező lépésként a kezdeményező gép egy *ACK* csomagot küld, amivel visszaigazolja a fogadó gép által választott *Kezdő sorszám+1* értéket. Amennyiben a túloldal nem kívánja a kapcsolatépítést, akkor egy *RST* csomagot küld, amelyben a *Nyugtasorszám* a kezdeményezési kérés *Kezdő sorszámának* eggyel növelt értékét tartalmazza. A TCP-kapcsolatok full-duplexek (az adatok egymástól függetlenül közlekedhetnek mindkét irányba), így a kapcsolat csak akkor záródik le, ha mindkét adatirány lezárult. A kapcsolat lebontásának menete az *5.b) ábrán* látható. A lezárást kezdeményező fél egy *FIN* csomagot küld, amelyet a túloldal visszaigazol, ha minden eddig küldött adatot sikeresen vett. A kapcsolat lezáródását jelzi az alkalmazás felé. Erre a helyesen megírt alkalmazás lezárja a kapcsolatot, amire szintén egy *FIN* csomag lesz a válasz, de most ellenkező irányba. A bontást kezdeményező oldal a vett *FIN* csomagra szintén egy *ACK* csomaggal válaszol. Ennek hatására a kapcsolat bontása lezárul. Mivel a TCP kapcsolatalapú protokoll, és a kapcsolatépítés háromszintű kézfogást használ, így a címek hamisítása jóval behatároltabb és nehezebb. A későbbi részekben még kitérünk a TCP/IP protokollcsalád biztonsági kihatásaira is, ott majd részletesebben is megvizsgáljuk ezt a témakört.



5. ábra TCP kapcsolat felépülése és lebomlása

ICMP protokoll

Az Internet Control Message Protocol lehetővé teszi az IP-csomagok továbbítása folyamán felmerült hibák kezelését, vagy egyéb vezérlőüzenetek továbbítását. Az ICMP-csomagok szerkezetét a *2.c) ábra* mutatja. A *Típus* mező az üzenet tárgykörét határozza meg, a *Kód* mező a *Típus* finomítására szolgál. Az I*CMP-ellenőrző összeg* a teljes ICMP-üzenetből számolt. Az adatrész tartalma a *Típus/Kód* páros értékétől függ. ICMP-csomagok esetében az IP-fejléc *Protokoll* mezője 1 értékű.

A *táblázat* a legfontosabb *Típusokat* határozza meg. A jól ismert ping parancs egy `ECHO REQUEST` ICMP-üzenetet küld a megcímzett gépnek, amely erre egy `ECHO REPLY` típusú üzenettel válaszol. A `DESTINATION UNREACHABLE` típusú üzenetek jelzik, ha a csomag valamely okból nem érhette el célját. A lehetséges kódok finomítják az üzenet jelentését, például a hálózat vagy a számítógép nem érhető el, darabolni kellene a csomagot, de a darabolás tiltott (az IP-csomag *DF* bitje 1 értékű), érvénytelen protokoll stb. A `PARAMETER PROBLEM` típus a csomag hibás szerkezetére utal. A `SOURCE QUENCH` csomagot torlódásvezérlésre szánták, viszont sportszerűtlen viselkedése miatt egyre ritkábban használják. A `TIME EXCEEDED` típus jelzi, hogy a lehetséges csomagélettartam lejárt, vagy a darabolt csomag néhány darabja nem érkezett meg. A `REDIRECT` típus célja, hogy helyesbítse a küldő gép útvonaltábláját. Miután megismertük a hálózatokon közlekedő csomagok felépítését, sorozatunk következő része a hálózati védelem tervezéséről és kivitelezéséről szól majd. Megismerjük a csomagszűrők beállításának ökölszabályait és azokat az eszközöket, amelyek segítségével az esetleges beállítási hibákat meg tudjuk találni.

Hivatkozásjegyzék

[1.] Andrew S. Tannenbaum: Számítógépes hálózatok (ISBN 963-545213-6)
[2.] W Richard Stevens: TCP/IP Illustrated, Volume 1 (ISBN 0-201-63346-9)
[3.] RFC791: Internet Protocol
[4.] RFC768: User Datagram Protocol
[5.] RFC793: Transmission Control Protocol
[6.] RFC792: Internet Control Message Protocol

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

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

Az OpenSSH száz meg egy előnye (1. rész)

Mick az ssh felszínét kapargatja.

Eljött a Paranoid Pingvin mai mesedélutánjának ideje. De ne aggódjanak – ez a mese pont ugyanolyan mazsoláznivalókról szól, mint amilyeneket már megszokhattak a Linuxvilágtól. Sőt, az az igazság, hogy az OpenSSH-ban olyan sok mazsoláznivaló található, hogy ez a cikk a következő számokba is átnyúlik majd!

Ebben a hónapban az ssh hátterével és szerkezetével foglalkozunk. Megnézzük, miképpen kell lefordítani, illetve telepíteni az OpenSSH-t, hogyan használhatunk ssh-t a telnet titkosított kiváltására, hogyan állítsunk be néhány alapvető változót, illetve miképpen használjuk az scp-t titkosított fájlátvitelhez. A következő hónapban az RSA/DSA bejelentkezésről fogok írni, a helyi kapu-átirányításról, a távoli parancsvégrehajtásról, illetve néhány más fejlett és hatékony ssh/OpenSSH képességről.

A rend kedvéért először arról szeretnék beszélni, miképpen került hozzánk ez a program, illetve azokról az emberekről ejtenék pár szót, akik elérhetővé tették számunkra.

Az SSH története

Amióta csak megszületett, a Unixban az volt az egyik legnagyszerűbb dolog, hogy a rendszer-karbantartási feladatokat többféleképpen is el lehetett végezni távoli konzolokról. Sajnos, a legtöbb ilyen módszer (telnet, rsh és az X, csak hogy néhányat említsünk) mindent egyszerű szövegként küld át a hálón, beleértve a jelszavakat is. Ahogy az Internetet egyre többen használták, és ahogy egyre több lelkes önjelölt kalózocska és unatkozó csomagszaglászó kisiparos jelent meg, a szövegalapú hálózati felügyelet túlhaladottá vált.

Néhány évvel ezelőtt azonban egy finn betyárszaki, *Tatu Ylonen* létrehozott egy lélegzetelállítóan jó dolgot, amit Secure Shellnek, azaz ssh-nak neveznek. Az ssh olyan eszközök csoportja, ami nagyjából a Sun rsh, rcp és rlogin parancsainak felel meg, egyetlen igen fontos eltéréssel: az üldözési kényszerrel. Az ssh-val mindent meg lehet valósítani, ami az rsh, rcp és a rlogin segítségével elérhető, kihasználva a választott szabad kódkönyvtárakat és azonosító módszereket. Az ssh egy változata erősen függ az RSA-tól, a kitűnő, de szabadalmak által védett módszertől, ami megköveteli, hogy minden őt használó program engedélyek birtokosa legyen (azaz fizessenek érte), kivéve azt az esetet, ha „nem kereskedelmi” módon használják. Gyakran azonban a nem kereskedelmi felhasználás is nehézkes lehet. Várjunk csak, az RSA US szabadalom 2000 szeptemberében lejárt – gond megoldva, nem? Majdnem: Tatunak is meg kell élnie valamiből, így aztán mire az RSA végre kötetlenné válna, az ssh vált kötötté, mivel Tatu cége, a F-Secure megszigorította az engedélykövetelményeket. Lényegében az ssh 2.0 változatától kezdve az ingyenesség és a szabad felhasználás többé már nem engedélyezett (az RSA-fejleményektől függetlenül). Akárhogy is nézzük, az ssh csak akkor válhat internetszabvánnyá (ahogy azt Tatu is szeretné), ha legalább egy ingyenes megvalósítása létezik.

Ezzel be is léptünk *Theo de Raadt* és az OpenBSD birodalmába. Az OpenBSD, a BSD ingyenes változatának, a NetBSD-nek egy biztonságra sokat adó hajtása. Theo és a nyílt forráskód közösségbeli hittestvéreink az OpenBSD fejlesztőgárdájában szerették volna beil-

1. táblázat Az ssh_config néhány alapvető értéke

Érték	*Lehetséges érték*	*Leírás*
CheckHostIP	*Yes No*	(Alapértelmezett=yes) Figyeljen-e az ügyfélgép IP-címének változására ismert számítógépkulcsok esetén, vagy sem. Figyelmezteti a felhasználót, ha ellentmondást talál.
Chiper	*3des blowfish*	(Alapértelmezett=3des) Melyik blokk-kódolást használja a ssh v.1 kapcsolatokhoz.
Chipers	*3des-cbc, blowfish-cbc, arcfour, cast128-cbc*	Milyen sorrendben próbálja végig blokk-kódolásokat a ssh v.2 kapcsolatokhoz.
Compression	*Yes No*	Használja-e a gzip programot a titkosított adatok tömörítésére vagy sem. Korlátozott sávszélesség esetén hasznos lehet, de valamelyest lassít.
ForwardX11	*Yes No*	(Alapértelmezett=yes) Átirányítja az X kapcsolatokat a titkosított csatornára, és emellett megfelelően beállítja a DISPLAY változót. Nagyon hasznos lehetőség!
Password-Authentication	*Yes No*	(Alapértelmezett=yes) Megpróbáljon-e (titkosított) Unix jelszóazonosítást végezni, az RSA/DSA azonosításon felül, vagy helyett.

leszteni az ssh-t az OpenBSD 2.6-os kiadásába. Csakhogy beleütköztek az ssh különböző megkötéseibe. Amint tudomásukra jutott, hogy egy svéd programozó, *Bjoern Groenvall* az ssh 1.2.12 (Ylonen legutolsó ingyenes – kivéve az RSA részt – változata) egy fejlettebb változatát mutatta be, az OpenBSD-s fickók egy percet sem vártak tovább, azonnal a nyilvánosság elé vitték. Az OpenSSH azóta az OpenBSD része, és mára már jóformán minden Unix-változatra átültették.
A Groenvall munkájára épülő OpenSSH (az ő OSSH nevű változata szintén elérhető) az ssh protokoll korábbi változatait egészíti ki modulrendszerrel és titkosítási eljárásokkal oly módon, hogy az OpenSSH-t egyetlen szabadalmaztatott eljárás vagy hasonló dolog használata nélkül is le lehet fordítani (például az ssh v.1 protokollok nélkül, ezek az RSA-tól függenek). Az OpenBSD csapat másik újítása, hogy az OpenSSH kódot szétválasztotta egy tiszta (clean) változatra, ami olyan egyszerű és felületfüggetlen amennyire csak lehetséges, illetve egy hordozható (portable) változatra, amit többféle Unix-változat alá is le lehet fordítani az OpenBSD mellett.
Ez az utolsó újítás a legérdekesebb a linuxosok számára: a tiszta változat megtartása teszi lehetővé a kód ellenőrizhetőségét, ami így alapvetően állandó és biztonságos. Csak miután Theo (aki egy igazi paranoid) rábólintott erre a kódra, készülhetnek el a hordozható fejlesztések. Így aztán mi egy olyan programcsomagból húzhatunk hasznot, amely nagyon biztonságos, emellett százszázalékosan Linux-megfelelő.
Mellesleg az OSSH felfedezésétől számítva, kevesebb mint két hónap kellett ahhoz, hogy az OpenBSD csapat kiadja az OpenSSH 1.2.2-t, és mindössze hat és fél hónap, hogy a teljes mértékben hordozható, és ssh v.2-megfelelő OpenSSH 2.0 megjelenjen. Ez még akkor is figyelemre méltó teljesítmény, ha tekintetbe vesszük, hogy Ylonen és Groenvall munkájára építettek, különösen, ha beszámítjuk a végeredmény kitűnő minőségét, illetve azt, hogy senki nem fizetett nekik ezért egy petákot sem!
Nos, ennyi lenne az ssh és az OpenSSH története. Remélem, egyetértünk abban, hogy elég lenyűgöző, akárcsak az OpenSSH maga, amely minden valószínűség szerint hamarosan a nyílt forráskódú Unixok elsődleges ssh változatává válik.
Egyébként az „ssh v.1.x” és a „ssh protocoll v.1” az ssh program kiadásnak, illetve a protokollnak felel meg, és nem igazán jelentik ugyanazt. De mivel a csomag és a protokoll változatszámai nagyjából megfeleltethetők egymásnak, itt most az „ssh v.1.x” kifejezést fogom használni, ha az RSA-alapú ssh/OpenSSH változatra gondolok, illetve az „ssh v.2.x” kifejezést, amennyiben a RSA/DSA-t támogató változatokra szeretnék utalni. Ha esetleg nem tudnánk, mi a különbség az RSA és a DSA között, legyen elég annyi, hogy mindkettő ugyanazt teszi, de a DSA-t nem köti semmiféle szabadalom vagy engedély.

Hogyan működik az SSH?

A Secure Shell működése erősen hasonlít a webes SSL-kapcsolatokra (nem véletlen, hogy az OpenSSH által használt titkosító függvényeket az OpenSSL nyújtja, amely a Netscape Secure Socket Layer forráskód könyvtárainak szabadon felhasználható változata). Mindkettő titkosított csatornákat épít fel, amelyek általános gépkulcsokra (host key) épülnek, vagy nyilvánossá tett bizonyítványokat (digitális aláírás – digital certificate) ellenőriznek egy megbízható szolgáltatónál (certificate authority, például a VeriSign). Lássuk, tulajdonképpen hogyan is épül fel ez a kapcsolat.
Először is, az ügyfél és a kiszolgáló (nyilvános) gépkulcsokat cserélnek. Ha az ügyfélgép még soha nem találkozott az adott nyilvános kulccsal, az ssh és a legtöbb böngésző rákérdez, hogy elfogadjuk-e a bizonytalan eredetű kulcsot. Ezután a gépek a kulcsokat használják arra, hogy létrehozzanak egy kapcsolatkulcsot (session key). Minden további adattitkosítás a kapcsolatkulcs alapján történik valamelyik blokk-kódolási módszer szerint, mint amilyen például a Triple-DES (3DES), a blowfish vagy az idea. Ezután a kiszolgáló megpróbálja azonosítani az ügyfélgépet RSA vagy DSA bizonyítványok segítségével. Amennyiben ez nem lehetséges, az ügyféltől hagyományos felhasználónév/jelszó párt vár (lehetőségünk van rhosts-típusú, tehát az ügyfél IP-címe alapján működő azonosításra RSA-kulcsokkal vagy anélkül, de az OpenSSH támogatja a Kerberos IV és az skey rendszereket is). Végül a sikeres azonosítás után kezdetét veheti a tulajdonképpeni kapcsolat: egy távoli héjprogram, biztonságos fájlátvitel, távoli parancsfuttatás stb.

Amint azt korábban már említettem, az ssh tulajdonképpen egy eszközgyűjtemény:

- sshd – kiszolgáló démonként működik minden más programhoz.
- ssh – az elsődleges eszköz: távoli héjprogram, távoli parancsvégrehajtás, illetve kaputovábbítás.
- scp – fájlátvitelhez használható eszköz.
- sftp – interaktív fájlátvitelre használható eszköz. Általában csak kereskedelmi ssh-csomagokban található meg.
- ssh-keygen – titkos, illetve nyílt kulcspárokat készít RSA vagy DSA azonosításhoz (ide értve a gépkulcsokat is).
- ssh-agent – az RSA/DSA azonosítás automatizálására szolgáló démon.
- ssh-add – titkos kulcsot tölt be az ssh-agent folyamatba.
- ssh-askpass – az ssh-add X felülete.

Ezen eszközök közül a legtöbb felhasználót csak az ssh érdekli, hiszen a „titkosított telnet” az ssh legegyszerűbb felhasználása.
Az scp, az ssh-agent, és az ssh-add, valamint az ssh a maga erős azonosítási és TCP-kapuátirányítási képességével együtt rendkívül hatékony csomagot jelent. Mivel paranoidok vagyunk, és minél több hálózaton átküldött anyagot szeretnénk titkosítani, ezzel a rugalmassággal tényleg sokat nyerhetünk.

Az OpenSSH letöltése és telepítése

Ha az OpenSSH legutolsó változatát szeretnénk letölteni akár forráskód, akár RPM formátumban, először is látogassuk meg az OpenSSH honlapját (lásd a Kapcsolódó címeket). Ugyanitt található az OpenSSL, ez elengedhetetlen az OpenSSH-hoz. Szükségünk lehet még a zlib-re, ez a freesoftware.com honlapján (lásd a Kapcsolódó címeket) érhető el.

Megjegyzem, nem biztos, hogy az RPM-ekkel könnyen boldogulunk. Amikor RPM-eket akartam telepíteni a OpenSSH.com-ról a SuSE Linuxot futtató laptopomra, minden kiválóan működött kivéve az sshd-t, ami nem települt fel a SuSE „chkconfig” csomag hiánya miatt. Az egyes Linux-változatok vagy szenvednek ebben a hibában vagy nem (már ha van rajtuk egyáltalán chkconfig). Nos, az is lehet, hogy a SuSE-nek saját RPM csomagja van az OpenSSH-hoz. Amennyiben az RPM nem működik, az OpenSSH-t (és valószínűleg az OpenSSL-t, illetve a zlibet is,) forrásból kell lefordítani. A Linux öreg motorosainak a „készítsd el saját telepítésed” stílus megszokott dolog, de ha nem tartozunk közéjük, akkor se keseredjünk el. Mindhárom csomag egy .configure parancsfájlt használ, ez pedig szükségtelenné teszi a legtöbb felhasználónak, hogy szerkesztenie kelljen bármilyen makefájlt. Ha rendszerünkben megvan a gcc fordító és az általában használatos programkönyvtárak, illetve ezek viszonylag naprakészek, a fordítás általában gyors és zökkenőmentes. Az én esetemben az OpenSSL 0.9.5a és a lib 1.1.3 változat telepítése után a következő lépéseket kellett megtenni az OpenSSH 2.1.1p4 telepítéséhez:

```
tar -xzvf openssh-2.1.1p4.tar.gz
cd openssh-2.1.1p4
./configure --sysconfdir=/etc/ssh
make
make install
```

Az INSTALL fájlban található utasításoknak megfelelően a configure parancsfájlt testre szabott értékekkel egészítettem ki: ahelyett, hogy minden beállítófájlt egyszerűen a /etc könyvtárba helyeznék, arra utasítottam, hogy hozzon létre és használjon egy alkönyvtárat /etc/sshd néven. Mivel az OpenSSH e változata az RSH és a DSA kulcsokat is támogatja, nem rossz gondolat mérsékelni azt a zűrzavart, amit az ssh a /etc könyvtárban kelt. Egy csak ügyfélalapú telepítés esetén mindössze ennyit kell tenni. Megjegyezzük, hogy a fent említett változatszámok már valószínűleg túlhaladottá váltak, mire ezt olvassuk: ne felejtsük el megnézni az OpenSSH honlapját a legfrissebb változatokért.

Ha a sshd Secure Shell démont is futtatni szeretnénk (azaz ssh kapcsolatokat kívánunk fogadni távoli gépekről), létre kell hoznunk indítófájlokat, valamint a SuSE esetében át kell szerkeszteni a /etc/rc.config fájlt.

A Red Hat könyvtár tartalmaz egy sshd.init fájlt, amit a /etc/rc.d könyvtárba lehet másolni, és a megfelelő futásszintű könyvtárból hivatkozni rá (/etc/rc.d/rc2.d stb.). Ugyanitt található az „sshd.pam”, ezt a /etc/pam könyvtárba kell másoljuk, amennyiben Pluggable Authentication Modules (PAM) rendszert és „openssh.spec”-et használunk (ezzel saját OpenSSH RPM csomagokat lehet létrehozni). Ezeket a fájlokat nyilvánvalóan a RedHat alatti használatra tervezték, de elképzelhető, hogy működnek más RedHat-alapú rendszeren is (Mandrake, Yellow Dog stb.).

A SuSE könyvtár szintén tartalmaz egy „openssh.spec” állományt a SuSE-hoz használható prpm OpenSSH csomagok létrehozásához, és egy „rc.sshd” fájlt, amit az /etc/rc.d könyvtárba lehet másolni (jelenleg /sbin/init.d a SuSE-ban). Továbbá tartalmaz egy „rc.config.ssd”-t is, ennek tartalmát a /etc/rc.config fájlhoz kell adni, hogy az rc.sshd parancsfájl helyesen működjön. Ezt egyszerűen meg lehet tenni a következő paranccsal:

```
cat ./rc.config.ssd > /etc/rc.config
```

Készítsünk egy közvetett hivatkozást (`ln -s`) az rc2.d illetve az rc3.d könyvtárakba, és a SuSE már készen is áll a titkosított héjprogramok

2. táblázat Az sshd_config értékbeállításai

Érték	*Lehetséges értékek*	*Leírás*
Port	*number*	(Alapértelmezett=22) A démon által figyelt TCP-kapuszám. A megváltoztathatóság lehetősége különösen akkor válik hasznossá, amikor kapucímátírást (Port Adress Translation) használunk, hogy több gépet rejthessünk el egyetlen IP-cím mögé.
PermitRootLogin	*yes no*	Elfogadjon-e rendszergazdai bejelentkezést, vagy sem. A leghelyesebb ezt az értéket „No”-ra állítani. A rendszergazdák előnyben nem részesített azonosítóval jelentkezhetnek be, majd su-val léphetnek be rendszergazdaként.
PasswordAuthentication	*yes no*	(Alapértelmezett=yes) engedélyezzen (titkosított) felhasználónév/jelszó belépést, vagy tartson ki az RSA/DSA kulcsalapú azonosítás mellett.
PermitEmptyPasswords	*yes no*	(Alapértelmezett=no) engedélyezzen-e üres jelszóval történő bejelentkezéseket a rendszerbe. Figyelmen kívül hagyható, ha a PasswordAuthentication=no. Ugyanígy nem vonatkozik az RSA/DSA kulcsok jelszavaira sem. (Az üres jelszó a kulcsok esetén elfogadható).
X11Forwarding	*yes no*	(Alapértelmezett=no) engedélyezze-e, hogy az ügyfelek X-windows alkalmazásokat futtassanak ssh csatornákon keresztül. Valójában semmit sem nyerhetünk azzal, hogy no-ra állítjuk, mivel az sshd_config nem képes kikapcsolni az általános TCP továbbítást is egyúttal (ami éppúgy használható X11-továbbításra is).

fogadására! A démon életre kel minden rendszerindításkor, vagy kézzel is indíthatjuk a `/etc/rc.d/rc.sshd start` paranccsal.

SSH mindenkinek, avagy mi ez a „titkosított telnet" dolog

Mi a helyzet, ha csupán interaktív héjprogramokat szeretnénk futtatni távoli rendszereken la Telnet? Nagy valószínűséggel még a beállítófájlokba sem kell belenézni, elég, ha ennyit begépelünk:

```
ssh távoli.gép.neve
```

Ezután megadjuk a jelszót (az ssh feltételezi, hogy ugyanazt a felhasználónevet szeretnénk használni a távoli gépen is), és ha nem gépeltük el, már használhatjuk is távoli héjprogramunkat! Ez vitathatatlanul egyszerű, és összehasonlíthatatlanul biztonságosabb, mint a telnet. Ha a távoli rendszeren más névvel szeretnénk dolgozni, mint helyi gépen, a -l kapcsolóval adhatjuk meg a kívánt felhasználónevet. Például ha a laptopomon „mick"-ként vagyok bejelentkezve, és szeretnék bejelentkezni ssh-val a kong-fu.mutantmonkeys.org-ra mint „mbauer", a következő parancsot használom:

```
ssh -l mbauer kong-fu.mutantmonkeys.org
```

Mit csinál ez a parancs? Egyáltalán nem látszik másnak, mint a telnet. Rákérdez, használhatja-e a kiszolgáló nyilvános kulcsát vagy sem, esetleg valamivel tovább tart a kapcsolat létrehozása, illetve a rendszer foglaltságától, a hálózattól stb. függően a kapcsolat némileg lassabb lehet, mint a telnet, de a legtöbb esetben nem sok különbséget lehet észrevenni. Viszont úgy jelentkeztem be a rendszerbe, hogy nem küldtem keresztül a hálózaton egyszerű szövegként a jelszavamat és a felhasználónevemet, és az összes adat szintén titkosítva van! Bármit megtehetek, amit csak akarok, akár egy `su` parancsot is kiadhatok, anélkül, hogy leselkedőktől kellene tartanom. Ráadásul mindez csupán egy kevéske lassulásba került!

Turkáljunk a beállítófájlokban!

Az OpenSSH beállítása szintén nem túl bonyolult dolog. A ssh-ügyfél és kiszolgáló viselkedésének irányításához mindössze két állományt kell módosítani: az ssh_config és sshd_config fájlokat. A csomag telepítésétől függően ezek a fájlok a vagy a /etc alatt, vagy a .configure --sysconfdir értékben megadott könyvtárban találhatók.
Az ssh_config a helyi gépről kezdeményezett ssh kapcsolatok általános beállítóállománya. Ezeket a beállításokat felülbírálhatjuk a parancssori kapcsolók és a felhasználók saját beállítóállományai segítségével (ezek helye a $HOME/.ssh/config). Például ha a /etc/ssh/ssh_config a következő sort tartalmazza:

```
Compression no
```

de a /home/bobo/.ssh/config-ban

```
Compression yes
```

szerepel, akkor valahányszor Bobó ssh-kapcsolatot kezdeményez, a tömörítés alapértelmezés szerint engedélyezett lesz, annak ellenére, hogy minden más felhasználó számára, akinek a $HOME/.ssh/config állománya nem tartalmazza ezt a beállítást, a tömörítés ki lesz kapcsolva. Másrészről viszont, ha Bobó a következő paranccsal indítja az ssh-t:

```
ssh -o Compression=no távoli.gép.neve
```

akkor ebben a kapcsolatban nem alkalmaz tömörítést.

Az ssh_config egy értéklistát tartalmaz, soronként egy értékkel, a következő formátumban: `kapcsoló érték(ek)`
Néhány kapcsoló kétértékű, azaz értéke vagy „yes" vagy „no". Mások viszont értékek vesszővel elválasztott listáját is tartalmazhatják. A legtöbb érték magától értetődő, de mindegyiket részletesen ismerteti az ssh(1) leírása. Az *1. táblázat* bemutat néhányat a legérdekesebbek és legfontosabbak közül (a dőlt betűk lehetséges értékeket jelölnek).
Jó néhány érték használható még ezeken kívül. Néhányat ezek közül a cikk második részében ismertetünk. A teljes lista pedig megtalálható a leírásban.

A Secure Shell Démon, az sshd beállítása és futtatása

Mindez kitűnően működik mindaddig, amíg mások által karbantartott rendszerekhez kapcsolódunk. De még nem beszéltünk arról, miképpen kell beállítanunk saját gépünket, hogy ssh-kapcsolatokat is fogadni tudjunk. Ahogy az már lenni szokott, ez is nagyon egyszerű. Akárcsak az ssh-ügyfél esetében, az sshd alapértelmezett viselkedését is egyetlen fájl, az sshd_config szabályozza, amely szintén vagy a /etc alatt, vagy a ssh beállításakor megadott könyvtárban található. Amint azt már az ssh-ügyfél esetében láttuk, a parancssori kapcsolók felülbírálhatják a beállítófájlban található értékeket. Az ügyféllel ellentétben azonban, a démonnak nincs külön beállítófájlja az egyes felhasználók könyvtáraiban, hiszen az egyszerű felhasználók nem befolyásolhatják a démon viselkedését.
A *2. táblázat* bemutat néhány dolgot azok közül, amelyeket az sshd_config-ban lehet beállítani. Természetesen rengeteg további érték is módosítható itt. Néhányat közülük a következő hónapban bemutatok majd, az eddigiek megértése bőven elegendő az induláshoz (feltételezve, hogy a közvetlen cél a telnet és az ftp kiváltása).

Az scp használata titkosított fájlátvitelhez

Még egy témát tárgyalunk az e havi részben. Az scp parancs, legtöbb működésében azonos a jó öreg rcp eszközzel, amely könyvtárak és fájlok gépek közötti másolására szolgált. (Az igazság az, hogy az scp a rcp forráskódján alapul.) Ha esetleg nem ismerjük egyiket sem, ezek nem interaktív programok: mindegyiket parancssorból kell indítani, amelyben meg kell adni a másolandó anyag és a cél pontos elérési útját.
Emiatt a nem interaktív jelleg miatt az scp kevésbé felhasználóbarát, mint az ftp: akár tetszik, akár nem, az scp használatához bele kell nézzünk a leírásba (vagy egy olyan cikkbe, mint ez), és meg kell jegyezni néhány kapcsolót. Ezzel szemben, mint általában a többi parancssoros eszköz, az scp is sokkal inkább használható parancs-

Kapcsolódó címek

OpenSSH: ➲ http://www.openssh.com

zlib: ➲ ftp://ftp.freesoftware.com/pub/infozip/zlib

SSH információk
➲ http://www.inf.bme.hu/~mulzs/sshfaq/ssh-faq.html
➲ http://www.boran.com/security/ssh_stuff.html

SSH hivatkozások (linkek)
➲ http://www.db.toronto.edu/~djast/ssh.html
➲ http://linuxmafia.com/pub/linux/security/ssh-clients
➲ http://ssh.gatordog.com

SSH FTP tükrök
➲ ftp://ftp.wiretapped.net/pub/security/cryptography/ssh/
➲ ftp://ftp.zedz.net/pub/crypto/ssh/

fájlokban, mint amilyen egy interaktív eszköz lehetne. Az scp „gyorstalpaló” használatát valójában könnyű elsajátítani. Az alapvető formátum a következő:

```
scp [ kapcsolók ] forrásmeghatározás \
       célmeghatározás
```

ahol a forrás és a cél meghatározásai lehetnek egyszerű Unix-típusú fájl- és útvonalnevek (pl.: `./docs/hello.txt`, `/home/me/mydoc.txt` stb.), vagy egy gép és felhasználó azonosítására is alkalmas karaktersorozat:

```
felhasználó@távoli.gép.név:útvonal/fájlnév
```

Például: tegyük fel, hogy a „crueller” gépen dolgozunk, és a „recipe” fájlt szeretnénk átküldeni a „kolach” nevű gépen található saját könyvtárunkba. Továbbá feltételezzük, hogy mindkét gépen azonos felhasználónevet használunk. A folyamat valahogy a következőképpen néz ki:

```
crueller: > scp ./recipe kolach:~
mick@kolach's password: *******
recipe        100%   |****************>|    13226
00:01
crueller: >
```

A parancs begépelése után a jelszót kellett megadnunk, a rendszer feltételezte, hogy azonos nevet használunk a távoli gépen is. Az scp ezután átmásolta a fájlt, miközben egy hasznos kis folyamatjelző sáv mutatja az események előrehaladtát. Ennyi az egész!
Ha a cruelleren „mick”-ként dolgozom, a kolachra azonban mbauer-ként szeretném felmásolni a fájlt, a data/pastries könyvtárba, akkor parancssor a következőképpen alakul:

```
crueller: > scp ./recipe \
       mbauer@kolach:~/data/pastries/
```

Most változtassunk egy kicsit. Tételezzük fel, hogy a /etc/oven.conf állományt szeretnénk letölteni a kolachról:

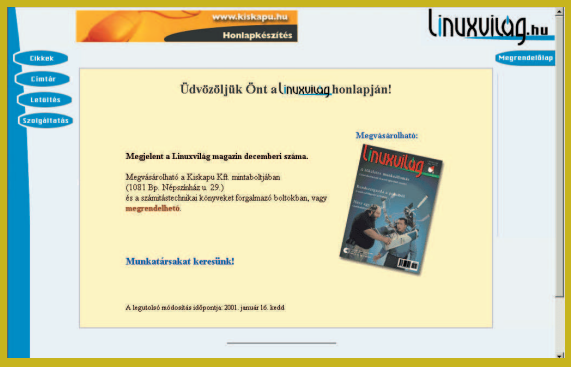
```
crueller: > scp mbauer@kolach:/etc/oven.conf
```

Mindenki érti a lényeget? Az egyetlen fontos dolog, amire emlékezni kell, hogy a forrás megelőzi a célt.
Habár a legtöbb felhasználó az ssh-t és az scp-t mindössze egyszerű bejelentkezésre és fájlátvitelre használja, ez csak a felszíne annak, amire az ssh képes. A következő hónapban azzal foglalkozunk, hogyan lehet az ssh-átvitelt még biztonságosabbá tenni RSA és DSA kulcsok segítségével, miképpen teszik lehetővé a „null-passparse” kulcsok azt, hogy az ssh parancsok parancsfájlokban is használhatók legyenek, hogyan lehet a bizonyítványokat a memóriában tárolni, hogy elkerüljük a nemkívánatos azonosító kéréseket, és azzal, mi módon lehet keresztülvezetni egyéb TCP-szolgáltatásokat egy titkosított ssh kapcsolaton.

Mick Bauer (mick@visi.com) alkalmazott biztonsági vezető az ENRGI hálózatmérnöki és tanácsadó cég minneapolisi részlegénél. 1995 óta Linux-rajongó, és 1997 óta vakbuzgó OpenBSD-s. Különös élvezetét leli abban, hogy ezeket az élvonalbeli operációs rendszereket rávegye arra, hogy elavult roncsokon fussanak. Mick szívesen vesz minden kérdést és hozzászólást.

Szakmai tanácsok

Az ssh-keygen programmal hozz létre egy kulcspárt, ezzel megoldhatod, hogy távoli rendszerre bejelentkezéskor ne kelljen jelszót megadni. A program futtatása után másold a .ssh/identity.pub fájlt a távoli gépre .ssh/authorized_keys néven.
Nem tudsz bejelentkezni ssh-val? Az ssh jelszót kér, amikor szerinted nem kéne ezt tennie? Használd a `-v` kapcsolót, így böngészheted a bőbeszédű tájékoztató- és hibaüzeneteket.
Állítsd be az RSYNC_RSH változót „ssh”-ra, így minden rsync forgalom titkosítva lesz.
Titkosíthatsz minden POP-forgalmat a géped és a levelező-kiszolgáló között, ha a fetchmailt ssh-n keresztül futtatod. Lásd a Linuxvilág honlapján (➲ www.linuxvilag.hu).



Ha a bash parancssorában meg akarsz találni egy korábban begépelt utasítást, nyomd le a CTRL-R billentyűkombinációt, majd írj be a keresett parancsból néhány betűt.
Ha le akarod másolni a webhelyed teljes tartalmát, de nincs parancssori hozzáférésed a kiszolgálóhoz, használd a `wget` programot a `--mirror` kapcsolóval.
Debian rendszereken futtathatod a következő cron feladatot éjszakánként: `apt-get update && apt-get -y --download-only dist-upgrade && apt-get autoclean`.
Ez tárolja a frissített csomagokat, így a telepítésük gyorsabb lesz.
A /usr fájlrendszert csak programok telepítése során kell írni. Mivel az fsck indításkor csak az írható fájlrendszereket ellenőrzi (befűzések száma stb.), ha a /usr-t csak olvashatóra állítod, meggyorsíthatod a rendszerindítást. Emellett, ha a rendszer összeomlik, a /usr-t nem kell majd fsck-zni.
A levelező-kiszolgálód nyílt levéltovábbító (open relay)? Használd a rlytest programot és győződj meg róla, hogy nem az.
Az inetd.conf fájlban tegyél megjegyzésbe mindent, amit nem használsz.
A .Xdefaults fájlban kikapcsolhatod a Netscape-ben a <blink> formázás megjelenítését a következő hozzáadásával:
`blinkingEnabled False`
A .netscape/preferences.js fájlból kikapcsolhatod a Netscape „Shop” gombját:
`user_pref("browser.chrome.disableMyShopping", true)`

A Linux betör az átjárók piacára

A Linux számára újabb hatalmas terület nyílik meg...

Mostanában, amikor az Internet forgalma félévenként megkétszereződik, a hálózatok építéséhez kapcsolódó termékek jelentik a piac egyik leggyorsabban fejlődő részét.
A Linux mindig is kulcsszerepet játszott a kiszolgálók területén, most azonban megnyílt a kapu, melyen keresztül a Linux dárdáját az Internet szívébe, a nagy sebességű átjárókba döfhetjük. Tulajdonképpen a nagygépekből (a Cisco és más vállalatok termékei) épül fel az Internet, ezek mozgatják egyik helyről a másikra az adatokat.
Ma a Cisco az átjárók piacának úgy 80 százalékát mondhatja magáénak. Termékei egy saját fejlesztésű program, az IOS segítségével hajtják végre a rájuk bízott feladatokat. Az átjárók piacának gyors növekedésével azonban egyre több cég jelenik meg, ezek mind a Cisco trónfosztásától remélik saját fellendülésüket. E vállalatok új módszert használnak: nem fektetnek munkát saját kutatásokba és fejlesztésekbe, hanem a piacon található alkatrészekből és programelemekből építik fel rendszereiket.
Az új módszer népszerűségét a hálózati processzornak köszönhetjük. Ez az új termék egy évvel ezelőtt jelent meg a színen, de máris száznál több új átjáróban és hálózati eszközben találhatjuk meg. Használatával egyszerűen helyettesíthetők a Cisco és hasonló gyártók által hosszú évek fáradságos munkájával kifejlesztett különleges processzorok. Az Intel, a Motorola, az IBM, az AMMC (ők nemrég vásárolták föl az MMC Networksöt), a Lucent és a Vitesse azonnal ráharapott az új csodára, de számos kezdő vállalat is ezt építi termékeibe.
Belső felépítésük tekintetében a hálózati processzorok a hagyományos processzorokra hasonlítanak, de a szerkezetük főleg az Interneten áramló adatcsomagok küldésére és fogadására lett kialakítva. A hálózati processzor hatalmas előnye, hogy az átjárókat programozhatóvá teszi, így a hálózati eszközök gyártói sokkal rövidebb idő alatt alakíthatják ki rendszereik új lehetőségeit, még az egyre erősebb szabványosítási törekvések ellenére is.
Az előre gyártott alkatrészekkel kiküszöbölhető az egyéni áramkörök tervezésével és elkészítésével „elfecsérelt” rengeteg idő.
Ez az új módszer elsöpri azokat az akadályokat, amelyek eddig csak a Ciscóhoz hasonló cégek számára tették lehetővé az átjárók fejlesztését. Ebben az új világban az átjárók piaca egyre inkább az otthoni PC-k nyüzsgő piacára hasonlít majd, és az új gyártók egyre merészebben vágnak bele az új eszközök fejlesztésébe. A kialakuló egészséges verseny nagymértékben csökkenti majd a Cisco előnyét, a végfelhasználók számára pedig nagyobb választási lehetőségeket és alacsonyabb árakat eredményez.
Mindennek természetesen van egy megkerülhetetlen feltétele: a szabványos gépekhez szabványos programok szükségesek. Itt játszhat fontos szerepet a Linux. Maga a Linux nem közvetlenül a hálózati processzoron fut, ez ugyanis egy viszonylag korlátozott utasításkészlet felhasználásával megírható kódot futtat, igen nagy sebességgel. De minden átjáró tartalmaz egy vagy több vezérlőprocesszort is, melyek a rendszer teljesítményének figyeléséért és az összetettebb feladatok végrehajtásáért felelősek. A vezérlőprocesszor frissíti például az útvonaltáblát (az Internet örökké változó térképét), illetve a nem szabványos protokollokkal vagy formátumokkal dolgozó csomagok átalakítását.
A vezérlőprocesszorhoz a legtöbb, átjárók gyártásával foglalkozó cég saját fejlesztésű rendszert használ, ilyen például a Cisco IOS-a, de említhetnénk a VxWorkshöz hasonló valós idejű rendszereket is. Néhány cég mégis a Linux mellett döntött, hiszen ez egy jóval nyitottabb, bővíthetőbb környezetet tesz lehetővé. Az MMC például nemrégiben a Monta Vista Software-rel közösen egy Hard Hat Linux-alapú nyílt átjárórendszert dobott piacra, amit teljes egészében az MMC alkatrészeiből, illetve az MMC és a Monta Vista együttes erővel kifejlesztett nyílt forrású programelemeiből építették föl. Ezen alap felhasználásával egy harmadik cég könnyűszerrel kialakíthat egy olyan átjárórendszert, mely akár a Cisco saját termékeivel is fölveheti a versenyt. De még az előre gyártott alkatrészek és programok sem teszik egészen a PC-khez hasonlóvá az átjárók piacát, hiszen a PC-s rendszereket a rájuk írt töméntelen mennyiségű alkalmazás teszi egyedivé, míg az átjárókat gyártó cégeknek saját maguknak kell gondoskodniuk a megfelelő programokról. Az előre gyártott elemek igenis lerövidíthetik a piacra kerülésig eltelt időt, de a rendszerek összeállítását végző vállalatok továbbra sem fognak lemondani arról, hogy egyéni lehetőségekkel bővítsék a terméket, hiszen ezzel a legtöbb esetben előnyhöz juthatnak a versenyben. A legtöbb átalakításon valószínűleg a vezérlőprocesszoron futó operációs rendszer fog átesni. A Linux alkalmazásának megvan az az előnye, hogy eszközkészlete hatalmas, minden igényt kielégítő, emellett pedig a forráskódja is bárki számára hozzáférhető.
A hálózati piac növekedésével egyre több lehetőség nyílik meg a Linux számára.
Az Internet növekedésével a forgalomtöbblet nem a nagyvállalatoktól, hanem az otthoni DSL-, kábelmodemektől és a vezeték nélküli kapcsolatoktól származik.
A hálózati processzorok fontos szerepet tölthetnek be e kapcsolatok összehangolásában. Az előre gyártott alkatrészek gyors fejlődését a nyílt forrású operációs rendszerek is követik. A Linux természetesen e területen sincs vetélytársak nélkül: a valós idejű operációs rendszereket (RTOS) fejlesztő cégek is igyekeznek tartani a színvonalat. A hálózati közösség azonban régóta elkötelezett híve az olcsó és mindennél gyorsabban fejlődő Linuxnak. Egy vezérlőprocesszorra írt rendszer forráskódja akár több millió sorból is állhat, tehát a programfejlesztés a hálózatépítés egyre fontosabb területévé válik. Évi tízmilliárd dollárnál is nagyobb forgalmával az átjárók piaca újabb hatalmas lehetőséget jelenthet a Linux számára.

Linley Gwennap (linleyg@linleygroup.com) a Linley Group alapítója és vezető elemzője, emellett a hálózati processzorokkal foglalkozó A Guide to Network Processors című könyv társszerzője
➲http://www.linleygroup.com/npu/

A MOD formátum (2. rész)

Írásunkban a No Starch Press kiadásában megjelent Linux Music & Sound című könyv egyik fejezetét bővítettük és gyarapítottuk.

Az előző részben, előző számunkban taglaltuk, mi is az a modulszerkesztő, szót ejtettünk a MIDI és a MOD fájlokról.

A WAV és az MP3

Guy Thornley írt egy GMid2Mod nevű programot, mely egy szabványos General MIDI fájlt alakít át .xm formátumba, a Gravis Ultrasound kártya hangszerkészletének felhasználásával. A GMid2Mod segítségével egy négycsatornás, négy GM-hangszert használó MIDI fájlt alakítottam .xm formátumba, majd ezt a MikModdal CD minőségű (16 bit, 44,1 kHz) WAV fájllá, végül e WAV fájlt a BladeEnc segítségével 128 kB/mp minőségű MP3 fájlba.
A fájlméretek összehasonlításából látható, hogy a modulokat készítő zenész jobban teszi, ha az eredeti formátumban terjeszti a kész dalt:

Interlude.mid	34 kB
Interlude.xm	737 kB
Interlude.wav	28 MB
Interlude.mp3	2,5 MB

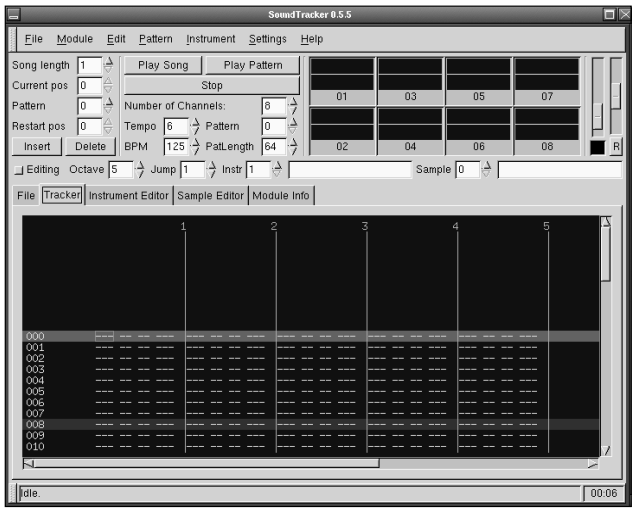
A MIDI és a modulfájlok egyik közös előnye – a WAV és az MP3 formátumokkal szemben –, hogy a dal szerkezetét megvizsgálhatjuk és módosíthatjuk is. Egy .xm modult vagy MIDI fájlt valamelyik szerkesztőbe betöltve könnyedén átírhatjuk vagy kiegészíthetjük a zenét, míg a WAV vagy MP3 formátumú zenéket lehetetlen részekre szedni. Ha a másik irányba szeretnénk haladni, akkor *Kókai István* Xm2Mid nevű programjával .xm modulokat alakíthatunk szabványos MIDI fájlokká. A dolog akkor ad tökéletes minőséget, ha a modul a GM-térképpel azonos, vagy ahhoz nagyon hasonló hangszerkészletet használ.

Linuxos modulszerkesztők

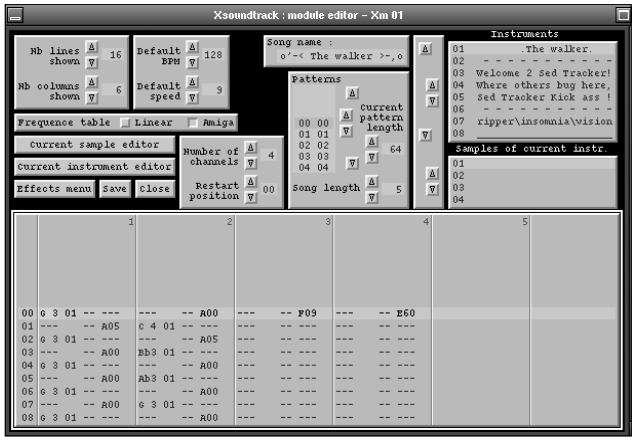
Múlt év augusztusában a Linux Sound & Music Applications honlap még 13 modulszerkesztőről tudott. Hogy melyiket (vagy melyeket) választjuk, azt nagyrészt az elérhető erőforrásaink és a rendszer grafikus képességei határozzák meg.
Az X-felhasználók számára *Michael Krause* gyönyörű, GTK felületű SoundTrackerét (*1. kép*), illetve *Cedric Roux* Xlib grafikájú Xsoundtrack nevű programját (*2. kép*) ajánljuk. A *Jason Nunn* által írt FunktrackerGOLD (*3. kép*) egy tökéletes, konzolalapú szerkesztő, melynek csupán az ncurses könyvtárra van szüksége a működéshez. Ezek a szerkesztők az állandó fejlesztésnek köszönhetően megbízható és jól használható állapotban vannak. A többi szerkesztő (például a SarahTracker, a StupidTracker, a Rapid Audio Tracker és az X felületre tervezett Insotracker) különböző fejlettségi állapotban érhető el.

Modulszerkesztőkre emlékeztető további programok

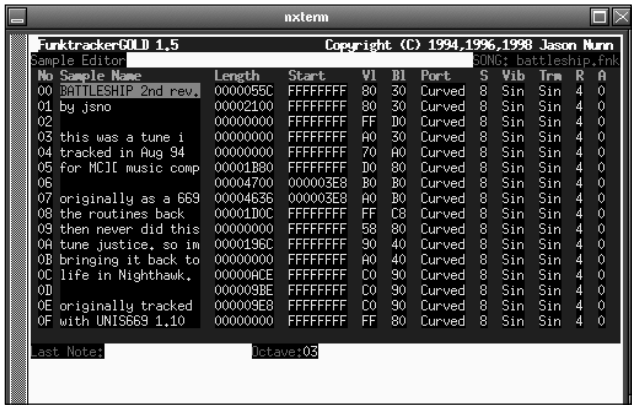
Ezen szerkesztők kezelői felületét és működési elvét olyan zeneprogramokban is megtalálhatjuk, melyekkel nem modulokat szerkeszthetünk. *Juan Linietsky* különleges Shake Tracker nevű programja (*4. kép*) a modulszerkesztők küllemét alkalmazta egy MIDI programban. Ha hangkártyánkon beépített hangkeltő áramkör (szintetizátor) is található, és SoundFont (sf2) támogatása is van, e programmal közvetlenül elérhetjük a kártya hangbankjait. A Shake Tracker fejlesztése nemrégiben kezdődött, de a legfrissebb változat már tökéle-



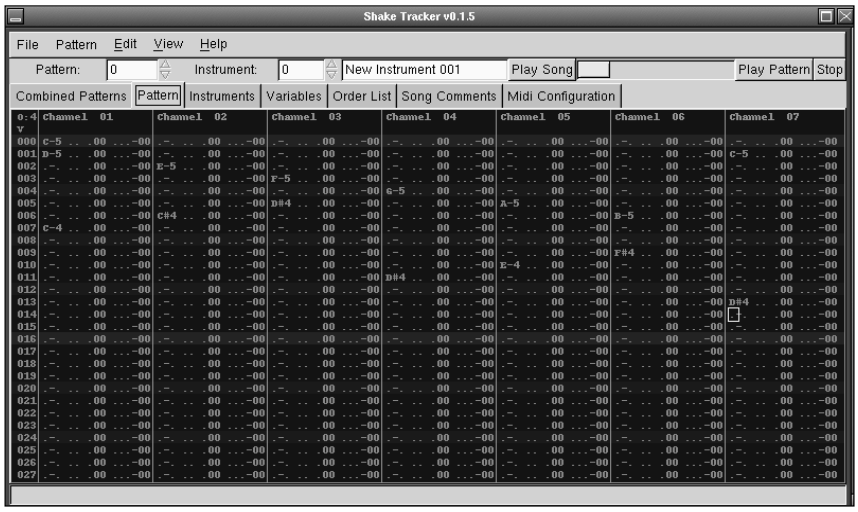
1. kép SoundTracker



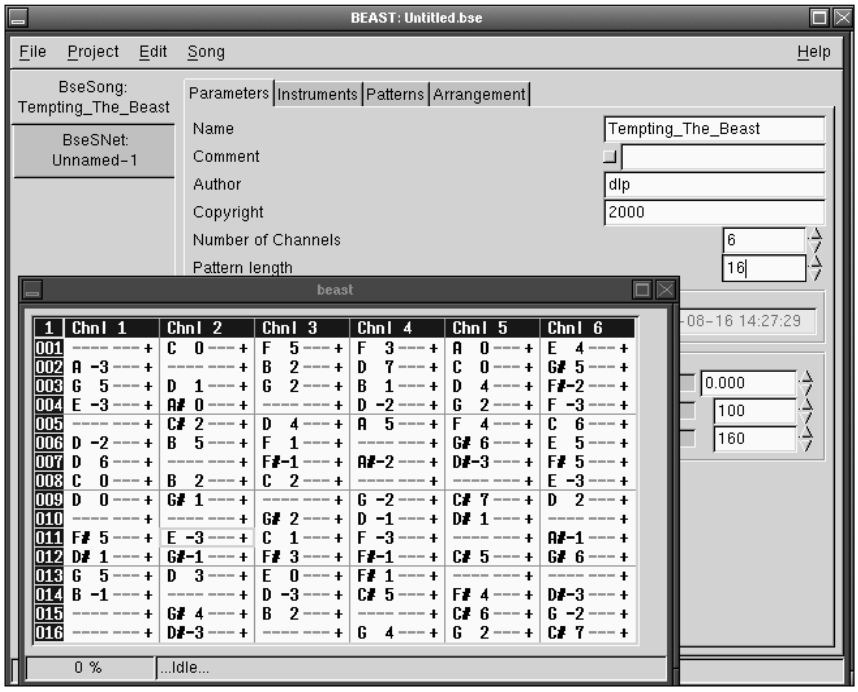
2. kép Xsoundtrack



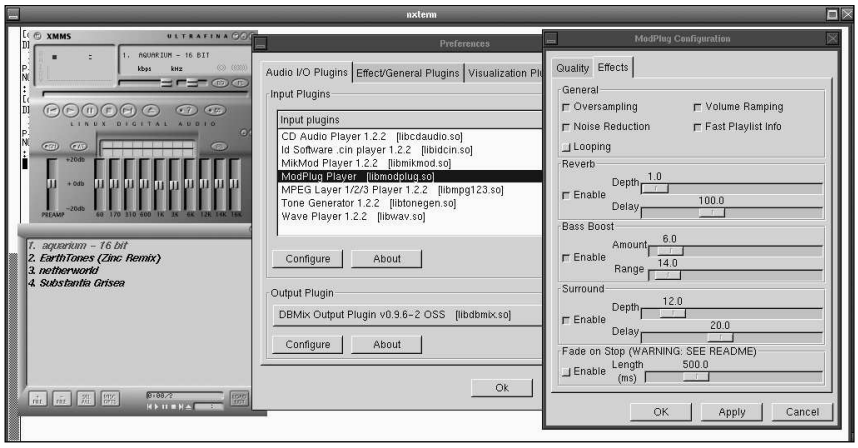
3. kép FuntrackerGOLD



4. kép Shake Tracker



5. kép BEAST/BSE mintaszerkesztője



6. kép XMMS MODPlug beállítása

tesen használható, és a szerző szívesen várja a felhasználók visszajelzését és javaslatait. *Tim Janik* és *Olaf Hoehmann* alakította ki a BEAST/BSE rendszert, mely a hangképzést egy modulszerkesztő kezelői felületével ötvözi (*5. kép*). A fájlok BSE formátumban kerülnek tárolásra, ezeket tehát nem használhatjuk fel modulszerkesztőkben. A Shake Trackerhez hasonlóan ez a program is korai állapotban van, de már működik, és lenyűgöző dolgokra képes. *David O'Toole* GNU-OCTAL nevű programja a windowsos Buzz Tracker linuxos testvére. A Buzz annyiban különbözik a többi modulszerkesztőtől, hogy hangképző szolgáltatással is bővítették, tehát a hangminták mellett valós időben előállított hangokat is felhasználhatunk a modulban. A GNU-OCTAL felépítése hasonló, és figyelmet érdemel, bár a program még a fejlődés korai szakaszában van (Sőt, akár csatlakozhatunk is hozzájuk: ne felejtsük el, hogy a linuxos programok készítői általában szívesen fogadják a segítő kezeket.).

Linuxos modullejátszók

Linuxos lejátszókat konzolos és X-alapú változatban is találunk. Már korábban is említettük, hogy minden Linux-változat tartalmazza a MikMod nevű lejátszót (*7. kép*), ez többféle változatban is elérhető (konzol, GTK, Qt, Xforms, Java). A Windows népszerű MODPlug lejátszójának linuxos változata a nagyszerű XMMS program MODPlug nevű bővítője (*6. kép*), valamint ez a Gmodplay lejátszómotorja is. Az *1. táblázatban* az elérhető linuxos modulszerkesztőket, a *2. táblázatban* a lejátszókat soroltuk fel. Az „Állapot" oszlopban a program használhatóságát láthatjuk. Minél több csillag szerepel itt, annál tökéletesebb és hibátlanabb a program. A „fejlesztés alatt" azt jelenti, hogy az alkalmazás fejlesztése folyamatban van. Ahol kérdőjelet látunk, azt az alkalmazást nem sikerült felépítenem, illetve futtatnom. Ezen és más linuxos modulszerkesztőkről a Linux Sound & Music Applications honlapon (➲ http://sound.condorow.net/mod.html) olvashatunk bővebben.
Néhány Windows és MS-DOS alatt futó szerkesztő és lejátszó bizonyára működik WINE, VMWare vagy DOSemu alatt is, de ezekkel nem kísérleteztem. Ha valaki kipróbálja ezeket, kérem, írja meg nekem a tapasztalatait.

A bemutatóprogramok és a modulok

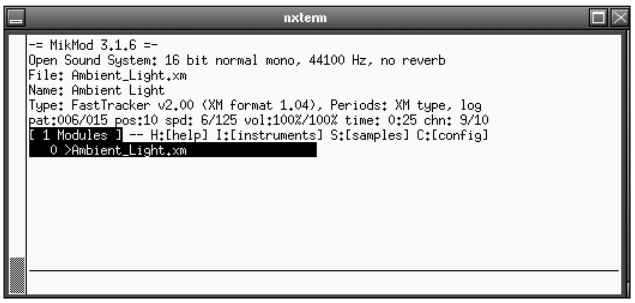
A bemutatóprogramok (demók) olyan programok, melyekben szép grafikákat, animációkat nézhetünk, és ezekhez általában zene is tartozik. A zene legtöbbször modul- vagy MIDI-formátumban található a programban, és egy beépített lejátszó segítségével szólal meg.
A MikModhoz tartozó libmikmod nagyszerű eszköz erre a célra. A kiváló Loop (➲ ftp://mustec.bgsu.edu/pub/linux; a forráskód is itt található) és State Of Mind

***1. táblázat* Linuxos modulszerkesztők**

Név	Grafika	Állapot	Fejlesztik?	Támogatott fájltípusok	Nyílt forráskód?	Szerződés
FunktrackerGOLD	konzol	*****	nem	FNK, MOD	igen	GPL
GNU-OCTAL	n/a	fejlesztés alatt	n/a	igen	GPL	
Industrial Tracker	n/a	fejlesztés alatt	?	IT	n/a	n/a
InsoTracker	X/Gtk	fejlesztés alatt	igen	IT	igen	GPL
Keg Tracker	X/Gtk (<1.2)	n/a	nem	?	igen	GPL
Rapid Audio Tracker	X/Gtk	?	igen	XM	igen	GPL
Sarah Tracker	konzol	fejlesztés alatt	nem	S3M	igen	GPL
Sound Tracker	X/Gtk	*****	igen	XM (a MOD-ot és az XM-et is betölti)	igen	GPL
Stupid Tracker	konzol	***	nem	MOD	igen	GPL
Voodoo Tracker	X/Gtk	***	?	XM	igen	GPL
Xsoundtrack	X/Xlib	*****	nem	XM	igen	ingyenes
maube	X/Gtk	***	nem	?	igen	GPL
ocsatracker	konzol	fejlesztés alatt	nem	S3M	igen	GPL

***2. táblázat* Linuxos modullejátszók**

Név	Grafika	Állapot	Fejlesztik?	Támogatott fájltípusok	Nyílt forráskód?	Szerződés
Gmodplay	konzol	***	nem	669, AMS, DBM, FAR, MTM, OKT, MOD, S3M, XM, WAV	igen	GPL
MikIT	X	***	nem	IT, XM, S3M, MOD	nem	n/a
MikMod	konzol és X	*****	igen	IT, XM, S3M, MTM, 669, STM, ULT, FAR, MED, AMF, DSM, IMF, GDM, STX, OKT, MOD	igen	GPL
Mod4XWin	X/XForms	****	nem	MTM, MOD, STM, 669, S3M	igen	n/a
MODPlug for XMMS	X	*****	igen	669, AMF, AMS, DBM, DSM, FAR, IT, MED, MDL, MOD, MTM, NST, OKT, PTM, S3M, STM, ULT, UMX, WOW, XM, WAV stb.	igen	GPL
S3MOD	konzol	****	nem	S3M, MO0D	igen	a szerző intézi
Xgmod	konzol és X/Qt	****	nem	MOD, 669, XM, MTM, S3M, ULT	igen	a szerző intézi
xmp	konzol és X	*****	igen	XM, MOD, IT, S3M, STM, MTM, MDL, ULT, MED, OKT, 669, FNK stb.	igen	GPL



7. kép MikMod szöveges módban

(➲ http://skal.planet-d.net/min/index.html) nevű bemutatóprogramokat ajánlom mindenkinek megtekinteni, mindkettő a libmikmod használatával játssza le a zenét.

A modulokról bővebben

A United Trackers és a MODPlug Central honlapokon rengeteg lejátszót és szerkesztőt találhatunk, de fellelhető itt sok modulgyűjtemény, hangminta és szerkesztési útmutató is. A MOD Archive a modulgyűjtők aranybányája. Ezek mellett az alt.binaries.sounds.mods és az alt.music.mods hírcsoportokban sok független zenész teszi közzé műveit, javaslatait, kérdéseit és az igencsak mozgalmas modulszíntérrel kapcsolatos értesüléseit, adatait.

SoundTracker

A SoundTracker a legnagyobb teljesítményű modulszerkesztő, amivel Linux alatt találkoztam. A kezelőfelület kialakításának köszönhetően minden egy képernyőről vezérelhető, így a program használatát bárki játszva elsajátíthatja. A SoundTrackerrel a zeneszerkesztés könnyű, és emellett felhőtlen szórakozás.

Beszerzés, telepítés

A SoundTracker forráskódként, valamint RPM és tar csomagokban is elérhető, ezenkívül a korábbi változatokhoz elég csak a frissítéseket telepítenünk. Az alábbiakban a teljes forráskódból telepítés menetét ismertetjük.

A program felépítése előtt tanulmányozzuk a SoundTracker honlapján található Requirements (követelmények) oldalt, ahol a program futtatásához szükséges könyvtárakat és feljlesztőeszközöket találjuk.

A SoundTracker X alatt működik, tehát a szükséges elemek legtöbbje valószínűleg megtalálható az általunk használt Linux-változatban. Ha azonban régebbi változat van a gépünkön, akkor a legfrissebb GNOME és GTK csomagokat is be kell szereznünk a SoundTracker hasznos szolgáltatásainak felépítéséhez és kiaknázásához. E két utóbbi csomag honlapját a ➲ http://www.gnome.org/ és a ➲ http://www.gtk.org/ címen találjuk. A *Michael Pruett* által írt libaudiofile-t is telepítsük (➲ http://www.68k.org/~michaelaudiofile/). A SoundTracker telepítési útmutatójában a felépítés pontos menetét is megtaláljuk, tehát a legfrissebb tudnivalókért mindenképp olvassuk el az INSTALL és README fájlokat.

A SoundTracker és az említett szükséges kiegészítők beszerzése után minden készen áll a program telepítésére. Írjuk be a `./configure -help` parancsot, mire a telepítési folyamathoz használható paraméterek listáját kapjuk. Ezt követően adjuk ki a `./configure` parancsot a megfelelő értékekkel együtt, s ekkor létrejönnek a program fordításához szükséges `Makefile` fájlok. Ha a beállítási folyamat során semmilyen hibával nem találkoztunk, akkor a `make` paranccsal indíthatjuk a fordítást. Ha most sem volt hiba, akkor rendszergazdaként (`su root`) adjuk ki a `make install` parancsot.

A SoundTracker telepítése ezzel befejeződött; a `soundtracker` parancs begépelésével indíthatjuk el a programot. Bővebb ismertetésre jövő számunkban kerítünk sort.

Kapcsolódó címek

MAZ Sound – kitűnő forrás
➲ http://www.maz-sound.com/

Linux Music & Sound Applications – a linuxos zenei alkalmazások legnagyobb gyűjteménye
➲ http://sound.condorow.net/

Modulgyűjtemény:

The MOD Archive – jól szervezett, hatalmas gyűjtemény
➲ http://www.modarchive.com/

Programok:

MODPlug for XMMS – az XMMS modullejátszó bővítménye
➲ http://modplug-xmms.sourceforge.net/

Impulse Tracker
➲ http://www.noisemmusic.org/it/

Scream Tracker
➲ http://www.united-trackers.org/resources/software/screamtracker.html

Octamed
➲ http://www.sonicspot.com/octamed.html

MIDI Manufacturers Association
➲ http://home.earthlink.net/~mma

TiMidity MIDI lejátszó
➲ http://www.i.h.kyoto-u.ac.jp/~shom/timidity

BladeEnc MP3 kódoló
➲ http://datacreek.com/mp3/resources/enc-dec.html

Xsoundtracker
➲ http://linux.davecentral.com/3909_audedit.html

FunktrackerGOLD
➲ http://www.doclib.org/Linux/apps/sound/players/funktrackergold-1..5-2/

Shake Tracker
➲ http:reduz.com.ar/shaketr/

Beast/BSE
➲ http://beast.gtk.org/

GNU-OCTAL
➲ http://www.gnu.org/software/octal/octal.html

GMid2Mod – átalakítóprogram
➲ http://www.voyager.co.nz/~guyat/index.html

Xm2Mid – átalakítóprogram
➲ http://petra.hos.u-szeged.hu/~pilu/

Hírcsoportok:

alt.binaries.sounds.mods

alt.music.mods

Dave Phillips felügyeli a linuxos zeneprogramokkal foglalkozó Linux Music & Sound Applications honlapot, ő maga pedig több mint 30 éve zenél. Zeneprogramokkal 1985 óta dolgozik, Linuxot pedig 1995 óta használ. Részt vett a MIT Press kiadónál 2000-ben megjelent *The Csound Book* című könyv összeállításában, és gyakran ír cikkeket. Legújabb műve a No Starch Press kiadónál megjelent *Linux Music & Sound*, melynek hamarosan magyar nyelvű kiadása is megjelenik a Kiskapu Kft. gondozásában.

Kódellenőrzés

Bármilyen furcsán hangzik is, a legjobb kódellenőrzés az, amelyikre ténylegesen sor kerül.

A nyílt forráskód és a szabad program egyik legtöbbet hangoztatott előnye a kód kölcsönös ellenőrzése. „Több szem többet lát, és minden hibát megtalál” a megszokott refrén. Ez természetesen azt feltételezi, hogy tényleg rendelkezésedre áll számos figyelmes segítő, akik a kódod minden részletét képesek ellenőrizni, és tudják, hogy mit kell keresniük. A kód átvizsgálása olyan, mint a szex, bárki képes rá, de tanulással és gyakorlattal sokkal jobb eredmény érhető el.

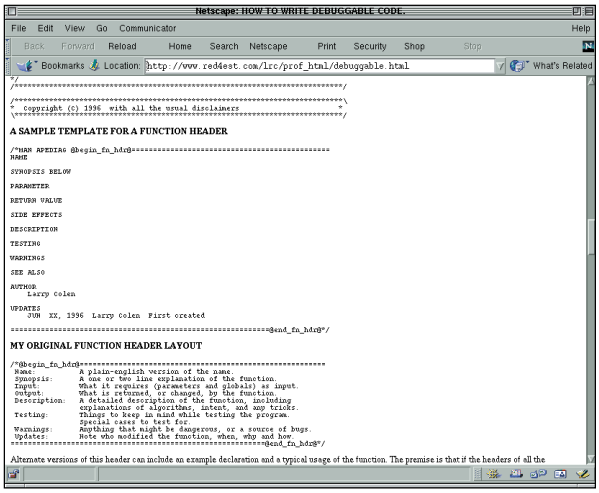
Azoknak a programozóknak, akik még nem ismernék a leckét, felsorolom a kód és a programterv ellenőrzésének előnyeit:

- A program életciklusa során korábban fellelt hibákat könnyebb javítani.
- Ha valaki más vizsgálja át a kódodat vagy a programtervedet, valószínű, hogy olyan hibákat is észre fog venni, amelyek felett te elsiklottál.
- Ha tudod, hogy a kódodat valaki más is meg fogja nézni, sokkal valószínűbb, hogy rendbe teszed és meggyőződsz róla, hogy a leírás naprakész.
- Sokat tanulhatsz mások kódjának olvasása során.
- Ha több ember is ismer egy kódot, biztosítva vagy a „Kamion szindróma” ellen. A Kamion szindróma az, amikor a programot ismerő egyetlen embert elüti egy kamion, vagy éppen nem érhető el, amikor szükség lenne rá.
- A kódellenőrzés segítségével lemérhető a különböző minőségbiztosítási folyamatok hatékonysága.
- Ha jól végzed a kód- és programterv-ellenőrzést, időt takaríthatsz meg és javíthatod a termék minőségét, a fejlesztés összes fázisában.

A programkód ellenőrzésének legfőbb hátránya, hogy időbe kerül és nemcsak az átvizsgálást végző személy idejébe, hanem másokéba is, akiket gyakran szintén határidők szorítanak. Bár tanulmányok sokasága kimutatta, hogy a projektre fordított munkaórák száma összességében kevesebb a jól végzett ellenőrzés esetén, mint anélkül, mindig ott a csábítás: hátha nincs is hiba a kódban és az ellenőrzés csak időpocsékolás. Ez azt jelenti, hogy megvárod, míg elkészül a kód, és csak utána fogsz hozzá a hibák kereséséhez, a gondok feltárásához.

A legjobb ellenőrzés az, amit ténylegesen el is végeznek. Egy gyors és felületes ellenőrzés, ami fényt derít a hibák egyharmadára többet ér, mint a jól átgondolt, tüzetes, mindenre kitérő, amelyre végül nem kerül sor.

Az ellenőrzéseknek két alapvető fajtája van: bemutató (walk through) és formális vizsgálat (formal inspection). A bemutató abból áll,



www.red4est.com/lrc/prof_html/debuggable.html

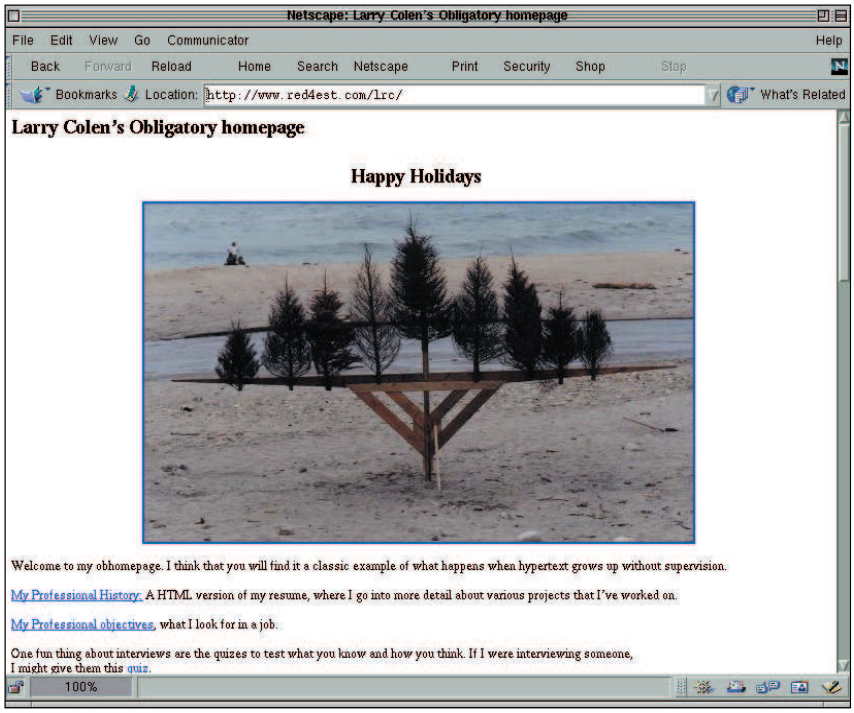
hogy a fejlesztő leül egy vagy több munkatársával és bemutatja nekik az ellenőrzésre váró kódot vagy tervet. Tapasztalataim szerint a bemutató alkalmával a hibák 80 százalékát maga a fejlesztő találja meg, magyarázat közben. *Eli Weber*, egy volt munkatársam, mondta egyszer: „Ha úgy tudnék a falakhoz beszélni, mint az emberekhez, nem lenne szükségem másra a programjaim ellenőrzéséhez.”

A formális vizsgálat esetén kiadják egy vagy több embernek az ellenőrzendő kódot vagy tervet. Gyakran az egyes emberek más dolgokra összpontosítanak a vizsgálat során, például az alkalmazott kódstílus vagy szabványok követésére, logikai hibákra vagy a leírás teljességére. A gyakorlatban ellenőrző listát szoktak készíteni arról, hogy kinek mire kell figyelnie. Ilyen szempontok például a kódolási stílus, a gyakori hibák, a biztonsági rések megtalálása stb.

Nincs abban semmi rossz, ha a kódolási stílust ellenőrző személy észrevesz egy VAGY helyett használt ÉS logikai operátort. Ilyenkor nyugodtan felírhatja azt, de nem kell törekedni arra, hogy mindenki minden szempontból ellenőrizze a kódot. Amikor a vizsgáló talál egy hibát, felírja annak helyét és súlyosságát. A fontossági sorrend változhat a „valami, amit esetleg megváltoztathatsz, ha nagyon nincs mit csinálnod, és a főnöknek rossz kedve van” tárgykörbe sorolandótól a „ha ezt a hibát nem javítod ki azonnal, olyan események láncreakcióját indítja el, ami az általunk ismert civilizáció megsemmisüléséhez vezet” csoportba tartozóig.

Ha mindenki elkészült a rábízott feladattal, az ellenőrök összeülnek megbeszélni a hibákat. Ezeket szinte bármilyen logikai sorrendben lehet tárgyalni: oldalanként, súlyosságuk alapján vagy aszerint, hogy melyik ellenőr találta meg őket. Az értekezletet általában a program szerzője vezeti, de ezt a szerepet más is vállalhatja. Az egybegyűltek minden feltárt hibánál megállapítják annak súlyosságát, vannak ugyanis esetek, amikor az átvizsgáló egymaga nem tudja eldönteni, hogy mennyire súlyos egy hiba, csak észleli, hogy valami nem jó. Ilyenkor megkérdezi a többiek véleményét. Ennél a pontnál nagyon fontos, hogy ne a hiba megoldására törekedjünk, hiszen a feladat most csak a feljegyzésük. Pontosan tudom, hogy milyen nehéz egy szobányi mérnököt lebeszélni arról, hogy ne rögtön a gondok megoldását keressék, de az ülés vezetőjének a megfelelő útra kell terelni a megbeszélés menetét, különben az hetekig eltart és többe kerül, mint amennyit megtakaríthatunk vele.

A kód átvizsgálásának így kell(ene) elméletileg működni céges környezetben vagy oktatási intézmények berkein belül, ahol a cégen belül az emberek együtt dolgoznak. De mi történik a nyílt forráskódos rendszerben fejlesztőkkel, akik gyakran nemhogy más városban, de esetleg más országban élnek és dolgoznak? Létezik néhány szűkebb körű projekt, amely levelezési listát használ kódellenőrzés céljára. Ilyen esetben a szerző bejelenti a változást, a lista tagjai pedig kedvük szerint ellenőrzik a kódot. Ha véleményük szerint valami nincs rendben („Te hitetlen, három oszloppal húztad be a kódot, amikor mindenki szerint csak kettővel kellett volna!”), írhatnak a szerzőnek, vagy a listára, és higgadtan és józanul megvitathatják a gondokat. Más projektek más modellt alkalmaznak: valakinek szüksége van egy programra, valamilyen célból. Keresi, de nem találja, így arra az elhatározásra jut,



www.red4est.com/lrc

hogy megírja. Amikor készen van, és annyira működik, amennyire a szerzőnek szüksége van rá, elérhetővé teszi a nagyközönség számára, például közzéteszi a freshmeaten vagy egy másik, hasonló jellegű fórumon. Valamikor később, valakinek kell egy program ugyanarra vagy nagyon hasonló célra. Megtalálja azt, amit te írtál, letölti, de nem tudja futtatni. Megkeresi azt a helyet, ahol a program elszáll, kijavítja a rosszalkodó kódrészletet és a szíve jósága által vezérelve, valamint a megszállott programozótársak iránti együttérzésből elküldi a javítást neked, azaz az eredeti szerzőnek.

Felhívnám a figyelmedet, hogy a fent vázolt két eset egyikében sincs biztosíték arra, hogy a kódodat ellenőrizni fogják, ha mégis, akkor nem biztos, hogy ez elég hamar történik meg ahhoz, hogy hasznodra váljék. Hogyan lehet a lelkiismeretes programozó biztos abban, hogy a kódját ellenőrzik, méghozzá időben, ahhoz, hogy haszna is legyen? A legtöbb megszállottnak megszállott barátai vannak. Az írók gyakran alkotnak irodalmi köröket, hetente egyszer találkoznak, és egymás írásait elemzik. Ugyanezt meg lehet tenni programok esetén is. A résztvevők megkaphatják a kódot vagy programtervet valamivel a találkozó előtt. A formális vizsgálat módszerét felhasználva megbeszélhetik a találkozón a felderített hibákat.

Egy másik megoldás: amikor a kód vagy programterv készen áll az átvizsgálásra, a szerző elküld egy levelet a megfelelő listára és meghívja programozótársait egy kellemes étterembe, ahol rendkívül finom fokhagymás pizza, és némi sör mellett kellemes hangulatban megejtik a programbemutatót.

Ha senki sincs helyben, aki segíteni tudna neked, ne keseredj el, a formális vizsgálat legnagyobb részét levélben is meg lehet oldani. A lényeg az, hogy kialakuljon egy csoport, melynek tagjai egymás kódját komolyan ellenőrzik, nem csak átfutnak a forráson. Hasonlóképpen a legtöbb hibát a programbemutatók során a szerző maga találja meg, a kód működésének ismertetése közben. Ha nincs a környéken senki, akit rá bírnál venni arra, hogy üljön le és figyeljen, amíg elmagyarázod a programod működését, esetleg a macskád sem bír *Richard Stallman* és *Dennis Ritchie* programozási tudásával, használhatod az Internetet, mint hallgatóságot: írj részletes leírást. Az, hogy leülsz a programhoz dokumentációt írni, rákényszerít arra, hogy kívülállóként nézzél rá és megtalálj olyan idegesítő hibákat, mint a rossz zárójelezés.

A kód átvizsgálása sokban növelheti a termék biztonságát. Először is, ha többen átnézik a kódot, kisebb az esélye annak, hogy hátsó kapukat találjanak a programban. Ilyen lehet például: „ha valaki f00Bidoo néven jelentkezik be, automatikusan rendszergazdai jogokat kap”. Ennek ellenére, a legtöbb támadás, főleg a nyílt forráskódú programoknál, ismert típushibák keresésével kezdődik, ezeket használják ki – mintegy rosszindulatú kódellenőrzést végezve. Ezen biztonsági rések lezárásához szükséges, hogy előbb találjuk meg a hibákat, mint az, aki kevésbé barátságos szándékkal keresi azokat.

A hibás kódrészletek felderítése – ahogyan ezt matematikus barátaink mondanák – szükséges, de nem elégséges feltétele a biztonságnak. Az OpenBSD biztonsága növelése értelmében kíméletlenül alkalmazni kell az ismertetett kódellenőrzési módszereket. Az OpenBSD Biztonsági csoportja számos kutatást végez ebben a témakörben. „Igazából erre összpontosítunk” – mondta *Theo de Raadt*. – „Megpróbálunk újfajta típushibákat felfedezni a régi kódban. A létező használatban lévő kóddal foglalkozunk, amelyben egyszerű hibák is lehetnek, amelyekre soha senki sem gondol. Ilyen például az fd_set túlcsordulása. Az OpenBSD-ből lényegében három éve kiirtottuk a puffertúlcsordulásokat.” Ha többet akarsz megtudni az OpenBSD-ről, a biztonsággal kapcsolatos elméleteikről és céljaikról, nézd meg a ➲ http://www.openbsd.org honlapot.

Összefoglalásul, a kódellenőrzés folyamán megtalált hibák többségét azért találják meg, mert valaki ránéz a kódra, ez lehet a kód ellenőre vagy a szerző egy bemutató során. Ugyanebben a szellemben állíthatjuk, hogy jobb, ha szerény vizsgálati módszert dolgozunk ki, mint ha elveszünk a részletekben, és a nagyra törő terveken alapuló átvizsgálásra időhiány miatt sohasem kerül sor. Az, akit foglalkoztat a biztonság, rengeteget nyerhet a kód gondos ellenőrzésével, de ez nem elegendő. Annak a szemével kell nézni a programot, aki ki akarja játszani a rendszer biztonságát. Nem csak azokra az esetekre kell figyelmet fordítani, amikor a program nem úgy működik, ahogyan azt elvárnánk. Azt is vizsgálni kell, hogy hogyan lehet a programot rávenni arra, hogy váratlan dolgokat műveljen.

Köszönetet szeretnék mondani *Theo de Raadtnak* a felbecsülhetetlen segítségéért, az OpenBSD-vel és a kódellenőrzés biztonsági szerepével kapcsolatban.

További érdekességek

Több hírt találsz róla, mint szeretnél a honlapján
➲ http://www.red4est.com/lrc.
Ugyanitt olvashatsz egy hosszadalmas és céltalan írást a programok minőségéről:
➲ http://www.red4est.com/lrc/prof_html/debuggable.html

Larry Colen
(lrc@red4est.com)
1994 márciusa óta foglalkozik Linuxszal és 1998 novembere óta hivatásául választotta.

Merre tovább, melyik úton?

Nézzünk körbe a webes világban: milyen módszerek és eszközök vannak, melyek a legígéretesebbek?

Amerikai testvérlapunk elérkezett a webes témákkal foglalkozó rovatának ötvenedik cikkéhez. Ez alkalommal nem egy témakör boncolgatására, hanem az internetes fejlődési irányvonalak átfogó elemzésére vállalkoztak, amit merész jövendöléssel is kiegészítettek.
A Kovácsműhely (At the forge) 1996 eleje óta jelenik meg rendszeresen a *Linux Journal*-ban. Az utóbbi néhány évben ezeken az oldalakon számos webbel kapcsolatos módszert, eljárást és alkalmazást ismerhettek meg az olvasók, ezek között minden megtalálható az egyszerű CGI programoktól kezdve a mod_perlt használó adatbázis-kezelő alkalmazásokig. Következzék most néhány gondolat a webes alkalmazások jövőjéről.
Egyrészt a dolgok soha nem álltak ilyen jól a webes alkalmazások fejlesztői számára: az eszközök még mindig igen gyors ütemben fejlődnek, és az összetett alkalmazások készítése egyre egyszerűbbé válik. Másrészt viszont a beágyazott programnyelvek, alkalmazáskiszolgálók és adatbázis-csatolók területe igencsak zsúfolt, így nehéz eldönteni, hogy nekünk pontosan mire van szükségünk.
Írásomban megkísérlem a webes módszerek és alkalmazások jövőjét megjósolni, ezért a szövegből következtetni lehet arra is, hogy milyen témaköröket érintünk a Kovácsműhely rovat későbbi számaiban. Az e havi cikkben ismertetett irányvonalakat követi saját tanácsadó cégem is, tehát az olvasó kitalálhatja, hogy az elkövetkező évben (vagy akár később) milyen javaslatokat teszek majd. Mivel mind a *Linux Journal*, mind a cégem elsősorban a linuxos kiszolgálókkal foglalkozik, ezért a témakörön belül főként a Linuxszal és az ingyenes programokkal kapcsolatos részterületekre összpontosítok.

Egy kis visszatekintés

A webes alkalmazások fejlesztésének története a Web hőskorában kezdődött. Azóta, hogy az első dinamikus oldalt letöltötték (ez még jóval a CGI megjelenése előtt történt, az Internet Explorer, a Netscape és az Apache pedig még csak gondolatban létezett), a programozók egyre nagyobb számban készítenek összetett alkalmazásokat az Internetre.
A CGI, vagyis a „Common Gateway Interface” megérkezésére sem kellett sokáig várni. A CGI eredetileg azért készült, hogy a nem webes alkalmazásokat webes felhasználói felülettel lehessen ellátni. A CGI-nek hála, létrejöhettek az első hordozható kiszolgálóoldali programok. A legtöbb webes alkalmazást még ma is a CGI segítségével írják, és ezt főleg egyszerűségének és teljesen felületfüggetlen kialakításának köszönheti. Az sem elhanyagolható tény, hogy a tárhelyszolgáltatók minden további nélkül engedélyezhetik felhasználóik számára a CGI programok futtatását, hiszen azok (kellő körültekintés mellett) nem veszélyeztetik a kiszolgáló biztonságát. CGI programot bármilyen kiszolgálóhoz és operációs rendszerhez készíthetünk, tetszőleges nyelven; majdnem biztos, hogy működni is fog. Azonban a CGI-nek is akad néhány hátrányos tulajdonsága. Például a kiszolgálónak minden olyan kérelemhez új folyamatot kell indítania, amely valamilyen CGI-programra irányul. Más szóval: egy percenként 60 új látogatót fogadó honlap kiszolgálója minden másodpercben új folyamatot indít.
Ez még önmagában nem is olyan aggasztó, hiszen egy egyszerű linuxos gép képes akár tizedmásodpercenként új folyamatot indítani, ugyebár? Az egyes folyamatok mérete és indulási sebessége azonban már fontosabb szerepet játszik.
A Perl, mely évek óta a kedvenc programnyelvem, bebizonyította, hogy segítségével hatékony CGI-programokat írhatunk. A CGI.pm modul rengeteg szolgáltatást bocsát rendelkezésünkre, ezek segítségével a CGI-vel kapcsolatos álmainkat tökéletesen megvalósíthatjuk (sőt, olyanokat is, melyeket valószínűleg soha nem használnánk ki). A Perl emellett a mintakeresésre is igen jól alkalmazható, valamint a népszerű internetes protokollokat és szabványokat támogató modulokat is megtaláljuk benne. A DBI (adatbázis-csatoló) modult is rengetegen használják. Segítségével a dinamikusan létrejövő oldalakba SQL-lekérdezések kimenetét igényeink szerint illeszthetjük be.
Bármennyire is tömör, rugalmas és biztonságos a Perl, a CGI-szabvány eredeti rendeltetésénél fogva alkalmatlan nagy mennyiségű dinamikus oldal egyidejű létrehozására. Minden Perlben írt CGI-program futtatásához új folyamatot kell indítani, a Perlt, majd a programot be kell tölteni a memóriába, a programot le kell fordítani a Perl belső kódjára, és végül a Perl futtatómotorjával értelmezni kell. Így nem csoda, hogy elég néhány tucat egyszerre futó CGI-program, és egy átlagos kiszolgáló máris megadja magát.
Ennek ellenére a CGI sikeres, mert egyszerű. Semmilyen más API segítségével nem írhatunk meg egy „Szia világ!” típusú programot ilyen könnyedén:

```
#!/usr/bin/perl -wT
use strict;
use CGI;

my $query = new CGI;
print $query->header("text/html");
print $query->start_html;
print "<P>Szia, világ!</P>\n";
print $query->end_html;*
```

Beágyazott nyelvek

Az utóbbi pár évben a webes programfejlesztés önálló szakterületté nőtte ki magát: a programozónak a rendszer, a hálózat és az adatbázis üzemeltetésével is tisztában kell lennie, emellett a megfelelő programozói módszereket és biztonsági szempontokat is fejben kell tartania.

A webes fejlesztéseknek jelenleg három olyan iránya van, melyek alapjaiban változtathatják meg a felhasználók és a fejlesztők helyzetét. A három irányvonal együttes használatát „alkalmazáskiszolgálónak” nevezzük.
Az első irányvonal inkább a régi elvek módosítását jelenti: a programozók egyre inkább eltávolodnak az egyszerű CGI-programoktól – az alkalmazásokat a kiszolgálóba vagy más környezetbe építik. És miért használjuk a CGI-programokat dinamikus oldalak készítésére? Mert a webkiszolgáló önállóan nem képes egyéni HTML-oldalak előállítására. Elméletileg lehetséges volna, ehhez azonban egy új Apache modult kellene írnunk C nyelven, majd azt a rendszerbe fordítani – ez azonban a legtöbb esetben óriási feladat lenne, és az időtakarékosság itt elsődleges szempont.
Szerencsére, létezik egy közbülső megoldás a két változat (csak kiszolgálóoldali, illetve csak külső programok használata) között. Mi lenne, ha a kiszolgálót egy egész programnyelvvel bővítenénk, és az általunk igényelt szolgáltatásokat ennek segítségével valósítanánk meg? Ez a módszer az utólagos bővítéseket és javításokat is nagymértékben leegyszerűsíti, és így a kiszolgáló újrafordítását és újraindítását is megúszhatjuk.
Ez a lényege a mod_perl használatának, mely az Apache kiszolgálót a Perl nyelvvel bővíti. Segítségével Perl nyelven elérhetjük az Apache legbelső szerkezetét és értékeit, így a lekérdezés tárgyául szolgáló oldallal, fájllal bármit megtehetünk. A mod_perl bármire képes az Apache-ban, amit máskülönben egy C nyelvű modullal hajtanánk végre, az egyéni válaszfejlécektől kezdve az azonosítás módjának megváltoztatásáig.
A CGI-programokkal ellentétben, ahol a Perl a programot egyszer fordítja le, egyszer futtatja, majd kilép, a mod_perl a program lefordított változatát a gyorstárba helyezi, majd annyiszor futtatja, ahányszor ez szükséges. Ne felejtsük el, hogy ez a módszer néha igencsak megnöveli a rendszer memóriaigényét, a programozóknak pedig rendkívül körültekintően kell eljárniuk.
Régebben a mod_perl volt az egyetlen olyan modul, mely egy beágyazott programnyelvvel bővítette az Apache-t, manapság azonban egyre több ilyen elem kerül rivaldafénybe. Az egyik ilyen a mod_snake, mely ugyanazt végzi a Python programok esetében, mint a mod_perl a Perlnél, azaz segítségével egyéni Apache-kezelőket írhatunk Pythonban. Még egy mod_tcl is létezik, mellyel az Apache-t beágyazott Tcl-lel bővíti, de jómagam még nem hallottam olyan honlapról, ahol ezt használnák.
Egy másik nyílt forráskódú webkiszolgálónak, az AOLServernek már régóta van beépített Tcl-támogatása. Így a dinamikus tartalmat Tcl-eljárásokkal is elkészíthetjük anélkül, hogy ehhez CGI-programokhoz kellene nyúlnunk. Aki a Tcl helyett a Pythont szeretné megismerni, annak figyelmébe ajánlom a PyWX (Python Web Extensions, Python webes kiegészítő) nemrég megjelent bétaváltozatát. A PyWX az AOLServer által támogatott Tcl szolgáltatások mindegyikét megvalósítja Python nyelven. Ezzel együtt jár az is, hogy a PyWX az AOLServer számára elérhető Tcl-kódokkal nem képes együttműködni, ennek ellenére számos lehetőség kihasználását nagymértékben megkönnyíti – elég csak a Webről letölthető rengeteg Python-modulra gondolnunk.

A kód és a HTML keverése

A második fő irányvonal, hogy a dinamikus kódot a HTML forráskódba ágyazzuk be. A Microsoft Active Server Pages (ASP) nevű rendszere talán a legjobb példa erre, de ezek mellett jó néhány más hasonló jellegű rendszert használhatunk. Linux alatt például a Java Server Pages (JSP), a HTML::Mason (ez a mod_perllel működik), a PHP és az ADP közül választhatunk.
Javával a rendszer első megjelenése óta foglalkozom, és régóta tudom, hogy ez a nyelv valóban megér egy kis odafigyelést. Mint sok más szakembert, engem is az appletek megjelenése taszított el tőle, ezek ugyanis lassúak, nem biztonságosak és hibásak voltak.
Az utóbbi években azonban a kiszolgálóoldali Java igencsak megerősödött. Minden Java „servlet” (kiszolgálóoldali programka) egy osztály, mely egy Java Virtuális Gépen (JVM) belül fut.
A servletek a dinamikus tartalomkészítés minden területén használhatók – a JDBC segítségével elérhetjük az adatbázisokat, HTTP-fejléceket húzhatunk le és módosíthatunk, és a válaszok tartalma a felhasználó igényei szerint módosítható.
A JSP oldalak segítségével könnyebben dolgozhatunk a servletekkel, feltéve, hogy a <% és %> tagok közötti rész kivételével minden szabványos, HTML nyelven íródik. Amikor a böngészőből egy JSP oldalt kérünk le, a JSP azonnal egy Java servletbe, majd ezt követően egy Java .class fájlba fordítódik. Ez a .class fájl töltődik be a servlet motorba, itt lefut és a „közelben” marad, egy későbbi meghívás esetére. A JSP-k és a servletek a Java Bean objektumokat is használhatják, ezekkel például a felhasználói szokásokat modellezhetjük. Ilyen elemek vezérlik a legtöbb mai háromrétegű webalkalmazás üzleti logikáját.
A mod_perl segítségével hatékonyan készíthetünk Apache-kezelőket, de néha túl alacsony szinten kell dolgoznunk miatta. Ennek kiküszöbölésére létezik néhány modul, amelyekkel a Perl kódot a HTML fájlba illeszthetjük. A HTML::Mason, mellyel a rovat tavalyi számaiban már foglalkoztam, a személyes kedvencem, és ezt egyszerű formai követelményeinek és az egymásba építhető sablonoknak köszönheti. Az idei YAPC::Europe találkozón Londonban láttam a Template Toolkit (Sablonkezelő Eszköztár) bemutatóját, mely a HTML::Masonra hasonlít, de bővítményeket is használhatunk benne. Míg a Java és a Perl főleg a kiszolgálóoldali webes programozási feladatok elvégzésére alkalmas, a PHP-t kifejezetten a dinamikus tartalomkészítésre fejlesztették ki. A PHP rengeteg szolgáltatással rendelkezik, ezeket sokféle fájltípushoz, adatbázishoz és internetes szabványhoz felhasználhatjuk. A PHP legfrissebb változatai Java-objektumokkal is képesek dolgozni, sőt, a CORBA csatolójára sem kell már sokáig várni. A PHP egyetlen hátránya, hogy nem bővíthető: ha későn jövünk rá, hogy például PDF fájlokkal is szeretnénk dolgozni, akkor ezt csak a PHP újrafordításával érhetjük el.
Az AOLServer felhasználói egy másik hasonló rendszerrel büszkélkedhetnek, melynek neve ADP (AOLServer Dynamic Pages) az ADP oldalak lehetővé teszik, hogy a Tcl-kódot a HTML fájlba illesszük be, ahol a Tcl az AOLServer által nyújtott szolgáltatások bármelyikét elérheti. Készíthetünk tehát olyan ADP oldalt, mely rekordokat olvas ki egy adatbázisból, egy másik kiszolgálóról letöltött HTML oldalt értelmezhetünk, vagy a felhasználói adatbevitel alapján számításokat végezhetünk.

Kapcsolatok az adatbázissal

A kiszolgálóoldali programozás világában a harmadik új irány az állandó adatbázis-kapcsolatok létrehozása. Az adatbáziskiszolgálókat eredetileg felhasználónként napi egy-két kapcsolatra készítették, és nem percenként vagy másodpercenként egyre. Gondoljuk csak el: ha egy CGI-programmal másodpercenként kapcsolódunk az adatbázishoz, akkor az eredeti terhelés 86 ezerszeresével dolgoztatjuk azt a szerencsétlen kapcsolatot! Már ha ez nem jelent különösebb gondot, azonban rengeteg olyan adatbázis létezik, ahol egyetlen kapcsolat is túl sokba kerül.
Az egyik megoldás lehet, hogy a kiszolgáló indulásakor létrehozunk egyetlen kapcsolatot, és ezt használjuk újra és újra, amikor egy programnak az adatbázisra van szüksége. Nagyjából ezt teszi a Perl,

Apache, mod_perl hármas együttes használata mellett alkalmazható Apache::DBI modul. Amikor a `$dbh->disconnect` segítségével leválunk az adatbázisról, az Apache::DBI csendben figyelmen kívül hagyja ezt, és életben tartja a kapcsolatot. A `DBI->connect` hívásakor az Apache::DBI beolvassa a kapcsolati karakterláncot, és az új kapcsolat indítása előtt megpróbál egy már létezőt fölhasználni. Mivel egy Apache-folyamat egyszerre csak egy HTTP-kérelmet szolgál ki, ezért folyamatonként egyetlen adatbázis-kapcsolatra van szükségünk. Az állandó kapcsolódást–leválasztást így megúszhatjuk.

Mindemellett ez azt is jelenti, hogy minden alfolyamatnak külön kell elérnie az adatbázist, ami egy erősen terhelt kiszolgáló esetén több száz vagy ezer egyidejű adatbázis-kapcsolatot jelent. Az AOLServer az adatbázis-kapcsolatok számát úgy csökkenti, hogy nem több folyamatot, hanem egyetlen, több szálat futtató folyamatot használ. Mivel a szálak ugyanabban a folyamatban kapnak helyet, így adatokat is képesek megosztani egymás között.

Az AOLServer ezt úgy használja ki, hogy a kapcsolatokból kis gyűjteményt hoz létre, és ha egy bejövő HTTP-kérelem igényt tart rá, akkor az egyiket kiválasztja, és ezen keresztül érhetjük el az adatbázist. A kapcsolatok igény szerint megoszthatók és nem kötődnek egy-egy szálhoz, így a kiszolgálónak jóval kevesebb kapcsolatot kell nyitnia az adatbázis felé.

A Java servletekkel és JSP oldalakkal való munka más megközelítést jelent. A Jakarta-Tomcat servlet/JSP rendszer általában a webkiszolgálón kívül található, tehát mindig a Tomcat folyamat részeként fut, az Apache alfolyamatok számától függetlenül. A Tomcat folyamaton belül tetszőleges számú servlet szál futhat egyidejűleg.

A servletek és JSP-k általában a JDBC segítségével érik el az adatbázist, és ugyanez igaz a Java Beanekre is (ezekkel a JSP-k és servletek magas szintű „gondolkodásra” képesek). A JDB azonban nem képes kapcsolatgyűjtemények kezelésére (connection pooling).

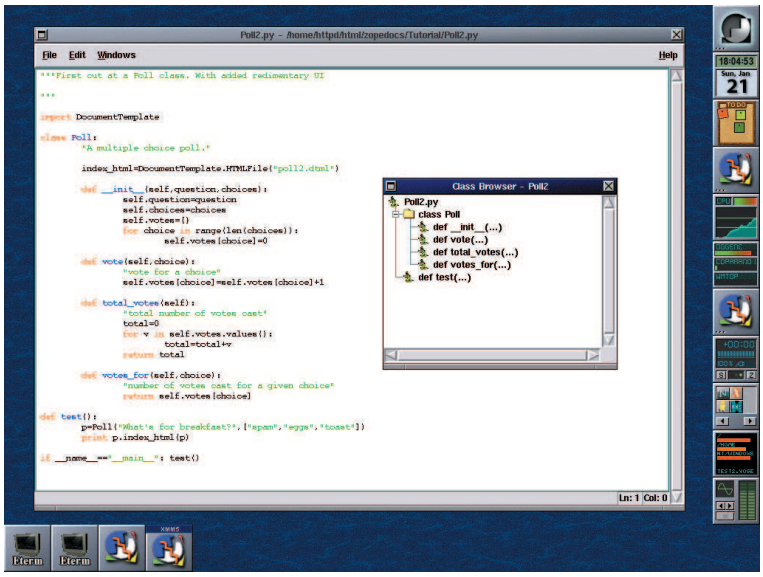
A JDBC 2.0 lehetővé teszi ezt is, de a művelet nem teljesen önműködő, és jelenleg kevés JDBC 2.0 meghajtó van.

Más nyelvek ezt teljesen másképpen oldják meg. Például a PHP adatbázis-meghajtói lehetővé teszik az állandó adatbázis-kapcsolatokat, de ezt a programozónak kérnie kell. Magyarul a `pg_connect` segítségével kapcsolódhatunk a PostgreSQL adatbázishoz, vagy állandó kapcsolatot is létrehozhatunk a `pg_pconnect` használatával. A két különböző elérési módszer megvalósítása az adatbázis programozójának feladata, a PHP programozónak pedig a kapott szolgáltatásokat kell a lehető legkedvezőbben kihasználnia. Mindezek közül az AOLServer állandó, gyűjteményes kapcsolatot biztosító módszere a legelegánsabb, hiszen bármilyen nyelvvel használhatjuk (bár általában ez a Tcl marad), és különösen jól méretezhető. A mod_perl Apache::DBI-je a Perl-programozók számára nagyszerű megoldás, mivel az önálló Perl-programokat és -modulokat nem kell módosítanunk azért, hogy az állandó kapcsolatokat kihasználhassuk. Az, hogy az Apache::DBI csak állandó, és nem gyűjteményes kapcsolatot ad, az Apache többfolyamatos felépítéséből következik. Az Apache 2.0, mely nagymértékben fog emlékeztetni az AOLServerre, valószínűleg már támogatni fogja a többszálas futtatást.

A JDBC gyűjteménykezelése (pooling) jó, még akkor is, ha már mindenki teljesen lemondott róla, és elkezdte saját osztályait írni. Azonban csak Java servletekkel működik, és nem segít az olyan kiszolgálóknál sem, melyeknek gyűjteményes kapcsolatokra van szükségük több szolgáltatás egyidejű működtetéséhez (mod_perl és JSP). A PHP rendszere talán a legkezdetlegesebb, hiszen nem bocsát rendelkezésünkre egységes felületet az adatbázis elérésére, az adatbázis-meghajtók nem kezelhetik önműködően a gyűjteményes kapcsolatokat, és a programok számára sem enged utat e kapcsolatok kihasználására. Az állandó kapcsolat viszont nagyon jól működik, és már ez önmagában is jelentős sebességnövekedést jelenthet.

Merre tartunk?

Bár az „alkalmazáskiszolgáló” kifejezést kétértelműsége miatt nem szeretem, tisztán látható, hogy a webes alkalmazások ebbe az irányba tartanak. Az alkalmazásokat már nem önálló programokként, hanem az alkalmazáskiszolgáló által rendelkezésünkre bocsátott objektumok és modulok egy csoportjaként kell elképzelnünk, melybe az általunk írt program tökéletesen illeszkedik. Sok esetben a lehető legkevesebb munkával is előállíthatunk bonyolult programokat, egyszerűen azért, mert a feladat nehezebbik részét mások már elvégezték helyettünk. Természetesen ez azt is jelenti, hogy az operációs rendszerre is másként kell tekintenünk, ugyanis valójában már csak az alkalmazáskiszolgáló legalsó rétege, s ez utóbbi a rendszer igazán lényeges eleme. Az ügyféloldali programok íróinak azt kell eldönteniük, hogy Windows, Unix vagy Macintosh rendszerre írják programjukat, a webes alkalmazások fejlesztőinek pedig az alkalmazáskiszolgálók közül kell választaniuk. Csakúgy, mint az operációs rendszerek esetében, az egyik alkalmazáskiszolgálóról a másikra történő áttérés is nehéz feladat. Ez, sajnos azt is magával hozza, hogy ha egy lassú, nehezen módosítható kiszolgáló mellett döntünk, akkor a jövőben lehet, nagyon komoly gondjaink lehetnek. Még az ugyanazon nyelvet és szabványokat használó alkalmazáskiszolgálók (például az Enhydra és az ATG Dynamo) is különböző objektumokkal és szolgáltatásokkal dolgoznak, így az áttérés eléggé fáradságos dolog.



A Zope kezelőfelülete

Az ingyenes programok hozzám hasonló híveinek ez azt is jelenti, hogy a nyílt forrású alkalmazáskiszolgálók legalább olyan fontosak, mint a nyílt forrású operációs rendszerek. Szerencsére az előbbi csoportból is számos változatot letölthetünk az Internetről. Működésükben és szolgáltatásaikban nagymértékben különböznek, de el kell ismernem, hogy a most ismertetendő módszerekhez még nem sokszor volt szerencsém. Remélem, hogy az elkövetkezendő hónapokban többet foglalkozhatok személyesen is ezekkel.

Talán a legismertebb alkalmazáskiszolgáló a Zope, ez sok részből épül fel, és kevesen ismerik elég jól. A Zope egy objektum-adatbázis, egy sablonozó rendszer és egy alapszintű tartalomkezelő rendszer. A Zope-pal még nem volt lehetőségem komolyabban foglalkozni, de a hallottak alapján egy rendkívül hatékony rendszer képe körvona-

lazódik előttem, amit főleg akkor használhatunk ki teljes mértékben, ha az igényelt szolgáltatást megvalósító modul már elérhető az Interneten.
Egy másik, gyakran emlegetett alkalmazáskiszolgáló rendszer az ArsDigita Content System, ennek fejlesztését nagyrészt az ArsDigita tanácsadó vállalat vezeti, s a GNU felhasználói engedély feltételei mellett használható. A rendszer egyik hátránya, hogy csak az Oracle adatbázis-kezelőt támogatja, ez egyébként egy kifogástalan rendszer, de elég drága és forráskódja sem nyílt. Erre is született már megoldás: az OpenARS nevű önkéntes fejlesztés a PostgreSQL adatbázissal igyekszik ötvözni a rendszert. Az OpenARS még nem készült el teljesen, de már most is hatalmas mennyiségű szolgáltatást találunk benne, melyek száma és hatékonysága minden bizonnyal tovább nő a jövőben.
Az XML jó néhány éve a legizgalmasabb viták tárgya a webes közösségben, de csak az utóbbi fél évben lehettünk szemtanúi egyre nagyobb mértékű elterjedésének. Az XML az adatokat a megjelenésüktől teljesen függetlenül írja le.
Az Enhydra egy Java-alapú alkalmazáskiszolgáló, mely sok tekintetben a Zope-ra hasonlít, de XML, Java servletek, JSP és Enterprise Java Beans használatát teszi lehetővé. Az Enhydra meglehetősen összetettnek tűnik, de legalább megbízható keretet teremt alkalmazásainknak.
Ha az XML-lel kívánunk dolgozni, akkor a Cocoon és AxKit csomagoknak is szenteljünk figyelmet. Az Apache Software Foundation által támogatott Cocoon most XML-adatok számára készít egy Java-alapú kiszolgálót. Az AxKit XML-alapú tartalomkészítést tesz lehetővé Perl nyelven, és így a programokat a tartalomtól, illetve a tartalmat a látványtól az XML, az XSL és a Perllel együtt az XSTL segítségével különíti el.
Végül az Oracle fejlesztését említeném meg, ennek neve Internet Application Server (IAS). Az IAS egy Apache-modul, mely egy Java futtatórendszerrel, az Enterprise Java Beansszel, JSP oldalakkal, a JDBC-csatolóval és az Oracle használatával végzi feladatát. Cikkem írásának időpontjában a rendszer még új és nem próbálták ki széles körben. Az Oracle természetesen ehhez sem ad forráskódot. Ennek ellenére az IAS futtatható Linux alatt és minden bizonnyal az Oracle-felhasználók kedvence lesz.

És merre tartok én?

Egészen idáig tanácsadói munkám legnagyobb része a Perlhez kötődött, ezt még mindig egy hatékony webes programnyelvnek tartom. Aki kérdezi, annak eddig azt mondtam, hogy nyolcvan százalékban Perllel, húsz százalékban pedig más programnyelvekkel (Java, Tcl, Python, C) foglalkoztam.
Az utóbbi időkben a webes programozói környezetek területén bekövetkezett jelentős változások hatására (ezt csak fokozta a szakma elmozdulása az alkalmazáskiszolgálók irányába) csapatommal együtt úgy gondoltuk, hogy nálunk is irányváltásra van szükség. Sok esetben még mindig a Perlt választanám, főleg akkor, ha a mod_perlt és a HTML::Masont is használhatom. Munkáinkhoz azonban egyre növekvő arányban alkalmazzuk a Java servleteket és JSP oldalakat, különösen a Tomcat servlet/JSP motorral és a PostgreSQL adatbázissal. A mod_perlről gyűjtött tapasztalatunk az AxKit felé terel bennünket, a servletek pedig egyre jobban ösztönöznek az Enhydra felfedezésére.
Az ACS-t már a korai időktől kezdve használjuk bizonyos nagyobb terjedelmű munkákhoz, javarészt a csomaghoz járó, már működő alkalmazások igen nagy száma következtében. Emellett az a tény, hogy az ACS ingyenes és Linux alatt is használható, nagyon egyszerűvé teszi a vele végzett munkát, hiszen az önkéntes fejlesztőközösség képzettségéhez és lelkesedéséhez nem férhet kétség, valamint a szolgáltatások, a leírás, az ellenőrzés és a hibajavítások miatt sem kell aggódnunk.
Összefoglalva: már most is számos módszer közül választhatunk, ezek némelyike csak az utóbbi időkben fejlődött ki. A rovat ötvenedik cikkének befejezéséhez közeledve és a jövőbe tekintve a webes fejlesztők számára hatalmas lehetőségeket látok. Különösen azok lesznek sikeresek, akik hisznek az ingyenes programok erejében és Linuxot használnak. Az eljövendő néhány év bizonyára rendkívül izgalmas lesz, és remélem, hogy néhány hónapon belül olvasóimmal is megoszthatom tapasztalataimat, és a kipróbált alkalmazásokkal használható példaprogramokat is.

További érdekességek

Az Apache Software Foundation, mely az Apache webkiszolgáló és a Jakarta-Tomcat, a Jakarta-Oro (regexp), valamint a Cocoon (Java-XML) projekteket támogatja:
➲http://www.apache.org/

A mod_perl honlapja
➲http://perl.apache.org/.
Itt mindenképpen olvassuk el *Stas Bekman* ismertetését a modulról, ez a programkörnyezetről összegyűlt adatokat is tartalmazza.

Az Oracle IAS
➲http://www.oracle.com/

Az AOLServer
➲http://www.aolserver.com/

Az AOLServert Python-értelmezővel bővítő PyWX
➲http://pywx.idyll.org/

Template Toolkit
➲http://www.template-toolkit.org/

HTML::Mason
➲http://www.masonhq.com/

Enhydra
➲http://www.enhydra.org/

Zope
➲http://www.zope.org/

AxKit
➲http://www.axkit.org/

ArsDigita
➲http://www.arsdigita.com/

Az OpenACS projekt
➲http://www.openacs.org/

Reuven M. Lerner (reuven@lerner.co.il) cége internetes tanácsadást vállal, székhelyük Modi'in-ben, Izraelben van. A *Core Perl* című könyv szerzője, mely a Prentice Hall kiadónál jelenik meg. Szeretettel vár mindenkit az ATF honlapján ➲http://www.lerner.co.il/atf/.

Memóriabütykölés

Hogyan léphetjük át az i386-alapú Linuxok folyamatszám-korlátját?

A folyamatok kezelése az operációs rendszer egyik legfontosabb feladata. Tervezése és megvalósítása nagymértékben befolyásolhatja a teljesítményt. Többfolyamatos operációs rendszerben számos folyamat fut egymással párhuzamosan, így növekedik a processzor kihasználtsága és a teljesítmény. A folyamatok párhuzamos futtatásával számos szolgáltatást biztosíthatunk, és egyszerre több ügyfelet is kiszolgálhatunk – ez a korszerű operációs rendszerek fő feladata.

Az Intel i386-alapú Linux felépítése támogatja a többfolyamatos működést. Megfelelő módszert választva a rendszer válaszideje a folyamatok ütemezése közben is alacsony marad, míg viszonylag nagy teljesítményt nyújt. Sajnos, a 2.2.x változatú rendszermagban egy olyan határ található, mely 4090-re korlátozza az egy időben futtatható folyamatok számát. Ez a szám bőven elegendő egy asztali gép esetében, de kevésnek bizonyulhat egy nagyvállalat kiszolgálóján.

Vegyük példaként egy általános webkiszolgáló működését, mely többfolyamatú/többszálú módszereken alapul. Amikor az ügyfél kérése megérkezik, a webkiszolgáló létrehoz egy új gyermekfolyamatot vagy szálat. Ily módon egy nagyobb terhelésű kiszolgálón könnyen előfordulhat, hogy egyszerre több ezer folyamat fut. Igaz, a nagyvállalati kiszolgálók túlnyomó része nem is Linuxot, hanem Solarist, AIX-ot, HP-UX-ot stb. futtat.

Számos Linux-fejlesztő észlelte ezt a gondot, és megpróbáltak úrrá lenni rajta. A 2.4-es rendszermagban már szerencsére megtalálható a megoldás. Mivel a 2.4-es mag több helyen is változott, sok esetben mégis a korábbi változat mellett maradunk, míg „hozzá nem szokunk” az újdonságokhoz. De hogyan lépjük át ezt a bűvös határt? Lehetséges olyan megoldást találni, mely képes a 2.2.x változat korlátainak áttörésére? Ahhoz, hogy minderre választ adjunk, először meg kell ismernünk a 2.2.x rendszermag folyamatkezelésének működését.

Az Intel i386 szerkezet és a Linux 2.2.x változatának memóriakezelése

A folyamatok kezelése szoros kapcsolatban áll a memóriakezeléssel. Mivel a memóriakezelés megvalósítása a gép felépítése alapján történik, először vessünk egy pillantást az i386 szerkezetére. A korszerű operációs rendszerekben széles körben alkalmazzák a virtuális memóriakezelés módszerét. Ennek köszönhetően a programok több memóriát használhatnak, mint amennyi ténylegesen elérhető. A programok által használt memóriacímek tehát virtuálisak, az elérés során a processzor által biztosított eljárások révén alakulnak át valódi címmé.

Két alapvető memóriakezelési módszer van: a szakaszolás (segmentation) és a lapozás (paging). A szakaszolás során a memóriát nagyszámú részekre osztják, majd részmutatók és eltolások (segment, offset) használatával érik el. Ezt az eljárást olyan korai rendszerekben is használták, mint például a PDP-11.

A lapozás azt jelenti, hogy a memóriát számos, azonos méretű lapra osztják fel, majd a lapokat a memóriakezelés alapegységeiként használják. A memóriaelérés során a laptábla adatai szerint végeznek átalakítást a program által használt cím és a tényleges cím között.

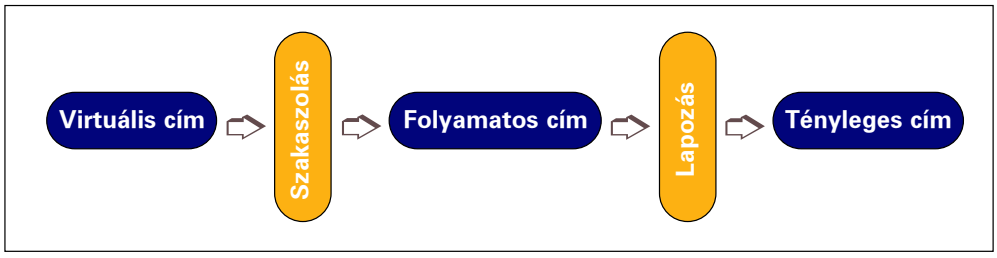
Az i386 szerkezetben használt memóriakezelés szakaszolásos lapozás. A virtuális memóriacím-tartományt két táblázat, az Általános Leírótábla (GDT) és a Helyi Leírótábla (Local Description Table – LDT) segítségével először szakaszokra osztják. Ezt követően a virtuális címet folyamatos címmé (linear address) alakítják. A folyamatos címeket kétszintű táblázathasználattal, a Lapkatalógus (Page Directory Table) és a Laptábla (Page Table) segítségével alakítják tényleges címmé. A virtuális és tényleges cím közötti átalakítás folyamata az *1. ábrán* látható.

Linuxnál a rendszermag a nullás szinten fut. Az Általános Leírótábla beállításával a rendszermag külön címterületre helyezi el programkódját és adatait. Minden egyéb program a hármas szinten fut, adataik és programkódjaik azonos címtartományban találhatók. Külön laptáblák létrehozásával ezek a felhasználói programok védhetők. A 2.2.x változatú Linuxban használt Általános Leírótábla a *2. ábrán* látható. A programok a Helyi Leírótábla (LDT) átírásával oldhatják meg, hogy más adat/kódterületet használjanak.

A 2.2.x rendszermag folyamatkezelése

Egy folyamat egy futó program és a hozzárendelt erőforrások összessége. Ez egy meglehetősen képlékeny megfogalmazás. Sokan a feladatokat (tasks) is egy-egy folyamatnak tekintik. Az egyszerűség kedvéért a továbbiakban mi továbbra is a folyamat elnevezést használjuk. A folyamatkezelés fogalma olyan műveletekkel függ össze, mint a rendszer indítása, folyamatok létrehozása és megszüntetése, ütemezés, folyamatok közötti kapcsolattartás stb. Linux esetében a folyamat valójában adatszerkezetek egy csoportja, mely magába foglalja a folyamat környezetét, az ütemezési adatokat, jelzőket, a folyamatok várakozási sorát, a folyamat azonosítóját, más folyamatokkal fenntartott kapcsolatát stb. Ezt az adategyüttest nevezzük Folyamatvezérlő Tömbnek (Process Control Block – PCB). A futtatás során a PCB a folyamatverem alján található.

A Linux folyamatkezelése nagymértékben támaszkodik a géptípus lehetőségeire. Az előbbiekben az i386 rendszerek szakaszolásos lapozásának alapjait tárgyaltuk csak, de a memóriaszakasz (segment) valójában nem csak egyszerűen a memória egy darabját jelenti. Például a Feladatállapot Szakasz (Task Status Segment – TSS) az i386-alapú rendszer egyik legfontosabb szakasztípusa. Rengeteg olyan adatot tartalmaz, mely a rendszer működéséhez szükséges. Minden folyamatnak van egy, a TR regiszter által mutatott TSS-e. Az i386 előírásai szerint a TR-ben tárolt mutatónak a GDT-ben kell kijelölnie egy leírót. Emel-



1. ábra Virtuális címek átalakítása

lett az LDTR-ben található mutatónak – mely egy folyamat LDT-jét adja meg – a GDT-ben is rendelkeznie kell egy megfelelő bejegyzéssel. Ahhoz, hogy a fenti követelményeknek megfeleljen, a 2.2.x Linux minden lehetséges folyamathoz egy GDT-t rendel. A párhuzamos folyamatok legnagyobb száma a rendszermag indításakor kerül meghatározásra. A rendszermag két GDT-bejegyzést tart fenn minden folyamathoz.

Rendszerindítás

A Linux 2.2.x változata alatt a rendszer indításakor bizonyos folyamatkezeléssel kapcsolatos adatok is betöltésre kerülnek. Ezek közül a legfontosabb a GDT és a folyamatlista.
Amikor a rendszermag elindul, meg kell határoznia a GDT méretét. Mivel minden folyamathoz két bejegyzés tartozik a GDT-ben, a GDT méretét a párhuzamosan futtatható folyamatok legnagyobb száma határozza meg. Linux alatt ez az érték fordítási időben, NR_TASKS néven érhető el. A GDT mérete 10 + 2 (APM-mel) + NR_TASKS * 2.
A folyamatlista valójában PCB-mutatók egy tömbje, melyet a következők szerint adunk meg:

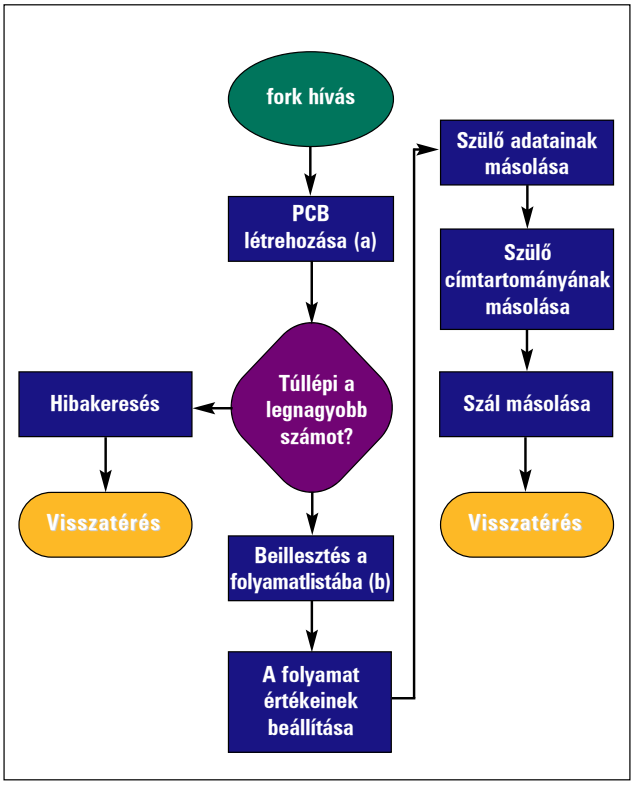
```
Struct task_struct *task[NR_TASKS] =
       {&init_task,};
```

A fenti sorban az init_task a főfolyamat (root process) PCB-je. Miután a folyamatot beillesztettük a folyamatlistába, a folyamatkezelő rendszer is megkezdheti munkáját. Megjegyezzük, hogy a folyamatlista mérete szintén függ az NR_TASKS értékétől.

Folyamatok létrehozása

A Linux 2.2.x változatánál a folyamatokat egy rendszerhívás, a **fork** hozza létre. Az új folyamat az eredeti folyamat gyermeke lesz. Másolat készítésével, klónozással létre tud hozni új szálat, mely tulajdonképpen egy pehelysúlyú folyamat. Mint látjuk, valójában a Linux 2.2.x alatt nincs tényleges „szál”. A *3. ábrán* a fork rendszerhívás működése látható.
A fork legfontosabb lépései a következők:
1. Az új folyamat PCB-jének létrehozása: a rendszermag két lapot lefoglal az új folyamat vermének, majd annak aljára elhelyezi a PCB-t.
2. Az új folyamat beillesztése a folyamatlistába: a rendszermagnak keresnie kell egy üres bejegyzést a folyamatlistában. Ha elértük a párhuzamosan futtatható folyamatok legnagyobb számát, a rendszermag nem talál ilyet, és sikertelenül végződik a rendszerhívás.
3. A szülőfolyamat címtartományának másolása: a gyermekfolyamatnak saját címtartománya van, de először a szülőfolyamattal megosztott címtartományt kap (copy-on-write módszerrel). Az új folyamat LDT-jéhez tartozó GDT-leíró is ilyenkor jön létre.
4. A TSS beállítása az új folyamathoz: az új folyamat TSS-e létrejön a PCB-ben, illetve létrejön a hozzá tartozó GDT leíró is.

2. ábra Az Általános Leírótábla szerkezete



3. ábra A fork rendszerhívás

Ütemezés

Az ütemező eljárás vázlata az alábbiakban látható. Itt azonban csak érintőlegesen foglalkozunk a folyamatváltással. A Linux 2.2.x változatánál a folyamatok közti váltást a switch_to eljárás végzi. A következők szerint működik:
1. Új TSS betöltése a TR beállításával.
2. A régi FS és GS regiszterek mentése a régi PCB-be.
3. Az LDT betöltése, ha az új folyamatnak szüksége van rá.
4. Új laptáblák betöltése az új folyamathoz.
5. Az új folyamat FS és GS értékeinek betöltése.

Megjegyezzük, hogy a TR és LDTR értékét a PCB szolgáltatja.

A legnagyobb folyamatszám túllépése

Mi is a folyamatok legnagyobb számára vonatkozó megszorítás? Az eddigiek alapján már tudjuk is, hogy egyáltalán miért van ilyen megszorítás. A Linux 2.2.x változatában megadott NR_TASKS állandó értékkel, már fordítási időben megszabja az egymással párhuzamosan futtatható folyamatok legnagyobb számát.
Ugyancsak az NR_TASKS, és ugyancsak fordítási időben szabja meg a GDT méretét. Amint az i386 szerkezet is meghatározza, a GDT legnagyobb mérete 8192x8 bájt, azaz összesen 8192 leírót tartalmazhat. 2.2.x változatú Linux esetében a rendszermag indításakor a GDT használata az alábbiak szerint történik:

1. NULL leíró (0 bejegyzés), fenntartott leíró (1., 6., 7. bejegyzés).
2. A rendszermag programkódjának és adatainak leírói (2. és 3. bejegyzés) és a felhasználói programkód és adatok leírói (4. és 5. bejegyzés).
3. APM BIOS leírók (8–11 bejegyzések).

Tehát összesen 12 bejegyzést használunk. Mivel minden folyamatnak két GDT bejegyzésre van szüksége, elméletileg (8192-12)/2=4090 folyamatot futtathatunk egy időben.

1. lista Rendszerindítás

```
# A GDT méretét a lehetõ legnagyobbra (8192) állítja a head.S fájlban.
# A desc.h-ban található GDT-meghatározásba illesszük be az egyes processzorokhoz
# tartozó bejegyzéseket, a többi bejegyzést pedig (az eredeti rendszerhez hasonlóan)
# a folyamatok számára tartja fenn.
CPU0: SHARED_TSS_ENTRY = 12
SHARED_LDT_ENTRY = 13
CPU1: SHARED_TSS_ENTRY + 1 = 14
SHARED_LDT_ENTRY + 1 = 15
...
CPUn: SHARED_TSS_ENTRY + n = SHARED_TSS_ENTRY + n
SHARED_LDT_ENTRY + n = SHARED_LDT_ENTRY + n
( n < NR_CPUS )
# Az NR_TASKS-ot makróról változóra módosítja, és ezt a start_kernel() rendszerindítási
# függvényben végzi:
Task = kmalloc(sizeof(void *) * NR_TASKS, GFP_ATOMIC);
# Ez dinamikusan foglalja le a folyamatlista tömbjét.
# A parse_options függvény módosításával egy nrtasks nevû kiegészítõ értéket is
# használhatunk, mely az egy idõben futó folyamatok legnagyobb számát jelzi.
# Így a felhasználó beállíthatja a folyamatok legnagyobb számát.
```

A nehézségek megszüntetése

Annak ellenére, hogy a GDT méretét a gép korlátozza, találhatunk megoldást a gondra. Egyetlen processzoron adott pillanatban csak egy folyamat futhat. Ennek megfelelően tehát nem kell GDT leírókat fenntartani az összes többi folyamatnak. Amikor egy folyamatot futtatni készülünk, leíróit dinamikusan állítjuk be.
A PCB felépítésének elemzése után megtalálhatjuk benne a TSS-t és az LDT-t – ha vannak egyáltalán. Folyamatváltáskor tehát a PCB mutató alapján találhatjuk meg ezt a két szakaszt, a következők szerint:

```
TSS: proc->tss
LDT: proc->mm->segments
```

2. lista: Megosztott GDT-bejegyzések használata

```
...
#define set_shared_tss_desc(addr,cpu)\
_set_tssldt_desc(gdt_table+
SHARED_TSS_ENTRY+2*cpu,(int)addr,235,0x89);

#define set_shared_ldt_desc(addr,size,cpu)\
_set_tssldt_desc\
       (gdt_table+SHARED_LDT_ENTRY+2*cpu,
       (int)addr,((size<<3)-1),0x82);
...
void __switch_to(task_struct *prev,
       task_struct *next){
...
if(next->tss.tr <= 0x0000ffff)
{
              //az eredeti kód helye
} else {
              set_shared_tss_desc(&next->tss),
                     smp_processor_id());
       set_shared_ldt_desc(&next->mm->segments,
              LDT_ENTRIES,smp_processor_id()); }
// az LDTR és a TR beállítása
...
}
```

Folyamatváltáskor a PCB mutatót valójában a folyamatlistából kereshetjük elő. Mivel mind a TSS, mind az LDT előkereshető, ezeket teljesen felesleges folyamatosan a GDT-ben tartani.
Megoldásunk az, hogy minden processzorhoz csak két GDT-leírót tartunk fenn, közös bejegyzéseket használva minden folyamathoz. Egy kétprocesszoros gépnél például négy bejegyzést kell fenntartani. Amikor az A folyamat fog futni az egyes processzoron, a harmadik és negyedik GDT bejegyzést állítjuk be az A folyamat TSS és LDT leíróihoz. Ezeknek a bejegyzéseknek a régi értékeit elvesztjük.
A fennmaradó GDT-bejegyzéseket ugyanúgy használjuk, mint az eredeti rendszer esetében.

Megjegyzés a megvalósításhoz

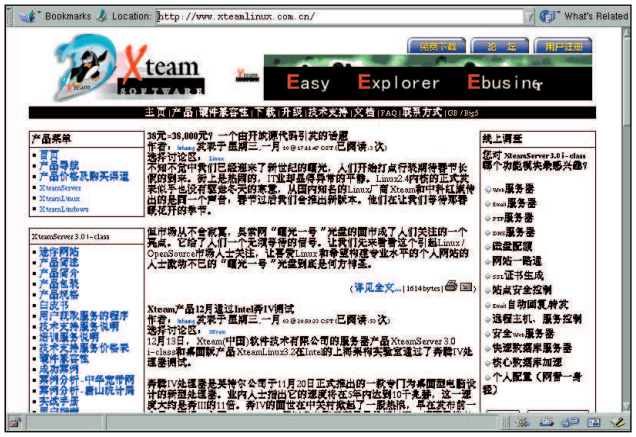
Megoldásunk alapját a folyamat TSS és LDT leírójának dinamikus beállítása képezi. (Lásd az első kódrészletet.)

Folyamatváltás

Az eredeti tervezés szerint a fork rendszerhívás végrehajtásakor a PCB tss.ldt és tss.tr elemeit használjuk az LDTR és a TR mutatóinak mentésére. Az eredeti eljárás szerint egy folyamat LDT-jében a mutató mérete átlépheti a 16 bites határt, így a mutató mentésére a tss.ldt mellett egy külön változót, a tss._ldth-t is használunk. Mivel a Linux 2.2.x változata nem használja a tss._ldth változót, változtatásunk nem fogja érzékenyen érinteni a rendszermagot. Az LDTR és a TR mentése ezután a következők szerint történik:

```
((unsigned long * ) & (p->tss.ldt)) =
       (unsigned long)_LDT(nr);
if (*((unsigned long *) & (p->tss.ldt)) <
       (unsigned long) (8192<< 3)
       set_ldt_desc(nr,ldt, LDT_ENTRIES);
       // az eredeti programkódban itt else {
       // majd nem csinálunk semmit,
       // hagyjuk, hogy a folyamatváltó programkód
       // kezelje az LDT-t és a TSS-t
}
```

A megvalósítás előnyeinek egyike az, hogy az tss.ldt értékének vizsgálatával könnyedén megtudhatjuk, vajon a folyamat sorszáma túllépte-e a 4088-at. Ez fontos lehet a teljesítmény miatt.



Ha a folyamat sorszáma 4088-nál nagyobb, nincs fenntartott bejegyzése a GDT-ben, és a megosztott GDT bejegyzéseket kell használnia. Ezeket a bejegyzéseket a következő kódrészlettel találhatjuk meg:

```
SHARED_TSS_ENTRY + smp_processor_id();
```

A második kódrészlet a megosztott GDT bejegyzések kezelését szemlélteti.
Ha elvégeztük az eddigieket, sikerült átlépnünk a folyamatok számára vonatkozó felső korlátot. A lilo beállítófájlba egy további értéket is elhelyezhetünk, mellyel meghatározhatjuk ezt az értéket. A következő programsor negyvenezerre állítja be a folyamatok legnagyobb számát:

```
Append="nrtasks=40000"
```

Tanulság

Az ismertetett megoldás szerint az egymással párhuzamosan futtatható folyamatok számának felső értékét két gigában határozhatjuk meg – elméletileg. A gyakorlatban azonban a gép és az operációs rendszer továbbra is korlátozzák ezt az értéket. Amikor új folyamat jön létre, a rendszermag a következők szerint foglal le számára memóriát:

```
Folyamatverem (2 lap) + Laptábla (1 lap) +
       Lapkatalógus (1 lap) = 4 lap
```

Ha tehát a számítógépnek 1 GB memóriája van, az operációs rendszer ebből 20 megabájtot használ, és minden folyamat öt lapot kap, a folyamatok legnagyobb száma a következő:

```
(1 GB - 20 MB) / 20 kB = 51404 ~= 50000
```

Gyakorlatiasabb példát nézve, ha minden folyamat legalább 30 kB memóriát használ, az előbbi érték a következők szerint alakul:

```
50000 × 2/3 = 33000
```

Azonban még ez a szám is lényegesen nagyobb, mint a 4090.

➲ http://www.xteamlinux.com.cn

Zhang Yong (leon@xteamlinux.com.cn) az Xteam Software Co., Ltd. vezető programmérnöke. Munkája a Linux számos területét érinti, ilyen a Linux rendszermag fejlesztése, a Linux I18N&I10N, hálózati alkalmazások stb.

Craig Hunt: Linux DNS Server Administration

A tartománynév-szolgáltatás (Domain Name Service, DNS) a hálózatkezelés és az Internet szerves része. A Linux-változatok általában a BIND-et, a Berkeley Internet Name Domain csomagot tartalmazzák, ez végzi a DNS kezelését. A Linuxról szóló könyvek azonban ritkán foglalkoznak a DNS használatával és beállításával. Bár a BIND beállítása nem a legegyszerűbb feladat, mégsem lehetetlen. A Craig Hunt által írt *Linux DNS Server Administration* ismerteti a DNS működését és a beállításához szükséges lépéseket. A szerző, mint ismert TCP/IP- és Linux-szakértő, körültekintően mutatja be a linuxos BIND működését.
A könyv első része a DNS szerkezetével, a használható protokollokkal és a BIND csomaggal foglalkozik. A /etc/hosts fájról szóló rész végre pontosan bemutatja a fájl használatát, előnyeit és hátrányait. A DNS szerkezetét a szerző a tartományokon, a tartománynév-keresésen és a kérelmek feldolgozásán keresztül mutatja be. A DNS-üzeneteket ábrák mutatják be, és arra is fény derül, hogy a DNS-adatbázisokat hogyan lehet összehangolni.
Ezt követi a BIND telepítése és üzemeltetése. A szakasz végén tippeket találhatunk a számunkra leginkább megfelelő DNS-rendszer felállításához.
A második részben a DNS beállítása a téma, s ennek három fejezetet szentelt az író. Az elsőben a resolv.conf, a host.conf és az nsswitch.conf szerepét és működését vesézi ki, a következőben pedig a gyorstárazás (caching) és a másodlagos (slave) kiszolgálók beállítását tárgyalja, s javaslatokat is tesz létrehozásuk módjára. A szakasz záró fejezetében az elsődleges (master) kiszolgáló létrehozásáról olvashatunk (ez felel az adott tartományért és beállítása talán a legösszetettebb). Minden fejezetben találunk példaként szolgáló beállításfájlokat, tartalmukat a szerző minden esetben részletesen elmagyarázza.
A harmadik szakaszban a BIND beállítása kerül sorra. Itt sok érdekességről olvashatunk: hogyan állíthatunk föl nem kiutalt részt-tartományokat egy zónán belül, mikor van szükség kiutalt tartományok létrehozására, illetve miként oszthatunk szét egy hálózati szolgáltatást. Ebben a részben kaptak helyet a DNS „finomhangolására” szolgáló módszerek is, melyek mind-mind a teljesítmény növelésében játszanak fontos szerepet. A dinamikus DNS (DDNS) sem maradhatott ki – ez a protokoll megszabadít bennünket a DNS beállításának nehezétől, hiszen a gépet utasítja arra, hogy a hálózaton elérhető adatok alapján hozza létre az adatbázist.
A könyv befejező részében a szerző egy működő DNS-szolgáltatás karbantartásának lépéseit ismerteti. A könyvhöz négy függelék is tartozik. Az „A” függelék a BIND 9 új lehetőségeit és a Beta 2 kiadás telepítését ismerteti. A „B” függelék a named.conf fájlban használható parancsokat foglalja össze, a „C” függelékben pedig a BIND által támogatott 41 erőforrásrekord kerül bemutatásra. Az utolsó függelékben a *Network Information Service* (NIS) kiszolgáló beállításáról olvashatunk.
Véleményem szerint e könyv viszonylag könnyen érthető, útmutatásai alapján otthoni hálózatomban is könnyedén sikerült egy DNS-kiszolgálót felállítanom. A kiadvány rengeteg ábrája, meghatározása és mintafájlja még könnyebbé teszi munkánkat.

Ralph Krause

Craig Hunt: Linux DNS Server Administration (ISBN: 0782127363)
Beszerezhető a Kiskapu Kft. mintaboltjában:
1081 Budapest Népszínház utca 29. Tel.: (06-1) 303-9119

Háromrétegű tervezés

Ismerjük meg a köztes programréteg kialakítását a mod_perl/Apache környezetben.

Néhány hónapja megvizsgáltuk a Masont (Linuxvilág 2000. november, 59. oldal), mely egy korszerű webfejlesztő eszköz. Ez a mod_perlt, az Apache-t és a sablonokat használja. Az egyik példánk egy sajtóközleményeket kezelő rendszer volt, amelyben Mason-elemek szedik elő a legfrissebb sajtóközleményeket egy adatbázisból. Ebben a cikkben az úgynevezett kétrétegű programozói stílusra ismerhetünk rá, azaz a Mason-programok közvetlenül beszélgetnek az adatbázissal a Perl DBI és mod_perl Apache::DBI használatával.

De ahogy sokan megjegyezték a Mason levelezőlistán, ez a megközelítés – amelyben az SQL utasítások közvetlenül a Mason összetevőkben szerepelnek – sokszor nem célszerű. Az adatbázis szerkezetének változása, vagy egy másikfajta adatbázis-kezelőre való áttérés arra kényszerítene, hogy módosítsuk magukat az összetevőket. Ráadásul a nem webes programoknak újra meg kell valósítaniuk az SQL hívásokat a saját kódjukban ahelyett, hogy egy egységesen fenntartott és ellenőrzött közös programkönyvtárat használnának. Mindkét gond megoldódik, ha egy új réteget adunk a programhoz, amely az adatbázis és a Mason-programok között helyezkedik el. Ezt az egyre népszerűbb felépítést nevezik háromrétegű megközelítésnek, mivel három, egymástól jól elhatárolható programcsoporttal van dolgunk. Ezek az adatbázis, a középső réteg és a megjelenítési réteg (a felhasználói program).

Most és a következő hónapban egy egyszerű webalapú címjegyzéket és határidőnaplót fogunk vizsgálni, amelyen bemutatjuk a háromrétegű megközelítést. Remélem sikerül megvilágítani a módszer előnyeit és hátrányait, és mindenki képes lesz értékelni e módszer jelentőségét, ha egy nagyobb webhelyet kell kialakítania.

Miután áttekintettük az általános felépítést, készek leszünk arra, hogy megismerkedjünk a Java Server Pages és a Jakarta-Tomcat rejtelmeivel és az alkalmazáskiszolgálókkal. Megvizsgáljuk az e megközelítésben felbukkanó csapdahelyzeteket, és azt is, hogy miként segíti ez a gondolkodásmód a fejlesztés egyszerűbbé és méretezhetőbbé tételét hosszú távon.

Az adatbázisok

Az első, talán legfontosabb réteg a relációs adatbázis. Ebben a példában PostgreSQL-t fogok használni, de használhatnánk bármi mást is, mondjuk Oracle-t vagy MySQL-t.

Az adatbázis megtervezése előtt meg kell határoznunk, hogy mit akarunk tenni, azaz mire akarjuk használni. A cikk céljaira bőven elég egy rövid és homályos leírás az elérendő célokról: szeretnénk egy címjegyzéket, ami a Weben keresztül megnézhető, kereshető és frissíthető. Ezen felül szeretnénk a webböngésző segítségével a határidőnaplóba találkozókat bejegyezni, onnan törölni, illetve módosítani.

Ehhez legalább két tábla szükséges, az egyikben tartjuk nyilván az embereket, a másikban a találkozókat. Az *1. listán* olvasható az emberek adatait tartalmazó táblát létrehozó kód.

Az egyes emberek keresztnevét, vezetéknevét, országát és elektronikus levélcímét mindig tároljuk, ezenkívül pedig a címet, a hozzá kapcsolható várost, az államot, az irányítószámot és egy megjegyzést is készíthetünk. Ezzel feltételeztük, hogy minden embernek van levélcíme – olyan feltevés ez, ami egyre inkább igaz, de nem biztos, hogy jó ez a jellemző, ha számos barátunk és üzletfelünk dolgozik a számítástechnikai iparon kívül.

Az embereket tartalmazó tábla minden eleme egyértelműen azonosítható a `person_id` oszlop által, amelyet a PostgreSQL automatikusan növel. Biztosak lehetünk abban is, hogy minden ember csak egyszer szerepel a listán, mivel a levélcímnek egyedinek kell lennie.

Ez megengedi, hogy két Szabó István nevű barátunk legyen, de azt is jelenti, hogy egy közös levélcímet használó házaspár tagjait nem tudjuk külön-külön felvenni a listára. Szintén nehézkes a több levélcímet használó emberek kezelése.

Egy új ember felvétele a táblába viszonylag egyszerű:

```
INSERT INTO People
       (first_name, last_name, address1,
       address2, email, city, state, postal_code,
       country, comments)
       VALUES
       ('György', 'Szy', 'Népszínház u. 31',
       'I. em. 7.', 'szy@linuxvilag.hu',
       'Budapest', NULL, '1081', 'Magyarország',
       'A szerkesztőség címe')
       ;
```

Az oszlopok többségének alapértelmezett értéke a `NULL`, tehát egy ember felvétele elvégezhető úgy is, hogy kizárólag a kötelezően kitöltendő oszlopokba írunk:

1. lista A People tábla meghatározása

```
CREATE TABLE People (
       person_id   SERIAL       NOT NULL,
       first_name  VARCHAR(20)  NOT NULL  CHECK  (first_name <>''),
       last_name   VARCHAR(20)  NOT NULL  CHECK  (last_name <>''),
       address1    VARCHAR(30)  NULL      CHECK  (address1 <>''),
       address2    VARCHAR(30)  NULL      CHECK  (address2 <>''),
       email       VARCHAR(50)  NOT NULL  CHECK  (email ~'@'),
       city        VARCHAR(30)  NULL      CHECK  (city <>''),
       state       VARCHAR(2)   NULL      CHECK  (state <>''),
       postal_code VARCHAR(10)  NULL      CHECK  (postal_code <>''),
       country     VARCHAR(20)  NOT NULL  CHECK  (country <>''),
       comments    TEXT         NULL      CHECK  (comments <>''),
       PRIMARY KEY(person_id),
       UNIQUE(email)
    );
```

```
INSERT INTO People
       (first_name, last_name, email, country)
       VALUES
       ('János', 'Kiss', 'janos@kiss.hu',
       'Magyarország')
       ;
```

Készítsük el most a találkozókat tartalmazó táblát. Ebben a People táblában tárolt személyekkel tervezett találkozóinkat tároljuk:

```
CREATE TABLE Appointments
       (
       person_id   INT4       NOT NULL
                              REFERENCES People,
       start_time  TIMESTAMP  NOT NULL,
       end_time    TIMESTAMP  NOT NULL,
       notes       TEXT       NULL
                              CHECK (notes <>''),
       UNIQUE(start_time)
       );
```

Most, hogy a találkozók táblája elkészült, hozzáadhatunk egy új találkozót úgy, hogy egy sort illesztünk a táblába:

```
INSERT INTO Appointments
       (person_id, start_time, end_time, notes)
       VALUES
       (1, 'January 22, 2001 14:00',
       'January 22, 2001 14:30', 'Tárgyalás')
       ;
```

Mivel a `person_id` a People táblából származó idegen kulcs, csak olyan emberrel szervezhetünk találkozót, aki megtalálható a People táblában. A mi céljainknak ez megfelel így is, de egy bonyolultabb, jobban megtervezett rendszer valószínűleg rugalmasabb lenne. Természetesen az adatbázis nem fogja hagyni, hogy egy időben több emberrel találkozzunk.

Középréteg

Miután megalkottuk a kiindulási adatbázist, gondolkozzunk el a középső réteg szerkezetén, amely elszigeteli egymástól az adatbázist és a webes alkalmazást. Ha egyszer egy másikfajta adatbázis-kezelőre váltunk, vagy az adatbázist ASCII vagy DBM fájlokra cseréljük, az objektumok rétege változatlan marad.
Ráadásul a nem webalapú alkalmazások is használhatják ezt a réteget arra, hogy az adatbázishoz hozzáférjenek, ezzel elérhetővé válik például a határidőnaplónk kimentése XML formátumban, vagy találkozók beolvasása más programokból.
Ezt a középső réteget szokás az alkalmazás ügymenetének, vagy üzleti logikájának (business logic) is nevezni. Az adatbázis lehetővé teszi, hogy adatainkat könnyen, gyorsan tároljuk és elérjük.
A Mason-összetevők segítségével könnyűszerrel készíthetünk dinamikus kimenetet a végfelhasználó számára. A középréteg megpróbálja rákényszeríteni az adatbázist, hogy minél több számítást végezzen el a beépített függvények, a nézetek és a tárolt eljárások használatával. Az alkalmazás működését meghatározó logika azonban a középső rétegben helyezkedik el.
A Perl legalább két lehetőséget kínál e réteg létrehozására. Az egyik lehetőség, hogy egy egyszerű Perl-modult hozunk létre, ennek függvényei és változói segítségével a feladat megoldható. Az ilyen

3. lista Retrieve-people.pl

```
#!/usr/bin/perl -w
use strict;
use People;
# Új People példány létrehozása
my $people = new People;
# A keresett személy kiválasztása név szerint
$people->set_current_person_by_name("Shai",
                                    "Re'em")
    || die "Hiba a személy kiválasztásakor";
# A személy adatainak lekérése
my $info = $people->get_current_info();
# Az adatok kiíratása
foreach my $key (sort keys %{$info})
{
    if (defined $info->{$key})
 {
        print "$key => $info->{$key}\n";
    }
}
print "-" x 60, "\n";
# ----------------------------------------------
# Új személy beszúrása
my $success = $people->new_person(
                  first_name => "Reuven",
                  last_name => "Lerner",
                  country => "Israel",
                  email => 'reuven@lerner.co.il',
                  phone => '08-973-2225');

if ($success)
{
    # Adatok kérése erről a személyről
    $info = $people->get_current_info();
    # Az adatok kiíratása
    foreach my $key (sort keys %{$info})
    {
 if (defined $info->{$key})
        {
            print "$key => $info->{$key}\n";
        }
    }
}
else
{
    print "Hiba!\n";
}
exit;
# ----------------------------------------------
# Most állítsuk a nevet valami másra
$people->update_first_name("Yochai");
# A személyhez tartozó adatok lekérése
$info = $people->get_current_info();
# Az új adatok kiíratása
foreach my $key (sort keys %{$info})
{
    if (defined $info->{$key})
    {
        print "$key => $info->{$key}\n";
    }
}
print "\n";
```

eljárásalapú programozói felület könnyen megírható, és ugyanolyan gyorsan fut, mint a többi Perl-program. A Perl lehetővé teszi az objektumalapú modul megírását is.
Bár kissé nehezebb Perl-objektumokat írni, és az eljárásaik is lassabban futnak le, mint az egyszerű függvények, a segítségükkel könnyebb átlátni és megírni a programot.
Néhány komoly kérdésre meg kell találnunk a választ, mielőtt belevágnánk a középső réteg létrehozásába. Milyen objektumokat hozzunk létre? Készíthetnénk egy olyan adatbázis-objektumot, amely minden lekérdezést úgy kezel, hogy a megfelelő SQL paranccsá fordítja le őket. Igen ám, de esetenként csak az emberről szóló adat érdekel, az esetleg vele szervezett találkozók nem, tehát legalább két objektumra lesz szükség, egy People és egy Appointment objektumra. Mivel az adatbázisunkat úgy terveztük meg, hogy minden találkozóhoz hozzárendelődik egy és csak egy ember, a találkozó objektumát csak az ember objektuma után határozhatjuk meg.

People.pm

A People.pm (a *2. és 4. lista* túl hosszú ahhoz, hogy az újság hasábjain közöljük, de a cikkhez tartozó összes lista letölthető tgz formátumban az ➲ftp.ssc.com/pub/lj/listings/issue81/ címről.) Ez egy objektummodul, amely néhány egyszerű feladatot végez el az imént létrehozott People táblával. Az objektum nem teljes és néhány helyen lenne mit csiszolni rajta, de arra jó, hogy bemutassa egy relációs adatbázis elérését egy objektumalapú középső rétegen keresztül.
Az alapötlet az, hogy létrehozzuk a People objektum egy új példányát, és aztán a határidőnaplóban ezen objektum segítségével kezeljük az embereket. Az adatbázisban szereplő összes ember kigyűjtéséhez a `get_all_full_names` eljárást használhatjuk, mint ebben a kódrészletben is látható. (Lásd még az ➲ftp.ssc.com/pub/lj/listings/issue81/ címről letölthető *4. listát*):

```
use People;
# Egy People objektum létrehozása
my $people = new People;
# Az összes név kiolvasása
my @names = $people->get_all_full_names();
# A nevek kinyomtatása
foreach my $name (@names)
{
    print "name\n";
}
```

Egy adott emberre vonatkozó adat megszerzéséhez vagy módosításához először meg kell adni a szóban forgó személyt. Mivel a középső rétegnek az a lényege, hogy megszabadítsa a felhasználót a kulcsok és egyéb elsődleges azonosítók használatától, a kívánt személy kiválasztható lesz a keresztnév, a vezetéknév vagy a levélcím alapján.
A levélcím biztosan egyedi az adatbázis szintjén, ezért a `set_current_person_by_email` a legbiztonságosabb eljárás. Ennek ellenére gyakran hasznos az embereket keresztnevük és vezetéknevük alapján azonosítani, ezért használható a `set_current_person_by_name` eljárás is. A jelenlegi megvalósításban az eljárás az első olyan sort fogja visszaadni az adatbázisból, amelyre a megadott név megegyezik. Ez nem feltétlenül az lesz, akit mi szeretnénk visszakapni. Miután a program beállította az kívánt személyt, adatait a `get_current_info` eljárással olvashatjuk ki:

```
# A kívánt személy beállítása név szerint
$people->set_current_person_by_name
        ➥("János","Kiss")
|| die "Hiba: a személy nem található!";
# Az adatok nyomtatása
foreach my $key (sort keys %{$info})
{
       if (defined $info->{$key})
       {
              print "$key => $info->{$key}\n";
       }
}
```

A People objektum minden példánya két adatot tárol: a pillanatnyilag kiválasztott személy azonosítóját (`$self->{current_person}`) és az adatbázis-azonosítót, amely az adatbázishoz kapcsol minket (`$self->{dbh}`). Azért érdemes tárolni az adatbázis-azonosítót, mert az adatbázishoz kapcsolódás viszonylag drága művelet. Időt nyerhetünk, hogy az adatbázishoz kapcsolódást a konstruktorban (létrehozóban) valósítjuk meg, és mindig ezt a kapcsolatot használjuk, amikor az objektum egy eljárását hívjuk.
Ez természetesen azt jelenti, hogy az adatbázis-kapcsolatot a Perl-objektum életének végén meg kell szüntetni. Ez egy kicsit cseles, mert a Perlben nincs kifejezetten destruktor (megszüntető), lévén a Perl egy szemétgyűjtő nyelv. A megoldás az, hogy egy `DESTROY` nevű eljárást hozunk létre, ezt kell az objektum megszűnésekor meghívni. A mi `DESTROY` eljárásunk egyszerűen csak lezárja az adatbázis-kapcsolatot, ezáltal az objektumot nyugodtan törölhetjük, nem fog memóriaszivárgást okozni sem az adatbázis-kezelőben, sem az ügyfélprogramban:

```
sub DESTROY
{
       # Saját magunkra hivatkozás megszerzése
       my $self = shift;
       # Adatbázis-azonosító megszerzése
```

5. lista Insert-appointment.pl

```
#!/usr/bin/perl -w
use strict;
use People;
use Appointments;
# Új People példány létrehozása
my $people = new People;
# Új Appointments példány létrehozása
my $appointments = new Appointments;
# A kívánt személy kiválasztása név szerint
$people->set_current_person_by_name("Hadar",
                                    "Re'em")
    || die "Hiba: a személy nem található!";
# Új találkozó hozzáadása a kívánt személlyel
my $result = $appointments->add_appointment
                    ➥($people,
                    'October 22, 2000 18:30',
                    'October 22, 2000 19:00',
                    'Társasjáték');
# Sikerült?
if ($result)
{
    print "Sikerült!\n";
}
else
{
    print "Nem sikerült: '$DBI::errstr'"
                            ➥unless $result;
}
```

6. lista Print-appointments.pl

```
#!/usr/bin/perl -w
use strict;
use diagnostics;
use Appointments;
# Új Appointments példány létrehozása
my $appointments = new Appointments;
# Az aznapi találkozók listájának elkészítése
my @appointments = $appointments->get_today();
# Végimegy a találkozókon
foreach my $appointment (@appointments)
{
    # Mindegyik találkozó mutatótömb, tehát az
    # az elemeit kell kiíratni
    print $appointment->{start_time}, ":\t";
    print "\t", $appointment->{first_name},
        " ", $appointment->{last_name}, "\n";
 print "\t", $appointment->{notes}, "\n";
}
```

```
        my $dbh = $self->{dbh};
        # Az adatbázis-kapcsolat lezárása
        $dbh->disconnect;
        return;
}
```

A `new_person` eljárás segítségével akár új személyeket is bejegyezhetünk. Értékként meg kell adni a kulcskészletet és az értékkészletet, amelyeket felhasználva aztán a a középső réteg létrehozza a megfelelő SQL-kifejezést:

```
# Új ember hozzáadása
my $success = $people->new_person
        (first_name => "Reuven",
        last_name => "Lerner",
        country => "Israel"
        email => 'reuven@lerner.co.il',
        phone => '08-973-2225');
print "A hozzáadás sikerült." if $success;
```

Mivel a Perlben a meghatározatlan (`undef`) érték automatikusan az SQL `NULL` értékévé fordítódik, a kitöltetlen oszlopokba `NULL` kerül.

Találkozók

Most, hogy megvan az adatbázisunkban tárolt embereket kezelő osztály, létre kell hoznunk a találkozók osztályát. Egyelőre csak az új találkozók beillesztésével és az aznapi találkozók megjelenítésével foglalkozunk.
Az Appointment.pm (*4. lista* az ➲ftp.ssc.com/pub/lj/listings/issue81/ címen) szerkezete lényegében hasonló a People.pm felépítéséhez, a konstruktorban (létrehozóban) nyitja meg az adatbázis-kapcsolatot, és az automatikusan meghívott `DESTROY` eljárással zárja le. Ezenkívül az Appointment objektum nem tárol más állapotot, egyszerűen csak egy csővezetéket hoz létre az adatbázis felé, ennek segítségével új találkozókat hozhatunk létre és az aznapi találkozókat jeleníthetjük meg.
Például az *5. lista* egy rövid programot tartalmaz, amely az Appointments.pm-et használja egy új találkozó létrehozására. Létre kell hoznunk az emberek és a találkozók egy-egy példányát.

Miután megvan ez a két objektum, beállíthatjuk a megfelelő személyt. Ha a `set_current_person_by_name` az `undef` hibaüzenettel tér vissza, a program hibaüzenettel leáll.
Ha a kívánt személy beállítása sikerült, akkor létrehozhatunk vele egy találkozót. A dátum és az idő formátumát a PostgreSQL határozza meg (elég sok formátumot elfogad).
Hasonlóképpen olvashatjuk ki a mai napra előjegyzett találkozók listáját a *6. listán* bemutatott program használatával (print-appointments.pl).
A program a `get_today` eljárást használja, amely egy mutatótömb-listát (list of hash references) ad vissza. Megjegyzendő, hogy a `get_today` megvalósítása a DBI **fetchrow_hashref** eljárását használja, ami közismerten sokkal lassabb, mint a **fetchrow_arrayref**. Ennek ellenére az a megoldás sokkal kényelmesebbé teszi az életünket, mert így megvalósítható a `print-appointments` a *6. listán* bemutatott módon.
Végül megnézhetjük, hogy egy adott személlyel a mai napon mikor és hányszor találkozunk. A `get_today_with_person` eljárást erre használjuk. Természetesen ez azt jelenti, hogy előbb létre kell hoznunk a People egy példányát, és ki kell választanunk a kívánt személyt a fent tárgyalt eljárások egyikével. A `get_today_with_person` megvalósítása a felhasználó által átadott első értékként a People egy példányát várja, így a megfelelő személy kerülhet az SQL-lekérdezésbe. A *7. listán* olvasható program bemutatja, hogyan tudhatom meg az összes aznapi találkozóm időpontját, amelyeket az unokaöcsémmel, Shaival beszéltem meg.

Az objektumok megtervezése

Az objektumok középrétegben történő használatának egyik fő oka az, hogy egy elvonatkoztatott réteget nyújt, azaz amíg a csatolófelület jól meghatározott és állandó, a tényleges megvalósítás változhat.
Igaz viszont, hogy mint minden alapos programozási módszer, a jó objektumok megtervezése is nehéz feladat lehet. A Perl szabad

7. lista Print-appointments-with-shai.pl

```
#!/usr/bin/perl -w
use strict;
use diagnostics;
use People;
use Appointments;
# Új People példány létrehozása
my $people = new People;
# Új Appointments példány létrehozása
my $appointments = new Appointments;
# A kívánt személy kiválasztása név szerint
$people->set_current_person_by_name("Shai",
                                    "Re'em")
   || die "Hiba: a személy nem található!";
# Az aznapi találkozók listájának elkészítése
my @appointments =
  $appointments->get_today_with_person($people);
# Végimegy a találkozókon
foreach my $appointment (@appointments)
{
    # Mindegyik találkozó mutatótömb, tehát
    # az elemeket kell kiíratni
    print $appointment->{start_time}, ":\t";
    print "\t", $appointment->{first_name},
        " ", $appointment->{last_name}, "\n";
    print "\t", $appointment->{notes}, "\n";
}
```

Forrásművek

Sok könyv és cikk szól a háromrétegű tervezésről.

Például a John Wiley kiadásában Robert Orfali, Dan Harkey és Jeri Edwards: The Essential Distrubuted Objects Surviving Guide című könyve. Ez jó bevezető a háromrétegű szerkezetek elméletébe, valamint egyes kereskedelmi megvalósításokat is bemutat. Elmagyarázza a sokrétegű szerkezet fogalmait, ezek közül sok fontos akad a webes alkalmazások szempontjából. A mű egy kicsit elavult, az OpenDocot említi, de nem szól a DCOM-ról. Mindamellett jó áttekintést ad ezekről a programozási módszerekről.

Szintén a John Wiley kiadó gondozásában jelent meg Jeri Edwards és Deborah DeVoe 3-Tier Client/Server at Work című kézikönyve, amely az elmélet néhány jelenlegi megvalósítását tárgyalja.

A könyvek elolvasása után az ember azt hiheti, hogy minden feladatot a háromrétegű tervezéssel kell megoldani. Ez természetesen nem így van, a tervezés függ többek között attól is, hogy mit és hogyan kell megvalósítani. Egy jó esettanulmányt írt általában a webalapú alkalmazáskiszolgálókról és különösen a háromrétegű tervezésről Philip Greenspun (az ArsDigita alapítója és szerzője a kiváló Philip and Alex's Guide to Web Publishing című műnek, amit a Morgan-Kaufmann adott ki). Az esettanulmány a weben is olvasható a ➲ http://www.arsdigita.com/asj/application-servers.adp címen.

Végül, a Sun elkezdte a Javára terelni az emberek figyelmét, ami szerintük egy kiváló nyelv a háromrétegű megoldásokhoz. Ezt a következő hónapokban meg fogjuk vizsgálni. Erről olvashatunk pár dolgot a következő könyvek első fejezeteiben: Ed Roman: Mastering Enterprise JavaBeans, John Wiley kiadó, valamint Richard Monson-Haefel: Enterprise JavaBeans, O'Reilly and Associates kiadó.

hozzáférést enged az objektum belsejéhez, ez azonban azt a veszélyt hordozza magában, hogy jó API hiányában az objektummal dolgozó programozó kísértést érezhet arra, hogy belenyúljon az objektum belsejébe, és közvetlenül a megvalósítással dolgozzon. Ennek az lehet a következménye, hogy a program működésképtelenné válik, ha a megvalósítás változik, márpedig pont ezt a helyzetet akartuk megakadályozni az objektumok használatával.

Továbbá azt szeretnénk, hogy objektumaink megvalósításai viszonylag függetlenek legyenek egymástól. A People és az Appointment objektumok megvalósításakor nagy kísértést éreztem arra, hogy megengedjem a találkozóknak, hogy a szóban forgó személy azonosítószámát megszerezzék és használják. Természetesen ez megszegte volna azt szétválasztási határt, amelyet az objektumok létrehozásával felépítettem. A megoldás – ami bevallom, nem olyan elegáns, mint szerettem volna – a get_current_person eljárás megalkotása lett. Ezáltal az Appointment objektum úgy tudhatja meg a kívánt személy nevét, hogy nem kell tudni, hogy az honnan származik. Végül a `get_current_person` visszatérési értéke bekerül az SQL-kifejezésbe, és összehasonlításra kerül a People.person_id-vel, ez bizonyos értelemben mégiscsak megszegi a szétválasztást.

Végezetül figyeljük meg, hogy az egyes objektumok nem tárolnak állapotadatokat. Viszonylag egyszerű lenne például, hogy a People objektum a People tábla minden sorát kiolvassa, és elérhetővé tegye az őt hívó objektumok számára. Valóban, ez a megoldás jelentősen csökkentené az adatbázis felé irányuló forgalmat, és megengedné, hogy Perlben végezzük el az adatok kezelését ahelyett, hogy mindig az SQL-hez kelljen fordulnunk.

Csakhogy ez a megoldás több gondot okoz, mint ahányat megold. Például mi történne, ha két példányt hoznánk létre a People objektumból? Két objektumunk lenne, mindkettőben az emberek összes adata benne lenne. Ha az egyik objektum állapota változna, ez semmilyen módon nem jelentkezne a másik objektumnál. Még rosszabb esetben: mi történne, ha mindkét objektum állapota azelőtt változna, mielőtt a változások megjelennek az adatbázisban? Egy jó adatbáziskezelő valószínűleg képes megbirkózni ezekkel a versenyhelyzetekkel, de a Perl-objektumok nem. Továbbá mi lenne, ha a People tábla százezer személy adatait tartalmazná? Ennyi adat beolvasása az ügyfélprogramba memóriapazarlás lenne, és az adatbázis-kezelő nagy teljesítményű lekérdezési és adatkezelési eljárásai is kihasználatlanul maradnának.

Az objektumaink ezért csak az adatbázishoz kapcsolódó csővezetékek, amelyek lehetővé teszik, hogy a webalapú alkalmazásunk anélkül tudjon beszélgetni az adatbázissal, hogy SQL-kifejezéseket kelljen beágyazni, vagy ismernie kellene a táblázatok szerkezetét. Az objektumok által nyújtott szabványos API használatával elérhető, hogy az alatta fekvő megvalósítást megváltoztassuk anélkül, hogy ezeket a változásokat a világ tudomására hoznánk.

Összefoglalás

Megvizsgáltuk, hogy milyen okok vezetnek a háromrétegű rendszerek használatára, és tanulmányoztunk egy ezzel a módszerrel készült egyszerű alkalmazást. Látható, hogy máris képesek vagyunk kisebb alkalmazásokat létrehozni. A következő hónapban befejezzük a most elkezdett program megvalósítását, ennek a végeredménye egy egyszerű, mod_perl/Mason és PostgreSQL hármast használó határidőnapló-program csontváza lesz. Megtárgyaljuk a háromrétegű megoldások méretezhetőségével kapcsolatos gondokat és néhány csapdahelyzetet.

Reuven M. Lerner (reuven@lerner.co.il) cége internetes tanácsadást vállal, székhelyük Modi'in-ben, Izraelben van. A *Core Perl* című könyv szerzője, mely a Prentice Hall kiadónál jelenik meg. Szeretettel vár mindenkit az ATF honlapján ➲http://www.lerner.co.il/atf/.

Zene füleinknek

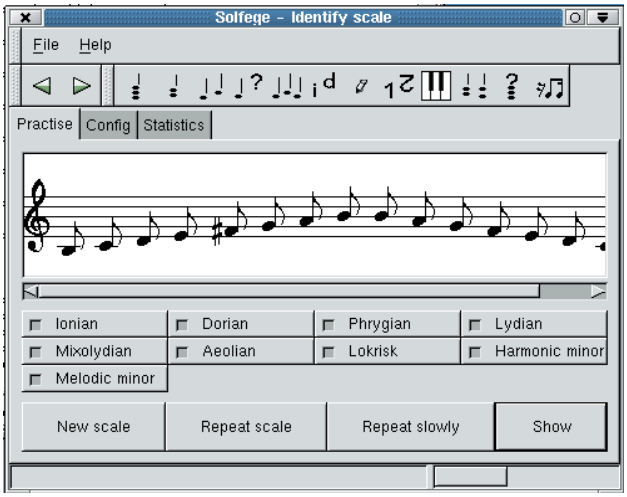
Gagné most zenét ír Linuxszal.

Fölhangosítanád a zenét, un petit peu, François? Ah, merci. Monsieur Debussy jó formában volt, amikor ezt írta, ugye, mon ami? Csodálatos! Hm, hát annak a Shakespeare-nek, annak ellenére, hogy angol, igaza volt abban: „ha a zene a szerelem kenyere, akkor csak játssz!”. Természetesen az ételhez jó bor is jár, és míg a szobát a Pelléas et Mélisande hangjai töltik be, addig én szívesen töltenék egy pohárka Medocot, mondjuk Cos-d’Estourelt. François kérlek, hozz föl egy keveset a pincéből.

Ah, mes amis. Jöjjenek csak be, foglalják el foteljaikat, François máris elkészül a terítéssel. Azonnal rendelhetnek valami ínycsiklandó finomságot, amint visszaér a borokkal. Éppen egy varázslatos operát hallgattunk, és közben rájöttem, hogy e pompás alkalomhoz egy kis zamatos borocska illik a legjobban.

A zene, akárcsak a főzés, nem cél, hanem egy egész életen át tartó utazás. A Linuxszal meg a világ nyílt forrású konyháinak néhány receptjével bárki fölfedezheti a zenét, legyen szó akár könnyed élvezetről, vagy a lélek legmélyéről jövő őszinte előadásról. Mais oui, hát nem értik? Ma még csak programozgat. Holnap már a zene világának ünnepelt csillaga. Azt mondják, a nagy iram előtt nem árt egy kis séta, non? Pontosan így van ez a zenével is. Először a leckék jönnek, egy kis gyakorlás, még egy kis gyakorlás, és máris a Carnegie Hallban találjuk magunkat, non?

Az útra azonban nem árt felkészülni egy kicsit, kezdjük, mondjuk a



1. kép Skálázás a Solfege segítségével

hangfelismeréssel és egy kis fülgyakorlattal. Először a GNU Solfege programot fogjuk használni.

A GNU Solfege *Tom Cato Amundsen* linuxos boszorkánykonyhájából érkezett hozzánk. A programocska segítségével fülgyakorlatokat végezhetünk. Ezek a gyakorlatok a hangközök, a hangmagasságok és ritmusok rejtelmeibe vezetik be a kezdő zenészt. A második rész a kottából olvasás (vagy inkább kottából éneklés), ennek során a tanuló egy rövid kottát (vagy akkordot) kap, és ezt kell kiénekelnie. A kottaolvasást egyedül is gyakorolhatjuk, bizonyos feladatokhoz azonban egy segítőtársra van szükség, „aki” lejátssza a kiéneklendő hangközöket. Itt lép a képbe a Solfege nevű program. A szolfézs eredetileg bizonyos magánhangzók (a, o, u) vagy szótagok (dó, ré, mi stb.) kiéneklését jelenti, ennek során megtanulhatunk pontosan énekelni és megismerjük a hangközöket.

Ha a Solfege honlapon a SourceForge oldalaira mutató hivatkozásra kattintunk, megtalálhatjuk a DEB és az RPM csomagokat, ezek egyszerűvé teszik a telepítést. De természetesen a legfrissebb, gzippel tömörített .tar fájl is letölthető. Néhány más csomagra is szükség lesz a program helyes működéséhez: Python 1.5.2, a GNOME könyvtárak (legalább 1.0.50-es változatúak legyenek) és egy PyGNOME nevű apróság (ez is 1.0.50 fölött legyen). A Python és a GNOME nagy valószínűséggel mindenki gépén megtalálható, a PyGNOME csomag (becsületes neve gnome-python) pedig a legtöbb friss Linux-változatban helyet kapott, de nem feltétlenül. Nekem a tar fájlból kellett telepítenem (ez is letölthető a Solfege honlapjáról):

```
tar -xzvf gnome-python-1.0.53.tar.gz
cd gnome-python-1.0.53
make
make install
```

Ha ez a helyére került, akkor minden készen áll a Solfege telepítéséhez. Amint már említettem, a legegyszerűbb megoldás a DEB vagy RPM csomagok letöltése a SourceForge oldalairól. Ha mégis saját magunk szeretnénk elkészíteni, az sem lesz túl bonyolult:

```
tar -xzvf solfege-0.7-24.tar.gz
cd solfege-0.7-24
make
make install
```

A program indításához egyszerűen írjuk be a solfege parancsot (az *1. képen* a Solfege látható, működés közben).

Nos, mes amis, itt elképzelhető, hogy kisebb-nagyobb nehézségekbe ütközünk. Alapértelmezés szerint a Solfege a /dev/music nevű eszközzel szeretne beszélgetni. Nálam az alapértelmezett MIDI eszköz a /dev/sequencer, tehát mindenképpen át kellett böngésznem az INSTALL fájlt (ehhez azért a François által előkerített 1987-es finom Musigny is hozzásegített). Aggodalomra semmi ok – a MIDI eszközt bármikor módosíthatjuk a File menü Preferences pontjának választásával. A következő menü Sound Setup lapján állítható be az eszközmeghajtó vagy egy külső lejátszóprogram (például a Timidity).

A fülgyakorlat egy életre szóló kaland a zenészek számára, így a Solfege-t akár a profik is megkedvelhetik.

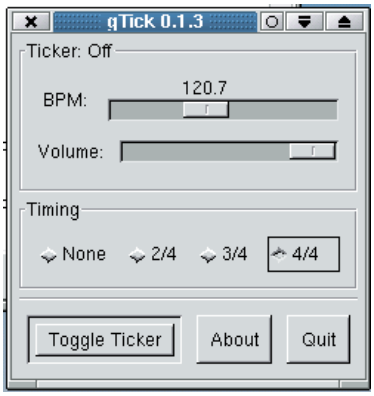
Akik már zenészek (vagy legalábbis ezt gondolják magukról), azok nyilván tisztában vannak azzal, mily fontos a zenében a ritmus és az időzítés pontossága. Egy jó metronóm sok zenész fegyvertárának egyik legfontosabb eleme. Ezt *Alex Roberts* is így gondolhatta, máskülönben miért írta volna meg a gtick nevű programot? Ez egy kis GTK-alapú program, mely az ütemet egész, 2/4-es, 3/4-es és 4/4-es képletekben jelzi. A sebesség 30-tól 250 bpm-ig (percenkénti ütemszám) állítható be.

A gtick telepítésével és használatával nyilván senkinek nem lesz gondja. Csomagoljuk ki a forrást (tar -xzvf gtick-0.1.3.tar.gz), lép-

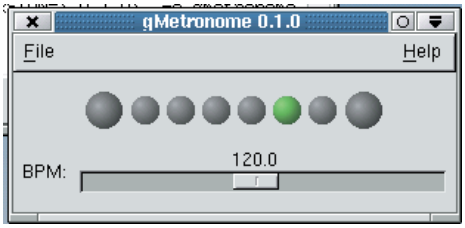
jünk be a létrehozott könyvtárba (cd), majd adjuk ki a make parancsot. A kapott bináris fájlt (gtick) másoljuk oda, ahol programjainkat szoktuk tárolni. Én például a /usr/local/bin könyvtárba helyeztem:

```
cp gtick /usr/local/bin
```

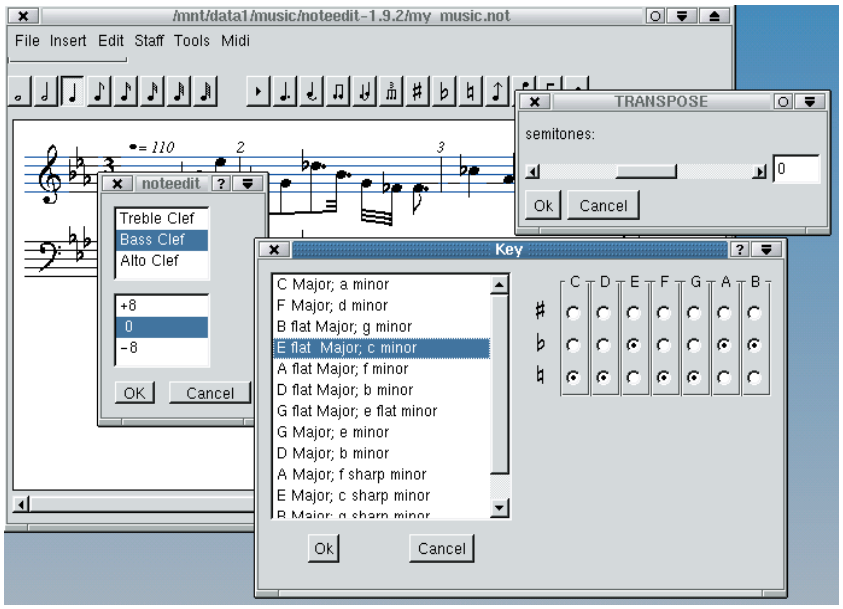
Egy másik jól használható metronómprogram a *David Lee* által írt, és szintén a GTK vezérlőelemeit használó gMetronome. Ez az alkalmazás valamivel látványosabban jeleníti meg a ritmust: egy ütem alatt nyolc lámpa villan fel az ablakban. A sebesség 0 és 250 bpm között állítható. Jelenleg ez a program nem ad ki hangot, tehát csak a szemünkre támaszkodhatunk. Lehet, hogy David barátunknak lenne egy-két szava Alexhez, non?
A gtick-hez hasonlóan a gMetronome telepítése és futtatása is gyerekjáték. A forráscsomag kibontása (tar -xzvf gMetronome-0.1.0.tar.gz) után lépjünk be az újonnan létrehozott könyvtárba, majd írjuk be a make és a make install parancsokat. A programot ezután a gmetronome paranccsal indíthatjuk.
Néha le is kell jegyeznünk a dallamot ahhoz, hogy megoszthassuk másokkal. A Linuxszal, valamint néhány nagyszerű és ingyenes programmal álmainkat igényes formába önthetjük, amit azután MIDI kottaként, vagy akár papírra nyomtatva is terjeszthetünk. Körülnéztem a számításba jöhető programok között, és örömmel jelenthetem, hogy számos olyan érdekességet találtam, melyek bizonyára alkotásra késztetik a bennünk rejlő tehetséget.



2. kép A gtick



3. kép A gMetronome formába önti a ritmust



4. kép Beethoven nyomdokaiba léphetünk a NoteEdit segítségével

Bocsássanak meg nekem, mes amis, de el kell ismernem, hogy jó néhány borízű chanson kivételével az Önök konyhafőnökének nem sok köze van a zeneszerzéshez. Ebből következik, hogy először egy könnyen használható kottaszerkesztő programot kerestem, és így jutottam el *Jörg Anders* művéig, a NoteEditig. A NoteEdit egy tetszetős küllemű alkalmazás: a hangjegyeket és a szüneteket kényelmesen, az eszköztárból behúzva illeszthetjük be a kottába. A programban válthatunk az előjegyzések és az ütemek között, az egérrel választhatjuk ki a hangszer hosszúságát, átültethetjük (transzponálhatjuk) a dallam tetszőleges részletét stb. Ha elégedettek vagyunk mesterművünkkel, a kész dalt elmenthetjük MIDI fájlként, vagy előkészíthetjük nyomtatásra MusicTex formátumban is. A NoteEdit a *4. képen* látható.
A NoteEdit egy QT alkalmazás, mely a TrollTech 2.x változatú „development” könyvtárfájljaira épül. Aki a KDE 2 környezetet használja, annak ezek a fájlok már telepítve vannak a rendszerén. A NoteEdit felépítéséhez töltsük le a legfrissebb forrásfájlokat és kövessük az alábbi lépéseket:

```
tar -xzvf noteedit-1.9.2.tgz
cd noteedit-1.9.2
./configure -without-libs
make
```

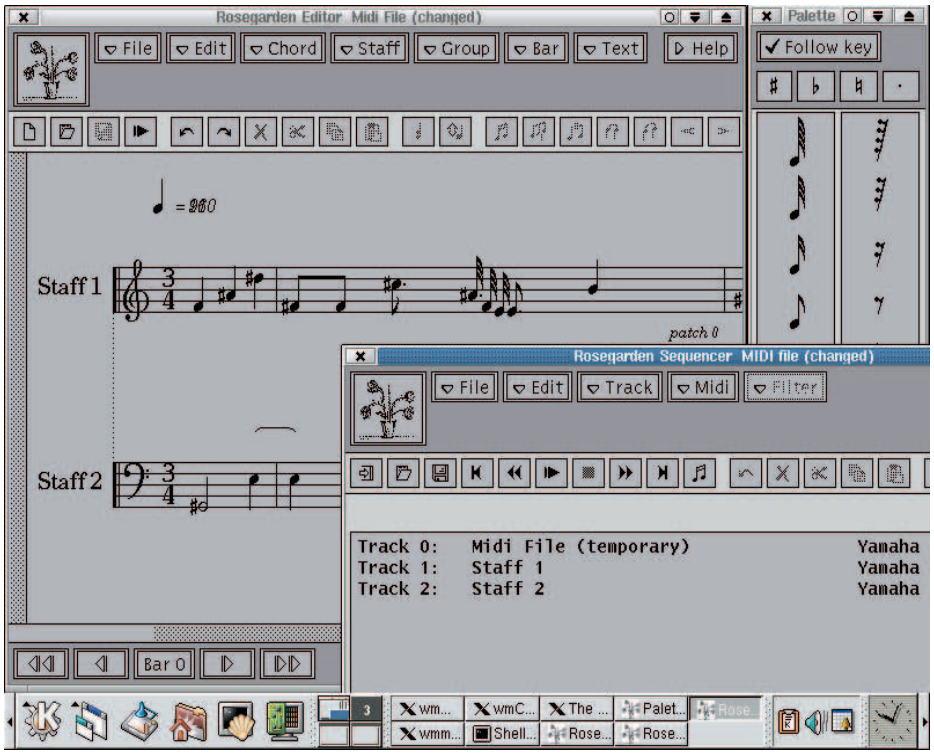
Figyeljük meg, hogy a beállítófájlt a -without-libs értékkel futtattam. A NoteEdit telepítésekor háromféle változat közül választhatunk. A legjobb kiépítésben a NoteEdit a TSE3 könyvtárfájlokat használja. Ezekkel MIDI fájlokból vagy egy csatlakoztatott MIDI-billentyűzetről is beolvashatunk adatokat. Most a legegyszerűbb felépítést mutatom be, az izgalmasabb kalandokra áhítozók keressék fel a program honlapját, ahol minden szükséges tájékoztatást megtalálhatnak.
Miután felépítettük a programot, indítás előtt egy környezeti változót is be kell állítanunk:

```
export NOTE_EDIT_HOME={{/a_NoteEdit/telepítési/
```

```
        könyvtára/}}
noteedit
```

Ennyi az egész. Innentől az alkotókészségünkön, no meg (legyen ez javaslat) egy-két pohár boron múlik minden. Néha a Play gombra kattintva lejátszhatjuk addigi ténykedésünk eredményét. Mesterművünket bármikor menthetjük, és ha kedvünk tartja, visszatérhetünk a mentett változathoz. Mielőtt a nyomtatásról beszélnénk, hadd mutassak be még egy kottaprogramot.

A *Chris Cannam*, *Andy Green* és *Richard Bown* neve által fémjelzett



5. kép Bátraké a szerencse

Rosegarden megér egy kis odafigyelést. Ez a program valamivel összetettebb, mint a NoteEdit: a csomag képes akkordok, triolák és mindenféle zenei bűbáj kezelésére, szóval a hírnév innen már tényleg csupán néhány kattintásnyira van. A beépített zongorabillentyűzettel használhatjuk MIDI sávrögzítőként is. A NoteEdithez hasonlóan itt is menthetünk MusicTex formátumban.

A Rosegarden nemcsak Linux, hanem bármely Unix-változat alatt használható. A honlapra ellátogatva jó néhány bináris fájl közül választhatjuk ki a rendszerünknek legjobban megfelelőt (a fájlokat akár ELF formátumban is letölthetjük). A Rosegarden kipróbálásához én a forráskódot töltöttem le és a következő parancsokat adtam ki:

```
tar -xzvf rosegarden-2.1-sources.tar.gz
cd rosegarden-2.1
./configure
make
make install
```

A programot felépítése és telepítése után a rosegarden paranccsal indíthatjuk. A Rosegardenben a korábban a NoteEdittel megkezdett káprázatos zeneművemet csiszolgattam tovább (ami iszonyatos, de mint említettem, nem sokat értek a zenéhez). Aki elég bátor, az pötyögje csak be az *5. képen* látható dallamocskát.

Tehát a nyomtatás.

A NoteEdit és a Rosegarden is képes MusicTex formátumban menteni, mely nem más, mint a Linux TeX csomagjának zenei bővítése (új betűtípusok, néhány újabb alkalmazás stb.). Segítségével gyönyörű kottákat nyomtathatunk. Sajnos, egyik program sem tartalmaz „Kattints ide a nyomtatáshoz” feliratú gombot, szóval fogadjuk el, hogy a hírnévhez előbb mélyen be kell hajóznunk a MusicTex óceánjára. Természetesen nem túl nehéz megbirkózni vele, de a MusicTex ismertetése jóval meghaladná e cikk kereteit. A NoteEdit honlapján olvasmányos bevezetést találunk a témáról, én azt javasolnám, hogy mindenki itt kezdje. A kemény munka jutalma sok-sok szép kotta lesz.

Egyébként a linuxos kottaszerkesztés még eléggé kialakulatlan terület: a fejlesztések egymástól függetlenül, több irányban is folynak. Vannak szép grafikus rendszerek (mint például a NoteEdit vagy a Rosegarden), és szöveges alkalmazások is, melyek a begépelt adatokból PostScript oldalakat készítenek. Néhány ezek közül igencsak gyermeteg, néhány viszont olyannyira összetett, hogy különszámot jelentethetnénk meg belőlük. Az azért megnyugtató, hogy mind a kezdő, mind pedig a profi zenészek találhatnak igényeiknek megfelelő programokat. Miközben François már a tálcáinkat mosogatja, hadd ajánljam olvasóim figyelmébe *Dave Philips* honlapját. Ő a készítője és karbantartója a „Sound and Midi Software for Linux” honlapnak, melyen az összes ismert linuxos zeneprogram megtalálható. Ide tehát mindenképpen érdemes bekukkantanunk.

Mon Dieu! Már megint ilyen későre jár, non? Attól tartok, mes amis, hogy most be kell zárnom Marcel bárját, de csak a legközelebbi alkalomig. Mivel mindenki gyalog jött, François tölt még egy-egy pohárkával vendégeinknek. E rövid kis körutazás után hallgassuk tovább Monsieur Debussy remekművét.

Az asztaluk legközelebb is várja Önöket!

Á votre santé! Bon appétit!

Marcel Gagné (mggagne@salmar.com) az ontariói Mississaugában él. A Salmar Consulting nevű hálózati tanácsadó cég elnöke. Szabadidejében repülőgépet vezet, sci-fi és fantasy regényeket ír, valamint a TransVersions nevű magazin szerkesztője. Mostanában *Linux System Administration: A User's Guide* című könyvén dolgozik, mely az Addison Wesley Longman kiadónál jelenik majd meg. Honlapján ➲ http://www.salmar.com/marcel/ pedig rengeteg érdekességre bukkanhatunk.

Új távlatok: az otthoni hálózat

Milyen eszközöket tudunk majd otthoni hálózatunkba kötni?

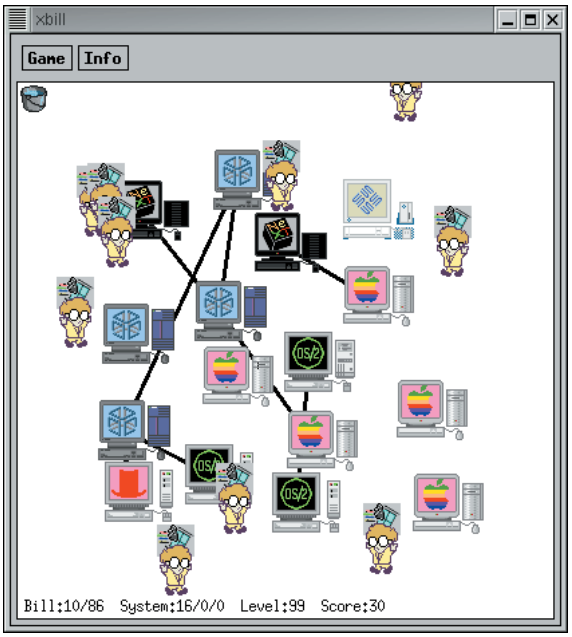
Manapság otthon még csak a számítástechnika megszállottjainál akad helyi hálózat, de olyan új termékek vannak készülőben, melyek mindennapivá tehetik a lakáson belüli LAN-t. Az új, gyakran Linux-alapú eszközök iránti érdeklődés robbanásszerűen nőni fog a jómódú – de nem feltétlenül műszaki beállítottságú – háztartásokban. Mint minden tudományos forradalomban, a különböző résztvevőknek megvan a saját nézetük a tudomány fejlődésének irányáról: ez pedig a leendő fogyasztók számára ellentmondásos tájékoztatást idéz elő.

Nemrég több fontos forgalmazó közösen létrehozta az Otthoni Internet Szövetséget (Home Internet Alliance). Az alapító tagok között nagy nevek is találhatók: Cisco Systems, Sun Microsystems, 3Com. Ismert fogyasztóközpontú cégek is képviselik magukat: a Best Buy, a CompUSA, a Panasonic vagy a Sears. Csatlakoztak lapkagyártók is, például a Texas Instruments és a Motorola.

Gyártók és forgalmazók együtt dolgoznak a gondok megoldásán, és mindegyiküknek megvan a maga részfeladata: széles sávú hálózati kapcsolat megosztása vagy zenei hálózat létrehozása (lásd a Linuxvilág 2000. novemberi számának 63. oldalán). Először, minden kétséget kizáróan, az egy feladatra alkalmas, egy forgalmazó által kifejlesztett hálózati megoldások jelennek majd meg, de amint az otthoni hálózatok elterjednek, számos új felhasználási lehetőség kerül előtérbe.

A TiVo DVR (digitális videofelfevő) kiválóan alkalmas arra, hogy tévéműsorokat merevlemezre vegyükn, ott tároljuk, majd később onnan megnézzük. Nincs azonban egyszerű megoldás arra, hogy két külön szobában elhelyezett televízión nézzünk műsorokat. Az otthoni hálózat lehetővé tenné, hogy egy DVR készülék több tévét is ellásson képfolyammal.

A távolabbi jövőben, szerintem, az otthoni hálózat sok tekintetben hasonlítani fog a mai irodai hálózatra, egy kiszolgálóhoz csatlakozik majd több ügyfél. A kiszolgáló kettős feladatot fog ellátni: otthoni átjáró (residential gateway) és tárolókiszolgáló (storage server) lesz.



Számos forgalmazó hirdeti tapasztalatát az otthoni átjárók terén, de napjainkban kevés ilyen termék kapható a piacon. Ezek az eszközök hasonlítanak a jól ismert WAN átjárókra, feladatuk az otthoni LAN és az Internet közötti kapcsolat megvalósítása, jellemző esetben széles sávú csatlakozáson át, ilyen lehet a bérelt vonal vagy a DSL.

A szokványos széles sávú modem és az otthoni átjáró között az a különbség, hogy az utóbbinak van egy LAN csatlakozási felülete, amin keresztül az adatokat az ügyfelek számára továbbítja.

A tárolókiszolgáló processzorból, hálózati csatlakozóból és egy vagy több merevlemezből áll, ezek szolgáltatják a tárolóterületet a lakásban található eszközök számára. Ezen a kiszolgálón tárolhatjuk majd a lakásban lévő PC-k merevlemezeinek mentését, digitális fényképeinket, zenét és videofelvételeket. Ez utóbbiakat bármikor lejátszhatjuk valamelyik hálózatba kapcsolt ügyfélen. Megfelelően programozhatónak kell lennie ahhoz, hogy a TiVóhoz hasonló alkalmazások fussanak rajta, amelyek ha kell, emberi beavatkozás nélkül elérhetnek bármit a széles sávú kapcsolaton keresztül az Internetről.

Az ügyféleszközök lehetnek PC-k, maces és linuxos rendszerek, ezek az otthoni átjárón keresztül érik el az Internetet, és a tárolókiszolgálót használják fontos adataik mentésére vagy azok megosztására (a tárolókiszolgáló eszményi esetben RAID-et használ, így nagymértékben csökkenthető a géphibából adódó adatvesztés). Fényképeket és zenefájlokat készíthetünk a PC-n vagy letölthetjük ezeket az Internetről, majd menthetjük a fájlokat a tárolókiszolgálóra.

Ha a fényképünk a kiszolgálón van, megnézhetjük a házban található bármelyik tévékészülékről vagy monitorról, ehhez csak egy egyszerű képügyfélre van szükség. A képügyfélnek csak processzora, hálózati csatlakozása és videokimenete van. Letölti a képet – esetleg mozgóképfolyamot – a kiszolgálóról, kicsomagolja és megjeleníti. Hasonló módon játszhat le hangfájlt a kiszolgálóról az erősítőhöz és a hangfalakhoz csatlakoztatott hangügyfél.

A hangügyfél anyagköltsége megegyezik a képügyfélével, tömeggyártás esetén pedig a felhasználói árnak húsz-harmincezer forint alatt kell maradnia, így megéri majd többet vásárolni egy-egy háztartásba. Más lehetséges hálózati ügyfelek az internetes készülékek, a nem PC-alapú eszközök. Ezekben van levelezőprogram és webböngésző is. Játékkonzolok csatlakozhatnak a hálózatra, így lehetőség nyílik a többszereplős játékokra. Még az olyan háztartási készülékeket is érdemes a hálózatra csatlakoztatni, mint a hűtőgép vagy a lakásban elhelyezett termosztátok, hiszen ezek is adatokat oszthatnak meg, és ezáltal hatékonyabban működhetnek.

Ez a felépítés a tárolókiszolgálóra hárítja az adattárolás és a számítás feladatát, így a lemezterület könnyen megosztható és bővíthető belső vagy külső meghajtókkal – „Édesem, hazafelé vegyél fel még egy pár gigabájtot!”. Eleinte az adott célra kifejlesztett termékeknek – mint a TiVo vagy az AudioReQuest – saját merevlemezük lesz, később a helyükbe olcsó, vékony ügyfelek lépnek.

A Linuxnak jó esélye van arra, hogy kulcsfontosságú szerepet játsszon ebben az egyre növekvő fogyasztói hálózatban. Ez különösen a tárolókiszolgálóra igaz: megbízható, de olcsó operációs

rendszer szükséges hozzá, beépített hálózatkezeléssel. Mivel a tárolókiszolgálónak valószínűleg bonyolultabb alkalmazásokat is kell futtatnia – ilyenek például a hang-, a kép- és a webtartalmat egyaránt kezelő alkalmazások –, olyan rendszerre van szükség, amihez kiforrott fejlesztői környezet és nyílt szabványok is rendelkezésünkre állnak. A Linux minden szempontból tökéletes választás.

Az otthoni átjáró lehet egy egyszerű hálózati eszköz, de valószínű, hogy futni fog rajta valamilyen tűzfal és esetleg más programok is. A Linux itt is szóba jöhet. Az ügyféleszközök egy részén futhat beágyazott Linux, ebben az esetben az a fontos, hogy a lehető legkisebb legyen az operációs rendszer által elfoglalt memória mennyisége, így az eszköz olcsó maradhat. A Linux mindenképpen jó választásnak tűnik a webböngészőt tartalmazó internetes készülékekhez.

Kérdéses még az otthoni hálózatok megvalósítása, és a fizikai összeköttetés típusa. Jelenleg a legolcsóbb megoldás a ház létező telefonkábeleinek felhasználása. Ez a módszer hosszú távon nem biztos, hogy beválik, hiszen a háztartásokban gyakran nincs lehetőség több telefoncsatlakozásra, főleg az Egyesült Államokon kívül. A hosszú távú megoldást valószínűleg a drótnélküli megoldások jelentik majd (ilyen például a 802.11). A 802.11b szabvány elegendő sávszélességet teremt több hangfolyam számára, azonban csak egy képfolyamot támogat. Meg kell várnunk az előkészületben lévő 802.11a szabványt, ugyanis ez lehetővé teszi majd több párhuzamos képfolyam átvitelét is.

Rövid távon ez a két átviteli közeg, valamint az áramhálózaton folyó adattovábbítás fog burjánzani, és átmeneti zavart okozni a piacon. A megoldás mind a két vagy akár három fizikai kialakítás támogatása az otthoni átjáróban, lehetőséget teremtve ezzel arra, hogy az otthonokban a különböző hálózati megvalósítást alkalmazó ügyfelek békésen megférjenek egymás mellett. Ez növeli ugyan az otthoni átjáró árát, de kevesebb fejtörést okoz a fogyasztóknak.

Ha ezekre a gondokra kielégítő megoldásokat találnak, és a költségek csökkennek, az otthoni hálózatok virágzásnak indulnak. Tanulmányok azt mutatják, hogy 2004-re az Egyesült Államok háztartásainak egyharmadában lesz széles sávú internetelérés. Ezek a háztartások lesznek az otthoni hálózat úttörői. Ahogyan a zene, a fényképeink és a videofelvételeink tömörített digitális fájlokká alakulnak át, az otthoni hálózat előnyei egyre inkább hangsúlyt kapnak. Az otthoni hálózatok elterjedése új lehetőségeket teremt a Linux számára a kiszolgálók (tárolás és hálózati átjárók) és a beágyazott rendszerek területén egyaránt.

Linley Gwenapp
(linley@linleygroup.com) a The Linley Group szakmai elemző cég alapítója és vezetője. A cég honlapja ➲ http://www.linleygroup.com címen érhető el.

Soldier of Fortune for Linux

Esettanulmány egy vérgőzös játék belvilágáról.

A Soldier of Fortune dobozára pillantva először szinte sokkolt a figyelmeztetések tömege, melyek világosan tudtomra adták: a játékban erőszakos jeleneteket, illetve animált, kiömlő vért láthatunk. Az alacsony erőszakszintű telepítési lehetőség is rögtön a szemembe ötlött. A CD tokjának belseje szintén alaposan ki volt tapétázva ilyen figyelmeztetésekkel.

A kézikönyvet ezúttal kihagytam, telepítettem a programot, elindítottam, és máris a főmenüben találtam magam. Itt még több figyelmeztetéssel szembesültem – számuk ekkorra már vagy kilencre nőtt –, többek közt egy gördíthető szöveges ablakkal a főmenü alján, mely tájékoztatott a játék vérengzős mivoltáról, illetve arról, hogy a szülői felügyelet eszközeit melyik menüpontban lehet elérni. A figyelmeztetések végső hatása pontosan ellenkezője lett annak, ami az eredeti szándék lehetett: ellenállhatatlan késztetést éreztem arra, hogy megnézzem, vajon mi a csuda miatt aggódnak ennyire ezek az emberek?

Elindítottam az első szintet. Rövid bevezető film után máris a sűrűjébe kerültem, feladatom egy csarnokba vezetett, ahol néhány terroristát kellett elkapni, és kiszabadítani a túszokat. Alig tettem három lépést a csarnokban, máris előbukkant az első rosszfiú, és egyenesen nekem szegezte pisztolyát. Puskámmal – próbálva jó gyerekként viselkedni – a lábára céloztam. Bumm! A lövés teljesen leszakította az ürge lábát, a letépett részt pedig jó néhány méterre áldozatom mögé vetette, nemcsak egy vértől csatakos combcsont és az ütőérből sugárban kiömlő vérzuhatag látványával jutalmazva lövésemet, hanem leendő orgyilkosom is döbbenetesen élethű halálsikolyt hallatott, majd vérző csonkját markolva lassan a padlóra csúszott, néhány pillanatig torokhangon hörgött, majd halálfélelmi tusája – és ezzel virtuális élete is – véget ért. Kissé hátracsúsztam a billentyűzettől, önkéntelenül egy „Hűha!" felkiáltással értékeltem a látottakat. Rövid szünet után lassan elvigyorodtam: „Ez érdekesnek tűnik".

Előnyök

- Lenyűgözően élethű grafika, összesen huszonhat anatómiailag pontos vérzóna
- Érdekes szintek
- Többjátékos mód támogatása
- Még egy iskolabuszos autóversenynél is érdekesebb

Hátrányok

- Gyenge történet

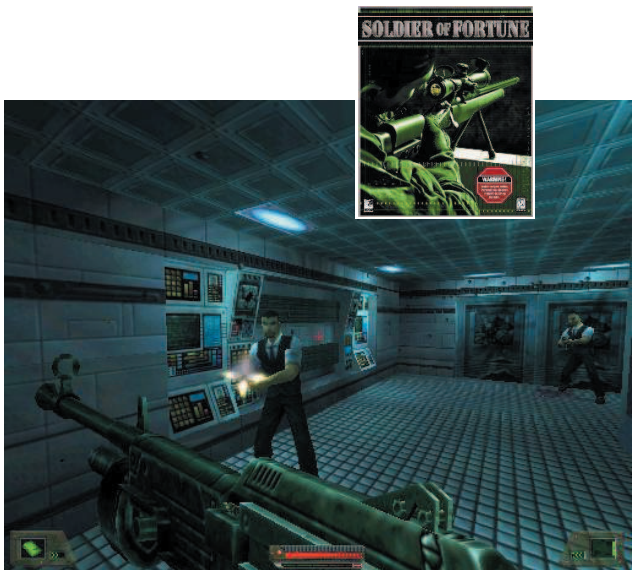
A történet

A Soldier of Fortune története valójában a Rainbow Six történetének egy elég gyenge átdolgozása. John Mullins szerepét játszod, a legnagyobb kaliberű (nem kell rosszra gondolni) zsoldosok egyikét, aki a terroristaellenes világ sötétebb felén munkálkodik. Egy olyan csapatban játszol, amit a jófiúk csak akkor hívnak, amikor valakinek olcsón kell elvégeznie a mocskos melót. A rosszfiúk csapata, elvakult terroristák egy csoportja a tömegpusztításra esküdött fel, és sikerült megszereznie néhány atombombát. A Soldier of Fortune-ban küldetések egész sorában vehetsz részt a csoportosulás, illetve vezetőjük ellen, feladatod a bombák visszaszerzése vagy megsemmisítése, mielőtt azokat a világ ellen fordíthatnák. Minden egyes küldetés bizonyos összegű pénzt ér, ezt a sikeres befejezés esetén lehet megkapni, illetve valamilyen módon a következő küldetésbe vezet át, például tovább gombolyítva a történet fonalát, és meghatározva a következő helyszínt.

Sajnos, a fenti rövid leírás hiába sejteti azt, hogy a játékban gördülékenyen követik egymást az események, megadott célkitűzések vannak, meghatározott részleteket kell megismerni és egymás mellé illeszteni, megadott mennyiségű pénzt kell megkeresni és összegyűjteni – a valóság más. Maga a történet körülbelül annyira elmélyült, mint egy teleregénynél, és csak időként találkozunk egymáshoz kapcsolódó részletekkel a történetben. Még szerencse, hogy ilyesmire nincs is szükség. Ez egyes szám első személyben játszódó lövöldözős játék, amitől nem is annyira történetet várunk, inkább tiszta célpontot a következő rosszfiúhoz, és valamilyen általános érzést, hogy fáradozásunk valamikor valamilyen eredményre vezet.

Beismerem, mindvégig vártam a történetben valamilyen fordulatot, az egész mesét a végén még érdekesebbé tévő hirtelen, megdöbbentő irányváltást. Amikor az előre megjósolható időpontban a fordulat megérkezett, csak csalódást hozott, mivel a történetben semmilyen előrehaladás nem jelezte közeledtét. Az, hogy mindezért a gyilkolásért pénzt kapsz, csak nagyon mellékes dologként került be a játékba, és az sem számít, hogy ténylegesen mit végeztél, hiszen semmit sem látsz abból a pénzből, amit rendes esetben kézhez kellene kapnod. Elkezded a küldetést, győzöl és megkapod a pénzt, esetleg addig játszod a küldetést, míg végül sikerül, és megkapod a pénzt, majd továbbléphetsz a következőre.

Ismét csak azt tudom mondani, hogy nem baj, ha egyszer elkezdesz játszani, a játék eléggé leköt ahhoz, hogy ne foglalkozz ilyesmivel.



A társaid: „Sólyom" Parsons és Sam Gladstone

A játék

A Soldier of Fortune – ha mást nem is mondhatunk róla – rendkívül részletes a fröccsenő vér és a kifordult belek látványában. A Raven cég saját fejlesztésű leképezési rendszerét használja, mely a GHOUL (vámpír) névre hallgat, ennek „köszönhetően” tudnak ennyi vérrel sokkolni bennünket.
A GHOUL – számos továbbfejlesztést tartalmaz a megvilágítás, a mintázatok, a hálózatkezelés és a modellezés területén. Segítségével a Soldier of Fortune programozói egyedi karaktereket hozhattak létre, amelyek huszonhat „vérterület” alapján különbözően válaszolnak a fegyverek találataira. Lőj lábon valakit, egy ideig fél lábon ugrál körbe-körbe, fájdalmasan jajgatva. Lőj torkon valakit, odakap a nyakához, gurgulázik, majd a padlóra rogy. A karakterek nem mindig halnak meg. Az egyik szobában vagy öt percig járkáltam körbe, miközben valami halk, szörcsögő hang forrását próbáltam megtalálni, végül csak az egyik földön fekvő áldozatomra bukkantam, aki halkan gurgulázott a torkán lévő lyukon keresztül. Hát ez szép. Az ellenség kezéből a fegyvert is kilőheted, ha elég jól célzol.
El kell oszlatnom a szülők aggályait, a Soldier of Fortune-ban minden tényleg ronda dolgot szabályozni lehet a szülői felügyelettel. Így le lehet tiltani és jelszóval lehet védeni a sebesülések látványát (Damage Skins), a vért (Blood), az animált haláltusákat (Death Animations), a leszakított végtagokat (Dismembered Limbs) – ez a kedvencem – és az úgynevezett felnőtt mintázatokat. Ezek elsősorban a feliratokban megjelenő, inkább felnőtteknek szánt szövegeket jelentik.
Bár a véres részletek nagyon jók, azok a játékosok, akik Rainbow Six típusú élethű harcra számítanak, valószínűleg csalódni fognak, mivel a játék többi része inkább Quake stílusú. A Soldier of Fortune jelenleg a Quake II motorjának egy kissé módosított változatán fut, így néha érezhetjük azt a régi szögletességet a játékban. A Raven cég viszont nagyon jó munkát végzett, valóban érdekes szintek készültek, köztük néhány különösen émelyítő és tekervényes. A szintek túlnyomó része tele van a lövöldözős játékokból jól megszokott elemekkel, például egy nagy piros gomb megnyomására nyíló ajtókkal, elsősegélyládákkal, vagy a szellőztetőrendszer megfelelő módon elhelyezett búvóhelyeivel. A játék hangulata kicsit a Duke Nukemre emlékeztet, például láthatod magad, ahogy ujjadon fegyvert pörgetve sétálsz tova, a gépi szereplők harci értelme pedig még közepes nehézségi szinten is alulmúlja egy hisztiző kétévesét. Be kell azonban vallanunk, együttesen éppen ezek a tényezők teszik olyan hihetetlenül élvezetessé a játékot.

Többjátékos mód

Úgy vélem, a többjátékos környezet remek szórakozást nyújt. A Soldier of Fortune többjátékos módja szakít a Quake hagyományaival, együttműködésre ugyanis nincs lehetőség. A kedvencek azonban nem hiányozhatnak. Találunk „Deathmatch” és „Capture the Flag” módot, illetve megjelent néhány új lehetőség is. Ilyen például az „Arsenal”, melynél véletlenszerűen kapsz egy-egy fegyvert a számítógéptől, és ezek mindegyikével meg kell ölnöd valakit. Vagy az „Assassin”, ahol egy megadott játékost kell megölnöd, miközben meg kell védened magadat a rád vadászó játékosoktól, vagy a „Realistic”, ahol a fegyverek élethűen sebzik meg a lábadat vagy kezedet, az elsősegélycsomagok pedig megadott testrészeket gyógyítanak, például a lábadat, hogy újra tudj futni, vagy a karodat, hogy újra célozhass stb. Amint a Quake, úgy a SOF többjátékos módja is eltorzít időt és teret, az eredetileg unalmasnak és hosszúnak ígérkező estéket többjátékos pályák és megszámlálhatatlan virtuális gyilkosságok rövidke sorozatává változtatja.

Behatolás az ellenség titkos kutatólaboratóriumába

Telepítés

A Loki cég – mint megszoktuk – remek és tökéletesen észrevétlen munkát végzett, amikor átültette Linuxra a Sodier of Fortune-t. A SOF 3D gyorsítást támogató videokártyát igényel, amelynek szolgáltatásait Mesa/OpenGL segítségével használja ki. A Loki által megadott, támogatott lapkakészletek: a Voodoo Banshee, Voodoo2, Voodoo3, G200, G400, Riva TNT és TNT2, illetve a GeForce 256. A program telepítésével egy általános VA Linux Systems 6.2.3 alatt (RedHat 6.2-alapú rendszer) Matrox G400-as kártyával, valamint a saját fapados Debian rendszeremmel XFree4 CVS fával és GeForce GTS kártyával sem volt semmilyen lényeges gondom. A telepítés egy kicsit sok helyet igényel (a teljes telepítés az egyetlen választási lehetőség, és 700 megabájtot igényel), viszont karakteres és grafikus módban egyaránt működik. A gépigény: 2.1-es változatú glibc, 2.2.x változatú rendszermag, OSS-megfelelő hangkártya és illesztőprogram. Ha a többjátékos módot szeretnéd használni, nyilvánvalóan szükséged lesz egy hálózati kártyára, illetve engedélyezned kell az IPX vagy a TCP/IP-támogatást a rendszermagban. A játék támogatja botkormány használatát. A gépre vonatkozó követelmények a Loki szerint 64 MB memória és Pentium II processzor. Én 128 megabájttal használtam egy 400 MHz-es Pentium II-esen, ott szépen futott.

Összegzés

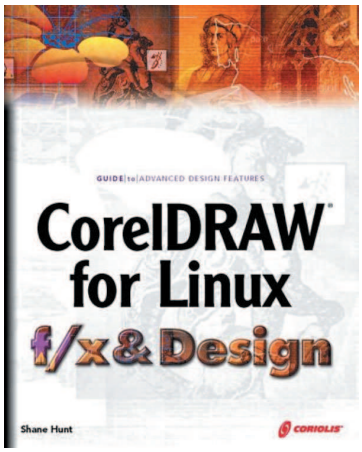
Azt kell mondanom, nagyon bírom a SOF-et. Öröm vele játszani. Jellegzetes program. Nem fél attól, hogy arcon csap. Azok a dolgok, amelyeket leginkább hiányoltam belőle teszik egyben ilyen nagyszerű lövöldözős játékká. Mondtam már, hogy mekkora buli? Ha valami olyat keresel, ami a Quake és a Rainbow Six között van félúton, valószínűleg neked készült.

Adatok

Gyártó: Loki Entertainment Software
E-mail: sales@lokigames.com
URL: http://www.lokigames.com
Ár: 45,95 dollár

J. Neil Doane (caine@valinux.com) havatásos szolgáltatásmérnök a VA Linux Systemsnél. A megfeszített munkavégzés, az alkalomszerű gépkarbantartás és a videojátékok mellett szabad idejében pilóta, gitáros és kezdő hódeszkás.

Shane Hunt: CorelDRAW for Linux: f/x & Design



Nem vitás, hogy a legtöbb számítógépes könyvet hosszabb ideig tart elolvasni, mint a program használatába beletanulni. És ami még ennél is rosszabb, hogy e könyvek általában annyira lefárasztják az olvasót, hogy teljesen elmegy a kedve az új programmal való ismerkedéstől. Szerencsére, a Coriolis Group kiadónál tavaly megjelent angol nyelvű CorelDRAW for Linux: f/x & Design esetében ez másképpen van. Hunt könyve a legtöbb segédkönyvvel ellentétben nem csupán a programról szól. Célja az, hogy a felhasználó maga ismerje meg a programot. Saját tapasztalatom alapján bátran kijelenthetem: a szellemes, világosan megfogalmazott fejezetek elolvasása után egyre jobban kiéhezve vágunk bele a következőbe.

A címben ott van a lényeg: a szerző az egész könyvet a linuxos CorelDRAW-val megvalósítható különleges hatásoknak szentelte. Az olyan egyértelmű témakörökkel, mint a nyomtatás vagy az oldalbeállítás, egyáltalán nem foglalkozik, ugyanis ez a könyv az alkotásról szól.

A kiadó nemes egyszerűséggel a Creative Professionals Press („A profi alkotók könyvtára”) nevet adta a sorozatának, melynek hírnevét Hunt műve csak tovább öregbíti.

A könyv 18 fejezetből áll, mindegyikben négy-öt témát tárgyal, s ezek akár önállóan is használhatók. A szerző a kezdők nehézségeit is figyelembe véve egyszerűbb feladatokkal indít, a könyv végére pedig már az első néhány példánál jóval bonyolultabb feladatok sem jelentenek gondot a tanulni vágyók számára. Nézzünk most egy példát a 2. fejezetből.

Shane hozzáállását jól jellemzi, hogy a munkát egy Tux pingvin megtervezésével kezdi. A 17 lépés elvégzése után el is készül kedvenc kabalánk. A dolog nagyszerűsége, hogy a tervezést teljesen az alapokról (egy üres képernyőről) indulva végezhetjük.

Ez így is van rendjén. A fejet, a testet, a szárnyakat, a lábakat és minden testrészt az olvasó maga készítheti el.

Hogyan történik mindez? A válaszból a mű egész szerkezetére és stílusára is következtethetünk, és arra is választ kapunk, hogy pontosan mit várhatunk el a könyvtől. Hunt úgy ír, mint ahogy egy autósiskolában dolgozó oktató tanítja a leendő vezetőket. Nem magyarázza el a váltó áttételeinek működését, mielőtt a vezetőülésbe engedné ülni a tanulókat. Inkább azt mondja: „Figyelj, ide beülsz, ez itt a kormány, az ott a váltó, elöl meg az ablaktörlő. Mehetünk.” „Vezetés” közben pedig mindent elmesél, amit tudnunk kell ahhoz, hogy Forma 1-es versenyzővé válhassunk. A könyvet éppen ez teszi házi könyvtárunk egyik legnagyobb kincsévé: azokhoz szól, akiknek a programmal gyorsan eredményt felmutatva kell dolgozniuk.

A 2. fejezetben maga Hunt összegzi ezt: „Nem kell csöpögnöd a tehetségtől ahhoz, hogy jó művész legyél. Elég, ha a rendelkezésedre álló eszközök használatát megtanulod, és elég türelmes vagy ahhoz, hogy megküzdj a felbukkanó nehézségekkel.”

Egy vektoralapú rajzprogram megtanulását egyszerű formákkal kell kezdenünk: vonalakkal, körökkel, négyzetekkel. Így jár el Shane is. Az utat egyszerű feladatokkal nyitja meg (mint amilyen a fent említett pingvin), majd a megfelelő gyakorlat elsajátítása után a könyvben ismertetett példáknál jóval összetettebb munkák elvégzése sem jelent majd gondot.

Hogy pontosan mennyit gyarapodhat tudásunk egy-egy gyakorlat végigkövetése után? Vegyük például a pingvinünket.

Ha ezt elkészítjük, akkor máris egy tucatnyi lehetőséget ismertünk meg, melyekkel már egyéni utakat is könnyedén bejárhatunk.

A bemutatott példákat látni kell, de az alábbi rövid lista étvágygerjesztőnek nem rossz. A CorelDRAW for Linux működésének ismertetését követően az alakzatok szerkesztéséről, az átlátszóság és az árnyékok beállításáról, az átmenetek és a körvonalhatások készítéséről, sőt, a bitképes ábrák beillesztéséről és módosításáról is szó esik. A mindennapi munkában jól használható fogásokat is elleshetünk, például a reklámcsíkok, rajzfilmfigurák, láncok, szögesdrótok (!) és még ki tudja mennyi minden készítésének rejtelmeit. A fentiek természetesen nem merítik ki a könyv teljes terjedelmét.

A könyv tetszetős szerkezete jól illeszkedik a tanulás menetéhez: a rengeteg képnek köszönhetően azonnal tudjuk, hogy a szövegben a program mely részéről van szó, és a képek melletti szövegből számos hasznos tippet és részletet ismerhetünk meg. Természetesen saját megjegyzéseink beírására is maradt elég hely.

Shane is a szájbarágós „most elmondom, hogy mit fogok elmondani, aztán elmondom, végül pedig elmondom, hogy mit mondtam el” stílusban értekezik, és ez még önállóbbá teszi az egyébként is külön-külön használható fejezeteket.

Bár Shane a CorelDRAW for Linuxról komolyan ír, sok példa mégis derültséget kelt az olvasókban. Akinek a CorelDRAW munkaeszköz, az lehet, hogy komolytalannak tartja a viccelődést. Mondjuk, egy függőhidat kell megtervezni, s ehhez a könyv tetoválással és szögesdrót készítésével foglalkozó fejezeteit kell átböngészni. Sebaj, tekintse ezt mindenki az új lehetőségek felé történő elmozdulásnak.

A könyvön tehát végigvonul Shane egyedi stílusa: imád a vidám és kedves dolgokkal szöszmötölni. De amiért igazán jó vétel a könyv, hogy emellett tökéletesen elmagyaráz mindent, amit erről a nem kevéssé összetett programról tudnunk kell.

Rengeteg számítástechnikai szakkönyvet veszek, és ha egy újat leemelek a polcról, mindig ugyanazt kérdezem magamtól: mit ad nekem majd ez a könyv? Sokszor a válasz sajnos az, hogy semmit. Shane Hunt CorelDRAW for Linux: f/x & Design című könyve azonban magasra tette a mércét. Minden egyes oldalon valami újat ismerhetünk meg a lehetőségekből, melyek a CorelDRAW-t a vektoros rajzprogramok királyává tették.

Clifford Anderson

Beszerezhető: Kiskapu mintabolt,
1081 Budapest, Népszínház u. 29.
Telefon: 303-9119,
E-mail: gyorgy@kiskapu.hu

Kell-e rendszerleíró adatbázis?

Úgy látom, egy csomó kérés arra irányul, hogy a dolgokat Microsoft-módra végezzük, bár ennek nem sok értelme van. Néhányan a frissen megtértek közül valamilyen okból el akarnak ugyan szakadni a Microsofttól, ennek ellenére azt szeretnék, ha minden úgy maradna, ahogy már megszokták. Mostanában hallom, hogy Linux Rendszerleíró Adatbázist követelnek. A fenébe is, *Charlie Brown*! Ez az egyik olyan eleme a Windowsnak, amely több bajt okoz, mint ahányat megold. Miért vennénk ki az egyszerű, könnyen olvasható és szerkeszthető (rendben, nem mindig könnyen olvasható) fájlokat a /etc könyvtárból és gyömöszölnénk bele egyetlen hatalmas fájlba, amelyet csak különleges segédprogramokkal szerkeszthetünk? Élénken emlékszem arra az ígéretre, miszerint a Windows rendszerleíró adatbázisát sohasem kell majd szerkeszteni, csak a rendszer fogja használni. A mai napig nem láttam olyan windowsos rendszert, amelynél ne kellett volna kézzel belenyúlni a rendszerleíró adatbázisba. Biztos vagyok benne, hogy a /etc könyvtárban levő fájlok kezelését sokkal egyszerűbb megtanulni, mint a rendszerleíróét. Mielőtt egyesek változásokért kiáltanak, miért nem vizsgálják meg, hogy a javasolt változtatások jobbá teszik-e a rendszert? A Linuxnak fejlődnie kell, de egy linuxos rendszerleíró adatbázist én nem neveznék kívánatos fejlődési iránynak.

SDMS

A Simple Document Management System pontosan az, aminek mutatja magát: egyszerű. Könnyű telepíteni és pofonegyszerű használni. Láttam már más dokumentumkezelési rendszereket, és bár ez nem tud néhány olyan dolgot, amit a többiek, de kiemelkedően jól működik. Az a legjobb benne, hogy böngészőn keresztül működik. A dokumentum akkor is kezelhető, elérhető, ha a világ másik felén van. Természetesen, ha a dokumentum érzékeny adatokat tartalmaz, biztonságos webkiszolgálót kell használni. A program ACL-t használ, ennek segítségével meghatározható, hogy ki mit tehet a dokumentumokkal. Szükséges: MySQL, MySQL-t és PHP4-et támogató webkiszolgáló, böngésző.
➲ http://sdms.cafuego.net/

djpim

A Daryl Jonel Personal Information Manager (djpim) egy jól átgondolt webalapú adatkezelő segédprogram. Egy maroknyi dologra használható, többek között adott részlegben futó tervezetek követésére. Egy komoly naptár azonban nagyon hiányzik belőle. Két helyen is előhívható ugyan egy kis naptár, ez azért nem ugyanaz. Ha szükségünk van egy programra, amely rendszeresen figyelmeztet feladatainkra, akkor használjuk ezt az ízléses és jól szervezett programot! Szükséges: MySQL, MySQL-t és PHP-t támogató webkiszolgáló, böngésző.
➲ http://www.tcomeng.com/djpim

IPTS

Ha embereink a város különböző pontjain szétszórva dolgoznak, vagy ami még rosszabb, az országban, akkor az Internal People Tracking Systemmel nyomon követhetjük őket. A weboldalon megjelenő mezőkhöz könnyen hozzá lehet adni újakat, a meglévők kényelmesen módosíthatók. Az IPTS használatának legnehezebb része minden bizonnyal az lesz, hogy az alkalmazottakat rászoktassuk a használatára. Szükséges: MySQL, MySQL-t és PHP-t támogató webkiszolgáló, böngésző, valamint a HTML::Template, CGI_Lite, DBI és DBD::mysql Perl-modulok.
➲ http://dev.wslogic.com/~anderson/ipts/

pad

Ha nagyon bizalmas adatokkal dolgozunk, ezzel a segédprogrammal feldarabolhatjuk az adatfájlokat több, titkosított fájlra. A titkosításhoz és a visszafejtéshez nem kell jelszó, viszont csak az juthat hozzá az eredeti fájlhoz, aki mindegyik darabot birtokolja. Így, ha a darabokat szétszórjuk különböző gépekre, akkor csak az juthat hozzá az adatokhoz, aki mindegyik géphez rendelkezik eléréssel. Az egyetlen hátránya, hogy a szükséges tárolóhelyigény így legalább kétszeresére növekszik. Szükséges: libcrypt, libm, libc, libdl.
➲ http://www.lammah.com/pad/

slmon

Pont ez hiányzott, egy újabb rendszerfigyelő eszköz... De várjunk csak, ez egy kicsit más, érdemes rászánni néhány pillanatot az ismerkedésre. Rengeteg különböző üzemmóddal dicsekedhet, sőt támogatja a többprocesszoros rendszereket. Az egyik nézetben egy ábrán szemlélhető minden processzor és a hozzá tartozó memória, egy másik nézetben egy hisztogramon a processzorok terhelése látható egy adott időpillanatban. Szükséges: libslang, glibc, libdl, libm.
➲ http://www.m00se.hsn.pl/slmon/

GtkPortFolio

Figyelem, tőzsdecápák! A GtkPortFolio egy nagyon jól megírt árfolyamletöltő. Használata egyszerű. Hozzá akarunk adni saját listánkhoz egy rövidítést, nem kell fájlokat szerkesztenünk, elég a rövidítést az AddSymbol mezőbe beírni, a program megjegyzi. Ha nem jut eszünkbe egy rövidítés, írjuk be a vállalat nevét a Search mezőbe, a program megnyitja a Nescape-et, és megmutatja a megtalált rövidítés(eke)t. Ez olyan egyszerű, amilyennek látszik. Szükséges: Gtk, Gtk:Gdk:ImlibImage, Finance::Quote, Finance:YahooChart, Time:localtime, LWP::Simple.
➲ http://www.centercube.com/GtkPortFolio/

nscache

Ha valaha használtuk a Netscape-et, tudjuk, hogy a gyorstárazáshoz elfoglalt lemezterület milyen nagyra nőhet. De lehet, hogy nem tudjuk? Hadd mondjam el: óriásira! E segédprogram segítségével GTK-s felületen keresztül böngészhetünk a Netscape gyorstárában. Két nézet közül lehet választani: a faszerkezet vagy a rendezett nézet közül. Ez utóbbiban az elemeket rendezhetjük URL (alapértelmezett), méret, utolsó használat ideje vagy mime-típus szerint. Mindkét nézetben le lehet törölni bármelyik fájlt. Itt az alkalom, hogy töröljünk néhány nagy méretű fájlt. Ehhez kattintsunk a rendezett nézetre, rendezzük a listát méret szerint, majd a lista alján kattintsunk jobb egérgombbal a legnagyobb méretű elemeken, és töröljük le őket. Szükséges: libdb, libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, glibc.
➲ http://nscache.sourceforge.net/

Viszlát a jövő hónapban.

David A. Bandel (dbandel@panamix.com) jelenleg Panamában él, ahol Linux/Unix szaktanácsadóként tevékenykedik.