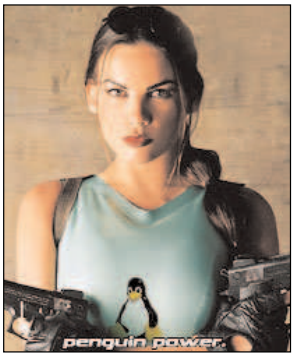
Változások – a kobzost már megettük...

Azt mondják, a változás az élet jele. Mi is változunk, alakulunk. Mint ahogy az a lapról már kívülről is látszik, valami történt. Egy kicsit vaskosabb, egy kicsit színesebb. És két hónap is szerepel az elején: február–március. Egyik kedves olvasónk kérdezte kétségbeesve, hogy ugye ez nem a lassú elhalás első jele? Örömmel jelentem be, hogy nem erről van szó, a lap – Olvasóinknak köszönhetően – él és virul. Az összevonás valójában egy kis csalás. Többen panaszkodtak, hogy miért jelenünk meg mindig a hónap végén. Miért nem a legelején, ahogy azt más lapok is teszik? Mivel nem szerettünk volna lapszámot kihagyni, összegyűjtöttünk másfél számnyi anyagot, és egyszerre két hónapot írtunk a borítóra. Így kicsit közelebb kerültünk a hónap elejéhez. Remélem. Aki már vett részt lapszerkesztésben vagy programírásban, az tudja, hogy is kell számolni: „Vedd a munkát, tervezd meg, mennyi idő kell hozzá, szorozd meg kettővel, adj hozzá még egy kicsit, és megkapod azt a határidőt, amiből először csúszol ki."



Szerencsére azért legalább látható a fejlődés. Igen, igen, az örök vita. Várjunk sokat, hogy elkészüljön, vagy már menet közben lássuk a terméket. Aki fejlesztett nagyobb rendszert, biztos emlékszik, hogy amikor végre sikerült a második részfolyamatot is lezárni, jött a megrendelő, és kért még egy „irinyó-pirinyó" kis változtatást, ami miatt az egész kódolást kezdhettük elölről. No, így vagyunk most mi a honlappal is. Már-már körvonalazódni látszott a kameloti vár, és végre lelkesen integettünk a bemutatón, hogy „De jó! Már van hírlevél!", „Naponta friss hírek!", mire Robin kobzosa közönyösen legyintett: „Á, az csak egy makett…"

A kobzost azóta megettük, és mikor a lap napvilágot lát, már minden működik. Ahogy ez a webes fejlesztéseknél is lenni szokott, biztos, hogy maradnak még hiányosságok, de igyekszünk. És van sok tervünk is, melyeket meg szeretnénk valósítani. Például a honlapra sok érdekes címet szándékozunk gyűjteni. Olyan címeket, amik érdekelhetik a linuxosokat. És remélem, hogy mindenkinek tetszeni fog a „Címtáram" szolgáltatás is. Ide mindenki felveheti saját kedvenc címeit, így nem kell: *a)* saját oldalt készíteni, *b)* beküldeni a kedvenc szexoldalak címét, hogy rakjuk fel a főoldalra, *c)* megvárni, míg Gyula végre hajlandó kirakni azokat a címtárba.

De vissza komolyabb vizekre. A lap tehát nem hal el, a tervek szerint tovább fejlődik. Hogy merre, az mindannyiunkon múlik. Minden kedves Olvasónk talál a lapban egy kérdőívet. Kérek mindenkit, ha van pár perce, töltse ki, hajtsa össze és adja postára a kérdőívet. Magyarországon nem kell rá bélyeget tenni, hisz a tudás pénz, és mi tudni szeretnénk mindenki véleményét, így mi fizetünk. Várunk ötleteket, témakéréseket is, miről írjunk (vagy miről ne írjunk már többet), mit rakjunk a lemezekre stb.

De nem csak az újság változott meg, megváltozott e röpke pár hónap alatt a Linux is. Most már nem arról szól a történet, hogy használjunk Linuxot, mert utáljuk ezt vagy azt a céget, és végre nem is arról, hogy miként lehet kiváltani egy használt rendszer valamelyik szolgáltatását Linuxszal. Most már végre eljutott oda a tömeg, hogy „igen, a Linux képes ezt és azt a feladatot is ellátni". Most jön az igazi megmérettetés. Egyre több helyen kezdenek használni komoly munkákra Linuxot. Hamarosan eljutunk oda, hogy egy vállalat hálózata valóban vegyes lesz: ötven gép közül harminc-harmincötön Windows, körülbelül ugyanennyin Linux, egy-két másikon Unix-változat, néhányon Macintosh, elszórva található pár Warp, és az egész mögül néhol még kilátszik a Novell. No, ez lesz a komoly feladat. Kétszeresen is, hisz a rendszerek is vizsgáznak, hogy mennyire képesek egymás mellett (és egymással) működni, és a rendszergazdák is, hiszen nem lesz elég csak Windowshoz, csak VMS-hez, vagy csak Solarishoz érteni. Tudni kell kezelni bárki nyűgjét, és ha ő éppen Macintosh alatt dolgozik, akkor az alatt kell a hálózati nyomtatót kezelni, ha 2000-et futtat, ott kell megoldani a CD-írást, és ha Linuxot, akkor az alatt kell megoldani, hogy fusson a StarOffice.

Készen állunk erre? Most hagyhatnék egy kis hatásszünetet, és világ szemére hányhatnám: „Nem!" De hogyan is állhatnánk készen? Kevés olyan szakembert ismerek, aki mindegyik (vagy legalább három) rendszerben otthon van, még kevesebbet, aki ráér másoknak bármit is elmagyarázni. Tehát mostantól nagy hangsúlyt kell fektetni az oktatásra, a nevelésre, a megismertetésre, a használat megkönnyítésére. Aki nem érti, hogy miért fontos ez, lapozzon a huszonhatodik oldalra, és olvassa el a széljegyzetet.

Az oktatás és megismertetés nagyon fontos, a nyílt fejlesztések pedig rendkívül hasznosak. Olyannyira, hogy ezt még a legmagasabb szinteken is látják, sőt, támogatják is. Amikor először hallottam a támogatásról, azt sustorogták, hogy ennek is biztos megvan az oka. Miért pont akkor? Miért pont az LME? De akár tényleges szakértelem hívta életre a támogatást, akár valamiféle mézesmadzag ez a közhangulatnak, jókor és jól jön! Szorítottam a fiúknak (és természetesen a lányoknak is, merthogy nem kevés gyönyörű linuxos lány van), hogy értelmesen tudják felhasználni a pénzt. Úgy nézem, sikerült! Úgy dolgoznak, ahogy a magyar tűzoltók is. Rendkívül kevesen vannak, és sokkal kevesebb van a zsebükben, mint amennyit megérdemelnének, mégis felveszik a harcot a tűzzel, megkeresik a tűzfészket, és gyorsan, hatékonyan lépnek.

A Linux-felhasználók Magyarországi Egyesülete a támogatási pénzből több pályázatot is kiírt, az egyiket egy tankönyvsorozatra. A tervezet szerint összesen 1650 oldalra rúgó anyagot várnak, olyan tananyagot, melyet minden érdeklődő elér, melyből megtanulhatja a számítógép használatát az alapoktól egészen a rendszergazdai szintig. Külön öröm számomra, hogy a pályázatot feltehettük a lap mellékletére, ha valakit érdekel, a teljes anyagot megtalálja a második lemezen. A tervezet a lap 14. oldalán olvasható.

Már csak egyetlen kérdés motoszkál bennem. Vajon mi lesz az anyaggal, amikor elkészül? Lehet, hogy a nyílt világ terméke egy zárt aktába kerül? Vagy a szellemiséget támogatva a teljes anyagot nyilvánossá teszik mindannyiunk javára? Lehet, hogy ez lesz az első nagyméretű könyv, ami Magyarországon valamelyik nyílt felhasználási szerződés alatt jelenik meg? Türelmetlenül várom!

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

Programvadászat

Rövid útmutató a JBLinux 1.1 megszelídítéséhez.

Mellékletünk második korongján a JBLinux 1.1-es változata szerepel, ez egy viszonylag új Linux-kiadás. Készítője *Ole André Vadla Ravn□s,* egy 18 éves norvég srác. Régóta a Linux rabja. 1999. októberében kezdett el dolgozni a JBLinuxon, ugyanis nem volt megelégedve egyik Linuxszal sem. Bizonyos szempontból kedvelte a RedHat kiadásokat, más okból szerette a Slackware-t, de egyik sem felelt meg maradéktalanul az igényeinek, ezért úgy döntött, létrehozza a saját változatát – teljesen az alapoktól. Kezdetben csak kísérletnek indult, mivel a saját Linux-változat létrehozása túl ijesztő feladatnak tűnt. Ennek ellenére két-három hónap munka után egy teljesen új, működőképes rendszert hozott létre. Így már valósággá vált az, amire ezelőtt még gondolni sem mert. Napról napra közelebb került a megvalósításhoz és körülbelül egy év múlva a nyilvánosság számára is elérhetővé tette a JBLinuxot. Sok biztató visszajelzést kapott, ez adott erőt a fejlesztés folytatásához.

A mellékelt CD könyvtáraiban szétnézve mindenkinek feltűnhet, hogy a programcsomagok jbl végződést kaptak, ezek valójában egyszerű tgz csomagok. Mindegyik csomagban van egy indexfájl, ezek a fájlok a /usr/share/installed könyvtárba kerülnek, így a Slackware csomagkezelésével is képes együttműködni. A csomagok kezelésére két program használható: az installpkg segítségével újabb csomagokat adhatunk rendszerünkhöz, az uninstallpkg programmal pedig eltávolíthatjuk azokat. Könnyedén készíthetünk saját ybl csomagokat is a makejbl program segítségével.

A JBLinux csomagjait Pentium vagy annál jobb processzorokra fordították, ezért 486-os és 386-os gépeken nem, vagy csak nagyon rossz teljesítménnyel futnak.

A JB Linux két legnagyobb előnye a gyorsaság és a csomagok frissessége, hiszen az 1.1-es változat is csak pár hónapos. A csomagválasztéka mindenki számára használható rendszerré teszi, megtalálhatók benne a programfejlesztéshez szükséges eszközök, különböző kiszolgálók, multimédia programok (CD-nyersmásoló, MP3-kódolók és-lejátszók, CD-lejátszók stb.).

Hamarosan elérhető lesz az 1.2-es változat beta1 kiadása is. Újdonságai közül néhány:

- 2.4.1-es rendszermag
- glibc 2.2.1
- gcc 2.95.2
- XFree86 4.0.2
- KDE 2.0.1

Telepítési útmutató

Rendszerünket CD-ről indíthatjuk, vagy ha erre gépünk BIOS-a nem képes, akkor DOS (Windows) alatt a rawrite.exe, Unix alatt pedig a `dd` paranccsal tudjuk a szükséges indítófájlokat hajlékonylemezre írni. Indulás után rövid üdvözlő és tájékoztató szöveget olvashatunk a készítőtől, melyben leírja, hogy ezt a változatot nem kezdőknek szánta. Ebben teljesen egyetértek vele, mivel alapvető linuxos ismeretek nélkül nagyon nehéz telepíteni. Így csak megerősíteni tudom: nem javasolt első Linux-változatként kezdők számára.

Következő lépésként be kell állítanunk, hogy rendszerünk csak IDE-eszközöket használ-e, vagy tartalmaz SCSI-egységeket is. Ha SCSI felületen lévő eszközeinket is akarjuk használni a telepítés folyamán, esetleg a CD-ROM vagy a merevlemez ilyen felülettel csatlakozik a rendszerünkhöz, akkor a megfelelő modult a memóriába töltve máris használatba vehetjük azt. CD-meghajtónkat kiválasztva egy lemezterület-felosztó menübe jutunk, ahol a cfdisk program segítségével minden szükséges lépést elvégezhetünk. A lemezterületek sikeres létrehozása után a befűzési pontok megadása következik. Miután ezzel is sikeresen végeztünk, a programok tényleges telepítése előtt már csak a lemezterület használni kívánt formázása van hátra. Választhatjuk az ext2-es fájlrendszert, ami a legtöbb Linux-változatban alapértelmezett, vagy az új ReiserFS jegyzőkönyvező (journaling) fájlrendszert. (Lásd írásunkat az 51-53. oldalon.) Ezt a fájlrendszert a biztonságos adattárolás érdekében kezdték el több helyen is használni.

A csomagok telepítésénél két lehetőség közül választhatunk, az egyik választáskor minden, a CD-n szereplő csomag felkerül a gépünkre, a másik választásnál saját magunk válogathatjuk össze a számunkra szükséges csomagokat. Amennyiben a teljes telepítést választjuk, legalább 2 GB lemezterületre lesz szükségünk. A grafikus felületek között is választhatunk, telepíthetjük a 3-as vagy a 4-es sorozatú XFree86-ot. Amennyiben saját magunk szeretnénk a telepítendő csomagokat kiválogatni, akkor kénytelenek vagyunk a teljes listát végignézni. A csomagok telepítése után a LILO beállítása és telepítése következik. Ha ezen is sikeresen túljutottunk, akkor újra kell indítanunk a rendszerünket. Első indításkor egy önműködően elinduló könnyen használható beállítóprogram `setup` fogad bennünket. Segítségével egyszerűen elvégezhetjük rendszerünk testreszabását. Ennek a programnak köszönhetően USB-s egeremet szinte azonnal használatba vehettem karakteres felületen is. A hangkártya, a hálózat és a grafikus felület beállítása sem jelentett gondot. Nekünk kell minden rendszermagmodult (az eszközmeghajtókat) betöltetni a memóriába. Elsőre sikerült minden elemet felismertetni a programmal, de mint erről a cikk elején már írtam, ehhez elengedhetetlen az alkatrészeink pontos ismerete, valamint azt is tudnunk kell, hogy melyikhez milyen modul szükséges. A nyelv, a billentyűzetkiosztás, az egér, a hálózat, a hang, az XFree86 stb. beállításait bármikor módosíthatjuk a későbbiek során a `setup` program segítségével.

Az újraindítás után USB-s billentyűzetem is azonnal feléledt, így a fentebbi beállításokat már ennek segítségével végezhettem el. Mindezek után egy kényelmesen használható rendszert kaptam. Meglepetésemre a telepítés alatt sehol sem kellett a rendszergazdai jelszót megadnom. Újraindításkor a rendszergazdát a rendszer jelszó nélkül beengedi, ezután azonnal érdemes jelszót adnunk! Nagyon hiányoltam viszont az erre vonatkozó figyelmeztetést.

Tapasztalataimat összefoglalva a JBLinux egy kényelmesen használható, jól beállítható és kedvező adottságokkal bíró rendszer. Természetesen, mint minden Linux-rendszernek, ennek is van még mit fejlődnie, de a kezdet biztató. Mindazoknak ajánlom a rendszer kipróbálását, akik egy kicsit más felépítésű Linuxot keresnek, mint az rpm és deb vonulat.

➲ http://www.jblinux.com

Rendszermag

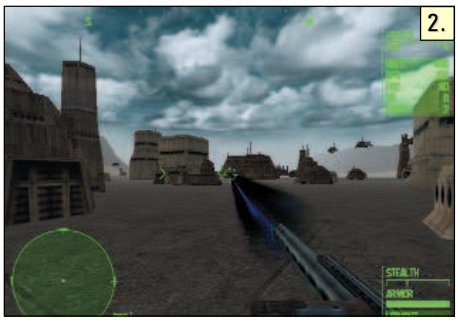
Mellékletünk első lemezére ismét felkerült a legfrissebb megbízható rendszermag, ez jelenleg a 2.4.1-es változat. A 2.4.0-s változathoz képest a legnagyobb újdonság, hogy most már hivatalosan is bekerült a ReiserFS támogatása a rendszermagba. Sok hibát is kijavítottak, ilyenek például a RAID 5, a HPT366, az egyes hálózati kártyák meghajtói, valamint az USB.

Játékok

Két játék is helyet kapott a szórakozni vágyók örömére, egyik a Heavy Gear II *(2. kép),* melyről egy átfogó leírást olvashatnak a 112–113. oldalon, a másik pedig a mindenki által ismert Quake III Arena. Futtatásukhoz elég erős gépre van szükségünk, ugyanis a legkisebb követelmény: 2.2.9 vagy későbbi rendszermag, glibc 2.0 vagy 2.1, Pentium 233 MHz, 8 MB VGA, vagy Pentium II 266 MHz 4 MB VGA, 4-szeres CD-ROM meghajtó, 64 MB memória – de 128 az ajánlott –, OSS-megfelelő hangkártya.

3.


1.



2.

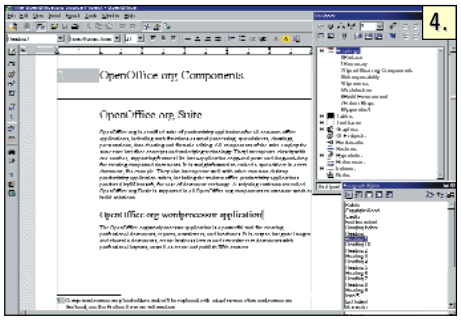
A Quake III Arena *(1. kép)* OpenGL-t használ a 3D-s gyorsításhoz, jelenleg a 3dfx Voodoo és a Matrox G200/G400 kártyákat a támogatja. Mindkét játékról megoszlanak a vélemények, de az akciójátékok kedvelői megegyeznek abban, hogy mindkettő alapmű.

Eazel Nautilus

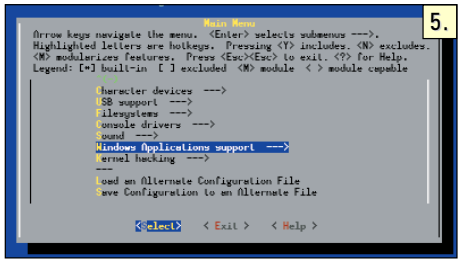
Az Eazel cég által készített Nautilus hálózati környezet szintén helyet kapott mellékletünkön. Olyan neves cégek támogatják, mint a Sun és a RedHat. A Nautilus héj egy új szemléletű programkörnyezet a GNOME-hoz, mely magába foglalja a fájlkezelőt *(3. kép)*, az Internet-böngészőt és a rendszerkezelő részeket. Ez az alkalmazás nemcsak a Linux könnyű használhatóságában teremt új lehetőségeket, hanem a helyi és a hálózati szolgáltatások egy helyről történő eléréséhez is hozzásegít. Hozzáférhető a GNU felhasználási szerződés alatt.
Néhány kiemelkedő tulajdonsága:

- haladó szellemű fájlkezelés,
- beépített fájlnéző,
- virtuális könyvtárak kezelése,
- URL-alapú címzés alkalmazása,
- ikonok használata.

Új felhasználói felületének jellemzői: nagyítható, több felhasználói fokozat állítható be a kezdőtől a haladóig, valamint testre szabható fájlrendszere van.
Kifinomult rendszerbeállítási lehetőségekkel bír a programtelepítés és a programfrissítés terén, valamint állandó hálózati tárhely elérése helyi meghajtóhoz hasonlóan.



4.



5.

Gimp 1.2.1

A Gimp grafikai program legújabb változata, az 1.2.1-es sok hibajavítást tartalmaz az előzőhöz képest. Lásd még cikkünket a 86. oldalon.

Glibc2.2

Előző CD-nken sajnos sérült volt a glibc2.2.tar.gz fájl. Ezt a hibát most orvosoljuk.

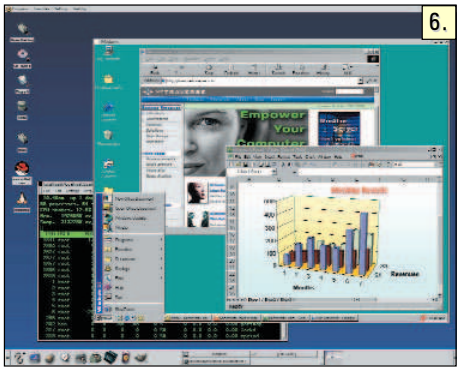
Openoffice.org

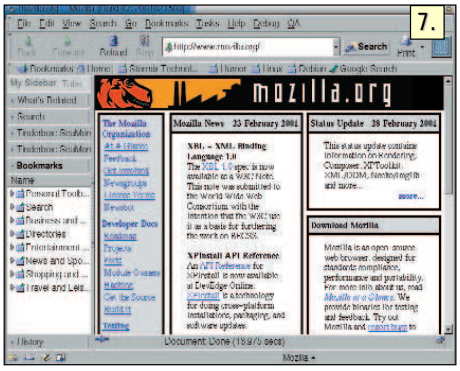
A Sun Microsystem kiadta a Staroffice forráskódját, így elkezdődhetett egy nyílt forrású irodai csomag fejlesztése. Ennek eredményeként született meg az openoffice_614. Egyelőre még fejlesztés alatt áll, de már most is eredményesen használható. *(4. kép)* Mivel a forráskódja elérhető, ezért átvihető bármelyik operációs rendszerre, melynek van grafikus felhasználói felülete és C++ fordítója.

Rendszerfrissítés

A rendszer frissítése mindig fontos feladat, még akkor is, ha nem hálózatba kapcsolt gépről van szó. Használat közben számos olyan hibára derülhet fény, ami aláásható a rendszer megbízhatóságát. Így a CD-inken eddig is rendszeresen közreadott frissítéseket, hibajavításokat ezután is minden hónapban megjelentetjük. Az összes Linuxhoz sajnos nem tudjuk feltenni ezeket a csomagokat, de a Linuxvilág honlapján szavazásra bocsátjuk, hogy a frissítések mely változatokhoz jelenjenek meg a CD-n.
Most a RedHat, a Mandrake, a Debian és a JBLinux frissítéseit adjuk közre. Segítségükkel mindegyik rendszert közel naprakész állapotba hozhatjuk.

Harmadik CD-nkre felkerült a Mozilla javított változata, az ingyenes Opera 5 beta böngésző, a Netscape 6 böngésző teljes készlete és legnagyobb anyagként a Win4Lin programcsomag időkorlátos változata a Netraverse cégtől.





Lindows vagy Winux?

Sokaknak gondot és kényelmetlenséget jelent megszokott windowsos programjaik nélkülözése Linux alatt. Ehhez kíván segítséget nyújtani a Netraverse cég Win4Lin programcsomagja, mellyel, Windowst futtathatunk linuxos számítógépeken, így bármelyik kedvelt programunkat használni tudjuk. A Win4Lin program egy virtuális számítógépet ad a Windows alá, így az észre sem veszi, hogy nem a valódi gép erőforrásait használja. Ennek köszönhetően az operációs rendszer és a felhasználói programok telepítése, valamint a rendszer beállítása nem jelenthet gondot. *(6. kép)* A fejlesztők sajnos csak az RPM-alapú rendszereket támogatják, így a Debian-alapú rendszereken nehézkes a telepítése.

A fejlesztők külön rendszermagfoltot (kernel patch) írtak, mivel a program működéséhez szükséges kiegészítő szolgáltatásokat azon keresztül érhetjük el. A 2.2-es sorozathoz találhatunk megfelelő foltokat a ⇨CD Win4Lin/LINUX/PATCH könyvtárában. A foltozáshoz szükséges a rendszermag teljes forrása és a megfelelő változatszámú folt. Mi a szerkesztőségben egy 2.2.18-as magot foltoztunk meg, és eközben érdekes változás történt a rendszermag beállításában *(5. kép)*

A foltozást a `patch` parancs segítségével tudjuk végrehajtani (például: a PATCH könyvtárban kiadjuk a `patch -p1 -b -d /usr/src/linux <Kernel-Win4Lin2-2218.patch` parancsot). Ez a lemez tartalmaz megoldást azok számára is, akik nem szeretnék rendszermagjukat újrafordítani, mivel előre elkészített RPM-csomagok közül választhatják ki a megfelelő magot. ⇨CD Win4Lin/LINUX/RPMS. Támogatott változatok: Caldera Linux, SuSE, Mandrake és RedHat, választhatunk egy- vagy többprocesszoros (SMP) gépekhez való új magot is.

A telepítő indításkor ellenőrzi, hogy a kiegészítő szolgáltatásokat támogatja-e a rendszermag. Amennyiben nem, hibaüzenettel kilép. Ez a program időkorlátos, ezért csak május elsejéig használhatjuk. Mandrake Linux alatt a telepítés gond nélkül lezajlott, miután a rendszermagot lecseréltem és a sh /Win4Lin/Win4Lin/install-win4lin.sh parancsot futtattam. Debian alatt már jóval nehezebb volt a telepítést elkezdeni. Először rendszermagot kellett fordítani, azután az előző parancs kiadásával próbálkoztam, de nem működött, az `rpm -i Win4Lin-5.1.1g-1.i386.rpm` parancsra is hibaüzenettel állt le, mivel hiányolta a /bin/sh-t (természetesen az sh a helyén volt). Ezért az `rpm -i --nodeps Win4Lin-5.1.1g-1.i386.rpm` segítségével telepítettem. Ezután a `winsetup` parancsot elindítva végre elkezdhettem a rendszer beállítását.

Mozilla 0.8

Újabb állomáshoz érkezett a Mozilla fejlesztése, amit az új változat kiadása is jelez. Ez a változat sokkal megbízhatóbb, gyorsabb, mint az előzőek. CD-nken megtalálható a program forrása, futtatható állományként pedig felkerült az i386 Linux és FreeBSD, valamint a PPC-re fordított változat is.

A könyvtárban található néhány állomány különböző kiegészítésekkel lett fordítva, ilyen közülük a mozilla-i686-pc-linux-gnu-0.8-talkback.tar.gz, mellyel összeomlás esetén adatokat küldhetünk a fejlesztőknek, a mozilla-i686-pc-linux-gnu-0.8-MathML-SVG-XSLT.tar.gz, mely MathML, SVG és XSLT kiegészítésekkel rendelkezik, a mozilla-i686-pc-0.8-gcc-2.95.2-glibc-2.2-slackware-linux-gnu-mathml-svg-xslt.tar.bz2, mely egy Slackware alatt fordított MathML, SVG és XSLT kiegészítésekkel, valamint a mozilla-powerpc-unknown-linux-gnu-0.8-PSM.tar.gz, mely PSM-támogatással készült Power PC-rex86 alá. *(7. kép)*

Opera 5 Beta 6

Az Opera böngésző, mint arról korábbi számunkban hírt adtunk (2000. november), mostanra mindenki számára ingyenesen elérhetővé vált. Ezért cserébe egy reklámszalagot kell elviselnünk a böngészőnk jobb felső sarkában. Régebbi CD-nken már megjelent az Opera 4 Beta, azonban úgy tűnik, nem is lesz belőle végleges változat, mivel a fejlesztők egy egész számot ugrottak. Számos újdonsággal gazdagodott ez a változat, egyszerű telepítés jellemzi minden rendszeren, választhatjuk a csomagkezelők (rpm, dpkg stb.) használatát, vagy akár a tar.gz fájlok kicsomagolásával is telepíthetjük. Ha rendszerünk rendelkezik a megfelelő könyvtárakkal, akkor válasszuk a kisebb méretű fájlt, ha nem, vagy nem vagyunk benne biztosak, használjuk a nagyobbik fájlt, a kettő közötti különbség, hogy a nagyobb minden olyan könyvtárat tartalmaz, ami szükséges a böngésző futtatásához. *(8. kép)*

Netscape 6.01

Ez a változat, mint a száma is mutatja, csak kisebb hibajavításokat tartalmaz. Gyorsabb lett, ez mindenképpen előnyére vált, hiszen aki használta a 6-ost, bizony tapasztalhatta a lassúságát. *(9. kép)*

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

Új Linux-kiadások

Hamarosan megjelennek a 2.4-es rendszermaggal „szerelt” Linux-változatok. A RedHat már elérhetővé tette a Fisher, azaz Halász fedőnevű következő kiadás próbaváltozatát a 2.4.0-s rendszermaggal és 4.0.2-es Xfree86-tal. A német SuSE Linux is kihozta a saját,

7.1-es változatát. Ezzel a lépéssel a Linux nagyvállalati körben is támogatott operációs rendszerré vált. A nagyvállalatok között található például az Oracle és a Compaq is. A RedHat Linux Fisher béta változata letölthető az ftp.fsn.hu-ról.

➲ http://www.redhat.com
➲ http://www.suse.com

A SuSE linux 7.1-es kiadása lesz a tervek szerint az első teljesen magyar nyelvű Linux. Március közepétől kapható. Két rendszermag, a 2.4-es és a 2.2.18-as is szerepel a telepítő CD-in. A telepítő sok segítséget nyújt, például egyszerűen újraméretezhetjük a windowsos lemezterületünket, emellett a rendszerindítás is grafikus felületen történik.

➲ http://www.suselinux.hu
➲ http://www.suse.com
➲ http://www.suse.de

Unixos guruképző

Ha valaki nem csak felhasználói szinten szeretne érteni a Unix-rendszerekhez, annak bizony számos adatra és sok-sok segítségre van szüksége. Az *ugu* (Unix Guru) oldalain

a kezdőktől egészen a „gurukig” mindenki hozzájuthat a szükséges támogatáshoz. Kereshető adatbázisának köszönhetően egyszerűen eligazodhatunk a leírások tengerében.

➲ http://www.ugu.com

ReactOS

Megjelent a ReactOS legújabb változata, a 0.0.17. Fejlesztői azt a célt tűzték ki, hogy egy Windows NT-megfelelő rendszert hozzanak létre, és a felhasználói alkalmazni tudják a már meglévő eszközmeghajtókat és felhasználói programokat. Elérhető a GPL felhasználói szerződés alatt, így mindenki ingyenesen hozzáférhet mind a forráskódhoz, mind pedig a futtatható állományokhoz. Egyelőre Intel- és Alpha-alapú számítógépekre érhető el, de fejlesztői szeretnék a jövőben több felületre is átültetni.
A 0.0.16-os változathoz képest sok változás történt a biztonság, az eszközkezelés és a rendszerindítás terén. Induláskor a shell.exe helyett a winlogon.exe fut le. Jól működő hajlékonylemez-vezérlőt, más fájlrendszerek támogatását, BIOS-szolgáltatások meghívását, a winsocks verem javítását, valamint egyszerű W32 telnet-ügyfelet tartalmaz.

PostgreSQL TOAST

A TOAST rövidítés a The Oversized Attribute Storage Technique, Túlméretezett Mező Tároló Eljárást takarja. A PostgreSQL

eddigi legnagyobb hiányossága volt a táblázatokban a korlátozott sorméret. A következő 7.1-es változatban azonban ez már nem jelenthet gondot, mivel ebben a TOAST segítségével már megoldották a 8 k feletti karakterszám tárolását.

➲ http://www.postgresql.org/projects/devel-toast.html
➲ http://www.postgresql.org

IBM

Az IBM bemutatta az eServer x430 kiszolgálóját. Ezzel a géppel igyekeznek az űrt kitölteni a kiszolgálópiac két véglete között. Az x430 az első olyan kiszolgáló, amely kihasználja a Linux Alkalmazáskörnyezet (Linux Application Environment, LAE) tulajdonságait. A LAE segítségével az egyprocesszoros Intel gépektől a nagygépekig bővíthető az eServer sorozat. Az IBM LAE-központot is nyitott Oregonban, ahol a programgyártók kipróbálhatják ezt a környezetet.

➲ http://www.ibm.hu
➲ http://www.ibm.com

Darwin

Az Mac OS X alapja a Darwin, egyelőre még csak Power PC-ken fut, de folyamatban van az i386-os rendszerekre történő átültetése. Fejlesztői szeretnének minél átfogóbb leírást adni, ennek elősegítésére egy nyílt kézikönyvet indítottak útjára. Bárki részt vehet a fejlesztésében, munkájával segítve a Darwin fejlődését.

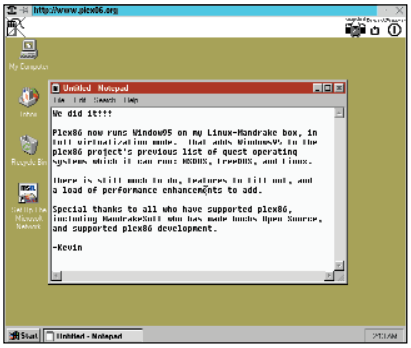
Hasznos adatokhoz juthatunk a

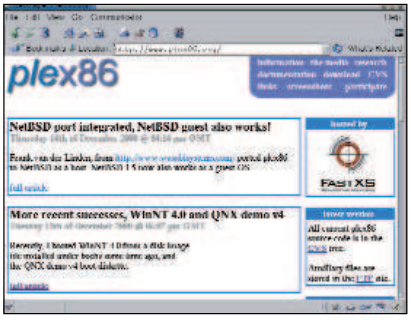
➲ http://www.darwinfo.com weboldalon is.
➲ http://www.opensource.apple.com/projects/documentation/
➲ http://www.darwin.org

plex86

A plex86 egy géputánzó program, a VMwarehez hasonló, viszont ingyenes. Segítségével a gazda rendszeren belül futtathatunk bármilyen másik operációs rendszert. Eddig csak Linuxon lehetett elérni, most azonban a Wasabisystems cégnek köszönhetően NetBSD alatt szintén működik és futtatja is azt. Szolgáltatásait és tudását tekintve még nem teljes, így napi használatra nem alkalmas. Sikeresen futtattak már WIN95-öt, DOS-t, FreeDOS-t és Linuxot.

➲ http://www.plex86.org





Debian-alapokon

Progeny

Egy új, egyelőre még csak béta állapotú Linux-változat, a Progeny fejlesztése mutat új irányt a Debianra épülő rendszerek számára. Grafikus felületen történik a telepítés, ezt a GTK+-ra építették. Automatikus gépfelismeréssel bír, a nyomtató beállítását pedig *Grant Taylor* foomatic nyomtató adatbázisa alapján lehet elvégezni. Jellemzői a következők:

- rendszermagja: 2.2.18
- glibc: 2.2
- X Window: XFree86 4.0.2
- böngésző: Mozilla 0.7.

Grafikus felületet készítettek a debconf programhoz, így a rendszer beállítását már ezzel végezhetjük el. Sok program bekerült a Woodyból (a Debian következő hivatalosan megjelenő változatából).

➲ http://www.progeny.com

Telemetry

A Telemetry 1.0, segítségével hálózati gépeket felügyelhetünk. A program Debian Potato-alapokon nyugszik. Könnyű beállítás és telepítés jellemzi, mindössze 3-4 kérdésre kell válaszolni, és a rendszer már működik is. Néhány jellemző szolgáltatása:

- felfedező modul, mely átvizsgálja a hálózatot, és SQL-adatbázisba írja az eredményt
- Apache/PHP/MySQL/PHPMyadmin felügyeleti felület
- Interneten keresztül történő teljes beállítás
- SSH/HTTPS támogatás
- könnyű telepítés, a hálózati kártya önműködő felismerése, önműködő lemezterület-kiosztás és formázás, nincsenek felesleges kérdések,
- grafikus jelentéskészítés.

A program által elfoglalt tárhely mérete mindössze 50 MB.

➲ http://www.openrock.com

Linuxinsider

Számos érdekes adattal szolgálnak ezeken az oldalakon, legyen szó akár Linux-változatokról, akár az adatbázisokról, akár a rendszermagról, esetleg szakmai hírekről. Minden látogató megtalálhatja a számára fontos híreket a Linux világából.

➲ http://www.linuxinsider.co

NetBSD a StrongARM-alapú PDA-kon

A NetBSD szinte mindenféle processzorral rendelkező gépen használható operációs rendszer (lásd BSD-körkép című írásunkat a 2001. januári szám 10. oldalán) A fejlesztők szeretnék minél többfajta számítógépre elérhetővé tenni. Maga az alaprendszer nagyon kicsi, ennek ellenére tartalmaz egy grafikus felületet is. Használhatjuk amigán, beboxon, dreamcaston, i386, mac68k, macppc, sparc és még sok más gépen is (a teljes listát megtalálhatjuk a ➲ http://www.netbsd.org címen).

A NetBSD csapat bejelentette a NetBSD/hpcarm változatot. Ez StrongARM-alapú PDA-kon fut, ilyen a HP Jornada 720 és a Compaq iPAQ H3600. Úgy tűnik ezzel a változattal és az előző évben megjelent hpcmips-változattal, lekörözi a linuxot a tenyérgépek piacán.

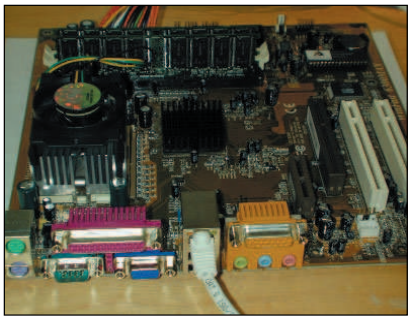
➲ http://www.netbsd.org

➲ http://www.netbsd.org/Ports/hpcarm

Linux a BIOS helyén?

Egy lelkes fejlesztőcsapat azon dolgozik, hogy Linuxszal váltsa ki a BIOS-t (Basic Input Output System). Az új projekt neve LinuxBIOS. Lecserélték a BIOS-t fürtbe kötött számítógépeiken, így a Linux segítségével indítják el a gépeket. Ennek a műveletnek a szabványa, a PXE az Inteltől

származik, ez azonban számos korlátot tartalmaz. A modern operációs rendszereknek valójában nincs is szükségük a BIOS-ra, mivel rendszerindítás után kiváltják a szolgáltatásait. (Egy egyszerű példa erre: ha a BIOS-ban nem állítjuk be a második merev-



lemezünket, akkor rendszerindításkor a BIOS nem ismeri fel, viszont Linuxot elindítva mégis használhatjuk a második merevlemezt). A LinuxBIOS segítségével elérhetjük, hogy a számítógépeken választható legyen a rendszerindítás módja, például helyi lemezről vagy hálózatról. Nem kell majd minden hálózati kártyához EPROM-ot készíteni. A sebessége pedig elképesztő: egyfelhasználós módban indítva három másodpercre van szükség.

Ha sikerül lecserélni a BIOS-t Linuxra, akkor rövidebb, biztonságosabb és rugalmasabb lesz a rendszerindítás.

➲ http://www.linuxbios.org

GPL-es játékgyár

A francia Nevrax cég egy GPL-es kiszolgálóalapú játékkörnyezetet készít. Főbb jellemzői: mesterséges értelem alkalmazása, hálózati játékmód támogatása, valamint 3D-s megjelenítés. A Snowball játék bemutatópéldánya letölthető az alábbi címről.

➲ http://www.nevrax.org

Sun

A Sun Microsystem mindenki számára ingyenesen letölthetővé tette a Solaris 8 operációs rendszerét. Eddig 75 dollárért kellett megvásárolni a CD-ket, amelyek tartalmazzák a rendszer telepítéséhez szükséges összes fájlt és a leírást. A Solaris 8 legfeljebb nyolc processzorig használható ingyenesen. Gépigénye átlagosnak nevezhető, gép- és alkatrész-támogatása elég széles, mindezek ellenére könnyedén előfordulhat, hogy valamelyik alkatrészünket mégsem támogatja. Bejegyzés után letölthető.

➲ http://www.sun.com/solaris/binaries/get.html
➲ http://www.sun.com
➲ http://www.sun.hu
➲ http://www.sun.de/Produkte/Software/Systeme_md_Platformen/Solaris/Solaris8/index.html

KDE 2.1

Megjelent a KDE 2.1 üzembiztos változata. Javítottak a megjelenítésen, a multimédiás képességeken, a biztonságon és a megbízhatóságon is.

A 2.1-es KDE része az 1.4-es Kdevelop fejlesztői környezet. Ez az első üzembiztos kiadás, amely a KDE2 könyvtárra épül.

➲ http://www.kde.hu
➲ http://www.kde.org

Linuxos honlap

A ➲ http://www.linuxlinks.com webhelyen több ezer linuxos hivatkozás közül válogathatunk kedvünkre. Különböző tárgykörökre osztva kereshetünk az adatok között. Választhatunk a kezdők oldalai, a biztonság, a hálózat, a leírások, a beágyazott Linux, a programok és a játékok közül. Hamarosan hasonló szolgáltatást indítunk a Linuxvilag.hu weboldalon is, természetesen elsősorban a magyar nyelvű forrásokra összpontosítunk. Így a kezdő linuxosok is nagyobb biztonsággal merülhetnek el a Linux rejtelmeiben.

➲ http://www.linuxlinks.com
➲ http://www.linuxvilag.hu/cimtar.phtml

A Kék Óriás óriást alkot

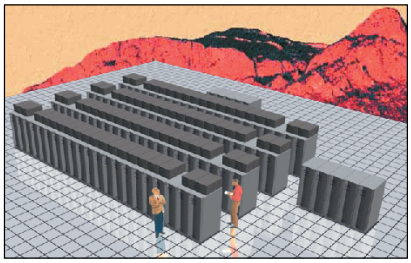
Az IBM kutatói a Yorktown Heightsban (USA) működő Thomas J. Watson kutatóközpontban a világ leggyorsabb számítógépének elkészítésével foglalkoznak. A tervek szerint valamikor 2003-ban fogják befejezni. A Blue Gene (Kék Gén) névre keresztelt gép különlegessége, hogy nem atomfizikusok, meteorológusok, matematikusok, kriptológusok, kozmológusok és mérnökök fogják használni, hanem biológusok. Arra szeretnék használni, hogy felderíthessék, miként veszik fel a fehérjék saját térszerkezetüket. Ezen adatok birtokában sokkal könnyebbé válik a gyógyszerek hatékonyságának növelése.

A Kék Gén százszor gyorsabb lesz, mint az eddig valaha készített bármelyik számítógép, de valószínűleg a feladat megoldása is nagy gondot fog okozni. Egyelőre még az is kérdés, hogy a fehérjék kialakulása számítógéppel modellezhető-e?

➲ http://www-5.ibm.com/hu/news/2001/bluegene.html

A világ ötszáz leggyorsabb számítógépe

A ➲ www.top500.org honlapon olyan csúcsgépek listáját találhatjuk meg, amit a legismertebb cégek, az SGI, az IBM, a Compaq stb. építettek. A listát az IBM által létrehozott 4938 Gflops teljesítményű gép vezeti, a második pedig az Intel ASCI Red, 2379 Gflops teljesítménnyel. További érdekességek az alábbi címeken.



➲ http://www.top500.org
➲ http://pdswww.rwcp.or.jp/
➲ http://kepler.sfb382-zdv.uni-tuebingen.de/start.shtml
➲ http://indy2.lpds.sztaki.hu/scc/
➲ http://www.iit.uni-miskolc.hu/~vadasz/it02_szgpek/

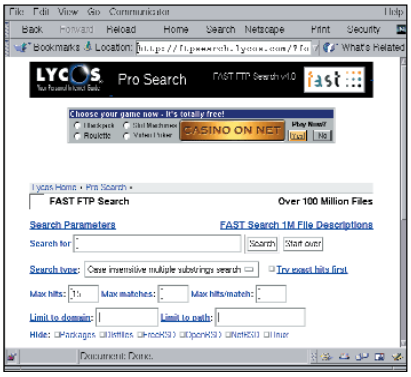
GNOME GKB

Megjelent a GKB billentyűzetátkapcsoló applet új próbaváltozata. Segítségével billentyűzetkiosztást lehet váltani, ha egyik éppen futó alkalmazás sem használja az ALT-SHIFT billentyűpárt, akkor ezek segítségével válthatunk, vagy beállíthatunk egyéb kombinációkat is.

➲ http://www.gnome.org
➲ http://www.gnome.hu

Ftp-keresők

Hasznos lehet egy gyors segédeszköz, ha szeretnénk megtalálni valamit az Internet rengetegében, de nem szeretnénk órákat eltölteni a fájlok felkutatásával. Az alábbi



két kereső segítségével pillanatok alatt megtalálhatunk mindent, ami az Interneten elérhető és e keresők adatbázisában is szerepel. Keresési idejük nagyon rövid, így hamar megtudhatjuk, sikerrel jártunk, vagy egy másik keresőt kell választanunk.

➲ http://ftpsearch.lycos.com
➲ http://ftpsearch.ntnu.no

Linux a hazai vállalati felhasználói körben

Nyolc-kilenc év alatt hobbiprogramból világméretű mozgalommá vált a Linux, ez egyedülálló a számítástechnika történetében. Szinte mindennap jelennek meg hírek, amelyek dicsérik, és arról tájékoztatnak, milyen nagy cégek csatlakoztak a „linuxos mozgalomhoz". Arról viszont kevesebbet olvashatunk, tulajdonképpen mire is használják ezt az operációs rendszert. Sokáig tartotta – talán még most is tartja – magát az a nézet, hogy otthoni játékszernek kitűnő, de ipari környezetben, igazi munkára nem alkalmas. Ennek egyik oka lehet, hogy a cégek általában nem dicsekszenek azzal, ha valamilyen nyílt forráskódú, ingyenes rendszert használnak, nehogy komolytalannak tűnjenek. Reméljük, cikkünk segít feloldani a magyar cégek félelmeit. Az Egyesült Államokban vagy Nyugat-Európában már az olyan hatalmas vállalatok, intézmények sem titkolják, hogy Linuxot használnak bizonyos feladatokra, mint a NASA, a Boeing Company, a Cisco Systems Inc., a Corel Computer Corp., a Mercedes-Benz AG, a Sony Electronics Inc., az O'Reilly Kiadó, a United States Postal Service, a Netscape Communication Corp., vagy a United States Army Publishing Agency.

Mi arra voltunk kíváncsiak, Magyarországon mennyire terjedt el a Linux használata, az egyes cégek milyen feladatokat bíznak rá. Ezért belevágtunk egy önkéntes adatszolgáltatásra épülő felmérésbe. Felvettük a kapcsolatot különböző cégekkel, és a felmérés eredményéből közreadunk egy válogatást. A folyamatosan bővülő anyagot a ➲ http://www.infopen.hu/ honlapon helyezzük el, és időről időre nyomtatásban is közreadjuk.

Linux-támogatás

Mára nemcsak a gép- és alkatrészgyártóknál általános a Linux támogatása, hanem a kis és nagy programgyártó cégek is sorra jelentik be, hogy átültetik különböző programjaikat Linux alá. Néhány ezek közül:

Fejlesztői rendszerek: Cygnus, IBM, Borland, Inprise, Informix, Magic Software Enterprises, Oracle, Compaq, PlugSys International, SGI.

Adatbázis-kezelők: IBM, Sybase, Oracle, Informix, Pervasive Software Inc., Progress.

Üzleti programok: SAP, IBM, Pervasive Software Inc., McAfee, Gentia Software, Computer Associates International Inc., Lotus, Check Point Software Technologies Inc., Progressive Systems Inc., Data Fellows Corporation, Macromedia Inc.

Hazai cégek kiszolgálóinak jellemzői

Fornax Rt.

A ➲ a http://www.fornax.hu címen elérhető kiszolgálót két 550 MHz-es Pentium III-as processzor hajtja, a rendszernek és az adatoknak két, 8 GB-os Ultra66-os merevlemez ad helyet. A másik webkiszolgáló egy Sun4U, u1 147 MHz-es processzorral, 128 MB RAM-mal és két, 4 GB-os SCSI2-es merevlemezzel felszerelve. A kiszolgálók forgalma napi átlagban 7500 látogató, ez havonta megközelítőleg 200 ezer érdeklődőt, illetve adatforgalomban havi 60 GB-ot, találati számban pedig körülbelül 220 ezret jelent. A gépek nagyon jól méretezhetők, terhelésük csúcsidőben 70-75 százalék. A géphiba miatti kiesés elhanyagolható, évente mindössze egy-kétszer fordul elő. A cégnél Oracle adatbázis-kiszolgálót is használnak. Régebben az iBCS2 emuláció segítségével futtatták a 7.1.3-as változatot, ma viszont a 8.1.5-ös linuxos változatot alkalmazzák. A használt adatbázis mérete nagyjából 1,2 GB, mely tőzsdei adatokat is tartalmaz. Az adatok mentését is Linuxon oldották meg a többfelületű rendszerben. Az Amanda nevű mentőrendszer automatikusan egy HP DAT24-esre és DDS2 kazettákra menti hat-hét gép anyagát, ezek között van Sun Solaris, Windows NT, Linux, sőt, régebben SCO Unix is volt. Rendszerük Debianon fut.

TvNet Kft.

Compaq Proliant kiszolgálót használnak 733 MHz-es Pentium III-as processzorral, valamint két, 18 GB-os SCSI merevlemezt RAID vezérlővel és 256 MB RAM-mal – ezen fut a központi honlapot kezelő webkiszolgáló. Már most tervezik egy új, erősebb gép beszerzését. Ez 833 MHz-es Pentium III processzort és hat, 9 GB-os merevlemezt tartalmaz, amelyeket RAID5-be kötve fognak használni. Ez lesz az új belső hálózati kiszolgáló, és ezen fut majd a levelező- és az adatbázis-kezelő kiszolgáló is. A jelenlegi belső kiszolgálón (333 MHz-es Celeron, 192 MB RAM-mal) egyszerre fut a Sybase SQLanywhere és a PostgreSQL adatbázis-kezelő kiszolgáló. Az egyik a számlázásért felelős, a másik pedig a szolgáltatás iránt érdeklődőket tartja nyilván. A Sybase-hez csak windowsos ügyfelek kapcsolódnak, a PostgreSQL-t az Apache–PHP3 pároson keresztül érik el az ügyfelek. A Sybase-hez is lehet PHP3-as felületet készíteni, a hozzá tartozó odbclib segítségével. Ez a kiszolgáló fájlkiszolgálóként is elérhető kétféle módon: a Samba csomag és nfs segítségével, így megközelítőleg harminc felhasználót szolgál ki. Ez a gép bonyolítja le a levelezést is. A hálózatkezelésben is szerepet kapott a Linux, az snmp felhasználásával. Kisebb feladatokra is Linuxot alkalmaznak, például irc, news-, ftp- és DNS-kiszolgálóként. A cégnél a RedHatet kedvelik. A Sybase-es ügyfeleken, a vezetők és marketingesek gépein kívül nincs Microsoft-alapú program az egész rendszerben.

Dunaferr Távközlési Intézet

Egy 200 MHz-es Pentium MMX-et használnak 32 MB RAM-mal valamint az operációs rendszernek és az alkalmazásoknak egy 1,2 GB-os és egy 2,1 GB-os IDE merevlemez ad helyet. A hálózati kapcsolatot 3c509-es hálózati kártya teremti meg. Erre a gépre a következő feladatokat bízták: céges telefonkönyv, Unix-alapú telefonközpont számlamegőrzése (ftp-vel tükrözés) és ftp-csere. A telefonkönyvet Apache és PHP3 segítségével oldották meg, az ftp-kiszolgáló feladatát a proftpd csomag látja el.

Egy SuSE Linux kezeli az Alcatel telefonközpont hangpostáját. A cégnél a Debiant kedvelik.

Westel Rádiótelefon

Fujitsu, Siemens, Digital Compaq gépeket használnak Linux futtatására és teljes internetszolgáltatást nyújtanak megközelítőleg ezer felhasználónak. A megoldásra használt programok a következők: levelezésre az exim, a sendmail, az imapd és az imp; webkiszolgálóként Apache; ftp-kiszolgálóként a proftpd; proxykiszolgálóként a Squid; adatbázis-kezelésre PostgreSQL; ssh-ra open-SSH. Saját fejlesztésű parancsállományokat használnak a szolgáltatások és a programok futásának ellenőrzésére, hiba esetén ezek figyelmeztető üzenetet küldenek SMS-ben a kijelölt szakembereknek. A 0660 SMS-rendszerét is Linuxon valósították meg. Terveik között szerepel egy ISDN-behívó útválasztó összeállítása. A cégnél a Debiant kedvelik.

Philos Laboratories

166 MHz-es Pentiumtól 600 MHz-es Athlonig terjed a Linuxot futtató számítógépek sora a cégnél. A rábízott feladatok: fájlkiszolgálóként Samba és nfs; webkiszolgálóként Apache; levelezőkiszolgálóként sendmail; ftp-kiszolgálóként wu-ftpd használnak. A linuxos gépeket és a negyven felhasználót NIS segítségével felügyelik. Mivel a cég játékprogram-fejlesztéssel foglalkozik, ezért még fordítókiszolgálót és GNATS hibakövető rendszert (bug tracking system) is használnak. A cégnél a Debiant kedvelik.

Budapesti Műszaki Főiskola

Két 366 MHz-es Celeron processzor, illetve 512 MB RAM és körülbelül 100 GB SCSI merevlemez található abban a gépben, amely az ➲ ftp://ftp.fsn.hu/ címen elérhető ftp-kiszolgálót futtatja. A megvalósításhoz a proftpd nevű programot használják. A gépet rsync szolgáltatással is el lehet érni, valamint fut rajta egy webkiszolgáló, a webfsd. Nagy, napi 150-230 GB-nyi adatforgalmat bonyolít le a gép. Ez havonta nagyjából 5 terabájtot jelent. Régebben is előfordult, amikor még csak 160 MB RAM volt a gépben, hogy 350 felhasználó egyszerre használta a gépet. Ilyenkor megközelítőleg 120 MB csereterületet használt tevékenyen, és a terhelés elérte a 100-150-es értéket is. A memóriabővítés óta a terhelést sikerült kettő környékén tartani. Itt is a Debiant részesítik előnyben.

Medicontur

486DX4-120 MHz-es processzor, 16 MB RAM, 1 GB merevlemez az „otthona” annak a Linuxnak, amelyet levelezésre és internetátjáróként használnak egy ISDN-vonalon keresztül. Erre az „aprócska” gépre csupán ezek a feladatok hárulnak: belső webkiszolgáló: Roxen Challenger; SQL-kiszolgáló: MySQL; LDAP-kiszolgáló: OpenLDAP; ftp-kiszolgáló: proftpd; tűzfal: ipfwadm; fájlkiszolgáló: Samba; levelezőkiszolgáló: sendmail, amely az Amavis víruskeresőt használva szavatolja a beérkező levelek vírusmentességét, és az ISDN-vonalon történt telefonbeszélgetések időtartamát is rögzíti. Körülbelül 15 felhasználót szolgál ki.

Sztaki

2000. március 6-án átadták hazánk legnagyobb teljesítményű számítógépét. A Sztakiban 28 PC-t kapcsoltak össze 100 Mbites hálózattal, így a szuperszámítógépek árának töredékéért közel 30 ezer Mflopsra tudták emelni a gépek összteljesítményét. A telep jellemzői: teljes memóriakapacitás: 3,84 GB; teljes merevlemez-kapacitás: 290 GB; hálózati áteresztőképesség: 34 Gb/mp; csúcsszebesség: 30 Gflop. A gépek műszaki jellemzői (ezekből áll a telep): DELL Precision 410M munkaállomás, két Intel Pentium III 500 MHz-es processzor, 128 MB ECC SDRAM, 9,1 GB Ultra2 SCSI merevlemez, 100 Mbites ethernet hálózati kártya, 3D-s gyorsító videokártya, 32 MB RAM-mal, 40-szeres sebességű SCSI CD-ROM olvasó, 15 colos DELL monitor. A hálózat 100 Mbites ethernet, 48 kapus Cisco 100 Mbites ethernet kapcsolóval (teljesen kétirányú, 24 Gb/mp). Többféleképpen lehet elérni a fürtöt: számos felhasználó számára szavatol legalább egy munkaállomást, illetve sok munkaállomást ad néhány felhasználónak. A Sztaki eddig a Paksi Atomerőmű Rt.-vel és az Országos Meteorológiai Szolgálattal tárgyalt az együttműködés lehetőségeiről, de várják a szupergyors program felhasználása iránt érdeklődő nagyobb vállalatok jelentkezését is. Alkalmazási területeik: a világegyetem vizsgálata, az atomerőműblokkok működésének modellezése, a meteorológiai előrejelzések, a szemcsés anyagok keverése és szétválasztása, a kémiai módszerek alkalmazása, az anyagtani vizsgálatok és a környezetvédelem. A fürt operációs rendszere RedHat Linux 6.1.

Összegzés

Minden informatikai rendszer elemekből épül fel, melyeknek együtt kell működniük. Ezt az együttműködést elérni nagyon nehéz, szinte lehetetlen feladat, különösen több gyártó esetén, kivéve, ha a megoldás nyílt szabványokon alapul, és könnyű az átalakítása, testreszabása. Az szűrhető le a fenti válogatásból, hogy leggyakrabban a nyílt szabványokon alapuló feladatok kerülnek át Linuxra. A legtöbb döntésben – hogy Linuxot használnak más rendszer helyett – elsősorban a rendszer megbízhatósága játszott főszerepet, ingyenessége csak „hab volt a tortán”. A példák azt mutatják, hogy mára már az egymásra kölcsönösen támaszkodó üzleti és nyílt forráskódú programokkal jól működő üzleti hálózatot lehet kialakítani Linuxon is.

Kapcsolódó címek

A Fornax Rt. honlapja ➲ http://www.fornax-monitor.hu/
A TvNet Kft. honlapja ➲ http://www. tvnet.hu/

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

Pályázati felhívás

A Linux szélesebb körű elterjedésének legnagyobb gátja kis hazánkban, hogy kevés a magyar nyelvű leírás. A nagyobb grafikus rendszerek, a Gnome és a KDE többé-kevésbé magyar nyelvűek, de ez még kevés. Hiányzik egy olyan szabad irodai programcsomag, amely már a telepítéstől kezdve magyar nyelven tart kapcsolatot a felhasználókkal. Szükség van továbbá olyan ingyenes tankönyvsorozatra, amely a kezdő és a haladó felhasználóknak nyújt segítséget a Linux alkalmazásához.

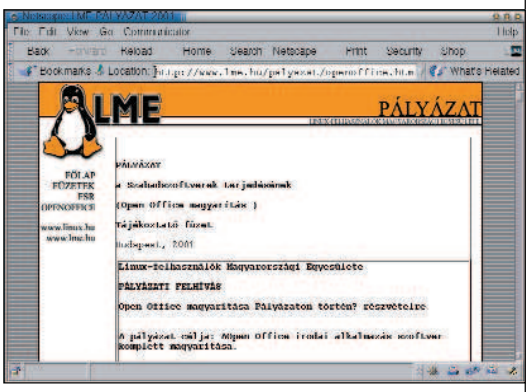
A helyzet javítása érdekében a Linux-felhasználók Magyarországi Egyesülete a Miniszterelnöki Hivatal támogatásával pályázatot hirdet. A pályázat három részből áll. Első része a *Pingvin füzetek* megírása hárommillió forintos díjazással. A tervek szerint az elkészülő 32 füzet a számítástechnika alapjaitól a Linux-rendszerfelügyeletig jutna el, összesen 1650 oldalon.

A pályázat célja a GNU/Linux operációs rendszer terjedését segítendő egységes, jól használható oktatási anyag elkészítése, amely egyaránt megfelel a számítástechnikával ismerkedő kezdők és haladók igényeinek. Pályázatokat várunk a füzetek megalkotására, akár egy-egy terület akár több terület együttes megírásához.

A pályázat második része a Fordítást Segítő Rendszer megalkotása, mely kétmillió forintos díjazással jár. A pályázat legfontosabb célkitűzése, hogy segítse a szabad programok magyarítását és az egységes szakszókincs megteremtését. A cél egy olyan webalapú alkalmazás kifejlesztése, amely a fordítási projekteknek egységes nyelvi hátteret teremt.

A pályázat harmadik része a StarOffice 5.2, illetve a leendő OpenOffice 6.0 teljes magyarítása célozza meg, kétmillió forintos keret erejéig. A pályázat célja egy szabad irodai programcsomag teljes magyarítása, mely segít a Linux irodai felhasználásának elterjedésében.

A Linux-felhasználók Magyarországi Egyesületének egyik legfontosabb feladata a szabad programok minőségi magyarításának és használatának ösztönzése. A pályázatok elbírálásánál ez a szempont kiemelt fontossággal



bír. Az egyesület számára külön öröm, hogy a Miniszterelnöki Hivatal segítségével támogathatja azokat a lelkes Linux-felhasználókat, akik hajlandók tevékenyen részt vállalni az egyesület céljainak megvalósításában.

A további tudnivalók e havi számunk második lemezmellékletén találhatók. A pályázatok beadási határideje: 2001. március 19. A pályázati részvétel természetesen ingyenes, minden jelentkezőt szeretettel várunk.

A MEH és az LME közös sajtóközleménye

A Miniszterelnöki Hivatal Informatikai Kormánybiztossága bruttó tízmillió forintos támogatást nyújt a kiemelkedő szakmai munkát végző Linux-felhasználók Magyarországi Egyesületének (LME). A támogatás fő célja az oktatás és a magyar nyelvű programok készítésének előmozdítása. Az Európai Unió ajánlása szerint „a szabad program kezdeményezés elkezdte átalakítani az informatika szabályait, ez az elkövetkező években további hatalmas változásokat hoz létre. A Linux-felhasználók Magyarországi Egyesülete az elmúlt két évben nagyon sokat tett a szabad program kezdeményezésért, jól működő civil szervezetté vált. Az LME legfőbb feladata a végfelhasználók megnyerése a szabad programok (többek között a Linux) irányában, ennek egyik legfontosabb lépése az oktatás és a magyarítás. Úgy gondoljuk, megérett a helyzet arra, hogy az állam támogassa a szabad programok kezdeményezését.” Az informatikai kormánybiztosság nevében *Kleinheincz Gábor* főcsoportfőnök jelentette be 2000. október 7-én, hogy a MEH első lépésként tízmillió forintos szerződést köt az LME-vel a szabad programok fordításának, oktatásának támogatására. A későbbiekben a kormánybiztosság az informatikai szerkezetfejlesztési programokban lehetővé kívánja tenni, hogy a felhasználóknak igény esetén módjuk legyen szabad programokat választani. *Varga Csaba Sándor*, az LME elnöke szerint, a kormányzati támogatás fordulópontot jelent az LME számára, hiszen a támogatás segítségével sokkal nagyobb lendülettel folytathatja tevékenységét.

Forrás: ➲ www.lme.hu/palyazat

A Hiltonból jelentjük

Az MSI kelet-európai körútja részeként február 20-án Budapesten is bemutatta saját és stratégiai partnerei, az nVidia, az AMD és a Seagate legújabb termékeit és fejlesztéseit.

MSI

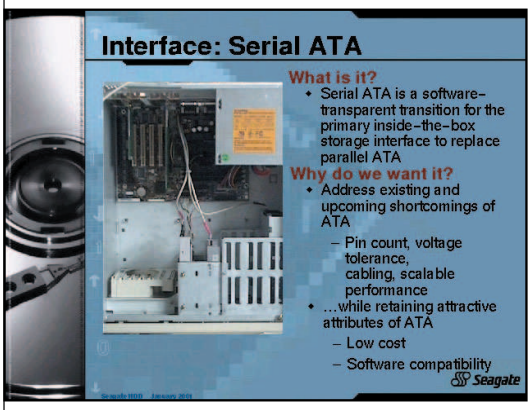
A MicroStar International a világ egyik legnagyobb OEM alaplapgyártó cége, melyet nagyon gyors fejlődés jellemez. Termékei tervezésénél és gyártásánál elsődleges szempontnak a megbízhatóságot tartja, így alaplapjai lehet, hogy nem a leggyorsabbak, de a legmegbízhatóbbak közé tartoznak.

Az alaplapok számos hasznos és különleges szolgáltatást is tartalmaznak. Ezek közül talán a legérdekesebb a PC2PC, ennek segítségével két számítógépet egyszerűen közvetlenül összeköthetünk USB-n, így nincs szükség hálózati illesztőkre. A másik nagyon hasznos szolgáltatás, a D bracket, amely egy PCI kártyahely hátlapján található. Négy LED van rajta, melyek az alaplap állapotáról nyújtanak adatokat a számítógép indulásakor.

A négy LED-nek összesen 16 állapota lehet, amelyek meghibásodás esetén utalnak a hiba jellegére, ezáltal megkönnyítik, felgyorsítják a hibakeresést és -elhárítást.

Az MSI mind Intel, mind AMD processzorokhoz készít alaplapokat. Közvetlenül a lapkagyártóktól kapja a lapkákat, így az elsők között próbálhatják ki és készíthetik el alaplapjaikat az új sorozatokhoz, az így nyert időt a minőség javítására fordítják. Legújabb alaplapjaik Intel processzorokhoz: 815EP Pro, Pro 266 Plus; AMD processzorokhoz: MS 6330 Lite, K7T 266 Pro-R. Legújabb terméktípusuk a grafikus kártyák, amelyeket nVidia lapkákra alapoznak. Termékkínálatukban a TNT-től Geforce2 GTS-ig (jelenleg a leggyorsabb grafikus lapka a piacon) lehet találni kártyákat.

A Seagate is újdonságokkal érkezett a bemutatóra, ezek közül a legérdekesebb a merevlemezek között a Barracuda család legnagyobb tagja, az egyelőre egyedülálló tárolóképességű 180 GB-os B180, valamint a Cheetah merevlemezcsalád másodpercenkénti 15 000 fordulatú új tagja, amely viszont a teljesítményével emelkedik ki a mezőnyből.



Szakmai téren a legkomolyabb fejlesztés a Serial ATA, amely a jelenlegi párhuzamos ATA sínt lesz hivatott felváltani. Ez hasonlít a profi SCSI megoldásokban használt száloptikás (Fiber Optic) csatolókhoz, amelyek igen nagy adatsebességet tesznek lehetővé, kis keresztmetszetű vezetéken. A Serial ATA ugyan nem száloptikára épül, de szintén gyors lesz, és legnagyobb előnye, hogy megszünteti a számítógépházakban jelenleg uralkodó kábelrengeteget, ennek köszönhetően javul a szellőzés, egyszerűbbé válik a szerelés, és nem utolsósorban csökken a vezetékek meghibásodásának lehetősége.

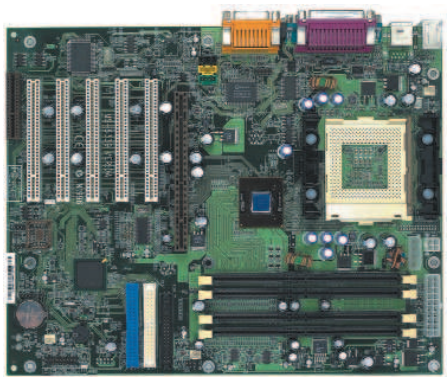
nVidia

Az nVidia a grafikus lapkakészletek piacát vezető cég, melynek lapkáit számos gyártó alkalmazza csúcskategóriájú grafikus kártyáihoz. Legfrissebb terméke a Geforce2 GTS GPU (Graphical Processor Unit), amely jellemzői alapján jelenleg a leggyorsabb a piacon. Bejelentették a Geforce3 piacra dobását is, az ezzel készült kártyák megjelenése hamarosan várható. Az nVidia széles körű támogatást nyújt a lapkáihoz, szinte minden operációs rendszerhez, így pl. a Linux, a Windows, a Solaris és a BeOS készít illesztőprogramokat.

AMD

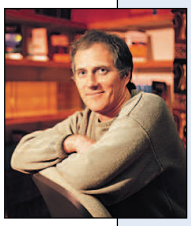
Az Intel nagy ellenlábasának legnagyobb előnye, hogy processzoraiban rézhuzalozási módszert használnak, amely kisebb ellenállású, így alacsonyabb hőmérsékletet, illetve nagyobb órajelet lehet elérni.

Jelenleg az Athlon és a Duron processzorcsaládokat gyártják, 2001 második felében várható az 1,4 GHz-es órajelű típusok megjelenése. Fejlesztés alatt áll a 64 bites processzoruk (X86-64), amely támogatja a 32 bites programokat is, de természetesen nagy előnye a 64 bites kialakításnak köszönhető teljesítménytöbblet. Az ígéretek szerint ez év végére piacra kerülnek az első példányok.



Intel 850-es lapkakészletű Pentium 4-es MSI alaplap

Beszélgetés Tim O'Reilly-val



Veszélyesnek tartom azokat a vállalkozási formákat, amelyek olyan erőforrásokat használnak ki, aminek a létrehozásában vagy gazdagításában nem vesznek részt.

Az ezredforduló utáni első Linux World Expón rögzítettem egy beszélgetést Tim O'Reilly-val. Tim számára az egyetlen érdekes játék az, ha előállhat egy látomással vagy látomások sorozatával, majd figyelheti, ahogyan ezek kibontakoznak – és láthatja, hogy ehhez a rendszer mely pontját kell megváltoztatnia.

Nemrég, meghallgattam a felvételt és úgy találtam, hogy amit mondott, ma is éppen olyan időszerű, mint annak idején – ha nem időszerűbb. Tim következetesen követi *Yogi Berra* lehetetlen tanácsát: „Ha leágazást találsz utadon, válaszd azt!" Visszatekintve Tim nem hagyott ki az útja során egyetlen leágazást sem. A választásai mögött azonban következetesen volt valami irányadó.

A beszélgetés tükrében elmondhatjuk, hogy Tim jóslatai kivételesen előrelátónak és pontosnak bizonyultak, főleg azért, mert egy valami érdekelt igazán, mégpedig az, hogy miért nem adta el a céget. Annak ellenére, hogy a kiadók nem a legnépszerűbbek a tőkés befektetők köreiben (akik többre értékelik az esetlegesen milliárdokat hozó cégeket a ténylegesen milliókat hozóknál), az O'Reilly mutatói régóta növekedésről és sikerről szólnak. Még fontosabb, hogy a programipar egyik legbefolyásosabb cégéről van szó.

O'Reilly-n nem mutatkoztak a céges hiúság jelei – legalábbis a többi céghez hasonlítva, melyek vagy szerepeltek már a tőzsdén, vagy a tőzsdei bevezetés felé tartottak. Ha más cégekhez viszonyítunk, az O'Reilly & Associates sohasem beszél önmaga eladásáról, a termékeiken és szolgáltatásaikon túlmenően. Az O'Reilly-t mindig is különösen tartózkodó pénzügyei jellemezték, igazi példaértékű magáncég, a szó legszorosabb értelmében. Tim már többször kifejtette nekem, hogy továbbra is ezen az úton szándékozik haladni.

Elgondolkodtam azon, hogy miért. A frissen tőzsdére került linuxos cégek számára 2000 a felfutás éve volt, a kiállítás idején még mindegyikük a befektetési láz hullámát lovagolta meg. A RedHat, a Cobalt Networks, az Andover és a VA Linux Systems, mind a tavalyi év utolsó negyedévében kerültek a tőzsdére, részvényeik roppant módon túlértékelődtek. Amikor Timmel beszélgettem, a VA Linux éppen felvásárolta az Andover.net-et (ez most a VA Linux OSDN részlege) több mint egymilliárd dollár értékű részvényért. Kevesebb mint két hónappal azelőtt, a VA tőzsdére való bevezetésekor jegyezték fel a legnagyobb elsőnapi árfolyam-emelkedést, a részvény 239 dolláron zárt, de a nap folyamán volt rá üzletkötés 320 dolláros árfolyamon is. A RedHat négy hónappal korábban jelent meg a tőzsdén, árfolyama 270 dollár körül tetőzött a VA bevezetésekor. Amikor Timmel beszéltem, a VA 270 dolláron állt, a RedHat pedig körülbelül százon. Annak ellenére, hogy ezek a számok jóval elmaradtak a kezdeti csúcsoktól, a két cég nagyon-nagyon kapós volt, és természetesen velük együtt a Linux is. Mára a linuxos befektetési hullám elült, békés, mint egy tó. A VA Linux részvényei napjainkban körülbelül 13 dollárt érnek, a RedHat részvényei nagyjából tizet. Eközben az O'Reilly & Associates vidáman halad tovább útján, úgy tűnik, ők voltak a legbölcsebbek, amikor nem adták el magukat, semmi pénzért.

Íme a Timmel folytatott beszélgetésem, mely még 2000. február elején készült.

Doc: *Az egyik haverom szerint, a kiállítók többet költöttek a pavilonokra, mint a világ összes linuxos cégének tavalyi bevétele. Ez szerinted is így van?*

Tim (nevetve): Nem hiszem, de mindegyik cégnek sok dolga akad, és sok elkölteni való pénze is.

Doc: *A Linux most nagyon izgalmas téma. Te fontos szereplő vagy ezen a területen. Nekem úgy tűnik, hogy könnyen eladhatnád a céget, ha akarnád, mégsem éltél a lehetőséggel. Miért?*

Tim: Ez igazából a második nagy robbanás, ennek mi is részesei voltunk. Az első a Web volt. A második a Linux és a nyílt forráskód robbanásszerű terjedése. Ha jobban belegondolok, voltak a Webet megelőzően is jelentős események. Lesznek a Linux után mások is. Elég régóta vagyunk a piacon ahhoz, hogy tudjuk, ezeket a folyamatokat sokkal inkább lehet hullámhoz hasonlítani, mint égbe szárnyaló töretlen vonalhoz. Az egyik hullám a másikra épül. Ha valamelyik hullámmal azonosulsz, rövid távon előnyökre teszel szert, de egyúttal bezárod magad a történelem egy adott állapotába. Lehettünk volna unixos cég, lehettünk volna PC-s cég vagy csoportmunka-programokra alapozó cég. Lehettünk volna tartalomszolgáltatók, internetes cég vagy most linuxos cég. Bízz bennem, ezek után is lesznek új hullámok. A Linux érdekes marad, de nem lesz olyan pezsgő téma, mint ebben a pillanatban. Nem is kell, hogy az legyen.

Doc: *Egyszerűen nem akarod magad beskatulyázni.*

Tim: Ez így van. Ezt a lehetőséget egyszerűen nem tartottam soha érdekesnek. Túl régi motorosok vagyunk. Jól végezzük a munkánkat, jól meghatározott szerepünk van, s ezt a szerepet játsszuk majd a továbbiakban is.

Doc: *Volt egy munkatársam, aki mindig azt mondta: „Ne mondd azt, hogy nem, inkább azt mondd: mennyiért?". Ez másképp megfogalmazva azt jelenti, hogy mindennek megvan az ára. Ha ez így van, és még mindig nemet mondasz – amikor olyan cégek, mint az Andover, melynek a múlt évi jövedelme csak töredéke volt az O'Reillyének, dollármilliárdokért kelnek el a tőzsdén – az azt jelenti, hogy az O'Reilly & Associates milliárdokat ér?*

Tim (nevetve): Merem remélni.

Doc: *Rendben, szóval szeretsz hosszú távon gondolkodni. Kifejtenéd ezt bővebben?*

Tim: Mint sokan mások, én is programozással kezdtem, és, mint sok programozó, én is láttam, hogy mekkora szükség van a jó szakkönyvekre. A céget 1978-ban indítottuk, eredetileg szakirodalom írására szakosodott tanácsadó cég voltunk. A nyolcvanas évek elején, amikor divatba jöttek a nyílt rendszerek, a könyveinket eladtuk különböző Unix-forgalmazóknak. Fokozatosan könyvkiadó lett a szakértő cégből.

Doc: *De ti többek vagytok egy egyszerű könyvkiadónál. Egyben meghatározó szereplői is vagytok annak a piacnak, amely számára kiadjátok a könyveket.*

Tim: Szeretünk részesei lenni ennek az üzletágnak, és mindannak, ami ezt az üzletágat megváltoztatja. Úgy vélem, a legtöbb cégnél jobban érezzük, hogy merre mennek a dolgok, hiszen a szakma szívverését érzékeljük. Ez nagyrészt azzal magyarázható, hogy régóta itt vagyunk, és sok szakterületet lefedünk.

Doc: *Nem kötődtök egy bizonyos szakterülethez vagy témához, vagy bármihez?*

Tim: Nem. Mindenről van könyvünk, amiről egy programozó, vagy más műszaki ember olvasni szeretne.

Doc: *De az is lehet, hogy tanácskozást szerveztek.*

Tim: Vagy olyasmit kezdünk el, mint a GNN, amit később eladtunk az AOL-nak.

Doc: *Ezt el is felejtettem.*

Tim: Van néhány dolog, amit én is szeretnék elfelejteni.

Doc: *Beszéljünk a szellemi tulajdonról, hiszen jól tudom, hogy foglalkoztat.* Jeff Bezosnak *köszönhetjük, hogy a programokkal kapcsolatos szabadalmak kérdése ismét előtérbe került. Neki sikerült szabadalmaztatnia az egykattintásos vásárlást, majd szabadalmi perbe fognia a* Barnes & Noble*-t. Sokat hallottunk téged erről a dologról nyilatkozni. Szeretném hallani, hogy mit jelent számodra a szellemi tulajdon.*

Tim: A szellemi tulajdon táptalaj. Nehéz létrehozni, de könnyű elpazarolni. Az ötletek, módszerek és lehetőségek talaja, amit bárki használhat. Ez gazdag talaj, amin üzleti vállalkozások is gyökeret ereszthetnek. Erre mindannyiunknak oda kell figyelni, és hozzá kell járulnunk a fejlődéséhez. De vannak olyan vállalkozási formák – főleg azok, melyek nagy felhajtást kerekítenek maguk köré és a tőzsdei befektetőkből tartják fenn magukat, igazi értékek szolgáltatása nélkül –, melyek nem dúsítják ezt a táptalajt. Ezek a vállalkozások kiszívják a tápanyagokat a rendszerből. Ha meg tudod győzni a fogyasztókat, hogy például nem kell fizetni a magas színvonalú műszaki adatokért, senki sem fog ilyet előállítani. Ezek a cégek kizsákmányolták a rövid távú lehetőségeket, ezzel egy csomó pénzt kihúztak a részvényesek zsebéből, ugyanekkor tönkre is tették azokat a feltételeket, amelyekre a sikerüket alapozták. Ez olyan, mintha azt mondanánk: „Hé, itt van ez a sok cucc, amihez majdnem ingyen hozzájutunk!" Úgy teszünk, mintha életképes vállalkozásunk lenne, pedig az előállítási költség alatt adjuk el a termékeinket. Ugyanez történik a természetes erőforrások kiaknázásakor a valós világban. Ugyan, a víz és a levegő ingyenes! Valaki egyszer megkérdezte *Ted Turnert*, hogy mire alapozná a vállalkozását, ha fiatalember lenne, azt mondta: „Vízre". És víztározókat vesz mindenfelé. Tudja, hogy van, aminek igazi értéke van, akkor is, ha jelenleg alulértékeljük.

Doc: *Gondolod, hogy ez a helyzet a programok által képzett szellemi tulajdonnal is?*

Tim: Egyszerűen veszélyesnek tartom azokat a vállalkozási formákat, melyek olyan erőforrásokat használnak ki, amelyek létrehozásában vagy gazdagításában nem vesznek részt. Ez kötelező jellegű a linuxos közösségben. Mindeddig a linuxos közösség jobban tartja magát, mint az internetes. Csatlakoztam az Internet-közösség Tanácsához 1995-ben, mert láttam, hogy az Internet nagy durranás lesz. Reméltem, hogy sikerül követniük a Sierra Club mintáját. Úgy gondoltam, a környezetvédő mozgalom jó példa lehet az olyan emberek számára, akik azt mondják: „Jogom van ezekhez a közös javakhoz. Felszólalhatok mellettük akkor is, ha nem birtoklom őket."
Itt van ez a közös örökség, az Internet kultúrájának, az internetes programok és a nyílt forráskód kultúrájának öröksége, amit a természetes erőforrásokhoz hasonlóan kizsákmányolnak. A felhasználóknak fel kell állniuk, és azt kell mondaniuk: „Ne tegyétek ezt velünk!"

Doc: *Tehát azt mondhatnánk az Amazonnak, hogy ne perelje be a Barnes & Noble-t a szabadalma miatt, győzze le őket más úton.*

Tim: Pontosan. Versenyezhetsz, de ne úgy tedd ezt, hogy közben elpusztítod az alapokat, amikre építesz. Megint azt kell mondanom, hogy a linuxos és a nyílt forráskódú közösség sokkal ügyesebb volt, mint az Internet-közösség öt évvel ezelőtt. *Linus* és a különböző linuxos cégek mindig odafigyeltek arra, hogy a közösség rendelkezésére bocsássák az eredményeket, hiszen ez a közösség attól a környezettől függ, amit ők hoznak létre. Sokkal inkább tudatosult bennük, hogy mi a közös. De nem hiszem, hogy ez olyan messzire ható, mint amilyennek lennie kellene. Hagyni kell az embereket, hogy gondolkodjanak ezen, hosszú időn keresztül. Nem tudom, hogy ismered-e *Stewart Brand* The Clock of the Long Now című könyvét. Ott voltam egy előadáson, amit *Brian Enóval* tartott, és amelyen Eno elmesélte beszélgetését egy nővel, aki egy gyönyörű házban lakott, egy lepusztult külvárosban. Amikor Eno megkérdezte, hogy milyen itt élni, a nő azt válaszolta: mesés. Ekkor Eno rájött, hogy a nő a szűk értelemben vett „itt"-ről beszél, és nem a tágabb környezetéről. Eno elmondta, hogy megértette, hogy a társadalomnak sok baja abból adódik, ahogyan meghatározzuk azt, hogy mivel törődünk, hogy mennyire helyi jellegű az, és gyakran elkerüli a figyelmünket a tágabb környezet. Rosszabb esetben mi magunk rejtjük el ezt a nem túl szép képet, eltakarjuk a mi kis valóságunkkal.

Doc: *Úgy gondolod, ez a hasonlat találó a jelenlegi helyzetre, a tőzsdei piaccal és az őt körülvevő nagyobb piaccal szemben?*

Tim: Igen. Sok vállalkozás rövid távra tervez, kevesek érdekeit veszi figyelembe. Nem töltenek el túl sok időt azzal, hogy megkérdezzék: Mi lesz ebből az egész kultúrának a haszna?

Doc: *Milyen messzire nyúlik vissza a nyílt forráskódú lelkiismeret?*

Tim: Sokat beszéltünk a nyílt forrású programokról, de az IBM PC ugyanekkora forradalom volt a maga idejében – sőt, még mindig nyílt forrású gép. Ettől lett másolható. A nyíltsága óriási fejlesztési hullámnak adott teret. De mi

A frissen tőzsdére került linuxos cégek számára 2000 a felfutás éve volt, a kiállítás idején még mindegyiküket a befektetési láz hullámát lovagolta meg.

történt ennek a felszínén? Létrejött egy új zárt birodalom, ami még zsarnokibb volt, mint az IBM a nagygépes időkben. Attól félek, hogy megint itt fogunk kikötni, a Linux ellenére. Évek óta mondogatom: attól, hogy egy programréteg szabad, még nem jelenti azt, hogy a következő réteg is az lesz. Várhatóan egyre több zárt forrású programot alapoznak majd szabad és nyílt forráskódúra, az új alkalmazások alapjába véve zártak lesznek. Nézd meg például az Amazont, vagy a MapQuestet, vagy az E-Trade-et. Ezeket a srácokat nem érdekli a nyílt forráskódú felhasználási szerződés. A GPL nem számít, ha nem terjeszted a programodat. Az E-Toys nem akarja eladni az e-kereskedelmi programját, mert nincs rászorulva, hogy ezt megtegye – de ez rendben is van. Ez ellen nincs semmi kifogásom. Megjegyezném azonban, hogy ezek a fejlesztések nem részei az ökoszisztémának. Nem adnak semmit vissza annak a közös talajnak, melyben mindannyian gyökerezünk.

Doc: *Jobban szereted a GPL-t, más felhasználási szerződéssel szemben?*

Tim: Amit nem szeretek benne, az a megszállott jellege. Úgy értem, hogy nem muszáj az embereknek GPL-t használniuk, de ez néhány embert el is riaszt a használatától. Véleményem szerint az a legfontosabb, hogy olyan kultúrát kell létrehoznunk, amiben a nyílt felületekre fejlesztő cégek megértik, hogy a saját érdekük a dolgokat mozgásban tartani. A Microsoftnak is több előnye származik abból, ha táplálja a Nyílt Forráskódú Közösséget, mint ha versenybe száll vele.

Doc: *Szerintem elkerülhetetlen, hogy az elkövetkező időben a Microsoft valamilyen módon támogassa a nyílt forráskódot.*

Tim: Lehetséges. De sok fontos pont van, amire az óriás cég támaszkodhat. Szerintem a Microsoft legnagyobb felismerése az volt, hogy léteznek ilyen szűk keresztmetszetek, és ha ezeket az átjárókat birtoklod, ellenőrizheted a piacokat, melyek ezektől függenek. És elhiheted nekem, sok ember gondolkodik hasonlóan az e-kereskedelem és a Web világában. Néhányuknak elég szilárd bástyákat sikerült építeni az ilyen fontos pontokra, ez menthetetlenül azt jelenti, hogy mások úgy táncolnak, ahogy ők fütyülnek. Aggódom amiatt, hogy ebben az új rétegben is megjelennek az olyan szereplők, akik a kulcspontok ellenőrzésével próbálnak érvényesülni. Lehet, hogy nem olyan a befolyásuk, mint a Microsoftnak, de a saját súlycsoportjukban pont olyanok.

Doc: *Mi a helyzet az olyan médiaóriásokkal, mint az AOL/Time Warner?*

Tim: Nem nagyon foglalkoztatnak a médiaóriások. Az már fontosabb, hogy hogyan férünk hozzá a széles sávú hálózathoz. A forrás számít: az, hogy milyen gyorsan építik ki, hogy milyen hozzáférést tesznek majd lehetővé, és hogy ki fogja az egészet ellenőrizni. De az az igazság, hogy a tevékeny termelő/tétlen fogyasztó modell már nem működik túl jól. Ma már mindenki kapcsolatban van mindenkivel. Akik az élvonalban vannak, egyre inkább úgy gondolnak a piacra, mint a folyamatos beszélgetések területére.

Doc: *A keresletet és a kínálatot azonos súlyúnak látjuk.*

Tim: Igen. És szeretjük az olyan műszaki megoldásokat, amelyek hasonlóképpen működnek. Biztosan érdekli a Linux Jorunal olvasóit, hogy amikor létrehoztuk a weblapunkat, mi valósítottuk meg az első Windows NT kiszolgálón futó webkiszolgálót. Miért tettük ezt mi, egy unixos cég? Mert láttuk, hogy abban az időben a Web megsértette ezt az alapvető modellt. Az böngészőprogramot használó emberek döntő többsége Windows-rendszert használt, de nem voltak windowsos kiszolgálók. Mindig azzal vicceltem, hogy a Web lesz a világ legnagyobb csak olvasható csoportmunka-rendszere. Hiszen a kialakuló helyzet leginkább erre utalt. Mi egyszerűen csak megpróbáltuk életben tartani a Webről alkotott egyenrangú szemléletet. Ez sikerült is.

Doc: *Mi lett a végeredmény?*

Tim: Hát nem lett nagy üzlet, de rákényszerítettük a Microsoftot a webkiszolgálók piacára. A sok rossz ellenére, ma már a fent említett felületet használó emberek is meg tudnak jelenni tartalommal a Weben. Korábban lehetett hallani az egyirányú eljárásokról, olyan emberektől, akik a Webet a saját elképzeléseik szerint akarták kialakítani, a párbeszédes modell helyett a televízió mintájára.

Doc: *Megpróbálták a Webet a jó öreg televízió történetének utolsó fejezetévé alakítani.*

Tim: Így van. Az NT kiszolgáló választása, mint webkiszolgáló, jó példája annak, hogy hogyan próbálunk meg egy kicsit „nagyobb léptékekben" gondolkodni. Ha azt mondtuk volna, hogy: „Mi elsődlegesen a Linux és a Unix irányába köteleztük el magunkat, és életünkben nem fogunk NT-t használni", azzal nem segítettünk volna senkin. Ehelyett azt mondtuk: „Hé, a felhasználók Windowst használnak, és átvágják őket!" Ugyanazokkal az eszközökkel kell ellátnunk őket, mint az a kultúra, melyet megpróbálunk építeni. És ez egy kulcsfontosságú fogalom. Miért adtuk ki a Windows 98 dióhéjban című könyvet? Mert meg akarjuk tanítani a windowsos felületet használó embereket arra, hogy mi módon gondolkodnak odaát, a Unixot használó tábor tagjai. Fel akarom ruházni őket hatalommal, és meg akarom mutatni nekik, hogy ők parancsolnak az operációs rendszernek. Ha elolvasod a könyvet, meglátod, hogy megtanít neked a unixos gondolkodásmódból annyit, amennyit ezen a csöppet félresikerült operációs rendszeren használni lehet.

Doc: *Hasonló üzeneteket viszel el a linuxos világba is arról, hogy mi a helyzet a külvilágban?*

Tim: A legfontosabb üzenetem a linuxos közösség számára a Web fontos szerepe. Legyen „a hálózat a számítógép" a jelszó. Ott mindenképp labdába kell rúgni. Kicsit aggaszt a az operációs rendszerek körül kialakult háború. Szerintem az igazi kérdés az, hogy hogyan fogunk játszani abban a világban, amely ezekre az operációs rendszerekre épül; bármelyik is legyen az. A Webre alapozott nyílt forráskódú projektekre is jobban oda kellene figyelnünk, nagyobb szerepet kellene vállalnunk bennük. Vegyük például az Apache-t, a SendMailt, a Qmailt. Még akár a MySQL-t is. Minden, ami a Weben alapul, igazán fontos. Fontos kérdés, hogy milyen lesz a jövő számítógépe. Szerintem egy óriási számítógépre fog hasonlítani. Tehát, hogyan

fejlesszük azokat az eszközöket, amelyek segítségével részt vehetünk ebben az egészben? A Collab.net-en például azt oktatjuk és kutatjuk, hogyan tudnak ez emberek hatékonyabban együttműködni. A Linuxról folytatott beszélgetésekből az derült ki, hogy a mozgalom a GNU projektben és a szabad programban gyökerezik. Szeretnék a usenetes gyökerekről is néhány szót ejteni. Azokról a gyökerekről, melyek az adatok megosztásának módjaihoz vezetnek vissza. Ha azt hisszük, hogy a beszélgetés egy adott program felhasználási szerződési modelljéről szól, és azt, hogy a Microsoftnak szabadon elérhetővé kellene tennie a programjai forrását, és hogy „mi" akkor fogunk győzedelmeskedni, ha „ti", egyszerű halandók szó nélkül követitek a papság utasításait, akkor mindannyiunknak annyi. A végén az egész a kapcsolattartás és a leleményesség kultúrájáról szól, ez pedig nem fölülről jön, hanem minden irányból. Bizonyos értelemben lehetőség nyílik egy új világ építésére. Túl sok embert látok, amint elhagyják ezt az új világot, és visszamennek dolgozni a régibe, vagy a réginek annak a részébe, amelyik arra a nyílt forrású új világra alapul, amit mindannyian építünk. Visszakanyarodtunk a beszélgetésünk elejére, amikor a pénzről beszéltünk. Ha nem hosszú távon gondolkodsz, a régi rend malmára hajtod a vizet.

Doc: *De nagy a kísértés – főleg pénzügyi okokból –, hogy rövid távra tervezzünk.*

Tim: Ez így van. És azok, akik ki akarják használni ezeket a lehetőségeket, nem szabad, hogy megfeledkezzenek a hosszabb távról, a közös érdekekről. Arról a világról, amit közösen építünk.

Doc: *És azért van lehetőség a pénzszerzésre is.*

Tim: Pontosan.

Miután kikapcsoltam a magnót, azt ajánlottam Timnek, hogy nézzük meg a Jabbert, az új, nyílt forráskódú, üzenettovábbító rendszert. Augusztusban, az általa szervezett nyílt forráskódú tanácskozáson, már a Jabberről tartott előadást, és nem sokkal később csatlakozott a Jabber.com igazgatótanácsához. Tim beszélgetése, amit a szabadalmakról folytatott Jeff Bezosszal, a Bounty Questté nőtte ki magát. Ez egy olyan webhely, ami összefogja a szabadalmakkal kapcsolatos változásokkal egyetértő közösségeket. Ez az ember pedig továbbra is a maga útját járja.

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

Elmozdulás a „miért?"-ről a „miért ne?" felé

Las Vegasban, a Comdexen a North Hallban mutatták be termékeiket az adatátvitellel foglalkozó cégek. Az internetes készülékek pavilonja közepén állított ki a Be, az az operációs rendszereket gyártó cég, amely nemrég állt át az internetes eszközök gyártására. Amíg a Be hősies erőfeszítéseket tett, a teremben más standokon kiállított termékekbe a Linux már teljesen beivódott, a „nagy újdonság" pedig teljesen nyilvánvalónak számított.

Az Internet Appliance nevű cég több polcnyi kiszolgálót mutatott be fejlett hibatűrő rendszerrel, erős tartalomvédelemmel, böngészőalapú kezelési és felügyeleti lehetőséggel és más nyalánkságokkal, amit csak remélhet az ember, hogy megtalál egy „kevesebb, mint harminczezer dollárért kapható" teljes adatközpontban. Fut Linuxon? Igen. Ez a nagy üzleti húzás? Nem.

Amíg a Be igyekezett mindenhol megjelenni, addig a Linux csendesen bemutatta új arcát: egyszerű árucikként jól eladható, alapvető operációs rendszerré vált. Álljon itt néhány Linux-alapú eszköz, amellyel a kiállításon találkoztam:

- Sony VAIO C1 PictureBook hordozható számítógép (Transmeta Crusoe processzorral)
- Snap Servers – hordozható összekapcsolt kiszolgálók a Quantumtól
- Agenda VR3 tenyérnyinél is kisebb digitális személyi titkár
- IBM NetVista N2200 vékony ügyfél
- Ericsson Nanorouter 2 átjáró és kiszolgáló
- Nokia Mediascreen Multimedia terminálok
- e-Appliance SuperScaler hálózati eszközök
- ZFLinux Devices – X86 lapkán (és lapkakészleten) beágyazott Linux
- a Gateway Connected Touch Pad internetkészüléke (szintén Transmeta Crusoe processzorral), mely elnyerte a ZDNet és a CNet „Best Consumer Product" díját.

Amikor elbeszélgettem a kiállítókkal, a legtöbben meg sem indokolták, miért Linuxot használnak árucikkeikben. Olyan volt, mintha azt kérdeztem volna tőlük, hogy miért használtak TCP/IP protokollkészletet, esetleg műanyagot a termékeikben.

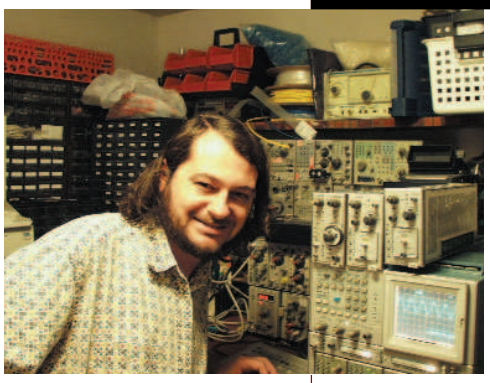
A beágyazott Linux fertőzővé vált. Ha éppen nincs kéznél jó téma a sajtóban, még mindig ott van gumicsontként: a beágyazott Linux gyors terjedése a legkülönfélébb eszközökben.

Egy csapat sráccal beszélgettem a Micronas pavilonjánál (a Micronas német cég, melynek multimédialapkái nagyon sok termékben megtalálhatók), akik szóba hozták a NetGem nevű új céget, amely szintén Linux-alapú termékeket gyárt (készülékeiket Európában a televíziós társaságok monitorként használják).

A két vezető európai cégről jut eszembe, hogy a Be kétségtelenül sok mindent tud ajánlani, csakhogy ezeket el is kell adnia. Ennél a pontnál pedig többé már nem annyira a Linux beágyazása okoz fejtörést. A közelmúlt során valamikor elmozdultunk a miértről a miért ne felé.

Doc Searls

Icarus – Tervezzünk áramköröket!



„Ebben a finomítási folyamatban a kód makrókra emlékeztet, és a tervezők egyszerűsíthetik is a feladatukat, ha RTL logikai könyvtárakat használnak.”

– Michael Baxter

A nyílt forrású programok egyre inkább túlmutatnak az operációs rendszerek, az internetes és az asztali alkalmazások, valamint a felhasználói felületek és parancsnyelvek világán. Két ilyen kevéssé ismert terület az elektronikus tervezéstámogatás (Electronic Design Automation – EDA) és az alkatrészleíró nyelvek (Hardware Description Language – HDL). A két leggyakrabban használt alkatrészleíró nyelv a VHDL és a Verilog. Ez utóbbit széles körben alkalmazzák logikai áramkörök tervezésére és utánzására a félvezetőiparban és más területeken.

A HDL-nyelvek az alkatrészek és gépek utánzására, összehasonlítására készültek, ez az adott felhasználástól függően különböző elvonatkoztatási szinteken történhet. E szintek tanulmányozásával megérthetjük, hogy miben különböznek a HDL-nyelvek a C, C++, Java és más, „hagyományos” programozási nyelvektől.

Egy „mi történik akkor, ha” típusú utánzás során a gép minden egyes részelemének működését megpróbáljuk a HDL eszközeivel jellemezni, tehát a feladatára vagyunk kíváncsiak, és nem arra, hogy ezt hány ellenállás vagy tranzisztor valósítja meg. Ez a fajta HDL-kód leginkább egy hagyományos számítógépprogramra emlékeztet.

A HDL-nyelvekben használt középső szint a Register Transfer Level (RTL). Ez a kód a regiszterek szintjén megvalósított szerkezeti felépítést írja le. A regisztereket flip-flopok vagy más logikai szerkezetek (például latchek – kallantyúk) alkotják, ezek különböző logikai tulajdonságokkal bírhatnak. A logikát leíró kód lehet viselkedésközpontú, de foglalkozhat azzal is, hogy mi kerül majd a tényleges áramkörre. Az RTL-t először a rendszer főbb elemeinek fölvázolásához, majd a végleges változat részleteinek kidolgozásához használják. Ebben a finomítási folyamatban a kód makrókra emlékeztet, és a tervezők egyszerűsíthetik is a dolgukat, ha RTL logikai könyvtárakat használnak. A logikai áramköröket a létező legalacsonyabb szinten is megtervezhetjük, így a kódban az áramkör pontos szerkezetét határozzuk meg. Ezt szerkezeti tervezésnek hívják, és leginkább a gépi kódú nyelvekre emlékeztet.

Egyetlen programnyelvvel minden szinten dolgozhatunk, és ezeket keverhetjük is egymással. A HSL nyelvek és a programnyelvek között van még egy nagy különbség: az idő. Bizonyos szempontból a HDL nyelvek valósítják meg tökéletesen a „programszámláló” elvét, hiszen ezeknél alapvető fontosságú az idő modellezése azért, hogy a logikai áramkörök viselkedését pontosan elemezhessük. Ebből következik, hogy a HDL-típusú és a hagyományos programnyelvek jelrendszere között óriási különbségek vannak. A HDL-nyelvek a párhuzamos rendszerműveleteket is tökéletesen kezelik. A teljesítmény növeléséért a HDL-fordítókat általában C vagy C++ nyelven írják meg. A fordító azonban a HDL-nyelv olyan elemeinek használatát is lehetővé teszi, melyek magvát a párhuzamos műveletvégzés alkotja, hiszen ez a jellemző igazán a gépre.

Mindezek eredményeképpen a Verilogot használó EDA eszközök a hagyományos programból megvalósított eszközöknél sokkal gazdagabb lehetőségeket kínálnak. Hogy mindezt még érthetőbbé tegyük, az alábbiakban közlünk egy beszélgetést *Stephen Williamsszel,* a GPL elvei szerint terjeszthető Icarus Verilog nevű EDA eszköz fejlesztőjével.

Michael: *Mi az Icarus Verilog fordító és hogyan működik?*

Stephen: Az Icarus Verilog az IEEE 1364-es szabványú Verilog alkatrészleíró nyelvhez készült fordító. Az EDA szakterületen jártas felhasználók nyilván felfedezik a hasonlóságot az Icarus és a VCS között, hiszen mindkettő széles körben elfogadott nyelv fordítója.

Működésének lényege, hogy a Verilog formanyelvét egy megjegyzésekkel kiegészített feldolgozási fába olvassa be, melyet azután egy tervezőgrafikonba „olvaszt be”.

E folyamat során történik a modulok példányosítása, a szimbolikus állandók kiértékelése és ismertetése, majd a rendszer ellenőrzi az eszközök csatlakoztatását. A folyamat végén egy kipróbált, működő rendszert kapunk.

A tervezőgrafikonból az optimalizálók és logikai összerendező eljárások egy új ábrát készítenek, mely a célnak jobban megfelel. Ezt végül a kódelőállító nézi át.

A folyamat végén a kódelőállító átvizsgálja a tervet és a kívánt formában menti a kimenetet. Az utánzás céljára C++ kód jön létre, ez az Icarus Veriloghoz tartozó különleges osztálykönyvtárra építkezik. A csomag egy XNF kódelőállítót is tartalmaz, ezzel a kész terveket más FPGA eszközökhöz továbbíthatjuk. Mostanában egy betölthető célkód-előállítón dolgozom, ezzel számos más célformátum támogatása is megvalósulhatna.

Michael: *Mikor kezdted az Icarus Verilog fejlesztését?*

Stephen: Jegyzeteim alapján 1998. novemberében került be a CVS-be a terv, de már legalább két évvel előtte megpróbáltam a munka beindítását. Ha emlékezetem nem csal, akkor már egy évvel a CVS-be kerülést megelőzően megtaláltam a követendő utat.

Michael: Mi ösztönzött arra, hogy az Icarus Verilog fejlesztésébe vágd a fejszédet?

Stephen: A szónoki válasz az lenne, hogy Linux/Alpha-rendszerem van. A legtöbb ember számára ez nem sokat mond. A lényeg, hogy az EDA-fejlesztők nem árasztják el programokkal a Linux/Alpha-rendszert használókat, de még a Linux/Intel felület sem kap megfelelő figyelmet. Egyszerűen nem tudom elképzelni, hogy az a sok EDA-fejlesztő miért szereti annyira a Microsoft termékeit.

A másik ok már valamivel összetettebb. Azért kezdtem neki, mert tudtam, hogy képes leszek rá, és jó munkát fogok végezni. Képességeim elegendőnek tűntek egy ilyen feladat végrehajtásához, ráadásul tovább bővültek az ismereteim. Úgy kezdtem bele, hogy sokkal többet tudtam a lapka- és az FPGA-tervezésről, mint a programmérnökök legtöbbje.

Michael: *Miért ragaszkodsz a nyílt forrású fejlesztéshez?*

Stephen: Szeretnék hozzáférni saját szellemi tulajdonomhoz. Régebben rengeteg nagyobb fejlesztési munkában vettem részt, ezek eredményeit azonban munkahelyvál-

tás vagy különleges megállapodások miatt nem használhattam föl. A jelenlegi munkáltatóm azonban semmit sem kötött ki az Icarus Verilog fejlesztésével kapcsolatban, tehát elfogadja, hogy ez az én munkám, amit a saját időbeosztásomban végezhetek. Ezt a szerzői jogi rendelkezések is nyilvánvalóvá teszik, és eddig úgy tűnik, hogy ez így minden résztvevő számára biztonságosabb.
A munkaadóm üzleti terveinek mellőzésével az egészet készíthetném zárt forrású, személyes munkaként is, de mi lenne abban az élvezet? És azt se felejtsük el, hogy a nyílt forráskódnak köszönhetően egyre több hasznos hibabejelentést kapok a programmal kapcsolatban. Néha még egész programrészletek is érkeznek, melyek új lehetőségeket valósítanak meg.

Michael: *Kik segítettek neked a legtöbbet?*

Stephen: A csomagban található README fájlból mindez kiderül, de talán a próbaüzem felállításában segédkező *Steve Wilson* segítsége nélkül lett volna a legnehezebb. Sokszor mondták, hogy a kipróbálás a fordítók fejlesztésének legfontosabb része, és eddigi tapasztalataim alapján igazat kell adnom mindenkinek.
Ismétlem, a felhasználóktól kapott hibabejelentések is óriási segítséget jelentettek. Ezek mindig jó minőségűek, részletesek, és a legtöbb esetben időszerűek. Nagyon-nagyon ritkán fordult elő, hogy egy felhasználó által beküldött hibabejelentés kivizsgálásakor az derült ki, hogy a hiba mégsem az Icarus Verilog „készülékében" van. Ha mégis, akkor általában valami egyszerű félreértésről volt szó.
A hibabejelentések és a változtatási kérelmek, javaslatok alapján döntöm el azt is, hogy mi lesz a fejlesztés következő lépése.

Michael: *Melyek az Icarus jellegzetes felhasználási területei?*

Stephen: Nos, erre nehéz válaszolnom, hiszen nem áll a hátam mögött egy piackutató részleg, mely ennek kiderítésével foglalkozik. Legfőbb forrásom e tekintetben is a felhasználóktól – általában hibabejelentések formájában – érkező visszajelzések.
Úgy tűnik számomra, hogy a nagy intézményekben dolgozó felhasználók nem férnek hozzá a méregdrága HDL-fejlesztőkörnyezethez, így az Icarus Verilog segítségével otthoni linuxos gépükön dolgoznak különböző könyvtárakkal és alrendszerekkel.
Számos olyan esetről hallottam, amikor valaki az Icarus megjelenése következtében anyagi vagy elvi okokból abbahagyta a díjkötelezett eszközökkel végzett munkát. A szerényebb igényű HDL-felhasználók azért szeretik az Icarus Verilogot, mert céljaiknak tökéletesen megfelel, az árversenyben pedig verhetetlen. A csomag azok kezébe adja a HDL-es tervezés lehetőségét, akik máskülönben biztosan nem engedhetnék meg magunknak.

Michael: *Összehasonlítható-e az Icarus egy „fizetős" EDA-eszközzel?*

Stephen: Az Icarus Verilog nem jelent veszélyt a minőségi, márkás eszközökre, hiszen ezek fejlesztői nálam sokkal hamarabb kezdték a munkát. Nemrég még azt gondoltam, hogy én a kis Icarusommal föl sem vehetem velük a versenyt. Manapság azonban egyre több olyan fizetős eszközt találtam, amelyeknek piaci jelenlétét szerintem az égvilágon semmi sem teszi szükségessé. A különbségek a nyelv elemeinek megvalósításában, az utánzás teljesítményében és az összerendezés (synthesis) minőségében jelentkeznek. Ami a nyelv megvalósítását illeti, az Icarus Verilog aránylag jól áll a versenyben, és a fejlődés e téren folyamatos. A jelenlegi szint pedig átlagosnak mondható.
Az utánzás teljesítményét nagymértékben rontja a g++ fordító, amely az előállított utánzást fordítja le. Attól tartok, hogy hiba volt a C++-t választanom közvetítő nyelvként, szóval hamarosan sima C-re vagy valami másra cserélem le. A terv a fordítás után azonban meglepően jól és gyorsan működik.
Az Icarus Verilogban megvalósított összerendezés még nem üzleti minőségű, bár jónéhányan eredményesen használják. Az Icarus Verilog összerendezőjének határain belül maradva akár Xilinx-típusú terveket is készíthetünk. Tudok olyan esetről is, hogy valaki az Icarusszal helyettesítette az Abelt a tervezési folyamatban.

„A nyílt forrás hatásaként rengeteg értékes hibabejelentést kapok. Rendkívül örülök a felhasználóktól érkező hibabejelentéseknek."

– Stephen Williams

Michael: *Létezik-e olyan feladat, amit az üzleti EDA-eszközökhöz képest az Icarusszal nehezebb megvalósítani?*

Stephen: Először is, az Icarus Verilog nem képes mindenre, amire egy EDA-felhasználónak szüksége lehet. Természetesen képes XNF-ben menteni, de a használni kívánt rész feltérképezéséhez és optimalizálásához a gyártótól függően más és más eszközöket kell igénybe vennünk. Nagyon nehéz beszerezni a gyártóktól a megfelelő adatokat, ezért a Netlist formátumokat kell használnom, ez pedig igencsak kiábrándító. Az is bosszant, hogy Linuxra nem léteznek mélyebb rétegekkel foglalkozó eszközök.

Michael: *Az Icarus Verilog nyílt forrású eszmeisége jelent-e valamiféle előnyt az üzleti eszközökkel szemben?*

Stephen: Az Icarus legnyilvánvalóbb előnye, hogy rugalmas munkakörnyezetet teremt. Ha egy terv működik az Icarusszal, akkor egy új számítógép vásárlása esetén is biztosak lehetünk abban, hogy azon is működni fog. Megfigyeltem, hogy az EDA-fejlesztők korlátlan időre szóló szerződéseket hirdetnek, a programot futtató operációs rendszerre és a számítógépre azonban ez nem igaz. Ráadásul X gyártó nem támogatja Y termék 1.0-s változatát a frissen vásárolt gépünkön. Ebből következik, hogy új gép vásárlásakor frissítéseket is kell vennünk, ez nemcsak méregdrága mulatság, hanem még kockázatos is, mondjuk egy 95 százalékban kész XC4013XL tervezés esetén. Láttam már olyat, hogy egy eszköz újabb változata tönkretette a majdnem kész tervet.

Michael: *Az üzleti EDA-fejlesztők lassan közelednek a Linuxhoz, bár néhányan már használják. Azonban kivétel nélkül viszolyognak a nyílt forrású fejlesztések befogadásától vagy indításától. Vajon miért?*

Stephen: Nos, azt hiszem, a cégek számára elég rázós lenne egy százezer dolláros program kifejlesztése után

„Egyszerűen nem tudom elképzelni, hogy számos EDA-fejlesztő miért szereti annyira a Microsoft termékeit."

– Stephen Williams

a nyílt forrású fejlesztésről ódákat zengeni, egyébként meg nem is tudom. Minden munkatársam nap mint nap panaszkodik, hogy Microsoft operációs rendszerekkel kell dolgozniuk, mert nincsen más választásuk. Az FPGA-gyártók kifejezetten mogorvák e tekintetben. Talán, ha majd ott is beköszönt a nyílt forrású fejlesztés tavasza...

Michael: *Mennyi kód van az Icarusban?*

Stephen: 50 000 C++ nyelvű sor, egy kis C-vel, lexszel és yacc-cal megtűzdelve. Úgy tűnik, hogy jó néhány hónapja nem hajlandó ennél nagyobbra nőni... De komolyra fordítva a szót: eljutottam arra a szintre, hogy legalább annyi kódot távolítok el belőle, mint amennyit hozzáteszek. Van egy kis próbalabor is, ahol 300 kisebb Verilog-ellenőrzés folyik, ez összesen 16 000 Verilog-sort érint. Ezeket egy kis Perl vezérli.

Michael: *Van valami ötleted, javaslatod a nyílt forrású programfejlesztők számára, mely elősegíthetné az Icarushoz hasonló munkák elindítását?*

Stephen: Nos, jó lenne, ha a g++ mondjuk, tízszer ilyen gyorsan fordítana. Nem is tudom, miért panaszkodom, hiszen például az MSVC++ egyszerűen képtelen lefordítani a fordítómat. A Linux/Alpha és a Cygwin32 binutils esetében a szimbólumtáblákkal is akadtak gondok.

Michael: *Az Icarus fejlesztésének alapját a Linux képezi. Nehéz feladat volt más operációs rendszerekre átültetni?*

Stephen: Segítőim az utolsó működő változat előre fordított bináris fájljait átültették Solarisra, NetBSD-re és MacOS-re is. Nemrégiben egy windowsos változat is elkészült, ezt a Cygwin32 Net kiadásának köszönhetjük. Az átalakítások legnehezebb részét mindig a dinamikus kapcsolások képezték – a HP/UX különösen makacs természetűnek bizonyult e téren –. Azért akinek van egy korszerű C++ fordítója, meg egy make programja, annak nem sok baja lesz az Icarus Verilog fordításával. A FAQ oldalon igyekeztem felsorolni a gyakoribb hibákat és azok kiküszöbölésének módját.

Michael: *Megemlítenél néhány olyan nyílt forrású EDA-eszközt, amelyek érdekelhetik az Icarus felhasználóit?*

Stephen: Ott van például a gEDA csomag, ennek honlapján ➲ http://www.geda.seul.org/, fellelhetünk számos érdekes EDA-eszköz oldalaihoz vezető hivatkozást. A GTKWave egy szintén értékes hullámforma-megjelenítő. Jól használható az Icarus VCD kimenetének tanulmányozására. Az Electronic VLSI Design System ➲ http://www.staticfreesoft.com/ is nagyon érdekes, és ami a legszebb, hogy Linux/Alpha alatt is működik. Egy mutatós EDA-lista található az Open Collector weboldalán ➲ http://www.opencollector.org/. Aki itt szétnéz egy kicsit, az rengeteg aranyos dolgot találhat.

Michael: *Tervezed-e, hogy az Icarust más HDL-nyelvek támogatásával is kiegészíted?*

Stephen: Egyelőre nem. A Verilog szabványosítási folyamattal kapcsolatos új adatok beépítése is éppen elég feladatot ró egyetlen emberre. Ha esetleg valamilyen véletlen folytán mégis „utolérném" a nyelvet, a fordításban olyan új nehézségek adódhatnak, melyek egy jó darabig ellátnak munkával.

Michael: *Hogyan képzeled el az Icarus jövőjét?*

Stephen: Ugyanazt szeretném látni, ami a gcc esetében is történt. Ha valaki idejön, és azt mondja, hogy sok pénzt fizet, ha folytatom az Icarus Veriloggal elkezdett munkát, annak nagyon fogok örülni...

Michael: *A munka mely részterületein van szükséged segítségre?*

Stephen: Kódelőállítókat kellene készíteni! Nagyon sok fáradságot jelent a kódelőállítók által használt API-k csiszolgatása. Legmerészebb álmom, hogy készítek néhány alapvető kódelőállítót, majd a segítőim az összes létező Netlist formátumhoz és utánzómotorhoz megírják a megfelelő kódelőállítót. A gcc is körülbelül így működik. Kipróbálók kellenek! Hiányukat leggyakrabban a hibabejelentések elemzésével hidalom át, de sokszor célszerűbb, hogy külön próbakörnyezetbe helyezzek egy-egy frissen kifejlesztett elemet. De nyilván én nehezebben találok meg egy hibát a saját kódomban.

Michael: *Véleményed szerint a nyílt forrású eszközök milyen szerepet játszhatnak az EDA-eszközök fejlődésében?*

Stephen: Nem igazán tudom, hiszen nem vagyok jós. Annyi bizonyosnak látszik, hogy hatásukra emelkedni fog az üzleti csomagok színvonala. Úgy vélem, a nyílt forrású eszközökben megvan a lehetőség arra, hogy a régi tervekkel is dolgozhassunk. A terv forrását tárolhatjuk, de a régi eszközök tárolásának semmi értelmét nem látom.

Michael: *Ikarus élete gyászos véget ért. Számodra van valami jelentősége a névválasztásnak?*

Stephen: Meglepődnél azon, hogy mennyire kevesen ismerik a történetet. „Icarus, hogyan kell leírni?" – általában ez az első kérdés. A név inkább apró utalás, nem a jelentése a lényeg. Én eredetileg programmérnök vagyok, nem géptervező. Na jó, tudom kezelni az oszcilloszkópot meg a logikaelemzőt, és 160 pontos csatlakozók tűire is forrasztgattam huzalokat, ennek ellenére én csak egy programmérnök vagyok, aki túl közel repül a Naphoz. Sokszor mondták, hogy egy Verilog-fordító elkészítése nagyobb munka, mint ahogy én azt képzelem. Ezt általában géptervező mérnökök mondogatták. De mostanában már nem hallok ilyeneket. Tudom, hogy korábban milyen tudásbéli hiányosságokkal kellett szembenéznem, de mára azért változott a helyzet.

Michael: *Mesélnél valamit a logóról?*

Stephen: Természetesen. Steve Wilson lényegesen részletesebben tudna róla mesélni, de röviden összefoglalva Steve nagybátyja, *Charles Wilson* nyugdíjas grafikus rajzolta. Művét ingyen felajánlotta az Icarus Verilog számára, és ezért én igen hálás vagyok neki. Soha nem használtunk más jelet. Ez újabb bizonyíték arra, hogy a nyílt forrású mozgalom a számítógépprogramok körén kívüli területeket is meghódíthatja.

Michael Baxter 1969-ben látta a 2001. Űrodüsszeia filmváltozatát, és azóta, azaz kilencéves korától foglalkozik számítógépekkel. Tapasztalt számítógépes mérnök: rendszer-, kártya- és FPGA logika-tervező. Michaelnek tíz bejegyzett szabadalma van.

Ismét a Linux nyert

Az Omaha Steaks még az internetes vásárlások csúcsidőszakának tartott karácsonyi hetek előtt elhatározta, hogy a várható hatalmas forgalom minél jobb kiszolgálásához átalakítja webáruházát. Az eOne Group nevű omahai webáruház-készítő és szolgáltató cég segítségével olyan honlap kiépítésébe kezdtek, mely helyből üzemeltethető, adatbázisokkal dolgozik, és a megrendeléseket rugalmasan (akár több címre is) képes kezelni.
A honlap alapjául az eOne saját fejlesztésű, gépfüggetlen, Java-alapú alkalmazása, a jCommerce szolgált. Nyílt rendszer lévén a jCommerce bármilyen operációs rendszeren működik, és a felhasználói igényeknek megfelelően egyszerűen átalakítható. Mivel a jCommerce valósította meg az Omaha Steaks által igényelt adatbázis-kezelést, és támogatta az XML és XHTML használatát, ezért a program kiválasztása nem jelentett nagy gondot. Az egyetlen eldöntetlen kérdés az maradt: milyen operációs rendszert használjanak.
Az Omaha Steaks hagyományos családi vállalkozás, mely 1917 óta működik. *Chad Bukowski*, az eOne vezető szakembere szerint kicsit tartott attól, hogy a vállalat hogyan viszonyul majd az eOne által oly kedvelt Linux nyílt forrású szellemiségéhez, de tudta, hogy a végső döntést mindenképpen az ár/teljesítményviszony határozza meg. A próbaüzemet több rendszeren, így IBM RS/6000 M80, AS/400 és RedHat Linux 6.2-t futtató Dell Intel gépeken is elvégezték. A kipróbálás során az AS/400 lelassult, ha az önálló, egyidejű felhasználói rendelések száma meghaladta az ötvenet; az RS/6000 úgy 150-200 felhasználóig bírta. A Dell gépek – Bukowski szerint – 250-400 egyidejű kérelmet is képesek voltak feldolgozni, különösebb teljesítménycsökkenés nélkül.
A Linux kiemelkedően jó eredményei láttán, ezt a rendszert választotta a cég. *Jeff Carter*, az Omaha Steaks vezetője szerint négy kedvező tényező hatására született a sikert hozó döntés:

- a Dell és az IBM rendszerek ára közötti hatalmas különbség miatt (egy Dell gép 8000, míg egy IBM 250 000 dollárba került, és ez utóbbhoz még a felhasználási szerződés alapján fizetendő díjak is hozzászámítandók);
- a Linux közkedvelt és kiemelkedően jó teljesítményt nyújt az Interneten;
- megbízhatóbbnak tartják a szintén Intel-alapú gépeken futó Windows NT-vel szemben;
- hatékonyan együttműködik a régi rendszerekkel is.

Az új honlap két hónapos próbaüzemet követően tavaly év végén nyílt meg. Ez ideig mind az Omaha Steaks, mind pedig az eOne elégedett az eredményekkel. Újraindításra még nem volt szükség, és Bukowski szerint „úgy, dalol, mint egy kismadár". A vásárlóknak tetszik az egyszerűsített rendelés, és az Omaha Steaks alkalmazottai is hamar megszokták az új rendszert. Carter különösen előnyösnek tartja, hogy a honlap üzemeltetéséhez elegendők a belső erőforrások: a kívülről paraméterezhető webkiszolgáló karbantartásához nincsen szükség javás programozókra. Carter így nyilatkozott az Omaha Steaks, az eOne Group, az eCommerce és a Linux kapcsolatáról: „A piacon senki más nem tud egy ilyen kellemes megoldást szállítani."

➲ http://www.omahasteaks.com/
➲ http://www.eonegroup.com/

Pingvinmentők

Az ausztráliai Pingvin Nemzeti Park dolgozói továbbra is ápolják az olajbalesetekben vagy más körülmények között megbetegedett pingvineket. A szervezet felkérte a segíteni szándékozókat: küldjenek minél több pingvinméretű kötött pulóvert, hogy kis frakkos barátaink gyógyulásuk közben ne fázzanak.
Az ausztráliai pingvinek fotói megtekinthetők:
➲ http://www.penguins.org.au/*webhelyen.*

Új linuxos oldal spanyolul értőknek

A Piensa.Com Systems nemrég indította a La Gaceta de Linux című újságot. Ezen az oldalon érhető el a Linux Gazette spanyol kiadása is. A Gaceta de Linux célja, hogy friss hírekkel lássa el a spanyolul beszélő linuxos közösséget. A honlap születésében fontos szerepet kaptak azok a vállalkozó kedvű fordítók, akik néhány nap alatt tökéletes minőségű fordításokat készítenek. „Elkötelezett hívei vagyunk a spanyol anyanyelvű Linux-felhasználók és a linuxos üzleti alkalmazások támogatásának" – mondta *Felipe Barousse*, a BCM-Piensa.Com Systems igazgatója. „Hisszük, hogy a Linux lehetővé teszi a csúcsmódszerek olcsó és megbízható terjesztését, olyan esetekben is, ahol nincs más választási lehetőség. A La Gaceta de Linux éppen ezért nagyszerű eszköz a kezünkben. Mire ezt olvassák, addigra a Gazeta do Linux nevű portugál változat is elérhető az Interneten."
A fordításokat önkéntesek készítik, és minden cikket szakértők néznek át, tiszteletben tartva a szerző jogait és véleményét. Bejegyzett látogatókat 22 országból tartanak nyilván, és a kezdés óta már száznál is több cikket fordítottak le. A belépéshez látogassunk el a ➲ http://www.gacetadelinux.com/register oldalra.

Zavaros a kép?

A Unix System V Release 3.0 rendszermagjának 80386-os, vagy annál fejlettebb processzorral rendelkező PC-k számára kifejlesztett változata… A Linux rendszermag az FSF által készített GNU segédprogramokkal működik.
Részlet a Microsoft Press Számítástechnikai Szótárból

Gyorsuló idő

Tavalyelőtt még senki nem hallott a Google-ról. Időközben a keresőről kiderült, hogy tulajdonképpen a linuxos közösség „ajándéka" a linuxos közösség számára, és már mindenki ismeri. Így az sem jelentett nagy megrázkódtatást, hogy a Google most már a Yahoo webkeresőjét is kiszolgálja. Bár a Google továbbra is egy jó kereső marad, a vállalat azon döntése, hogy a keresési módszereit szabadalmaztatni kívánja, nem aratott sikert a szabadalmaktól elvből viszolygó linuxos közösségekben.
Nos, nem baj: nem egy olyan kereső található még a Weben, melyet linuxos-unixos elkötelezettség jellemez. Ezek közé tartozik a Fast Search és a Transfer ASA (FAST), melyek a ➲ http://www.alltheweb.com/ címen érhetők el. Ez a norvég vállalat több irodát is üzemeltet az Egyesült Államokban, és a Dell céggel kötött megállapodást. A két vállalat eszerint közös erővel dolgozik a világ legjobb és leggyorsabb keresőjén.
Múlt év elején a Lycos komoly befektetést hajtott végre a FAST cégben, és mára a FAST négy alapvető keresőjének (FAST Web Search, FAST FTP Search, FAST MP3 Search és FAST MultiMedia Search) résztulajdonosa. Ezeket a ➲ http://www.alltheweb.com/, illetve a ➲ http://www.lycos.com/ címen találjuk, és mindkettő ugyanazt a keresőmotort alkalmazza.
A FAST motorja FreeBSD-n fut, fejlesztéséhez FreeBSD-t és Linux-rendszereket alkalmaztak. Valójában a FAST első motorját, az FTP Searchöt a Free Software Foundation GPL szerződése alapján terjesztették, és ez a változat ma is letölthető az ➲ ftp://ftpsearch.ntnu.no/pub/ftpsearch/ címről. Az eredmények megjelenítéséről az Apache és a PHP gondoskodik. A linuxos kötődésekről még annyit, hogy a FAST emberei részt vettek a PHP kidolgozásában is, sokan közülük pedig a trondheimi egyetem Unix-lelkületű számítógépes klubjából („Program-vareverkstedet", ➲ http://www.pvv.org/) érkeztek.
A FAST által eladott termékek zárt forrásúak (ez a keresőmotorra is vonatkozik), egyébként ez a helyzet minden mai webes keresővel. Ha bárki tudna nyílt forrású keresőről, kérjük, tudassa velünk…
A FAST által alkalmazott eljárásokról részletesebben olvashatunk a vállalat honlapjának „Technology" menüpontját választva.

A keresőkkel kapcsolatos tavaly év végi hírek között szerepelt a Yahoo novemberi húzása is: a cég üzleti és gazdasági tárgykörébe tartozó „Business to Business" (cégek közötti) és „Shopping and Services" (vásárlás és szolgáltatások) nevű csoportjaiba való belépés lehetőségét reklámozta. 199 dollárért a Yahoo a Business Express programban részt vevő cégeknek azt ígéri, hogy az általuk beküldött ismertetőket azonnal elemzi, és ha a kapott adatok megfelelőek, akkor elhelyezi azokat a két csoport valamelyikében.
A ➲ http://docs.yahoo.com/info/suggest/faq.html címen található kérdések és válaszok oldalon azt írják, hogy az elfogadó vagy elutasító döntésről hét munkanapon belül értesítik a cégeket. Elutasítás esetén ennek okát is közlik, és a felhasználó fellebbezhet a döntés ellen.
Eközben az Open Directory Project ➲ http://www.dmoz.org/ gyors növekedésbe kezdett. Néhány, keresés alapján elmondhatjuk, hogy a két szolgáltatás egymás komoly vetélytársa lehet. A Yahoon csak azzal „segíthetünk", ha a cégnek dolgozunk, vagy fizetünk a listáért. Az Open Directory Projecthez azonban mi magunk is rengeteget tehetünk hozzá: kedvenc témánk szerkesztőjévé is válhatunk, ehhez mindössze a minden témánál megtalálható hivatkozásra kell kattintanunk.

Ők mondták

A rosszindulatot soha nem lehet pusztán gyengeelméjűséggel magyarázni. *(Joseph E. Arruda)*

A fekete lyukak akkor jöttek létre, amikor Isten nullával akart osztani. *(Steven Wright)*

Lehet, hogy a sas tud repülni, viszont a menyét nem repül bele egy repülőgép hajtóművébe. *(ismeretlen a Slashdoton)*

A minden varázslattól megkülönböztethető tudománynak még van hova fejlődnie. *(Don Marti – amennyire mi tudjuk)*

Minden fejlődést erősen bíráló gondolkodás előz meg. A fejlődés a kultúra terjedése és a gondolatok áramlása olyan emberek között, akik először vonakodnak meghallgatni azokat, majd elősegíti a saját gazdasági és politikai gondjaik megoldását. *(Antonio Gramsci)*

Nem arra van szükség, hogy higgyünk valamiben, hanem hogy felderítsük azt, ez pedig éppen az ellenkezője. *(Bertrand Russell)*

Az csak egy tévhit, hogy az emberek ellenállnak a változásnak. Az emberek mások akaratának nem szívesen engedelmeskednek, ők a saját elképzeléseiket szeretnék megvalósítani.... Ez az, amiért sikeresek azok a fejlesztő vállalatok, amelyek éveken át figyelemmel kísérik az emberek ötleteit, és lehetővé teszik számukra új tervek megvalósítását, hogy még több tapasztalatra tehessenek szert. *(Rosabeth Moss Kanter)*

A hosszú távú tervezés nem a jövő döntéseiről, hanem a jelen döntéseinek jövőjéről szól. *(Peter F. Drucker)*

Nem számíthatunk saját ítélőképességünkre, ha nem használjuk képzelőerőnket. *(Mark Twain)*

A felfedezés az, amikor valaki olyasmit lát vagy gondol, amit addig még senki. *(Szent-Györgyi Albert)*

Az Internet soha nem hátrál meg. *(Vint Cerf)*

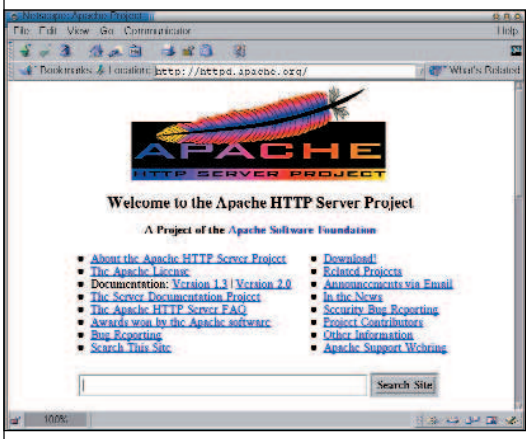
Ha eleve lehetetlen dolgot próbálunk csinálni, az csakis egy sikertelen vállalkozást eredményezhet. *(Michael Oakshott)*

Az unalmas emberek a legjobb vásárlók. *(John Taylor Gatto)*

Csak nem értik, ugye? Az összes Linux-alkalmazás fut Solarison, ez a mi Linux-változatunk. *(Scott McNeal)*

Az Apache fejlődik

A Netcraft cég havonta megjelenő felmérésében az Interneten HTTP-szolgáltatást nyújtó gépeket hasonlította össze. Az Apache 1995 közepén néhány százalékos részesedéssel nyitott a piacon. Ugyanazon év végére az Apache, az NCSA cég és az „egyéb" csoport összesen már harminc százalékot mondhatott magáénak. Ekkor jelent meg a Microsoft IIS (e-üzleti megoldás) is. Azóta az Apache és az IIS áll az első két helyen, de az Apache előnye jelentős. 1997. decemberében az IIS-t az „egyéb" csoporttal közösen értékelték, és így a második helyet érte el. A piacvezető Apache ugyanekkor már önmagában elérte az ötvenszázalékos részesedést.

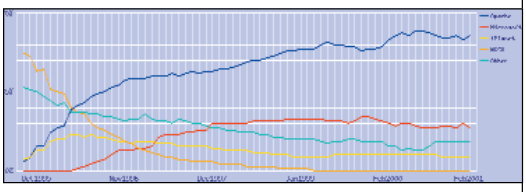


Az Apache webkiszolgáló honlapja

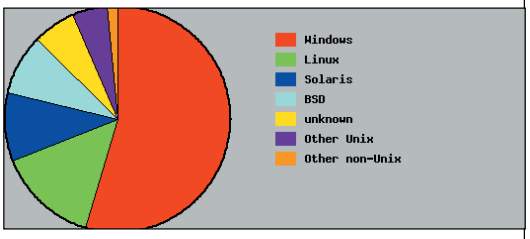
Az 1998 óta ismert történet a várakozások szerint alakult. Az Apache részesedése 59,67 százalékra csökkent a 1999-es 60,02 százalékról, a Microsoft pedig a 2000. évi 19,56-ról 20,17 százalékra erősödött, míg a Sun iPlanetje (ide tartoznak a régi Netscape kiszolgálók is) 7,12-ról 6,92 százalékra esett vissza. Ezen számok azonban csak a piaci részesedés alakulásáról tanúskodnak, és nem derül ki belőlük az, hogy mindhárom terméket egyre többen használják:

- Apache: 590 305 új felhasználója van, számuk 12 705 194-ről 13 295 499-re nőtt.
- Microsoft IIS: 352 960 új felhasználója lett, számuk 4 140 977-ről 4 493 937-re nőtt.
- Sun iPlanet: 27 169 új felhasználója akad, számuk ezzel 1 514 106-ról 1 541 275-re nőtt.

A dagály még a legerősebb hajókat is megrázza...
A Netcraft talán legérdekesebb adatait a rendelkezésre állási idők jelentik. A cég a legalább öt honlapot üzemeltető kiszolgálókat heti több alkalommal is lekérdezte, és a legjobb ötven nevét és elérhetőségét a honlapján közzéteszi. Valóban meglepő eredmények születtek: látható, hogy a Slashdot kiszolgálóját szinte mindennap újra kellett indítani. Ez egészen 1999 végéig visszamenőleg így volt, amikor is az üzemidőt sikerült 60 napra feltornázni, sőt, az egyik kiszolgálójuk 2000. májusa óta folyamatosan működik. A Slashdot mára egy jónak mondható 190 napos eredményt tudhat magáénak. A Netcraft jelentése szerint a Slashdot Apache/1.3.12 (Unix) mod_perl/1.24 kiszolgálót használ, mely Linuxot futtat. Ezzel szemben a Microsoft Microsoft-IIS/5.0/Windows 2000 kiszolgálója 75,22-os átlaggal „büszkélkedhet". Azért megjegyeznénk, hogy például a Starbucks már a 215. napi üzemen is túl van, mióta a kiszolgálót Windows 2000 alól futtatják. A Netcraft állítása szerint a Windows 2000-t használó honlapok közül ennek van a legmagasabb rendelkezésre állási mutatója. Az igazi bajnok a www.charite.de, ez cikkünk írásakor 836,49



A piac megoszlása a kiszolgálók között 1995. augusztus és 2001. február közt



SSL-t használó operációs rendszerek megoszlása az USA-ban 2001. február

napja (két és fél éve) megszakítás nélkül működik, ők Netscape-Commerce/1.1-et használnak IRIX-en.
Az Exodus Communications kiszolgálója üzemelteti a legtöbb honlapot: a californiai Santa Clarában lévő egyik telephelyükön 548 működik. A négy legjobban elérhető kiszolgáló a Zope birtokában van, mindegyiken Linux fut. Átlagos rendelkezésre állási idejük 384 nap, ez épp a Netcraft által figyelt időtartamnak felel meg. Három gépen a Zope ZServer 1.1b1, a negyediken pedig az Apache 1.3.4 fut. Az Exodus listáján található még a smellygig.com (Apache/Solaris, 277 nap), a cluetrain.com (Apache/Solaris, 258 nap), a mysql.com (Apache/Linux, 183 nap), a whitehouse.com (igen, a szexoldal, Apache/BSD, 162 nap) és a slashcode.com (Apache/Linux, 190 nap).
A Netcraft egyik legszellemesebb kimutatása a webkiszolgálókat fejlesztő cégek honlapjának rendelkezésre állási idejéről árulkodik (amikor a hóhért akasztják...). A listán hatodik helyen áll a VA Linux, kilencedik a Sun. A Microsoft a huszonötödik helyre került.

A Netcraft legfrissebb adatait a
➲ http://www.netcraft.com/survey/ címen találjuk.

Mindig azok a programhibák...

A fenti kódrészletben vigyázzunk a hibákra! Csak bizonyítottam, hogy hibátlan, de nem próbáltam ki.

Donald Knut

Így kell ezt!

Látod, ahhoz, hogy egy Linuxhoz hasonló rendszert létrehozz, nemcsak ügyes programozónak, hanem titkolózó taplónak is kell lenned :-)

Linus Torvalds

A programozók igyekeznek egyre együgyűbb emberek által használható programokat gyártani, a világegyetem pedig az igénynek megfelelően igyekszik az egyre együgyűbb embereket legyártani. Úgy nézem, a világegyetem áll nyerésre...

Jól van, telepítettem a Linuxot, de hol van az ingyen sör, amiről mindenki beszél?

„Az élet" – írta Franz Kafka, illusztrálta Salvador Dali.

A fenti SlashDot aláírásokat a ➲ www.ipa.net/jamesncinis/sig.html *oldalon találtuk.*

Egy új tárolórendszer előnyei

A michigani állami egyetemen 1989-ben alapított Julian Samora Kutatóintézetnek egyetlen célja van: az Egyesült Államok közép-nyugati részén élő, latin ajkú közösségek igényeit kielégítő kutatások és képzések vezetése. Az intézetnek küldött adományok a kutatók munkáját és műveik kiadását segítik. Ez a kutatási program az amerikai és a közép-nyugati latin nyelvű közösségek társadalmi, gazdasági, oktatási és politikai helyzetét helyezi a középpontba. Az intézet által szolgáltatott adatok a latin ajkúakról és az ő számukra készülnek.

Az intézet egy évtizeddel ezelőtt kezdte meg kisebb terjedelmű könyvek és jelentések kiadását. Azóta a kutatási és kiadási kör tízszeresére növekedett. Három évvel ezelőttig az intézetben általános rendszer volt az, hogy a megjelentetett munkákat egyszerűen fájlokban gyűjtötték össze.

A kiadással és rendszergazdai teendőkkel is foglalkozó *Danny Layne* így beszél erről: „Ha nyomtatásban volt szükség egy könyvre, akkor előkerestük a megfelelő fájlt, kinyomtattuk, majd visszatettük a helyére."

A kiadott anyagok számának növekedésével egyre több fájl készült, és a fájlok egyre összetettebbé és nagyobbakká váltak. A kutatók már egy-egy fejezetet is több fájlra daraboltak, így nem volt ritkaság, hogy egy könyv elektronikus változata akár húsz fájlból tevődött össze. A kutatók grafikonokat, ábrákat, PowerPoint-bemutatókat is készítettek. A könyvek nemcsak nyomtatásban, hanem az intézet honlapján is megjelentek. „A végére már egy csomó olyan fájltípussal kellett dolgoznunk, amiket alig ismertünk" – mondja Layne. Az asztali gépek merevlemezei nem voltak képesek eleget tenni a hatalmas tárolási és letöltési igényeknek. „Ha az egyik PC merevlemeze csütörtököt mondott, akkor egy régi szalagról kellett visszamásolnunk az egész odaveszett adathalmazt, ami nem egy túl hatékony módszer." Így tehát az állam és az egyetem pénzéből az intézet vásárolt egy központi tárolórendszert, ahová a kiadott anyagok és a honlap fájljai kerülnek. Mivel az intézet korlátozott erőforrásokkal és kevés szakemberrel rendelkezett, így a tárolórendszernek mindenképpen megbízhatónak, egyszerűen telepíthetőnek és felügyelhetőnek kellett lennie, és emellett a bővíthetőség is fontos szempont volt. Layne megjegyzése: „A keresés során a Winchester Systems nevű céghez jutottunk el, tőlük vettünk egy FlashDisk külső RAID tárolórendszert, hét, 9 GB-os meghajtóval."

Layne elmondta, hogy három évvel ezelőtt az intézet gyakorlatilag nem rendelkezett kitüntetett tárolórendszerrel, csupán néhány asztali PC-vel. „Abban az időben a FlashDisk lehetővé tette, hogy egy kis intézet a hatalmas egyetemen belül valóságos nagykiadóvá nője ki magát. Az egyetem számos más tanszéke irigykedik a tárolórendszerünkre. És van is rá okuk..." A FlashDiskhez két egymás mellé elhelyezett Dell PowerEdge 2200-as, egy WindowsNT és egy linuxos gép csatlakozik. A rendszer gyors, megbízható RADI 5-típusú tárolást nyújt több, különböző operációs rendszert futtató kiszolgáló számára. Így nem szükséges minden kiszolgálóhoz külön tárolórendszert vásárolni. Layne szerint az egyetlen tárolórendszer kezelése jóval egyszerűbb feladat, mint ha három-négy ugyanilyen telepet kellene felügyelni.

A belső hálózathoz csatlakozó WindowsNT kiszolgáló jelenti az aktív kiadványok és az azokhoz használt adatbázisok fő elérhetőségét. A FlashDisk lehetővé teszi, hogy minden kutató saját tárolóterülettel rendelkezzen, az asztali gépen rendelkezésre álló tárterülettől függetlenül. A kutatók egy Windows-alapú PC vagy egy Mac segítségével a FlashDisken tárolt WindowsNT, Mac és Linux fájlokat is elérhetik.

A felületfüggetlen programoknak köszönhetően a rendszer valóban egységesen működhet. Ezen programok közé tartozik a WindowsNT kiszolgálón futó Services for Macintosh és a Linuxon működő Netatalk. Ez utóbbi segítségével a nyomtatók hálózati eszközökként üzemelhetnek.

A FlashDisk hatalmas mennyiségű ábrát is tartalmaz. Összességében a FlashDisk a kutatók számára a belső hálózaton hozzáférést enged rengeteg nagyméretű fájlhoz, legyen szó szövegekről, grafikákról vagy bármilyen fájlformátumról. Ha egy könyv nyomtatásban már nem jelenik meg, akkor CD-ROM vagy DVD készül belőle. Eközben a külvilággal való kapcsolatot teremtő Linux-kiszolgáló tárolja az intézet honlapjának fájljait. A FlashDisken körülbelül hétszáz weboldal található; a honlap 3000 találatot kap naponta. A FlashDisk beállítása, a linuxos fájlok és a Linux operációs rendszer tárolása könnyebb feladat volt, mint azt Layne képzelte. „Mi egyszerűen csak a FlashDisk használati utasítását követtük. Fölmerült egy megválaszolatlan kérdés, arra választ kapnunk az ügyfélszolgálattól telefonon, és már működött is."

Bár a FlashDisk hatalmas tárolóhelyet nyújt, Layne azt szeretné, hogy ne foglaljanak felesleges helyet a fájlok régebbi változatai. Szerinte a tárolóterület szervezése, tisztogatása nem akkora feladat, hogy egy rendszergazda ne boldogulna el vele egyedül. „Kutatóinkkal megbeszéltük a FlashDisken belül érvényes tárolási szokásokat, így ma már közvetlenül ők felelősek az adatok létrehozásáért, tárolásáért, törléséért és a biztonsági mentések elkészítéséért."

Layne azt is elmondta, hogy a 9 GB-os meghajtókat nemsokára 18 GB-os merevlemezekre cserélik, ezzel megkétszerezve a tárolóterület méretét: „A Winchester Systems vállalta, hogy egy későbbi bővítéskor meghajtóinkat kedvezményes áron nagyobbakra cseréli. Ez igazán nagyvonalú és kedves szolgáltatás. És igen, a FlashDisk eddig egyszer sem romlott el, még csak árnyalatnyi teljesítménycsökkenésről sem kell beszámolnom."

Elizabeth M. Ferrarini

Linux-index 2001. február–március

1. A programsorok száma egy átlagos elektromos fogkefében: 3000.
2. Az emberi agyban végzett számítások száma másodpercenként, milliószor milliárdban mérve: 20.
3. Az 1998-ban eladott beágyazott processzorok száma milliárdokban: 4,8.
4. 1998-ban a számítógépekbe tervezett beágyazott processzorok százalékos aránya: 2,5.
5. A beágyazott lapkák száma egy átlagos családi autóban: 20.
6. A mikroprocesszorok száma egy amerikai háztartásban: 40.
7. A cégek közti (business-to-business) elektronikus üzletág 2002-re tervezett eladásai milliárd dollárban (USA): 1,3.
8. Az amerikaiakat érő reklámok mennyiségének növekedése 1971 és 1991 között: 6-szoros.
9. A lélekgyógyászoknál a stressz miatti kezelések aránya százalékban: 75 és 90 között.
10. A húsfogyasztás növekedése a népesség legutóbbi megkétszereződése során: 4-szeres.
11. Betegségfajták száma: 250.
12. Gyomnövények száma: 220.
13. A gabonatermés állatok takarmányozására fordított része, százalékban: 40.
14. A Sunday New York Times kinyomtatásához szükséges fák száma: 3200.
15. A bebörtönzött foglyok száma (millió) 2000-ben (USA): 1,3.
16. Az amerikaiak ekkora százaléka kerül élete során börtönbe, ha ez a bebörtönzési arány megmarad: 5.
17. A börtönbe kerülés százalékos esélye egy afroamerikai férfinál (USA): 28.
18. Amennyivel a Yahoo többet ér, mint az összes amerikai magazin piaci értéke együttvéve: harmincmilliárd dollár.
19. Az IBM Linuxra fordított támogatása – a programra, a gépekre, a szolgáltatásokra és a Nyílt Forráskód Közösségére – 2001-ben: egymilliárd dollár.
20. Az X-sorzatú kiszolgálók száma, amiből az IBM össze kívánja állítani a világ legnagyobb linuxos szuperszámítógépét: 1024.
21. A szekrények száma, amibe mindez belefért: 32.
22. Az összeg, amennyivel a tervek szerint 2003-ig az internetfejlesztésre és e-kereskedelemre fordított összeg meghaladja Németország és Franciaország bruttó hazai termék mutatóját: 2,8 billió dollár.
23. Az év, amikorra *Vint Cerf* szerint az Internetre kapcsolt eszközök száma meghaladja a világ telefonjainak számát: 2006.
24. Vint Cerf jóslata szerint az Internetre kapcsolt eszközök száma 2006-ig, a mobiltelefonok kivételével: 900 millió.
25. A repülőgépjáratok száma a világon a következő 24 órában: 42 300.
26. Ahányan utaznak ezeken a járatokon: háromillió.
27. Lezuhant járatok száma (szerencsére): 0.
28. Azon utazók száma, akik tavaly az Internet segítségével tervezték meg utazásukat és helyfoglalásukat: 52 millió.
29. Alkalmazásfejlesztők száma, akik tervezik, hogy vezeték nélküli alkalmazásokat írnak a jövő évben: 40%.
30. A Linux, mint webkiszolgáló felület helyezése : #1.
31. A Linuxot futtató webkiszolgálók száma a világon 2000. májusáig: 36%.
32. A Linux helyzete a leggyorsabban fejlődő kiszolgáló operációs rendszerek között: #1.
33. A linuxos kiszolgálókon használt internetes alkalmazások száma: 40%.
34. Kézi készülékek és hordozható gépek száma 2002-ig: 55 millió.
35. Amikorra a kézi készülékek és a hordozható gépek száma várhatóan meghaladja a PC-két: 2005.

Forrás:

1–17. Richard Saul Wurman: Understanding (Megértés) című könyve ➲ http://www.understandingusa.com
18. Forbes
19–21. IBM
22. Nortel & International Data Corp.
23–24. Domain Street
25–27. Boeing
28–29. Evans Data Corp.
30–31. Netcraft
32–35. IDC

Szerzői jog, Guthrie stílusában

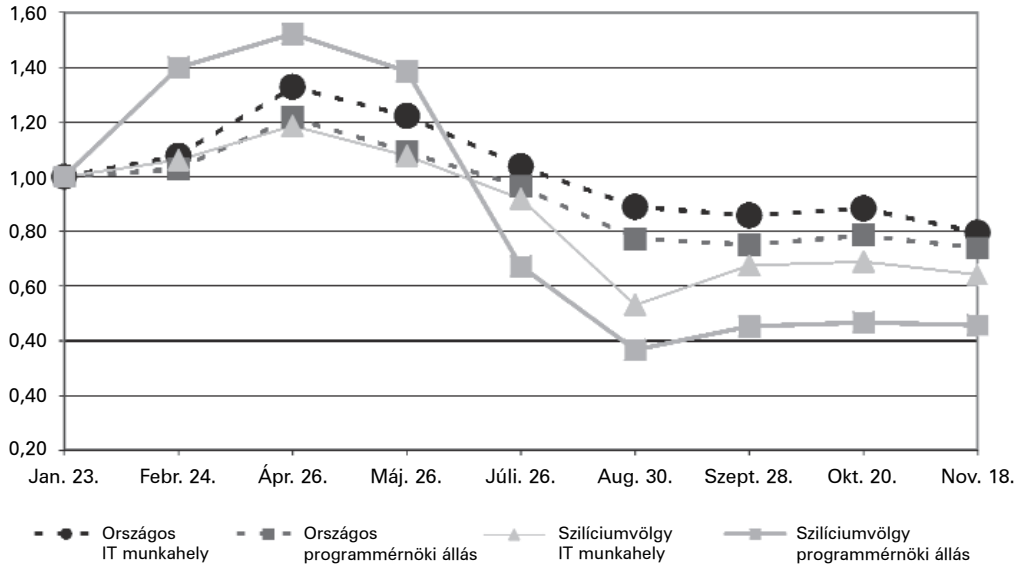
Amikor *Woody Guthrie* az 1930-as években egy kis Los Angeles-i rádióállomás műsorában énekelt, a szövegek után érdeklődő hallgatóknak egy kis daloskönyvet küldött. Ebben találtam: „Ezt a dalt az 154085. számú szerzői jogi bejegyzés védi 28 évig. Ha bárkit azon kapunk, hogy énekli, az menthetetlenül a jó barátunk lesz, mivel nem vagyunk kisstílűek. Adjátok ki. Írjátok át. Énekeljétek. Bulizzatok rá. Jódlizzatok rá. Mi csak megírtuk, és soha nem akartunk ezenkívül mást."

Pete Seeger, 1967. június

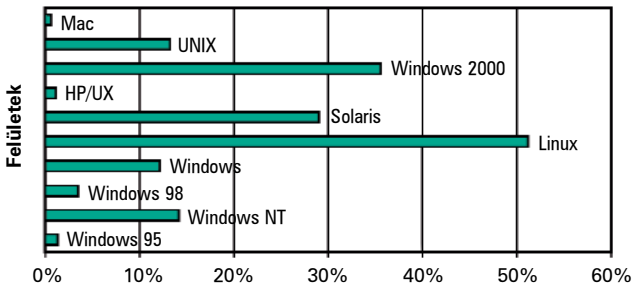
Auu! A fejem!
Ismételd utánam: az elsőbbséget élvező megszakításkérés egy agyrém.
– Linus Torvalds a linux-680x0 levelezőlistán.

Munkaerő-piaci folyamatok

1. ábra **Munkaerőkereslet az elmúlt 11 hónapban (normalizált)**



2. ábra **Felületek iránti kereslete (18 hónap, normalizált)**



A dot-com cégek bukása és a választások okozta bizonytalanságok hatással voltak a munkahelyek számának alakulására. Igaz, hogy tavaly április óta a kínált állások száma csökkent, de ez a folyamat lassulni látszik. Az *1. ábra* a 2000. január és november közötti időszakban kínált állások számának változását mutatja. Ami nem változik, az általában nem igazán érdekes, ebben az esetben azonban örülhetünk annak, hogy a számok csökkenő irányt mutatnak.
(Az értékeket a 2000. januári helyzetet alapul véve számítottuk ki, azaz a 2000. januárhoz tartozó érték 1,00.)

A felületek iránti kereslet változásai

Az elmúlt 18 hónapban több felület is versengett az elsőbbségért. Érdekes lehet figyelemmel kísérni, hogy a főbb felületek iránti kereslet milyen gyorsan változik. A keresletet az adott időszeletre, a folyamatvonal meredeksége írja le. A *2. ábrán* láthatjuk, hogy a régebbi felületek iránt lelassult a kereslet növekedése, míg az újabb felületeknél, mint a Windows 2000 és a Linux, gyorsult.
Az egyetlen kivétel a Solaris, ezt ott találjuk az új jövevények közelében. Ez azt mutatja, hogy a Sun befolyása növekszik a piacon.
Ezek a mutatók azért ilyen kedvezőek, mert hosszú időtartamra vonatkoznak. Ha az utóbbi hónapokra nézzük ugyanezeket a kimutatásokat, megfigyelhetjük, hogy az általános folyamatot követve, minden felület szakemberei iránt csökkent a kereslet. Mindemellett a rövid távú hatás még nem érzékelhető a hosszú távú elemzésekben.

Reginald Charney

Új termékek

A Linux2order webhely

A Linux2order adatbázis több mint 9000 nyílt forráskódú alkalmazást és segédprogramot tartalmaz, amelyeket a világhálón keresztül szabadon letölthetünk és lefordíthatunk. A címeket naponta frissítik. A programok mellett leírást és bírálatot egyaránt találhatunk. A felhasználók a kiválasztott fájlokat közvetlenül letölthetik, de lehetőségük van arra is, hogy egyedi CD-ROM formájában megrendeljék azokat.
Adatok: Linux2order,
5252 North Edgewood Drive,
Provo, Utah 84604,
telefon: 801-222-9414,
➲ http://www.linux2order.com/

ProcureMind 1.6

A ProcureMind 1.6 (Mindflow Technologies) egy olyan program, amelyet beszerzési tervek kidolgozására és nyomon követésére terveztek. A program a korábban használt táblázatok helyét hivatott átvenni. Adataink bevitelét, megjelenítését és módosítását így egyetlen program felhasználásával végezhetjük el. A részprogramok továbbfejlesztett változataiban (az 1.6-os változatban a ProcureStat, a ProcureSave és a Smartsourcing Desktop) olyan újdonságokkal találkozhatunk, mint amilyen például a hatékonyabb költségelemzés.
Adatok: Mindflow Technologies, Inc.,
6504 International Parkway, Suite 2400, Plano, Texas 75093,
telefon: 972-930-9988,
e-mail: info@mindflow.com,
➲ http://www.mindflow.com/

iConnect Suite

Az iConnect olyan webalapú eszközkészlet, amely lehetővé teszi azt, hogy a kisebb és nagyobb méretű vállalatok egyaránt megjelenhessenek alkalmazásaikkal a Weben. A távoli kiszolgálóra telepített alkalmazások egy szabványos Java-környezetes böngésző birtokában bárhol elérhetők az Interneten keresztül.
A csomagban megtaláljuk a HTTP-alapú alkalmazásokat, az adatkezelő szolgáltatásokat, a fájlátvitelt, a megosztás és az összehangolás támogatását, találunk ügyfélalkalmazásokat, webalapú munkakörnyezeteket, munkaszervező és irányító eszközöket is. A program kipróbálható változata az Internetről ingyenesen letölthető.
Adatok: SoftKnot Corporation, 46500 Fremont Boulevard, Suite #716, Fremont, California 94538,
telefon: 510-226-2768,
e-mail: mmovahed@
➲ softknot.com,
➲ http://www.softknot.com/.

Wing IDE for Python

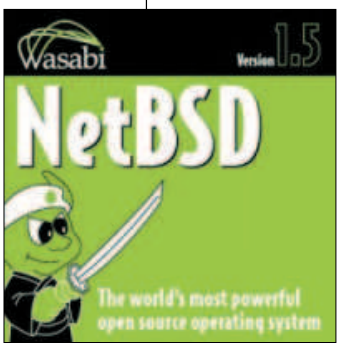
A Wing IDE fejlesztőkörnyezet, melyet a Python programozási nyelvhez állítottak össze. A programfejlesztők beépített termékkezelőt, grafikus hibakeresőt, forráskódböngészőt és forráskódszerkesztőt is találhatnak a csomagban. Az IDE képes együttműködni a Makefile-okkal és más külső beállítóeszközökkel is. Ezenkívül támogatja a Tkinter, a PyGtk és a PyQt eszközökkel készített hibakereső programok használatát is.
Adatok: Archaeopteryx Software, Inc., PO Box 1937, Brookline, Massachusetts 02446-0016,
telefon: 617-232-0059,
e-mail: info@archaeopteryx.com,
➲ http://archaeopteryx.com/

WARP Performance Suite

A WARP Performance Suite programot az internetes alkalmazások hatékonyságának növelésére fejlesztették ki. A cél az volt, hogy a teljesítmény már a webes tartalom származási helyén is egyszerűsíthető legyen.
A csomag jelenleg a következő elemeket tartalmazza: Intelligent Content Distributor (Értelmes tartalomterjesztő), Load Balancer (Terheléselosztó) és Global Load Balancer (Általános terheléselosztó). A Dynamic Content Director (Dinamikus tartalom igazgató), a Cache Master (Gyorstárkezelő) és a Secure (Biztonság) összetevők 2001 első negyedévében készülnek el.
Adatok: WARP Solutions, Inc.,
627 Greenwich Street, New York, New York 10014,
telefon: 877-688-WARP,
e-mail: info@warpsolutions.com,
➲ http:/www.warpsolutions.com/

NetBSD 1.5 CD-ROM

A NetBSD 1.5 lemez egy teljes értékű Unix-szerű operációs rendszer. A szolgáltatások között megtalálhatjuk a legkülönfélébb felhasználói segédprogramokat, fordítókat több programnyelvhez, az X Window rendszert, a tűzfalat, a vezeték nélküli, illetve azonnal használható (Plug and play) eszközök támogatását és a teljes forráskódot is.

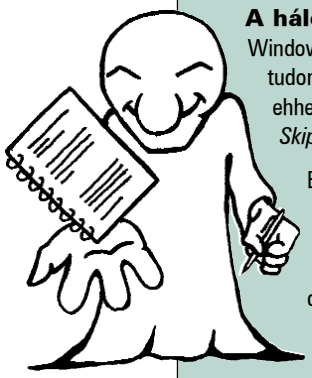


A NetBSD előnyei közül feltétlenül ki kell emelni a programok hordozhatóságát, a felületek közötti egységességet, a magas szintű hálózatkezelést és az emulációs lehetőségeket is.
A Wasabi változat a leírás mellett telepítési útmutatót is tartalmaz.
Adatok: Wasabi Systems, Inc.,
104 West 14th Street, 4th Floor,
New York, New York 10011,
telefon: 917-974-9815,
e-mail: info@wasabisystems.com,
➲ http://www.wasabisystems.com/

INTERNETpro Small Enterprise Server 2012

Az INTERNETpro Small Enterprise Server 2012-t a dinamikus IP-címzést használó virtuális magánhálózatokhoz (VPN) fejlesztették ki. A 2012 az elektronikus kereskedelemben és a távközlésben történő felhasználást rengeteg szolgáltatással igyekszik kényelmesebbé tenni: titkosított virtuális magánhálózat, tűzfal, webkiszolgáló, levelező kiszolgáló, beépített útválasztás, nagy kiterjedésű hálózatok (WAN) összekapcsolása, webgyorstárazás, proxykiszolgáló, címtiltás és dinamikus tartalomfigyelés.
A 2012 a 8 gigabájtos merevlemeznek és 450 MHz-es processzornak köszönhetően akár 150 felhasználó egyidejű kiszolgálására is képes.
Adatok: Internet Appliance, Inc.,
40515 Encyclopedia Circle, Fremont, California 94538,
telefon: 510-413-1068,
e-mail: sales@internetappliance.com,
➲ http://www.internetappliance.com/

A hónap szakmai tanácsai



A hálózat beállítása

Windows NT alatt az ipconfig nevű programmal tudom módosítani az IP-beállításokat. Létezik ehhez hasonló linuxos program?
Skip Bigelow, sbigelow@aarp.org

Bár a kérdéses feladatra több menüvezérelt program is használható (például a RedHat netcfg parancsa) a /sbin/ifconfig mindig kéznél van. Segítségével az összes működő csatolóról (ethernet, ppp, loopback stb.) részletes tájékoztatást kaphatunk.
Mario, mneto@argo.com.br

A vmlinuz betöltődik, aztán nem történik semmi

Gépemre a RedHat 6.0-s Linuxot szerettem volna telepíteni. A CD-ROM-ról hibátlanul elindult a gép, a boot: üzenet után ENTERT ütve az alábbi üzenetek jelentek meg:

```
Loading initrd.img.................
Loading vmlinuz.......
```

Ezután a számítógép leállt.
Amikor a Win98 indítólemezről indítottam el a gépet, a RedHat CD-ROM /dosutils/autoboot nevű programját futtattam. Meglepetésemre azt közölte velem, hogy a Linuxot az X rendszerrel együtt hibátlanul telepítettem, és a végén ezt írta ki:

```
Congratulations, you have installed
linux successfully.
The system is rebooting......
```

Majd újraindítás után:

```
Loading linux.......
```

És itt megint lefagyott.
A RedHat 5.0, Bluepoint 1.0 és 2.0, Turbolinux, Slackware változatokkal is szerencsét próbáltam, de az eredmény teljesen ugyanez. Miért?
ekun, xx@public1.ptt.js.cn

A Linuxot a loadlin paranccsal is indíthatod (ez történt akkor is, amikor Windows alól indítottad a Linux telepítését), de valami különös oknál fogva ez nem működik, ha LILO-t használsz. Az egyik lehetőség, hogy Windows alól telepítesz (ahogy eddig is tetted), majd a RedHat CD-jéről indítod a rendszert. Ezután a befűzött Windows-lemezrészre másold át a rendszermagot (a /boot-ban van). A RedHat CD-ről másold át a Windowsra a loadlint és állítsd be, majd indítsd ezzel a rendszert. Nálam a loadlin beállításfájljai így néznek ki:

```
moremagic:/drv/c$ cat linux.bat
c:\linux\loadlin\loadlin
@c:\linux\loadlin\boot
moremagic:/drv/c$ cat linux/loadlin/boot
c:\linux\loadlin\vmlinuz
root=/dev/sda6
ro
```

Marc Merlin, marc_bts@valinux.com

Láttam már ilyesmit. Olyankor fordul elő, ha a rendszermagot nem a megfelelő processzorral használod, de ha ez rögtön telepítés után jelentkezik, akkor lehet, hogy komoly géphibáról van szó. Próbálj ki más memóriamodulokat (nem biztos, hogy ezzel lesz a baj, de célszerű először ezt megnézni).
Robert Connoy, rconnoy@penguincomputing.com

Kábelmodem megosztása

Már jó néhány linuxos weboldalt átböngésztem, de eddig sehol sem találtam választ kérdésemre. Hogyan oszthatnám meg otthoni kábelmodemes kapcsolatomat egy linuxos és egy windowsos gép között? A kábelmodem az utóbbiban található.
Samuel Fung, samfz@hotmail.com

Ha a helyedben lennék, én inkább átraknám a linuxos gépbe a kábelmodemet. Hogy miért? Egész egyszerűen azért, mert a Windowsban alapból nincs semmiféle ilyen típusú hálózatkezelési képesség (tehát nem használható átjáróként, ráadásul a csomagszűréshez vagy az IP-továbbításhoz hasonló biztonsági lehetőségeket sem ismeri), ezzel szemben a Linux mindezeket a feladatokat képes elvégezni. Nem írtad meg a kábelmodemed típusát, de javaslom, látogass el a ➲ http://www.linuxdoc.org/ címre, ahol a géped hálózati beállításaival kapcsolatos összes adatot nagy valószínűséggel megtalálhatod.
Felipe Barousse, fbarousse@piensa.com

Rendszerindítás üzenetek nélkül

Ki lehet valahogyan kapcsolni az indításkor a képernyőre kerülő rengeteg üzenetet?
Nicholas, vunch@pacific.net.sg

Ennek legegyszerűbb módja a `set console=ttyS3,38400n8` vagy valami ehhez hasonló használata, mellyel a konzol kimenetét egy soros kapura továbbítod.
Marc Merlin, marc_bts@valinux.com

Rútságok...

Ma reggel megpróbáltam belépni a Linuxba és meglepve tapasztaltam, hogy nem tudok. A bejelentkező üzenet a szokásos módon megjelenik, de amikor a felhasználói név beírása után ENTERT ütök, a jelszót kérő: üzenet helyett a rendszer ismét a felhasználói nevet kéri. Semmilyen egyéb üzenet nem jelenik meg, kivéve egyet és ez így szól: `/var/hackr0x/login: No such file or directory`. A sor egyébként olyan gyorsan eltűnik, hogy csak jó néhány próbálkozás után tudtam kibetűzni.
Victor, victor@angolatelecom.com

Nos, a gépedet feltörték. Ebben a pillanatban lehetetlen megállapítani, hogy milyen kár érte a rendszert, és a kijavítással sem érdemes foglalkozni – a legjobb, ha a fontos adatok mentése után újból telepíted az alaprendszert. Ehhez természetesen valahogyan be kell jelent-

kezned: a LILO parancssorába írd be a `linux init=/bin/bash` sort, majd használd a következő parancsokat: `mount -wno remount; mount -a; etc/rc.d/init.d/network start` (ezzel feláll a hálózat). Esetleg a vész esetére fenntartott hajlékonylemezről vagy CD-ről is indíthatod a rendszert.
Újratelepítés után véletlenül se kapcsold vissza a hálózatra a gépet: először nézz át mindent. Telepítsd a legújabb hibajavításokat, ne indíts felesleges démonokat, és ha lehetséges, akkor védd tűzfallal a gépet.
Marc Merlin, marc_bts@valinux.com

Minden nagyobb Linux-változatban található egy lista az adott változatba beépített biztonsági javításokról, az újratelepítés után olvasd el ezt. Azután minden felesleges programot el kell távolítanod – ez az egyik legegyszerűbb és leghatékonyabb biztonsági rendszabály, amit célszerű szem előtt tartanod.
Don Marti, dmarti@linuxjournal.com

Nincs démon, de nyomtatnom kell!

Ha bármit megpróbálok a nyomtatóra küldeni, a `Job is queued, but cannot start daemon` üzenetet kapom. Ha közvetlenül a cat paranccsal küldök ki szöveget a nyomtatóra, akkor a művelet hibátlanul lezajlik, de ez nem indítja el a démont. Az `lpc status` azt mondja, hogy nem fut semmilyen démon. Már jó néhányszor telepítettem nyomtatót Linux alatt, de ilyesmivel még nem találkoztam. Már az lpr csomag eltávolításával és újratelepítésével is megpróbálkoztam, mondanom sem kell, hogy sikertelenül. Ráadásul az összes általam ismert trükk csődöt mondott. Még a *Running Linux* című könyv sem segített, pedig az elég sok bajból kihúzott már. Vajon mi a hiba oka?
Jim Jerzycke, kq6ea@amsat.org

Győződj meg arról, hogy a nyomtatási sorrendért felelős démon (spooler) fut-e. SuSE alatt arra is oda kell figyelni, hogy a /etc/rc.config fájlban az alábbi sor is szerepeljen:
`START_LPD="yes"`
Ha ez nincs ott, vagy ha értéke „no”, akkor a módosítás elvégzése után a SuSEconfig futtatásával frissítheted a rendszert.
Robert Connoy, rconnoy@penguincomputing.com

A / jel a hálózati maszkban

Mostanában ismerkedem a linuxos tűzfalakkal, mivel az egyik ügyfelem számára be kell állítanom egyet. A tűzfal szabályait meghatározó parancsfájlban (Ipchains) a következő sorokat találtam:

```
INT0="eth0"
IP0="192.168.1.125/24"
NET0="192.168.1.0"
```

Mit jelent az IP-cím után álló /24?
Kerülhet egy változóba két hálózat? Így gondoltam:

```
NET0="192.168.1.0,192.168.10.0"
```

Fabio Losnak, fabiolosnak@yahoo.com

A `/24` az IP-címben azt jelenti, hogy a hálózati maszk `255.255.255.0` (tehát a hálózati azonosító 24 biten foglal helyet).
Egy változóba valószínűleg nem tehetsz két hálózatot, de ez a parancsfájlt beolvasó programtól függ.
Marc Merlin, marc_bts@valinux.com

Az email-fiókok letiltása a Sendmail segítségével

A levelezőkiszolgálónk (RedHat 6.2, Sendmail Single Switch) továbbítóként (smart relay) működik a szabad zónában (a DMZ-ben). A belső hálózatban egy másik levelezőkiszolgáló van (RedHat 7.0, Sendmail Single Switch), mely SMPT és POP3 szolgáltatást is nyújt.
Az a feladat, hogy el kell választanunk azokat a felhasználókat, akik csak belső levelezést folytathatnak, illetve azokat, akiknek a levelei az Internetet is elérhetik.
Hogy tovább bonyolítsuk a dolgokat, minden felhasználónak vezetéknév.keresztnév@tartomány.com formájú címet kell kapnia. A Sendmail képességei lehetővé teszik mindezek megvalósítását? Ha nem, milyen programokat kell használnom?
Michael Philips, mike.philips@ieionline.com

A feladat megoldására számos módszer létezik. Az egyik legegyszerűbb, ha a `/etc/mail/access` fájlban a megfelelő ügyfeleknél letiltod a külső levelezést. A helyzet nem is olyan bonyolult, csupán némi tervezés szükséges, valamint a Sendmailt kell egy kicsit megbütykölni. Látogass el a ➲ http://www.sendmail.org/ oldalra, ott megtalálsz minden szükséges tájékoztatást.
Felipe Barousse, fbarousse@piensa.com

A telnet-kapcsolatok kifutnak az időből

A Linuxban be lehet állítani a telnet időtúllépési korlátját? Jelenleg ugyanis minden kapcsolatom megszakad úgy öt perc múlva.
Jan Dubroca, jan.dubroca@delta-air.com

Valószínűleg a hálózaton kívülre nyitsz telnet-kapcsolatot egy olyan IP-álcázást (masquerading) végző kiszolgálón keresztül, mely a TCP-kapcsolatokat öt perc elteltével megszakítja.
Linux alatt a megoldás (a tűzfalon):

```
# A bújtatás időtúllépésének kijavítása
#                    tcp   tcpfin   udp
ipchains -M -s       86400 60       120
```

Marc Merlin, marc_bts@valinux.com

Nem hinném, hogy a telnet dob ki öt perc után, sokkal valószínűbb, hogy a héj. A héj időtúllépését a `/etc/profile` fájlban állíthatod be. Minden bizonnyal ebben egy, az alábbihoz nagyon hasonló sor is szerepel:
`TMOUT=300`
Az itt megadott érték másodpercekben értendő. A számot növelve több időt hagyhatsz a felhasználóknak, de ha az egész sort törlöd, akkor az időtúllépést le is tilthatod.
Paul Christensen, pchristensen@penguincomputing.com

A rendszermag belülről

Ha valaki azt várja, hogy a Linux-világ csak a Linuxról szóljon, az olyan, mintha a Kis Márta élete című újság csak Kis Márta életéről szólna. Nem, várjunk csak, rossz példa. Mindegy, a lényeget valószínűleg mindenki megértette. A Linux, ez a POSIX-megfelelő rendszermag, mindenki számára hozzáférhető program. (Egyébként a Linuxvilág kéthármad akkora terjedelemben és tizedakkora csapattal készül, mint egy Kis Márta lap. Nem kell egy csomó új receptet kiagyalnunk, cserébe viszont Mártáéknak sem kell %{$_} Perl kódot tördelniük.)

Tehát mivel a Linux csak van és működik (milyen unalmas!), ezért mi a hasábokon webkiszolgálókról, fejlesztőeszközökről és más, a Linuxon (és egyéb POSIX-megfelelő rendszermagon) futó érdekes eszközökről írogatunk. Szaktekintély és Kovácsműhely című rovatunkat például kiadhattuk volna „POSIX világ” címmel is.

Ha valami mindenki számára hozzáférhető, az még nem jelenti azt, hogy nincs hírértéke. Végül is, a marhahús is minden sarkon kapható, ennek ellenére mostanában tele van vele minden híradó. Szóval itt az ideje, hogy a rendszermaggal foglalkozzunk.

Clay Clairborne, egy Los Angeles-i Linux-tanácsadó egyik ügyfele azt szerette volna, hogy Linux alatt is olvasható legyen a Digital Unix (vagy hogy is hívják mostanában) formátumú merevlemeze. Persze nem annyi volt az egész, hogy a rendszermagba fordítjuk a UFS-támogatást, de a szabadon hozzáférhető kódnak és leírásnak, illetve a szakértő fejlesztőknek köszönhetően mégsem volt lehetetlen a feladat. A régi/új fájlrendszer hozzáadásának mikéntjéről a következő oldalon olvashatunk. Ha az ext2fs-t használjuk, akkor is nagyszerű segítséget jelent Clay Clairborne cikke.

Bizonyára sokan hallották már, hogy a SuSE Linux már támogatja a Reiser fájlrendszert. A ReiserFS-be könnyű beleszeretni, hiszen ha a linuxos gépünk egy áramszünet alkalmával leáll, újraindítás után nincs szükség az fsck hasz-nálatára. De mi az igazi különbség a ReiserFS és a régi Unix vagy a Linux ext2fs fájlrendszere között? *Chris Mason* mesél nekünk a b-fákról és arról, hogy mi is történik akkor, amikor a ReiserFS megmenti adatainkat egy váratlan leállás során.

Az Internet? Itt eddig egyszerű volt az élet: az átjárók a forgalmat terelték, a rendszergazdák pedig indították vagy fogadták a kapcsolatokat. De a terheléselosztók, a tűzfalak és a hálózati címek feloldásának korában az internetes forgalmat irányító gépek világa egyre különlegesebb állatkertté változik a szemünk előtt.

Nenad Corbic és *David Mandelstam* a WANPIPE-ról ír, mely egy linuxos meghajtóprogram a PCI-os WAN-csatlakozók számára. Segítségükkel bármilyen internetes készüléket összeállíthatunk, akár egy CSU/DSU nélkül közvetlenül a T1 vonalra kapcsolódó eszközt is. Az Interneten rengeteg biztonsági, teljesítménybeli nehézség adódhat, és a WANPIPE lehet legfontosabb ellenszereink egyike.

Melyik volt az első rendszermag, melyben szabvány-API segítségével megvalósítható az internetes telefonálás („Voice over IP”)? Természetesen a Linux. *Greg Herlein* elmagyarázza, hogy mi rejtőzik a /dev/phoneN mögött, illetve hogyan használhatjuk az ohphone-hoz hasonló programokat arra, hogy ingyen (na jó, a helyi hívás díjáért) telefonáljunk a világ másik végére. Sőt, valódi telefont is használhatunk.

Végül: mi a közös ezekben a rendszermaggal foglalkozó témákban? A szabadság, természetesen. A számunkra sztálinista rémálomnak tűnő, központilag, zárt körben (cégen belül) fejlesztett programok helyét egyre inkább a szabad piaci rendszer veszi át. És ez nagyon jó.

Néha előfordul, hogy valaki egy jól működő, ingyenes rendszerhez megpróbál pénzért elemeket továbbadni, az így elérhető teljesítménynövekedésre hivatkozva. De ez az udvariatlan és pusztító szándékú viselkedés nem csak azért bajos, mert „nem illik össze a forradalmi elvekkel”. Hiszen ha egyszer elhatároztuk, hogy megszabadulunk a Nagy Cég vitatható minőségű termékeitől, akkor a szép új világot nem egymással acsarkodók uralmában kívánjuk elképzelni, nem igaz? És itt nem csak a szellemiségről beszélünk. Keressünk rá a Weben valamelyik nem GPL-es Linux modulra, és biztos, hogy ráakadunk több száz régi üzenetre, melyekben a szerencsétlen felhasználók segítséget kérnek, mert a fizetős modul összeakadt valamelyik szabványos Linux-modullal és porrá zúzta a rendszermagot.

Természetesen most még csak a fentebb vázolt folyamat elején tartunk, de a legnyilvánvalóbb tévutakat már könnyűszerrel képesek vagyunk kikerülni. Hosszú távon a szabadság erkölcsileg és anyagilag is megéri. E havi számunk Vezérfonalaként azt boncolgatjuk, milyen hasznos árucikk válhat kedvenc rendszermagunkból.

Don Marti

A fájlleírók megzabolázása

Előfordul, hogy munkánk során egy-egy olyan hibával találkozunk, amelynek kiküszöböléséhez bele kell néznünk a rendszermag forrásaiba.

Tudtam, hogy a linuxos szakik gyorsabban raknak össze új fájlformátumokat és lemezterület-típusokat, mint ahogy Carlie az ingyenes italjegyeket osztogatta a Linux Journal partiján.

Minden egy telefonhívással kezdődött. „Tudnának olyan Linux-rendszert építeni, amely képes befűzni és olvasni egy DEC-meghajtót, és elérhetővé tenni az adatokat NT-munkaállomások számára Sambán keresztül?” A hívó a General Dynamics munkatársa volt. Ez a cég gyártja sok egyéb között az US haditengerészet atom-tengeralattjáróinak többségét, mi pedig egy haszonelvű linuxos vállalkozás vagyunk, a lehető leggyorsabban szerettem volna visszahívni.

Ekkor már jó néhány éve foglalkoztunk linuxos feladatok megoldásával, úgyhogy éppen a szakterületünkbe vágott a dolog. A Cosmos Engineering Company 1984 óta szállít rendelésre készülő számítógéprendszereket az ipar szereplői számára. 1996-ban kezdtünk el kizárólag Linux-rendszerekkel foglalkozni. Ugyanebben az évben mutattuk be a „Linux on a Disk” („Linux egy lemezen”) nevű rendszerünket. A következő évben a RedHat Hardware Partners alapítóivá váltunk.

A DEC-meghajtók Quantum 9 GB tárkapacitású SCSI-2 egységek voltak, OSF lemezrészekkel és UFS fájlrendszerrel. Bár akkor még nem voltam biztos benne, tudtam, hogy a linuxos szakik gyorsabban raknak össze új fájlformátumokat és lemezterület-típusokat, mint ahogy Carlie az ingyen italjegyeket osztogatta a Linux Journal találkozóján. Például a 2.2.15 és a 2.3.99pre9 között a támogatott idegen lemezterület-típusok száma háromról tizenötre nőtt. Az UFS a 2.0.xx változattól található meg a rendszermagban. Csakhogy az UFS-nek több változata is létezik, nem? Ezeknek a száma szintén tovább emelkedett a 2.4.0-használható-1 felé menetelés közben.

Így megnéztem a rendszermag pillanatnyi fejlesztői fáját, ez akkoriban a 2.3.99pre9 volt. Hét különféle UFS-típust támogatott ez a rendszer, bár a DEC-et külön nem említették, valamint az OFS-lemezrész támogatása is friss jövevény volt. Ezen felbuzdulva visszahívtam *John Loefflert* a General Dynamics Electronic Systemsnél. Egy óvatos igennel kezdtem, majd megkérdeztem: „Kaphatnánk egy meghajtót kipróbálásra, hogy pontos választ adhassunk?” Egy FedEx kézbesítéssel később felépítettem a legújabb fejlesztői rendszermagot, mely minden olyan fájlrendszert és lemezrész-típust támogatott, ami kicsit is hasonlított a számunkra szükségeshez.

Ahogy azt előre sejtettem, az OSF-lemezrész és a csak olvasható UFS-típusú Sun fájlrendszer támogatásával befűzött meghajtó működőképesnek látszott. Az ellenőrzőlemezen mindössze három fájl volt: rc.local, hosts és egy viszonylag nagy tároló, az oilpatch.tar. Ahogy kérték, elfaxoltuk nekik a két kisebbik fájl tartalmát, mintegy bizonyítékul, hogy a Cosmos Engineering Company el tudja vállalni a szóban forgó feladatra felkészített Cosmos 500 linuxos kiszolgálók összeállítását.

A kért gépek mindegyike négy, 9 gigás SCSI DEC OSF meghajtót képes kezelni, és lehetővé teszi a hálózaton kapcsolódó NT-kiszolgálóknak az adatok olvasását. Ez a feladat később tovább bővült a fájlok és könyvtárak törlésének lehetőségével. Rendben, semmi gond, a 2.4.0-test1 rendszermag rendelkezik kísérleti UFS-írás képességekkel. A gép semmi különös dolgot nem tartalmazott. Minden kiszolgáló egy 500 MHz-es Pentium III-as gép volt, ASUS i404BX ATX alaplapra ültetve, 128 MB memóriával egy közepes ATX toronyházban. A 18 GB IBM 2Ultra SCSI rendszerű meghajtóra RedHat Linux 6.1 került. Az egyetlen dolog, ami egyedivé tette a rendszert, hogy képes volt olvasni az idegen lemezformátumot. Ehhez kellett egyedi rendszermagot készíteni.

Meglehetősen lenyűgöző volt a papírmunka, ami egy olyan horderejű céggel történő megállapodáshoz kellett, amely rombolókhoz szokott alkatrészeket szállítani. Soha nem készítettünk olyan szerződést, amelyben ennyi kormányzati szabályozás szerepelt volna. No igen, olyat sem, amiben külön szabályozás lett volna arra, miképpen kell kezelni a fennmaradó összeget, amennyiben a költségek meghaladják az ötszázezer dollárt.

Az első ellenőrzőlemez, valamint számos azt követő, elég nyilvánvalóan valótlan adatokkal volt megtöltve. Soha nem tudtuk, mi a valódi adat és mi nem. Bármi is volt, az biztos, hogy sok volt belőle, és igencsak arra törekedtek, hogy a házon belüli adatforgalom abban a mederben folyjék, amit ők készítettek. Tudtuk, hogy a végfelhasználó az amerikai kormány, és a munkák során kiderült, hogy ez a hadsereget jelentette. Később, amikor segítettem a General Dinamicsnak megoldani egy SCSI lezárási gondot, elmondták, hogy a SCSI-meghajtók fémdobozokba vannak zárva, azokat pedig rázkódásmentes sínekben rögzítették. Elmondták, hogy ilyet használnak mindenhol. A „mindenhol” ebben az esetben az atom-tengeralattjárókat, rombolókat és egyéb fegyverrendszereket jelentette. Azt is megtudtam, hogy *Dr. Lee* üldözése közben, Los Alamosban eltűnt egy nukleáris titkokat tartalmazó táskagép merevlemeze azóta, sok figyelmet fordítottak kormányzati körökben a nemzet katonai adatainak biztosítására. Lehet, hogy ennek semmi köze nem volt a magától értetődő tömeges adatvándorláshoz. Ez azonban pusztán csak eszmefuttatás. Nem tudom, hogy a jövőben mire fogják a kiszolgálóinkat használni, de nem is akarom tudni. Amikor a General Dinamics-szal beszéltem, folyamatosan az volt az érzésem, hogy elmondhatják ugyan nekem, de akkor, sajnos kénytelenek lesznek… Nos, ugye sejtik, mire gondolok.

A szerződésnek megfelelően elkezdtük összeállítani és ellenőrizni az első kiszolgálókat. A munka azonban zátonyra futott, amikor kiderült, hogy a DEC-meghajtóról a hosszabb fájlok másolásával már gond van. A fájlok 96 kB-ra csonkolódtak, és ennek nem lett volna szabad megtörténnie. Miért pont 96 kB? Ezt szerettem volna kideríteni. Az eset további érdekessége volt, hogy a hosszú fájlokat – például a rendszermagtárolót – sértetlenül fel lehetett másolni a DEC-meghajtóra, majd visszamásolni. Ha azonban a felmásolás után a rendszert leállítottuk, majd újraindítás után másoltuk vissza a DEC-meghajtóról, akkor az eredmény helyes hosszúságú állomány volt ugyan, de a belsejében csak szemetet találtunk. Gyanítottam, hogy a gond a fájlleírók (inodes) memóriabeli kezelése és lemezen történő tárolása körül lehet.

Így megkezdtem hosszú utazásomat Leíróországban. Előzőleg már hallottam a fájlleírókról, és tudtam, hogy a fájlokhoz tartozó adatszerkezetek, ezenkívül mindig megfelelő mennyiségben szabadnak kell lenniük. Kaptál már valaha „no space on the device” (nincs szabad hely az eszközön) üzenetet, miközben 4 GB szabad helyed volt még? Ez történik, ha nincs elég szabad leíró. A leírók azért elég ködösek voltak nekem. A következő néhány héten azonban többet megtanultam róluk, mint amennyire valaha is emlékezni fogok.

Ezek a kicsiny adatszerkezetek határozzák meg a fájlok tulajdonságait. Az ext2 fájlrendszeren egy leíró 128 bit hosszú. Kényelmes helyzetben vagyok, ugyanis minden, ami ezután következik, egyaránt igaz a Linux ext2 rendszerére és a DEC UFS fájlrendszerre. A leíró valójában egy adattábla, amely megadja a fájl tulajdonosát, engedélyeit, készítésének és módosításának adatait, a fájlméretet, és ami a legfontosabb, a fájl adatainak lemezen elfoglalt helyét. Tulajdonképpen mindent, kivéve a fájlnevet. A fájlnév teljességgel mellékes a fájl szempontjából, csupán a mi kedvünkért van ott. Tulajdonképpen egy könyvtárbejegyzés. Mondjuk, ha szeretnénk meghallgatni egy dalt, és a The_Train_They_Call_the_City_of_New_Orleans.mp3-ra kattintunk, ez egy könyvtárbejegyzés, mely arra a leíróra mutat, ami tudja, hol van az a dal, és ki játszhatja le. Egy leíróra mutathat több könyvtárbejegyzés is, ezeket hívjuk közvetlen vagy egyenes hivatkozásnak (hard link). Egy fájlnak lehet egynél több neve is, de csak egyetlen leírója. Ez az, amiért a Sun berkein belül a fiúk azt mondják: „A leíró a fájl.” Minden fájlnak, ideértve az eszközöket, a könyvtárbejegyzéseket és közvetett hivatkozásokat (soft links), szüksége van leíróra.

Egy-egy lemezrészen viszont csak korlátozott számú fájlleíró van. Amikor a fájlrendszert létrehozzuk, meg kell határoznunk a készítendő leírók számát, és ez már nem változik, mindig ugyanannyi marad. Ha kifutunk a keretből, mert túl sok pici fájlt készítünk egy olyan lemezterületen, melynek kis leírótáblája van, akkor bizony trutyiban vagyunk, még akkor is, ha a szabad hely cicabájtokra rúg. A rendszert, amelyik egyik lemezterületén kifutott a szabad helyből, még megtévesztheted egy másik lemezterületre mutató közvetett hivatkozással. Azzal a rendszerrel azonban nem tehetsz semmit, amelyik kifogyott a szabad leírókból. Legalábbis addig, amíg le nem törölsz valamit, ezáltal fölszabadítva egyet. A közvetett hivatkozás – ez valójában egy fájl, mely egy másik könyvtárbejegyzésre mutat – is igényel leírót.

A leírók olyan adatszerkezetek, amelyek általában a lemezen tárolódnak, és a felhasználáshoz vagy módosításhoz a memóriába kerülnek.

Az adatszerkezetbe pillantva könnyű megérteni, miért is olyan lényeges a 96 kB.

A leíróban tárolt adat első blokkja foglalkozik az idővel, a dátummal, a tulajdonjogokkal stb. Ezután, a 0x28 eltolási címen találjuk azt, ami a lényeg – a fájl adatainak lemezen elfoglalt helyét. Említettem már, hogy különféle leírótípusok tartoznak a különféle fájltípusokhoz, és hogy most csak a DEC UFS szabványos fájlokhoz tartozó fájlleírókról beszélünk?

Az ext2 vagy DEC UFS 32 bites fájlrendszer adatszerkezetében egy négybájtos szó mutat a fájlrendszer adatblokkjainak egyikére. E rész első 48 bájtja tizenkét négybájtos mutatót tartalmaz a fájl adatainak első tizenkét blokkjára. E fájlrendszer esetében minden adatblokk 8 kB hosszú. Ezeket a blokkokat egyenes blokkoknak (direct blocks) nevezik, mivel a címük közvetlenül a leírókban tárolható. Ez egyúttal azt is jelenti, hogy a kisméretű állományok (96 kB vagy kisebbek) gyorsabban érhetők el. Nagyobb fájlok esetén, a leíróban található 13. címmező már nem egy 8 kilobájtos adatblokkra mutat, hanem egy 8 kilobájt méretű címtáblára, azaz egy olyan címmezőre, ami 2048 újabb 8 kB-os adatblokkra mutat. Ezeket áttételes blokkoknak (indirect blocks) nevezik. Még nagyobb fájlok esetében a leíró 14. címmezője egy olyan 8 kB-os blokkra mutat, amelyben mind a 2048 címmező újabb 8 kB-os címblokkra mutat. Ezeket kétszeresen áttételes blokkoknak nevezik. Ez valóban nagy fájlméretet tesz lehetővé.

Az volt a gondom, hogy a Linux csak az egyenes blokkokat tudta olvasni a DEC fájl adataiból. Minden olyan fájlolvasási kísérlet, ahol az áttételes blokkokat is használni kellett volna, „elérési kísérlet az eszköz végén túlra” („attempt to access beyond end of device”) hibajelenséget okozott.

Két dologra lett volna szükségem a feladat megoldásához.

1. A lemezen elhelyezkedő adat szerkezetének megértésére, különös tekintettel a leírók szerkezetére, valamint a címadat tárolására.
2. Világosan átlátni, hogy az operációs rendszer mit tesz azzal az adattal, amit a lemezen talál, és legfőképpen miképpen olvassa a fájlleírót.

A kettő közül egyikkel sem voltam tisztában, viszont segítségképpen ott volt a Linux-közösség. Már a munka megkezdésénél tájékoztattam a felhasználói csoportom (Linux Users Los Angeles) tagjait arról, hogy milyen kihívással kell szembenéznem. Ennek hatására számos ember osztotta meg velem megérzéseit és javaslatait. *Christopher Smith* még ennél is tovább ment, küldött nekem egy levelet: „Ha szándékodban áll kinyuvasztani az egyik ilyen meghajtót és a SCSI vezérlőt, esetleg elszórakozhatok vele tengernyi szabadidőmben.” Átvittem neki egy kiszolgálót és a General Dynamicstól kapott egyik próbameghajtó másolatát.

Segítsége felbecsülhetetlennek bizonyult a feladat megoldásában. *Dan Kegel* azt ajánlotta, hogy írjak a rendszermag listájára, levele megadta a bátorságot a cikk elküldéséhez. Ő mutatta meg a linux-fsdevel (linuxos fájlrendszerek fejlesztésével foglalkozó) levelezőlistát is. Igen sok segítséget kaptam mindkét listáról.

Peter Swain írta: „Úgy tűnik, az áttételes blokkkezelés zavarodott meg, az „endian” vagy a 64 bitesség miatt. Lehet, hogy a DEC egy nem szabványos megvalósítást használ, de te biztos használhatod, ha máshogy nem, akkor egy befűzési kapcsolóval.” *Jim Nance* szintén hasonlóan gondolkodott: „Úgy hangzik, mint valami 32/64 bit hiba, esetleg valamilyen bájtsorrend a gond.” Azt is javasolta, hogy vizsgáljuk meg a rendszermag felhasználói módba ültetésével, és elküldte nekünk, hogy hol tudunk ebbe az irányba elindulni. *Peter Rival* (Tru64 QMG Performance Engineering) azt ajánlotta, hogy lépjünk kapcsolatba *Marcus Barrow*-val (Mission Critical Linux), akit a következőképpen jellemzett: „az MCL-hez pártolása előtt ő volt a helyi UFS-nagyszaki”.

Marcus Barrow e szavakkal válaszolt a levelemre: „Örömmel vetnék rá egy pillantást.” „Ne írj mindhárom próbalemezre Linux alól, ugyanis a fájlrendszered meghibásodhat” – figyelmeztetett. Természetesen akkor már klónozott meghajtókat használtunk. Marcus a tapasztaltkról is részletesen írt: Először azt gondoltam, hogy a gondokat a 8k/1k blokk/darab kezelése okozza. Második körben még jobban összezavarodtam. Nem értem, miért tudod olvasni az egyenes blokkokat, és miért nem az áttételeseket. Különösen, hogy a Linux képes a saját áttételes blokkjait olvasni.

Lye Seaman rámutatott: „A feladat lényegesen egyszerűbb lehetne, ha meglennének a megfelelő fájlok az idegen OS /usr/include/fs/* könyvtárából… Biztos megvannak valahol, valakinek a múzeumában.” A támogatás, amit ennek a két listának a tagjai nyújtottak, fontos szerepet játszott abban, hogy végül be tudtuk fejezni ezt a munkát.

Megvolt továbbá a rendszermag forráskódja, és tulajdonképpen ez tette lehetővé a megoldást. A szabad program azt jelenti, hogy követhető, mi zajlik a rendszerben és ennek köszönhetően módosítható is. Kinyomtattam a lényeges fájlokat, a linux/fs/ufs/inode.c-t

és hozzá tartozó társait, majd felütöttem a könyveket. A kutatómunka során találtam pár kitűnő anyagot a Szabad Forrás Közösségében, ezek bemutatták a fájlrendszereket, valamint hogy milyen programokat kedvelnek. (Lásd Javasolt oktatóanyagok.)
Elméletben már értettem, hogy mit csinál az OS, hiszen birtokomban volt a forráskód, amit futtattam, és nyitva állt a lehetőség, hogy megváltoztassam és új futtatható állományokat készítsek. Ahhoz, hogy megértsem, miképpen épül fel a fájlrendszer, tudnom kellett olvasni a lemezen található adatot, és meg kellett értenem, hogyan szerveződik, valamint meg kellett néznem, miképpen épül fel ténylegesen egy fájlleíró a célmeghajtón. Ki akartam olvasni egyet, és látni, hogy mi módon mutat az áttételes blokk a fájl hiányzó adataira, ha egyáltalán oda mutat.
Ekkorra már a leírók elég valóságosnak tűntek számomra. Kiterveltem, hogy elkapok egyet, nevezetesen a rakoncátlan oilpatch.tar leíróját. Mivel be tudtam fűzni a lemezterületet, majd pedig újra tudtam fűzni írható-olvasható módon a rendszermag kísérleti változatával, meg tudtam változtatni az oilpatch.tar jogait. Meg is tettem. Tudtam, hogy az általam keresett résznek valahol a lemez elején kell elhelyezkednie, ezért kimásoltam ezt a részt egy fájlba a következő paranccsal:

```
dd if=/dev/sda of=root-patch
  ➥bs=1k count=100000
```

Ezután ismét befűztem a DEC lemezterületet, megváltoztattam az oilpatch.tar tulajdonosát rendszergazdáról (ID 0) senkire (nobody) (ID 99), majd leválasztottam, és készítettem egy újabb fájlt ezzel a paranccsal:

```
dd if=/dev/sda of=nobody-patch
  ➥bs=1k count=100000
```

A két fájl közti különbségkeresés kiderítette, hogy egy bájtérték bizonyos eltolási értéknél a fájlokban 0-ról 99-re változott. A leírón belül a négybájtos fájltulajdonos-azonosító ismert helyen van, így a helyes értékekre beállított `dd` parancs a következő lehet:

```
dd if=/dev/sda
  ➥of=inodecopy_for_oilpatch.tar
  ➥bs=128 count=1
  ➥skip=offset_from_front
```

Ez kiírja a fájlleírót egy állományba, amit így részletesen megvizsgálhattam. Ki tudtam nyomtatni, elolvastam a rejtett üzeneteit és megtaláltam a fájladatokat. Szerencsére az oilpatch.tar könnyen átlátható szöveges állománynak bizonyult. A szöveg darabjai úgy illettek egymáshoz, mint egy kirakójáték darabkái, ez pedig igen hasznos, ha a helyes sorrendet szeretnénk megállapítani.
A következőket állapítottam meg: az első 12 mutató a fájl első 12 adatblokkjára mutatott, a blokk első négy bájtja a 13. mutatóra mutatott, ami pedig a fájl 13. blokkjára. A mutatók blokkjának következő négy bájtja a fájladatok 14. blokkjára mutatott és így tovább.
Nem akadt sehol semmi furcsaság a működésben: semmi vad „nagy endian, kis endian” gond, sehol egy váratlan bitcsere. Minden nagyon szép és következetes. Pontosan olyan volt, ahogyan én is megvalósítottam volna. Valójában a Linux is ezt teszi az ext2 fájlrendszerben, valamint ezt várja a rendszermag UFS kódja is, viszont ez nemigen segített megmagyarázni, tulajdonképpen miért nem működik. Sok időt töltöttem a hiba keresésével, ami nem is létezik. Lassan levontam a következtetést, hogy a hiba egyáltalán nem a rendszermag UFS-kódjában van. A leírót úgy olvasta be, ahogy azt kell. A DEC UFS-rendszert ufs_type=sun-ként lehet beolvasni. Igazság szerint az UFS-kód már jó ideje közkézen forgott, ezért nehezen tartottam elképzelhetőnek, hogy hibás. A hiba mélyebben gyökeredzik: a Linux virtuális fájlrendszer (VFS), valamint a 2.4.0-test1 rendszermag és UFS-kód közötti kapcsolattartási hibában, amit a felsőbb szintű kód megváltoztatása okozott. Az első áttételes blokk mutatójának átadása történt hibásan, ez már a VFS-hez is rosszul került el, és ez rontotta el az UFS-kódot.
Az elképzelés ellenőrzése egyben a hiba kiküszöbölését is jelentette számomra. Először a 2.3.99-es majd a 2.4.-test sorozatú rendszermagokat használtam, mivel ezekben megtalálható az OSF lemezrészkezelés, mely a 2.2 változatból hiányzik. Ha igazam volt, és az UFS-kód csak mostanában romlott el, a 2.2 rendszermag hiba nélkül fogja olvasni a DEC-meghajtót, feltéve, ha tudja kezelni a lemez felosztási tábláját. Az OSF-felosztást kezelő kód visszaültetése 2.4-es rendszermagról 2.2.16 forrásba igen könnyű volt – még számomra is –, és az eredményül kapott 2.2.16-os rendszermag mindent hibátlanul hajtott végre a DEC OFS-meghajtókon, beleértve a nagy fájlok naphosszat tartó írását és olvasását. Mivel a 2.2.16 jobb választás egy ipari környezetben, ráadásul én is ilyen feladatra szántam, továbbléphettünk, és be tudtuk fejezni a munkát. Ezután elkezdhettük szállítani a 2.2.16-os rendszermaggal ellátott gépeket a General Dynamics és ügyfelei számára.
Természetesen megjelentettem a 2.4.0-test1-ben fellelhető hibát a linuxkernel és a linux-fsdevel listán. Írtam, hogy „sajnos, találtam egy hibát”. *Alan Cox* azonban így válaszolt: „Egy hiba felfedezése mindig jó hír. Nagy valószínűséggel ez a hiba nem derült volna ki a 2.4.0 kiadása előtt.”
Nagyjából ennyi a történet.
A munka során a későbbiekben még néhány apróbb megoldásra váró feladat akadt. Ilyen volt, amikor az első rendszerünk már működött, akadt még egy kis gond azzal, hogy a Windows 2000 munkaállomások a hosszú fájlneveket kezelik, de a sambás fiúk felkészültek a Microsoft legújabb trükkjeire, így a legújabb Samba-változatokra frissítés megoldotta a gondot.
A továbbiakban rögzített kapcsolatokat kellett létrehoznunk a meghatározott SCSI-azonosítójú DEC-meghajtók és a befűzési pontok közt, a telepített DEC-meghajtók számától függetlenül. Mivel a Linux a SCSI merevlemezeket sda, sdb, sdc neveken szereti emlegetni, olyan sorrendben, ahogy megtalálja őket, a lemezek befűzését egy kisebb programmal kellett megoldani. Az eredeti Tekram vezérlők nem igazán örültek a külső rázkódásmentesített kereteknek, így Adaptecre cseréltük azokat. Ezek természetesen átlagos kihívásoknak számítanak ebben a szakmában. A valódit a fájlleírók megrendszabályozása jelentette.

Javasolt oktatóanyagok

Rémy Cardtól a Design and Implementation of the Second Extended File System (Laboratoire MASI–Institut Blaise Pascal) ➲ card@masi.ibp.fr Theodore Ts'o (Massachusetts Institute of Technology) ➲ tytso@mit.edu és Stephen Tweedie (University of Edinburgh) ➲ sct@dcs.ed.ac.uk ➲ http://khg.redhat.com/HyperNews/get/fs/ext2intro.html

David A. Ruslingtól a The Linux Kernel ➲ david.rusling@arm.com, Chapter 9: The File System ➲ http://www.linuxdoc.org/LDP/tlk/tlk.html

Clay J. Claiborne, Jr. (cjc@linuxbeach.net) a Cosmos Engineering Company munkatársa. Harminc éve dolgozik a számítógépiparban. 1995 óta Linux-rajongó, 1996-ban megalkotta a Linux on Disket. Ezenkívül Linux-oktató a Los Angeles City College-ben, valamint a Linux Users Los Angeles (LULA) elnöke.

Betölthető rendszermagmodulok programozása és rendszerhívás-elfogás

A jelenlegi processzorok két üzemmódban képesek működni: rendszermag és felhasználói üzemmódban. Rendszermag üzemmódban több utasítást használhatunk, ezenkívül az egész memóriát és az összes regisztert elérhetjük. A megszakításvezérlők és az operációs rendszer szolgáltatásai is a rendszermag üzemmódban (röviden magmódban) futnak. Ezzel szemben, ha a processzor felhasználói módban fut, akkor kisebb utasításkészlettel dolgozhatunk, a memóriának és az eszközöknek pedig csak egy részét látja a processzor. Ezt az üzemmódot a könyvtárfüggvények és a felhasználói programok használják. E két üzemmód teremti meg a mai operációs rendszerekben a biztonságot és megbízhatóságot.

A programok legtöbbször felhasználói módban futnak. Magmódba csak különleges, az operációs rendszerrel kapcsolatos feladatok elvégzéséhez váltanak át.

Az operációs rendszer szolgáltatásait rendszerhívásokon keresztül érhetjük el. A rendszerhívások a rendszermaghoz vezető „átjárók”, melyeket programból megvalósított megszakításokkal hoztak létre, az operációs rendszer pedig magmódban kezeli azokat.

Az operációs rendszer rendszerhívási táblázatot tart fönn, ennek mutatói a rendszermagban lévő, a hívásokat végrehajtó rendszerfüggvényekre hivatkoznak. A program számára ez a táblázat teszi elérhetővé az operációs rendszer szolgáltatásait. A különféle rendszerhívások listáját a `/usr/include/sys/syscall.h` fájlban találjuk meg. Linuxban ez a fájl a `/usr/include/bits/syscall.h` fájlt is magában foglalja.

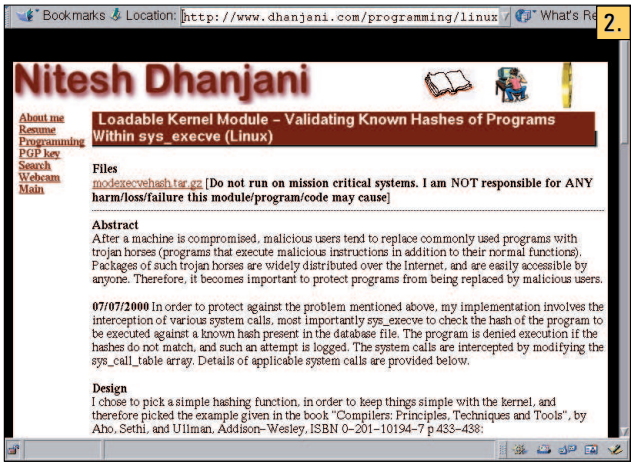
A betölthető modulok olyan kódrészek, melyeket szükség esetén a rendszermagba tölthetünk. Ezek további képességekkel bővítik a magot, anélkül, hogy a gépet újra kellene indítanunk. Linuxban a modulokat gyakran alkalmazzák új eszközmeghajtók elkészítéséhez. A betölthető modulok használata helyett a bővítéseket fordítjuk be közvetlenül a rendszermagba, így mindig egyetlen fájllal dolgozhatunk. Ennek viszont az a hátránya, hogy minden bővítés alkalmával újra kell fordítanunk a magot.

A mag programozása nemcsak rendkívüli összetettsége miatt nehéz, hanem azért is, mert a hibakeresés nagyon sok időt vesz igénybe. Az operációs rendszer hibakereséséhez minden új mag telepítése után újra kell indítanunk a gépet. A fejlesztéshez ezért mindenképpen a betölthető modulokat ajánljuk, hiszen így csak ritkán kell újrafordítanunk a magot, és a gép újraindítására sincsen szükség. A másik fontos ok, hogy mivel a felhasználónak nem kell a létező magot eltávolítania vagy helyettesítenie, sokkal szívesebben veszi a modulokkal megvalósított új lehetőségeket igénybe.

A mag modultámogatásában fontos szerepet játszik a rendszerhívások elfogása, ennek a lehetőségnek az alábbi példákhoz hasonló módon vehetjük hasznát. Megjegyeznénk, hogy a továbbiak megértéséhez szükséges némi C programozói ismeret.

A rendszerhívások

Minden operációs rendszerben a rendszerhívások teszik lehetővé, hogy a felhasználói programok közvetlenül elérjék a mag szolgáltatásait. Különbséget kell tennünk a rendszerhívások (system calls) és a könyvtárfüggvények (library functions) között. A könyvtárfüggvények egy adott programhoz kapcsolódnak és „hordozhatóbbak”, hiszen nem függenek a magtól. Azonban sok könyvtárfüggvény



rendszerhívásokkal hajt végre bizonyos feladatokat a magban. Ennek bemutatására nézzük meg most az alábbi C programot (*pelda1. c)*), mely megnyit egy fájlt és kiírja annak tartalmát:

```
#include <stdio.h>

int main(void)
{
   FILE *myfile
   char tempstring[1024]

   if(!(myfile=fopen("/etc/passwd","r")))
   {
      fprintf(stderr,"Nem lehet megnyiti a
       fájlt\n");
      exit(1);
}

while(!feof(myfile))
{
   fscanf(myfile,"%s\n",tempstring);
```

```
    fprintf(stdout,"%s\n",tempstring);
}

exit(0);
}
```

Ebben a programban az fopen függvénnyel nyitottuk meg a `/etc/passwd` fájlt. Fontos azonban megjegyezni, hogy az fopen nem rendszerhívás. Az fopen az open rendszerhívást hívja meg a tulajdonképpeni I/O művelet elvégzéséhez. Egz program által meghívott rendszerhívások listáját a strace programmal jeleníthetjük meg. Azt feltételezve, hogy a fenti programot `a.out` néven fordítottuk le a gcc-vel, a `strace ./a.out` parancs hatására az `a.out` által használt rendszerhívásokat tekinthetjük meg.
A mag a rendszerhívás segítségével vált át a folyamat gazdájának felhasználói azonosítójára (userid). Ha tehát a fenti programot egy közönséges felhasználó indítaná el, értékként az általa nem olvasható `/etc/shadow` fájlt adva meg, az open nem lenne képes a feladatát végrehajtani, így a fenti if feltétel „igaz” értéket kapna, és a „nem lehet megnyitni a fájlt” hibaüzenet jelenne meg.

Példa a rendszerhívások elfogására a betölthető modulokon keresztül

Tegyük fel, hogy az exit rendszerhívást szeretnénk elfogni, s minden ezt meghívó folyamatnak egy, a konzolra kiírandó üzenetet kívánunk átadni. Ehhez meg kell írnunk saját exit rendszerhívásunkat, majd meg kell adnunk a magnak, hogy az eredeti exit helyett a sajátunkat hívja. A saját exit hívás végén akár az eredeti exitet is meghívhatjuk. Ehhez a rendszerhívási táblázatot (sys_call_table) kell módosítanunk. Nézzük meg a `/usr/src/linux/arch/i386/kernel/entry.S` fájlt (természetesen csak akkor, ha i386-rendszerű gépen dolgozunk). Ez a fájl a mag összes rendszerhívását és a sys_call_table-ben elfoglalt helyüket tartalmazza.
A sys_call_table-t úgy kell módosítanunk, hogy a sys_exit a saját exit hívásunkra mutasson. Az eredeti sys_call-ra mutató hivatkozást mentenünk kell és saját rendszerhívásunk végén, ennek kell átadnunk a vezérlést. A fentieket az *pelda2 c*-ben látható kóddal oldhatjuk meg. A példa programját a `gcc -Wall -DMODULE -D__KERNEL__ -DLINUX -c pelda2.c` paranccsal fordíthatjuk le. Ekkor a pelda2.o nevű modul jön létre, melyet a rendszermagba úgy tölthetünk be, hogy rendszergazdaként kiadjuk az `insmod pelda2.o` parancsot. Most győződjünk meg arról, hogy a konzolon vagyunk-e (hiszen a printk csak a konzolra ír), majd futtassunk egy olyan programot, mely az exit rendszerhívást használja. Például az `ls` parancs kiadására a `„Szia! A sys_exit hívásakor ez volt a hibakód: 0”` üzenet jelenik meg.
Most adjuk ki az `ls` parancsot egy nem létező fájlra, így a program 0-tól különböző hibakóddal lép ki. Tehát az `ls nem_létező_fájl` hatására a `„Szia! A sys_exit hivasakor ez volt a hibakod: 1”` szöveg jelenik meg.
A betöltött modulok listáját az `lsmod` paranccsal tekinthetjük meg. A modul eltávolításához az `rmmod pelda2` parancsot használjuk.

Egy érdekesb példa: a sys_execve elfogása a trójai falovakkal szembeni védelemhez

A trójai falovak olyan programok, melyek valamelyik rendszerszolgáltatásba vagy programba beépülve káros kóddal egészítik ki azokat. Ha egy gépre betörnek, akkor a rossz szándékú behatoló gyakran helyez el trójai falovakat a rendszerben. Ezeket sok helyről le lehet tölteni, tehát mindenképpen meg kell akadályoznunk, hogy kedves ajándékként valaki ezekkel „bővítse” programjainkat.
Következő példánkban is a rendszerhívások elfogásával foglalkozunk: a védelem megvalósításáért a `sys_execve` segítségével

pelda2.c

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <sys/syscall.h>

extern void *sys_call_table[];


asmlinkage int (*eredeti_sys_exit)(int);

asmlinkage int sajat_exit (int error_code)
{


       /*minden meghíváskor kiírja a szöveget a
       konzolra*/
       printk ("Szia! A sys_exit hívásakor ez
       ➥volt a hibakód:  %d\n",error_code);
       /* az eredeti sys_exit-et is meghívjuk*/
       return eredeti_sys_exit(error_code);
}

/* ez a függvény a modul betöltésekor kerül
   meghívásra */
int init_module()
{
       /* mentjük az eredeti sys_exit-re
          mutató hivatkozást */

eredeti_sys_exit=sys_call_table[__NR_exit];

       /* módosítjuk a sys_call_table-t, hogy
          az eredeti sys_exit
          helyett a sajátunkat hívja */

sys_call_table[__NR_exit]=sajat_exit_fuggveny;
}

/* ez a függvény a modul eltávolításakor kerül
   meghívásra */
void cleanup_module()
{
/* a modul eltávolításakor az __NR_exit megint
   az eredeti *sys_exit-re mutat */

sys_call_table[__NR_exit]=eredeti_sys_exit;
}
```

ellenőrizzük, hogy a programhoz tartozó indexelés egyezik-e az előzőleg egy adatbázisban tárolt indexeléssel. Ha a két adat nem egyezik, a program nem indulhat el, és minden ilyen próbálkozásról naplóbejegyzés készül. A védelmet például az alábbi lépésekkel valósíthatjuk meg:

1. Elfogjuk a `sys_execve` hívást, majd a futtatandó fájl kiszámított fájlleíróját (inode) összehasonlítjuk az adatbázisban található értékkel. A fájlleírók a fájlrendszer adatszerkezetei, a fájlokkal kapcsolatos adatokat tartalmazzák. Mivel minden fájlhoz egyedi fájlleíró tartozik, az összehasonlítás eredményéből pontosan meghatározhatjuk a következő lépést. Ha nincsen találat, akkor

az eredeti `sys_execve` meghívása után visszatérhetünk. Ha azonban találunk ilyet, akkor kiszámítjuk a programhoz tartozó indexelést, majd ezt összehasonlítjuk az indexelt adatbázissal. Ha itt is van találat, akkor meghívjuk az eredeti `sys_execve`-t és visszatérünk. Ha nincs találat, akkor a próbálkozásról naplóbejegyzést készítünk, és hibajelzéssel visszatérünk.

2. Elfogjuk a `sys_delete_module` hívást. Ha a modul nevével hívtak bennünket, akkor hibajelzéssel térünk vissza. A modult nem engedjük törölni.
3. Elfogjuk a `sys_create_module` hívást és hibajelzéssel térünk vissza. Az új modulok beillesztését megtiltjuk, hiszen nem szeretnénk, hogy egy rossz szándékú programozó modulja elfogja az 1. lépésben említett `sys_execve` hívást.
4. Elfogjuk a `sys_open` hívást, így megelőzhetjük azt, hogy az indexelt adatbázist vagy a naplófájlt bármilyen program illetéktelenül megnyissa írásra.
5. A `sys_unlink` hívás elfogásával megakadályozzuk az idexelt adatbázis vagy a naplófájl törlését.

Tartsuk szem előtt, hogy a fenti módszer nem jelent teljes körű védelmet; de első nekifutásra nem is rossz. Egy rossz szándékú felhasználó például módosíthatja a `/dev/kmem` magszimbólumait, vagy közvetlen eszközeléréssel írhat a lemezre, s így az open megkerülésével is írhat az indexelt adatbázisba. Vagy, mivel ez a példa egy betölthető modul, a behatoló egyszerűen letilthatja a modul rendszerindításkor történő betöltését a `/etc/rc.d` fájlok módosításával. Mindezek mellett még számos más rendszerhívással módosítható vagy törölhető az indexelt adatbázis vagy a naplófájl.

Ami a legfontosabb: legyünk tisztában azzal, hogy a betölthető modulokat a behatoló saját terveinek végrehajtására is felhasználhatja. Például a `sys_execve` függvényhívás elfogásával trójai falovat, a read és write rendszerhívások elfogásával pedig billentyűzetfigyelő eljárást építhet be a rendszerbe. Behatolás esetén a betölthető modulok rugalmassága és hatékonysága tehát komoly veszélyforrást jelent. A Kapcsolódó címek részben felsorolt honlapokon további példaprogramokat is találhatunk a témával kapcsolatban.

Gustavo Rodriguez-Rivera (grr@cs.purdue.edu) a Purdue Egyetem vendégprofesszora és a Geodesic Systems programmérnöke. Érdeklődési körébe az operációs rendszerek, a hálózat- és memóriakezelés tartozik.

Nitesh Dhanjani (dhanjani@dhanjani.com) a Purdue Egyetem végzős hallgatója. Operációs rendszerekkel, hálózatokkal és a biztonsággal foglalkozik. Több cég, így például az Ernst & Young LLP, számára végzett már biztonsági felméréseket, szabadidejében pedig tanácsadást is vállal.

Kapcsolódó címek

Betölthető magmodulok Linux alatt *(1. kép)*
➲ http://packetstorm.securify.com/groups/thc/LKM_HACKING.html

Linux Kernel Module Programming Guide (Ori Pomerantz)
➲ http://howto.tucows.com/LDP/LDP/lkmpg/

Nitesh Dhanjani munkái *(2. kép)*
➲ http://www.dhanjani.com/programming/linuxexechash/

Várakozás elkerülése minden 21. újraindításnál

Mikor a rendszer az újraindításkor befűzi a lemezrészeket, általában húsz alkalommal kimarad az fsck futtatása, a huszonegyedik újraindításkor viszont az összes fájlrendszert leellenőrzi.
Ez a hosszas várakozás bizony bosszantó lehet. Honnan lehet tudni, hogy hányadik befűzésnél tartasz egy bizonyos fájlrendszer esetében? Írd be:

```
# dumpe2fs /dev/hda7 | grep
          ➥'[mM]ount count'
dumpe2fs 1.19, 13-Jul-2000 for EXT2 FS
0.5b, 95/08/09
Mount count:                7
Maximum mount count:        20
```

Látható, hogy a /dev/hda7 az utolsó fsck óta hét alkalommal lett befűzve, és az fsck húsz alkalommal ugorja át az ellenőrzést.
Ha az összes fájlrendszerednek azonos a befűzés-számlálója (mount count), akkor a rendszer valamennyit egyszerre fogja ellenőrizni.
Ezen könnyű segíteni:

```
# umount /dev/hda6
# tune2fs -C 9 /dev/hda6
tune2fs 1.19, 13-Jul-2000 for EXT2 FS
0.5b, 95/08/09
Setting current mount count to 9
# mount /dev/hda6
```

Így a befűzés-számlálót 9-re állíthatod.
Figyelem! A tune2fs programot kizárólag befűzetlen fájlrendszeren futassuk. A tune2fs akkor is végrehajtja feladatát, ha a fájlrendszer használatban van, de gyanítom, hogy ez veszélyes, szóval tőled függ, óvatos leszel-e.
Tételezzük fel, hogy négy fájlrendszered van. Állítsd be a befűzés-számlálókat a következőképpen: 1,6,11,16. Ezáltal az ellenőrzés egyenletesen oszlik meg közöttük.
A fentiek végrehajtásával a várható újraindítási idő azonos marad, csak az indítási idő egyenetlensége csökkent. Az, hogy kedvelni fogod-e ezt a módszert, attól függ, mennyire vagy türelmetlen, de az eredeti megoldásnál kétségkívül jobb.
Ha igazi rögeszmés vagy és szereted, ha sokszor fut az fsck, vagy ha több mint 20 fájlrendszered van, a legnagyobb befűzési számot is megváltoztathatod a `tune2fs -c N` végrehajtásával. Az `N=-1` érték kikapcsolja az ellenőrzést. Jó, ha tudod azt is, hogy a `tune2fs -i 2` jelentése: „kétnaponta ellenőrizz”. Ez akkor jöhet jól, ha például hordozható számítógépet használsz.

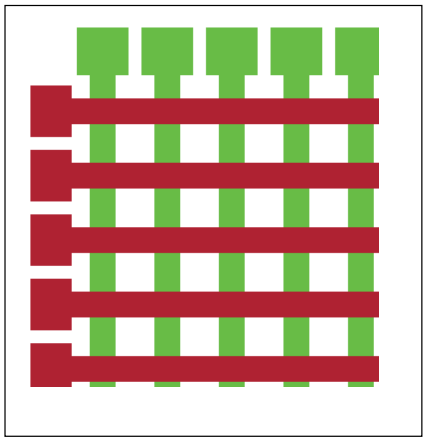
Linux futtatása sérült memóriával

A BadRAM folt segítségével tökéletesen lehet Linuxot futtatni olyan hibás memóriával, ami egyébként sok galibát okozhat.

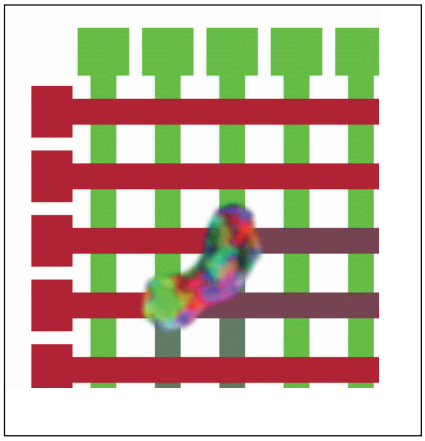
A memóriamodulok meghibásodásának számos oka lehet. Mielőtt orvosolnánk ezeket a hibákat, nézzük meg az okokat. Megértésükhöz vessünk egy pillantást az *1. ábrára*, amelyen a hibátlan memória vázlatos ábráját láthatjuk. A memóriában kiterítve több, majdnem ugyanannyi sor és oszlop található. A memóriacellában minden kereszt általában egy bitet tárol. Ebből következően, egy adott memóriacella címe a szóban forgó sor és oszlop címéből tevődik össze. A memóriamodulokba sorban, egymás után betápláljuk ezt a két „félcímet", mivel a címsín csak fele olyan széles, mint szeretnénk. Ezt a címfelezést egyébként az alaplap végzi.
A memóriamodulok hibáinak okait tekintve a gyártás áll az első helyen. Ez nagyon érzékeny eljárás, főleg a lapka fizikai méretei miatt, mivel a világban egyre nagyobb teljesítőképességű memóriákat igényelnek. A lapka gyártói nem takarékoskodnak a környezeti erőforrásokkal, mivel elég sok erősen tisztított vegyszert használnak fel egyetlen apró kis „homokszármazék" előállításához. Ezenkívül a lapka érzékenysége miatt viszonylag sok már a gyártás során meghibásodik.
Az egyik részleges megoldás az lehet, hogy adatismétléses tárolással látják el a lapkákat, azaz minden harminckét memóriacella-sor tartalmaz egy 33. sort is, így ha egy sor meghibásodik, a jelet átirányítja a külön sorra. Ezt a módszert alkalmazza néhány gyártó, ennek ellenére bizonyos meg nem erősített hírek szerint a selejt aránya még így is hatvan százalék körül mozog. Talán felesleges is mondanom, hogy az ilyen és hasonló okok miatt a memóriagyártás nem olcsó mulatság.
A gyártás során keletkezett hibákat okozhatja egy apró szennyeződés is, amely a levilágítás vagy a fejlesztés során kerülhet az egyik rétegre, ahogy az a *2. ábrán* is látható. Az árnyékolt terület, amely hibás működéshez vezethet, egy apró szennyeződés eredménye.
A másik hibaforrás a kisülés. Ez ugyanaz a jelenség, mint amit egy pamut vagy nejlonpulóver levételekor tapasztalhatunk. A feltöltődés magas feszültségen, egymáshoz közeli felületeknél jöhet létre. Ha a felületek elég közel kerülnek egymáshoz, a töltés hirtelen átugrik, ionizálja a levegőt, kisül, és nagyon rövid ideig igen nagy feszültséget gerjeszt. Teljesen ugyanaz, mint a villámlás, csak természetesen nem olyan erős. A lapka a kis mérete miatt igen érzékeny ezekre az erős kisülésekre. A kisülés a táregységeket, a sorok és oszlopok közötti összeköttetést, valamint a címválasztó logikát teszi tönkre, amint az a *3. ábrán* is látható. Ez a gyakorlatban azt jelenti, hogy az egész sor, az oszlop vagy mindkettő használhatatlanná válik, itt is ezt jelzi az árnyékolt terület. Ezekkel a memóriahibákkal találkozhatunk a leggyakrabban.
Az utolsó hibaforrás, amivel eddig még személyesen nem is találkoztam, a lapka fokozatos elöregedése. Ilyenkor a lapkán az egyébként élesen szétválasztott áramkörök összemosódnak, keverednek. Ez egy természetes folyamat, és ha a lapkát kellően hűvös helyen használjuk, valószínűleg egy-két évtized is kell hozzá, hogy bekövetkezzen. A memória esetében ez talán nem is olyan nagy gond, hiszen ennyi idő alatt már úgyis elavulttá válik a sebessége, illetve a mérete miatt.
Az ilyen sérüléseknél a legfontosabb dolog, hogy nem okoznak véletlenszerű hibákat. Az előfordulhat, hogy a hibás bitek valamilyen minta szerint jelentkeznek, de ha egyszer hibásak, akkor azok is maradnak.
Fontos, hogy a hibák nem keletkeznek csak úgy maguktól. A bajt a hirtelen kicsapódó energia okozza, vagy egy nagyon lassú folyamat. Szóval nem kell arra számítanunk, hogy folyamatosan egymás után jelentkeznek majd a hibák, még akkor sem, ha a modulunk sérült. Mivel a hibák a memóriában nem terjeszkednek, általában kijátszhatók egy ügyes dinamikus memóriafoglalási módszerrel.

A memóriahibák keresése

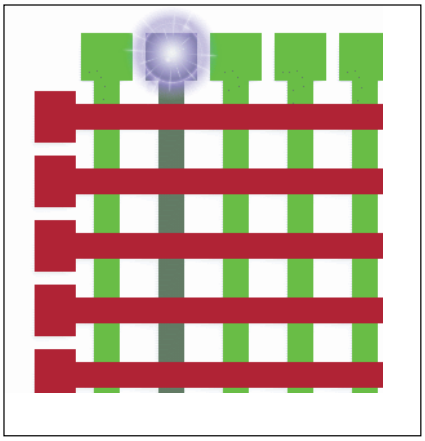
Alapjában véve a memóriahibák keresésére két eljárást használhatunk. Az egyik lehetőség egy program használata, mely átfésüli a teljes memóriát, és jelenti a hibás címeket. Ilyen program az i386-os rendszerekhez készült *memtest86,* mely a mezőnyből hibakeresési képességeivel tűnik ki. Természetesen az ellenőrzés megbízhatósága függ a hiba állandóságától is, de ez általában megfelelő, én már hónapok óta használom a gépem anélkül, hogy változtak volna a hibák a modulomban.



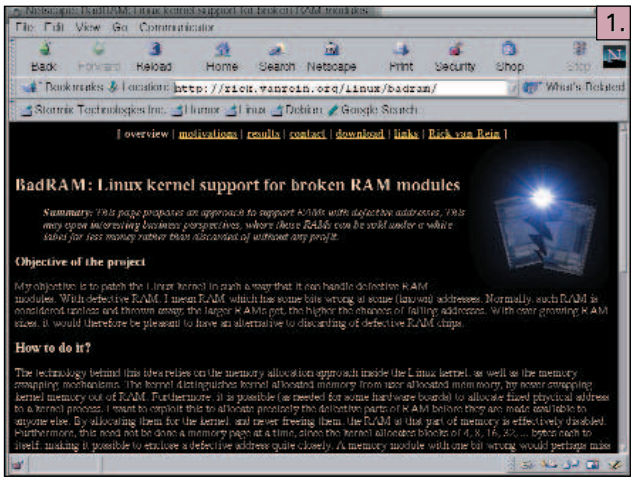
1. ábra Hibátlan memóriaszerkezet

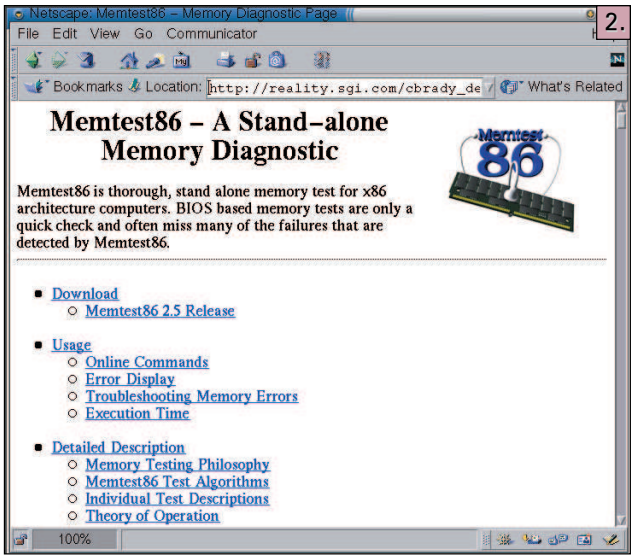


2. ábra Memóriamaratási hiba



3. ábra Az átmeneti tár sérülése kisülés következtében





A másik lehetőség a gépi hibakeresés. A harminctűs SIMM modulok korában elterjedt megoldás volt, hogy használtak egy kilencedik bitet a párosságadat (parity) tárolására. A kilencedik bit lényege, hogy külön helyezkedik el a memóriamodulon, és a tárolt adatok páros (vagy éppen páratlan) voltát jelzi. Íráskor ezt az adatot is újraszámolja a rendszer, olvasáskor pedig ellenőrzi: „párossá” teszi az egész lapkát – vagy páratlanná, ahogy éppen az alaplap kívánja. Íráskor ez a bit megjelenik, és visszaolvasáskor ellenőrzésre kerül. Ha olvasáskor az ellenőrzés hibát jelez, a rendszer párossági hibát vált ki.
A párosságjelző korszerű megfelelője az ECC (Error Correction Code, hibajavító kód). Ez általában valamilyen CRC-jegyzék a tárolt bitekről, és a 32-ből három hibás bit felderítésére alkalmas, valamint jól javít legfeljebb két hibás bitet. (Nem vagyok teljesen biztos a pontos értékekben, de az elmélet a lényeg.) Tehát az ECC RAM-mal lehetséges a hibák felderítése, és jól is javítja a kisebbeket.
Az újabb memóriamodulokban, melyeket a PC munkaállomásokban is használunk, általában nincsen sem párosság-ellenőrzés, sem pedig ECC. Másrészről a csúcskategóriás kiszolgálókban, mint a VA Linux és a Sun, rendszeresen találhatunk ECC-vel ellátott, de legalább párosságjelző memóriákat. Ha jól tudom, a legtöbb PC képes lenne ECC-memóriával működni, de többnyire nem használunk ilyen modulokat, ugyanis ezek drágábbak egyszerűbb testvéreiknél. Igen, az olcsó PC-k világa visszakacsint ránk. A párosságjelző vagy ECC-RAM tulajdonságaira támaszkodva működés közben is felderíthetők lennének a hibák, és nem volna szükség ellenőrző programokra (mint amilyen a memtest86), de a BadRAM foltot, amit ebben a cikkben tárgyalunk, a lehető legegyszerűbb szerkezetben kívántuk megtartani, hátha bekerülhet a rendszermagba, ezért nem tudtunk külön foglalkozni benne az ECC különleges lehetőségeivel.

A hibák megfogalmazása mintákban

Tegyük fel, hogy a memtest86-hoz hasonló memóriaellenőrzők a modulokban keresnek hibákat. Amint azt említettem, a kereső kigyűjti a fellelt hibás címeket. Mivel egy egész sor vagy oszlop is meghibásodhat egy kisülés következtében, ilyenkor bizony hatalmas listák keletkezhetnek. Természetesen az ilyen listák unalmasak, és az esetlegesen fellépő hibák miatt veszélyes volna a rendszermagba gépelni azokat. Sőt, a rendszerindításkor rendelkezésre álló korlátozott erőforrások miatt gyakran használhatatlanok is. Az lenne a tökéletes, ha az összes fellelt hibáról készíthetnénk egy rövid kis leírást. Szerencsére a memóriahibák általában szabályos alakzatokban helyezkednek el, például a `0x1234` címtől kezdve, minden `0x0040` bájtonkénti előfordulással, de ennél bonyolultabb mintázat is kialakulhat. A szabályosságot akkor a legkönnyebb megállapítani, ha binárisan vizsgáljuk az adatokat.
Ennek oka, hogy a sorok és oszlopok címzése úgy történik, hogy címbiteket küldünk e vezetékekre, így a cím egy részének megfejtésével eldönthetjük, hogy a használt címvezetékek megfelelő területre esnek-e vagy sem.
Egy egyszerű hiba tulajdonképpen egy olyan cím, amelynek minden bitje értékes adat. Például a `0x1234,0xffff`, ahol az első szám az alapcím, és a második a maszk. A maszk „1” bitje jelzi, hogy melyek a megfelelő bázisalapcím használható bitjei. Ha ezen a címen kívül a `0x0040`-nel nagyobb cím is rossz, akkor ennek jelzésére csak módosítanunk kell a maszkot. A cím/maszk pár most `0x1234,0xffbf` lett. Láthatjuk, hogy ez helyes, mert a lefedett címek mindegyike A cím, ahol `(A & 0xffbf)=0x1234`, vagy pontosabban `0x1234` és `0x1274`. Ha ott tizenhat hibás címet találunk ezzel a kitöltési tényezővel, és az egész a `0x1234,0xfc3f` címen kezdődik, már meg is vannak a hibás címek: `0x1234, 0x1274, ..., 0x15f4`.
Ez az eljárás jól működik sorok és oszlopok hibáinak keresésénél, vagy ha egy apró szennyeződés kikapcsol néhány szomszédos bitet a lapkán. De mi történik, ha a modulban további hibák is vannak? Vagy ha több modulban is akadnak hibák? A tények feltárásához legtöbbször feljegyezzük az említett cím/maszk párt. Már öt ilyen pár alapján is elindulhatunk. Öt párral általában nem fordulnak elő különleges nehézségek.
A memtest86 a 2.3-as változattól indul, és képes előállítani BadRAM mintákat. Ezeket közvetlenül a parancssorba írhatjuk be, ha a Linux rendszermagba befordítjuk a BadRAM csomagot. A fenti példában a memtest86 jelenti a mintákat, általában így:

```
badram=0x1234,0xfb3f
```

Írjuk ezt le. Ha megvannak az adatok, indítsuk újra a rendszert, és a parancssorba írjuk be a következőket:

```
LILO: linux badram=0x1234,0xfb3f
```

Ha a rendszer gond nélkül betöltődött, tiltsuk le a hibás memóriarészeket. Ezek után az `/etc/lilo.conf` fájlt is bővíthetjük az alábbi sorral:

```
append="badram=0x1234,0xfb3f"
```

A módosítás után futtassuk a LILO-t.! A következő rendszerbetöltéseknél már nem kell beírnunk a cím/maszk párokat.

A rendszermag javítása

Az én régi ZX Spectrum számítógépemet nagyon egyszerűen bővíthettem további 32 kB memóriával. A Sinclair memóriabővítő készlet egy 64 kB tárméretű lapkából állt, amiben mind az alsó, mind pedig a felső egység tovább dolgozhatott, ha az egyik meghibásodott, és az ára is kedvezőbb volt, mint az „igazi” 32 kB-os bővítő lapkának. Az alaplapon kapcsoló segítségével beállíthattam, hogy a modul melyik felét akartam használni. Alapjában véve a BadRAM ugyanezt teszi, csak kicsit ravaszabban.

A memóriakezelő egység, mely minden Linux-változat nélkülözhetetlen része, átirányítja a felhasználó programja által elérhető „felhasználói területet” a megfelelő fizikai memórialapra. Ez nagy, összefüggő memóriaterületnek látszik, de valójában darabokban helyezkedik el a memóriamodulban (és néha az átmeneti tárolóba másolódik). Ha egy felhasználói szintű program elfoglalja a memóriát, a rendszermag egyszerűen oldalakat ad neki a fizikai memóriából.

A Linux egyetlen, fizikai memóriacímzéssel működő része a rendszermag, ez a memória elejére töltődik be – természetesen itt nem lehetnek hibák –, és a felhasználói programok számára fenntartott memóriaterületből foglal el blokkokat.

Ez a memória rendszerbetöltéskor töltődik fel fizikai lapokkal. Valójában a BadRAM azokat a lapokat hagyja ki, amelyek a parancssorban beírt cím/maszk pároknak megfelelnek.

Ez azt jelenti, hogy a BadRAM tulajdonképpen a rendszerbetöltéskor végzi el a feladatát. Tehát egyetlen hatása, hogy a szabad memóriából elfoglal egy csipetnyit. Az alapos ellenőrzőprogramok képesek ezt kimutatni, de a teljesítményre nincs érezhető hatással.

Különböző alkalmazások

Ha ehhez hasonló csomagokat készítesz, és véletlenül a SlashDot is megírja, bizonyára csomó érdekes levelet kapsz, néha érdekfeszítő gondolatokkal. Egyik javaslat, hogy rendszerbetöltés közben végezzünk memóriavizsgálatot. Mókás ötlet, de nem célszerű.

A memtest86 kiváló munkát végez, de elnyammog néhány óráig. Ezt nyilván senki sem szeretné kivárni a rendszer indításakor, de rossz memóriaellenőrzést sem akarunk. (Számos PC BIOS-szal próbálkoztunk, és a BadRAM általában elvégezte az ellenőrzést.)

A memtest86-ot indító LILO-bejegyzés, vagy a memtest86 indítólemez mindig a kedvenc megoldásaim közé tartoztak. Érkezett néhány jelentés olyan alaplapokról, melyek hibáznak egy bizonyos közvetlen memóriacímen, amit egy alaplapra szerelt eszköz rövidzárlata okoz. Az illető azt írta, hogy a hibás címek elkerülésére négy 512 megabájtos modult tett a számítógépébe, de a gondját igazából a BadRAM oldotta meg, egyetlen memórialap kihagyásával. A két gigabájtból letiltott négy kilobájtot, és a gép most gond nélkül működik.

Beszéltem valakivel, akinek valamelyik „PC otthonra!” programból származik a számítógépe. Az átlagos felszereltségű Compaq Deskpránál nem lehet kikapcsolni a néhány régebbi ISA kártya által igényelt 15–16 M közötti kiterjesztett memóriát. Szóval, a 24 megabájtra bővített memóriával nem működik a Linux 2.2 mem=24 M beállítása, mivel a 15–16 MB közötti területet is használná. De miután a `badram=0x00f00000, 0xfff00000` sorokkal a rendszermag tudomására hozta a hibát, a számítógép készen állt a memóriabővítésre, és most jobban megy, mint valaha.

Kaptam visszajelzéseket olyan emberektől, akik régebbi gépeiken párosságellenőrző és ECC RAM-okat is használtak, memóriahibákat tapasztaltak és a hibás lapok megbízhatatlanná váltak. Programkód betöltése közben a memórialap „under evaluation” állapotba került, aztán az az ötletük támadt, hogy a programkód a merevlemezről másolódik be, és hiba esetén onnan visszaállítható. Jól is működött a dolog egy darabig, amint eltűnt a hiba, a lap megint „megbízhatóvá” vált. Ha azonban a program tárolása sem működött, a lap kimaradt a használatból. Ez a megoldás érdekesnek tűnik, de futásidőben követelne processzorteljesítményt, így kénytelenek vagyunk fényűző szolgáltatásnak minősíteni.

Lehetséges bővítések a jövőben

Van néhány eljárás, amellyel növelhető a BadRAM hatékonysága. Az ECC modulok jól használhatók a biztonság növelésére a javítható és a javíthatatlan hibákhoz egyaránt. Mivel nekem sajnos egyik memóriamodulom sincs, a dolog meghaladja lehetőségeimet. Mivel a processzorteljesítményre folyamatosan van szükség, én ezt a szolgáltatást is a fényűzés tárgykörbe sorolnám.

Másik lehetőség a magtáblafoglaló-kezelő (slab allocator) fejlesztése. A táblák kicsi, egyforma és újra felhasználható memóriablokkok, amelyeket a rendszermag tömbökbe rendez. Lehetőség nyílna arra, hogy a hibás címekkel kapcsolatos adatot részletesebben nyerjük ki, ha a táblákhoz BadRAM oldalakat használnánk, és így elkerülhető lenne a hibás címekre eső táblák lefoglalása.

A BadRAM cím hibaadatait lehet bővíteni a lapokkal, megelőzve a hibás címek felülírását.

Eszményi esetben a memóriaveszteséget nullára csökkenthetnénk. A gyakorlatban viszont, a BadRAM így sem végezne jobb munkát, mivel egy átlagos rendszer nem használja egyszerre az összes memórialapot. Éppen ezért kétlem, hogy egyelőre a dolog megérné a processzorteljesítmény és a kódolási képesség ezzel járó növekedését.

Reménykedem abban is, hogy a memóriaforgalmazó vállalatok is magukénak érzik ezt az ötletet, és piacra dobják a hibás (olcsó) memóriamodulokat is. Örülnék, ha lenne modulok „silányságát” jellemző rendszer is, melyet a BadRAM rendszermag leírásában is közölhetnék.

Mindezek után nincs más hátra, mint hogy linuxos barátaimnak sok szerencsét kívánjak a hibás memóriák kereséséhez. Talán egy kevésbé érett operációs rendszer néhány vakbuzgó felhasználójától kapunk majd néhányat...

Rick van Rein
PhD hallgató a Twentei Egyetem informatikai szakán. Közelről figyeli a GNU közösség fejlesztéseit, ebből merít ösztönzést mindennapi munkájához. A BadRAM az ő ajándéka a GNU közösség számára.

Kapcsolódó címek

A BadRAM jelenleg nem a rendszermag része, de megtalálható a
➲ http://home.zonnet.nl/ és a
➲ vanrein/badram/ címen. *(1. kép)*

A BadRAM részletes alkalmazási példái, a rendszermagba fordítás után megtalálhatók a Documentation/badram.txt/ könyvtárban.

A rendszer mérési adatai
➲ http://home.zonnet.nl/vanrein/benchmarks.html.
A Memtest86 program megtalálható a
➲ http://reality.sgi.com/cbrady_denver/memtest86 címen. (*2. kép)*

A táblafoglaló leírása megtalálható *Jeff Bonwick* The Slab Allocator: An Object-Oriented Kernel Memory Allocator című írásában, mely az USENIX kiadásában jelent meg 1994-ben.

A WANPIPE belső működése

Szerzőink feltárják a WANPIPE meghajtóprogramok szerkezetét és a hozzákapcsolódó felhasználói felületet.

A Linux a kezdetek óta használatos a hálózati eszközök operációs rendszereként. Olyan eszközökre gondolunk itt, mint például hálózati cím fordítását végző eszközök (NAT), tűzfalak, virtuális magánhálózatok (VPN), levelezőrendszerek és webgyorstárak. Ebből következően értelemszerűen merült fel az igény arra, hogy Linux támogassa a kapcsolódást a nagy kiterjedésű hálózatokhoz (WAN).
A Sangoma 1994-ben kezdte kifejleszteni Linuxra a WANPIPE kódot és segédprogramokat, hogy felváltsák a mások által írott meghajtóprogramokat a Sangoma WAN-kártyáihoz.

Értelmes csatolókártyák

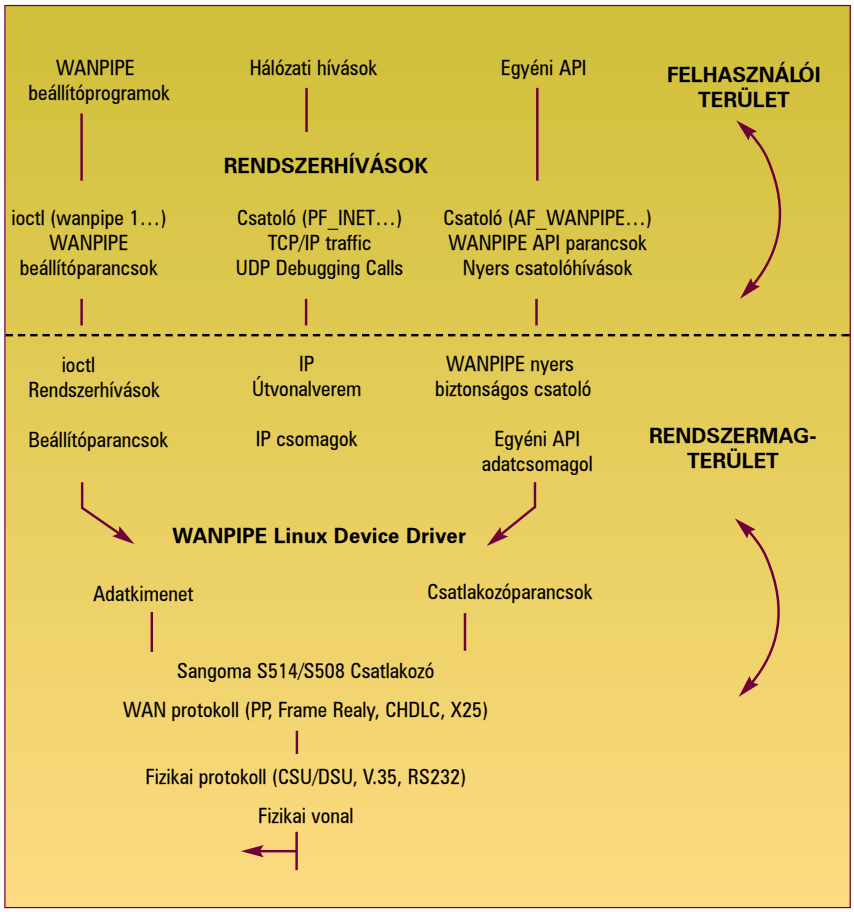
A WANPIPE a Sangoma S sorozatot támogatja. Ebben a sorozatban olyan értelmes csatolókártyák (pl. S514 PCI és S508 ISA) találhatók, amelyek gyári programja a legtöbb WAN protokollt támogatja, többek között a Frame Relay-t, a PPP-t, a HDLC-t, az X.25-öt és a BiSyncet. Mivel a protokollok támogatását a gyári programban valósították meg, az eszközmeghajtó programok sokkal egyszerűbb felépítésűek. Könnyebb őket más operációs rendszerre átültetni, hiszen kevesebb a hibalehetőség és a processzor terhelése is a lehető legkisebb. Ezeknek a tulajdonságoknak köszönhetően egy viszonylag lassú (például 486-os) gépből is lehet nagyteljesítményű T1 útválasztót készíteni egy Sangoma csatolókártya és a Linux segítségével.
Annak köszönhetően, hogy a protokollok a kártyában vannak, lehetővé válik a protokoll megvalósításának kipróbálása egy adott operációs rendszer alatt, tudva azt, hogy bármely más operációs rendszer alatt pontosan ugyanúgy fog működni. Szükség esetén a protokoll használat közben frissíthető, nem kell újrafordítani a meghajtóprogramot vagy a rendszermagot.
A Sangoma csatolókártyák két különböző felülettel csatlakozhatnak a külvilághoz, soros V.35/X.21/EIA530/RS232 csatlakozóval a T1 (CSU/DSU kártyán). A T1 csatlakozóval bíró kártya segítségével a kiszolgáló közvetlenül, külső CSU/DSU egység nélkül kapcsolódhat a T1 vonalhoz.

WANPIPE eszközmeghajtók

A Sangoma S514 és S508 kártyák legfeljebb négy nagy sebességű szinkronvonalat képesek kezelni, mindegyik vonalon egy többcsatornás WAN-protokollal. A meghajtóprogram felépítése a következő feltételekből következik:

- több csatolókártya támogatása, mindegyik csatolókártyán akár négy tényleges kapcsolattal
- mindegyik kapcsolatnak legfeljebb 255 logikai csatornája lehet
- minden kapcsolat külön vezérelhető és beállítható
- minden egyes logikai csatorna külön vezérelhető és beállítható
- több protokoll támogatása: Frame Relay, CHDLC, PPP, X25, SDLC stb.
- könnyen bővíthető (jövőbeli protokollokkal)
- útválasztó- és API alkalmazások egyidejű támogatása
- biztonságos illesztő az API alkalmazásokhoz (a csomagok csendes eldobása nem egedélyezett)
- helyi és távoli hibakeresés támogatása
- SNMP támogatása
- Gyors Tx és Rx útvonalak
- a proc fájlrendszer támogatása: kimutatások és hibakeresés
- meghajtóprogram üzeneteinek naplózása és eseménynaplózás
- könnyű frissíthetőség.

A következő tervezési szabályokat alkalmazták a meghajtóprogram kifejlesztésekor:



A WANPIPE meghajtóprogram kapcsolata a rendszermaggal

1. A WANPIPE meghajtóprogramok minden tevőleges fizikai kapcsolatot egy eszközre képeznek le a rendszermagban. Minden egyes fizikai vonal beállítható, újraindítható vagy ellenőrizhető anélkül, hogy a többi vonallal összeütközésbe kerülne. A WANPIPE meghajtóprogramok legfeljebb tizenhat eszközt támogatnak, négy kártyát, mindegyiken négy fizikai kapcsolattal.
2. Mivel a WAN protokollok sok logikai csatornával bírnak egyetlen fizikai kapcsolaton keresztül is, ezért minden csatorna egy hálózati csatolóra képeződik le. A WAN protokollok, mint a Frame Relay vagy az X.25, a logikai csatornák segítségével valósítják meg a pont–több pont kapcsolatokat. A WANPIPE minden fizikai vonalon 255 X.25 csatornát és 100 Frame Relay csatornát támogat. Minden egyes logikai csatorna újraindítható és újra beállítható anélkül, hogy le kellene állítani a többi csatornát ugyanazon a fizikai kapcsolaton.
3. A karbantartó/hibakereső felület hívásai az UDP-protokollon alapulnak és függetlenek az operációs rendszertől. A Sangoma egy egységes UDP-alapú felületet alkalmazott az adatgyűjtésre és a WAN-kapcsolatok karbantartására, ez kiválthatja a gyakran bonyolult és költséges SNMP-alapú segédeszközöket. A Sangoma szerint meg kell adni a lehetőséget a felhasználóknak, hogy könnyen beavatkozhassanak a rendszer működésébe távolról is, a WANPIPE csomagban található segédprogramok segítségével. Mivel a rendszer UDP-alapú és operációs rendszertől független, lehetséges például egy Windows alatt futó grafikus alkalmazásból távvezérelni egy Linux rendszert. A rendszer nem használ jelszavakat, de felkészíthető magas biztonsági követelményeket támasztó környezetben való működésre is.
4. Minden egyes hálózati csatoló API- vagy útválasztó módban működhet. Az IP-útválasztáson kívül a Sangoma számos vevője használja az S sorozatot saját alkalmazásai adatátviteli építőkockájaként, amelyhez a mi API-nkat használják. Ezek az alkalmazások a legkülönfélébb feladatokat látják el:

   - hírek és pénzügyi adatok egyirányú továbbítása műholdas kapcsolaton keresztül
   - mobilhálózati kapcsolók felügyelete X.25 használatával
   - IBM nagyszámítógépek vagy vezérlőkártyák utánzása SDLC-n, X.25-ön vagy BSC-n keresztül
   - katonai eszközök vezérlése HDLC LAPB-n keresztül.

A legnagyobb rugalmasság eléréséért a tervezés során gondoltak arra, hogy az API-hívások és a szabványos IP-útválasztó forgalom egy időben megférjen bármelyik fizikai kapun. Például néhány X.25 logikai csatorna a szabványos IP-kapcsolat fenntartására használható adott helyszínek között, miközben más csatornákat egy POS-terminálból származó hitelkártyaadatok továbbítására használunk, ez azonban nem IP-alapú. A meghajtóprogram képes API- és útválasztó csomagok egyidejű fogadására, a hálózati csatoló beállításainak függvényében.

5. Nem lehet az API-csomagokat eldobni a veremből. Az IP-vermek csendben eldobják azokat a csomagokat, amelyek már nem férnek a verembe. Ez a viselkedés elfogadott az IP világában, ahol az adatok sértetlenségét a magasabb szinten elhelyezkedő TCP-protokoll teremti meg. Az olyan hibajavító protokollok esetében azonban, mint a BSC, a HDLC LAPB, az X.25 és az SDLC, érvényes az a feltevés, hogy ha a csomagot visszaigazoljuk a kapcsolati szinten, akkor szavatoljuk az alkalmazás elérését. Ezért volt mindenképpen szükséges az, hogy a WANPIPE nyers illesztőverme ne dobja el csendben a csomagokat.
6. A WANPIPE eszközmeghajtókat modulként kell betölteni a Linux rendszermagjába. A modulok használatával, a rendszermag újrafordításával összevetve, könnyebb frissíteni a meghajtóprogramot. A számítógépet sem kell újraindítani a modulok lefordítása után.

1. táblázat **Hálózati csatolók beállítása és indítása**

Felhasználói térben futó parancsok	A rendszermag terében futó (meghajtóprogram) műveletek
Beállítási fájl létrehozása: 1. Gép típusa: PCI vagy ISA 2. Gépbeállítások 3. WAN-protokoll 4. Protokollbeállítások 5. A gyári program helye 6. Hálózati csatoló neve 7. IP-címek	Semmi
A WANPIPE meghajtóprogram moduljainak betöltése.	1. A /proc fájlrendszer könyvtárainak modprobe wanpipe.o beállítása 2. A csatolókártyákra mutató tömb lefoglalása: a mérete megegyezik a támogatott kártyák legnagyobb számával. Ebben a tömbben tárolódnak a csatolókártyák beállításai. 3. Az ioctl rendszerhívások engedélyezése
A beállítási fájl értelmezése és a beállítási adatok elküldése a meghajtónak IOCTL híváson keresztül.	1. Csatolókártya beállítása 2. Erőforrások lefoglalása 3. Megfelelő protokollprogram betöltése 4. A gyári program indítása a kártyán 5. Megszakítások ellenőrzése
Az IP-adatok és protokollok beolvasása a beállítási fájlból minden megadott csatolón futó protokollhoz, és ezen adatok átadása a meghajtónak ioctl rendszerhíváson keresztül.	1. Hálózati csatoló beállítása 2. Hálózati csatolófelület lefoglalása (Eszközszerkezet) 3. Eszközszerkezet előkészítése és bejegyzése 4. Az eszköz magánterületének lefoglalása és beállítása
Az ifconfig() használata, minden egyes hálózati csatoló életre keltése, és helyi valamint P2P IP-címeinek beállítása, a csatoló alapértelmezett átjáróként való bejegyzése, ha ez a lehetőség be volt állítva.	1. A meghajtóprogram meghívja a dev->open() függvényt, amely engedélyezi a adatcserét és a megszakításokat a csatolókártyán

2. táblázat A rendszermag IP-útvonaltáblája

Cél	Átjáró	Maszk	Jelzőbitek	Mérték	Hiv.	Használat	Csatoló
201.1.1.2	0.0.0.0	255.255.255.255	UH	0	0	0	wp1_fr16
192.168.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
0.0.0.0	0.0.0.0	0.0.0.0	U	0	0	0	wp1_fr16

3. táblázat Figyelőprogramok és jellemző parancsok

Hibakereső neve	Protokoll	Használat: modem állapotának ellenőrzése
Fpipemon	Frame Relay	fpipemon -i wp1_fr16 -u 9000 -c xm
cpipemon	Cisco HDLC	cpipemon -i wp1_chdlc -u 9000 -c xm
Ppipemon	PPP	ppipemon -i wp1_ppp -u 9000 -c xm
Xpipemon	X25	xpipemon -i wp1_svc -u 9000 -c xm

Ahol: ?pipemon -i <hálózati csatoló neve || IP-cím>
-u <a beállítási fájlban megadott UDP-kapu>
-c <parancs>

A fenti tervezési szabályok alkalmazásával a WANPIPE eszközmeghajtók a legtöbbet hozzák ki az S sorozatú kártyákból. A világosan meghatározott adat, hibakereső és (újra) beállító útvonalak lehetővé teszik a hatékony, változtatható és karbantartható működést. A meghajtóprogram felépítése az *ábrán* látható.

A WANPIPE és a Linux rendszermagja

Az ábrán bemutatjuk, hogyan illeszkedik a WANPIPE eszközmeghajtó a Linux rendszermagjához. A Linux két végrehajtási szinten működik, felhasználói módban és rendszermag módban. Minden alkalmazás, démon és segédprogram felhasználói módban fut, míg a rendszermag és az eszközmeghajtók rendszermag módban futnak. A felhasználói módban futó alkalmazások rendszerhívásokon (pl.: ioctl) keresztül tarthatnak kapcsolatot a rendszermaggal.
Az eszközmeghajtók a rendszermag részei, ezek teremtik meg a kapcsolatot az alkatrészek és az operációs rendszer között. A meghajtóprogramokat általában belefordítják a rendszermagba, vagy különálló modulok formájában készítik el, amelyek futásidőben szükség esetén betölthetők vagy eltávolíthatók.
A Sangoma az elemes felépítést választotta a WANPIPE esetében, mert a modulokat könnyű frissíteni, és újra betölteni a rendszermag futása közben, így szükségtelen a költséges újraindítás.
A WANPIPE hálózati eszközmeghajtó, tehát hálózati csatolófelületet használ a Linux rendszermagverméhez való kapcsolódáshoz. A hálózati csatolók 3. szintű protokolladatokat (IP) és a meghajtó belépési pontjait tartalmazzák, így a rendszermag vermét használva (a hálózati csatolón keresztül) vezérelhető a meghajtóprogram működése: csatoló leállítása, elindítása, kimutatások készítése és adatátvitel.

A WANPIPE beállításai

A WANPIPE beállítási folyamata egy részletes beállítási fájl elkészítésével kezdődik, amely megadja a gép-, a protokoll- és az IP-beállításokat, valamint a csatolókártya gyári programjának a helyét.
Amikor kész, a WANPIPE meghajtóprogram moduljai betöltődnek a rendszermagba. A modulbetöltés folyamata lefoglalja a szükséges erőforrásokat, létrehozza és beállítja a proc fájlrendszeren a rendszerkönyvtárakat, és engedélyezi az ioctl rendszerhívásokat. Mivel a betöltött moduloknak nincs elég adatuk ahhoz, hogy teljesen beállítsák a kártyát, ioctl rendszerhívások használatával kell a beállítási fájl tartalmát a meghajtóprogram tudomására hozni. A WANPIPE beállításának végső lépése a hálózati csatolók felélesztése az ifconfig() parancs segítségével. A folyamatot az *1. táblázat* mutatja be.

A WANPIPE és az útválasztás

A rendszermag IP-rétege felelős a csomagok továbbításáért, azaz eljuttatja a címzéssel ellátott csomagokat azok rendeltetési helyére. Az IP-réteggel együttműködve, az útvonaltábla (lásd a *2. táblázatot*) határozza meg minden bejövő csomag továbbítási rendjét.
Ha a WANPIPE hálózati csatoló (wp1_fr16) működik, a rendszermag a csatoló IP-adataival frissíti saját útvonaltábláját.
A wp1_fr16 csatolóhoz két bejegyzés tartozik. Az első megadja a célhálózatot, a második bejegyzi alapértelmezett útvonalként; ez azt jelenti, hogy minden olyan IP-cím, amely nincs külön említve feljebb az útválasztó táblázatban, a wp1_fr16 csatolón keresztül megy tovább.
A meghajtóprogram sikeres beállítása után a hálózati csatolók és az útválasztó táblázatok a szokásos parancsokkal nézhetők meg és módosíthatók:

- ifconfig – hálózati csatolók megjelenítése vagy módosítása
- route – útvonaltábla megjelenítése vagy módosítása.

A WANPIPE és az API-k

Egyéni RAW (nyers), nem IP-csomagok küldése és fogadása a kártyára és a kártyáról egy API segítségével valósítható meg. Mivel az adatok nem IP formátumban érhetők el, ezért a hálózati csatolónak nincs IP-címe. Ez eltávolítja a rendszermag útvonaltáblájából a bejegyzést és leválasztja az IP-útválasztó vermet a WANPIPE meghajtóprogramról. A nem IP adatcsere a meghajtóprogram RAW illesztőfüggvényének használatával érhető el. Ahogy a névből is látszik, a csomagok minden módosítás nélkül mennek át.
Azért, hogy a kártya szintjén visszaigazolt csomagok ne veszhessenek el soha, egy biztonságos megoldást dolgoztunk ki: egy egyéni WANPIPE illesztő szavatolja mindkét irányban a kézbesítést.
A WANPIPE illesztő az Alan Cox és társai által kidolgozott RAW illesztőn alapul.

Fejlesztés a biztonságos WANPIPE-pal

A következőkben példát adunk a WANPIPE API-készletének használatára. Az X.25-re esett a választásunk, mert talán ez a legbonyolultabb vonalprotokoll, hívásbeállítással és -megszakítással, logikai csatornakezeléssel és kivételkezeléssel. Az X.25 csomagkapcsolt WAN-protokoll, amely (általában) nyilvános csomagkapcsolt hálózatot használ a csomagok célba juttatására. A gyakorlatban ez hasonlít a TCP/IP-re, de ez elvileg teljesen más. A vonali sebesség majdnem mindig 256 kb/mp alatt van, általában 64 kb/mp alatt. Az X.25 működése a telefon működésére hasonlít. A hívást kezdeményezni kell, és ha a hívást fogadták, kezdődhet az adatátvitel. Az adatátvitel befejeztével a hívást bontani kell. A biztonságos WANPIPE illesztő segítségével az X.25 API programozása nagyon hasonlít a TCP/IP programozására.

A példa folytatásaként feltesszük, hogy a WANPIPE meghajtóprogram beállítása és elindítása sikerült, és az X.25 kapcsolat él. (Lásd az 1. és 2. listát a következő címen: ➲ ftp://ftp.ssc.com/pub/lj/listings/issue82).

WANPIPE hibakeresés

A WAN-ok természetükből adódóan meglehetősen bonyolultak. Általában több szereplő vesz részt, egy vagy több távközlési vállalat, gyakran egy nyilvános hálózatszolgáltató, sokszor egy különálló internetszolgáltató, hogy még nagyobb legyen a zűrzavar. Az elkerülhetetlen kezdeti gondok és a folyamatos hibakeresés gyakran kilátástalan egymásra mutogatásba torkollik.

Emiatt a WANPIPE fejlesztésekor a legnagyobb figyelmet a hibakeresésre fordítottuk. A Sangoma nézete az, hogy a vevőinek elég hibakeresési adatot kell adnia ahhoz, hogy maguk is megtalálhassák a hiba kiküszöbölésének módját. Ezen túlmenően, ha segítség kell, a Sangoma műszaki támogatásért felelő szakembereinek is elég adattal kell bízniuk ahhoz, hogy a bajt távolról megoldják.

Az xPipemon programok

Minden WAN-protokollhoz tartozik egy hibakereső segédprogram, amelyet arra használhatunk, hogy meghatározzuk a meghajtóprogram és a fizikai vonal állapotát, megtudjuk a protokoll állapotát és kimutatásait, valamint a nyers és értelmezett vonalvizsgálati eredményeket. A megfigyelőprogramokban alkalmazott adatátvitel UDP-alapú. Távoli rendszerek hibakeresése elvégezhető az Interneten keresztül, anélkül hogy be kellene jelentkezni a felhasználó gépére, miközben a rendszerbiztonság szigorúan szabályozható. Az UDP-hívások operációs rendszertől függetlenek, ez azt jelenti, hogy egy távoli linuxos gépről elvégezhető egy FreeBSD vagy Windows gépben futó WANPIPE kártya hibakeresése is.

Mivel a rendszerbiztonság fontos kérdés, az UDP hibakereső parancsok letilthatók az UDPPORT 0-ra állításával, vagy ami még jobb, kicsi TTL-érték választásával. Például, ha a TTL értéket 1-re állítjuk, csak a gépre bejelentkezett felhasználók vagy az első útválasztó előtt elhelyezkedők tudják a hibakeresőt működtetni. A TTL és UDPPORT értékeket a WANPIPE beállítási fájlban lehet megadni.

A figyelőprogramok és jellemző parancsok friss listája a *3. táblázatban* olvasható. Windows, X és más grafikus környezetben a bonyolult parancssorból vezérelhető programokat egyszerű grafikus alkalmazások váltják fel.

A meghajtó a karbantartási kérelmet az UDP/IP-vermen keresztül veszi fel. Minden beérkezett kérelem egy alacsony fontosságú sorba kerül. Egy alacsony fontosságú szál kezeli a kérelmeket és az eredményt visszaküldi a verem tetejére, a kezdeményező IP-címére. UDP hibakereső kérések a hálózatról is jöhetnek, ekkor a kérés a vonalon keresztül visszamegy. A hálózati csatolón keresztül érkezett karbantartási kapcsolatokat a rendszer másként kezeli, mint a „felülről” jövő forgalmat. A hálózaton keresztül csak a kimutatások kérhetők le, míg felülről a felhasználónak lehetősége van újra beállítani, kipróbálni és elindítani a CSU/DSU egységet és vonalvizsgálatot futtatni.

A proc fájlrendszer

A proc fájlrendszer egy memóriába leképezett virtuális könyvtárszerkezet, amelyből a rendszermagról és a meghajtóprogramokról olvashatunk ki adatokat. A rendszerfelügyelő programok (pl.: SNMP) a proc fájlrendszerből szedik a rendszermag, illetve meghajtóprogram kimutatásait és állapotait. A WANPIPE meghajtó úgy csatolódik a proc fájlrendszerhez, hogy létrehozza a /proc/net/wanrouter könyvtárat. Ez a könyvtár minden WANPIPE eszközhöz tartalmaz egy virtuális fájlt. A WANPIPE beállításai és kimutatásai a megfelelő /proc fájlok olvasásával kaphatók meg. A wanpipe1 eszközre vonatkozó tx, rx és hibakimutatásokat így tudhatjuk meg:

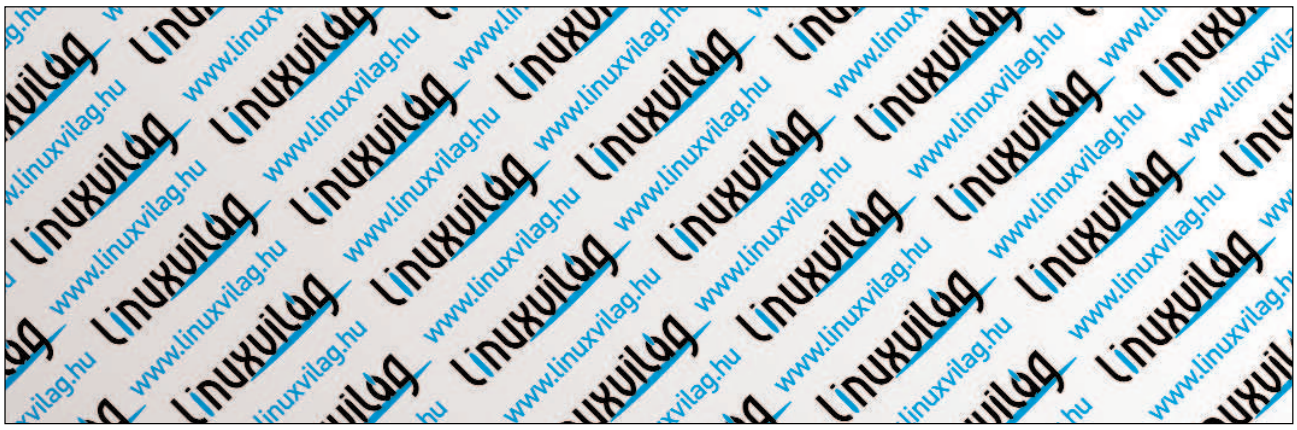
```
cat /proc/net/wanrouter/wanpipe1
```

Naplóbejegyzések

Minden WANPIPE eseményt a syslog démon jegyez fel a /var/log/messages fájlba. Megjegyzendő, hogy a syslog átállítható, hogy máshová naplózza az üzeneteket. Ha folyamatosan figyelemmel akarjuk kísérni az üzeneteket, adjuk ki a `tail -f /var/log/messages` parancsot.

Nenad Corbic
adatátviteli szakember. A Sangoma linuxos fejlesztői csoportjának vezetőjeként azon munkálkodik, hogy a Sangoma linuxos vásárlói hozzáférjenek a legjobb nagy kiterjedésű hálózati (WAN) adatátviteli megoldásokhoz. Nenad felelős a WANPIPE eszközmeghajtó tervezéséért és fejlesztéséért, a WANPIPE minőségbiztosításáért, új termékek fejlesztéséért és harmadik szintű vevő- és fejlesztőtámogatásért. Ezenkívül részt vesz a Linux útválasztó projektben és a beágyazott Linux fejlesztésében.

David Mandelstam
műszaki igazgatóként tevékenykedik. A Sangoma növekedésének motorja a kezdetektől fogva, elkötelezett fejlesztője a WAN adatátviteli megoldásoknak. David a Sangoma egyik alapítója, ő vezényli a műszaki fejlesztéseket és a vállalati kutatási tevékenységet, irányítja az új termékek kifejlesztését és áttekinti az egész gyártási folyamatot.



A telefonos rendszermag-API

Greg Herlein elmeséli, hogyan épül be a telefoneszközök meghajtóprogramja a Linux rendszermagjába.

Ritkaságszámba ment, még egy évvel ezelőtt is, az internetes telefonálás. Sokan gondolták, hogy nem is lesz ez jó soha igazi telefonhívások lebonyolítására. Ma, köszönhetően a Net2Phone, a Deltathree.com és a DialPad szolgáltatókhoz hasonlóaknak, amelyek Interneten keresztül bonyolított ingyenes vagy nagyon olcsó telefonhívásokat kínálnak, az IP-n keresztüli hangátvitel (VoIP) már majdnem húzóágazatnak számít. Bár ezekhez a szolgáltatásokhoz még nincsenek linuxos ügyfélprogramok, a Linux nincs lemaradva. A 2.2.14-es rendszermaggal a Linux nagy előnyt szerzett a számítógépes telefonálás területén: ez az első korszerű operációs rendszer, amely a rendszermag szintjén ad programozói felületet a telefonos alkalmazásokhoz. Ráadásul kiváló minőségű nyílt forráskódú programok már használják is ezt az API-t. Körbetelefonálhatjuk a világot a Linux és az Internet használatával – ingyen.

Ebben a cikkben bemutatjuk, hogyan építették be a telefoneszközök meghajtóprogramjait a rendszermagba oly módon, hogy egységes programozói felületet nyújtsanak a különféle típusokhoz. Ezután megtárgyaljuk az API tervezési és működési alapelveit. Mi módon választhatók szét az adatokra és az eseményekre vonatkozó adatok. Végül tárgyaljuk a telefonálás elemeinek (pl.: csengetés vagy a kézibeszélő felemelése) kezelését, ezt az úgynevezett „aszinkron eseményértesítő” folyamat végzi.

Új lehetőség a rendszermagban: a telefonálás támogatása

Sok ember kérdezte, miért kell egy új telefonálást támogató API-t a rendszermagba építeni. Még *Alan Coxot* is meg kellett győzni! Hiszen a legtöbb internetes telefonálásra alkalmas program képes kezelni a hangkártyát, és ha a hangkártya támogatja a teljes kétirányú (full-duplex) üzemmódot, és a felhasználónak van egy jó minőségű mikrofonos fejhallgatója, akkor a hívás minősége megfelelő lehet. Miért bonyolítsuk a dolgokat egy új API-val?

A válasz elég egyszerű: a hangkártyák nem telefonkészülékek! A hangkártyák nem tudnak vonalhangot adni, csörögni, foglaltat jelezni, vonalat bontani vagy hívó felet azonosítani, pedig ezek mindegyike szükséges a telefonkészülék rendes működéséhez. Egy hangkártya valóban használható némi korlátozással telefonálásra, ha egy mikrofonos fejhallgatót csatlakoztatunk hozzá. Az igazi telefonkártyákhoz azonban zsinór nélküli telefon is köthető, és máris megszabadulunk számítógéphez kötöttségünktől, amit a rövid zsinór okozott. Bonyolult üzleti telefonrendszerekhez is csatlakozhatunk. Ezenkívül a hangkártyákat nem lehet a helyi telefontársaság által felszerelt falicsatlakozóba bedugni, ezért nem tudjuk befolyásolni a kimenő és bejövő hívásokat, nem használhatjuk a díjszabáscsökkentő alkalmazásokat, vagy más, napjainkban íródó új nemzedékbeli telefonos alkalmazásokat.

Minden komoly Interneten keresztüli VoIP-megoldásnak tömöríteni kell a hangadatokat. Azért, hogy a megoldás megfeleltethető legyen a többi VoIP-alkalmazással és eszközzel, támogatnia kell a széles körben elfogadott tömörítőkodekeket, mint például a G.723.1 vagy a G.729a. Sajnos a programbeli megvalósítás esetén ki kell fizetni a kodekek felhasználási szerződésében kiszabott díjat, ez pedig akár hat számjegyű is lehet (dollárban). Ezek a könyvtárak nem lesznek nyílt forráskódúak, az biztos. A telefonkártyák (mint a lent említett Quicknet kártya) a gépünkbe beépítve tartalmazzák ezeket a kodekeket, ugyanis a felhasználási szerződésben kikötött díjat a kártya gyártója már rendezte. Ezzel elkerülhető a példányonkénti díj fizetése körüli hercehurca, hiszen a díj a kártya árába be van építve, és tetszés szerinti felhasználási szerződés alkalmazható a termékre, nem pedig csak az, mit a kodek szerzője előírt. Más szavakkal fogalmazva, írhatunk nyílt forráskódú VoIP-alkalmazást, és mégis használhatjuk a fejlett kodekeket. Nem túl gyakoriak azonban az olyan hangkártyák, melyek ismerik ezeket a tömörítőket.

1. lista A phone_device adatszerkezet

```
struct phone_device {
     struct phone_device *next;
     struct file_operations *f_op;
     int (*open) (struct phone_device *, struct
file *);
     struct file *file_p;
     int board;
     int minor;
};
```

Ráadásul a hangkártyák nem csatolhatók telefonokhoz vagy telefonrendszerekhez, és nem támogatják a kártyás hangtömörítőket. Az igazsághoz hozzátartozik, hogy a telefonkártyák viszont nem hangkártyák. A hangkártyák hangkeltő képességei magasan felülmúlják a telefonkártyákéit. Például a hangkártyák sztereók, a telefonkártyák monók. A hangkártyák képesek felvenni és lejátszani a zene tartományába eső frekvenciákat (20 Hz–20 kHz), a telefonkártyák azonban a beszédhangok frekvenciatartományára (általában 300 Hz–4 kHz) lettek korlátozva. A hangkártyák képesek bonyolult zenei hang keltésére, gyakran támogatják a MIDI-szabványt, és sokféle hanghatás létrehozható velük. A telefonkártyák mindezeket nem tudják. A felépítésük és a tulajdonságaik teljesen különbözők. Ebből következően az eszközmeghajtóiknak és az API-knak is különbözőnek kell lenniük. De várjunk csak! – mondhatnánk. – Mi a helyzet azokkal a nagyszerű programokkal, amelyek hangkártyával működnek, mint például a hangfelismerők vagy szövegfelolvasók? Nem lenne-e jó ezeket a telefoneszközön is használni csekély ráfordítással? Mindez lehetséges. A Linux telefonos API-ját úgy tervezték, hogy ne zárja ki eleve az olyan programok használatát a telefonkártyán, amelyeket eredetileg hangkártyához terveztek. A programkódot természetesen módosítani kell egy kicsit, de ez nem túl nehéz feladat. Erről több szó esik majd a cikk vége felé.

Az új API létrehozásának serkentésében a Quicknet Technologies Inc. járt az élen, mely többféle telefonkártyát is gyárt. Tavaly novemberben *Ed Okerson* és én (ekkor még a Quicknet alkalmazottja voltam) elküldtük a mai API ősét Alan Coxnak. Hetekig tartó megfeszített levelezés és számtalan programsor megírása után megszületett a Linux telefonos API-ja.

Vágjunk bele, lássuk, hogyan működik!

Az alapok: telefoneszközök

Az operációs rendszer szintjén minden eszköz egy szám. Ha belenézünk a /dev könyvtárba, sok fájlnevet láthatunk, de mélyen lent a Linux az eszközöket a fő- és alszámaik alapján azonosítja. Bizonyos típusú eszközök összessége egy közös főszámmal rendelkezik, az egyes eszközpéldányoknak ezen a típuson belül saját alszámmal kell bírniuk. Például ha kiadjuk az ls -al /dev/ttyS0 parancsot, a következőket láthatjuk:

```
gherlein@tux:~/lj > ls -al /dev/ttyS0
crwxrwxr-x   1 root      uucp        4,   64 Oct
       ➥27 06:23 /dev/ttyS0
```

Figyeljük meg, hogy a fájl jogosultságait leíró maszkban az első karakter egy "c", ez azt jelzi, hogy karakteres eszközről van szó. Az eszköz tulajdonosa a rendszergazda, és az uucp csoportban van. A következő két szám nem a fájl mérete, ahogy közönséges fájlok esetében lenne, hanem a fő- és alszám. A ttyS0 esetében a főszám 4 és az alszám 64.

A Linux telefonos API a telefonszerű eszközökhöz a 100-as főszámot rendeli hozzá. Elképzelhető, hogy az általunk használt Linux-változat nem hozza létre ezeket az eszközöket, ellentétben a /dev/ttyS*, a /dev/audio és más régebbi, széles körben elfogadott eszközleképezésekkel. H a /dev/phone* eszközök nincsenek meg a rendszerünkben, létre kell hoznunk őket a Linux telefonos API használata előtt. Könnyen megtehető ez saját kezűleg is a következő parancs kiadásával (rendszergazdaként):

```
mknod /dev/phone0 c 100 0
```

Ezzel egy új eszközfájlt hoztunk létre, amelynek a neve /dev/phone0. Ez egy karakteres eszköz 100-as főszámmal és 0-s alszámmal. Az mknod parancs részletes megismeréséhez olvassuk el a leírását. Gyakorlatilag csak annyi eszközfájlra van szükségünk, ahány fizikai eszköz csatlakozik a számítógéphez, de egyes emberek egyből létrehozzák az összes eszközfájlt 0-tól 15-ig.

Figyeljünk arra, hogy a /usr/src/linux/Documentation/devices.txt pillanatnyilag egy hibás kijelentést tartalmaz. Ebben a fájlban van felsorolva a főszámok hivatalos kiosztása, és jelenleg az szerepel itt, hogy a Linux a 159-et használja a telefoneszköz főszámaként, és a 100 már nem helyes. Ez egy elismert hiba, amit kijavítanak a közeljövőben. A helyes főszám a linuxos telefoneszközre (és a rendszermag által használt szám) a 100.

A phonedev modul – eszközközvetítés

Alan Cox fejlesztette ki a phonedev modult. A fejlesztés során hasonló megközelítéssel élt, mint amit a Video4Linux projekt során követett. Sok gyártó fog olyan terméket előállítani, ami telefoneszközként is használható. Ahelyett, hogy minden egyes telefoneszköz gyártója egy saját főszámot foglalna le, mindenki használhatja a 100 főszámot és a /dev/phoneN eszközfájlokat (ahol N az eszköz száma). Mindenkinek egy egységes, egyszerű programozói felületet kell nyújtania a felhasználói térben futó alkalmazások számára, azaz mindenkinek a közös API-t kell követnie, bár mindenki nyújthat kiegészítő szolgáltatásokat a termékéhez. A gyártók fogják megírni a saját eszközmeghajtó moduljaikat, amelyek megvalósítják a közös API-t és egy külső csatlakozási felületet az adott alkatrész belső részleteihez. A phonedev modul megoldotta azt a feladatot, hogy az aleszközszám futási időben képződjön le egy kártyagyártó által szállított modulra. A forráskód a /usr/src/linux/drivers/telephony/phonedev.c fájlban van, a phonedev.h és a telephony.h fejlécfájlokat pedig a /usr/include/linux könyvtárban találjuk. Lássuk, hogyan működik!

Minden telefoneszköznek egy phone_device adatszerkezetet kell használnia (lásd *1. lista*). Hasonlóképpen, minden telefoneszköznek meg kell hívnia két függvényt a phonedev modulban, hogy bejegyezze és törölje magát. Ezeket így adják meg:

```
extern int phone_register_device
     (struct phone_device *, int unit);
extern void phone_unregister_device
     (struct phone_device *);
```

Betöltéskor a phonedev modul beállítja magát és várja, hogy más telefoneszközöket kiszolgálhasson. Amikor egy telefoneszköz meghajtóprogramja betöltődik (a később tárgyalt modprobe-bal vagy insmoddal), meghívja a phone_register_device függvényt. Ennek leegyszerűsített működése: egy mutatót tart fenn a phonedev modulban, amely a phone_device adatszerkezetre mutat, megkeresi az első megnyitott alszámot és hozzárendeli azt a telefoneszközhöz, azután megnövel egy számlálót, amely azt követi nyomon, hogy valami használja a phonedev modult (nehogy használat közben törlődjön).

A gyakorlatban ez az alábbi következményeket vonja maga után: az először betöltődő telefonos modulhoz rendelődik hozzá az első elérhető (a legkisebb) alszám. Rendkívül fontos ezt megérteni abban az esetben, ha különböző gyártóktól származó moduloknak kell együtt élniük egy rendszerben, és kívánatos, hogy bizonyos eszköz egy meghatározott alszámon legyen elérhető (azaz egy adott /dev/phoneN eszközön keresztül). Más szavakkal, ha van egy ABC kártyánk és egy XYZ kártyánk, és azt szeretnénk, hogy az ABC legyen a /dev/phone0, akkor meg kell bizonyosodnunk róla, hogy az ABC meghajtóprogramja töltődik be először.

Minden eszköznek legalább az alapvető műveleteket értelmeznie kell, amelyekkel az eszközzel kapcsolatba léphetünk. Ilyen műveletek a megnyitás, az olvasás, az írás, a lezárás stb. Ezek mind a „fájlműveletek adatszerkezet” részei (lásd a linux/fs.h fejlécfájlt a részletekért). Minden eszköz úgy adja meg ezeket a függvényeket, ahogy számára megfelelő.

Ha egy program megnyitja a /dev/phoneN eszközt, akkor a phonedev modul fájlműveletek adatszerkezetében szereplő fopen függvényhívásra kerül a vezérlés. Ez a függvény a következő műveleteket hajtja végre:

- Megszerzi az aleszközszámot a /dev/phoneN fájlleíróból (inode).
- Összeállít egy karakterláncot a fő- és alszámok felhasználásával, ennek formátuma char-major-%d-%d. Például ha az alszám 0 (/dev/phone0 esetén), a karakterlánc char-major-100-0 lesz.
- Ezt a karakterláncot felhasználja a request_mmodule meghívásánál, mely egy modul betöltésére irányuló kérés. Ennek ugyanaz a hatása, mint a modprobe programnak (valójában egyszerűen elindít egy külön rendszermagszálat és ebben egy modprobe-ot). Ezzel lehetővé teszi, hogy – ha az eszköz egy modul és nem a rendszermag része – a kmod betölthesse a modult, mielőtt a phonedev használni próbálná.
- Ezután meghívja a telefoneszköz moduljának fopen függvényét, amely ténylegesen megnyitja a megfelelő eszközt.

Ahogy látható, a phonedev modulnak kettős szerepe van. Egyrészt az alszámokat betöltéskor dinamikusan hozzárendeli a telefoneszközökhöz, másrészt lehetővé teszi a telefoneszközök moduljainak betöltését futási időben. Ez világos, mégis jól kell ismerni a modprobe-ot és a modulok függőségi viszonyait ahhoz, hogy teljesen megértsük a telefonos modulokat és kölcsönhatásaikat.

A modprobe és a telefoneszközök

Ha a kmodot használjuk a szükséges rendszermagmodulok betöltéséhez, akkor létre kell hoznunk egy csomó másodnevet a /etc/modules.conf fájlban. Először is a rendszernek tudnia kell, hogy a char-major-100 a phonedev modul. Adjuk hozzá ezt a sort a fájlhoz:

```
alias char-major-100   phonedev
```

Ezután a rendszerrel közölni kell, hogy milyen telefoneszköz-modulok tartoznak az egyes alszámokhoz. Ahogy fent megtudhattunk, ez nem feltétlenül biztosítja azt, hogy a phonedev modul pont ezt az alszámot osztja ki a telefoneszköz-modulnak a betöltéskor. A következő példákban a Quicknet Technologies Inc. ➲ www.quicknet.net telefonkártyáit fogjuk használni. Ezekhez a kártyákhoz van meghajtóprogram a Linux rendszermagban, és viszonylag olcsóbbak, mint más telefonkártyák. A Quicknet eszközmeghajtója az ixj.o, és a modul neve ixj. Ez a meghajtóprogram az összes telefonkártyájukhoz használható (elég okos ahhoz, hogy kezelje az ISA, a PCI vagy a PCMCIA változatot és ismerje az egyes kártyatípusokon használt telefoncsatoló-áramköröket). Ha azt szeretnénk, hogy a Quicknet meghajtóprogramja a /dev/phone0 eszközhöz rendelődjön, akkor adjuk ezt a sort a /etc/modules.conf fájlhoz:

```
alias char-major-100-0  ixj
```

Ahogy emlékezhetünk még a phonedev fopen függvényének elemzéséből, a phonedev egy char-major-%d-%d formájú karakterláncot hoz létre, és a számok helyére a 100-at (főszám) és a kért alszámot helyettesíti be. Példánkban a /dev/phone0 megnyitási kísérlete azt eredményezi, hogy a phonedev megpróbálja betölteni a char-major-100-0 eszközt. Ez az eszköz ismeretlen a rendszermag modulbetöltője számára. A fenti alias parancs ezt a karakterláncot az ixj névre képezi le. Amikor megpróbáljuk megnyitni a /dev/phone0 eszközt, a phonedev modul automatikusan betölti az ixj modult, azután meghívja az ixj modul fopen függvényét, ez megnyitja az eszközt. (Természetesen feltettük, hogy a rendszermag támogatja a modulok betöltését, azaz `CONFIG_KMOD=y`).

Az egyszerű telefoneszköz-API

A Linux telefonos API-jának alapvető vívmánya, hogy telefoneszközre és a telefoneszközről csak hangadat írható, illetve olvasható az írás és az olvasás műveletek segítségével. Ez teljesen más – és másképpen is kezeli a meghajtóprogram –, mint az eseménytípusú adatok.

Hangadat

Mi is pontosan a hangadat, és miben különbözik az események adataitól? A hangadat a telefoneszköz mikrofonjáról származó hangjel analóg–digitális (A/D) átalakításának (és esetleg adattömörítésének) az eredménye. A telefon kézibeszélőjének felvételekor keletkező jel egy esemény. A sípszó, amelyet a készülék egy számbillentyűjének lenyomása idéz elő, szintén esemény, annak ellenére, hogy hangkibocsátással jár. Egy bejövő hívás által előidézett csengetés is esemény. Röviden minden, nem mikrofonon keresztüli bemenet esemény. Minden mikrofonon keresztül érkező bemenet (és a hangszórón megjelenő kimenet) hangadat. Ezt a hangadatot olvassuk és írjuk a telefoneszközről, illetve telefoneszközre a szabványos read és write rendszerhívásokkal.

A legtöbb telefoneszköz tömöríti a hangadatokat. Valójában az adattömörítés nélkül nem képzelhető el egy sikeres internetes telefonalkalmazás. Ezeket a tömörítési módszereket hívják „hangkodekeknek” vagy röviden „kodekeknek”. Vannak széles körben elterjedt és használt kodekek. A Linux telefonos API-ja tartalmaz állandókat ezekhez a kodekekhez, de egy adott telefoneszköz nem feltétlenül támogatja ezek mindegyikét. Egy ioctl rendszerhívással kérdezhető le, hogy milyen kodekek vannak használatban.

Az egyik nagy különbség a linuxos telefonos API és a hangkártyákhoz használatos API-k között az, hogy a linuxos telefonos API „adatkocka-központú”, míg a hangkártyák „bájtközpontúak”. Az adatkocka-központú eszköz időegység alatt egy meghatározott méretű adatkocka olvas be. Ez azért van így, mert minden hangkodek adott ideig dolgoz fel egy bizonyos ideig tartó hangadatot (általában 10, 20 vagy 30 ezredmásodpercnyi adatot). Mivel a tömörítés alkalmazása a bevett szokás a hálózati telefonos alkalmazásoknál, ezért ez a helyes és elvárt működése a telefoneszközöknek. A hangkártyáknál nincs ilyen megkötés, bármelyik megszólítás alkalmával tetszőleges számú bájtot írhatunk vagy olvashatunk. Az API meghatározza minden kodekhez az „adatkocka méretét”, és a nyers tömörítetlen hangadat is egy lehetséges kodekválasztásnak felel meg. Például a LINEAR16

2. lista A telefon csörgetése

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/telephony.h>
#include <linux/ixjuser.h>


int
main(int argc, char *argv[])
{
  int ixj=-1;
  char pname[80], maxrings;

  if(argc >= 2)
    sprintf(pname, "%s", argv[1]);
  else
    sprintf(pname, "/dev/phone0");

  printf("%s megnyitása\n",pname);
  ixj = open(pname, O_RDWR);

  if(ixj<1)
  {
    printf("Eszközmegnyitási hiba:
%s\n",pname);
    perror("open ");
    exit(-1);
  }

  if(argc >= 3)
    maxrings = atoi(argv[2]);
  else
    maxrings = 2;

  ioctl(ixj, PHONE_MAXRINGS, maxrings);

  if(!ioctl(ixj, PHONE_RING))
  {
    printf("Nem veszik fel.\n");
  }
  else
  {
  printf("Felvették a telefont.\n");
  }
  close(ixj);
}
```

3. lista A kivétel-adatszerkezet használata

```
void
getdata(int ixj1)
{
   fd_set              rfds,wfds,efds;
   struct timeval      tv;
   union telephony_exception ixje1;
   int                 nmax, size, state, nlen;
   char                buf[480], jbuf[480],
                       dtmf1;
   char                date[5], time[5], clen[2],
                       pnum[11], cname[80];

   nmax = ixj1+1;

   /* fájlleírók törlése */
   FD_ZERO(&rfds);
   FD_ZERO(&wfds);
   FD_ZERO(&efds);

   /* mindegyik beállítása az ixj1-re */
   FD_SET(ixj1, &rfds);
   FD_SET(ixj1, &wfds);
   FD_SET(ixj1, &efds);

   /* a select időtartamát állítsuk kicsire */
   tv.tv_sec = 0;
   tv.tv_usec = 300;

   /* várjuk az idő leteltére, vagy
      vagy eseményre a fájlleíróban */
   select(nmax, NULL, &wfds, &efds, &tv);

   if(FD_ISSET(ixj1,&wfds))
   {
      /* A fájlleíró írásra kész
         - küldjünk adatokat! */
   }

   if(FD_ISSET(ixj1,&rfds))
   {
      /* A fájlleíró olvasásra kész
         - olvassuk be az adatokat! */
   }

   if(FD_ISSET(ixj1,&efds))
   {
     /* megkérdezzük az eszközt,
         hogy milyen kivétel történt */
     ixje1.bytes = ioctl(ixj1,
                   PHONE_EXCEPTION);

     /* ellenőrizzük, hogy a felhasználó
        lerakta vagy felvette a telefont */
     if(ixje.bits.hookstate)
     {
       if(ioctl(ixj1, PHONE_HOOKSTATE))
       {
         printf("Felvette\n");
       }
       else
       printf("Lerakta\n");
     }
   }
}
```

kodek (tömörítetlen 16 bites hangminták) alapértelmezett adatkocka-mérete 240 bájt – ez 30 ezredmásodpercnyi adatnak felel meg 8000 Hz-en mintavételezve. Az eszközről minden alkalommal 240 bájtot lehet beolvasni, vagy semmit. Természetesen ez a viselkedés megváltoztatható. Különböző ioctl hívásokkal beállítható a tömörítést nem végző kodekekhez az adatkocka mérete.

A telefoneszközök vezérlése – az ioctl rendszerhívás

A telefoneszközzel közlendő parancsokat, amelyek egy bizonyos választ váltanak ki, nem a write hívással írjuk az eszközre, hiszen mint fent említettük, csak a hangadatok kerülnek írás műveleten keresztül az eszközre. Az alapvető telefonszolgáltatások vezérlését ioctl rendszerhívásokkal valósítják meg. Ezeket az alapvető ioctl függvényeket a /usr/include/linux/telephony.h fejlécfájlban adták meg. A gyártók kibővíthetik ezt az alapvető függvénykészletet, de azok a függvények csak a saját eszközmeghajtójukra korlátozódnak, és nem lesznek részei a közös linuxos telefonos API-nak.
Ezt a lehetőséget egy példával világíthatjuk meg a legjobban. A telefonos API-t egy Quicknet Internet PhoneJACK kártyával használjuk, amelyhez egy telefon csatlakozik (a telefonos szakemberek FXS kapunak vagy POTS kapunak nevezik). A *2. lista* rövid példaprogramja megcsörgeti a telefont.
Láthatjuk, hogy ha a felhasználó nem adja meg a parancssorban az eszköz nevét, akkor a /dev/phone0 nyílik meg. A csengetések legnagyobb számát egy ioctl hívással állítjuk be a PHONE_MAXRINGS állandó segítségével. Ezután a telefont csörgésre késztetjük a PHONE_RING ioctl által. A példaprogram egy egyszerűsített változata az LGPL-es ring.c modulnak, amely a Quicknet Software Developer Kit (SDK) része. Egyszerűsége miatt nem várhatunk tőle többet, mint hogy bemutassa a telefoneszköz használatának módját és az ioctl hívásokon keresztüli vezérlést, de a valódi programokat sem kell sokkal jobban bonyolítani, az API elég egyszerű. Az összes linuxos telefonos API-hoz tartozó ioctl állandó a /etc/include/linux/telephony.h fejlécfájlban van meghatározva.
A hangkártyákhoz írott programok könnyen átültethetők telefoneszközökre, ugyanis csak a hangadatokat írjuk és olvassuk a read és a write rendszerhívásokkal. Ráadásul az ioctl hívásokban használt alacsony szintű állandók úgy lettek meghatározva, hogy ne ütközzenek a létező hangkártyák ioctl állandóival. Egy hangkártyás alkalmazás átültetése telefonkártyára főként abból áll, hogy a saját kódunk által kiadott hangkártyás ioctl hívásokra adott hibákat kezeljük. Lehetségessé válik (bár talán nem könnyű) egy héjalkalmazást írni, amely megnyitja a telefoneszközt, elindít egy gyermekfolyamatot, (amely megörökli a telefoneszköz megnyitott fájlleíróját) és ebben a folyamatban futtatja a hangkártyával működő alkalmazást. Bár ez nem teljesen átlátszó, de lehetséges és a gyakorlatban nem is lehet olyan nehéz.

Az aszinkron eseményértesítő

Az eszköz telefonos oldalán bekövetkező eseményekről a telefont működtető felhasználói térben futó alkalmazást értesíteni kell.
A régi, nem túl szép módszer erre az, hogy a program állandóan lekérdezi az eszköz állapotát és a változásokat. A Linux telefonos

API-ja természetesen elkerüli ezt a megoldást, és két másik módszert alkalmaz, mindkettőt általában „aszinkron eseményértesítőnek” nevezik. Az első módszer jelzéseket (signal) használ, a második a kivételbitet állítja be a fájlleíró kivétel-adatszerkezetében. Tekintsük át sorban mindkét módszert.

A jelzések használata eseményértesítőként a következő három lépést foglalja magába: először SIGIO jelzéshez kell a jelzéskezelő függvényt előkészíteni és megadni, másodszor be kell állítani a futó folyamat folyamatazonosítóját (PID), hogy fogadja a jelzést, harmadszor engedélyezni kell a jelzés létrehozását a megnyitott fájlleírón az fcntl rendszerhívás segítségével. E három lépést részletesen tárgyalja *W. Richard Stevens* Advanced Programming in the UNIX Environment című könyvében a 12. fejezetben. (A könyvet egyébként is érdemes elolvasni.) Ismét egy rövid példával világíthatjuk meg a legjobban a dolgot. Tegyük fel, hogy van egy megnyitott fájlleírónk, a neve ixj1, és ez egy megnyitott telefoneszköz. A következő programrészlet engedélyezi a jelzésekkel megvalósított aszinkron eseményértesítést:

```
signal(SIGIO, &getdata);
fcntl(ixj1, F_SETOWN, getpid());
oflags1 = fcntl(ixj1, F_GETFL);
fcntl(ixj1, F_SETFL, oflags1 | FASYNC);
```

A kapcsolódó jelzéskezelő függvény (a getdata a fenti kódban) dolgozza fel az adatokat. Amikor jelzés érkezik, nem tudjuk, hogy milyen esemény történt, csak azt tudjuk, hogy történt valami. A programunknak ezután meg kell kérdeznie a telefoneszköztől egy ioctl híváson keresztül, hogy milyen eseményt észlelt. Ráadásul, ha egynél több megnyitott eszközfájlleíróval van dolgunk, nem tudhatjuk, hogy melyik küldte a jelzést. A jelzésekkel nehéz dolgozni, nem megbízhatóak többszálú környezetben, ezért egyes fejlesztők elkerülik őket. Ezen tényezők korlátozzák a módszer hatékonyságát, ez a hibalehetőség használhatóbb kiküszöbölése felé vezet: a fájlleírók kivétel-adatszerkezetében levő „kivételbit” használata felé.

A programok gyakran élnek azzal a lehetőséggel, hogy beállítják az írási és az olvasási adatszerkezeteket valamilyen értékre, azután a select( ) rendszerhívással kivárják, hogy a fájlleíró írható vagy olvasható legyen. Kevésbé ismert a select( ) rendszerhívás a kivétel-adatszerkezetre. A linuxos telefonos API ezt a kivétel-adatszerkezetet használja a telefonos események jelzésére. A *3. lista* egy egyszerű példát mutat az ixj1 fájlleíró felhasználására.

Ez a végtelenül leegyszerűsített (és nem túl hasznos) példa kizárólag arra alkalmas, hogy bemutassa a kivétel-adatszerkezet és a select( ) használatát az események észlelésére. Esemény bekövetkeztekor a telefoneszköz meghajtóprogramja beállítja a kivétel-adatszerkezet megfelelő bitjét, ennek hatására a select( ) visszatér. A felhasználó megvizsgálhatja az olvasási, az írási és a kivétel-adatszerkezeteket, és megállapíthatja, hogy az eszközmeghajtó által megjelölt saját fájlleírója milyen műveletek elvégzésére kész. Ha az adatok olvasásra készek, akkor az FD_ISSET(ixj1,&rfds) kifejezés IGAZ értékkel tér vissza. Az FD_ISSET(ixj1,&wfds) kifejezés akkor tér vissza IGAZ értékkel, ha az eszköz kész adatok fogadására. Végül az FD_ISSET(ixj1,&efds) akkor ad IGAZ értéket, ha telefonos esemény következett be. Hogy állapíthatjuk meg, hogy pontosan mi történt?

Az API egy különleges PHONE_EXCEPTION ioctl hívást tesz lehetővé, ehhez a visszatérési érték megfejtését segítő telephony_exception adatszerkezet tartozik. A hívás által az adatszerkezetben beállított bitekből következtethetünk arra, hogy milyen telefonos esemény történt (sokféle lehet). A fenti példában a „hookstate” bitet vizsgáltuk az if(ixje.bits.hookstate) kifejezéssel. A bit beállítása jelzi az állapotváltozást. Ezután egy ioctl hívással döntjük el, hogy letették vagy felvették a telefont. Ennek a módszernek a részletes magyarázata meghaladja cikkünk kereteit, de az érdeklődők utánanézhetnek a /usr/include/linux/telephony.h fájlban, hogy milyen események észlelhetők.

A jelenleg elérhető nyílt forráskódú programok

Már ma is sok nyílt forráskódú program használja ezt az API-t. A legjobban ismert és legszélesebb körben használt program az ophone, amely a nyílt forráskódú OpenH323 programkönyvtárat használó konzolos alkalmazás. Az ophone része az OpenH323 projektnek ➲ http://www.openh323.org, és több ezer ember használja nap mint nap ingyenes, csúcsminőségű, Interneten keresztüli telefonhívásokhoz. Az ophone nem csak a Linux telefonos API-ját támogatja teljes mértékben, hanem más H.323-alapú termékekkel is megfeleltethető, mint például a Microsoft NetMeeting vagy a Cisco hangátvitelre képes útválasztói. Ennél részletesebben terjedelmi okokból nem írhatunk erről a remek programról, de mindenkit bátorítunk a webhelyük megtekintésére. Az OpenH323 programkönyvtárat kifejlesztő céget nemrég vásárolta fel a Quicknet Technologies vállalat, hogy továbbra is biztosítva legyen ennek a nyílt forráskódú projektnek a fejlődése. Az ilyen üzleti háttérrel és a nyílt forráskód melletti elkötelezettséggel bíró OpenH323 projekt várhatóan még sikeresebb lesz a közeljövőben.

Összefoglalás

A Linux telefonos API-ja egységes és teljes felületet teremt a telefont használó programok Linux alatti fejlesztéséhez. Bár még csak egy gyártó (a Quicknet Technologies Inc.) szállít az API-nak teljesen megfelelő meghajtóprogramot, sokan mások is dolgoznak hasonló meghajtóprogramok fejlesztésén. Az API egyszerű felépítésű, gondosan megtervezett, nem ütközik a jelenlegi hangkártyák API-jával, és ugyanazzal a felülettel képes többféle gyártó termékét kiszolgálni. A következő évben biztosan kifejlesztenek még néhány nagyon érdekes telefonos programot Linuxra.

Greg Herlein (greg@herlein.com)
A californiai San Franciscóban él és dolgozik.1994 óta lelkes Linux-fejlesztő. Jelenleg a Herlein Engineering Linux/Unix tanácsadással foglalkozó cég munkatársa.

Kapcsolódó címek

OpenH323 projekt
➲ http://www.openh323.org/

OpenSwitch projekt
➲ http://www.openswitch.org/

Quicknet Linux termékek
➲ http://www.linuxjack.com/

LinuxTelephony.Org
➲ http://www.linuxtelephony.org/

Asterisk PBX projekt
➲ http://www.asteriskpbx.org/

Voxilla – Nyílt forráskódú telefonálás
➲ http://www.voxilla.org/

OpenGatekeeper projekt
➲ http://www.opengatekeeper.org/

Packetizer – a VoIP módszertani oldala
➲ http://www.packetizer.com/

Az IETF IP-telefonálás munkacsoportja
➲ http://www.ietf.org/html.charters/iptel-charter.html

Jegyzőkönyvezés a ReiserFS segítségével

Ismerjük meg a Reiser fájlrendszert,
milyen a felépítése és miféle szolgáltatásokat nyújt.

Napjainkban a Linuxszal együtt megjelent néhány új fájlrendszer is, olyan szolgáltatásokkal, melyeket eddig mind az asztali gépek, mind a kiaszolgálók esetében hiányoltunk. Először röviden áttekintjük a ReiserFS néhány fontos szolgáltatását, majd kicsit részletesebben kitérünk a jegyzőkönyvezési réteg működésére.
A ReiserFS a fájlrendszer minden objektumát egy egyszerű B* fában tárolja. A fa a következőket támogatja:

- dinamikus fájlleíró-kiosztás (i-node allocation)
- tömör, indexelt könyvtárak
- átméretezhető elemek
- 60 bites eltolások.

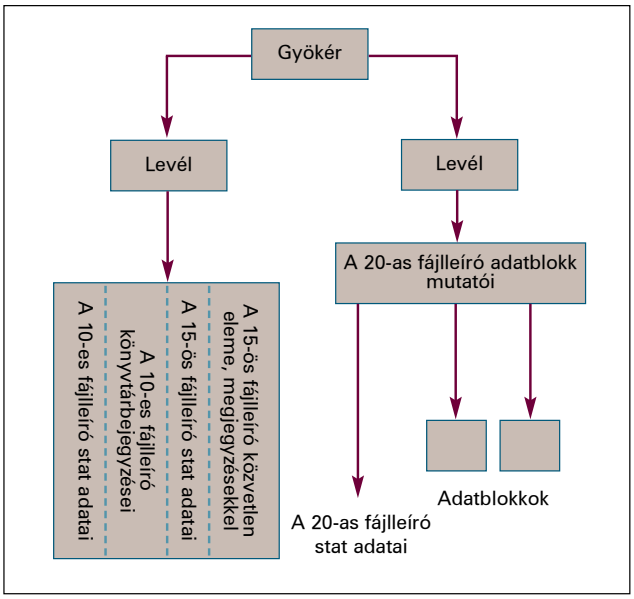
A fában található elemek négy alapvető csoportra oszthatók: a kimutatásadatokra, a könyvtárelemekre, a közvetett és a közvetlen elemekre. Az elemek közt egy kulcs alapján lehet keresni. A kulcs egy azonosítót, a keresett objektum eltolását és az elem típusát tartalmazza.
A ReiserFS könyvtárai tartalmuk változását követve növekednek és csökkennek. A fájl eltolását a könyvtárban a fájlnév egy darabjának használatával tartja nyilván a rendszer. Az így kezelt fájlbejegyzéseket egy fában tárolva nagyméretű könyvtárak hozhatók létre, miközben nem kell a teljesítmény különösebb csökkenésétől tartani, illetve megőrizhető az NFS és a megszokott könyvtárműveletek megfelelő támogatása.
A fájlok esetében a közvetett elemek adatblokkokra mutatnak, a közvetlen elemek pedig becsomagolt fájladatokat tartalmaznak. Így a becsomagolt fájladatok tárolása közvetlenül a fában történik, és a fa csomópontjaiban lévő helyet meg lehet osztani más objektumok elemeivel is. Tehát a nagyméretű fájlokhoz a ReiserFS az ext2 által használtakhoz hasonló blokkmutatókat tárol, de a kisebb fájlok adatait képes egybecsomagolni, ezzel lemezterületet takarít meg.
A fa egyensúlyának megtartásával a fenti elemek mindegyike átméretezhető. Mód nyílik a becsomagolt fájladatokhoz való hozzáfűzésre. Ha a kimutatásadatok közt újabb mezőre van szükségünk, és a lefoglalt terület megnövekszik, be tudja fogadni az újabb adatokat. A lemezformátum sokkal több részletére érdemes kitérni, ezért az érdeklődőknek ajánlom, nézzenek el a ReiserFS honlapjára (lásd a Kapcsolódó címet).

Nagyméretű fájlok támogatása

A ReiserFS jelenleg két fő lemezformátummal rendelkezik.
A 2.4-es kóddal együtt bevezetett új formátum 60 bites fájleltolásokat tesz lehetővé, míg a 2.2-es kódban használt formátum 32 bites eltolásokkal működik. Amikor egy régebbi fájlrendszert fűzünk be az új rendszermaggal, a régi formátumot a rendszer megőrzi, és nem engedi nagyméretű fájlok használatát.
Létezik olyan befűzési lehetőség is, mely az új formátumra alakítja át a fájlrendszert, azonban ennek 2.2-es rendszermag alatti változata egyelőre bétaállapotú. Nem szeretnék elavult adatokat közölni a cikkben, így inkább a ReiserFS honlapjának meglátogatását javaslom, ahol további részletek találhatók a nagyméretű fájlok támogatásáról.

Hogyan működik a jegyzőkönyvezés?

Mielőtt a jegyzőkönyvezés működéséről beszélnénk, először tekintsük át a megoldandó feladatot. Ahhoz, hogy a rendszer esetleges összeomlása után is hibamentes fájlrendszerünk legyen, a frissítésnek atominak kell lennie, azaz vagy teljesen végrehajtódik, vagy semennyire. Például ha blokkokat szeretnénk fűzni egy fájlhoz, frissítenünk kell a fájl blokkmutatóit, blokkokat kell keresni a szabad blokkok listájából, és frissíteni kell a szuperblokkot. Ha a rendszer a változtatások közben összeomlik, lehet olyan fájlmutató, mely továbbra is a szabad blokkok listáján lévő blokkra mutat, vagy a szuperblokk kimutatásadatai nem frissülnek, esetleg a lefoglalt terület elvész (azaz sem a fájlban, sem a szabad blokkok listáján nem szerepel majd).



1. ábra

A ReiserFS jegyzőkönyve egy egyszerű, előreíró jegyzőkönyvezési rendszer (kizárólag metaadatokkal dolgozik). Ennek alapötlete az, hogy mielőtt bármilyen változtatást rögzítenénk a lemezen, azt előbb a jegyzőkönyvbe írjuk. Összeomlás esetén a végrehajtott műveletsorokat újra lejátsszuk, ez lényegében nem jelent mást, mint a kérdéses adatok másolását a jegyzőkönyvből a fő lemezterületre.
Tulajdonképpen nem a változtatások jegyzőkönyvbe rögzítése, az ami megnehezíti a jegyzőkönyvezést. A feladat nehézkes része az, hogy a fájlrendszert ne lassítsuk le egy ráérős teknősbéka sebességére.
A sebesség megtartásának legnyilvánvalóbb módja az, hogy a jegyzőkönyvet nagy méretű, egymást követő adagokban írjuk ki, így csökkentjük a kiírt blokkok számát. A legtöbb fájlművelet kisszámú blokkon végez változtatást, a jegyzőkönyv ezt a tulajdonságot kihasználva több műveletet is képes egyetlen atomi egységben kezelni.
A módosított területeket nem lehet kiírni, míg át nem másoltuk őket a jegyzőkönyvbe, és nem szabadíthatók fel, míg ki nem írtuk őket.
A nagyobb méretű műveletek több magmemóriát foglalnak le ugyan,

de egyéb egyszerűsítési lehetőségeket is felvetnek. Mivel a ReiserFS mindent egy kiegyensúlyozott fában tárol, a fát gyakran kell módosítani és kiegyensúlyozni. A fa blokkjait lefoglaljuk, módosítjuk, majd egy későbbi kiegyensúlyozás során felszabadítjuk. A nagyobb méretű műveletekkel növeljük annak esélyét, hogy egy-egy blokk felszabadul, mielőtt rögzítenénk a jegyzőkönyvbe vagy a lemezre.
A blokkok esetében gyakori, hogy újra és újra jegyzőkönyvezzük őket. Ha a szuperblokkot az első, a második és a harmadik művelet is érinti, mindegyiknél ki kell egyszer írni a jegyzőkönyvbe. A lemezre azonban nem kell rögzíteni, csak miután a harmadik művelet is véget ért. Az írások száma ezzel összességében kisebb lesz, és ezek túlnyomó része is a soros felépítésű jegyzőkönyvbe történik. Ezzel egyes esetekben előfordulhat, hogy a jegyzőkönyvezés gyorsabb lesz, mint az eredeti fájlrendszer.
Amikor lehetséges, a be- és kiviteli műveletek jegyzőkönyvezését egy külön szál végzi, a kreiserfsd. Ennek révén lehetővé válik a háttérben történő végrehajtás, a felhasználói folyamatok lelassítása nélkül. Emellett viszont a jegyzőkönyv meghatározott méretű, így elképzelhető, hogy a felhasználói folyamatoknak várakozniuk kell, amíg egy-egy új művelethez hely szabadul fel a jegyzőkönyvben. Kulcsfontosságú feladat, hogy a jegyzőkönyvre várakozó folyamatok ne kössenek le olyan erőforrásokat, melyekre a már műveletet végző folyamatoknak is szükségük van.
A fájlrendszerek többségének nem kell tisztában lennie azzal, hogy egy jegyzőkönyvezési réteg is gondoskodik a dolgok biztonságos menetéről, van azonban néhány olyan szabály, amit be kell tartani. Először is nem biztonságos a piszkos pufferek módosítása. SMP-rendszerek esetében egy másik processzor is írhat a pufferbe, míg azt módosítjuk. Ez azt jelenti, hogy a módosítások a művelet teljes végrehajtása előtt a lemezre íródhatnak.
A legtöbb művelet csak korlátozott számú puffert módosít, de a fájlokba történő írások és a csonkítások gyakorlatilag korlátlanok. Ahelyett, hogy a jegyzőkönyvezési rétegben támogatnánk a végtelen hosszú műveletsorokat, inkább összefüggőségi támpontokat iktatunk be a műveletsorokba. Ha a folyó műveletsort be kell fejezni, a fájlrendszer összefüggő állapotának fenntartásához elegendő mennyiségű adatot írunk a jegyzőkönyvbe, majd új műveletsort indítunk el. Ha adat-jegyzőkönyvezést is használunk, az fsyncnek is hasonló ellenőrzéseket kell végeznie.
Szintén a jegyzőkönyvezési réteg által hozott új szabály a kizárólag metaadatokat használó jegyzőkönyvezésnél jelenik meg a blokkok újrafelhasználásával kapcsolatban. Képzeljük el a következő két műveletet:

1. A 200-as blokk lefoglalása, beillesztés a fába.
   A 200-as blokk megváltoztatása és jegyzőkönyvezése.
   A 200-as blokk felszabadítása.
   Az első műveletsor lezárása és végrehajtása.
2. A 200-as blokk lefoglalása adatblokknak.
   A 200-as blokk megváltoztatása, fsync a lemezre.
   A második műveletsor lezárása és végrehajtása.

2. ábra

Rendszerösszeomlás után

Az összeomlást követően a műveletsorokat szabályosan újrajátsszuk. Az egyes műveletsorok újrajátszása közben a 200-as blokk jegyzőkönyvezett változatát a fő lemezre másoljuk, a második műveletsor újrajátszását követően pedig a 200-as blokk egy adatblokk egy fájlban. Csakhogy az fsync által a 200-as blokkba írt adatok már nincsenek a helyükön. A ReiserFS úgy kerüli el ezt a helyzetet, hogy sosem foglal le adatblokkot, amíg nullára nem csökken annak esélye, hogy egy jegyzőkönyv-visszajátszás elavult adatokkal felülírhatja annak tartalmát. Amikor a fájlrendszer betelik, azt jelenti, hogy a műveletsorokat ki kell írnunk a lemezre, és újra felhasználható blokkokat kell találnunk. Ugyanilyen ellenőrzéseket kell végrehajtani, ha egy adatblokkot jegyzőkönyvezzünk, majd később közvetlenül felülírjuk.
Most, hogy nincs szükség az fsck futtatására minden rendszerösszeomlás után, még körültekintőbben kell bánnunk az elveszett állományokkal. Egy leválasztott fájl valójában még nem törlődik, amíg az azt megnyitó folyamatok futása be nem fejeződik. Ha a rendszer összeomlik, mielőtt a törlési művelet befejeződne, a jegyzőkönyv összefüggő fájlrendszert fog létrehozni, viszont valamennyi lemezhelyet továbbra is lefoglal a fájl számára. Mivel a fájl nem szerepel a könyvtárfában, a blokkok visszaállítására nincs többé lehetőség.
A fenti hiba legkönnyebb kiküszöbölése, ha a fájlt egy különleges könyvtárba helyezzük. A ReiserFS könyvtárak nagyon gyorsak, és nem kell különösebben aggódni a zárolások miatt sem, ha a fájlnevek nem ütköznek egymással. Összeomlást követően a könyvtárt kiolvassuk, és befejezzük a fájltörléseket az összes megmaradt objektumhoz. A különleges könyvtárnak valójában fájlnevekre sincs szüksége, csak a fájlok megtalálásához szükséges adatokra. Ez a javítás jelenleg ugyan nem található meg a hivatalos ReiserFS kiadásokban, de a helyzet hamarosan megváltozik.

A felhasználói terület műveletsorai

A felhasználók időről időre szeretnék tudni az alkalmazásprogramozási felület által a felhasználói területre kivitt műveletsorok változatszámát. A ReiserFS jegyzőkönyvezési rétegét befejezett műveletek támogatására tervezték, ezek általában nagyon hamar végrehajtódnak, így nem működne jól egy általános műveletsorkezelő szerepében.
Az azonban nem lenne jó ötlet, hogy atomi írásokat engedélyezzünk a felhasználói terület számára, és ezzel nagyobb ellenőrzést tegyünk lehetővé a műveletek csoportosítására. Ilyen módon ugyanis egy alkalmazás kérhetné egy 64 kB-os fájl létrehozását egy megadott könyvtárban, és mindezt atomi műveletként kezelhetné. Mindeddig elég csekély tervezési munka folyt ezen a területen.

VM beillesztése

Ahogy fogyatkozik a rendszer memóriája, a rendszermagnak ki kell ürítenie a piszkos átmeneti tárak adatait a lemezre, ezáltal memórialapok szabadíthatók fel. Csakhogy a még végre nem hajtott műveletsorok által lekötött memória nem szabadítható fel a végrehajtásig, így a VM gyakorlatilag tehetetlen a fájlrendszer segítsége nélkül. Biztosak szeretnénk lenni abban, hogy a jegyzőkönyvezés a lekötött pufferek miatt nem foglalja el a rendszer memóriájának túlságosan nagy hányadát.
Együtt fogunk dolgozni a VM fejlesztőivel, hogy megfelelően sikerüljön jelezni a fájlrendszernek a memóriahiányt. Az API jelenleg

nincs kőbe vésve, de a felhasználók a jelek szerint hajlanak egy, a laphoz tartozó tárürítő visszahívásos függvény támogatására, illetve egy általános memóriaigény-bejegyző rendszer használatára. Egyelőre nem tudni, hogy mindebből mi valósul meg a 2.4-es rendszermagban, és mi marad a 2.5-ös változatra.

A ReiserFS és az LVM

Az LVM nagy csokor szolgáltatással bővíti a Linuxot, ezek egyike a csak olvasható pillanatfelvételek készítése egy meghajtóról. Ennek elkészítése roppant gyorsan zajlik. Egy *másolás íráskor* eljárás pedig gondoskodik a pillanatfelvétel frissítéséről, ahogy az eredeti meghajtót módosítjuk. Ezzel a módszerrel gyakorlatilag bármely fájlrendszeren a legtöbb programhoz állandó elérésű, összefüggő biztonsági mentések készíthetők.

A jegyzőkönyvezett fájlrendszer ezt megnehezíti egy kicsit. Amikor a syncet meghívjuk ReiserFS-en, csak kiírjuk a metaadatok változását a jegyzőkönyvbe, azzal a tudattal, hogy egy esetleges rendszerösszeomlást követően a visszajátszás mindent megfelelő állapotba állít vissza. Egy csak olvasható LVM pillanatfelvétel esetében a jegyzőkönyv visszajátszása nem lehetséges. Ehelyett új, általános hívásokat alkalmazhatunk, ezek mindent kiürítenek a lemezre, és szüneteltetik a fájlrendszer újabb módosításait. Amíg a szünet tart, az LVM frissíti a pillanatfelvételt, így az a jegyzőkönyv visszajátszása nélkül is összefüggő lesz. Amikor az LVM végzett, megszüntetjük a fájlrendszer zárolását, és ezután megszokott módon folytatódnak az írási műveletek.

Minden fájlrendszerrel dolgozó műveletnek tudni kell várnia a jegyzőkönyvre, ennek megvalósítása a ReiserFS esetében könnyen ment. Az LVM 0.9 és a ReiserFS 3.6.18 képesek erre a szolgáltatásra, de egyelőre nem tudni, az általános hívások mikor kerülnek be a rendszermagba. Ettől függetlenül a hiányzó részleteket pótló javítások elérhetők lesznek mind a ReiserFS, mind az LVM honlapján.

Az LVM egy másik szolgáltatása az egyik meghajtó tartalmának áthelyezése másikra. Ha olyan területet találunk a lemezen, mely az átlagosnál nagyobb forgalmat bonyolít le, a kérdéses blokkokat áthelyezhetjük egy másik, gyorsabb meghajtóra. Tulajdonképpen a teljes jegyzőkönyvezési területet át lehetne helyezni egy gyorsabb meghajtóra, ezzel csökkentve a fejek terhelését és a fejléptetések számát. Így ténylegesen javítható a sok jegyzőkönyvezést igénylő alkalmazások teljesítménye.

Programbeli RAID

A 2.2-es rendszermagokban a RAID programja lekötött puffereket is kiírhat a lemezre, ezzel megszegheti az írások sorrendjére vonatkozó, a dolgok összefüggő állapotban való tartásához szükséges szabályokat. Csak a meghajtók csíkokra osztása és egybefogása lenne teljesen biztonságos, a tükrözés csak addig megbízható, amíg nem használjuk az üzem alatti helyreállító (on-line rebuild) programkódot. A 2.4-es rendszermagokban minden programból megvalósított RAID-szint megfelelően együttműködik a jegyzőkönyvezett fájlrendszerekkel.

ReiserFS és NFS

A ReiserFS gondokkal küszködik az NFS-támogatást illetően, mivel 64 bitnyi adatot igényel egy objektum azonosításához a fában, az NFS viszont a fájlleírókat azok 32 bites azonosítója alapján próbálja meg azonosítani. A jó hír az, hogy az NFS fájlkezelőnek elegendő helye van a ReiserFS által igényelt többletadat tárolásához. Néhány rendszermagfejlesztő készített olyan API-felületeket, amelyek a fájlrendszer számára lehetővé teszik az ellenőrzést a fájlkezelők egy része felett. A cikk megjelenésének időpontjában valószínűleg már vannak nyilvános javítások, amelyek megfelelő NFS-támogatást adnak a ReiserFS-hez.

Írások gyorstárazása

A teljesítmény növelésére egyes újabb lemezmeghajtók alapértelmezett állapotban visszaírásos gyorstárazást használnak. Ez azt jelenti, hogy a meghajtó a műveletet még azelőtt befejezettnek jelzi, hogy az adat ténylegesen az adathordozóra került volna. A blokk továbbra is a meghajtó gyorstárában van, ahol megváltozhat az írások sorrendje. Ebben az esetben a metaadatok változásai még azelőtt kiíródnak, hogy jegyzőkönyv végrehajtaná a blokkműveleteket, és ez hibához vezethet, ha időközben megszűnik a rendszer tápellátása. Nagyon fontos, hogy azoknál az eszközöknél, amelyek nincsenek ilyen hibaesetekre felkészítve, a visszaírásos gyorstárazást letiltsuk. Egyes RAID-vezérlőkártyáknak saját akkumulátorral felszerelt visszaírásos gyorstáruk van, ezek áramkimaradás esetén sem vesztik el tartalmukat. Használatuk ugyan biztonságos, de rendszeresen ellenőrizni kell az akkumulátorok állapotát. Megdöbbentő teljesítménynövekedést lehet tapasztalni a hasonló gyorstárak használatával, főleg jegyzőkönyv-igényes alkalmazásoknál, például a levelezőkiszolgálóknál.

Levelezőkiszolgálók

A levelezőkiszolgálók a legrosszabbak a jegyzőkönyvezett fájlrendszerek számára, mivel minden egyes fájlművelet befejeztéről meg kell bizonyosodniuk. Az fsyncet használják, esetleg egyéb trükkök egész sorát, hogy elkerüljék az üzenetek elvesztését egy esetleges rendszerösszeomláskor. Emiatt a fájlrendszernek gyakran rendkívül kicsi műveletsorokat is le kell zárnia.

A levelezőkiszolgálóknak valamilyen gyors módszerrel a lemezre kell írnunk az új fájlokat, az adatok jegyzőkönyvezése ebben segítségükre lehet. Az fsync hívás alatt jegyzőkönyvezzük az adatblokkokat és a fájlnak a fába helyezéséhez szükséges metaadatokat, ezzel egy nagyméretű, soros írást hozva létre. Ha az írásra kerülő fájl csak egy átmeneti várakozási sorba kerül, lehet, hogy soha nem írjuk ki a lemezre. Gyors, csak jegyzőkönyvezésre használt meghajtóval együtt alkalmazva az adat-jegyzőkönyvezés jelentős teljesítménynövekedést hozhat. Jelenleg a ReiserFS 2.4-es programkódja nem támogatja az adat-jegyzőkönyvezést. A 2.2-es változatú rendszermagokhoz készült kiadásban van adat-jegyzőkönyvezés, de ha át akarjuk ültetni a 2.4-es rendszermagra, akkor az eltérő gyorstárazó rendszere miatt a kódot át kell alakítani.

A versenytársak

Az új linuxos fájlrendszerek megjelenése rendkívül fontos. A rendszergazdák megválaszthatják azt a terméket, amely a legjobban megfelel az általuk futtatott alkalmazáshoz, a programozók pedig mások eredményeivel hasonlíthatják össze saját döntéseiket. Egészében véve a Linux csak nyerhet azon, hogy a közösség tagjai kiválasztják és használják az egyes fájlrendszerek leghasznosabb szolgáltatásait.

Chris Mason
(mason@suse.com) Mielőtt belefogott a ReiserFS fejlesztési tervezetébe rendszergazdaként tevékenykedett. Jelenleg teljes munkaidőben rochesteri otthonából (New York) dolgozik a SuSE-nak.

Kapcsolódó cím

➲ http://www.reiserfs.org/ további adatokat találhatunk a ReiserFS telepítésével és használatával kapcsolatban, illetve a résztvevő programozókról is. Ezenkívül olvashatunk a lemezformátumról, és a leggyakrabban felmerülő általános kérdésekre is választ lelhetünk.

Könnyű álmok (4. rész)

A hálózati határvédelem alapjai

Ez cikk keretein belül két témát érintünk: a hálózati védelem kialakításának alapdokumentumát, a határvédelmi tervet (általában a security policy kifejezés használatos), valamint kissé közelebbről megismerkedünk a csomagszűrők tulajdonságaival.
Sokan rutinból állnak neki egy biztonsági rendszer kivitelezésének. Ha azonban nemcsak a lélek megnyugtatása a cél, hanem a valós védelem, akkor a védelmi rendszert beállítás előtt meg kell tervezni. Miután felmértük mit akarunk megvédeni és mitől, ezeket az ismereteket valamilyen félreérthetetlen szabálykönyvbe rögzíteni kell. Ennek egy lehetséges formáját mutatjuk meg.
A Linux rendszermag számtalan hasznos és biztonságot növelő lehetőséget ad, például a Linux csomagszűrője sok tekintetben felülmúlja a pénzes rendszereket. Segítségével szabályozhatjuk az átáramló forgalmat annak feladója és célja szerint, kialakíthatunk olyan belső hálózatot, melynek teljes forgalma az útválasztó másik oldalán egyetlen számítógépnek látszik (masquerade), így a teljes hálózatnak csak egyetlen közismert hálózati címre van szüksége. Mivel az útválasztó Linux-alapú, szükség esetén futtathatunk rajta levélkiszolgálót, webgyorsítótárat vagy akár hálózati forgalomelemzőt.
A jobb érthetőség kedvéért állítsunk fel egy olyan példát, melyen keresztül elmagyarázható, hogy a rendszer mire képes, és mire nem.

Egy képszerű példa

Képzeljünk el egy bolygót városokkal, azokban házakkal, bennük lakásokkal. Minden háznak egyedi címe van. A házak lakói gyakran küldenek egymásnak csomagokat postán, akár másik városba. Minden városban van legalább egy postahivatal, ahol a más városokból érkező, vagy azok felé induló csomagok útját meghatározzák, sőt nagyobb városokban akár több ilyen hivatal is lehet. Az egyes csomagokat postahivatalok döntik el, hogy merre kell továbbszállítani. Mi történik akkor, ha más városokban rosszindulatú emberek unalmukban bombákat küldözgetnek csomagjaikban? Együtt élhetünk a dologgal vagy megpróbálhatjuk megakadályozni. Ennek kézenfekvő módja a postahivatalokban a csomagok továbbításának valamilyen szabályozása, a veszélyes csomagok kiválogatása és megsemmisítése. Mivel a csomagok a városokba csak postahivatalokon keresztül juthatnak be, itt könnyen megmondhatjuk, hogy mely városokból nem fogadunk el csomagokat, mert tudjuk, hogy az adott város lakói általában rosszindulatúak, vagy bizonyos lakók csomagjait tiltjuk ki, mondván, ők megbízhatatlanok. A csomagokat fel is bonthatjuk, és csak akkor tiltjuk meg a továbbításukat, ha azok valóban bombát tartalmaznak, ehhez természetesen több postásra lesz szükségünk.
Most fordítsuk le a fenti példát hálózatokra. A példabeli bolygó megfelel a Föld óriáshálózatának, az Internetnek, a városok az Internet egyes alhálózatai. A házak a hálózaton lévő számítógépek, a lakások a kapuknak, a lakók pedig a kapukon figyelő démonoknak felelnek meg. Ebben az esetben egy kapuban egyszerre csak egy lakó lehet. Minden hálózat bejáratánál van egy útválasztásra alkalmas hálózati elem, ez a példánkban említett postahivatal. Ha a saját hálózatomat (a város) fenyegetve érzem (korábbi cikkeinkben igyekeztünk bemutatni, hogy együgyűség lenne nem félni), akkor annak hálózati kapcsolódási pontjainál (a postahivatalok) érdemes valamilyen védelmet alkalmazni (csomagok kiválogatása). Ehhez előre meg kell határozni, hogy mely hálózatrészek között milyen forgalom haladhat át. Ezt a tervet a gyakorlatban hálózati határvédelmi tervnek, vagy az adott hálózat határvédelmi politikájának hívják (security policy). Lásd a mellékletben (58. oldal). Ha okos jelelosztónk van, ami a forgalmat a csomagok forrása és célja szerint szűri, akkor a rendszert csomagszűrő rendszernek (Packet Filter) hívják. Ha a csomagszűrő képes a csomagok által kifeszített kapcsolatot egységként kezelni (akár UDP-n is!), akkor az már állapottartó csomagszűrő (Stateful Packet Filter – SPF). Ez példánkban a több csomagból álló kapcsolat. Az SPF rendszerek ezenkívül gyakran képesek a csomagok tartalmát is megvizsgálni, és valamilyen szinten a protokollok vizsgálatát is elvégzik. A teljes protokollelemző rendszereket alkalmazásszintű tűzfalnak (Application Level Firewall – ALF) hívják. Ez példánkban azt jelentené, hogy a postahivatal kibont minden egyes csomagot, és azok tartalma alapján dönti el, hogy továbbítható-e. Az alkalmazásszintű tűzfalak általában minden átvitt protokollra egy-egy különálló szűrő-programot tartalmaznak, ezek az úgynevezett alkalmazásátjárók (application gateway vagy egyszerűen csak gateway). Egy jól tervezett alkalmazásátjárón a protokoll minden eleme finoman szabályozható. Ezekről a rendszerekről a későbbiekben bővebben is lesz szó.

Határvédelmi politika (HP)

A HP tartalmazza, hogy ki, mivel, hogyan, mikor és mihez férhet hozzá. Minél pontosabban le tudjuk írni a rendszer szabályos működését, annál biztosabb, hogy az ettől eltérő eseteket ki tudjuk zárni. Leírható benne például, hogy a cég pénzügyi rendszerét kizárólag Bogár Elek használhatja, csak a saját munkaállomásáról és a cég által adott azonosító alrendszert használva. Délután öt órától reggel nyolcig nem léphet be, ilyenkor ugyanis szórakozik vagy alszik.
A HP-t kétféle hozzáállással lehet megalkotni, az egyik a „mindent szabad, ami nem tilos”, a másik az ellenkezője, „minden tilos, amit nem engedélyezek”. Biztonsági rendszerek tervezésénél általánosan elfogadott hozzáállás, hogy az utóbbi szerint kell eljárni. Így nem fordulhat elő az, hogy valami elkerüli a biztonsági politika kialakítóinak figyelmét, és lehetőséget hagy a visszaélésre. Mivel minden tilos, amit nem engedélyezünk, csak a valóban engedélyezett forgalom juthat át. A továbbiakban figyelmeztetés nélkül ezt tekintjük alapértelmezettnek. Az otthoni felhasználók esetén célszerű tudatosan védekezni, a cégek hálózati védelménél viszont szinte kötelező HP-t készíteni, mivel a védelemnek mindenképpen jól meghatározottnak kell lennie. A védelem alanyai gyakran nem örülnek túlzottan a védettségnek, mivel ez gyakran eddigi jogaik csorbulásával és szokatlan kényelmetlenségekkel járhat. A rendszer védelmi szintje és kényelme között nagyjából fordított arányosság áll fenn. Ilyenkor a vezetőség által elfogadott, aláírt HP nagyon sokat segíthet. Egy cég védelmi elveit mindig a cég vezetőinek bevonásával kell kialakítani, hiszen ők azok, akik a HP-t be nem tartó alkalmazottakkal szemben felléphetnek.
Fontos szerepe van a HP-nak a védelmi eszközök megválasztásában

1. lista /etc/ipchains.conf

```
# /etc/ipchains.conf - csomagszűrő beállító állomány az ipchains rendszerhez
#
# Azbesztkabát 1.0.0 - készítette a Könnyű álom szabadcsapat
#          az Úr 2001. évének február havában
#

:input DENY
:forward DENY
:output ACCEPT
:modem_be

-A input -i lo -j ACCEPT
-A input -p tcp -i ppp0 ! -y -d __modemip__ -j modem_be
-A input -p udp -i ppp0 -d __modemip__ -j modem_be
-A input -p icmp -i ppp0 -d __modemip__ -j modem_be
-A input -p tcp --dport auth -j REJECT

# minden, ami eddig nem került engedélyezésre, azt elutasítjuk és naplózzuk
-A input -j DENY -l

# innen a modem bejövő csomagjait vizsgáljuk
-A modem_be -p icmp --source-port destination-unreachable -j ACCEPT
-A modem_be -p udp -s x.y.z.s3 domain --destination-port 1024: -j ACCEPT
-A modem_be -p udp -s x.y.z.s3 ntp --destination-port ntp -j ACCEPT
-A modem_be -p tcp -s x.y.z.s1 smtp --destination-port 1024: -j ACCEPT
-A modem_be -p tcp -s x.y.z.s1 pop3 --destination-port 1024: -j ACCEPT
-A modem_be -p tcp -s x.y.z.s2 3128 --destination-port 1024: -j ACCEPT
-A modem_be -p tcp --source-port www --destination-port 1024: -j ACCEPT
-A modem_be -p tcp --source-port https --destination-port 1024: -j ACCEPT
-A modem_be -p tcp --source-port ssh --destination-port 1024: -j ACCEPT
-A modem_be -j DENY -l
```

is. Ha a HP csak a hozzáférés irányát határozza meg, akkor elegendő lehet egy csomagszűrő alkalmazása, ha azonban megköveteli a kapcsolat protokollszintű szűrését, módosítását vagy az erős azonosítást a tűzfalon való áthaladáshoz, akkor nem kerülhető el az alkalmazásszintű tűzfalak használata.

A határvédelmi politika tartalma

Vizsgáljuk meg, mit kell egy HP-nak tartalmaznia. Mindenekelőtt meg kell határozni a rendszer számára értelmezett hálózatokat. A határvédelem szempontjából kiemelt rendszereket is egyedi – egycímes – hálózatokként érdemes kezelni. A hálózatok leírását láthatjuk a példa HP első részében. Ahol nincs megadva hálózati maszk, ott az 32 bites. Ez azt jelenti, hogy nem egy valódi hálózatról van szó, hanem egy hálózati címről, például a szolgáltató levélkiszolgálójáról. A későbbiekben így tudjuk majd megadni azt a szabályt, hogy oda és csakis oda fogjuk engedélyezni a levelek továbbítását. Ezek után a hálózatok logikai neveit használva a lehető legpontosabban le kell írnunk, hogy milyen forgalom engedélyezett az egyes alhálózatok között. Ezt a példánkban bemutatott megoldásnál szabályokban adjuk meg. Egy szabály mindenképpen tartalmazza a forrás és célhálózat logikai nevét, az átvitt protokoll meghatározását és a célkaput (néhány esetben az utóbbi kettő megadása nehéz, vagy nem lehetséges). Ha lehetséges, a forráskaput is meg kell határozni. Általában itt csak tartományt tudunk megadni, de ez is segít a szabályos működés pontosabb leírásában. A biztonsági rendszerek aranyszabálya szerint az utolsó szabály mindig a „minden más tilos” (catch all). Az egyes protokollok átvitelének általában külön lefektetett szabályrendszere van, ezt egy külön dokumentumban rögzítik. Innen derül ki, hogy a HP-t milyen védelmi eszköz tudja megvalósítani. Ha például az elvárás az, hogy a nyilvános webkiszolgáló típusát ne lehessen meghatározni, akkor látszik, hogy egy tiszta csomagszűrő rendszerrel ezt nem lehet megvalósítani. Ha az általánoson kívül még valamilyen megszorítást tudunk tenni az adott kapcsolatra, akkor azt megjegyzésként érdemes az adott szabályba illesztenünk. Ilyen lehet például az, hogy a cég általános levéltovábbítási szabályozása nem ír elő semmilyen különleges eljárást a csatolt állományokkal kapcsolatban, de a pénzügyi rendszerbe menő levelekből el kell távolítani a futtatható állományokat. Például határvédelmi tervünk egy későbbi személyi tűzfal kialakításához lett kitalálva, azokat a kapcsolatokat engedélyezi, melyek egy jellemző otthoni Internet-felhasználónál szükségesek.

Csomagszűrő rendszerek

A HP-k elvárásainak az egyszerű csomagszűrő rendszer gyakran nem felel meg, mivel a legkisebb protokollon belüli beavatkozáshoz is alkalmazásszintű szűrésre van szükség. Ebből arra következtethetnénk, hogy a csomagszűrő felesleges, de ez nem igaz. Az alkalmazásszűrő tűzfalak első védelmi vonala csomagszűrő, így mindenképpen meg kell ismerni helyes beállításának alapjait. Alkalmazásszűrő tűzfal esetén a csomagszűrő alrendszer hárítja el a támadások jelentős részét, mivel csak az olyan kapcsolatok létrejöttét engedélyezi, amelyek a HP által engedélyezettek. Így az átjárókhoz csak az engedélyezett kapcsolatok juthatnak el, ezzel nehezítve a rendszer megbénítását (DoS).

Az egyszerű csomagszűrő rendszerek alapvető tulajdonságai:

- nagyon gyors, mivel alkalmazásszinten nem elemez
- átlátszó (transparent) – a felhasználóknak nem kell tudniuk róla
- olcsó, hiszen szinte minden útválasztásra alkalmas eszköz tartalmazza – így az ingyenes operációs rendszerek is. Érdekes tény, hogy néhány pénzért kapható operációs rendszer nem tartalmaz csomagszűrési lehetőséget.
- nagyon nagy terhelésnél olcsóbb megoldás a vas cseréje, mint több rendszer használata – nehezen méretezhető
- nagy rendszereknél a beállítás nehezen áttekinthető
- a már beállított csomagszűrő esetleges hibáit nehéz meghatározni
- bizonyos protokollok szűrése nehézkes (például FTP, NFS).

2. lista /etc/ppp/ip-up.d/ipchains

```
#!/bin/sh
# /etc/ppp/ip-up.d/ipchains - csomagszűrő
# élesítő script
#
# Azbesztkabát 1.0.0 - készítette a Könnyű álom
# szabadcsapat
#          az Úr 2001. évének február havában
#

modemdev=ppp0
ipchainsconf=/etc/ipchains.conf

modemip='ifconfig $modemdev | perl -ane 'print \
       ((split(/:/, $F[1]))[1]) if /ine/''
# Debiannál
modemip=$PPP_LOCAL

tmpdir=.tmp.'date +'%Y.%m.%d''
mkdir -m 700 /tmp/$tmpdir
if [ $? != 0 ]; then
    echo "Hiba: Az átmeneti könyvtárat nem \
       sikerült létrehozni."
    echo "      A csomagszűrő beállítása \
       sikertelen."
    exit 1
fi
sed s/__modemip__/$modemip/ $ipchains.conf > \
       /tmp/$tmpdir/ipchains.conf
ipchains -F
ipchains -I input 1 -j DENY
ipchains -X
egrep -v '^\s*(#.*)?$' /tmp/$tmpdir/ \
       ipchains.conf | ipchains-restore
ipchains -D input 1
rm -rf /tmp/$tmpdir
```

A fenti kijelentésekhez érdemes néhány megjegyzést fűzni.

- Az átlátszóság pontosabban azt jelenti, hogy sem az ügyfeleknek, sem a kiszolgálóknak nem kell tudniuk, hogy tűzfal választja el őket. Ha a tűzfal nem átlátszó, akkor az ügyfélnek erre fel kell készülnie, mivel nem a célkiszolgáló címére kell kapcsolódnia, hanem a tűzfal egy meghatározott kapujára, és a tűzfal itt kérdezi meg tőle, hogy ma hová szeretne menni. Ez természetesen az ügyfél módosítását, illetve további beállítást követel. Az ügyfél módosítása azt jelenti, hogy az eredeti protokollt kiegészítik olyan elemekkel, aminek segítségével a rendszer a tűzfallal közölni tudja eredeti úti célját, szükség esetén az áthaladó azonosítását is itt teszi lehetővé.
- A hálózatok építői gyakran feledkeznek meg róla, hogy majd minden útválasztó tartalmaz csomagszűrési lehetőséget. Ezt általában nem állítják be, vagy nem elég körültekintően, így a rendszer beállítása mindenhonnan módosítható. Pedig egy ilyen rendszernél célszerű meghatározni, hogy honnan fogad el módosítási kéréseket. Ez a legkevesebb, amit minden útválasztóban be kellene állítani, de ha már az útválasztónál meg tudjuk akadályozni a tiltott forgalmak áthaladását, akkor ésszerű megtenni, hisz így a tiltott forgalom nem terheli feleslegesen a mögöttes hálózatot.

3. lista /etc/ppp/ip-down.d/ipchains

```
#!/bin/sh
# /etc/ppp/ip-down.d/ipchains - csomagszűrő
# leállító script
#
# Azbesztkabát 1.0.0 - készítette a Könnyű
# álom szabadcsapat
#          az Úr 2001. évének február havában
#

ipchains -F
ipchains -X
ipchains -P input ACCEPT
ipchains -P forward DENY
ipchains -P output ACCEPT
```

- A méretezhetőség tűzfalaknál is azt jelenti, hogy a rendszer kis és nagy terhelés esetén is hatékonyan használható. Amennyiben a nagy áthaladó forgalom miatt a szűrést nem lehet egy géppel megvalósítani, akkor a szolgáltatások külön gépekre bontásával oldható meg. Ettől erősebb terhelés esetén, az egyes szolgáltatásokon belül is osztályozható a forgalom, például forráscím szerint. Így szélsőséges forgalommennyiség is szűrhető. Ennek a megoldásnak alkalmazásszűrő rendszereknél van igazán értelme, hiszen azon rendszerek lényegesen erőforrás-igényesebbek. A tiszta csomagszűrőknél ennek csak akkor van értelme, ha a csomagszűrő szabályrendszere nagyon sok szabályból áll, és azokat nem lehet egyszerűsíteni. A csomagszűrő rendszereknél mindig arra kell törekedni, hogy egy csomag a lehető legkevesebb vizsgálaton essen át, így kevesebb időt kell a rendszermagnak egy csomag fedolgozásával töltenie. Ennek egyszerű módja a csomagok bejövő láb szerinti válogatása, ahogy azt a későbbi példában látjuk.
- Bizonyos protokollok szűrése azért nehézkes, mert a protokoll tevezői úgy álmodták meg rendszerüket, hogy sem a forrás, sem a célkapu nem előre meghatározott. Így ha például az A hálózatból a B hálózatba át akarjuk engedni az NFS protokollt a hozzá tartozó csingilingikkel, akkor nem tehetünk mást, át kell engednünk szinte a teljes hálózati forgalmat. Erre a gondra csak egy alkalmazásszűrő nyújthat megnyugtató megoldást, az is csak abban az esetben, ha ő maga kezelheti folyamatosan a csomagszűrőt. Ha tehát az átmenő kapcsolat azt követeli, hogy a 2317-es kaput nyissuk ki, akkor a rendszer a csomagszűrőben engedélyezi a csomagok bejövetelét erre a kapura, de kizárólag az adott ügyfélről.

Hogyan varrjunk Azbesztkabát?

Az informatikában egyre általánosabbá válik az üldözési mánia. Sokszor nem tiszta, hogy mitől félünk, mennyire félünk, egy azonban biztos: félünk. A számítógépet otthon használók is tudnak már a veszélyekről, és a világháló számos olyan fórumot kínál, amely tájékoztat a rendszerre leselkedő veszélyekről, és megoldásokat kínál. Ezen megoldások összefoglaló neve (personal firewall) azbesztkabát, és számtalan változata létezik. Lényege, hogy rendszerünk kap egy olyan tűzfalat, amely nem egy mögöttes hálózatot véd, hanem saját gépünket. Megvédi viselőjét – innen az elnevezés. A tapasztalatlanabb Linux-felhasználók bizonyos szempontból rosszabb helyzetben vannak, mint más operációs rendszereket használó társaik. A nagyobb tapasztalattal rendelkezők hajlamosak a kérdést a következő kijelentéssel elintézni: Nincs szükség azbesztkabát fejlesztésére, itt van a csomagszűrő. Ez a kijelentés igaz ugyan, de két dolgot nem vesz figyelembe. Egyrészt a csomagszűrő helyes beállítása némi tapasz-

talatot kíván, másrészt csak a kívülről induló támadások ellen nyújt védelmet. Nem segít akkor, ha a rendszert szabályos forgalomnak álcázva támadják meg. Például ha egy webkiszolgáló azután, hogy arra valaki a rendszerről csatlakozott, visszatámad. Ez a HP szempontjából szabályos, engedélyezett kapcsolatnak minősül, így nem szűrhető ki pusztán csomagszűrő segítségével. Hajlamosak lennénk azt gondolni, hogy egy ilyen támadást szinte lehetetlen kivitelezni, de sajnos nem ez a helyzet. Sok olyan támadást tartanak számon, amit trójai kiszolgálók követtek el a mit sem sejtő ügyfelek ellen.
Az azbesztkabát határvédelmi politikája (AHP) egy hétköznapi ember felhasználási szokásaihoz lett idomítva. Ettől szükség esetén természetesen el lehet térni, ehhez azonban célszerű elolvasni legalább az IPCHAINS-HOWTO-t [1.] [2.], és segítséget kérni tapasztaltabbaktól. Bátrabbak kezdhetik rögtön az engedélyezni kívánt protokoll RFC-jének elolvasásával és a **tcpdump** vagy az **ethereal** programok lehetőségeinek tanulmányozásával. Segítségükkel figyelni lehet a hálózati forgalmat, és meg lehet állapítani, hogy az adott protokoll hogyan engedhető át a lehető legkisebb lyuk megnyitásával.

Az engedélyezett forgalom

Most nézzük meg röviden, mi miért van a példa határvédelmi tervben. Mit szeretne egy átlagos otthoni felhasználó csinálni a nagy Interneten? Lássuk:

- webböngészés; ha webezne, akkor használni akarja az Interneten lévő rendszereket és szolgáltatójának webgyorstárát (webcache).
- ftp-állományok átvitele; ha ftp-zne, ezt megteheti szolgáltatójának webgyorstárán keresztül – így azonban csak letölteni tud (ekkor ehhez nem kell újabb beállítás), vagy közvetlenül, ekkor azonban némi terhet vesz a vállára (ezt a problémát jelenleg nem vesszük figyelembe, de az ip-filter későbbi tárgyalásakor megoldjuk).
- levelezés; ha levelezne, akkor a leveleit le akarja tölteni levelezési kiszolgálójáról (POP server), küldeni pedig postakiszolgálóján (SMTP server) keresztül fog.
- egyéb szükséges szolgáltatások; minden szolgáltatás igénybevételéhez szüksége lesz az Internet névfeloldásának használatára (DNS-kiszolgálók), és ha rendszerének óráját az Internet atomóráihoz akarja hangolni, akkor szüksége lesz az időszolgáltatók (ntp servers) elérésére is.

Azbesztkabátunk mindezt a lehetőséget megadja. Mitől szeretné megvédeni magát egy átlagos Internet-felhasználó? Nem szeretné, ha illetéktelenek lépnének be a gépére, vagy használnák valamire, amit ő nem szeretett volna. A rendszerek telepítőjének jelentős része azonban olyan szolgáltatóprogramokat is telepít, melyre valójában nincs szüksége, és aminek a hibájából azonban gyakran lehetségessé válik a rendszerre való behatolás. Mivel az AHP-ben megengedett forgalmak között nincs a rendszerbe befelé irányuló forgalom, így annak helyes megvalósítása esetén hiába van hibás kiszolgáló a gépre telepítve, a csibészek nem tudnak azon keresztül behatolni. Jelen cikk tartalmaz egy csomagszűrő beállítást, ami kielégíti a példa HP követelményeit (*1. lista*). A cikk megírása idején a 2.4-es rendszermagsorozat még nem mondható megbízhatónak, így a beállítások a 2.2-es sorozat csomagszűrő rendszeréhez megfelelőek. A beállításban szereplő `x.y.z` kezdetű példa címeket természetesen le kell cserélni a szolgáltató valós levelező, posta és webgyorstár kiszolgálójának hálózati címére. A `__modemip__` betűsorozatot a beállítóállományt betöltő parancsfájl fogja kicserélni a modem pillanatnyilag érvényes hálózati címére, mivel ez általában behívásonként változik. A *2.* és *3. lista* két egyszerű parancsfájlt tartalmaz, melyek feladata, hogy a csomagszűrőt élesítsék, illetve kikapcsolják. Ha a két parancsfájlt a megadott könyvtárba helyezzük el, és nem feledkezünk meg a futtathatóvá tételükről, akkor a védelem tárcsázás után önműködően élesedik, a kapcsolat lezárultakor pedig leáll. Ez igaz a Debian Potato rendszerre, más Linux-változatokban azonban más lehet a helyzet. Ennek kiderítését az olvasóra bízzuk.
A csomagszűrő beállításaiban néhány dolgot érdemes megmagyarázni. Ha valami az alábbi bekezdésben nem világos, akkor javasolt elolvasni az ipchains leírását [2.]. Ennek ismertetése ezen cikk kereteit meghaladja, ezenkívül található hozzá jól érthető magyar leírás is. Az egyes lábakra (csatoló) érkező forgalmat célszerű különválasztani, ezért használjuk a modem_be láncot (chain). Ennek a módszernek akkor fogjuk igazán hasznát látni, ha egy 12 lábú tűzfalat kell beállítanunk, és a csomagszűrő valahol hibázik. Itt a hibát egyszerűen megtalálni már csak akkor lehet, ha a csomagokat döntés előtt gondosan osztályoztuk.
Ha egy postakiszolgáló felé kapcsolatot kezdeményezünk, akkor az esetenként az auth (113-as) kapun át visszanyit a gépünk felé, a levél feladójának kiderítésére. Mivel a levelezőrendszerek e szolgáltatás nélkül is működnek, ezt mi tiltjuk. Van azonban egy kis bökkenő. Mivel mi minden bejövő SYN-es csomagot válasz nélkül a földre dobunk (a DENY célra), így a postakiszolgáló azt hiheti, hogy a válasz érkezik, csak még nem ért oda. Ebben a helyzetben ezért használunk REJECT célt, így a másik rendszer azonnal visszakap egy ICMP csomagot, amely tájékoztatja, hogy az adott kapu nem érhető el. Így a levél továbbítását azonnal megkezdhetjük. Ha ezt nem használjuk, akkor a kiszolgáló rendszer 90 másodpercig várni fogja a választ, mielőtt a levéltovábbítást megkezdhetnénk.
A modem_be láncon belül, ha a csomag valamelyik kitételnek megfelel, akkor továbbítását elfogadjuk, ha nem, akkor végül beleesik a „minden más tilos” szabályba. Amint láttuk, a csomagok érvényességét az input láncból kiindulva vizsgáltuk meg. Amennyiben a rendszeren *kizárólag* átmenő csomagok vannak, akkor szokás még a forward láncban megadni a szabályokat, de ez igen ritka. Ki lehet alakítani teljesen egyéni megoldást is, de a tapasztalatokat nem figyelembe venni a biztonságmódszertanban nem túl hálás hozzáállás.

A csomagszűrő magasabb szintű beállítására még visszatérünk, mikor a hivatalos megbízható rendszermag nem csak nevében lesz megbízható. Akkor már az ip-filter lehetőségeivel ismerkedünk meg, ami a rendszermag új nemzedékének csomagszűrője. A következő alkalommal Bozó megteszi azt, amit már oly régen szeretne, jól beolvas a VLAN-nak. Szóval a hálózatok alacsony szintű támadásairól ejtünk pár keresetlen szót.

Kapcsolódó címek
[1.] http://www.math.bme.hu/LDP/HOWTO/IPCHAINS-HOWTO.html
[2.] http://linux.hu.rulez.org/ipchains

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.

Hálózati határvédelmi terv

A rendszer fizikai csatolói

a csatoló logikai neve	a csatoló fizikai neve	a csatoló hálózati címe	egyéb megjegyzés
if_local	lo	127.0.0.0/8	
if_internet	ppp0	interIP/interMASK	gw interGW

A rendszerben ismert hálózatok

logikai megnevezés	hálózati cím	a csatoló neve
helyi_h	127.0.0.0/8	lo
modem	x.z.y.b1	ppp0
internet	0.0.0.0/0	ppp0
out_mail	x.y.z.s1	ppp0
out_cache	x.y.z.s2	ppp0
out_domain	x.y.z.s3	ppp0

A rendszerben engedélyezett hálózati forgalom

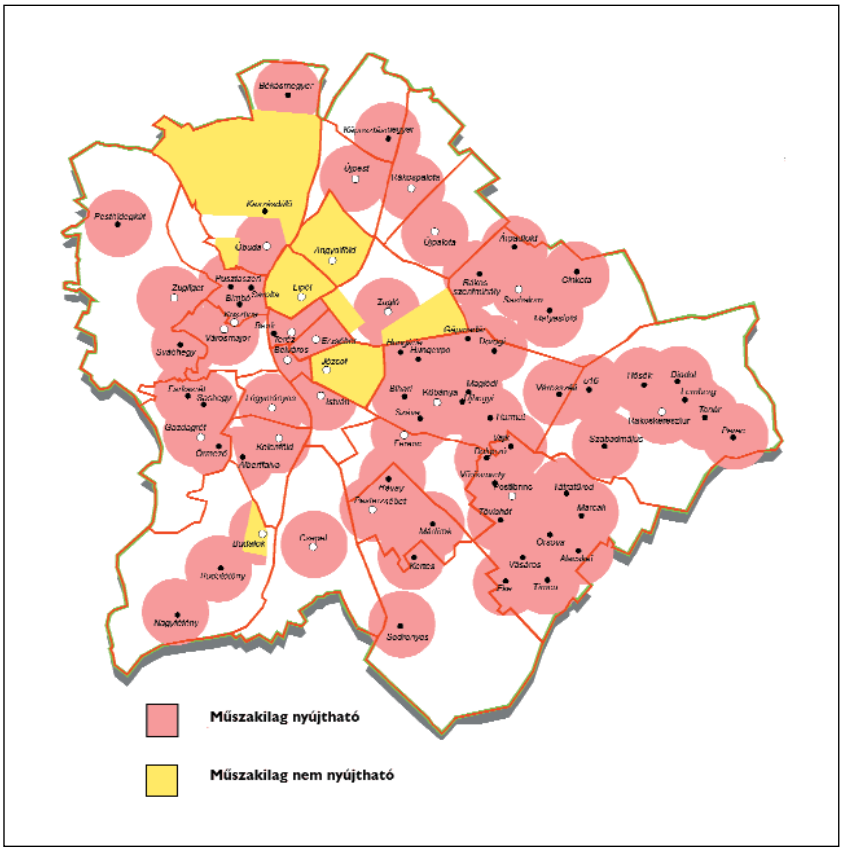
szabály	forrás	cél	protokoll	forráskapu	célkapu	művelet
1.	helyi_h	helyi_h	*	*	*	ACCEPT
2.	modem	out_domain	UDP	>1023	53/domain	ACCEPT
3.	modem	internet	UDP	123/ntp	123/ntp	ACCEPT
4.	modem	out_mail	TCP	>1023	25/smtp	ACCEPT
5.	modem	out_mail	TCP	>1023	110/pop3	ACCEPT
6.	modem	internet	TCP	>1023	80/www	ACCEPT
7.	modem	internet	TCP	>1023	443/https	ACCEPT
8.	modem	out_cache	TCP	>1023	3128	ACCEPT
9.	modem	internet	TCP	>1023	22/ssh	ACCEPT
10.	modem	internet	ICMP	3/dest-unreach	*	ACCEPT
11.	*	*	*	*	*	DENY

A széles sávú internet-hozzáférés

Áttekintés a Magyarországon elérhető gyors internetkapcsolatokról és szolgáltatásokról.

Cikksorozatomban arra a nem kis feladatra vállalkozom, hogy egy csokorba gyűjtöm a hazánkban elérhető kapcsolódási szolgáltatásokat. A lehetőségeket igyekeztem úgy összeválogatni, hogy az ismertetések az otthoni és üzleti felhasználók számára egyaránt hasznos segítségül szolgáljanak. „Ne felejtsd el, hogy Magyaroszágon vagy…” – szoktuk oly sokat hangoztatni, ha az kerül szóba, hogy egy adott területen mekkora lemaradás érzékelhető hazánk és a fejlettebb országok között. Az internetezők különösen hajlamosak ehhez hasonló siránkozásra, tegyük hozzá, nem éppen ok nélkül. A magyarországi felhasználók túlnyomó többsége a mai napig egy szál modemmel felfegyverkezve lép az Internet csatamezőjére, és bizonyára minden érintett tisztában van azzal, hogy ez a létező leglassabb megoldás. Még egy 56 K-s modem sem képes másodpercenként 5-6 kilobájtnál több adat átvitelére, aki pedig töltött már le ilyen sebességgel mondjuk egy 20 megabájtos fájlt, már érti, hogy miért szántam el magam e cikk megírására...

Az Internettel kicsit is komolyabban foglalkozók számára egyértelmű, hogy valami egészen másra lenne szükség a kényelmes internetezéshez. Az analóg behívás másik hátránya – és ebben, sajnos nagyobb testvérével, az ISDN-nel is osztozik –, hogy a havidíjon túl a használat után is fizetnünk kell, ez az összeg pedig telefonszáma formájában jelentkezik. A Matáv kedvezményes időszakra vonatkozó tarifája – a hétköznap 18–07 óráig, illetve hétvégén 15–07 óráig indított helyi hívások díja legfeljebb nettó 150 forint lehet – az otthoni felhasználók bajait valamelyest enyhíti. Ez azonban szinte semmit sem jelent a cégeknek, hiszen éppen a csúcsidőszakban szeretnék jobban kihasználni az Internet nyújtotta előnyöket. Nekik más megoldások, például bérelt vonal, ADSL, tévémodem, mikrohullámú vagy műholdas kapcsolat után kell nézniük, ha a háló használatát valóban költségtakarékosan és hatékonyan szeretnék megoldani. Mindemellett az egyéni felhasználókban is jogosan merül fel az igény a gyorsabb, korlátlan hozzáférésre. Először tehát vizsgáljuk meg, milyen lehetőségek közül választhatnak ők (külön megemlítem, ha egy adott szolgáltatás cégek számára is megfelelő választásnak tűnik).



A MatávNet ADSL szolgáltatása jelenleg csak Budapest egy részén érhető el

ISDN

Az első szóba jöhető megoldás az ISDN, azaz a digitális telefonvonal kiépítése, illetve az ezen történő adatátvitel. ISDN-vonalat jelenleg csak a Matáv szerel, ISDN-alapú előfizetést azonban szinte az összes nagyobb szolgáltatónál vásárolhatunk, az analóg csatlakozással egyező áron. Az ISDN kettő, másodpercenként 64 kilobit átvitelére képes, adattovábbításra szolgáló „B” csatornával (ezért szokás ISDN2-nek is nevezni) és egy, a kiegészítő adatforgalom által igénybe vett „D” csatornával bír, melynek sebessége 16 kilobit másodpercenként. Az előfizetést megvásárolhatjuk egy csatornára, ekkor a legnagyobb elérhető sebesség 7 kilobájt/másodperc, illetve két csatornára is, így 14 kilobájttal „száguldanak” az adatok. Az analóg vonal ISDN-re történő átépítése után 4875 forintot kell fizetnünk a vonalért. Ehhez jön még az internetelőfizetés ára, ez megközelítőleg 4-5000 forint csatornánként. A 128 kilobites kettős vonalért tehát kétszer ennyit kell fizetnünk. És természetesen ott van még a forgalmi díj, ami a MindenkiNek csomag megszűnésével még komolyabb terheket ró a felhasználókra. Akinek tehát van havonta úgy 20 000 forintja a dologra és beéri a modem teljesítményének mintegy háromszorosát jelentő 14 kilobájt másodpercenkénti le- és feltöltési sebességgel, máris válthat az ISDN-re.

Tévémodem

A kábeltévé-hálózatokon keresztül történő internetezés nálunk sem újdonság: sok helyen a feltételek már a kilencvenes évek elején is adottak voltak ehhez. A régi soros rendszer jellemzője, hogy a kábel csak viszonylag alacsony hullámhosszú jeleket képes átvinni, és csak a végpontok felé – kétirányú kapcsolatra éppen ezért nem volt lehetőség. Az internetezéshez viszont mindenképpen szükséges a

kétirányú adatforgalom. Az újabb csillagpontos hálózatok építése területenként változó ütemben halad; több vidéki nagyvárosban, például Szegeden, Nyíregyházán szinte teljes egészében kiépült, és Budapesten is a legtöbb kerületben már találunk ilyet (külön megjegyezném, hogy a VIII. kerületben nincs, rúgja meg a ló – *főszerk*). A csillagpontos kiépítésű hálózat képes 5-10 Mbit/másodperc sebességű adatátvitelre is, de ezt nagymértékben korlátozza a kimenő vonalak, a szolgáltató hálózatát a külvilággal összekötő kábelek sávszélessége. A helyzet annyival összetettebb a telefonvonalat használó szolgáltatásoknál, hogy a kábeltévé-hálózatok több, kisebb-nagyobb cég kezében vannak, így a csillagpontossá való átépítés és az internetszolgáltatás beindítása jóval lassabban halad.

Az egyik legelső kábeltévés internetszolgáltató a TvNet Kft., szolgáltatásuk jelenleg Budapest V., IX., XIII., XIV., XIX. és XX. kerületében érhető el, saját kábelhálózatuk azonban nincs, a szolgáltató kábeltévés társaságoktól bérel vonalakat. Egy számítógép csatlakoztatása 48 000 forintba kerül. A napi 24 órában, korlátlanul használható szolgáltatásért 10 000 forint havi átalánydíjat kell fizetnünk, tehát forgalmi díjat nem számolnak fel. A fentiek magánszemélyek esetén bruttó, céges előfizetőknél nettó árakat jelentenek. Ez így elég szépnek tűnik, a tvnetes előfizetésnek is vannak azonban hátrányai. Belföldre hatalmas, akár 2-300 kilobájt másodpercenkénti sebesség is elérhető, mindkét irányban, a BIX és a TvNet hálózata közötti 100 megabites sávszélességnek köszönhetően. A külföldi vonalakkal kicsit más a helyzet. A cég sokáig az EuroWebtől bérelt egy 2 megabites vonalat, ezt később bővítették 4 Mbitre, majd kiegészítették a PSINet 4 Mbites vonalával. Így összesen 8 Mbites sávszélesség vezet külföldre, ebből azonban egy 4 Mbites darabot a kiemelt (úgynevezett „VIP") előfizetők számára tartanak fenn, akik (kapcsolatuk típusától függően) havi átalányért használhatják ezt a többletet (a pontos árakat a cég honlapján megtaláljuk). A rendszer teljesítménye egyébként kerületenként igen változó: sokáig a XIII. kerületben volt a leggyorsabb, mostanában viszont arrafelé is gondok jelentkeznek. Sok helyütt nem ritkaság a hosszabb-rövidebb ideig tartó szolgáltatáskimaradás sem, ennek gyakoriságát azonban folyamatos fejlesztésekkel igyekeznek egy elfogadható szint alá szorítani. A tényeket összefoglalva elmondható, hogy a TvNet megbízható szolgáltatásával úttörő szerepet vállalt a kábeltévés Internet budapesti elterjesztésében.

Jelenleg a UPC (United Pan-European Communications) a legnagyobb ügyfélkörrel bíró kábeltévés internetszolgáltató Magyarországon. Kétféle szolgáltatásuk létezik, az egyik Broadband, a másik pedig Chello névre hallgat. A Broadband kapcsolatokat először vidéken, például Nyíregyházán, Miskolcon, Szolnokon, Nagykanizsán vezették be, erre az ottani kábeltévés internetszolgáltatók (Szabinet stb.) felvásárlásával kerülhetett sor. A Broadbandot tulajdonképpen a Chello próbaüzemeként képzelhetjük el. Több Broadband csomag is létezik, melyek az egy hónapban letölthető adatmennyiségben különböznek. A legkisebb, 7000 forintba kerülő változatban 1 GB, a legdrágább, 40 000 forintos előfizetésnél 15 GB a havi letöltési korlát. A sebesség minden csomag esetében 300 kbit/másodperc letöltés, 64 kbit/másodperc feltöltés. A letöltési korlátot túllépő összes előfizető egyetlen 64 kbit/másodperces vonalra kerül, a következő hónap elsejéig. A büntetővonal éppen csak arra elég, hogy a felhasználó letölthesse a leveleit.

A Broadband szolgáltatást folyamatosan fölváltja a Chello, mely egyelőre Budapest négy kerületében: az I., a II., a XI. ingyenes részein és a XIII. kerületben, valamint Miskolcon érhető el. A Chello egy jóval nagyobb teljesítményű, rendkívül ígéretes szolgáltatás. Díjcsomagok nincsenek, és 2001. december 31-éig a 25 000 forintos indítási díjat is elengedik. Így a fentebb említett területek csillagpontos és visszirányúsított kábelhálózattal bíró részein máris előfizethetünk a Chellóra, mely 512 kbit/másodperces letöltési és a kétcsatornás ISDN2 teljesítményénél egykategóriás 128 kbit/másodperces feltöltési sebességet tesz lehetővé. Mindezért havi bruttó 10 900 forintot kell fizetnie az egyéni felhasználóknak. Megemlíteném, hogy a szerződés tiltja a végpontokon a web- és egyéb kiszolgálók elhelyezését, ennek ellenére a Chello azon cégeknek is hatalmas könnyítést jelenthet, melyek a szolgáltatójuk géptermében elhelyezett kiszolgálójukat analóg vagy ISDN vonalon keresztül kénytelenek elérni.

A Chello bevezetése körül rengeteg téves adat, innen-onnan hallott értesülés, pletyka kapott szárnyra. Az Index és más portálok fórumaiban állandó vita tárgya az esetleg létező letöltési korlát. Többektől hallani, hogy a Chello-felhasználók 5 GB-ot tölthetnek le havonta, majd ennek elérése után (a Broadbandhez hasonlóan) egyetlen 64 kbit/másodperc sebességű vonalra kerülnek a szabályszegők. Azt is rebesgetik, hogy e korlátozás nincsen világosan megfogalmazva a szerződésben, csak egy bizonyos „Fair Use Policy" (a megfelelő használat elve) nevű kitétel szerepel benne, mely talán burkoltan jelenthet valamiféle korlátozást.

Az valóban kissé furcsa, hogy a lassan két hónapja és több száz előfizetővel működő Chello honlapján nemhogy a „megfelelő használatról", de még a szolgáltatás tulajdonságairól, például a sebességről és az árról sem lehet találni egyetlen árva szót sem. Csak próbaképpen ellenőriztem az osztrák Chello honlapját is ➲ www.chello.at, ott minden említett tájékoztatás megtalálható, ráadásul korlátozásról sem esik szó.

A dolog tisztázása érdekében megkerestem *Hagymásy Andrást*, a UPC Magyarország kommunikációs igazgatóját. Kérdésemre, hogy létezik-e a hirdetésekben korlátlannak hirdetett Chello esetében havi letöltési határ, nemmel válaszolt. A „megfelelő használat" felől is érdeklődtem. Amint megtudtam, ez az elv a Chello esetében nem került bevezetésre. Hagymásy András azt is elmondta, hogy a Chellón jelenleg „félhivatalosan" próbaüzem folyik. Ha a rendszer sávszélességével kapcsolatban nem lépnek föl gondok, azaz a napi több száz megabájtot letöltő emberek nem foglalják le túlságosan a hálózatot az átlagos, levelezgető és Webet böngésző felhasználóktól, akkor a Chellón soha nem is kerül bevezetésre a letöltési korlát.

A kábeltelevíziós Internetről összességében annyit, hogy nagyszerű megoldást jelenthet az otthoni és a vállalati felhasználók számára egyaránt. A napi 24 órában, forgalmi díj és korlátozás nélkül elérhető Internet minden bizonnyal egyre több felhasználót fog elcsábítani a telefontársaságoktól. Utóbbiak, pontosabban a Matáv egyetlen visszavágási lehetőséggel bír, és ezt ADSL-nek hívják.

ADSL

Jó néhány évvel ezelőtt a nagyobb telefontársaságok elkezdték az aszinkron digitális előfizetői vonalak (ADSL) kifejlesztését, ezzel próbáltak versenybe szállni a kábeltévés társaságok által kínált rengeteg szolgáltatással. Az eljárás lényege, hogy a sodrott réz érpáron a beszédhez használt hullámhossznál jóval magasabb tartományban aszinkron adatátvitelt tesz lehetővé. Az „aszinkron" itt azt jelenti, hogy a letöltési sebesség nagyobb, mint a feltöltési. Az ADSL-re történő áttéréshez a sodrott réz érpárral csatlakozó felhasználók esetében nincsen szükség a vonal átépítésére. A Matáv ennek ellenére eleinte csak meglévő ISDN vonalon volt hajlandó ADSL-szolgáltatást nyújtani, február közepétől azonban már analóg vonalon is működik a szolgáltatás, amennyiben ezt a lakásban lévő rézkábel minősége lehetővé teszi. Február közepe hatalmas árzuhanást is hozott: a kezdeti időszakban a Matáv és a MatávNet havi nettó 160 ezer forintot kért a korlátlan ADSL-hozzáférésért, és ez az ár most bruttó 12 900 és 18 900 forint között mozog, de egyelőre csak magánszemélyek fizethetnek elő rá. A MatávNet sajtófőnöke, *Pohly Ferenc* nem hivatalos közlése szerint az üzleti csoportok meghirdetését márciusra tervezik.

A MatávNet az ADSL csomagban 384 kbit/másodperces letöltési, és 64 kbit/másodperces feltöltési sebességet nyújt, ez azonban annak függvényében változhat, hogy a végpont (tehát a gépünk) milyen

messze található a legközelebbi ADSL-központtól. A szolgáltatást megrendelhetjük határozatlan időre, illetve 1 vagy 2 évre. Határozatlan idejű szerződéskötés esetén 59 900 forint egyszeri összeget kell kifizetnünk csatlakozási díjként, a havi díj pedig 18 900 forint. Egy-és kétéves szerződésnél a csatlakozási díj 29 900 forint, a havi előfizetési díj egyéves szerződésnél 14 900, kétéves esetén pedig 12 900 forint. Megjegyzendő, hogy ezek mellett egy hálózati csatolót is vásárolnunk kell, ami egyszeri hétezer forint körüli összeget jelent. Nyilvánvaló, hogy anyagilag az egy-két éves szerződéskötés éri meg a legjobban, de mi történik akkor, ha egy év múlva sokkal olcsóbb és gyorsabb szolgáltatásokat indít a MatávNet vagy bármely más internetszolgáltató? Ha ilyen esetben szabadulni szeretnénk a szolgáltatástól, akkor visszamenőleg ki kell fizetnünk az akciós ár és a listaár közti különbözetet. Sok ISDN-előfizető számára az ADSL megjelenése éppen ezért igencsak bosszantó volt. Mindezek ellenére az ADSL egy nagyszerű megoldás, mely végre versenyhelyzetet teremtett – igaz, csak Budapesten.

Pohly Ferenc az Index fórumán azt is közzétette, hogy valószínűleg még idén megjelennek a nagyobb (768 kbit-es) ADSL-csomagok – bízzunk benne, hogy így lesz.

Műholdas kapcsolat?

Elsőként az Astra műholdon keresztül vált lehetővé a digitális adatátvitel. Néhány éve műholdon keresztül is internetezhetünk, a dolognak azonban van egy elvi korlátja, ugyanis az otthon felszerelhető olcsó parabolaantennák csak vételre képesek, adatküldésre nem.

A kétirányú kapcsolat a Weyland-Yutani révén már Magyarországon is megvalósítható, azonban a szükséges felszerelés ára és az előfizetési díj még megfizethetetlenül magas. Az internetezéshez kétirányú kapcsolat kell: a csak adatfogadásra használható műholdas előfizetés mellé ezért egy földi (modemes, ISDN, kábeltévés stb.) vonalat is igénybe kell vennünk.

Emellett mindenképpen szükség van egy DVB (Digital Video Broadcast) kártyára, mely többféle változatban létezik. Az olcsóbbak – 35 ezer forint körül – csak az antennáról bejövő jel fogadására alkalmasak, a csúcskategóriás, hang- és képkimenettel ellátott termékekkel viszont műholdas adásokat is nézhetünk. Ez utóbbiak úgy ötvenezer forintba kerülnek, és csatlakoztathatunk hozzájuk egy kódkártya fogadására képes eszközt 25 ezer forintért, így már a kódolt adásokat is élvezhetjük. A helyzet külön bosszantó fricskája, hogy a legtöbb kártyához évek óta „éppen most fejlesztik” a linuxos meghajtókat.

Az első műholdas internetszolgáltató a luxemburgi Europe Online, mely kétféle üzemmódban teszi lehetővé az Internet elérését. Unicast üzemmódban minden egyes bejövő bit a műholdról érkezik, tehát a földi kapcsolaton csak adatküldés folyik. Ekkor a Europe Online proxy-kiszolgálóinak valamelyikéhez kell csatlakoznunk, hiszen a meglátogatott honlapok, FTP-kiszolgálók másképp „nem tudnák”, hogy a modemmel lekért adatokat az antennán keresztül kell továbbítaniuk hozzánk. Ez a felállás 120-400 kbit/másodperces letöltési sebességű kapcsolatot tesz lehetővé.

A másik üzemmód a Multicast, ezzel a földi kapcsolatot mindkét irányban felhasználják. Multicast üzemmódban 2 megabájt másodpercenkénti letöltési sebességet érhetünk el. Ehhez a Europe Online programját kell telepítenünk. A Fazzt elnevezésű alkalmazás a GetRightra és más „okos” FTP-programokra emlékeztet. A programnak megadhatjuk a kiszemelt HTTP- vagy FTP-címeket, majd a kérelem elküldése után a rendszer visszajelez, hogy körülbelül mikorra érkezik meg az ily módon „megrendelt” letöltés a gépünkre. A kérelem leadása után akár bonthatjuk is a modemes kapcsolatot, az antennán keresztül történő letöltéshez erre ugyanis már nincs szükség. A fájlok a megadott időpontban villámgyorsan letöltődnek.

A megrendeléstől a megérkezésig eltelt idő a fájl méretétől függ: a kérelmeket a központi gép ez alapján várakozási sorokba rendezi. A kisebb, 10 Mbájt körüli fájlok egy-két perc alatt megérkeznek, de egy 7-800 megabájtos fájlra valamivel többet kell várnunk – a rendszer terheltségétől függően akár egy-két órát is. Természetesen ez így még mindig jóval gyorsabb minden más kapcsolatnál. A legszebb az egészben, hogy a két üzemmód egyszerre is működik, sőt, egyszerre több Multicast kérelmet is leadhatunk.

Aki ezek alapján arra számít, hogy az egész bizonyára elképzelhetetlenül drága, az most kellemesen fog csalódni. A Europe Online előfizetési díja havi nettó 3840 forint (az Infotechna nevű cégnél), egyéves előre fizetés esetén pedig két hónapot ingyen kapunk. Magyarországon több cégnél is előfizethetünk műholdas internet-hozzáférésre, és nem csak a Europe Online szolgáltatásaira. Az EOL-lal egyébként mostanában gondok vannak: a cég inkább a Multicastra összpontosít, nemrégiben pedig egyoldalúan felmondta a szerződést minden viszonteladójával. Ennek ellenére a műholdas Internet szédületesen jó dolognak ígérkezik, főleg ha azt vesszük figyelembe, hogy a legkisebb falvakban is tökéletesen használható, és ez semmilyen más széles sávú kapcsolatról nem mondható el. A műholdas Internet a fentebb említett többi szolgáltatással egyetemben nem csak a Windows-felhasználók kiváltsága: több honlapról is letölthető a kártyához és az adatátvitelhez szükséges Linux alatt működő meghajtó.

Ki tudja, merre...?

A fentiekből kiderülhetett, hogy bár az otthoni felhasználók által is megfizethető magyarországi széles sávú internetelérés még fejlődőben van, már most is sok jól működő szolgáltatás közül válogathatunk, és lehetőségeink a jövőben egyre bővülnek. Én bízom abban, hogy a Matáv-monopólium lejártával végre igazi árverseny alakulhat ki a magyar piacon is, az ADSL-lel foglalkozó telefonos cégek és a kábeltévés internetszolgáltatásban érdekelt vállalatok pedig egyre alacsonyabb áraikkal igyekeznek majd elhódítani egymástól a felhasználókat. Addig azonban várnunk kell még egy gyötrelmes évet… Következő számunkban a vállalkozások számára szóba jöhető megoldásokról ejtünk szót: bérelt vonal, mikrohullámú kapcsolat, nagyobb ADSL-csomagok. Addig is kellemes böngészést és jó nagy sávszélességet mindenkinek!

Borai János (borai.janos@linuxvilag.hu) az ELTE amerikanisztika szakos hallgatója, 1997-ben ismerkedett meg a Linuxszal. Szabadidejében zenél, jelenleg egy otthoni stúdió kiépítésén fáradozik.

További kapcsolatok

ISDN, ADSL

- ➲ http://www.isdn.matav.hu/
- ➲ http://www.matav.hu/
- ➲ http://www.matavnet.hu/
- ➲ http://www.datanet.hu/

Kábelnet

- ➲ http://www..hu/
- ➲ http://www.broadband.hu/
- ➲ http://www.chello.hu/
- ➲ http://www.matavkabel.hu/

Műholdas kapcsolat

- ➲ http://www.europeonline.hu/
- ➲ http://www.infotechna.hu/
- ➲ http://www.weyland-yutani.hu/

Az OpenSSH száz meg egy előnye (2. rész)

Nézzük meg, hogy még mi mindenre jó az SSH!

A legtöbb SSH-felhasználó nem kerül kapcsolatba a két legegyszerűbb szolgáltatással: a titkosított távoli héjprogrammal és a titkosított fájlátvitellel. Ezt vesztük ki az előző hónapban. Ez eddig rendben is volna. Végül is minek megtanulni olyan dolgokat, amelyeket soha nem használunk. Önképző olvasóink közt azonban minden bizonnyal akad olyan is, aki az SSH megmaradt 99 előnyéből is értékelni tudna valamit. Úgyhogy nézzünk is körül az SSH, és különösképp az OpenSSH igazán hasznos tulajdonságai közt.

Engedjenek meg egy megjegyzést: ebben a cikkben az általános értelemben vett Secure Shellre, azaz „Secure Shell Systemre” az „SSH” kifejezéssel hivatkozom.

Nyilvános kulcsú titkosítás

A nyilvános kulcsú titkosítás (Public Key Criptography) teljes leírása egyszerűen nem férne el egy olyan cikkben, melynek egyébként az OpenSSH-ról kellene szólnia. Ha teljesen ismeretlen ez a terület, ajánlom az RSA Crypto FAQ-t, vagy talán még ennél is jobb forrásként *Bruce Schneier Applied Cryptography* című kitűnő könyvét. Számunkra most elegendő lesz annyit tudni, hogy a nyilvános kulcssémában minden felhasználónak szüksége van egy kulcspárra. A titkos kulcsot (private key) használhatjuk digitális aláírás készítéséhez és a kapott titkosított anyagok visszafejtéséhez. Levelező partnereink a nyilvános kulcsunk (public key) segítségével ellenőrizhetik az állítólag általunk aláírt üzeneteket, illetve ez alapján kódolhatják az üzeneteket, ha azt szeretnék, hogy csak mi olvashassuk azokat. Az *1. ábra* alján láthatjuk, miképpen lehet a két felhasználói kulcspárt használni az egyik felhasználótól a másikhoz érkezett levél hitelesítésére, titkosítására, visszakódolására és ellenőrzésére. Figyeljük meg, hogy Bob és Alice nyilvános kulcsa is jelen van a levelezőtárs gépén, titkos kulcsaikat viszont mind a ketten biztonságos helyen tartják.

Ahogy láthatjuk, az üzenet útja során négy fontos állomást különböztethetünk meg: 1. Bob hitelesíti az üzenetet saját titkos kulcsa segítségével; 2. ezután Bob titkosítja az üzenetet Alice nyilvános kulcsával; (Figyelem: eltekintve attól, hogy Bob megtart leveléből egy másolatot, még ő sem tudja visszakódolni az üzenetet – ezt csakis Alice teheti meg.) 3. Alice megkapja az üzenetet, és visszakódolja a saját titkos kulcsa segítségével; 4. végül Alice Bob nyilvános kulcsát használva ellenőrzi, hogy az üzenetet valóban Bob titkos kódjával hitelesítették-e.

Az olyan tömbkódolásokhoz viszonyítva, mint amilyen az IDEA vagy a blowfish, ahol ugyanazt a kulcsot használják titkosításra és visszakódolásra egyaránt, ez talán kissé csavarosnak tűnhet. A tömbkódolásokkal ellentétben, amelyek esetében a kémkedéstől vagy más támadástól mentes, biztonságos kulcscseréknek kell lehetővé tenni mindkét fél számára a kulcs megszerezését, a nyilvános kulcsokat használó titkosítási rendszert könnyebb biztonságosan használni.

Ez azért lehetséges, mert a PK (Public Key) séma szerint úgy küldhetnek a felek üzeneteket egymásnak, hogy előtte semmiféle titkos kódot vagy hasonlót nem kell cserélniük. Egyetlen hátulütője van a dolognak: a nyilvános kulcsokon alapuló eljárások lassabbak és sokkal számításigényesebbek, mint más titkosítási eljárásosztályok, például a tömbkódolások és a folyamkódolások (stream chiper), például a 3DES és az RC4. Történetesen azonban a PK kódolás kitűnően használható olyan biztonságos kulcsok előállítására, amit aztán más eljárásokban lehet felhasználni.

A gyakorlatban a PK titkosítást sűrűn használják azonosításra („Tényleg te vagy?”) és kulcsegyeztetésre („Melyik 3DES kulccsal kódoljuk a további adatokat?”), de annál ritkábban a teljes adatfolyam vagy fájlok tömeges kódolásához. Ez a helyzet mind az SSL, mind az SSH esetében.

Magasabb szintű SSH-elmélet: Hogyan használja az SSH a PK titkosítást

A kulcsegyeztetés az egyik legelső dolog, mely akármelyik SSH-ügyletben végbemegy, és ennek köszönhetően lehetségessé válik egy viszonylag gyenge azonosítási módszer (felhasználónév és jelszó) használata. A Diffie–Hellman Kulcscsere Protokoll működése igen bonyolult, és messze meghaladja e cikk kereteit (További részletekért látogassunk el a ➲ http://www.ietf.org/ címre, ahol elolvashatjuk a „draft-ietf-secsh-transport-07.txt” című vázlatot.) Elég annyit tudni, hogy ennek az egész nagy prímszámkavalkádnak a kapcsolatkulcs (session key) lesz az eredménye. Ezt mindkét fél ismeri, de ez ténylegesen nem kerül át az eddig titkosítatlan kapcsolaton keresztül. Minden további csomag adatmezője ezzel a kapcsolatkulccsal lesz titkosítva, a két gép által közösen választott tömbkódolás szerint, átlátszóan, de az adott ssh-folyamat fordítása és beállítása alapján. Többnyire a következő kódolások valamelyike használatos: Triple-DES (3DES), blowfish vagy IDEA. Maga az azonosítás csak a kapcsolat titkosítása után kezdődhet el. Mielőtt még belemerülnénk az RSA/DSA azonosítás rejtelmeibe, álljunk meg egy pillanatra és nézzük meg, miképp lehet a kulcsegyeztetés átlátszó, feltételezve, hogy PK-titkosítást használ, és mint általában, a titkos kulcsok jelszóvédettek. Az SSH két különböző kulcspárt használ: a gépkulcsokat és a felhasználói kulcsokat. A gépkulcs egy különleges kulcspár, amihez nincsen jelszó rendelve. Mivel ezeket jelszó begépelése nélkül is bárki használhatja, az SSH egyeztetni tudja a kulcsokat és a felhasználó számára teljesen átlátszó, titkosított kapcsolatokat állíthat fel. Az SSH-telepítés része a gépkulcs (-pár) készítése. A beállításkor készített gépkulcs az adott gépen azonosítás nélkül használható, biztonsági okokból. Mivel a gépkulcs az adott gépet azonosítja és nem az adott felhasználót, minden gépnek csak egy gépkulcsra van szüksége. Megjegyezzük, hogy gépkulcsot minden SSH-t futtató gép használ, függetlenül attól, hogy futtat-e SSH-ügyfélprogramot, például `ssh` héjat, SSH-démont, vagy mindkettőt.

A felhasználói kulcs természetesen az egyes felhasználókhoz van rendelve, és a felhasználó azonosítására szolgál azon a gépen, amivel kapcsolatot kezdeményezett. A legtöbb felhasználói kulcsot az alkalmazás előtt a megfelelő jelszóval ki kell nyitni.

A felhasználói kulcsok biztonságosabb azonosítási eljárást tesznek lehetővé, mint a felhasználónév/jelszó azonosítás (még akkor is, ha minden azonosítás titkosított csatornákon keresztül folyik). Emiatt

alapértelmezés szerint az SSH mindig PK-azonosítást próbál először, és csak ezután tér vissza a felhasználó/jelszó azonosításra.
Amikor elindítjuk az `ssh`-t, az először megnézi a $HOME/.ssh könyvtárunkat, hogy lássa, van-e titkos kulcsunk (`id_dsa` néven). Amennyiben van, a program kérni fogja a kulcs jelszavát, majd a titkos kulcs segítségével készített aláírást a nyilvános kulcsunkkal egyetemben elküldi a távoli kiszolgálónak.
A kiszolgáló ellenőrzi, hogy a nyilvános kulcs engedélyezett-e, azaz jogszerű felhasználóhoz tartozik-e, és mint ilyen, jelen van-e a megfelelő `$HOME/.ssh/authorized_keys2` fájlban. Ha a kulcs engedélyezett, és azonos a kiszolgáló által előzőleg eltárolt másolattal, a kiszolgáló ennek segítségével fogja ellenőrizni, hogy az aláírás ehhez a kulcshoz tartozó nyilvános kulccsal lett-e elkészítve. Ha sikerrel jár, a kiszolgáló engedélyezi a kapcsolatot, esetleg az után, hogy egy felhasználónév/jelszó azonosítást is végrehajt.
(Megjegyzés: a fenti két bekezdés az SSH Protocol v.2 által használt DSA-azonosításra vonatkozik; az RSA-azonosítás valamivel bonyolultabb, de azon kívül, hogy más fájlneveket használ, a felhasználó szemszögéből feladatát azonos az előzővel.)
A PK azonosítás biztonságosabb a felhasználónév/jelszó módszernél, mivel a digitális aláírásokat nem lehet visszafejteni, vagy az előállításukhoz felhasznált titkos kulcsot más módon kikövetkeztetni, még a nyilvános kulcs ismeretében sem. Azáltal, hogy a hálózaton csak digitális aláírásokat és nyilvános kulcsokat küldünk el, biztosíthatjuk, hogy a kémkedő még akkor sem tud elegendő adatot gyűjteni a jogosulatlan bejelentkezéshez, ha a kapcsolatkulcsot valahogy fel is tudná törni.

Beállítások és az RSA-és DSA-azonosítás használata

Rendben. Most már készen állunk arra, hogy saját kulcspár készítésével az ssh-bubusok iskolájában következő osztályába lépjünk. Nézzük, mit kell tennünk.
Elsőként az ügyfélgépen, vagyis azon a gépen, ahol a távoli konzolt szeretnénk majd működtetni, le kell futtatnunk az `ssh-keygen` programot. *(1. lista)* Itt meghatározhatunk pár dolgot: többek közt azt, hogy RSA- vagy DSA-kulcsokat szeretnénk használni, a kulcs hosszát, egy tetszőleges megjegyzésmezőt, a kulcsfájlok nevét és a jelszót – ha akarunk ilyet –, amivel a titkos kulcsok rejtjelezzük.
Most, hogy az RSA-szabadalom lejárt, az eljárás kiválasztása tulajdonképpen tetszőleges. Másrészt viszont, az IETF-hez internetszabvány-ajánlásként elküldött SSH Protocol v.2 DSA rendszerű kulcsokat használ. Ha mindez nem igazán fontos kérdés olvasóink számára, én a DSA-t javaslom. De ha valaki mégis RSA-t kedveli, nyugodtan használhatjuk azt is. A `-d` kapcsoló állítja be a DSA-eljárást, az alapértelmezés ugyanis az RSA.
A kulcshossz már fontosabb jellemző. *Adi Shamir* Twinkle című írásában ismertet egy csak elméletben létező, de megvalósíthatónak tűnő számítógépet, melyet az 512 bitnél rövidebb RSA/DSA-kulcsok nyers erővel (brute force) történő feltörésére lehetne használni, ennek fényében én az 1024 bites kulcsok használatát ajánlom. A 768 is rendben lenne, de nem észrevehetően gyorsabb, mint az 1024. A 2048 viszont egyértelműen kész halál: nem sokkal biztonságosabb, viszont mindent érezhetően lelassít. Az alapértelmezett kulcshossz az 1024, de a `-b` kapcsolót követő számmal más értéket is megadhatunk.
A comment (megjegyzés) mezőt egyik ssh-folyamat sem használja: szigorúan csak a mi kedvünkért létezik. Én általában a helyi levélcímemet írom oda. Ezáltal, ha megtalálom a kulcsot valamely másik rendszerem `authorized_keys` (jogosult kulcsok) fájljai között, legalább tudom, honnan érkezett. A megjegyzéseket a `-C` kapcsolóval adhatjuk meg.
A jelszó és a fájlnév megadható a `-N8` és a `-f` kapcsolókkal, de nem kötelezőek a parancssorban. Ha nincsenek megadva, a program úgyis rákérdez.

Az első listában egy 1024 bit hosszú DSA-kulcspárt készítek, megjegyzésként pedig az mbauer@sprecher.enrgi.com címet adom meg. Az `ssh-keygen`-re bízom, hogy megkérdezze, milyen fájlba mentse a kulcsokat. Ez lesz a titkos kulcs neve. A nyilvános kulcs neve ugyanez lesz, de a végére a .pub kiterjesztés kerül. Ebben a példában elfogadtam az alapértelmezett id_dsa, és így az id_dsa.pub fájlnevet. Szintén az `ssh-keygen`-re hagytam, hogy a jelszót megkérdezze. A csillagsorozat (**********) valójában nem fog megjelenni a képernyőn, csak azért szúrtam be a példába, hogy jelöljem, egy hosszú jelszót gépeltem be, ami nem jelenik meg a képernyőn.

1. ábra Nyilvános kulcsú titkosítás

Egyébként a jelszavak mindent vagy semmit alapon adhatók meg. A jelszavunk lehet üres is, ha az új kulcsot gépkulcsként vagy SSH-t tartalmazó parancsfájlokban akarjuk használni, vagy pedig egy hosszú, nagy- és kisbetűket, számokat és központozást tartalmazó karaktersorozatnak kell lennie. Nem olyan bonyolult, mint amilyennek hangzik. Például egy dalszöveg sora szándékos, de kiszámíthatatlan félregépelésekkel könnyen megjegyezhető, de nehezen kitalálható. Minél inkább véletlenszerű egy jelszó, annál erősebb.
Ez minden, amit az ügyféloldalon tenni kell. Az egyetlen dolog, amire szükségünk van a távoli gépen – ahová erről az ügyfélről szeretnénk belépni –, hogy egy új nyilvános kulcsot helyezzünk a `$HOME/.ssh/authorized_keys2` fájlba, itt a $HOME a saját könyvtárunk elérési útja. Az `authorized_keys2` egy nyilvános kulcsokat tartalmazó lista, minden igen hosszú sorában egy bejegyzéssel, amit a könyvtárat birtokló felhasználó bejelentkezéséhez lehet felhasználni.
Ha szeretnénk eljuttatni a nyilvános kulcsunkat ahhoz a távoli géphez, amin jogosultságunk van, egyszerűen küldjük át a nyilvános kulcsunkat tartalmazó fájlt (a fenti példában `id_dsa.pub`) a távoli gépre, és toldjuk az `authorized_keys2` fájl végéhez. Hogy miképpen szerezzük meg az állományt, az nem igazán számít. Emlékezzünk, ez a nyilvános kulcsunk, így ha egy kukucskáló le is másolná is útközben, akkor se történne semmi. Ha azonban van némi üldözési kényszerünk, egyszerűen írjuk be a `scp ./id_dsa.pub` *`távoligépnév:/saját-könyvtár`* parancsot – miként azt a múlt havi írásunkból már megismerhették. Azután, hogy hozzáadhassuk az `authorized_keys2` fájlhoz, jelentkezzünk be a távoli gépre és gépeljük be:

```
cat id_dsa.pub >> .ssh/authorized_keys2
(feltételezve, hogy a saját könyvtárunkban
vagyunk.)
```

1. lista DSA-kulcspárok készítése

```
mbauer@homebox:~/.ssh > ssh-keygen -d -b 1024
        ➥-C mbauer@homebox.pinheads.com
Generating DSA parameter and key.
Enter file in which to save the key
(/home/mbauer/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again: ******************
Your identification has been saved in
/home/mbauer/.ssh/id_dsa.
Your public key has been saved in
/home/mbauer/.ssh/id_dsa.pub.
The key fingerprint is:
95:a9:6f:20:f0:e8:43:36:f2:86:d0:1b:47:e4:00:6e
mbauer@homebox.pinheads.com
```

Ennyi! Mostantól ahányszor SSH-val jelentkezünk be a távoli gépre, a kapcsolat valahogy olyasféleképpen fog kinézni, mint amit a *2. listában* láthatunk.
Figyeljék meg, hogy amikor az `ssh`-t meghívom, a `-2` kapcsolót használom, ez ugyanis arra utasítja az SSH-t, hogy csak az SSH Protocol v.2 módszert használja. Alapértelmezés szerint a Protocol v.1 módszert alkalmazza, de az csak az RSA-kulcsokat támogatja, holott mi DSA-kulcsokat másoltunk át. Azt is érdemes megfigyelni, hogy a kulcsra csak helyi elérési úttal/fájlnévvel hivatkozom: ez jó példa arra, hogy amikor RSA- vagy DSA-azonosítást használunk, az általunk begépelt jelszó csak a helyileg tárolt titkos kulcsunk kinyitásához szükséges, és semmilyen formában nem továbbítódik a hálózaton.
A fenti egyszerű példához csak egyetlen megjegyzést szeretnék még hozzáfűzni. Két dolgot tételeztünk fel a távoli gépről: 1. ugyanaz az ottani felhasználói nevünk, mint az itteni és 2. a távoli kiszolgáló ismeri az SSH Protocol v.2-t. Ha az első feltétel nem teljesül, a `-l` kapcsolóval megadhatjuk a távoli géphez tartozó felhasználónevet (avagy a `-l` nélkül, az scp-stílusú felhasználónév@gépnév írásmódot használva, például mick@zippy.pinheads.com).
Ha a távoli gép sshd-démonja nem támogatja a Protocol v.2-t, akkor újra kell próbálkozni a `-2` kapcsoló nélkül, és hagyni, hogy az SSH visszatérjen a név/jelszó azonosításhoz, hacsak nincs egy RSA-kulcspárunk, melynek nyilvános része be van jegyezve a távoli gépen.
A fentieket RSA-kulcsokkal végezve, ugyanezeket a lépéseket kell megtennünk, csak más fájlnevekkel:

1. RSA felhasználói kulcspár készítése az ssh-keygen segítségével, például

```
ssh-keygen -b 1024 -C mbauer@homebox.pinheads.com
```

2. Minden távoli gépre, ahova csak csatlakozni szeretnénk, fel kell másolni a nyilvános kulcsunkat a `$HOME/.ssh` könyvtárban található `authorized_keys` fájl egy külön sorába. Az RSA-kulcsok alapértelmezett fájlnevei az identity és az identity.pub.

Még egyszer, ha az `ssh`-t a `-2` kapcsoló nélkül futtatjuk, akkor alapértelmezés szerint RSA-azonosítást használ.
Mi történik, ha elfelejtjük az RSA- vagy DSA-kulcshoz használt jelszavunkat? Hogyan fogunk visszakerülni a távoli gépre, és mi módon tudjuk megváltoztatni az immár használhatatlanná vált kulcsot az authorized_keys fájlban? Nem kell aggódni: amennyiben nem sikerül belépni RSA/DSA azonosítással, az SSH automatikusan visszavált név/jelszó azonosításra, és rákérdez a távoli gépen megadott jelszavunkra. Ha ezt a „visszaváltás” lehetőséget szeretnénk kikapcsolni, és kizárólag az RSA/DSA bejelentkezéseket akarjuk megengedni, akkor az `sshd_config`-ban található `PasswordAuthentication` értéket változtassuk „no”-ra, minden távoli gépen, ahol `sshd` fut. Emlékezzünk rá, hogy amikor a dolgok kiszolgálóoldaláról beszélünk, az sshd az RSA és a DSA azonosítást egyaránt engedélyezi, ha egy ssh-ügyfél ilyen kérelemmel fordul hozzá. Két értékkel közvetlenül engedélyezhetjük, illetve tilthatjuk a protokollokat, ezek az `RSAAuthentication` és a `DSAAthentication`.

2. lista Bejelentkezés DSA-kulcsok segítségével

```
bauer@homebox:~/ > ssh -2 zippy.pinheads.com
Enter passphrase for DSA key
'/home/mbauer/.ssh/id_dsa':
Last login: Wed Oct  4 10:14:34 2000 from
homebox.pinheads.com
Have a lot of fun...
mbauer@zippy:~ > _
```

Egy vagy több felhasználói kulcs erősebbé teszi az azonosítási biztonságot, és jobban kihasználja az SSH által nyújtott lehetőségeket, mint a név/jelszó azonosítás. Emellett ez az első lépés afelé, hogy az SSH-t parancsfájlokban használjuk. Egyetlen gond jelentkezhet a PK-titkosítás automatizálásával kapcsolatban: bár a PK-alapú azonosítás átlátszó, a megelőző kulcsazonosítás nem az.
Hogyan tudnánk biztonságosan átlépni, vagy egyszerűsíteni ezt a folyamatot?

Erős azonosítás egyszerűsítése az ssh-agent segítségével

Több lehetőség is van: az egyik, hogy jelszó nélküli kulcsokat készítünk, így senkinek sem kell jelszót begépelnie, amikor a kulcsot használja, a másik pedig az `ssh-agent` használata.
Ez utóbbi tulajdonképpen egy saját titkos kulcsgyorstár a memóriában, mely lehetővé teszi, hogy ismételten használhassuk a kulcsot, ha egyszer már begépeltük a jelszavunkat. El kell indítani az `ssh-agent`-et, majd be kell tölteni a kulcsot az `ssh-add` paranccsal, végül meg kell adni a kulcshoz tartozó jelszavunkat. Az ily módon „kinyitott” titkos kulcs a memóriában marad, így minden további `scp`- vagy `ssh`-hívás képes lesz használni a gyorstárazott, kinyitott kulcsot, a jelszó újrakérdezése nélkül is.
Ez nem tűnik túl biztonságosnak, pedig az. Először is, csak az ssh-agent folyamat tulajdonosa használhatja a betöltött kulcsokat. Például ha „root” és „bubba” bejelentkeznek, és mindketten elindították a saját `ssh-agent` folyamatukat, valamint betöltötték a megfelelő titkos kulcsaikat, akkor sem tudnak egymás gyorstárazott kulcsaihoz hozzáférni, nem áll fenn annak a veszélye, hogy „bubba” a rendszergazda azonosítóit használja fel `scp` vagy `ssh` folyamatokhoz.
Másodszor, az `ssh-agent` csak helyi `ssh`- és `scp`-hívásokra válaszol, a hálózatról közvetlenül nem érhető el. Más szavakkal, ez egy helyi szolgáltatás, ezért nem fordulhat elő az sem, hogy egy külső, behatolni próbáló személy ellopjon, vagy valamely módon megváltoztasson egy távoli `ssh-agent` folyamatot.
Az `ssh-agent` használata meglehetősen magától értetődő: egyszerűen gépeljük be az `ssh-agent` parancsot, majd hajtsuk végre a képernyőn megjelenő utasításokat. Ez utóbbi talán kicsit zavarosnak tűnhet, és valószínűleg nem túl egyértelmű: az `ssh-agent`, mielőtt a háttérbe vonulna, rövid, az általunk futtatott parancsértelmezőnek megfelelő környezeti változóra vonatkozó meghatározást ír a képernyőre, amit végre kell hajtanunk, mielőtt bármilyen kulcsot betölthetnénk. A végrehajtáshoz egyszerűen jelöljük ki ezeket a parancsokat az egérrel, majd jobb gombbal másoljuk őket a parancssorba (lásd a *3. listát*).

3. lista Az ssh-agent indítása

```
mbauer@pinheads:~ > # Megjegyzés: ezek itt
fordított aposztrófok.
mbauer@pinheads:~ > eval `ssh-agent`
Agent pid 11518
mbauer@pinheads:~ > _
```

4. lista Cat futtatása a távoli gépen

```
mbauer@homebox > ssh mbauer@zippy.pinheads.com \
    cat /var/log/messages | more
Oct  5 16:00:01 zippy newsyslog[64]: logfile
turned over
Oct  5 16:00:02 zippy syslogd: restart
Oct  5 16:00:21 zippy ipmon[29322]:
16:00:20.496063
ep0 @10:1 p 192.168.1.103,33247 -> 10.1.1.77,53
PR
udp len 20 61  K-S K-F

satöbbi...
```

5. lista Xterm átirányítása a távoli gépről

```
mick@homebox:~/ > ssh -2
mbauer@zippy.pinheads.com
Enter passphrase for DSA key
'/home/mick/.ssh/id_dsa':
Last login: Wed Oct  4 10:14:34 2000 from
homebox.pinheads.com
Have a lot of fun...
mbauer@zippy:~ > xterm &
```

6. lista FTP átirányítás ssh segítségével

```
mick@homebox:~/ > ssh -2 -f mbauer@zippy -L \
7777:zippy:21 sleep 20
Enter passphrase for DSA key
'/home/mick/.ssh/id_dsa':
mick@homebox:~/ > ncftp -p 7777 localhost
```

A *3. listában* az út harmadánál járok: elindítottam az `ssh-agent` folyamatot, és az `ssh-agent` kinyomtatta a BASH írásmód szerinti változómeghatározásokat.
Megjegyzem, hogy a kivágás és másolás minden xterm alatt is működik, a tty (szöveges) konzol alatti működéshez futnia kell a `gpm`-nek. Ha minden más csődöt mond, még mindig begépelhetjük a parancsokat. Ha az `ssh-agent` már fut, az SSH_AUTH_SOCK és az SSH_AGENT_PID pedig meghatározva és exportálva van, akkor eljött az idő, hogy betöltsük a titkos kulcsunkat. Egyszerűen csak gépeljük be az `ssh-add` parancsot, egy szóközt, majd pedig a betölteni kívánt titkos kulcs nevét a teljes elérési úttal. Ha nem adunk meg fájlt, akkor a program automatikusan megpróbálja betölteni a $HOME/.ssh/identity-t. Ez azonban az RSA-azonosításhoz rendelt alapértelmezett felhasználói titkos kulcs neve, amennyiben a kulcsunkat másképpen hívják, vagy DSA-kulcsot használunk, meg kell adnunk a nevet, mégpedig teljes elérési úttal, azaz például:

```
ssh-add /home/mbauer/.ssh/id_dsa.
```

Az ssh-add programot annyiszor futtathatjuk (annyi kulcsot tölthetünk be), ahányszor csak akarjuk. Ez hasznos lehet, ha RSA- és DSA-kulcspárokat is használunk, és különböző SSH-változatokat futtató távoli gazdagépeket érünk el akár csak RSA-kulcsokat, akár DSA-kulcsokat egyaránt támogatnak.

Jelszó nélküli kulcsok a héjprogramok számára

Az `ssh-agent` hasznos lehet, ha bejelentkezéskor indítunk parancsfájlokat, vagy többször kell az `ssh`-t, illetve az `scp`-t futtatnunk. De mi a helyzet a `cron` munkákkal? Nyilvánvaló, hogy a `cron` nem tud név/jelszó választ adni, vagy begépelni a jelszót a PK-azonosításhoz.
Ez az a hely, ahol jelszó nélküli kulcspárokat kell használni. Egyszerűen futtassuk le az `ssh-keygen`-t a fentiek szerint, de jelszó begépelése helyett üssünk ENTER billentyűt. Valamilyen fájlnevet is begépelhetünk az alapértelmezett „identitiy” vagy „id_dsa” helyett, ha ezt a kulcspárt nem alapértelmezett felhasználói kulcspárnak szánjuk valamilyen automatizált feladatokat futtató különleges azonosítóhoz. A `-i` kapcsoló használatával megadhatunk egy bizonyos kulcsot az `ssh` és `scp` folyamatokhoz. Például ha `scp`-t használok egy `cron` munkában, mely naplófájlokat másol, az `scp` sor valahogy így nézne ki:

```
scp -i /etc/script_dsa_id /var/log/messages.*
       ➥scriptboy@archive.g33kz.org
```

Mikor a parancsfájl fut, ez a sor jelszókérés nélkül hajtódik végre: ha a jelszó értéke ENTER, az SSH elég okos ahhoz, hogy ne háborgassa feleslegesen a felhasználót.
Emlékezzünk azonban arra, hogy a távoli gép oldalán a `/etc/script_dsa_id.pub`-ban rejtőző kulcsot a megfelelő `authorized_keys2` fájlba, jelen esetben a `/home/scriptboy/.ssh/authorized_keys2` állományba kell másolni.
Figyelem! Mindig védjük az összes titkos kulcsunkat. Ha kétségeink vannak, hajtsunk végre egy `chmod go-rwx titkos_kulcs_fájl` parancsot.

Távoli parancsvégrehajtás SSH használatával

Itt az ideje, hogy visszatérjünk ebből a PK-varázslásból, és megnézzünk egy egyszerű, de a parancsfájl-készítéshez elengedhetetlen SSH-képességet: a távoli parancsokat. Mindeddig az `ssh` parancsot kizárólag héjprogramok indítására használtuk. Azonban ez csak az alapértelmezett viselkedése, ha ugyanis az `ssh`-t úgy indítjuk, hogy utolsó értékként egy parancsot adunk meg, akkor az SSH a megadott parancsot fogja végrehajtani a távoli gépen a héjprogram helyett.
Tegyük fel, hogy egy gyors pillantást szeretnék vetni a távoli rendszer naplójára, ahogy azt a *4. listában* megfigyelhetjük. Amint látható, a „zippy” nevű gép visszaküldi a `/var/log/messages` állományának tartalmát a konzolomra. Figyeljük meg, hogy a kimenetet átirányította a helyi gépre, további feldolgozás végett.
Két dologra hívnám fel a figyelmet: 1. A további felhasználói közbeavatkozást igénylő programok futtatása trükkös megoldást igényel, ezért kerülendő. A héjprogram kivételével, az `ssh` ugyanis akkor működik a legjobban, ha felhasználói bemenetet nem igénylő programokat futtat. 2. Minden azonosítási szabály továbbra is érvényes:

ha egyébként jelszót kellene megadni, továbbra is meg kell tenni. Ezért, ha az SSH-t `cron` munkából vagy más, nem interaktív környezetből indítjuk, bizonyosodjunk meg róla, hogy jelszó nélküli kulcsokat használunk, illetve betöltöttük a kulcsokat az `ssh-agent`-tel.
Mielőtt befejeznénk az SSH a parancsfájlokban témakör tárgyalását, hanyag lennék, ha nem ejtenék pár szót az „rhosts” és az „shosts” azonosításról. Ezek azok a módszerek, amelyek révén a belépés automatikusan engedélyezett minden olyan felhasználó számára, aki a következő fájlokban meghatározott gépekről lép be: `$HOME/.rhosts`, `$HOME/.shosts`, `/etc/hosts.equiv`, és `/etc/shosts.equiv`.
Gondolom, mindenki kitalálta, hogy az rhosts-elérés elképesztő módon nem biztonságos, mivel teljes egészében a forrás IP-címében és a gépnevekben bízik, ezeket pedig igen sokféleképpen lehet hamisítani. Ezért aztán, az rhosts-azonosítás alaphelyzetben le van tiltva. Az shosts már más, annak ellenére, hogy hasonlóképpen viselkedik, mint az rhosts a kapcsolódó gép azonosságának ellenőrzése itt gépkulcsellenőrzéssel történik. Továbbá az shosts eljáráson keresztül csak a fellépni szándékozó gép rendszergazdája képes átlátszóan kapcsolódni.
Egyébként, az rhosts eléréseket RSA- vagy DSA-azonosítással egyesíteni nagyon hasznos móka, különösen, ha jelszó nélküli kulcsokat használunk. Az shosts és rhost PK azonosítással vagy anélkül történő, SSH alatti használatával foglalkozó további adatokért tekintsük meg a `sshd(8)` leírását.

TCP-kaputovábbítás SSH-val: VPN a tömegeknek!

Íme, megérkeztünk a végkifejlethez (és ahhoz a részhez, amihez az rhosts/shosts teljesebb megvitatásának feláldozása árán mentettem helyet): a kapuátirányítás témájához. Az `ssh` lehetőséget ad a távoli bejelentkezésre és parancsvégrehajtásra, `scp`-vel pedig megoldható a fájlmásolás. De mi a helyzet az X-szel, a POP3-mal és az FTP-vel? Nem kell aggódnunk, az SSH ezeket is, sőt, a legtöbb TCP-alapú szolgáltatást biztonságossá tudja tenni!
Az X-alkalmazások távoli konzolunkra továbbítása kivételesen egyszerű. Először a távoli gépen szerkesszük át a `/etc/ssh/sshd_config` fájlt és állítsuk az X11Forwarding értéket „yes”-re (az OpenSSH 2x változatban az alapértelmezett érték a „no”), második lépésként létesítsünk ssh-kapcsolatot a helyi konzolról a távoli gépre, a megfelelő azonosítási módszerrel, és végül futtassunk olyan X alkalmazást, amilyet csak akarunk. Ennyi az egész! Mondanom sem kell (remélem), a helyi rendszeren X-nek kell futnia. Ha fut, a távoli alkalmazás minden X kimenetet a helyi munkafelületre irányít át.
Miután az `xterm &` parancsot kiadjuk, egy új xterm ablak jelenik meg a helyi gépen. Ugyanilyen könnyedén futtathatunk Netscape-et, GIMP-et vagy bármi mást, amit csak a helyi X-kiszolgálónk kezelni képes, és telepítve van a távoli gépen.
Az X11 az egyetlen szolgáltatáscsoport, melynek automatikus továbbítására az SSH-t közvetlenül felkészítették. A többi szolgáltatás is könnyen továbbítható a `-L` kapcsolóval. Figyeljük meg a *6. listát*! Az `ssh`-sor első része már ismerős: SSH Protocol v.2-t használok és más felhasználónévvel (mbauer) jelentkezek be a távoli gazdagépre (zippy). A `-f` kapcsoló azt jelzi az `ssh`-nak, hogy a háttérben fusson (fork background), miután elindította az utolsó értékként kapott parancsot, mely jelen esetben `sleep 20`. Ez azt jelenti, hogy az `ssh` 20 másodpercig fog aludni ahelyett, hogy egy héjprogramot indítana el.
Húsz másodperc bőven elegendő, hogy elindítsuk a csatornában futó (tunneled) FTP-kapcsolatot, ez a `-L` kapcsolót követő varázslás következtében jön létre. A `-L` helyi továbbítást (local forward) jelöl: egy átirányítást az ügyfélrendszerünkön található helyi TCP-kapuról a kiszolgálórendszer távoli kapujára. A helyi továbbítás a következő írásmódot követi: helyi_kapu_szám:távoli_gépnév:távoli_kapu_szám ahol a helyi_kapu_szám tetszőleges kapu a helyi gépen, a távoli_gépnév a távoli kiszolgáló neve vagy IP-címe, a távoli_kapu_szám pedig a távoli gép azon kapuja, ahová a kapcsolatot továbbítani akarjuk. Megjegyzem, `ssh`-val bármely felhasználó létrehozhat helyi átirányítást magas (1024-nél nagyobb) kapuszámokon, de csak a rendszergazda engedheti meg nekik, hogy kivételes (1024-nél kisebb) kapukat használjanak. A fenti példánál maradva, miután az `ssh` „elalszik” visszatértünk a helyi héjprogramunkba, és 20 másodpercünk van arra, hogy felépítsük az FTP-kapcsolatot. Megfigyelhető, hogy a 6. listában `ncftp`-t használok: ennek az az oka, hogy az `ncftp` támogatja a `-p` kapcsolót, amivel rávehetem a 7777-es helyi átirányított kapuhoz való csatlakozásra. Az is látható, hogy az `ncftp`-nek a saját rendszerem nevét adtam meg a távoli gép neve helyett, ugyanis a kapcsolatot az `ssh` fogja átirányítani a távoli helyre.
Húsz másodperc múltán, az ssh-folyamat megpróbál leállni, de mivel a helyi átirányítást használó FTP-kapcsolat még mindig működik, az `ssh` egy üzenetet küld erről, és élő marad mindaddig, míg az átirányított kapcsolat le nem zárul. Semmi sem gátolhat meg bennünket abban, hogy egy bejelentkező héjprogramot indítsunk távoli parancs helyett (egyszerűen csak el kell hagynunk a `-f` kapcsolót, majd egy másik virtuális konzolon keresztül elindíthatjuk az FTP-t stb.). Ha a `-f` kapcsolót és a `sleep` parancsot használjuk, akkor sem vagyunk pontosan 20 másodperces várakozásra kárhoztatva – az alvási időtartam lényegtelen (mindaddig, amíg elég időnk van elindítani az átirányított kapcsolatot).
Így aztán, bármilyen távoli parancsot futtathatunk, ami előidézi a megfelelő szünetet, de logikus dolog a `sleep`-et használni, hiszen pontosan ilyesmire való. Még egy ötlet: ha egy adott helyi átirányítást minden ssh-kapcsolatnál használni akarunk, akkor megadhatjuk a munkakönyvtárunkban található beállítófájlban (`$HOME/.ssh/config`). Az írásmód a `-L` kapcsolóéhoz hasonló:

```
LocalForward 7777 zippy.pinheads.com:21
```

Kicsit bővebben: a „LocalForward” értéknév után egy szóköz vagy egy Tab következik, majd a helyi kapuszám, újabb szóköz, a távoli gép neve vagy IP-címe, kettőspont szóköz nélkül, végül a távoli kapu száma követi. Ugyanezt az értéket a `/etc/ssh/ssh_config` fájlban is használhatjuk, ha azt szeretnénk, hogy valamennyi ssh-folyamatra érvényes legyen.
Nos, ezek lettek volna az SSH és az OpenSSH magasabb szintű alkalmazásai. Most már el kell köszönnöm, a további részleteket és újabb képességeket olvasóinknak kell kikeresniük a leírásokból. Ne feledkezzenek meg a titkosításról!

Mick Bauer (mick@visi.com)
az ENRGI hálózatmérnöki és tanácsadó cég minneapoloisi részlegének biztonsági vezetője. 1995 óta Linux-rajongó, és 1997 óta az OpenBSD megszállottja. Különös élvezetét leli abban, hogy ezeket az élvonalbeli operációs rendszereket rávegye arra, hogy elavult roncsokon fussanak. Mick szívesen fogad minden kérdést és hozzászólást.

Kapcsolódó cím

RSA Crypto FAQ
➲ http://www.rsasecurity/rsalabs/faq/

Az xinetd használata

Bemutatjuk, hogyan érdemes hozzákezdeni az xinetd beállításához.

A biztonságosabb xinetd hozzáférés-vezérlést, továbbfejlesztett naplózást és erőforráskezelést kínál a számunkra. A RedHat 7 és a Mandrake 7.2 változatok esetében már az xinetd lett a szabványos internetdémon. A cikk az xinetd 2.1.8.8pre3 változata alapján készült. Szeretném felhívni a figyelmet a démon néhány érdekesebb szolgáltatására.
Úgy tűnik, hogy az xinetd eredeti szerzője, *Panagoitis Tsirigotis* (panos@cs.colorado.edu) abbahagyta a projekt fejlesztését. Munkáját *Rob Braun* (bbraun@synack.net) vette át, és jelenleg ő felel a csomag karbantartásáért. Az egyetlen gond, amit a csomag jelenlegi állapotában észrevettem az, hogy néhány fejlécfájllal ki kellett egészítenem, hogy a ***s e l e c t ( )*** megfelelően együttműködjön a régi libc5-öt használó rendszeremmel.

Az xinetd

Az xinetd esetében az inetd-től megszokott sorok helyett zárójelezett, kibővített írásmódú sorokat láthatunk. Ezenkívül a szolgáltatás új naplózó és hozzáférés-vezérlő lehetőségekkel egészült ki. Az inetd lehetővé teszi, hogy TCP-kapcsolatainkat a Vename-féle ***tcp_wrapper*** programmal vezérelhessük, az UDP-kapcsolatok vezérlésére azonban nem teremt lehetőséget. Ezenkívül, miközben az inetd használata mellett szabályozni tudjuk kapcsolataink sebességét (a wait, illetve a nowait kapcsoló után feltüntetett értékkel), nincs lehetőség a példányok számának korlátozására. Ez könnyen szolgáltatásmegtagadás (denial of service) alapú támadásokhoz vezethet.
Az xinetd-t általában az alábbi paranccsal szoktam elindítani, a parancsot az internetszolgáltatások elindításához használt parancsfájlok egyikében helyezem el:

```
/usr/sbin/xinetd -filelog /var/adm/xinetd.log -f
/etc/xinetd.conf
```

A parancs szerint az xinetd tevékenységét a `/var/adm/xinetd.log` fájlba naplózza, és a `/etc/xinetd.conf` beállítófájlt használja. Ebben a cikkben túlnyomórészt ezzel a fájllal foglalkozunk majd.

Beállítások a fordításhoz

A fordítás során három, a hozzáférés-vezérlésért felelős kapcsolóra kell odafigyelnünk: libwrap, loadavg (küszöbfigyelés a terheléskiegyensúlyozáshoz) és az IPv6-támogatás. A legtöbb libwrappel dolgozó démonhoz (ilyen például a portmapper és a sendmail is) hasonlóan, a beállítófájlban elhelyezett „with-libwrap” kapcsoló arra vonatkozik, hogy az xinetd-nek támogatnia kell a /etc/hosts.allow és a /etc/hosts.deny fájlok használatát. Ez a kapcsoló pontosan úgy működik, mint az inetd esetében, és támogatja az összes, az inetd által vezérelt démont. Fontos megjegyezni, hogy az xinetd használata mellett nincs szükség többé a tcpd-re, mivel az xinetd a hozzáférések vezérlését is elvégzi. A libwrap-támogatás mégis nagyon hasznos abban az esetben, ha korábban az inetd/tcpd párost használtuk, és nem szeretnénk megváltoztatni a hozzáférési fájlokat.
A második érdekes beállítás a terhelés figyelése, amit a ./configure parancsfájlban elhelyezett with-loadavg kapcsolóval éleszthetünk fel. A sendmail képes arra, hogy túlságosan nagy terhelés esetén lebontsa a kapcsolatokat. Ezzel a kapcsolóval korlátozni tudjuk az egyes szolgáltatások (vagy akár az összes szolgáltatás) kapcsolatait a gép átlagos terheltsége alapján.

Végül az IPv6-támogatást a ./configure fájlban elhelyezett with-inet6 kapcsolóval tudjuk engedélyezni. Ha engedélyezzük, akkor az xinetd támogatni fogja az IPv6-címek és -kapcsolatok használatát. Ennek a hatását természetesen csak akkor fogjuk érezni, ha a rendszermag (és a hálózat) is támogatja az IPv6 protokollt. Az IPv4 támogatása természetesen továbbra is megmaradt.

1. táblázat **Az xinetd által használt irányelvek**

Irányelv	*Leírás*
socket_type	a hálózati csatlakozó típusa
protocol	IP protokoll, általában TCP vagy UDP
wait	igen vagy nem, egyenértékű az inetd wait/nowait kapcsolójával
user	a folyamat futtatásához használt felhasználói azonosító
server	a futtatni kívánt program teljes elérési útja
server_args	a futtatni kívánt alkalmazás kapcsolói, értékekkel együtt
instances	az elindítható példányok legnagyobb száma
start max_load	az a terheltségi szint, ahol le kell állítani a szolgáltatást (nem kötelező)
log_on_success	a sikeres tevékenységek naplózása
log_on_failure	a sikertelen kapcsolatok naplózása
only_from	azok a hálózatok, illetve gépek, ahonnan engedélyezzük a kapcsolódást
no_access	azok a hálózatok, illetve gépek, ahonnan nem engedélyezzük a kapcsolódást
disabled	ezzel a kapcsolóval tudunk a defaults{} részben szolgáltatásokat kikapcsolni
log_type	a napló elérési útja és típusa, FILE vagy SYSLOG
nice	a szolgáltatás elindításakor beállítandó nice szint
id	a szolgáltatásnak a naplókban használt neve

A beállítófájl

Az xinetd beállítófájlt (általában /etc/xinetd.conf) kétféleképpen hozhatjuk létre: kézzel, vagy pedig az inetd.conf fájlból elkészítve. Az első megoldás igen időigényes, ráadásul sok hibalehetőséget hordoz magában. Ha az önműködő létrehozás mellett döntünk, akkor az itox segédprogramot vagy az xconv.pl parancsfájlt használhatjuk. Habár az xconv.pl tökéletesen helyettesíti az itox segédprogramot, az utóbbinak továbbra is jó használt vehetjük. Ne feledkezzünk meg arról, hogy az itox újbóli lefuttatásával felülírjuk a korábban létrehozott fájlt. Az itox és az xconv használatában nincs különbség.
Mi most az itoxot fogjuk megnézni:

```
$ itox < /etc/inetd.conf > xinetd.conf
```

Az újabb segédprogram, az xconv, jobban bánik a megjegyzésekkel és a tcpd használatával, mint az itox. Az itox esetében meg kell

2. táblázat **A naplózó irányelvek értékei**

Érték	*Sikeres/Sikertelen*	*Leírás*
PID	sikeres	a sikeres kapcsolatfelvétel után elindított folyamat azonosítója
HOST	mindkettő	a távoli gép címe
USERID	mindkettő	a távoli felhasználó RFC 1413 azonosítója
EXIT	sikeres	az elindított folyamat sikeres lefutását jelzi
DURATION	sikeres	a kapcsolat időtartama
ATTEMPT	sikertelen	a kapcsolat sikertelenségének az oka
RECORD	sikertelen	kiegészítő adatok a sikertelen kapcsolatról

jelölnünk azt a könyvtárat, amely a démonokat tartalmazza (például /usr/sbin). Az első olyan rész, amit valószínűleg nem akarunk kihagyni a parancsfájlból, a defaults, amely az xinetd szolgáltatás alapértelmezett beállításait tartalmazza:

```
defaults
{
        instances       = 25
        log_type        = FILE /var/adm/servicelog
        log_on_success= PID HOST EXIT
        flags   = NORETRY
        log_on_failure= HOST RECORD ATTEMPT
        only_from       = 129.22.0.0
        no_access       = 129.22.210.61
        disabled        = nntp uucp tftp bootps who
                          shell login exec
        disabled        += finger
}
```

Rögtön láthatjuk, hogy az xinetd beállító értékek írásmódja a következőképpen néz ki: <irányelv> <műveletjel> <érték>. Az *1. táblázatban* azokat a irányelveket foglaltuk össze, amelyeket az xinetd értelmezni tud. A flags, type, env és passenv irányelvekkel most nem foglalkozunk. A későbbiek folyamán azonban részletesebben is kitérünk majd az only_from és a no_access irányelvekre, illetve a naplózási lehetőségekre is.

Kétféle műveletjelet használhatunk: = és +=. Ha az = jelet használjuk, akkor a bal oldalon álló irányelv felveszi a jobb oldalon megadott értéket. A += segítségével új értékeket adhatunk hozzá egy korábban meghatározott irányelvhez. Ha nem használjuk, akkor az értékeket felülírja. A szolgáltatásokat az alábbi formában kell megadni:

```
szolgáltatás_neve
{
        irányelv = érték
        irányelv += érték

}
```

A szolgáltatást a /etc/services fájlban is fel kell tüntetni, hogy a csatlakozó (socket) és a protokoll beállítása megfelelő legyen.

Néhány szó a hozzáférés-vezérlésről

Nézzük meg egy kicsit közelebbről, hogyan működik a hozzáférés-vezérlés az xinetd esetében. Először is, az xinetd a kapcsolatokat figyeli, nem pedig a csomagokat. Ennek következtében képes megakadályozni egy SYN vagy egy connect() kísérletet, ha az egy olyan gépről érkezik, amelynek nincs engedélye az adott szolgáltatás használatára, viszont semmit sem tud tenni a FIN-pásztázások ellen (ennek során olyan TCP-csomagok érkeznek a kapuhoz, amelyekben be van állítva a FIN kapcsoló). Nem támaszkodhatunk tehát az xinetd szolgáltatásra, ha a kapuk pásztázását szeretnénk megakadályozni. Egy találékony betörő ezen adatok felhasználásával hozzáférés-vezérlési listákat gyűjthet össze. Másodszor, az xinetd (legalábbis a 2.1.8.8pre3 változat) névkeresést hajt végre, amikor egy rendszer kapcsolódni próbál. Korábban ez a keresés a program indításakor zajlott le, de ez mostanra már megváltozott.

A hozzáférés-vezérlés használata meglehetősen egyszerű. Az első irányelv az only_from, amely azokat a hálózatokat vagy gépeket sorolja fel, ahonnan engedélyezzük a kapcsolatot. Az így felállított szabályokat a no_access irányelv segítségével szűkíthetjük. Az irányelvek után hálózati számokat (például 10.0.0.0 vagy 10), hálózatneveket (például *.my.com vagy .my.com), IP-címeket és gépneveket egyaránt megadhatunk. Ha minden kapcsolatot engedélyezni szeretnénk, akkor használjuk a 0.0.0.0 címet.

A szolgáltatás beállításai

Nézzünk meg néhány alapvető alkalmazást. Az első szolgáltatás, amit megvizsgálunk, az echo, amely az inetd és az xinetd esetében egyaránt belső szolgáltatás.

```
service echo
{
        socket_type     = stream
        protocol        = tcp
        wait    = no
        user    = root
        type    = INTERNAL
        id      = echo-stream
}
```

Az echo szolgáltatás rendszergazdai jogokkal fut és tcp protokollt használ. Az id irányelv után megadott echo-stream azonosító fog megjelenni a naplófájlokban. Az only_from és a no_access irányelvek hiánya azt jelzi, hogy a szolgáltatás korlátlanul hozzáférhető. Most nézzünk meg egy másik gyakran használt szolgáltatást, a daytime-ot:

```
service daytime
{
      socket_type       = stream
      protocol          = tcp
      wait              = no
      user              = nobody
      server            = /usr/sbin/in.date
      instances         = 1
      nice              = 10
      only_from         = 0.0.0.0
}
```

A szolgáltatáshoz továbbra is bárki kapcsolódhat, de a szakasz megadja, hogy a kérelmeket kiszolgáló program „nobody” felhasználó jogaival fusson. Ez a példa nem sokkal bővebb az előzőnél. Most egy másik szolgáltatást, az ssh1-et fogjuk megnézni, amely azt hivatott megakadályozni, hogy az sshd kifogyjon az erőforrásokból.

```
service ssh1
{
      socket_type       = stream
      protocol          = tcp
      instances         = 10
      nice              = 10
      wait              = no
      user              = root
      server            = /usr/local/sbin/sshd1
      server_args       = -i
      log_on_failure    += USERID
```

```
        only_from       = 192.168.0.0
        no_access       = 192.168.54.0
        no_access       += 192.168.33.0
}
```

Most a korábban látottakra építettünk. Emlékezzünk vissza arra, hogy az sshd-t a -i kapcsolóval kell elindítani abban az esetben, ha azt az inetd-hez vagy az xinetd-hez hasonló kiszolgálóról indítjuk, éppen ezért ezt a kapcsolót feltüntetjük a server_args irányelv mellett (ha a server irányelv mellett adnánk meg kapcsolókat, akkor nem tudna rendesen működni). Egyszerre legfeljebb tíz ember használhatja a szolgáltatást, ami nem okoz nehézséget annál a kiszolgálónál, ahonnan a példát vettük. A szokásos adatok mellett az RFC 1413-nak megfelelően naplózzuk azokat a felhasználókat is, akik nem tudtak kapcsolódni a szolgáltatáshoz. Végül két hálózatot sorolunk fel, ahonnan nem lehet elérni a szolgáltatást.

Az xinetd és a naplózás

A logging irányelv sokféle kapcsolót megért, és így különféle adatokat kaphatunk a kiszolgálóról (lásd 2. táblázat).

A szolgáltatások naplózására szolgáló sorok általában úgy néznek ki, ahogy azt az alábbi példákban is láthatjuk. Az olyan szolgáltatások esetében, amelyek sikeresen kapcsolódtak a gépünkhöz, általában a folyamat azonosítóját, a gép nevét és a kilépés időpontját szoktuk rögzíteni:

```
log_on_success = PID HOST EXIT
```

Ez elegendő adattal lát el bennünket azokban az esetekben, amikor a kiszolgáló természetes működésében jelentkező hibák okát keressük. A sikertelen kapcsolódások esetében ezzel szemben más jellegű adatokra lesz szükségünk:

```
log_on_failure = HOST RECORD ATTEMPT
```

Feljegyezzük a gép nevét, a kapcsolat sikertelenségének az okát és némi kiegészítő adatot a kapcsolódó gépről (ez néha tartalmazza például a kapcsolat felépítésével kísérletező felhasználó azonosítóját). Ezeket az adatokat célszerű nyilvántartani a naplóban, hogy jó rálátást nyerhessünk a kiszolgáló működésére.

Ha visszatekintünk az alapértelmezett beállításokat tartalmazó részhez, akkor láthatjuk, hogy a naplózás a /var/adm/servicelog fájlba történik. Minden esemény naplózását az xinetdére bíztuk. A legtöbb bejegyzés valahogy így fest majd:

```
00/9/13@16:05:07: START: pop3 pid=25679
from=192.168.152.133
00/9/13@16:05:09: EXIT: pop3 status=0 pid=25679
00/10/3@19:28:18: USERID: telnet OTHER :www
```

Ezen adatok birtokában már nekivághatunk az xinetd működésében felmerülő hibák felkutatásának. A napló tartalmából könnyen kideríthetjük azt is, hogy próbálkoztak-e olyan címekről kapcsolatot felépíteni a gépünkkel, amelyeket kizártunk az adott szolgáltatás használatából. Egyszerűen keressünk rá a naplóban a „FAIL” (Sikertelen) szóra, és máris megjelennek a megfelelő bejegyzések:

```
00/10/4@17:04:58: FAIL: telnet address
from=216.237.57.154
00/10/8@22:25:09: FAIL: pop2 address
from=202.112.14.184
```

A biztonsági kérdésekről még nagyon sokat lehetne beszélni, elég most annyi, hogy a naplóban talált IP-címeket nem szabad teljesen megbízható adatoknak tekinteni, ugyanis azokat viszonylag könnyű hamisítani.

Az xinetd.log fájl ezzel szemben az xinetd-vel kapcsolatos adatokat tárolja. A fájl tartalma nagy segítségével jelenthet, amikor a kapcsolatok hibáinak okát próbáljuk felderíteni.

```
00/10/25@21:10:48 xinetd[50]: ERROR: service
echo-stream, accept:
Connection reset by peer
```

Az xinetd átállítása

Az xinetd.conf fájlt akár az xinetd futása közben is átszerkeszthetjük. Ha szeretnénk életbeléptetni az új beállításokat, akkor egy SIGUSR1 jelet kell küldenünk a futó xinetd folyamatnak:

```
# ps -ax | grep xinetd
50  ?  S     5:47 /usr/sbin/xinetd -filelog
/var/adm/xinetd.log  -f/etc/xinetd.conf
# kill -SIGUSR1 50
```

Ha biztosak szeretnénk lenni abban, hogy a szolgáltatás az új beállításokkal újra lett indítva, akkor nézzünk bele a naplófájlba. Mielőtt kijelentkeznénk, mindenképpen célszerű elvégezni ezt az ellenőrzést; győződjünk meg arról is, hogy újra be tudunk-e jelentkezni. Megjegyzendő, hogy a -HUP kapcsoló hatására nem újraolvassa a folyamat a beállításokat, hanem beszünteti működését. Ezzel a kis trükkel szándékozták megállítani azon betörőket, akik nem ismerik a program leírását.

Mikor használjuk az xinetd-t

Én magam szinte minden szolgáltatásomhoz xinetd-t használok; az egyetlen olyan szolgáltatás, amelynél az xinetd használata a teljesítmény rovására megy, az az Apache. Túlságosan sok folyamatnak kell elindulnia túlságosan rövid idő alatt. A DNS-szolgáltatásokat szintén nem célszerű betölteni az xinetd-be, mivel ebben az esetben is komoly teljesítménycsökkenést tapasztalhatunk.

A sendmailt azonban az xinetd-n keresztül futtatom, mivel így egészen pontosan meghatározhatom, hogy ki kapcsolódhat a kiszolgálóhoz és ki nem. Nálam a sendmail az alábbi módon van beállítva:

```
service smtp
{
        socket_type     = stream
        protocol        = tcp
        wait    = no
        user    = root
        server = /usr/sbin/sendmail
        server_args     = -bs
        instances       = 20
        nice    = 10
        only_from       += 0.0.0.0
        no_access       += 129.22.122.84
                           204.0.224.254 }
```

Az xinetd használata olyan kis mértékben csökkenti csak a teljesítményt, hogy az még egy nagyobb forgalmú levélkiszolgálón is elhanyagolható.

Összefoglalás

Remélem, ez a cikk segít az Olvasónak abban, hogy az xinetd szolgáltatást saját igényeihez igazíthassa. Amint láthattuk, az xinetd lényegesen több lehetőséget tartalmaz, mint az inetd. A Solar Designer webhelyéről (➲ http://www.openwall.com) letölthetünk egy kiegészítést az xinetd egy régebbi (2.2.1-es) változatához, amely lehetővé teszi, hogy a példányokat IP-címenként szabályozhassuk. Ezzel elkerülhetjük az egyszerűbb táblafeldolgozó támadásokat, ne feledkezzünk meg azonban arról, hogy az IP-cím megváltoztatásával ez egyszerűen kijátszható. Nem tudom, hogy ez a kiegészítés bekerült-e az xinetd későbbi változataiba.

José Nazario
doktori tanulmányainak befejezéséhez közeledő biokémikus. Melléktevékenységként foglalkozik a különböző Linux- és Unix-változatokkal. Szabadidejében szeret horgászni és fényképezni.

Telepek fejlődési lehetőségei

A Linux szerepe a géptelepek jövőjében.

A telepek építése tíz évvel ezelőtt, mindössze egy dolgot jelentett, kis számú és nagy teljesítményű kiszolgálók összekapcsolását és megosztott lemezeket. Ezáltal az állandó hálózati hozzáférésű ügyletek feldolgozását végző alkalmazások használhatóbbakká és – néhány esetben – méretezhetőkké váltak. Azóta láthattuk az Internet-alapú rendszerek, a vékony árukiszolgálók (thin commodity servers) felemelkedését és a Linux-alapú nyílt, méretezhető, megosztott feldolgozás teljesen új modelljét.
A linuxos világ számos forgalmazója megpróbál újra olyan telepkialakítási eljárásokat kidolgozni, amelyek a hagyományos Unix OLTP rendszereken már tíz évvel ezelőtt léteztek. Nem biztos, hogy ez a legjobb megoldás. A telepkészítés módszereinek az olyan típusú alkalmazások és szerkezetek irányába kell fordulnunk, amelyek internetes adatközpontokba települnek, és linuxos vékony kiszolgálók csoportjaira épülnek. Az internetes adatközpontban – legyen az egy telephely vagy egy cég – megtalálható alkalmazásszerkezeteknek van néhány olyan közös jellemzőjük, amelyek alapvetően különböznek a régi üvegpaloták szerkezetétől. Bővebben kifejtve az internetes adatközpont-szerkezetekben többnyire fellehető néhány közös minta:

- A szerkezetek többrétegűek (multitiered), különösen igaz ez a webkiszolgálókra, alkalmazáskiszolgálókra és adatbázis-kiszolgálókra.
- Minden réteg kiszolgálók tucatjait tartalmazhatja.
- Minden réteg egyedi követelményeket támaszt a megosztott és a lemásolt adatokkal szemben.

A nagy forgalmú e-kereskedelemmel foglalkozók vagy tartalomszolgáltatók esetében a webes szolgáltatás alapját képező, megosztott erőforráskészlet egészének kezelése sokkal összetettebb feladat, mint egyszerű kiszolgálók hibatűrő működését biztosítani.
Míg néhány alkalmazás, mint például az OLTP vagy más hasonló e-kereskedelmi hibajavító rendszer továbbra is megosztott lemezterületen alapuló megoldásokat igényel, a korszerű alkalmazások sok esetben képesek kihasználni a középréteg által nyújtott programalapú többszörözés (replication) előnyeit.
Hiányzik azonban belőlük az a háttér, amely egyetlen nagy számítási környezetté olvasztaná egybe ezeket a megoldásokat. Az alkalmazásvezérlés – beleértve a hibajavítást – továbbra is olyan kulcsfeladat marad, amivel az üzemeltetőnek manapság mindenképpen szembe kell néznie. Szerencsére azonban a lehetőségek köre egyre bővül.
Az alkalmazások, amelyeknek megosztott adatelérésre van szükségük, manapság gyakran gépek tucatjait használják, többnyire több megjelenési ponton (Points of Presence, POPs). Ez azt jelenti, hogy többé már nem elegendők a hagyományos fürtkészítés megosztott lemezmegoldásai, amelyek többnyire fizikailag megosztott SCSI-kapcsolatokon alapulnak. E megoldások lehetőségeit behatárolja az alkalmazott fizikai vezetékelés és a SCSI-címkorlát, így általában csak két-három rendszert képesek kiszolgálni.

A webkiszolgáló réteg

A webkiszolgáló réteg a Linux erőssége, csak olvasható vagy program által készülő adatok jellemzik. Ezt a típusú adatot meg lehet osztani (és gyakran meg is teszik) több kiszolgáló közt, hálózati csatlakozású, NFS-t használó tárolóeszközök segítségével. Minden kiszolgáló csatolja az NFS-eszközt, és máris elérheti a megfelelő adatot. Ehhez a sorhoz a bonyolult megosztott SCSI vezetékezési sémák teljesen szükségtelenek, ugyanis a TCP/IP hálózat az összes szükséges kapcsolatot biztosítja a kiszolgálók és a tárolóegység között. Emellett fontos megemlíteni a telepvezérlő (clustering) programot, ez folyamatosan felügyelheti a webkiszolgálók teljesítményét és épségét, valamint megfelelő lépéseket tehet az így nyert adatok alapján.

Az alkalmazási réteg

A hagyományos munkaterületeken az alkalmazások jellemzően kiszolgálóalapúak voltak, és relációs adatbázisokon alapultak.
A legtöbb esetben az adat nem az alkalmazási rétegben helyezkedett el, hanem minden az adatbázisba került. Ez a kép sokkal részletgazdagabb az Internet és a Linux világában. A Java és az Enterprise JavaBeans felemelkedése arra ösztönözte a fejlesztőket, hogy az alkalmazásréteg elemeit az adatbázison kívül, a fájlrendszerben helyezzék el. Ráadásul az alkalmazások manapság sokkal változatosabb adatokkal dolgoznak, ideértve a folyamatos műsorszórást, a szöveges adatokat és sok más olyan adattípust, amelyeket általában nem szoktak SQL-háttéren tárolni.
Emiatt az alkalmazási réteg olyan hibajavító és adatmegosztó követelményeket támaszt, amelyeket sem a hálózatcsatolt tárolók, sem a megosztott SCSI-eszközök nem képesek kielégíteni. A hálózatcsatolt tárolók általában azért nem működnek, mert az NFS protokoll nem képes megfelelően megvalósítani az összefüggőséget egy olyan esetben, ahol egy időben több kiszolgáló is írhatja ugyanazt az adatot.
A megosztott SCSI azért nem megfelelő, mert többszörözést, és nem megosztást kellene alkalmazni több rendszer vagy több POP között. Sok esetben éppen ezért, az alkalmazás inkább saját magának készít másolatot.Vegyünk például egy valutaváltó kereskedelmi alkalmazást, amit egy tucat nagy banknál használnak. Ebben az esetben a valós idejű kereskedelmi adatok többszörözése alkalmazásszinten folyna, a hibajavító eljárást pedig telepszinten lehetne megoldani.

Az adatbázis réteg

Az adatbázisréteg a hagyományos fürtöző megoldások birodalma, ezt a nagy Unix-terjesztők – Sun, HP és az IBM – fejlesztették ki.
Az eredeti elképzelés szerint két kiszolgálót kell csatlakoztatni néhány SCSI-meghajtóhoz. Az egyik közülük a tevékeny mester, a másik pedig tétlen várakozó. A korszerűbb megvalósításokban több kiszolgáló használ egyetlen nagy teljesítményű háttértárat (SCSI-eszköz vagy fiber channel), és egy párhuzamos adatbázis-kezelőt (például az Oracle Parallel Server – OPS) alkalmaznak, hogy kiszolgálják a gépek által kezdeményezett egyidejű eléréseket.
Csakhogy a jó minőségű fiber channel SAN-ok igen drágák, csakúgy, mint az OPS-hez hasonló programok. Ezeknek a megoldásoknak ár/teljesítmény aránya gyakran elfogadhatatlan a linuxos vékony piac vagy egy internetes adatközpont számára. A tevékeny/tétlen SCSI-megoldás erőforrás-pazarló és csak korlátozott méretezhetőséget tesz lehetővé. Szerencsére vannak más megközelítések is, amelyek jobban illeszkednek a Linuxon futó alkalmazásokhoz.
Sok adatbázis forgalmazó fektetett munkát például a megosztásnélküli többszörözőprogrammegoldásokba (Informix Replicator, DB/2 Replication, Oracle Multi-master and hot standby). Ezeknek az

eszközöknek a használatával megoldható, hogy több kiszolgáló osztozzon az adatbázisrétegen, közös, megosztott tárolóeszköz használata nélkül. Maguk az adatbázisok azonban gyakran híján vannak azoknak a képességeknek, amelyek segítségével felderíthetnék a hibákat, megkezdhetnék a hibamentesítést, és szavatolhatnák az alkalmazás épségét. Ennek következtében a legjobb az, ha egy olyan megoldással társulnak, amely biztosítja ezeket a szolgáltatásokat.
Egy életből ellesett példa egy lapkakészítő műhely, amely egy önműködő gyártási alkalmazást futtat Intel-alapú vékony kiszolgálókon. Az alkalmazás nagy fontossága végett a lapkagyártó három többszörözést is kér. Így két másik gép áll készen arra, hogy az esetlegesen leállt kiszolgáló feladatait átvegye. A választott megoldás az Informix adatbázis-többszörözö szolgáltatás volt, amely fizikailag elkülönített gépeken futott, megosztott tárhely igénybevétele nélkül, géptelepkezelő motorral összekapcsolva.

A megosztott tárolók jövője

Néhány más alkalmazásnak fizikailag megosztott tárhelyre van szüksége. Mivel a megosztott SCSI-eszközök továbbra is ritkaságszámba mennek, egy ilyen rendszer összerakása nagy figyelmet, valamint teljes körű – mind a gépgyártóktól, mind a programterjesztőktől – minőségbiztosítást igényel. (Mikor ellenőrizted utoljára a gépi program szintjén a merevlemezedet?) És mikor mindezt nagy fáradsággal összeraktuk, amint láttuk, elég komoly korlátokba ütközünk mind a csomópontok számát illetően, mind pedig a lehetséges fizikai vezetékezés terén. A fiber channel megold ugyan néhányat a gondjaink közül, de rögtön okoz is helyettük néhányat, ideértve a drágaságát, a több terjesztő közötti működési nehézségeket, illetve a kezelési módot, mely sok linuxos vékony kiszolgáló felhasználójának szokatlan lehet.

A következő évben nagy változások várhatók a megosztott tárolók területén. Az ipar vezető cégei két új tároló-összekapcsolási módszert mutatnak be, így a megosztott tárolók ára a vékony kiszolgálók számára is elfogadhatóra mérséklődhet. Így is képesek létrehozni az internetes adatközpontoknál megszokott, nagyszámú csomópont-mennyiség szerinti megosztást is képesek lesznek létrehozni.
A Cisco és más, a piac változásaira érzékeny cégek mint például a 3ware, már támogatja a SCSI-over-IP módszert, amely lehetővé teszi, hogy hálózati kapcsolókkal ellátott Ethernet hálózaton SCSI-protokollt futassunk.
Az Intel és a kiszolgálóforgalmazók az Infiniband nevű új I/O szabvány mögött sorakoztak fel. Ez olyan kapcsolt I/O rendszert képes szolgáltatni, amelyet akár a kereskedelmi kiszolgálók alaplapján megtalálható lapkakészletekben is alkalmazni lehet. Igazság szerint ezek a fejlesztések kiegészítik egymást – a Gigabit Ethernet kártyák ülhetnek Infiniband szerkezeteken, és futtathatják a SCSI-over-IP protokollokat. Következésképpen a közeljövőben várhatóan rendszerek tucatjai vagy akár százai is osztozhatnak egyazon tárhelyen, ennek a haladó kapcsolatrendszernek köszönhetően.
És hogy mi határozza meg a jövő adatközpontjának szerkezeti alapjait? A telepkészítési módszerek, amelyek segítségével a megismert kapcsolatok révén valóban használható és méretezhető megosztott megoldásokat készíthetünk a linuxos gépekből.

Ken Dove
jelenleg a PolyServe Inc. vezető mérnöke. Tizenkét éven át a Sequent Computer Systems vezető programmérnöke volt, majd az IBM alkalmazásában állt.

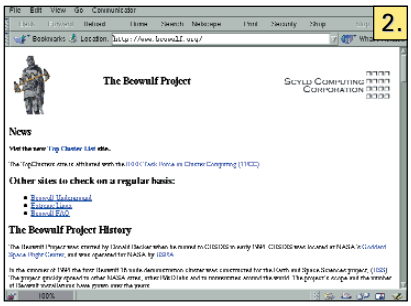
Távérzékelés Linuxszal

Egy cég belevágott...
Időt, pénzt takarított meg a Linux és a Beowulf telep alkalmazásával.

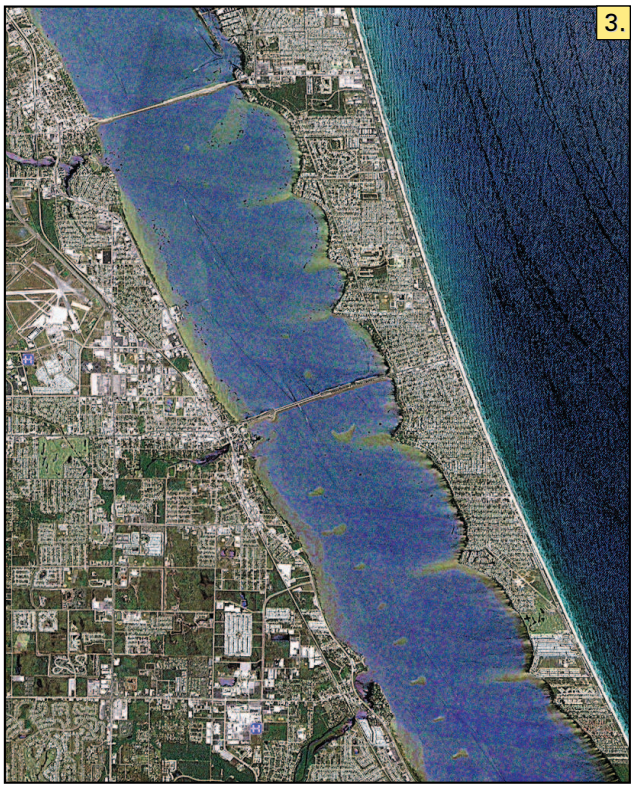
Nagy méretű műhold- és légi felvételeket dolgoz fel ügyfelei számára az ImageLinks Inc., a floridai Melbourne-ben. Ehhez a munkához több gigabájtnyi képadatot kell számításigényes eljárások segítségével háromdimenziós képekké alakítaniuk vagy összetett adatfeldolgozást végrehajtani. Írásunkban bemutatjuk, hogy a cég mi módon állt át a Linuxra, és ebből milyen előnyei származtak.

1996-ban engedélyt kapott – korábban titkos – kormányzati programok forgalmazására. Ez körülbelül 5000 sor objektumorientált C++ kódot jelentett – 15 évi fejlesztés eredményét. A cég indulásakor nagy teljesítményű SGI és Sun kiszolgálókra alapoznunk. Több mint félmillió dollár értékű gépparkot béreltünk, ennek havi költsége több mint 15 ezer dollárra rúgott. A berendezéseken túl, sokat kellett költenünk a fordítókhoz, a programkönyvtárakhoz és egyéb programokhoz tartozó felhasználási szerződésekre. Abban az időben még egy egyszerű memóriabővítést is a gyártótól kellett megrendelnünk, ellenkező esetben érvényét vesztette volna a karbantartási szerződésünk. A szakembereink egy része otthon már Linuxszot használt és elkezdtünk azon gondolkodni, hogy hogyan lehetne átültetni a meglévő programunkat Linuxra. Egyik nap ebéd közben beszélgettünk erről, majd visszafelé menet megálltunk egy számítógépboltnál, megvettük a cég hitelkártyájára azt, amire szükségünk volt, és megkezdtük a kód átültetését. Feltelepítettünk egy 5.2-es RedHat Linuxot és nekiláttunk a munkának. A rákövetkező néhány hónapban *Dave Burken* és *Ken Melero* átbogarászták a projektet, megkeresték és kijavították a felületfüggő kódrészleteket. A kódunk sokat használt sablonokat, ezt a fordító nem kezelte jól, így súlyos nehézségekbe ütköztünk. Mire megjelent a 6.0-s RedHat, a gcc közvetlenül támogatta a sablonokat, és az átültetést gyorsan be lehetett fejezni.

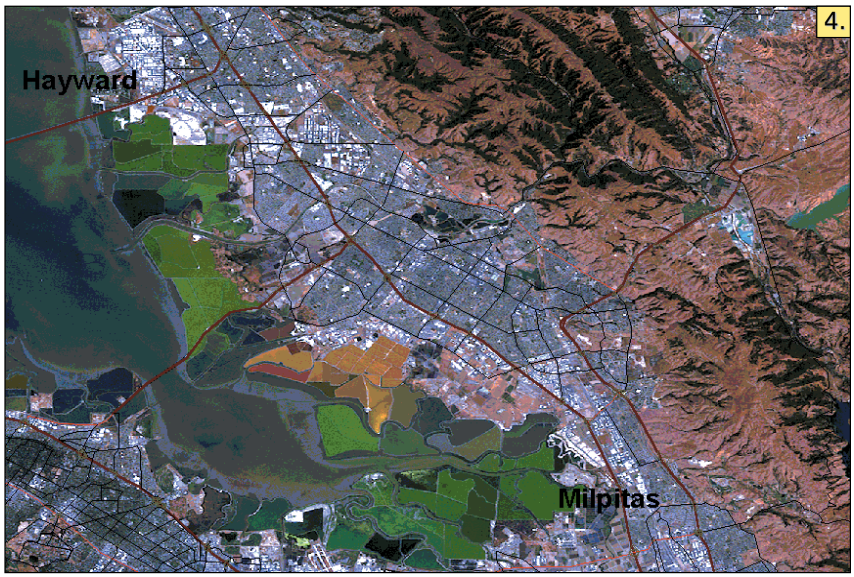

1.



2.


3.

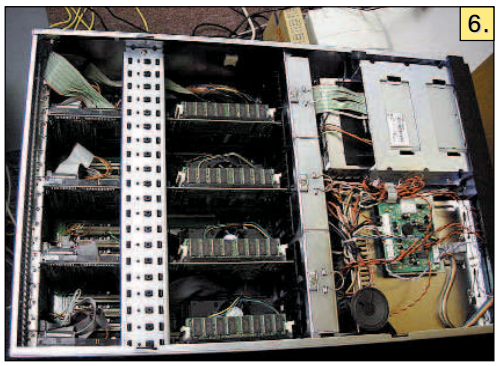
Abból indultunk ki, hogy a linuxos átültetés Intel-processzoron sokkal olcsóbb megoldás lesz, de azt nem vártuk, hogy a gép teljesítménye megegyezzen a nagy teljesítményű munkaállomásokéval. Szerencsére, a második pontban tévedtünk. Az első visszajelzés, ami azt sejtette, hogy javult a rendszer teljesítménye, a fordítás idejének csökkenése volt. A `make World` parancs, amely lefordította programjaink teljes forráskódját, 10–12 óra alatt futott le a Silicon Graphics Indigo 2-s gépén. Ugyanez a művelet kevesebb, mint két óra alatt befejeződött a kétprocesszoros Pentium gépen, a végrehajtható állomány mérete pedig magáért beszélt. A gcc által létrehozott futtatható állományok feleakkorák voltak, mint a régi, bérelt fordítók kimeneti fájljai. Ez azt mutatta, hogy a nyílt forráskódú fejlesztői eszközök hatékonyabb kódot készítettek. A teljesítménynövekedés világosan megmutatkozott, amikor élesben elkezdtünk használni néhány linuxos rendszert. A legszélsőségesebb példa a több érzékelő által készített kép összeolvasztása (cross-sensor image fusion) volt. E módszer segítségével több különböző műholdas felvételből állítjuk elő az új terméket, például ötvözhetünk nagy felbontású fekete-fehér felvételeket kis felbontású színes felvételekkel. A kiindulásnál használt képek az esetek többségében különböző szögből készültek, különböző időpontokban, más-más méretben és felbontással. Ezen tényezők figyelembevételével a program összetett átalakításokat hajt végre a háromdimenziós térben, a műholdfelvételekből egy belső, háromdimenziós modellt állít elő. Amikor ezzel végez, értelmes minta-újravételező eljárások pásztázzák a háromdimenziós modellt, és előállítják belőle megfelelő vetítéssel a kívánt méretű térképet. Ehhez több gigabájt képadatot kell összetett képfeldolgozó és három-



4.


7.


5.


6.

1. kép Az ImageLinks rendszerén futó bWatch program képernyőképe. A piros vonalak adatátvitelt jelölnek a gépek között és mutatják, hogy a folyamat CPU-hoz kötött

2. kép www.beawful.org

3. kép Egy összeolvasztással készült kép részlete. A képen a floridai Melbourne látható, a Landsat 5 színes és az indiai IRS 1C műhold 5 méteres felbontású képeiből

4. kép Összetett kép a kaliforniai Milpitas térségről. Több műholdfelvételből és vektorrétegből tevődik össze

5. kép Az ImageLinks Beowulf telepe 12 gépből, RAID merevlemezből, egy 100 BT hálózati kapcsolóból és a tápegységből áll

6. kép Négy 650 MHz-es Pentium III gép 384 MB memóriával egy 4U szekrénybe szerelhető egységben

7. kép Jeff Largent kipróbálja a gépeket a szekrénybe szerelés előtt

dimenziós átalakító eljárásoknak alávetni. Megszokott volt, hogy ez a folyamat a régebben alkalmazott munkaállomásokon egy egész hétvégét vett igénybe. A linuxos gépek alkalmazása rengeteget javított ezeknek a feldolgozásoknak a futásidején. Ez az óriási javulás a közönséges gép és a remek programok által előállított hatékony kód jobb teljesítményének köszönhető.

A következő jelentős nyereséget a Beowulf géptelepek alkalmazása hozta. A Beowulf telep egy csomó közönséges hálózati kapcsolaton keresztül összekötött számítógéppel megvalósított költségtakarékos szuperszámítógépet takar. A legtöbb esetben adott célra összeállított rendszermaggal futó linuxos gépeket jelent, amelyek Ethernet hálózaton keresztül csatlakoznak egymáshoz. Az egyik gép a mester vagy vezérlő, ez ellenőrzi és ütemezi a szolgagépeket, valamint ez felel minden, a külvilággal folytatott kapcsolattartásért. Régebben a szuperszámítógépek által használt programokat az adott gépfelépítéshez igazították. A közelmúltban a PVM és az MPI-hez hasonló párhuzamos programkönyvtárak fejlődésével, ez a feladat sokkal általánosabb lett. E programkönyvtárak segítségével a programozók beazonosíthatják a kódban azokat a részeket, amelyeket párhuzamosan is fel lehet dolgozni. A programkönyvtárak foglalkoznak a részletekkel, ezek képezik le a kódot a szuperszámítógép felépítésének megfelelően. Nálunk főleg nagy számításigényű eljárásokat kell használni, de szerencsére ezek az eljárások nagymértékben párhuzamosíthatók. Másképpen fogalmazva a kód legnagyobb része lebegőpontos számítás, és a feladatokat könnyű olyan részekre felosztani, amelyek nem igényelnek jelentős kapcsolattartást a processzorok között.

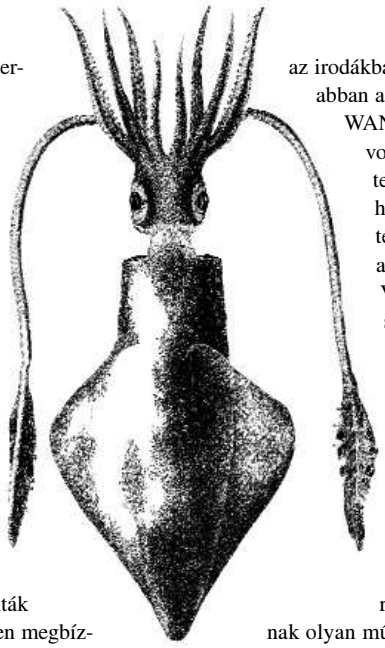
A megvalósításunk lényege az volt, hogy feldaraboltuk a képeket, és minden képkockát más-más gépnek adtunk feldolgozásra. Egy tizennégy számítógépből álló fürtöt építettünk, bedrótoztuk a PVM-et a kódba és egyenesen arányos teljesítménynövekedést tapasztaltunk, ahogy egyre több processzort adtunk a fürthöz. A program futásának vizsgálata kimutatta, hogy az adatok közlése rövid időszeleteket vesz igénybe, a gépek idejük jelentős részét a megfelelő számítások végzésével töltik. A feladatunkra eszményi megoldásnak tűnt a fürtözéses telepek alkalmazása, ha nagyobb sebességre vágyunk, egyszerűen több adatot kell a rendszerbe nyomni, újabb processzorokat kell hozzáadni. A Beowulf telep beindításával az összetett többérzékelős összeolvasztásos feladatok futásideje nagymértékben csökkent.

A teljesítménynövekedés és a gazdaságosság mellett, más előnyünk is származott a váltásból: javult a rendszer megbízhatósága, jobb leírásokhoz jutottunk, ráadásul gyakoriak a programfrissítések is.

Mark Lucas
az Egyesült Államok légierejének nyugalmazott tisztje és az ImageLinks Inc. vezető műszaki tisztje, valamint a ➲ remotesensing.org alapítója. Ők támogatják a távérzékelő térinformatikai rendszerek nyílt forráskódú fejlesztését. Főiskolai villamosmérnöki és egyetemi informatikusi végzettsége van.

Squid proxykiszolgáló telepítése Linux alá

Egy példán keresztül bemutatjuk e „sokcsápos puhatestű” proxykiszolgáló telepítésének, beállításának és karbantartásának rejtelmeit.



A SAS intézet irodahálózata felhasználóinak internetelérését számos proxykiszolgáló biztosította világszerte, valamint a SAS európai központjában, a németországi Heidelbergben. Ezek a kiszolgálók a Squid (a szó jelentése tintahal) proxykiszolgáló programot futtatták, amely a GNU felhasználási szerződés hatálya alá tartozik. Dióhéjban, a Squid lehetővé teszi a gyorstárazást, illetve az adattovábbítást az FTP, a HTTP vagy a GOPHER protokollok alatt elérhető adatelemek számára. A webböngészők ezután a helyi Squid gyorstárkiszolgálót használhatják HTTP-proxynak, ezáltal mérsékelve az adatelérés idejét és a szükséges sávszélességet. A Squid az adatokat a memóriában vagy a merevlemezen tárolja.
A Squid kiszolgálókat többszintűen is lehet szervezni, hogy a központi kiszolgálók nagy adattárakat tegyenek elérhetővé az alacsonyabb szinten található kiszolgálók számára.
A SAS EMEA környezetében már régebben is használták a Squidet, és igen jól szerepelt. A program különlegesen megbízható és zökkenőmentes internetelérést nyújt az ügyfelek számára.
Az eredeti proxykiszolgálók 10.22-es HP-UX rendszert futtató HP munkaállomásokon a Squid 2.1 változatát használták. A rendszer sok különféle gépen futott, de többnyire 64 MB memóriával és 4 GB merevlemezzel rendelkező HP9000/720 munkaállomásokat használtak. Ennek az összeállításnak azonban elég nehézkes a támogatása.
A gépek sorra elérték azt a kort, amikor egymás után jöttek elő a hibáik, ezenkívül a növekvő internethasználat és az irodák bővülése miatti terhelésnövekedéshez már ezek a beállítások sem voltak megfelelőek. A legfőbb gond a lemezkezelés volt: a megnövekedett használat mellett a meglévő 100 MB-s naplóterületek és a 2 GB-os, tulajdonképpeni gyorstárkönyvtárak egyaránt szűkösnek bizonyultak.
Elkezdtünk valamilyen megoldás után kutatni, hogy fenntarthassuk a szolgáltatást. Mivel magával a Squid programmal elégedettek voltunk, és a beállításának rejtelmeiben is eléggé elmélyedtünk már, a Squid használata mellett döntöttünk, és a gépi környezet lecserélésének lehetőségeit kezdtük el vizsgálni. Mivel a Squid egy nyílt rendszer, és Linux alatt kedvező a támogatottsága, jó ötletnek tűnt, hogy a feladatot átlagos SAS EMEA Intel számítógépeken futó Linux-alapú alkalmazásokkal oldjuk meg.
Ehhez egy DELL asztali PC-t választottunk, 256 MB memóriával 500 MHz Intel Pentium processzorral, és belső, 20 GB IDE merevlemezzel. Mivel a DELL erősen kötődik a RedHathez, logikus választásnak ez a rendszer tűnt. Ráadásul a SAS nemrégiben a RedHat partnereként adott ki termékeket.

Szerkezet

A SAS EMEA eredeti szerkezete három központi „szülő” Squid-gyorstárat használt, közvetlen internetelléréssel, és „gyermek” Squid kiszolgálókat az országos irodákban. Néhány kisebb ország elérése közvetlenül a központi főhadiszállás kiszolgálóin keresztül történt, és úgy éreztük, az olcsóbb gépek lehetővé teszik majd, hogy ezekbe az irodákba is proxykiszolgálót telepítsünk. Továbbá, abban a néhány országban, ahol a SAS egymással WAN hálózaton keresztül összekapcsolt irodákkal volt jelen, az olcsóbb gépek lehetőséget teremtettek arra, hogy ide is proxykiszolgálókat helyezzünk el. Ezek a fejlesztések csökkentették a webforgalom válaszidejét, és mérsékelték a WAN hálózataink túlterheltségét.
Végül akadt néhány kifogásunk az eredeti szerkezet rugalmasságát és elérhetőségét illetően, és úgy éreztük, hogy az átalakított kiszolgáló és ügyfélrendszer beállításai növelik a belső felhasználóink számára nyújtott szolgáltatásaink színvonalát.
Az új szerkezet tulajdonképpen alapjaiban nem változott meg. Továbbra is három központi kiszolgálónk volt, csakhogy most Linux fut rajtuk. Több gyermekproxyt telepítettünk, és néhány irodában háromszintű rendszert építettünk ki. Például néhány országnak olyan műholdas irodái voltak, amelyek mindössze egyetlen, a főhadiszálláshoz kapcsolódó WAN hálózaton keresztül csatlakoztak a SAS belső hálózatához. Ebben az esetben proxykat telepítettünk a műholdas irodákba, amelyek lehetőleg a központi iroda helyett az országos központhoz kapcsolódtak.
A rendszerünk újítása volt még a Trend Interscan Virus Wall termék alkalmazása a HTTP-vírusok kiszűrésére. Három víruskereső rendszert telepítettünk szintén RedHat alá. Ezek a rendszerek a jelenlegi Squid szülő gyorstárak mögött helyezkedtek el, és egy vírusszűrő réteget képeztek a Squid gyorstár és a külső Internet között. Mivel a víruskeresők alapjában véve egyszerű folyamvizsgáló rendszerek, a legmagasabb szintű Squid kiszolgálókat állítottuk be úgy, hogy válogathassanak közöttük.

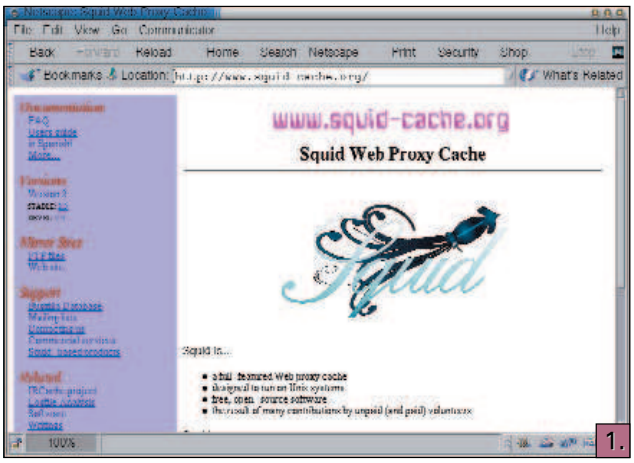
Telepítés

Az eredeti HP-UX kiszolgálók telepítése úgy történt, hogy egy ismert rendszer lemezének tartalmát sokszorosítottuk. Ez több szempontból sem volt kielégítő megoldás, nem beszélve arról, hogy igencsak nehéz volt gondoskodni a lenyomat karbantartásáról, ha valamelyik programot foltozni kellett, esetleg frissíteni szerettük volna a Squidet stb.
A célunk az volt, hogy parancsfájlokon alapuló önműködő telepítőrendszert hozzunk létre, amit a helyi iroda személyzete könnyedén és gyorsan végre tud hajtani. Az elgondolást végül sikerült kielégítően megvalósítani, ráadásul a rendszernek további előnyei is származtak a biztonsági helyreállítás és beállításkezelés terén.
A gépek felépítéséhez egy KickStart összeállítást készítettünk.
A KickStart a RedHat egyik önműködő telepítés-elősegítő eszköze. Alapesetben megadhatjuk a telepítőnek, miképpen ossza fel a merevlemezt, mely RPM-csomagokat telepítse, és végrehajthatunk néhány helyi rendszerbeállító héjparancsot is.
Mi a KickStart-beállítónkat egy hajlékonylemezre írtuk a szokásos RedHat indítóeszközökkel együtt, és úgy állítottuk be, hogy a CD-ről telepítsen.

Tehát a proxykiszolgálóhoz olyan PC-t terveztünk, ami hasonlít az általunk elvárt összeállításhoz, majd leszállítottuk a CD-t és a hajlékonylemezt a távoli irodának, ahol a telepítést elvégezhetik.
A telepítési folyamat majdnem teljesen önműködő, mindössze három adatot kell megadni: a gépnevet, az IP-adatokat és a billentyűzet típusát, ugyanis néhány irodánkban az adott nyelvnek megfelelő billentyűzetet használnak. A KickStartban minden más beállítás előre rögzített. Például a telepítés nyelve mindig angol, a kiválasztott csomagok mindig ugyanazok, és a merevlemez is mindig ugyanúgy lesz felosztva. A teljes telepítés a lemez behelyezésétől kezdve a frissen települt operációs rendszer elindulásáig még tíz percet sem vesz igénybe. Ez sokkal gyorsabb, mint ahogy kézzel megoldhatnánk, és jóval kevesebb időt vesz igénybe, mint egy HP-UX telepítése. Nyilvánvaló, hogy ennek van néhány mellékhatása a biztonsági mentések és helyreállítás terén is.

Karbantartás

Rendszerünk futtatása során a szokásos gondokkal kerülünk szembe: a beállítási fájlokat folyamatosan frissíteni, a programokat újabb változatokra cserélni, a naplófájlokat forgatni, a folyamatokat pedig felügyelni kell. A múltban ezeket a feladatokat héjprogramok és cron-feladatok átláthatatlan kavalkádja oldotta meg, sőt, még azt is elnéztük, ha a rendszerek különböző felépítésűek voltak. Első körben ezeket a feladatokat az ***rdist*** csomaggal oldottuk meg, de ez sem volt kielégítő, így tovább kerestünk.
A legmegfelelőbbnek a cfengine tűnt. Ez a program lehetővé teszi számunkra, hogy egy központi helyen létrehozzuk az általános eszközmeghatározásokat és összeállításokat, ahonnan aztán szétoszthatjuk azokat az összes kiszolgálóra. A ***cfengine*** alkalmazása messzemenően sikeresnek bizonyult, annak ellenére, hogy a beállítások gondos megtervezését és a csomaghoz tartozó leírások alapos áttanulmányozását igényelte.
A szétosztani kívánt állományok közt néhány teljesen szokványos is akadt, ezeket egyszerűen átküldhettük a cfengine-nel, úgy ahogy voltak. Mások azonban, bár valamilyen általános sémán alapultak, minden rendszeren egyedi módosításokat igényeltek. Jó példa erre a Squid fő beállításfájlja. Az ilyen esetekben cfengine-nel áttöltöttük a mintaállományt és egy kis parancsfájlt, ami tudta, miképpen kell elvégezni a módosításokat. Kihasználva a cfengine képességét, hogy áttöltés után parancsokat képes végrehajtani, lefuttattuk ezt a parancsfájlt, majd jeleztük a Squidnek, hogy újra töltse be a beállításfájljait. Így egy központilag irányított és összehangolt beállítórendszert tudtunk létrehozni.
A cfengine-rendszer az általunk karbantartott helyi NIS-térképre épült. Ez a NIS-térkép egyszerűen összegyűjtötte a gépneveket, és különböző tulajdonságokat rendelt hozzájuk. Például a SQUID-CHILD kulcsszóval a második szinten elhelyezkedő gépeket jelöltük meg, melyek egy SQUID-MASTER-hez kell kapcsolódjanak. Ezt a NIS-térképet dolgoztuk fel a cfengine-hez szükséges osztályok előállításához, így végül a beállításadatokat központilag tároltuk, és nem az egyes kiszolgálókon.
Több gondot okozott viszont a telepített programok karbantartása. Nagyjából RedHat 6.2-alapú rendszert futattunk, de mialatt a proxykat telepítettük, a RedHat kiadott néhány frissítést, amire szükségünk volt. Főként a biztonsági javításoknak van számunkra jelentős szerepe. Ezenkívül van néhány általunk készített RPM-csomag is, ráadásul újabb Squid-változatot használunk, ami tartalmaz pár olyan lehetőséget, amit a RedHat még nem támogat. Az általunk használt útválasztó démon is különleges, néhány olyan útválasztó protokollt is ismer, mely az alapcsomagban nem található meg. Végül, van néhány olyan RPM-csomagunk is, ami soha nem is volt a RedHatben.
Magától értetődő lépés volt egy FTP-kiszolgáló létrehozása. A Network Appliance Fileren futtattuk a beépített FTP-szolgáltatást, mely a RedHat-változatokat is tartalmazta. Van egy tükrözőnk, ami mindig letölti a legfrissebb javításokat a RedHat tükörhelyről. Az FTP-kiszolgálón egy külön fában tároljuk a saját RPM-csomagjainkat.
Jó időbe tellett, mire sikerült összehozni a csomagfrissítő rendszert, de még mindig nem voltunk teljesen elégedettek. A legjobb eszköz, amit találtunk, az ***autorpm*** volt. Ezt az eszközt be tudtuk úgy állítani, hogy önműködően vizsgálja az FTP-kiszolgálónkat, és folyamatosan telepítsen minden RedHat-frissítést, viszont hagyja figyelmen kívül mindazokat az RPM-eket, melyeket nem kértünk első telepítéskor. Emellett beállítottuk, hogy az összes saját RPM-csomagunkat is folyamatosan frissítse.



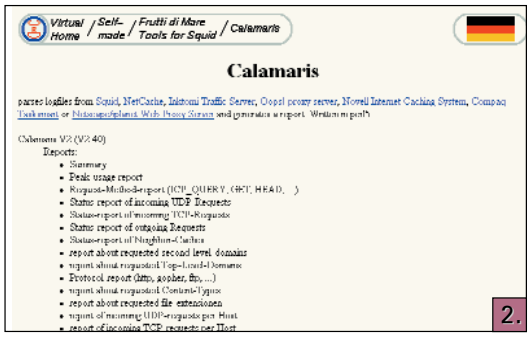
1.

Egy kis gondunk azért akadt, mivel az autorpm nem képes önműködően kezelni az előforduló körkörös függőségeket. Igaz, úgy tűnik, ez inkább az adott RPM-ek hibája, nem pedig az autorpm hiányossága.

Vissza a telepítéshez

Ez a gond a csomagokkal a telepítésnél is visszaköszönt. Nagyon örültünk neki, hogy a telepítési időt tíz perc alá sikerült szorítanunk, és mindössze néhány percet vett igénybe a régi rdist fájlok átalakítása végrehajtása is. Most azonban néhány gépen több mint tíz percet vett igénybe az új csomagok telepítése, és emberi közbeavatkozás is szükségessé vált a beállítások befejezéséhez. Ez különösen a Squid esetében volt nehéz számunkra, ahol a saját beállítófájlunknak egy újabb Squid-változatra volt szüksége.
Magánál a telepítésnél is felmerült ugyanez a nehézség. Nem volt arra lehetőség, hogy a közbeavatkozásunk nélkül fejeződjön be a helyi telepítés, amennyiben frissítettük valamelyik programot. Ez nagyon fontos kérdés volt számunkra. Előfordult néhányszor, hogy valamilyen hibát tapasztaltak a távoli irodában és mi nem voltunk elérhetők. Ezekben az esetekben távolról szerettük volna elérni az irodát, és egyszerűen újratelepíteni a proxykiszolgálójukat, esetleg egy új gépkiépítésen. Ez viszont csak akkor működik, ha nem kell beavatkoznunk a folyamatba.
Végül visszatértünk a kezdetekhez, és újra megfontoltuk az egyik alapfeltételezésünket – ugye azt mondtuk, hogy egyszerű RedHat CD-ket szállítottunk, majd KickStarttal, cfengine-nel, és autorpmmel hangoltuk be a rendszert. Most úgy határoztunk, inkább elkészítjük a saját Linux-változatunkat, mely RedHat alapokon nyugszik, de tartalmazza az általunk készített új vagy felfrissített RPM-csomagokat is, és néhány beállításfájlt. Az alapötlet az volt, hogy a kezdeti telepítés hozzon létre egy működő proxykiszolgálót, és később az önműködő időzített héjprogramok elvégzik a szükséges javításokat. A saját változatunk meglehetősen sikeresnek bizonyult és egyik napról a másikra sikerült felépítenünk új RedHat kiadásból. Vettük az alap-RedHatet és számos RPM-et törültünk a fából. Természetesen ez nem volt különösebben tudományos alaposságú munka, nem töröl-

tünk minden RPM-et, csak azokat, amelyek a legtöbb helyet foglalták el. Ezután a fába illesztettük a saját csomagjainkat, átalakítottunk pár vezérlőszerkezetet, és már írtuk is a CD-t.
Mivel az új kiadványunk tartalmazta a cfengine-t és az autorpmet, beállíthattuk a KickStart utótelepítést úgy, hogy futtassa le ezeket a folyamatokat a telepítést követő első újraindítás alkalmával. Ezáltal az új gép gyorsan naprakésszé válhatott az új beállításoknak megfelelően. Csakhogy mikor átálltunk a RedHat 6.2-re, egy KickStarttal kapcsolatos érdekes dologba ütköztünk. Az új változat CD-ről telepítése közben nem feltétlenül kérdezi meg az IP-címet és más hálózati beállításokat a felhasználótól, amennyiben azok nincsenek jelen a KickStart beállítófájljában. A kissé megváltoztatott leírás figyelmes átolvasása után megtudhattuk, hogy ez a szabadság növelésért történt, nekünk viszont komoly fejfájást okozott. Végül átírtunk egy részt az Anaconda telepítőben, így visszaállítottuk az eredeti viselkedést.



Biztonsági mentés és helyreállítás

Több különféle mentési és helyreállítási módszert is számításba vettünk, végül úgy döntöttünk, hogy a legegyszerűbbet választjuk, vagyis nem készítünk mentést. Mikor megnéztük a proxykat, észrevettük, hogy csak a naplóadatok és gyorstárszerkezetek különböznek az egyes gépeken. A naplóadatok időszakonként egy központi helyre másolódtak további vizsgálatok végzéséhez, a gyorstár pedig, bár értékes, időnként nyugodtan kitörölhető és újra létrehozható.
Az általunk alkalmazott linuxos proxyknak mostanáig nem volt semmilyen nagyobb szolgáltatáskihagyása vagy alkatrészhibája, de ha ilyesmi történik, a rendszer újratelepítését javasoljuk. Alkatrészhiba esetén bízunk benne, hogy minden irodában akad egy tartalék PC, amin a telepítést meg lehet ismételni.
Végeredményképpen nemcsak hogy nem kellett mentéseket végeznünk, de rugalmas alkatrészkészletről vagy tükröző módszerekről sem kellett gondoskodnunk. Tulajdonképpen a különleges alkatrészek használata csökkentené a rugalmasságunkat, hiszen nem biztos, hogy lenne tartalék alkatrész a távoli irodában.
Úgy számítjuk, hogy leállás esetén a távoli iroda megközelítőleg húsz perc alatt helyre tudja állítani a szolgáltatást, feltéve, hogy van a közelben egy felhasználható PC. Megpróbálkoztunk a böngészők beállításával is, hogy az egész folyamatot átlátszóvá tegyük az asztali PC-t működtető felhasználók számára. Ez egy különösen jól kihasználható megoldást adott a kezünkbe.

Ügyféloldali megfontolások

A régi proxybeállítás abban bízott, hogy az ügyfél közvetlen módon hivatkozik a névvel azonosított proxykiszolgálóra. Azokon a helyeken, ahol egynél több proxykiszolgáló futott, a proxykiszolgálóként használt gépnév egy ***lbnamed*** által kezelt tartományban volt. A mi változatunkat kissé módosítottuk, de még így sem felelt meg maradéktalanul az igényeinknek. Az lbnamed az rpc.rstatd-t használja, hogy a listában található gépek terheltségadatait meghatározza, majd pedig a súly függvényében különféle gépneveket ad vissza, ezáltal biztosítva egy korlátozott képességű terheléselosztást. A gyakorlatban azonban többnyire mégsem osztja el hatékonyan a terhelést, annak ellenére, hogy tartalmaz egy olyan hasznos tulajdonságot, miszerint a nem üzemelő gépeket törli a listából. Sajnos, ennek a hasznos szolgáltatásnak sokat levon az értékéből az a tény, hogy a gépek súlyozásánál csak a terhelést (load) veszi figyelembe (az alapváltozatban). Ha a Squid kiszolgáló leáll, a gép terheltsége általában csökken, ezáltal a meghibásodott gép a lista tetejére kerül. C-ben létezik ugyan egy módszer arra, hogy a terhelésen kívül néhány más értéket is lekérdezzünk, de az ilyen irányú próbálkozásaink nem bizonyultak túl gyümölcsözőnek. Amikor telepítettük, az általános benyomásunk az volt, hogy valamivel azért sikeresebb, mint amilyen az egyszerű DNS-címforgatás lett volna.
A kipróbálást a három új Linux-kiszolgálónkon hajtottuk végre, és az egyik vizsgált tényező az volt, hogy a válaszidőnek csökkennie illene, ha egy olyan adatra küldünk http-kérést, ami nagy valószínűséggel megtalálható az adott proxy gyorstárában. A lekérdezéseket kezelő proxy-gyorstárak értelmes kiválasztásának ötlete igen könnyen megvalósítható, a legtöbb böngésző által támogatott Proxy Auto Configuration (PAC) fájl segítségével. Ez a lehetőség azon alapul, hogy létezik valahol egy olyan proxykiszolgáló, ami képes PAC-fájlokat visszaküldeni.
A mi feladatunk teljesen egyszerűvé vált, mivel valaki már készített egy példakódot a PAC-fájlokhoz, ami kiegyensúlyozta a terhelést néhány proxykiszolgáló közt. A Sharpnál készült Super Proxy Script nevű munkát vettük alapul, amely URL-kategorizálást használt. Ez egy nagyon egyszerű, de eredeti ötlet, ami elemzi az elérni kívánt URL-t, majd visszaadja a használandó proxy nevét. Ennek segítségével egyenletesen osztja el a terhelést a kiszolgálók közt, de azonos URL-re vonatkozó lekérdezések mindig ugyanazt a proxyt használják. Kihasználtuk a PAC-fájlok azon tulajdonságát is, hogy proxykiszolgálók listáját is vissza tudják adni. Ennek eredményeképpen, ha a listában elsőként szereplő kiszolgáló nem válaszol, a böngésző önműködően a következőt fogja használni, a felhasználó számára észrevétlenül.
A központ gépei esetében két vagy három proxyt tartalmazó listákat adtunk vissza a telepen belüli helyzetüktől függően. Azokon a helyeken, ahol csak egyetlen proxy volt elérhető, nem használtunk URL-elemzést, és csak két proxynevet adtunk vissza, a helyi kiszolgáló nevét, valamint hiba esetére, a központi kiszolgálóét. A központi kiszolgáló használata hiba esetén lehetővé tette számunkra a legnagyobb rugalmasságot. Ha a helyi proxy le is állna, a hozzá tartozó böngészők észrevennék a hibát, és a központi kiszolgálót vennék igénybe. Az ügyfelek 30 percenként ellenőrzik (MSIE és Netscape esetén), hogy helyreállt-e már a kapcsolat az eredeti kiszolgálóval, és ha így van, akkor visszatérnek a használatához.
Mivel néhány ügyfelünk Microsoft Explorer 5.0-t használ, a proxy.pac fájlt wpad.dat névre neveztük át. Így lehetővé vált, hogy a „beállítatlan” IE5-ügyfelek egy http://wpad.helyi.tartomány/wpad.dat formátumú URL-re történő WPAD rendszerű keresés használatával önműködően megtalálják a wpad.dat fájlt. A WPAD használata nem volt feltétlenül szükséges, de hasznosnak találtuk. A naplófájljaink elemzése rámutatott arra, hogy segítségnyújtó irodánkat mentesíthetjük jó néhány vándorló felhasználó hívásától, akik máskülönben segítséget kértek volna a kézi proxybeállítás elvégzéséhez, amikor másik irodába mentek át.
Jelenleg Apache webkiszolgálót használunk a PAC-fájlok visszaküldéséhez, és az Apache futhat a helyi proxykiszolgálón, vagy bármely más helyileg elérhető webkiszolgálón. Lehetséges, hogy egy lecsupaszított webkiszolgáló használata – ilyeneket Perlbe is átültettek – biztonságosabb lenne, és kevesebb fejfájást okozhatna. Ezt a megközelítést még nem vizsgáltuk meg.
A WPAD kiszolgálóink jelenleg több DNS-bejegyzéssel is rendelkeznek, így ha valamelyik WPAD kiszolgáló elromlana, még mindig rugalmasak maradhatunk. Azokon a helyeken, ahol csak egyetlen WPAD kiszolgálót alkalmazunk, ott az új ügyféloldali folyamatra

bízzuk, hogy az előzőleg gyorstárazott beállításokat használja fel. Van viszont egy kis hiba ebben a megközelítésben, abban az esetben, ha a távoli helynek több proxykiszolgálója is van: az URL-elemzés ugyan gondosan kiválasztja a megfelelő helyi proxykiszolgálót, az viszont egyszerűen körbeküldi a lekérdezést a három központi rendszeren. Kihasználhatnánk a Squid néhány képességét a gyorstár-összesítők (digest) egymás közti cseréjére, s így a helyi proxy arra a központi kiszolgálóra küldené az adatot, ahol a legnagyobb valószínűséggel érne el találatot, de a gyakorlati tapasztalatok azt mutatják, hogy a WAN hálózat sávszélesség-felhasználásának növekedése miatt ez nem igazán hatékony megoldás. Ehelyett hagyjuk végrehajtani a körbekérdezést a központi kiszolgálókon, és a központi kiszolgálók közötti gyorstár-összesítők cserével érjük el a találatot, amennyiben lehetséges.

Fejlesztési lehetőségek

Említettük már, hogy a proxykiszolgálóinkon lehetőleg egy viszonylag korszerű RedHat-változatot szerettünk volna fenntartani. Bár a kisebb beállítás-változtatásokat a cfengine-re és az autorpm-re bíztuk, nem hisszük, hogy a teljes OS hálózaton keresztül történő frissítése megvalósítható lenne.

Ehelyett inkább új telepítések szétküldését terveztük a távoli irodák számára, ahol azokat igény szerint feltelepíthetik. Úgy számoltuk, évente megközelítőleg háromszor-négyszer fordul elő ilyen eset. Mivel ilyen tiszta és irányított telepítési eljárásunk van, viszonylag kevés hátránnyal jár a gyakori frissítés. Mivel a távoli iroda megtarthatja az előző változat anyagát, az előző változatra való visszatérés sem okoz túl nagy nehézséget.

Nagyon fontosnak tartottuk ezeket a frissítéseket, mivel igen sok hibát tapasztaltunk a korábbi HP-UX proxyk működése során, amelyek a programfrissítések és javítások hiányából eredtek. Például láttunk HP-UX gondokat, pedig ezekre lettek volna foltok, és a Squid és pár eszköz régebbi változatát futtattuk, mivel csak régebbi Perl-változatunk volt.

Az új változatra átállás akár rendes munkanapon is történhet a távoli irodában, az ehhez szükséges húsz percre az ügyfelek egyszerűen a központi hibaelhárító kiszolgálókat fogják használni; olyan irodák esetén, ahol több kiszolgáló is van, a saját helyi hibaelhárító kiszolgálóikat használhatják.

Rendszerfigyelés

Elemzés szempontjából a leghasznosabb adat a gyorstár tevékenységéről készített kimutatás. Ezt az adatot az MRTG (Multi Router Traffic Grapher) és a Squidbe épített SNMP-ügyfél használatával érjük el. Kicsit kényelmetlen ugyan beállítani, de hasznos ábrákat tudunk vele készíteni a proxyk teljesítményéről. Sok számadatot gyűjtünk be a proxykról, ami a belső weboldalról elérhető. Van olyan kódunk is, amely áttekintőlapokat készít az összes kiszolgálóról. A kiszolgálóink NIS-térképén végigmenve ez az eljárás gyors átnézeti képet is szolgáltat néhány számadatról. Minket különösen a gyorstártalálatok és tévesztések folyamata érdekel, illetve az összesített kérelmek növekedése.

Pillanatfelvételek előállítására és adatelemzésére a Calamaris eszközt (lásd a Kapcsolódó címeket) is használjuk.

Van egy másolatunk a HP OpenView hálózatkezelő termékekből is. Jelenleg csak a gépek állapotának figyelésére használjuk, de tervezzük a távoli gépeken futó Squid-programok felügyeletét is, hogy figyelmeztessen bennünket, ha bármelyik meghibásodna.

Elkezdtük cronból ötpercenként futtatott cfengine-t használni arra, hogy ellenőrizzük a különféle folyamatok állapotát, és szükség esetén megkíséreljük az önműködő újraindítást.

Továbbá a helyi irodákhoz internethasználati jelentéseket továbbítunk. Jelenleg a naplófájlokat a SAS adathalmazba gyűjtjük, és a SAS jelentéskészítő eszközt használjuk fel a jelentésekhez, van ezenkívül pár olyan termékünk is, mint amilyen a SAS IT Service Vision vagy a SAS WebHound, amelyek hasonló forgalomelemzésre képesek. Ezek nagyon hatékony eszközök, és az adatok sokkal teljesebb vizsgálatát teszik lehetővé. Elégedettek vagyunk a linuxos megoldásunk megbízhatóságával és teljesítményével.

Semmi kétségem afelől, hogy az alkatrészárak csökkentésével lehetőség nyílik arra, hogy olyan helyekre is telepítsünk proxykiszolgálókat, ahol eddig nem volt gazdaságos nagyobb rugalmasságot nyújtani a többi hely felé. Mivel az új összeállításunk rugalmasabb, és mindent egybevetve jobban beállított, mint az előző volt, sokkal kevesebb gondunk van most, mint a régi proxyrendszerekkel volt.

Az eddigi leggonoszabb hiba, amit a működés során tapasztaltunk, a fájlrendszerhiba volt. Az egyik távoli proxy áramkimaradás miatt leállt, és nem tudott elindulni fájlrendszerhiba miatt. Ideiglenes megoldásként átalakítottuk az indítókódot, hogy legyen elnézőbb az ilyen rendszerleállásokkal szemben, hosszú távú megoldás viszont inkább valamilyen jegyzőkönyvező fájlrendszer használata lehetne, amilyenek mostanság kezdenek hozzáférhetővé válni.

Azt vettük észre, hogy egyre több helyi kapcsolat épült ISP-szolgáltatók felé, ezért majd csiszolnunk kell az összeállításunkon, hogy támogassa a HTTP-alapú víruskeresést irodaszinten és az internet-proxy közvetlen kapcsolatán egyaránt. Kiegészítjük a cfconfig és autorpm eszközöket néhány új feladattal, hogy telepítsék és állítsák be az új részegységeket.

Mivel most már van egy receptünk a saját Linux-kiadásaink elkészítéséhez, valószínűleg elkezdünk foglalkozni egy általánosabb, belső felépítés gondolatával. Ez valamilyen szinten biztosan megtörténik, most, hogy a SAS-termékek már Linuxon is elérhetők. Azt reméljük, hogy a belső használatra szánt szabványos kiadások készítése bizonyos beállításbeli szabadságot ad majd. Valószínűleg átgondoljuk azt is, milyen más szolgáltatásokat lehetne haszonnal futtatni a Linuxon. Például átrakhatnánk néhány DNS/BIND szolgáltatást Linux alá.

Ian Spare
jelenleg tanácsadóként dolgozik a németországi SAS Intézetnél. Idejét leginkább hódeszkázással vagy sílécen szereti tölteni, de amikor éppen nem a havon tartózkodik, tanácsokat ad, vagy Unix- és Linux-rendszereket felügyel szerte Európában, a Közel-Keleten és Afrikában. Nincsenek gyermekei, viszont van három macskája.

Kapcsolódó címek

cfengine
➲ http://www.gnu.org/software/cfengine

Squid
➲ http://www.squid-cache.org *(1.kép)*

autorpm
➲ http://www.kaybee.org/~kirk/html/linux.html

lbnamed
➲ http://www.stanford.edu/~riepel/lbnamed

Proxy Auto Configuration
➲ http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html

Sharp „Super Proxy" Script
➲ http://naragw.sharp.co.jp/sps/index.html

Calamaris Squid Reporting Tool
➲ http://cord.de/tools/squid/calamaris *(2. kép)*

Rendszergazda-képzés Linuxszal

Egy főiskolai gépteremben rendhagyó képzés zajlik.

A rendszergazdai teendők ellátása minden operációs rendszer esetében nagyon fontos, azonban a legtöbb egyetemen és főiskolán nem foglalkoznak rendszergazdák képzésével. Vajon akkor hogyan tanulhatják meg a diákok a szükséges fogásokat? A válasz egyszerű és közhelyszerű: maguktól. A következő ésszerű kérdés: mi teszi lehetővé azt, hogy valakiből rendszergazda lehessen? A válasz: egy rendszergazdának tisztában kell lennie az operációs rendszerek és a hálózatok alapvető működésével. Egy hagyományos, gyakorlati programozást is tanító oktatási rendszerrel ellentétben, a rendszergazdák képzésénél inkább az elméleti összefüggésekre kell helyezni a hangsúlyt. Például egy merevlemezes gyorstár beállításához nem kell ismerni a laptáblázatok kiosztását.

A Grand Valley Állami Egyetemen indítottunk egy tantárgyat, ennek keretében a diákok az operációs rendszerekről és a hálózatkezelés alapjairól tanulhatnak, és az egész oktatást a rendszergazdai teendőkre alapoztuk. Hallgatóink végzős informatikusok, akik egyébként sehol nem tanulnának az operációs rendszerek és a hálózatkezelés szabályairól és elveiről. Az órákon a legkülönbözőbb témák kerülnek terítékre: a felhasználók és csoportok kezelése, a fájlmegosztás és a folyamatok. A hálózatkezelés területéről pedig az alkalmazásréteg-protokollok, az átviteli réteg és a hálózati eszközök beállítása került a tantervbe.

A tantárgy két fő részből áll: az elméleti oktatásból, melynek során a diákok az operációs rendszerek és a hálózatkezelés elméletéről hallgathatnak, és a gyakorlatokból, ahol a tanult elvekkel a gyakorlatban szabadon kísérletezhetnek. Ebben a cikkben egy rövid ismertetést adunk arról, hogy az oktatást miként segíti a Linux.

Az EOS Linux gépterem

Az operációs rendszerekkel való kísérletezésre felállított EOS gépterem huszonnégy Pentium III-as gépből áll, ezekben egyenként 128 MB memória, 10 GB merevlemez, egy hajlékony- és egy ziplemezmeghajtó van, és mindegyiken RedHat 6.2 fut. A gépterem nemcsak kísérleti terep, hanem mindennapi munkakörnyezet is: a gépeket a végzős informatikusok és a programozók használják, de más szakok hallgatói is idejárnak kipróbálni az operációs rendszereket. Tehát a gépterem nem tisztán kutatóegység, hiszen egészen hétköznapi teendőkre is használják a gépeket. Éppen ezért fel sem merült az, hogy félévenként huszonnégy diáknak adjunk rendszergazdai jogosultsága a gépekhez. Így viszont azzal a nehézséggel szembesültünk, hogy a hallgatóknak a legalapvetőbb teendők elvégzéséhez is rendszergazdaként kellene belépniük az operációs rendszerbe.

A megoldás a zipmeghajtók használata lett, ennek köszönhetően minden hallgató saját Linux-környezetében dolgozhat. Mindenki készít egy indítólemezt és egy ziplemezt, mely az alapfájlrendszert tartalmazza. Az óra kezdetén a diákok leállítják a gépüket, behelyezik az indítólemezt és az alaplemezt, majd újraindítják a rendszereket. Innentől kezdve mindenki saját Linux-változatával dolgozik, ezt természetesen rendszergazdaként használhatja, és kényelmesen elvégezheti rajta az aznapi feladatokat. A nap végén egyszerűen leállítják a gépeket, kiveszik a lemezeket, és az utánuk érkező felhasználók már a merevlemezről dolgozhatnak, a szokásos RedHat 6.2 környezetben.

Az indítólemez elkészítése

Jelenleg a 2.2.13-as rendszermagot használjuk, ez ráfér egy hajlékonylemezre és semmiféle módosításra nincsen szükség. A rendszermag egyedi beállítására azonban szükség van, két okból is.

Az első a SCSI-emuláció beállítása, ehhez a CONFIG_CHR_DEV_SG és a CONFIG_SCSI értékét igazra állítjuk. A zipmeghajtók ugyanis IDE csatolósak, és ezeket SCSI-emulációval kell működtetnünk, mert tapasztalataink szerint az IDE-meghajtó nem kezeli jól a nagy fájlokat.

A rendszermag beállításának másik fontos oka a merevlemez letiltása. Említettük, hogy a laboratórium gépeinek merevlemezein RedHat 6.2 található. Ha nem tiltanánk le a merevlemezek elérését, a hallgatók a zipről történő rendszerindítás után a `mount` paranccsal egyszerűen befűzhetnék a helyi merevlemezt, és innentől kezdve teljhatalmat élveznének felette (akár a rendszergazda jelszavát is megváltoztathatnák). A merevlemez letiltását két változó (a CONFIG_BLK_DEV_IDE és a CONFIG_BLK_DEV_HD_IDE) hamisra állításával oldottuk meg.

A rendszermag további beállításai lehetővé teszik a hálózati eszközök, például a SysV init stb. engedélyezését. Miután a rendszermagot beállítottuk, egyszerűen lefordítjuk azt. A hajlékonylemezen a mke2fs programmal először létrehozzuk az ext2 fájlrendszert, majd a lefordított rendszermagot rámásoljuk. A lemezen egy indítórészt is létre kell hoznunk (`cp /boot/boot.b /mnt/floppy`) és a LILO-t az alábbiak szerint kell beállítani:

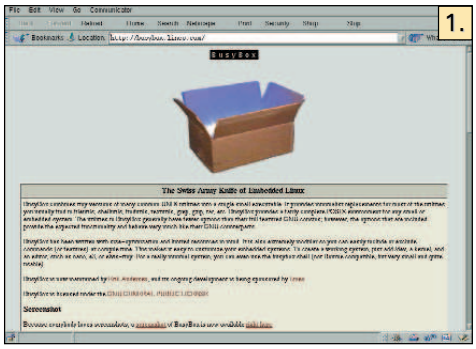
```
boot=/dev/fd0 map=/mnt/floppy/map
install=/mnt/floppy/boot.b
prompt
compact
```

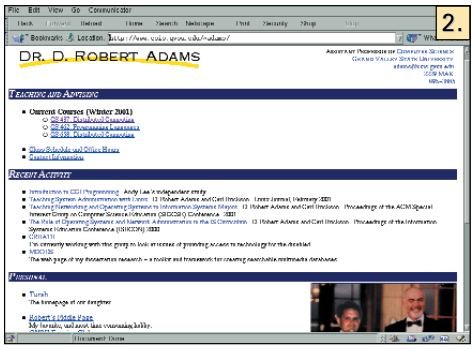
```
timeout=50
image=/mnt/floppy/vmlinuz
        label=linux
        root=/dev/sda1
        read-only
```

Ez a LILO-beállítás indíthatóvá teszi a lemezt, és a /dev/sda1-et állítja be gyökérlemezként. Mivel SCSI-emulációt használunk, ezért a /dev/sda1 (az első SCSI lemez) a ziplemez.
Ezt követően a `/sbin/lilo -C /mnt/floppy/lilo.conf` paranccsal telepítjük az új LILO-fájlt.

Az alaplemez elkészítése

Az alaplemez a Slackware 7.0-s Linuxon alapul. Azért választottuk a Slackware-t, mert pontosan szabályozható, hogy mely csomagok kerüljenek telepítésre, és így a rendelkezésre álló 100 megabájtos helyet tökéletesen kihasználhatjuk. Az óra anyagából kiindulva az a, ap és n csomagokat telepítettük, az alábbi parancsok segítségével:





ext2 lemezrészt hozunk létre a ziplemezen

```
# fdisk /dev/sda1
```

fájlrendszer létrehozása a ziplemezen

```
# mke2fs /dev/sda1
```

befűzzük a ziplemezt

```
# mount /dev/sda1 /mnt/zip
# cd /mnt/zip
# tar -zxvf /tmp/slackware/a1/aaa_base.tgz
# sh install/doinst.sh ; rm -rf install
```

Az utolsó három lépést minden szükséges csomag esetében meg kell ismételnünk.
Sajnos, a helyszűkéből adódóan sok hasznos csomagot ki kellett hagynunk. Például a C/C++-t magában foglaló d, és a rendszermag forrását tartalmazó k csomagokat is, hogy csak a legfájóbb hiányosságokat említsük. Ezen a helyzeten természetesen változtatni szeretnénk: azt tervezzük, hogy egy nagyobb cserélhető lemezt alkalmazunk a feladatra. A Lineo nevű, beágyazott Linuxszal foglalkozó vállalat Linux-változata kis terjedelmű, és minden szokásos linuxos eszközt rendelkezésünkre bocsát. A BusyBox pedig egy nyílt forrású kezdeményezés, melynek célja, hogy a hagyományos linuxos eszközöket egyetlen, kisméretű futtatható fájlba sűrítse (lásd Kapcsolódó címek).

Kapcsolódó címek

The Role of Operating Systems and Network Administration in the IS Curriculum. Az Information Systems Education Conference (ISECON) anyagából, 2000, D. Robert Adams és Carl Erickson.

Teaching Operating Systems and Networking to Information Systems Majors. A Special Interest Group of Computer Science Education (SIGCSE) Technical Symposium anyagából, 2000. D. Robert Adams és Carl Erickson.

A CS437 honlapja: *(2. kép)*
➲ http://www.csis.gvsu.edu/~adams/CS437

BusyBox: *(1. kép)*
➲ http://busybox.lineo.com/
Busy Box: The Swiss Army Knife of Embedded Linux, Linux Journal, 2000. október, szerző: Nicholas Wells.

Önműködő folyamat

Hallgatóink a félév első óráján elkészítik indító- és alaplemezeiket. Természetesen az első óra alkalmával nem mindegyikük tudása áll olyan szinten, hogy erre önmaguktól képesek legyenek. Ezért kitaláltunk egy önműködő rendszert, ennek segítségével néhány parancs kiadásával elkészíthetők a lemezek. Először is a lemezek működő változatát fájlba mentjük az alábbi parancsokkal:

```
dd if=/dev/fd0  of=floppyimage
dd if=/dev/sda1 of=zipimage
```

A hallgatóknak tehát csak a `dd` parancsot kell használniuk a fájlok kiírásához:

```
dd if=floppyimage  of=/dev/fd0
dd if=zipimage     of=/dev/sda1
```

Összefoglalás

A Linux nálunk nagyszerűen bevált a rendszergazdák képzésében. A ziplemezek használatával minden diákunk függetlenül dolgozhat saját kis Linuxával, anélkül, hogy a gép merevlemezén található rendszernek bármi baja esne. Bár egy ziplemezen csupán 100 MB fér el, a nyílt forráskódnak köszönhetően a Linux-változatot sikerült egyetlen ziplemezre sűrítenünk, sőt, a gép merevlemezének módosítását is egyszerűen megakadályozhattuk. A gépeket sokan távolról is szeretnék elérni, és az alkalmazott módszernek egyetlen hátránya az, hogy a ziplemezről való újraindításkor ez lehetetlen. Úgy hidaltuk át a nehézséget, hogy a nem ziplemezt használó gépeket megkülönböztető névvel láttuk el. A felhasználókat pedig figyelmeztettük arra, hogy a ziplemezes gépek nem érhetők el folyamatosan a hálózaton keresztül. A további részletek iránt érdeklődők az órajegyzeteket, a hallgatók feladatait és az egyéb programokat a képzés honlapján megtalálhatják.

D. Robert Adams
a Grand Valley State University segédprofesszora. Kutatási és érdeklődési körébe tartozik az objektumközpontú programozás, a Palm számítógépek programozása és a különféle programnyelvek.

Carl Erickson
a GVSU segédprofesszora. Ma jelenleg a XiphNet, Inc. programszerkezeteket kutató csapatában dolgozik. Szakterülete az elosztott és az objektumközpontú programozás.

Ez aztán a fejlődés!

Cikkünkben a vi egy okosabb változatát mutatjuk be.

A Linuxhoz tartozik egy vim nevű program, mely a vi szerkesztő bővített változata. Nemcsak a vi utánzására képes, hanem szó szerint több száz további szolgáltatást is nyújt, ilyenek például a súgórendszer, a több ablak, a programfordítás, a hibajavítások, a fejlett keresőrendszer és még sorolhatnánk.

Az első lépések

Alapértelmezés szerint a vim vi-megfelelő módban indul el. Ez azt jelenti, hogy az új lehetőségek nagy része nem használható. Bekapcsolásukhoz egy `$HOME/.vimrc` nevű fájlt kell létrehoznunk.
Az *1. listán* egy példafájl látható, melyet úgy is létrehozhatunk, hogy a létező .exrc fájlt lemásoljuk és átnevezzük. Nulla bájt hosszúságú fájllal is működik a dolog.
A vim szerkesztőnek két változata létezik: az egyik, amikor a vim parancs a szerkesztő konzolablakában indul el. Ennél a változatnál abban a terminálablakban folyik a szerkesztés, amelyben a parancsot adtuk ki. A gvim viszont saját ablakot hoz létre. Ez utóbbi módszer hasznosabb, hiszen így a konzolos változatban elérhetetlen menüket és az eszköztárat is használhatjuk.

A súgórendszer

A vim egyik leghasznosabb újítása a bármikor elérhető, beágyazott súgórendszer. A `:help` parancs jeleníti meg a súgóablakot. A parancs után annak a parancsnak a nevét kell beírnunk, melyről adatokat kívánunk szerezni. A `:help /` például a / (keresés) parancsot ismerteti (*1. kép*).
A szövegben a hagyományos szerkesztőparancsokkal (h, j, k, l, Page Up, Page Down stb.) mozoghatunk. Eközben néha az alábbihoz hasonló sorokkal találkozhatunk:

```
/<CR> Search forward the [count]'th latest used
pattern |last-pattern| with latest used |{offset}|.
```

A függőleges vonalak (`|`) közé zárt szöveg vim hiperhivatkozás. Álljunk ezek egyikére (mondjuk a `|last-pattern|`-re), nyomjuk le a Ctrl+] billentyűket,ezzel az adott elemre ugorhatunk.
Az ugrás előtti helyre a Ctrl+T lenyomásával térhetünk vissza.
A súgórendszerből való kilépéshez a vim szokásos kilépés parancsait (`:q` vagy `ZZ`) használhatjuk.

Fájl létrehozása

A vim új lehetőségeinek erejét az alábbi programszöveg beírásával magunk is megtapasztalhatjuk:

```
#include "even.h"
int even(int value)
{
  if (value & 1) == 1)  // Itt hibás a zárójelezés.
       return (1);
  return (0);
}
```

Az első észrevehető újdonság, hogy a szöveg színes. A programlista minden elemtípusa (kulcsszó, karakterlánc, állandó, fordítási elv stb.) más-más színt kap. Ezt a `:syntax on` parancs engedélyezi a .vimrc fájlban. A `:syntax off` kikapcsolja e megkülönböztetést. Egy másik hasznos dolog, hogy nincsen szükség behúzásra (azaz a Tab billentyű nyomogatására a behúzni kívánt sorok elején). Mindezt a `cindent` értéknek köszönhetjük:

1. lista Egy .vimrc példa

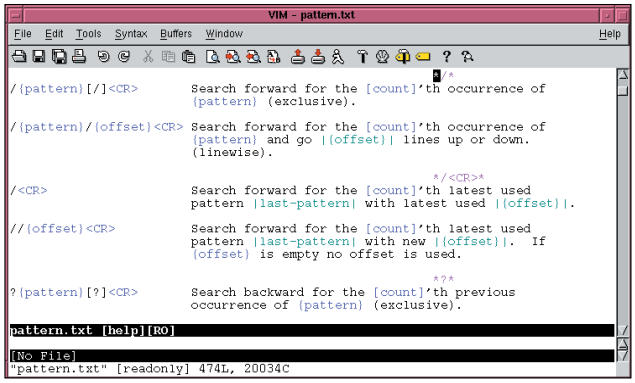
```
:syntax on
" A vimben a sorok összetartozását a \ jelzi.
:autocmd FileType *
\       set formatoptions=tcql nocindent
comments&

:autocmd FileType c,cpp
\       set formatoptions=croql cindent
\       comments=sr:/*,mb:*,ex:*/,://

:set autoindent
:set autowrite

:ab #d #define
:ab #i #include
:ab #b /*************************************
:ab #e **********************/
:ab #l /*-------------------------------------*/

:set shiftwidth=4

:set notextmode
:set notextauto

:set hlsearch
:set incsearch

:set textwidth=70
:set guioptions=-v
```

```
:autocmd FileType c,cpp     :set cindent
```

Ez csak a C és C++ fájlokra vonatkozik. Sajnos, cikkünk terjedelme miatt nem ismertethetem az összes automatikus lehetőséget, de a `:help autocmd` parancs kiadása után mindent megtudhatunk.
Egy megjegyzés: a .vimrc példafájlban található parancsok kicsit összetettebbek, mint a fent említett alakok, de végeredményben ugyanazt művelik.

Visszavonás és ismétlés

Akkor kezdjünk el szórakozni egy kicsit. Álljunk egyik return parancs „r” betűjére és írjuk be, hogy xxxxxx, mire a return eltűnik. Most nyomjuk le az u billentyűt az utolsó változtatás visszavonásához. Ekkor megjelenik az „n”.
A régi vi szerkesztőben csak egyszintű visszavonásra volt lehetőség,

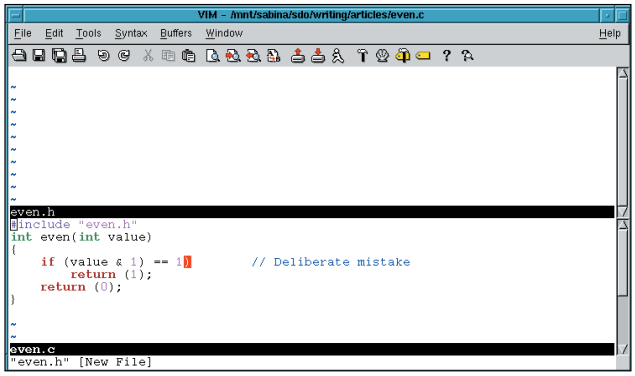


1. kép A vim súgóablaka

a vimben viszont elvileg akárhány módosítást visszavonhatunk. Nyomjunk megint „u”-t, és nicsak, az „rn” is megjelent. Az „u” többszöri begépelésével az egész „return” szót visszavonhatjuk.
És a visszavonás visszavonása? Az is megy: a használandó billentyűk a CTRL+R. Ezt néhányszor ismételve visszavonhatjuk a törlést, mire a „return” fokozatosan ismét eltűnik.

Több ablak

Ha már C fájlt szerkesztünk, akkor fejlécfájlra is szükségünk lehet. Milyen szép is lenne, ha a prototípust egyszerűen bemásolhatnánk a C fájlból a fejlécbe. A vi-jal ezt nem lehet, a vimmel viszont gyerekjáték. Először is töltsük be mindkét fájlt a szerkesztőbe. A `:split even.h` parancs kettévágja az éppen használt ablakot, hatására a felső részbe az even.h, az alsóba pedig az even.c kerül (*2. kép*).

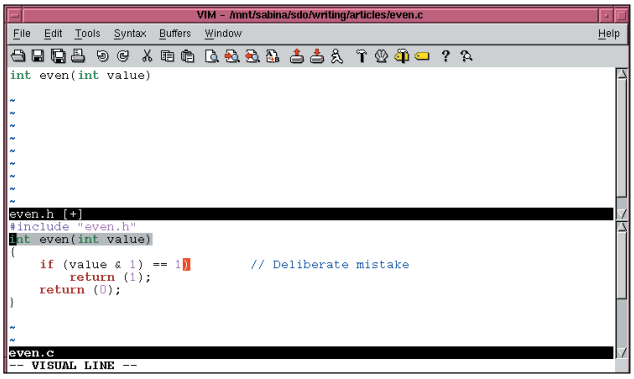


2. kép Egyszerre több ablakban is dolgozhatunk

Ha az alsó ablakból a felsőbe kívánunk átlépni, akkor nyomjuk le a CTRL+W, J billentyűket. A felső ablakra a CTRL+W, K billentyűkkel válthatunk. Az ablak bezárásához a `ZZ` vagy a `:quit` parancsot használhatjuk.

Másolás és beillesztés megjelenítő üzemmódban

A vimnek néhány új üzemmódja is van. Megjelenítő módban a szerkeszteni kívánt szövegrész kijelölése után egy szerkesztőparancsot kell beírnunk. Az alábbi példában az `even` függvény első sorát lemásoljuk, és a felső ablakba illesztjük.
Először is lépjünk az alsó ablakba (CTRL-W J) és álljunk az `even` függvényre. Ezt követően lépjünk át megjelenítő módba a `V` paranccsal. Ekkor az éppen szerkesztett sor kijelölődik, tehát ezt a sort fogja érinteni a beírandó szerkesztőparancs. A nyíl billentyűkkel léphetünk feljebb vagy lejjebb, és eközben a kijelölést is változtatjuk. Ha a kezdőpont fölé lépünk, akkor az afölötti rész jelölődik ki, ilyen egyszerű az egész.
A kijelölés megjelenítő módban soronként történik. Ha a `V` parancsot gépeljük be (karakterenkénti megjelenítő mód), akkor a szöveg minden egyes karakterével külön foglalkozhatunk. A megjelenítő üzemmód harmadik típusa blokkonként dolgozik, és ezt a CTRL-V billentyűkkel indíthatjuk. Ez utóbbi főleg táblázatokkal való munka esetén hasznos, hiszen így az oszlopokat, sorokat külön-külön kezelhetjük.



3. kép Sorok másolása megjelenítő üzemmódban

Akkor tehát a másoláshoz lépjünk soronkénti megjelenítő módba és jelöljük ki a kérdéses sort. Ezt az `y` paranccsal másolhatjuk át a tárolóba. Váltsunk át a felső ablakra, ahol a `p` paranccsal illeszthetjük be az imént másolt szövegrészt. Már csak egy `;` kell a sor végére, és kész is van a fejléc. A *3. kép* az említett parancsokat mutatja működés közben.

Programfordítás

Programunk fordításához egy Makefile is elkélne, ennek megírását azonban most vállalkozó szellemű olvasóimra bízom. Ha a Makefile a helyére került, akkor máris kihasználhatjuk a vim és a make összehangolásának lehetőségét. Mindössze a `:make` parancsot kell kiadnunk a vimből, és a szerkesztő elindítja a make programot és rögzíti annak kimenetét.
Programunkban azonban van egy hiba is:

```
even.c:4 parse error before '=='
```

A vim beolvassa a `:make` parancs kimenetét és látja, hogy a 4. sorban hiba van, tehát erre a sorra állítja a helyőrt, és a hibaüzenetet kiírja a képernyő alsó sorába, még a fájlokat is felcseréli, ha ezt a hiba megjelenítése megköveteli.
Így már igen könnyű a hibajavítás. A következő hibára a `:cn`, az előzőre a `:cp`, az elsőre pedig a `:cc` paranccsal ugorhatunk. Ha a fájl javítását befejeztük, és a következő fájl első hibájára szeretnénk lépni, akkor a `:cnf` parancsot kell használnunk.
A vim tehát igen okos, ha hibakeresésről van szó. Ha a fájl elejéről sorokat törlünk vagy beszúrunk, a vim akkor is emlékezni fog a hibák helyére. Ez a program egyik hatalmas előnye a vi-jal szemben, ahol az első néhány hiba kijavítása után könnyen elcsúszhatott az egész sorszámozás.

Egy használt függvény keresése

A Linux `grep` parancsa nagyszerűen alkalmas azon sorok megtalálására, melyben egy bizonyos függvény vagy annak meghatározása szerepel. Egyszerűen adjuk ki az alábbi parancsot:

```
$ grep -n even *.c
```

Ennek hatására a könyvtárban található összes `.c` kiterjesztésű fájl minden olyan sora megjelenik, mely tartalmazza az `even` szót, és a sor elején közli a sorszámot is.
A vimben is van egy `:grep` parancs, működése nagyon hasonlít

a `:make`-re. A Linux grep parancsát futtatja, figyeli a kimenetet és a megtalált helyek között a `:cn`, `:cp`, `:cc` és `:cnf` parancsokkal lépkedhetünk, akárcsak a `:make` esetében.

A behúzások javítása

Ha a `cindent` értéket bekapcsoljuk, a vim új szöveg beszúrása esetén megfelelően behúzza a sorokat. De mi a helyzet a már létező szöveggel? Itt lép be a képbe a vim `=` parancsa.
Ez a parancs egy szövegtömbön futtatja le a vim belső, a behúzást vezérlő programját de az equalprog lehetőséggel a feladatot akár egy külső programra is rábízhatjuk.
Nézzük, miként működik mindez egy szabványos C kódrészlet esetében. Az `=` parancsot kétféleképpen hívhatjuk meg. Az első módszer, hogy az `= {motion}</>` parancsot használjuk, a második, hogy előbb megjelenítő üzemmódba lépünk, kijelöljük a szövegrészt, majd kiadjuk az `=` parancsot.
A rész behúzásának átállítását kezdhetjük azzal is, hogy a szakasz első `{` jelére állunk. Most írjuk be az `=%` parancsot. Ezzel a vimet arra utasítjuk, hogy innentől kezdve húzza be a szöveget egészen addig, ahol a második parancsot eléri. Jelen esetben a második parancs a `%`, ennek hatására a vim a záró `}` jelre ugrik.
Ha mindezt megjelenítő módban kívánjuk elvégezni, akkor álljunk az első `{` jelre, majd lépjünk soronkénti megjelenítő módba a `V` paranccsal. Ezután álljunk a a záró `}` jelre, ehhez bármilyen ugróutasítást használhatunk, majd az `=` paranccsal húzzuk be a kijelölt részt.

Keresés

A vimnek akad néhány érdekes keresési lehetősége is. Az első ezek közül a szűkítő (más szóval növekményes) keresés, ezt a `:set incsearch` kapcsolja be. Kereséskor alapértelmezés szerint az egész parancsot be kell írnunk (például az include keresésekor a `/include`-t). A szűkítő keresés használata során azonban már az `i` beírása után az első i-vel kezdődő szóra ugrunk, az `n` után az első in-nel kezdődő szóra stb. Ahogy a szót karakterenként felépítjük, úgy közeledünk a keresett szóhoz.
Egy másik fontos újítás a kiemelő keresés (hlsearch). Ez annyit jelent, hogy a vim kiemeli a megtalált kifejezést, nemcsak az első előfordulást, hanem mindegyiket. Ez a lehetőség hasznos lehet például félregépelések megtalálásában, hiszen a vim a hibás szó minden előfordulását kiemeli. A kiemeléseket a `:nohl` paranccsal átmenetileg – a következő keresőparancs begépeléséig – ki is kapcsolhatjuk.
A vim a keresésekről adatbázist is készít. Tegyük fel, hogy már jó néhány keresést indítottunk és szeretnénk visszatérni egy régebbi eredményéhez. Ilyenkor a `/` paranccsal indítsunk egy keresést, majd a Fel, illetve Le billentyűkkel lépkedhetünk a régebbi keresések között.

Billentyűzetmakrók

A vim egyik leghatékonyabb parancsa a `.` (pont), mely az utolsó szerkesztési műveletet ismétli meg. E parancs azonban nem nyújt minden esetben megfelelő megoldást, hiszen csak egyetlen parancsot ismétel. De mi történik akkor, ha valami összetettebb feladatot szeretnénk elvégezni? Ekkor használhatjuk a billentyűzetmakrókat. Segítségükkel több parancsot rögzíthetünk egy tárolóba, majd ezeket végrehajthatjuk.
Hogy jobban megértsük a makrók működését, nézzünk most egy példát. Tegyük fel, hogy a következő sorokat kívánjuk módosítani:

```
stdio.h
time.h
unistd.h
stdlib.h
```

Mondjuk az a vágyunk, hogy ez a négy kifejezés #include alakra módosuljon (például `#include <stdio.h>`).

Egy megjegyzés RedHat-felhasználóknak

Alapértelmezés szerint a RedHat egy vim-minimal nevű csomagot telepít, mely a vim lecsupaszított változata. Ebből egy rakás hasznos szolgáltatást kihagytak, például az írásmód szerinti színkiemelést. A szerkesztő teljes változatának használatához telepítsük az alábbi csomagokat:

vim-X11-*változat*.i386.rpm
vim-common-*változat*.i386.rpm
vim-enhanced-*változat*.i386.rpm

A **változat** a rendszerhez kapott vim változatszáma.

Kezdjük a `qa` parancs beírásával, ennek hatására az ezt követő parancsok az `a` tárolóba kerülnek. A tárolókat az ábécé betűivel jelölhetjük. Most végezzük el a szerkesztést. Ugorjunk a sor elejére (`^`), illesszük be az `#include` fordítási elvet, majd tegyük `<` és `>` jelek közé a fájlnevet. A `q` paranccsal állítsuk le a makrórögzítést. A szöveg most így néz ki:

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

A helyőr most a második soron áll. A makró futtatásához a `@a` parancsot használjuk, ennek eredményeképpen a fájl az alábbiak szerint változik:

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

A `@a` parancs elé számot írva megadhatjuk, hogy a makró hányszor fusson le. A `2@a` kiadása után tehát ezt kapjuk:

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

A program tartalomjegyzéke

A vimhez kapott `ctags` parancs segítségével C fájlok tartalomjegyzékét készíthetjük el, ezt a vim „tags” fájlnak nevezi. Az alábbi parancs hatására létrejövő tags állomány a munkakönyvár C fájljaiban található összes függvény helyét tartalmazza:

```
ctags *.c
```

A létrehozott tags fájl rendkívül hasznos segítséget jelent a C programozók számára.
Tegyük fel, hogy egy fájlt szerkesztünk és az `olvas_bekezdes` függvényt keressük. A kódot böngészve rájövünk, hogy az `olvas_bekezdes` az `olvas_mondat` függvényt hívja meg. Szeretnénk megtudni, hogy ez a függvény mit csinál. Ha a függvény nevére állunk és a Ctrl+] billentyűket lenyomjuk, a vim ezen függvény meghatározásához ugrik. Itt pedig az derül ki, hogy az `olvas_mondat` az `olvas_szo` függvényt használja, tehát most erre állunk, és itt nyomjuk le a Ctrl+] billentyűket, hogy a függvényhez ugorjunk.

A vim a tagverem segítségével kíséri figyelemmel, hogy eddig mely helyeken jártunk. Ha azt szeretnénk megállapítani, hogy hol járunk, ebben a listában használjuk a `:tags` parancsot.

```
:tags
# TO tag               FROM line  in file/text
1  1 olvas_mondat      3  olvas_mondat();
2  1 olvas_szo         8  olvas_szo();
>
```

Ebben a példában az `olvas_szo` függvényről (ez nem szerepel a listában) először az `olvas_mondat`-ra, majd az `olvas_szo`-ra ugrottunk. Ha egy lépéssel szeretnénk visszaugrani, akkor a CTRL+T parancsot használjuk.

Kifinomult ugrási műveletek

Ha a CTRL+] paranccsal egy tagra lépünk, akkor az egész képernyő ugrik. Ha ehelyett a CTRL+W CTRL+] billentyűket nyomjuk le, akkor az éppen használt ablak kettéválik, és az ugrás az új ablakban történik.

Interaktív ugrás

Tegyük fel, hogy öt-hat kis program található a munkakönyvtárban, és az egyik `main` függvény helyét keressük. Ha a

```
:tag main
```

parancsot adjuk ki, a szerkesztő a `main` első előfordulására ugrik, és nem biztos, hogy ezt szeretnénk.
A `:tselect` parancs az adott névhez illeszkedő összes tagot megjeleníti, s ezek közül mi magunk választhatjuk ki a megfelelőt. Például:

```
:tselect main
# pri kind tag                  file
> 1 F C f    main                 a_test.cpp
       int main{ int argc, char* argv[] )
2 F    f       main                acp.cpp
       int main( int argc, char* argv[] )
3 F    f       main                add.cpp
       int main(int argc, char* argv[])
Enter nr of choice (<CR> to abort): 3
```

Mi történik, ha a névnek csak egy részét ismerjük, például a *valami*-process-*valami*-data nevű függvényre szeretnénk ugrani? A `:tags` parancs szabványos kifejezéseket is elfogad. Ebben az esetben a

```
:tag /process.*data
```

paranccsal az első ilyen sorra ugorhatunk. Ha a megadott kifejezésre több függvény is illeszkedik, akkor a `:tselect` segítségével választhatunk közülük.

Sortörés

A programok kódot és megjegyzéseket tartalmaznak. A kódnak pedig saját szerkezete van. Nincs szükségünk olyan szerkesztőre, mely a kód írásakor sortörést használ, viszont a megjegyzések hagyományos szövegek, ezeknél nyugodtan használhatjuk e lehetőséget. Sőt, a legtöbb esetben kifejezetten előnyös, ha a hosszú szöveges sorokat megtörjük.
A vim szerkesztő képes elkülöníteni a kódot a megjegyzésektől. Beállíthatjuk, hogy csak a megjegyzések esetében végezzen sortörést, a kódot pedig hagyja változatlanul. Ehhez az alábbi módosításokat kell elvégeznünk a .vimrc fájlban:

```
:autocmd FileType c,cpp
\       set formatoptions=croql
\       cindent
\       comments=sr:/*,mb:*,ex:*/,://
```

(Megjegyzés: a sorok folytonosságát a `\` jelzi)
Az `:autocmd` parancs arra utasítja a vimet, hogy az alatta elhelyezett parancsokat akkor hajtsa végre, ha C vagy C++ fájllal dolgozunk (ezt a .c és a .cpp kiterjesztésből állapítja meg).
A `formatoptions` hatására a megjegyzéseknél sortörést hajt végre, a kód viszont változatlan marad. A

```
set comments= sr:/*,mb:*,ex:*/,://
```

sor tudatja a vimmel, hogy a megjegyzések hogyan néznek ki. Itt a C nyelvben (`/*` és `*/`) és a C++-ban (`//`) megszokott megjegyzésjeleket állítottuk be. Azt is elmagyaráztuk a vimnek, ha egy C-stílusú megjegyzés közepén járunk, akkor minden sort `*` karakterrel kezdjen. Ezen értékek beállítása után, ha `/*`-t írunk és ENTER-t nyomunk, a vim a következő sor elejére magától kiteszi a `*` jelet.

Összegzés

Jelen esetben nem áll módomban hosszasan elemezni e lehetőséget, de remélem annyi kiderült, hogy egy igen hasznos szolgáltatásról van szó, mellyel érdemes eljátszadoznunk egy kicsit. Aki részletesebb tájékoztatásra vágyik, az olvassa el a beépített súgót. A vim egy rendkívül rugalmas szerkesztő, igyekeztem bemutatni néhány érdekesebb lehetőségét, de ezzel is csupán a felszínét érintettem. Több száz vagy akár ezer olyan parancs létezik a vimben, amit itt meg sem említettem. Ezekről a szerkesztővel kapott leírásban vagy a beépített súgóban olvashatunk.
Bízom benne, hogy a vim néhány új lehetőségének ismertetésével sokakat sikerült bevezetnem a hatékonyabb szerkesztés világába. Hogy innen merre haladunk, az csupán rajtunk múlik.

Steve Qualline
jelenleg San Diegóban él és a Nokia mobiltelefonokkal foglalkozó részlegénél dolgozik, hétvégeken pedig igazi mozdonyvezető a californiai Polwayben található turistavasútnál.

Ötlet

Mozgás a vi-ban

A vi-ben a (,), [,], { és } jelek között is mozoghatunk: a % parancs segítségével a jelpár záró tagjára ugrunk. Nézzünk egy példát:

```
( ezmegaz [ amaz
itt egy másik sor
{  ]
} )
```

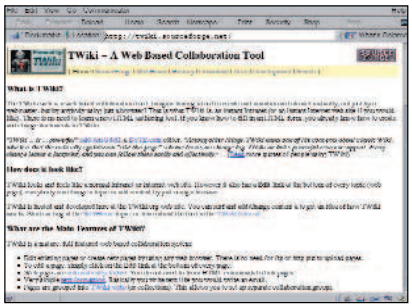
Ha itt az első kapcsos zárójelre állunk, a % paranccsal a jelpár záró tagjára, azaz egy sorral lejjebb ugorhatunk. Ezt a módszert használhatjuk megjegyzésekben is, és így gyorsan mozoghatunk előre-hátra terjedelmes programjainkban. A C-hez hasonló nyelvekben pedig kitűnő eszköz a függvények kezdetének és végének betájolására.

Andamooka

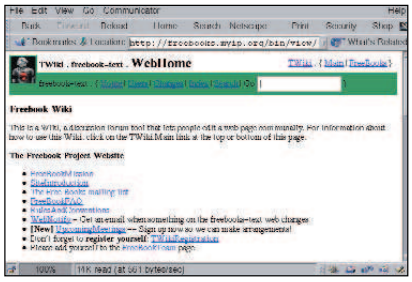
A nyílt forrású programfejlesztés hatására nyílt könyvek jelennek meg.

Alig több mint egy évvel ezelőtt könyvet kezdtem írni a KDE-alkalmazások fejlesztéséről. A KDE kódjába rengeteg új alkalmazásfelület és alrendszer épült be – ebből lett később a KDE 2.0 –, és a fejlesztőknek új adatokra volt szükségük, ha lépést akartak tartani. Vajon hogyan lehetne egy könyvet – amit megírnak, szerkesztenek, átnéznek, kinyomtatnak, terjesztenek – még azelőtt megjelentetni, mielőtt a témájául szolgáló programcsomag újabb változata megjelenne? A megoldás egyik része, hogy a programfejlesztő csapat néhány tagját szerzőtársként kell bevonnia, így megteremthető annak feltétele, hogy a leírtak a kiadáskor valóban naprakészek legyenek. A másik része, a nyílt kiadási szerződés (Open Publication License) vezetett el az Andamooka és a nyílt támogatás megszületéséhez.

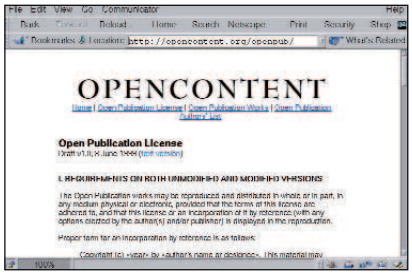
Az Andamooka webalapú olvasórendszer célkitűzése az, hogy a bárki által elérhető szöveg mindig a legfrissebb adatokat tartalmazza, ezért megteremti annak feltételeit, hogy az olvasók a szöveg mellé megjegyzéseket helyezhessenek el.



➲ www.twiki.sourceforge.net



➲ www.freebooks.myip.org/bin/view/



➲ www.opencontent.org/openpub

Nyílt tartalom

A nyílt tartalom olyan adat – legtöbbször szöveges, de lehetnek mellette képek vagy bármi más –, amit bárki szabadon módosíthat és terjeszthet. Ha ez valakit a nyílt forrású programokra emlékeztet, nem csodálkozzom: az alapötlet ugyanis onnan származik.

Valyon miért van szükség a programok esetében bevált nyílt forrású fejlesztés bevezetésére a könyvek világában? Olvassuk csak el *Davis Wiley* írását a *Nyílt forrású tartalomfejlesztés*ről (➲ http://opencontent.org/bazaar.shtml), és az *Eric Raymond* tollából származó *Miért jó nekünk a GNU FDL?* című művet (➲ http://www.gnu.org/philosophy/why-gfdl.html), vagy *Benjamin Crowell* írását *Életképes-e a nyílt forrású könyv?* (➲ http://www.lightandmatter.com/article/article.html) című esszét. A lényeget viszont én is összefoglalhatom. Tehát a nyílt forrású modell:

- többé-kevésbé annak a kezébe adja a programot, akinek a legnagyobb szüksége van rá;
- többféle szolgáltatást nyújt, hiszen a nyílt forrású adat szabadon módosítható;
- jóval megbízhatóbb, mert sok felhasználó próbálja ki, sokféle környezetben;
- azokat a részelemeket tartalmazza, amelyekre a legnagyobb kereslet mutatkozik, köszönhetően annak, hogy a felhasználóktól rengeteg visszajelzés érkezik a készítőkhöz. Sőt, gyakran a felhasználókból kerülnek ki a későbbi tehetséges fejlesztők.

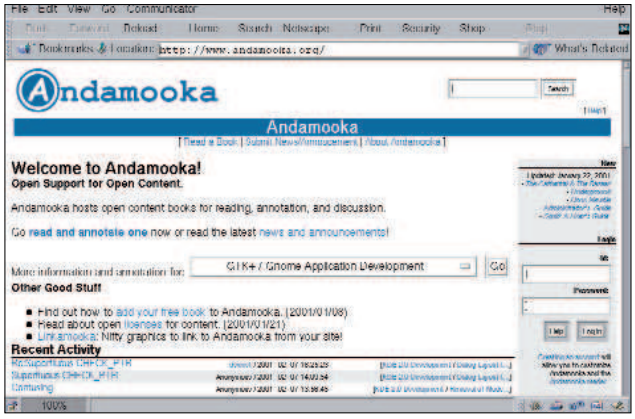
A nyílt tartalomfejlesztést tulajdonképpen ugyanez a négy elv határozza meg. A szerzők sokkal több olvasót elérhetnek, ha ingyen hozzáférhetővé teszik műveiket. A nyílt forrású programok leírása esetében is jogos a követelés, hogy azt bárki módosíthassa, és így a programmal együtt a felhasználók kapcsolódó tudása is bővülhessen.

A tartalom akkor megbízható, ha pontos és könnyen befogadható. Az anyag, pontosság és a közérthetőség tekintetében pedig csak tovább erősödik, ha a visszajelzéseket beépítjük az új változatokba. Más szóval, a nyílt tartalom elve alapján minden olvasóból szerző válhat, és mindegyikük a munka azon részével foglalkozik, amelyhez képessége, kedve van. Végül, ha a művet sokan olvassák, akkor rengeteg visszajelzés fog érkezni a hiányosságokkal kapcsolatban. Így a szerzők sokkal könnyebben megtervezhetik a következő változatot. Ha a szerző egyetért, a módosító javaslatokkal akár be is épülhetnek a műbe, ezáltal egy emberre kevesebb feladat jut, a végeredmény minősége pedig remélhetőleg sokkal jobb, mint ha az egészet egy ember, vagy egy pár fős csapat vállalná magára.

A nyílt tartalom elvét az Open Publication License (OPL), a GNU Free Documentation License és a Linux Documentation Project is magában foglalja. Mindhárom lehetővé teszi, hogy a hozzá kapcsolódó művet az olvasó vagy a felhasználó módosíthassa és terjeszthesse. Az OPL azonban a nyomtatott formában való terjesztés elé gátat szab. Egy ilyen tiltásnak köszönhetően egy könyvkiadó vállalat visszanyerheti a megjelentetés – ez rengeteg munkaórát és pénzt felemésztő folyamat – költségeit. Azon kiadók számára, melyek úgy döntenek, hogy a könyv első megjelenésekor kihasználják e tiltást, azt tanácsolom, hogy később váljanak meg tőle, például a költségek visszanyerése után vagy egy meghatározott idő elteltével. A papíron megjelenő változat lehetővé teszi, hogy a mű a „hagyományos” piacokra is eljuthasson.

Nyílt támogatás

Mára már világossá vált, hogy a nyílt tartalom elvének alkalmazása kiemelkedő minőségű művek megszületését teszi lehetővé. A Weben akár most is olvashatunk nyílt tartalmú könyveket. Néhány cím kedvcsinálóként: *Carey Bunks* Grokking the GIMP, *Havoc Pennington* GTK+/GNOME Application Development, *Walsh és Leonard Müllner* DocBook: The Definitive Guide és természetesen a *David Sweet* és mások által írt KDE 2.0 Development. A megfelelő eszközök birtokában a szerzők a szöveg mellett mást is elhelyezhetnek a honlapon. Az Andamooka honlapja ezen eszközöket ingyen bocsátja a leendő szerzők rendelkezésére. A műveket feltölthetik böngészőn keresztüli olvasás vagy letöltés céljára. Módjukban áll bejelentéseket, híreket



➲ www.andamooka.org

közzétenni, fórumokat működtetni, ahol mindenki elmondhatja a véleményét. A szerzők a kapcsolódó forráskódokat, példaprogramokat is feltölthetik. A fórum látogatói közös nyelven beszélnek – hiszen végül is ugyanazt a könyvet olvassák!

Az Andamooka szíve-lelke a megjegyzések beszúrásának lehetősége. A fentebb vázolt nyílt tartalom legnagyobb részét a könyvek fejezetei végén található megjegyzéshalmaz jelenti, ezt bárki bővítheti. A megjegyzéseket az OPL elvei alapján küldhetjük el, így azok a könyvvel együtt szabadon terjeszthetők. A megjegyzésekkel teletűzdelt könyveket a Weben is olvashatjuk, de le is tölthetjük, nyomtatás, CD-re írás stb. céljából.

A megjegyzések természetesen rendkívül sokfélék lehetnek, és ez a minőségre is vonatkozik, így valamilyen értékelőrendszert kellett felállítani. Az Andamooka a SlashDot kísérleti jellegű pontozási rendszerét használja. Az egyes megjegyzések pontozása az olvasók körében végzett alkalomszerű felmérések alapján történik. Az olvasó három osztályba (r) sorolhatja a megjegyzéseket: jó (r=3), semleges (r=2), rossz (r=1). Mivel az általában magas pontszámot kapott megjegyzéseket szeretnénk a lap tetejére helyezni (hogy a látogató először ezekkel szembesüljön), ezért minden megjegyzés pontszámot (S) kap. Ha egy megjegyzésre N alkalommal adtak le szavazatot, a pontszáma: S=SQRT(N)*Átlag(r)/ Általános_Deviancia(r). Emberi nyelven ez azt jelenti, hogy egy megjegyzés pontszáma magasabb, ha:

- több ember olvasta el és szavazott rá;
- a szavazók általában magasabb osztályba sorolták;
- ha többen sorolták ugyanolyan magas osztályba.

A minőségi megjegyzések arányának növelése céljából a beküldők úgynevezett „Karma” pontokat kapnak. E személyes pontszám attól függ, hogy az olvasóközönség hogyan értékeli a beküldött megjegyzéseket, valamint attól, hogy az illető mennyi adatot töltött fel és milyen gyakorisággal. A rendszer e két utóbbi eleme még fejlesztés alatt áll. Reményeink szerint a rendszer bebizonyítja működőképességét, és az olvasók saját maguk osztályozzák majd a legjobb megjegyzéseket. Hogy sikeres lesz-e az Andamooka? Kiderül.

Nyílt fejlesztés

A következő lépés a nyílt fejlesztéseket lehetővé tevő rendszer kialakítása lesz. A modell szintén a nyílt forrású fejlesztésekből származik, azonban van néhány figyelemre méltó különbség. A nyílt forrású fejlesztésben érdekeltek másokkal közösen szeretnék könyvüket vagy más szellemi terméküket elkészíteni, s így a rengeteg feladatot egyenlő mértékben lehet szétosztani a résztvevők között, a végeredmény minősége pedig általában messze felülmúlja az egy ember által készített művekét. Ugyanez már régóta zavartalanul működik a programok fejlesztésében, szóval miért ne lehetne ezt az elvet bármilyen digitális adatra kiterjeszteni?

A szerzők (és itt nem feltétlenül számítástechnikával foglalkozó emberekre gondolok) általában kevésbé értenek a dolgok programozási részéhez, mint a programozók. Ők a CVS-hez hasonló rendszereket sem képesek használni, hiszen ezek telepítése, használata egy képzett rendszergazdának is komoly feladatot jelent. Értelemszerű tehát, hogy szükség van egy könnyen használható projektkezelő és csapatmunkatámogató rendszerre. A TWiki nevű webalapú alkalmazás és Wiki Wiki nevű rokona jó példa erre, de esetükben átalakításokra van szükség ahhoz, hogy ezen rendszereket a nyílt forrású fejlesztésben felhasználhassuk. A Benjamin Crowell által alapított FreeBooks társaság tagjai (magam is közéjük tartozom) hasonló rendszerekkel kísérleteznek, első lépésben szabad könyvet írunk a szabad könyvekről. Csatlakozz hozzánk, a részleteket a honlapon megtalálod!

Az is érdekes kérdés, hogy a nyílt tartalomfejlesztés vajon ugyanakkora sikert ér-e el, mint a nyílt programfejlesztés. Eddig már jó néhányan csatlakoztak a Freebooks társasághoz, és a KDE 2.0 Development öt nyelvre fordítása is szépen halad. Ezek talán kevésnek tűnnek a nyílt programfejlesztésben részt vállaló tízezres tömeghez képest, azonban nem szabad elfelejtenünk, hogy a dolog csupán nemrég indult el, és az eddigi érdeklődés derűlátásra ad okot.

Összegzés

Az Andamooka az első lépés a teljesen nyílt tartalomfejlesztés megszületése felé. A honlapon most és a közeljövőben megtalálható könyvek „zárt” fejlesztés keretein belül készültek, de a nyílt szerződés lehetővé teszi a megjegyzések beküldését és a módosított anyag szabad terjesztését. Az Andamooka lehetőségeit kihasználva a szerzők közvetlenül kapcsolatba léphetnek olvasóikkal. A szerzők és az olvasók párbeszéde, együttműködése olyan tudásalapot teremthet, melyre alapozva a könyv egyre tökéletesebb változatai jelenhetnek meg. Jómagam a KDE 2.0 Development című művemen fogok tovább dolgozni, az Andamooka segítségével. Kérek mindenkit, írjon, szerkesszen, fordítson, jelezze a hibákat, javasoljon új témákat. A folyamatosan fejlődő művet később ingyenes, szabad, elektronikus változatban jelentetem meg. A KDE 2.0 Development a nyílt forrású tartalomfejlesztés kísérleti terepévé válhat.

David Sweet
(http://www.andamooka.org/~dsweet)
a marylandi egyetemen fizika- és káoszelméletből szerzett PhD fokozatot. A későbbiekben figyelme középpontjába a biztonsági rendszerek kerültek. Írásait a Nature, a Linux Journal és a Physical Review Letters magazinokban olvashatjuk.

Kapcsolódó címek

➲ http://www.andamooka.org/

Open Publication License
➲ http://www.opencontent.org/openpub

GNU Free Documentation License
➲ http://www.gnu.org/copyleft/fdl.html

Linux Documentation Project
➲ http://www.linuxdoc.org/manifesto.html

Twiki
➲ http://twiki.sourceforge.net/

FreeBooks
➲ http://freebooks.myip.org/

A számítógépes grafika és a GIMP

Bevezetés a legkedveltebb ingyenes grafikai program, rejtelmeibe.

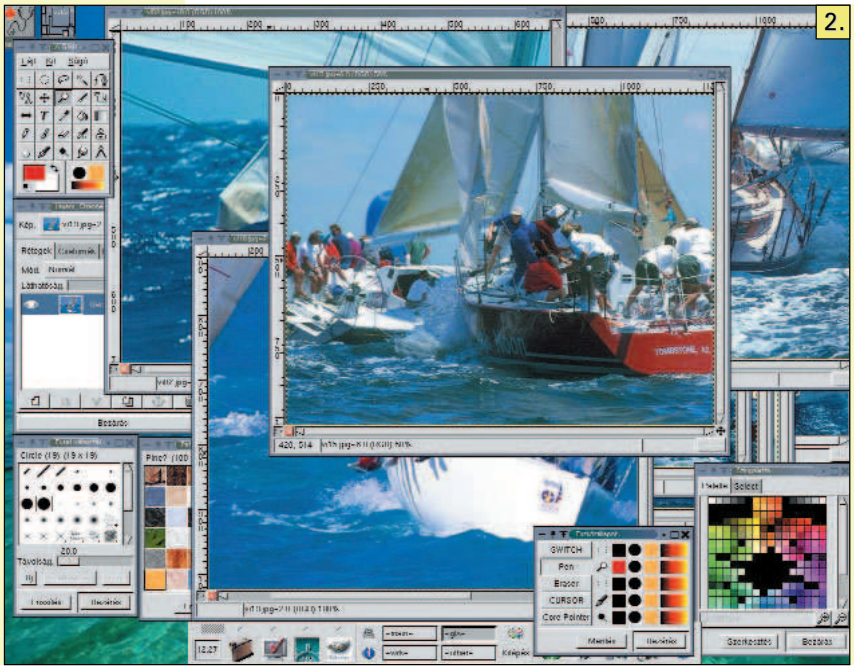
A GIMP megjelenése óta szinte minden összehasonlításban az első helyet foglalja el a linuxos képszerkesztő programok között. Tudása, használhatósága eléri, néha meg is előzi a digitális képfeldolgozásban és grafikában eddig ismert, drága programokat.
Vajon miért ilyen sikeres a GIMP? Hogyan lehet vele dolgozni? Hogyan lehet a GIMP rejtett tartalékait kihasználni? Ezekre és hasonló kérdésekre próbál választ adni a cikksorozat.

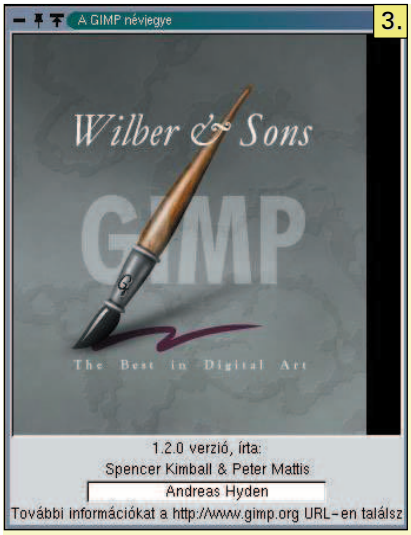
Rövid bemutató

A GIMP a GNU Image Manipulation Program angol szavakból alkotott mozaikszó (GNU képszerkesztő program).
A fejlesztést *Spencer Kimball* és *Peter Mattis* kezdte 1995 nyarán a Berkeley Egyetemen. Fél évvel később a korai próbaváltozat megjelent az Interneten és ezzel elkezdődött a nagy kaland.

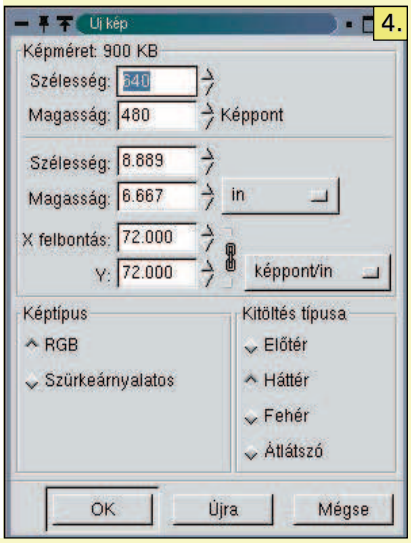
Ez a GIMP-változat még a Motif könyvtárat használta, mely abban az időben még kereskedelmi termék volt. Ez a tény azonban megakadályozta széles körű elterjedését a linuxos felhasználók között. Ezért 1996. júliusában megszületett a GIMP Toolkit (GTK), a GIMP eszközkészlet. Ekkor csatlakozott sok fejlesztő a munkához, segítve a tervezést, a fejlesztést és az ellenőrzést. Hosszú munka után 1997. februárjában megjelent a 0.99 változat. Ez már támogatta a rétegeket (layers) és számos bővítmény készült hozzá. Végül a hosszadalmas munka gyümölcseként 1998. május 19-én megjelent az első megbízható változat: a GIMP 1.0.
A GIMP célja elsősorban a számítógépen felhasznált képek szerkesztése, rajzolása, módosítása volt. Mára elsősorban a webes felületre szánt grafikák elkészítésének kiváló eszközévé vált, de használják programfejlesztés során a program grafikai megjelenésének tervezésére is (lásd *5. kép*).
Ma már minden szabadon terjeszthető Linux-rendszer része, de elérhető Win32 és OS/2 alatt is. A program magyarítása jelenleg folyamatban van, még nem teljes ugyan, de már most is használható.

1.

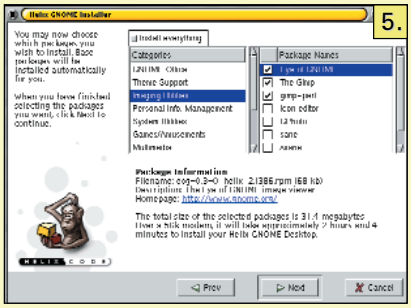

2.

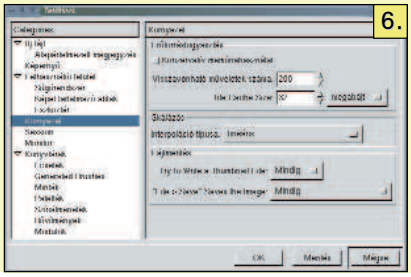


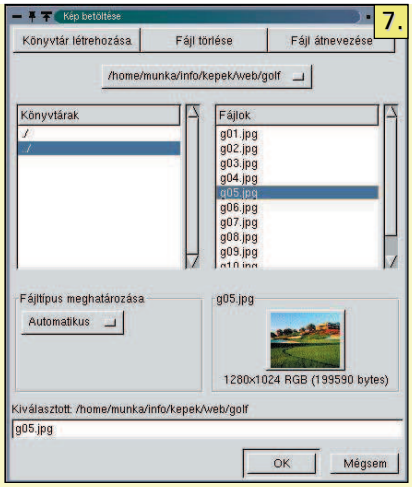
3.



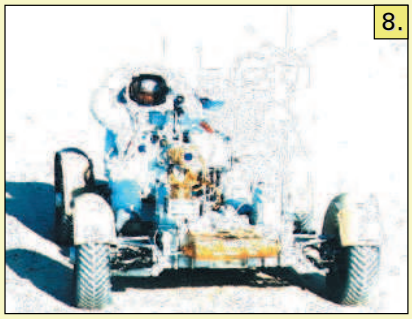
4.


5.


6.



7.

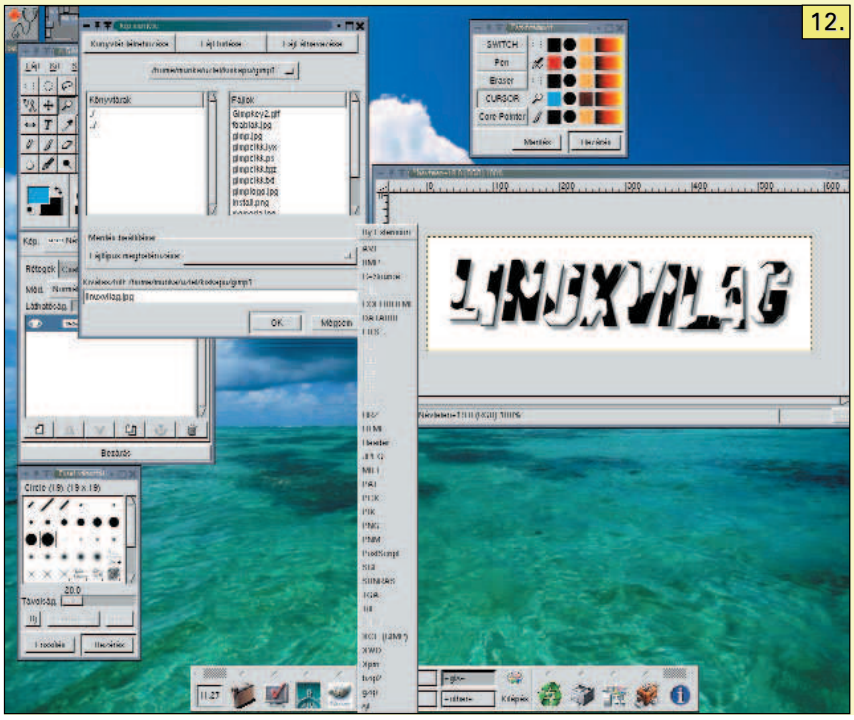

8.


9.

1. *kép* A GIMP irányítópultja
2. *kép* A GIMP használat közben, jól látszanak a párbeszédablakok, és a megnyitott képek
3. *kép* A GIMP indulása
4. *kép* Új kép létrehozása menüpont
5. *kép* GIMP-pel készített grafikák, gombok a programokban
6. *kép* A gyorstár használatának beálltása
7. *kép* A képek megnyitása esetén jól látható a kis előkép
8. *kép* Befejezetlen grafika
9. *kép* Olajfestés
10. *kép* A GIMP-felhasználók kedvenc billentyűzete
11. *kép* Montázs: a rétegek, maszkok kiegészítő hatékony használata
12. *kép* Mentés
13. *kép* A menü használatának lehetőségei

10.


11.



12.

A GIMP tulajdonságai

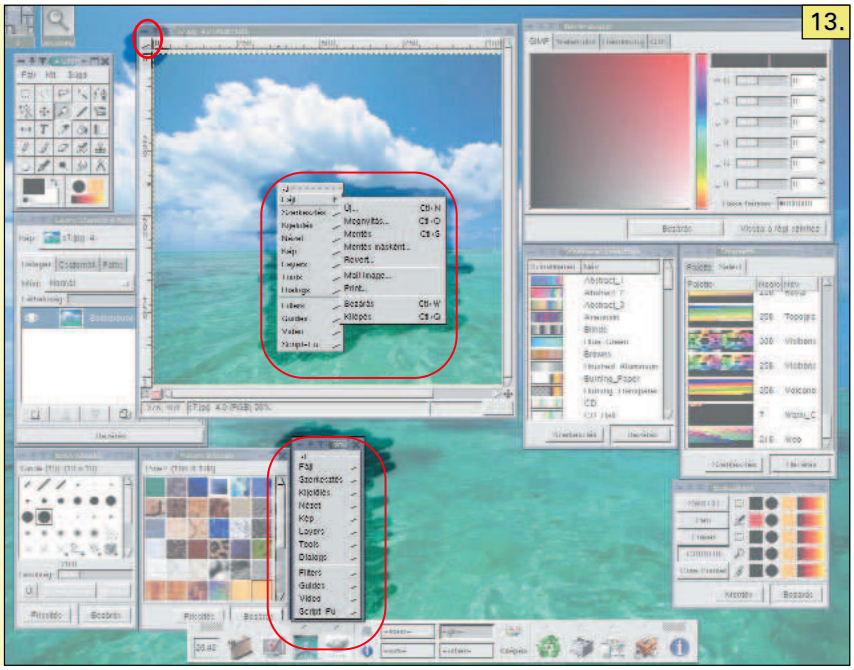
- Teljes rajzeszközkészlet: ecset, festékszóró, ceruza, toll, másolás, elmosás.
- A hatékony memóriagazdálkodás eredményeként a szerkeszthető kép méretét csupán a szabad háttértár korlátozza!
- Kezeli a rétegeket (layer) és csatornákat (channels).
- Rendkívül hatékony színátmenet és színkeverés.
- A GIMP belső szolgáltatásainak elérése a Script-Fu nyelven keresztül.
- Többlépcsős visszavonás/visszaállítás (ennek is csupán a szabad háttértár szab korlátot).
- Korlátlan számú, egyszerre megnyitható kép.
- Animációk készítése.
- Többek között az alábbi képformátumok támogatása: gif, jpg, png, xpm, tiff, tga, mpeg, ps, pdf, pcx, bmp, psd. Ezek a képek betölthetők, menthetők, átalakíthatók a GIMP-pel.
- kijelölési eszközök közül négyzet, ellipszis, szabadkézi, Bezier-görbe, egybefüggő területek kijelölés használata.
- Több mint száz beépített bővítmény, valamint a beépített Script-Fu segítségével korlátlanul fejleszthető.
- Egyéni ecsetek és kitöltőminták készítése.
- Nyomásérzékeny tábla támogatása.
- Hatékony a betűkezelés.
- Átalakítási eszközök: elforgatás, kicsinyítés, nagyítás, torzítás.
- Lapolvasó támogatása, képernyőkép mentése.

Telepítés

A legfrissebb változat a cikk írásának időpontjában az 1.2.1. Az utolsó változat letölt-

hető az ➲ ftp.gimp.org címről (és a tükörkiszolgálókról), de minden Linux-változatban megtalálható valamelyik korábbi változat. Érdemes letölteni az új változatokat, mert a program nagyon lendületesen fejlődik, szinte minden frissítésnél bővülnek a program szolgáltatásai, és az újabb változatok egyre megbízhatóbbak. A GIMP használatához GTK 1.2.8 szükséges.

A program futtatásához legalább 12-20 MB szabad memória kell, természetesen minden megnyitott kép kezeléséhez nélkülözhetetlen további memória. Egy 640x480 pontból álló kép mérete a használt rétegektől függően megközelítőleg 3 MB. Így a legkisebb javasolt memória 32 MB, az ajánlott pedig a munka jellegétől függően 64–256 MB. Ha gépünk viszonylag kevés memóriával rendelkezik, állítsuk nagyobbra a gyorstár méretét (Tile Cache Size, *6. kép*).


13.

Használat

A GIMP indulása után rögtön megtaláljuk a főablakot és a párbeszédablakokat. (Lásd *2. kép*). A program használatának a legfontosabb irányítópultja a GIMP eszközkészleteinek ablaka (*1. kép*). Itt találhatók a Fájlműveletek: az új kép létrehozása, a megnyitás, a bezárás, a külső eszközről beolvasás (lapolvasó és képernyő!), a beállítás és a párbeszédablakok megnyitása. A Kiterjesztések tartalmazza a kiterjesztés kezelőit és a beépített képelőállító eszközöket (feliratkészítés, háttérkészítők, gombok, betűkészlet elkészítését, kitöltőmintákat, weboldalra feliratokat, gömböt stb.), valamint a böngészőkezelést. Ezenkívül a Súgó is itt található. Megnyitás esetén (*7. kép*) kiválasztjuk a könyvtárat és a fájl nevét, valamint a formátumát, a GIMP pedig megpróbálja automatikusan felismerni, de mi is kiválaszthatjuk. A kiválasztott fájl esetén megjelenik a kis nézőkép. Új kép létrehozása esetén (*4. kép*) a kép méretét megadhatjuk képpontban, hüvelykben, milliméterben, centiméterben, nyomdai pontban és akár általunk létrehozott mértékegységben is. Kiválaszthatjuk a kép típusát (színes vagy szürkeárnyalatos) és a háttér színét. Miután létrehoztunk egy képet, nézzük meg, milyen eszközeink vannak.

A főablakon található gombok balról jobbra:

- négyzet kijelölése
- ellipszis kijelölése
- szabadkézi kijelölés
- egybefüggő területek kijelölése
- kijelölés Bezier-görbével
- intelligens kijelölés a terület körülhatárolásával
- mozgatás
- nagyítás
- kivágás
- átalakítás (forgatás, méretezés, torzítás, perspektivikus torzítás)
- tükrözés
- szöveg
- szín beállítása képről (pipetta)
- kitöltés (színnel, mintával)
- színátmenet
- ceruza, toll
- ecset
- radír
- festékszóró
- képrészlettel, mintával kitöltés
- elmosás, élesítés
- világosítás, sötétítés
- elkenés, maszatolás
- mérőeszköz.

Alatta megtaláljuk:

- az előtér és háttér színét
- az ecset méretét, illetve típusát
- a kitöltőmintát és
- a színátmenet beállítását.

Első látásra szokatlan lehet a menük elérése és használata, ugyanis az eddig megszokott gyakorlattól kicsit eltér. Az egyes menüket háromféle módon érhetjük el: első lehetőségünk, hogy a kép bal felső sarkában található kis háromszögre kattintunk, majd kiválasztjuk a kívánt menüt, a második, hogy a képen az egér jobb gombjával kattintunk, a harmadik lehetőség pedig, hogy a sokat használni kívánt menüt a felső sorára kattintva „kihelyezzük” az asztalra külön ablakként (*13. kép*). Így mindenki megtalálhatja a számára legkényelmesebb megoldást.

Ezenkívül lehetőségünk nyílik a navigátorablak használatára is. Ezt a *Nézet* menüben találjuk. A képek használata során szükség lehet arra, hogy egyszerre lássuk a teljes képet és a nagyítást. Erre találták ki a *Nézet/Új nézet* menüpontját. Így egyszerre több ablakban láthatjuk a képünket. A Nézet menüben található az *Info* ablak, melyben nemcsak a képre vonatkozó adatokat találhatjuk meg, hanem az adott területre vonatkozó színeket.

Összegzés

A GIMP egy rendkívül okos és összetett program, nem szabad félnünk az első lépésektől! Aki egy másik rajzolóprogramhoz szokott hozzá, az ismerkedés időszakában természetesen idegennek érezheti a program környezetét, de ígérem, hogy rövid gyakorlás után mindenki örömmel használja majd ezt a csodálatos szerkesztőt.

Sorozatunk következő részében a GIMP lehetőségeit, rejtett titkait fogjuk bemutatni egy-két kézzelfogható példán keresztül. Bízom benne, hogy sikerül azokkal is megismertetni (és megszerettetni) a programot, akik már régebben telepítették a rendszerükre, de még soha nem merték használni. Mind a cikkel, mind a GIMP-pel kapcsolatban szívesen válaszolok Olvasóink kérdéseire!

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

Kapcsolódó címek

➲ www.gimp.hu
➲ www.gimp.org

A SoundTracker ismertetése (3. rész)

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

Előző számunkban már szót ejtettünk a SoundTrackerről, mely az egyik legnagyobb teljesítményű modulszerkesztő Linux alatt. Eddig a beszerzéséről és a telepítéséről számoltunk be.

Egy kis ismerkedés

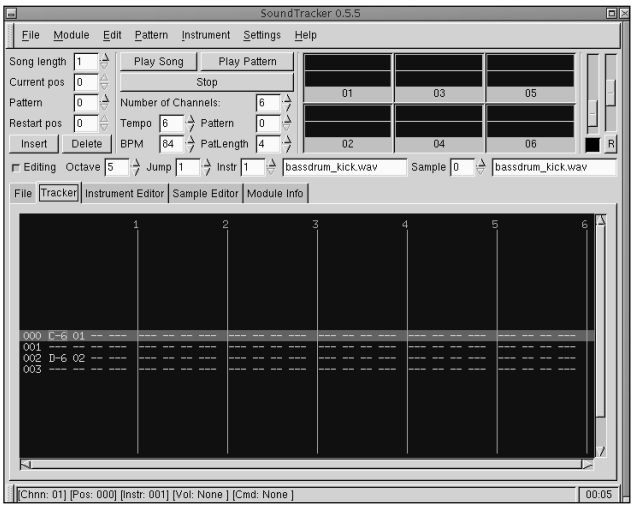
A jelenlegi 0.5.5-ös változathoz leírásként csupán egy README fájl jár, és ennél azért valamivel többre lenne szükség. Minden modulszerkesztő hasonló felépítésű, és a tájékozódásban segítséget nyújtó leírások a United Trackers és a MODPlug Central honlapján érhetők el. Ha ezelőtt még soha nem használtunk modulszerkesztőt, bizonyára szeretnénk megtudni, hogy mire is képes. A fent említett két honlapon hatalmas .MOD és .XM gyűjteményeket találunk. Érdemes legalább néhányat letölteni, hallgatni, tanulmányozni, hiszen a profi zenészek moduljaiból sok trükköt elleshetünk.

A modulokban mintavételezett hangokat használhatunk, tehát mielőtt belevágnánk a zenélésbe, gyűjtsünk össze legalább néhány tucat hangmintát. Ezeket más modulokból is kiszedhetjük, de a United Trackers és a MODPlug Central honlapokon is rengeteget fellelhetünk. A SoundTracker a libaudiofile által támogatott formátumokkal képes dolgozni, tehát moduljainkban szabadon keverhetjük a WAV, AIFF és AU fájlokat.

A SoundTracker monó hangszerekkel dolgozik. Sztereó hangminta betöltésekor erre egy kedves üzenettel figyelmeztet bennünket, majd három lehetőség közül választhatunk: a hangminta bal vagy jobb sávját használhatjuk, vagy a fájlt monoformátumra alakíthatjuk a két sáv összeolvasztásával. Ez utóbbi tűnik a legjobb választásnak. Fontos, hogy előzőleg csiszoljuk tökéletesre a szerkesztéshez használandó hangmintákat. A finomhangolás és a hurkok, azaz a hangmintán belüli ismétlődő részek beállítása mindenképpen szükséges, és a SoundTracker segítségével pontosan meghatározhatjuk a minta hangerejét és a sztereotérben elfoglalt helyét (panning) is. Bár a program tartalmaz egy hangmintaszerkesztőt is, javaslom, hogy inkább egy komolyabb programot (MiXViews, Snd, DAP stb.) használjunk e célra. Ezek a programok kifejezetten a hangminták szerkesztésére készültek, segítségükkel a munkát nagyobb felbontásban, több hatás felhasználásával végezhetjük, mint a modulszerkesztők beépített hangmintaszerkesztőivel.

A szerkesztés megkezdése előtt be kell töltenünk azokat a hangmintákat, melyeket használni szeretnénk. A SoundTrackerben a modulok szerkesztéséhez 128 hangszert alkalmazhatunk, tehát egész hangmintakönyvtárakat tölthetünk be, s a hangszerek között a *Module Info* fül fölött található *Instr* mezővel válthatunk. A kiválasztott hangot kipróbálhatjuk valamelyik billentyű lenyomásával (lásd az *1. táblázatot*).

Név: SoundTracker♦**A program alkotója:** Michael Krause♦**Változat:** 0.5.5
Honlap: ➲ http://www.soundtracker.org/
FTP: ➲ ftp://ftp.soundtracker.org/pub/soundtracker/
Támogatott meghajtók: OSS/Free (rendszermag), ALSA, OSS/Linux, ESD (EsounD)
Támogatott operációs rendszerek: Linux 2.2.x, SunOS 5.7, FreeBSD
Fejlesztik? Igen.
Terjesztés: Ingyenesen letölthető (GPL), a forráskóddal együtt.



1. kép A SoundTracker szerkesztőképernyője

A SoundTracker használata

Az *1. képen* a SoundTracker főképernyőjét láthatjuk. Nemsokára elmagyarázom, hogyan vihetünk be ide adatokat, egyelőre azonban elég annyit tudnunk, hogy négy oszlop jelöl egy-egy sávot, illetve csatornát, és minden sor egy lépést jelent. Nézzük meg például, mit jelentenek az alábbiak:

Lépés	Magasság	Hangminta	Hangerő	Hatás (parancs és értékek)
000	C-6	01	—	—
001	—	—	—	—
002	D-6	02	—	—
003	—	—	—	—

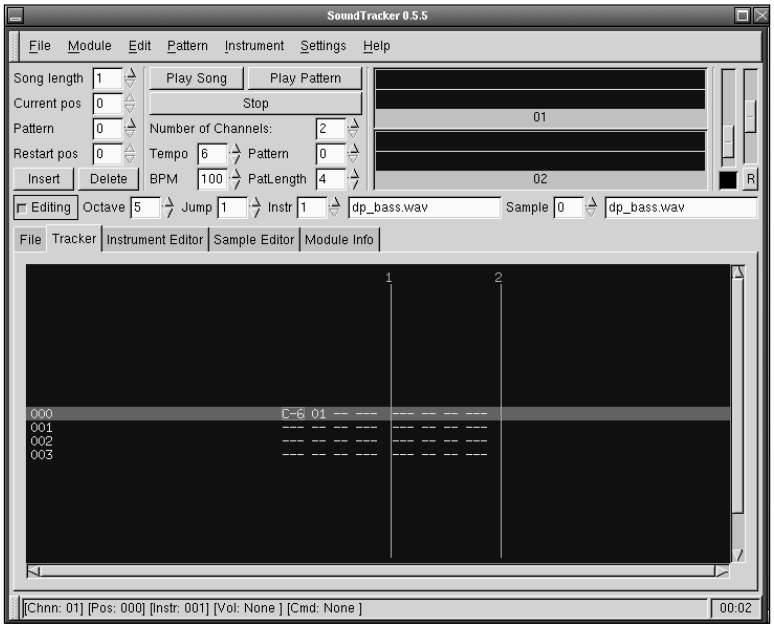
Itt azt láthatjuk, hogy a 01. számú hangszer – történetesen egy basszusdob – a 0. lépésnél a 6. oktáv C hangján, alapértelmezett hangerővel és hatások nélkül szólal meg, a 2. lépésnél pedig a 02. számú hangszer (egy pergődob) teszi ugyanezt a D6 hangmagasságon. A Play Pattern gombra kattintva láthatjuk, hogy a kijelző folyamatosan görgeti a panelt az elsőtől a negyedik lépésig, majd zökkenőmentesen újrakezdi a lejátszást.

A hangszerek a megadott hangmagasságokon szólalnak meg, és egy csatornán belül természetesen többféle hangszert is felhasználhatunk, tetszőleges összeállításban. Az értékek lépésről lépésre változhatnak, de ugyanazok is maradhatnak.

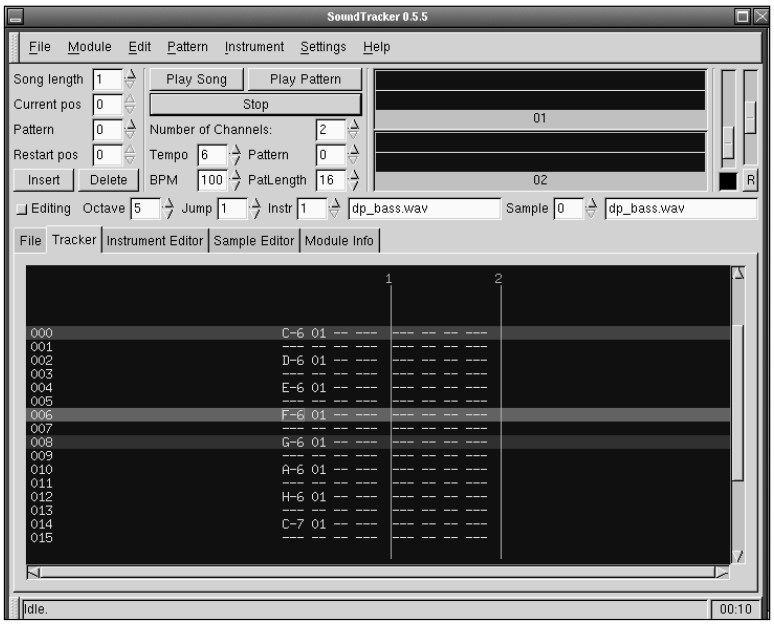
A hatásparancs és -értékei határozzák meg a hatást és annak erősségét. Hatásokat olyan lépéseknél is meghatározhatunk, ahol az adott sávon nem szólaltatunk meg semmilyen hangszert. Például a 000. lépésnél megszólaltatunk egy hegedűt, majd a 001–010. lépések

1. táblázat **A billentyűzeten található hangjegyek**

Hangjegy:	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	
Billentyű:	Q	2	W	3	E	R	5	T	6	Z	7	U	Felső oktáv
Billentyű:	Y	S	X	D	C	V	G	B	H	N	J	M	Alsó oktáv



2. kép Négy lépést bemutató panel



3. kép A C dúr hangsor, basszushanggal

alatt fokozatosan elhajlítjuk a hangot stb. Érdemes megjegyezni, hogy dobjaink életszerűségét nagymértékben növelhetjük azzal, ha legalább két-három hangmagasságon használjuk azokat.

Az 1. képen bemutatott események teszik ki egy panel egy csatornáját. Egy panel legfeljebb 64 lépésből és 32 sávból állhat. A sávokat és paneleket másolhatjuk, kivághatjuk és beilleszthetjük. Alapértelmezés szerint a program minden sávot lejátszik, de valamelyik oszcilloszkópra kattintva ki- és bekapcsolhatjuk az egyes csatornákat; a jobb gombbal kattintva pedig az adott sávot önmagában hallgathatjuk.

Ahogy az *1. képen* is látszik, a program vezérlői az ablak bal felső részében találhatók, tőlük jobbra az oszcilloszkópok, alattuk pedig a fájlkezelő, a modulszerkesztő, a hangszerszerkesztő, a mintaszerkesztő és a modul tájékoztató képernyője helyezkedik el (az alsó rész elemei között fülekkel válthatunk). Most, hogy már tudjuk, miből épül fel egy sáv, elkezdhetünk dalt írni.

A panelek szerkesztése

A panelekbe a számítógép billentyűzetéről, vagy egy külső MIDI-billentyűzetről vihetünk be adatokat. Mindkét módszert használhatjuk valós időben, ekkor a panel lejátszása közben folyamatosan szerkeszthetjük azt, illetve szerkesztő üzemmódban is; a lépéseket ilyenkor egyesével illeszthetjük be, és a helyőr helyzetét a nyíl billentyűkkel állíthatjuk. Nézzük meg először a számítógép billentyűzetének használatát.

Szerkesztés a számítógép billentyűzetéről

Ha az *Editing* (Szerkesztés) üzemmódot nem kapcsoltuk be az ablak bal szélén látható gombbal, akkor szabadon, a dal vagy a panel módosítása nélkül játszhatunk az Instr ablakban kiválasztott hangszeren. Ha a szerkesztés gombot bekapcsoljuk, és a helyőrrel valamelyik sáv első, hangmagasság mezőjére állunk, akkor egy billentyű lenyomásakor nem csak megszólal a hangszer, de a billentyűnek megfelelő hangjegy is bekerül a panelbe. Az Octave ablakban az alsó billentyűsor (Y, X, C, V stb.) által megadott oktávot állíthatjuk be. A valós idejű szerkesztés ugyanígy történik, de szerkesztés üzemmódban előtte a Play Pattern gombra kell kattintanunk, mire a panel lejátszása megkezdődik, és a lenyomott billentyűk ugyanúgy bekerülnek a panelbe, mint kézi szerkesztés esetén. A hangjegyeket a DELETE billentyűvel törölhetjük. A billentyűzet segítségével nemcsak játszani és szerkeszteni tudunk, hanem a program egyéb lehetőségeit is vezérelhetjük: lejátszás indítása, leállítás stb. A billentyűzet és az egér használatát hamar meg lehet szokni, és nemsokára már nagyon gyorsan írhatunk dalokat. A billentyűzetről elérhető lehetőségeket a *2. táblázatban* foglaltuk össze.

Szerkesztés MIDI-billentyűzettel

Ha a Linux az ALSA-meghajtókat használja, és a SoundTrackert is ALSA-támogatásra állítottuk be, akkor a szerkesztéshez MIDI-billentyűzetet is használhatunk. Ez teljesen megegyezik azzal, mint amikor a számítógép billentyűzetével szerkesztjük a paneleket, de természetesen egy igazi MIDI-billentyűzet, mely egy zongorára emlékeztet, jóval elegánsabb és kényelmesebb megoldás. Válasszuk a *Settings* menü *MIDI Configuration* pontját. Ha itt a *Volume* gombra kattintunk, a SoundTracker a MIDI Velocity (sebesség) értékét használja hangerőként. A csatornagombokra kattintva a MIDI-bemenetet egy adott sávra korlátozhatjuk, tehát az 1-es MIDI-csatorna az első, a 2-es a második sávra vonatkozik stb. Ekkor azonban a MIDI-billentyűzet MIDI Out csatornáját is át kell

2. táblázat A billentyűzetkiosztás

Az alábbiakat és a billentyűzetet leíró táblázatokat a SoundTracker forráscsomagjában található README fájlból ollóztuk ki, és *Michael Krause* engedélyével közöljük.

Figyelem, néhány szolgáltatás csak a billentyűzet segítségével érhető el. A parancsok jó része a nagyszerű amigás ProTrackerben megszokott billentyűkön található. A betűk és a számok pedig egy zongora billentyűit utánozzák (lásd *1. táblázat*).

Sávszerkesztő

Jobb Ctrl	Dal lejátszása
Jobb Alt	Panel lejátszása
Jobb Winmenü	Csak az élő lépést játssza le
Space	Leállítás; szerkesztő üzemmód ki-be
Escape	Szerkesztő üzemmód ki-be; a lejátszás nem áll le
Shift+Space	Többcsatornás szerkesztés („polifónia”)
F1, F2...F7	Oktávváltás
Bal Ctrl + 1, 2...8	Egy hangjegy beírása után hány lépést ugorjon le a szerkesztő
Jobbra/balra	Mozgás az oszlopok és sávok között
Fel/le	Mozgás a lépések között
PageUp/PageDown	Gyorsabb mozgás a lépések között
F9	A 0. lépésre ugrik
F10	A panel negyedéhez ugrik
F11	A panel feléhez ugrik
F12	A panel háromnegyedéhez ugrik
Tab	Csatornaváltás jobbra
Shift+Tab	Csatornaváltás balra
Bal Ctrl + balra/jobbra	Előző és következő hangszer (Bal shifttel gyorsabban)
Bal Ctrl + le/föl	Előző és következő hangminta (Bal shifttel gyorsabban)
Bal Alt + balra/jobbra	Előző és következő panel (Bal shifttel gyorsabban)
Bal Ctrl + b, majd nyíl billentyűk	Kijelölés
Bal Ctrl + c	Másolás
Bal Ctrl + x	Kivágás
Bal Ctrl + v	Beillesztés
Bal Shift + F3	Sáv kivágása
Bal Shift + F4	Sáv másolása
Bal Shift + F5	Sáv beillesztése
Bal Alt + F3	Panel kivágása
Bal Alt + F4	Panel másolása
Bal Alt + F5	Panel beillesztése
Delete	Éppen a lépésnél található hangjegy törlése
←	Az élő hangjegy törlése és visszalépés eggyel
Insert	Lépés beszúrása; az egész sávot egy lépéssel lejjebb mozgatja
A többi billentyű	Játék és szerkesztés

Ha a SoundTracker nem képes beállítani a billentyűzeten található hangokat, akkor ezt a *Keyboard Configuration* ablakban magunknak kell elvégeznünk. A „nincs hang” (---) jelet beillesztő billentyűt viszont minden esetben külön kell beállítanunk. A többi billentyűparancsot a GTK+-ban megszokott módon állíthatjuk be: menjünk a kérdéses menüpontra, kattintsunk rá, s még az egérgomb elengedése előtt nyomjuk le a választott billentyűkombinációt.

Hangszerszerkesztő

A *Burkológörbe-szerkesztő* (Envelope Editor) üzemmódban a CTRL és a középső egérgomb együttes lenyomásával nagyíthatjuk vagy kicsinyíthetjük a görbét. A középső gomb önmagában a kijelzőt görgeti, új pontokat pedig a bal egérgombbal hozhatunk létre.

Hangmintaszerkesztő

Az ismétlődő rész (hurok) határait a SHIFT és a bal/jobb egérgombok segítségével állíthatjuk be.

állítanunk az itt meghatározott sávra. Ha a gombot nem kapcsoljuk be, akkor a szokásos módon a TAB és SHIFT+TAB billentyűkkel válthatunk a sávok között. A megfelelő ügyfél- és kapuszámokat is állítsuk be. Nálam minden működött az alapértékekkel.

A modulszerkesztés

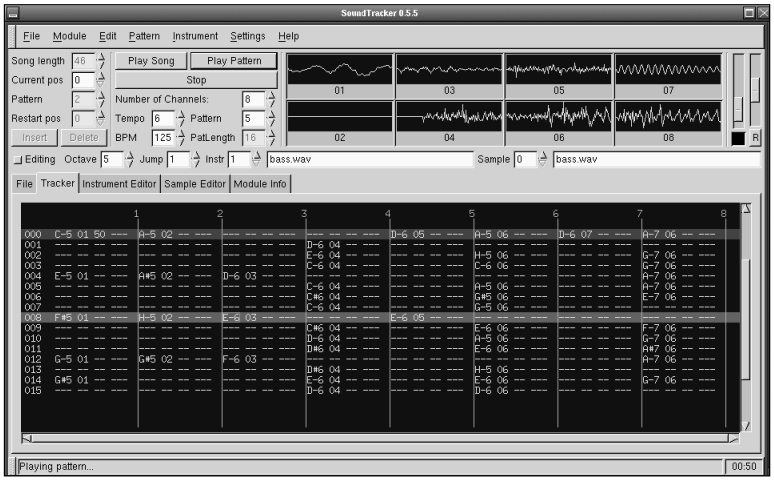
A szerkesztés, vagyis a zenekészítés a SoundTrackerben egy öt lépésből álló egyszerű művelet:

1. Állítsuk be a panel számát és hosszát az ablak bal felső részén lévő vezérlőkkel.
2. Kattintsunk a *Editing* gombra, s válasszuk ki az 1-es hangszert.
3. Kattintsunk a *Sample Editor* (Hangmintaszerkesztő) fülre, hogy betöltsük a hangszerhez választott hangmintát.
4. Kattintsunk a *Tracker* fülre, a nyíl billentyűkkel álljunk valamelyik sáv első, hangmagasság oszlopára, és a fentebb ismertetett módszerrel hozzuk létre a dallamot a számítógép- vagy a MIDI-billentyűzet segítségével. Ha valamit elrontottunk, használjuk a DELETE billentyűt.
5. Ismételjük e lépéseket az összes használni kívánt sáv esetében. Szerkesztés közben bármikor visszahallgathatjuk addigi próbálkozásaink eredményét, majd ismét visszatérhetünk a hangjegyek és hatások beírásához.

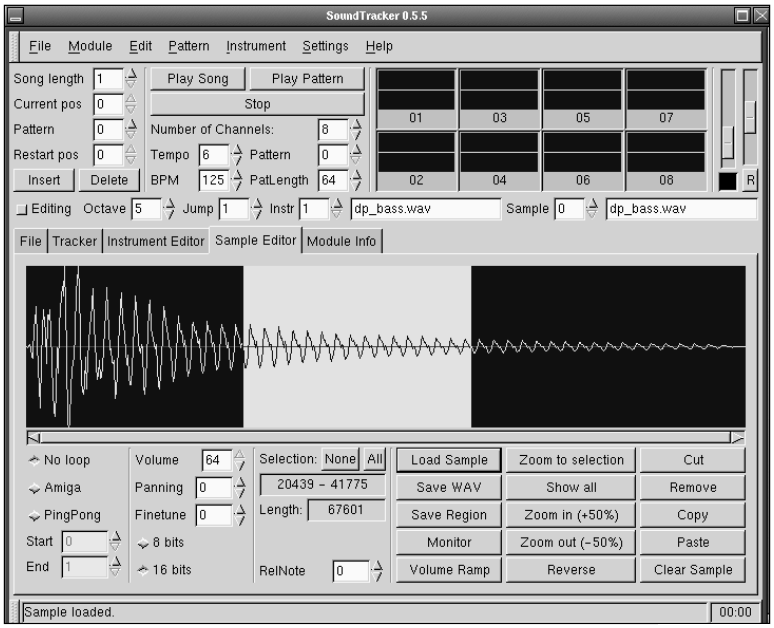
A lépések

Már volt szó arról, hogy a sávok egy-egy lépése négy oszlopból áll, ezek a hangmagasságot, a megszólaltatni kívánt hangszer számát, a hangerejét és a hatásait határozzák meg. Az utolsó oszlop a hatások megadására szolgál. A hatásparancs és annak értékei összesen három karakter hosszúságúak lehetnek. Csak a hangmagasság oszlopában használhatjuk az ismerős „hangjegy-oktáv” alakot, a többi oszlopban, a beállítástól függően a tízes vagy a tizenhatos számrendszerbeli számokat kell megadnunk.

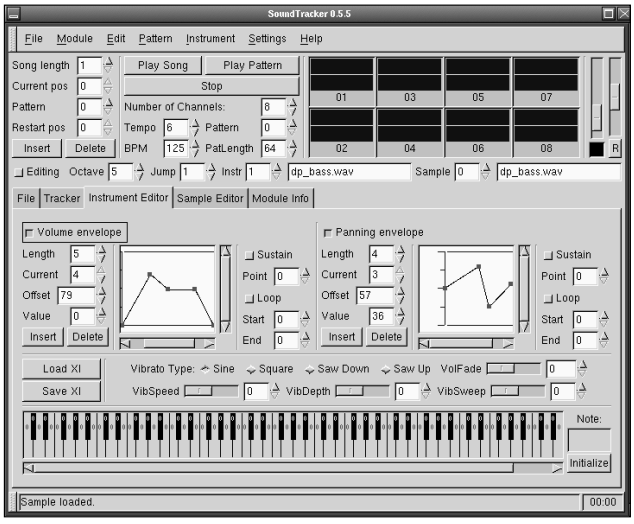
A hatásparancsokkal nemcsak hagyományos hanghatásokat – vibrato, tremolo, LFO, szűrés stb. – érhetünk el. Itt határozhatjuk meg azt is, hogy a hangminta lejátszása annak melyik részétől induljon, de akár a panelt is megtörhetjük, vagy ismétlődő részeket állíthatunk be.



4. kép Egy valamivel összetettebb panel



5. kép A Soundtracker Sample Editor (Hangmintaszerkesztője)



6. kép Instrument Editor (A Hangszerszerkesztő)

Egy másik nagyon hasznos lehetőség a finomhangolás megadása, ugyanis majd zeneszerkesztés közben rájövünk, hogy sokszor a félhangnál finomabb beállításra van szükség a pontos időzítéshez, illetve a disszonancia elkerüléséhez.

A *2. kép* kiemelt sora egy jellegzetes lépést mutat be: egy hangszert a C 6 hangmagasságon, alapértelmezett hangerővel és hatások nélkül szólaltatunk meg. A lépés egy négylépéses panel elején áll.

A *3. képen* a C-dúr hangsoron zongorázunk végig, 16 lépésen keresztül, C 6-tól C 7-ig – a kiemelt lépés az F-6 hangnál található. Az 1. és 2. képeken a lejátszási sebességet 100 bpm-re (beats per minute, vagyis az ütemek száma másodpercenként) állítottuk. A 2. és 3. képen két csatorna van jelen, ezek közül a második teljesen üres. Figyeljük meg, hogy a sebességet két vezérlő, a Tempo és a BPM határozza meg. Az alapértelmezett hatos Tempo esetén a BPM valóban a másodpercenkénti ütemszámot jelöli, kisebb érték megadásával azonban gyorsíthatunk ezen. Az itt beállított sebesség az egész dalra vonatkozik, de akár lépésenként is állíthatjuk az F hatásparancs segítségével. A panelek hosszúsága és a csatornák száma az általános részben módosítható.

A *4. képen* egy bonyolultabb dalszerkezetet látunk, melyet egy létező dalból vettünk át. A kép az ötödik panel lejátszása közben készült, így a jobb felső sarokban lévő oszcilloszkópok működés közben láthatók. A panel tizenhat lépésből áll; basszusgitár, gitár, dobok és cinek hallhatók benne.

Hogyan lesz a panelből dal?

A panelek elkészítése után megadhatjuk, hogy azokat milyen sorrendben és hányszor szeretnénk lejátszani. Nagyon egyszerű dolgunk van: a bal felső sarokban látható *Song length* mezőben állíthatjuk be, hogy a dal összesen hány panelváltás hosszúságú legyen, az INSERT és DELETE gombokkal pedig beszúrhatunk vagy törölhetünk egy lépést. A *Current Position* mezőben állíthatjuk az éppen szükséges lépést, a *Pattern* mezőben pedig az ahhoz tartozó panel számát. A *4. képen* látható példán a dal 46 panelváltásból áll. A Tracker ablakban természetesen egyszerre csak egy panel látható, de a Current Position mező léptetésével előre-hátra mozoghatunk és megtudhatjuk, hogy például a 0–3. lépéseknél a második panel, a 4–6. lépéseknél pedig az első panel szólal meg stb.

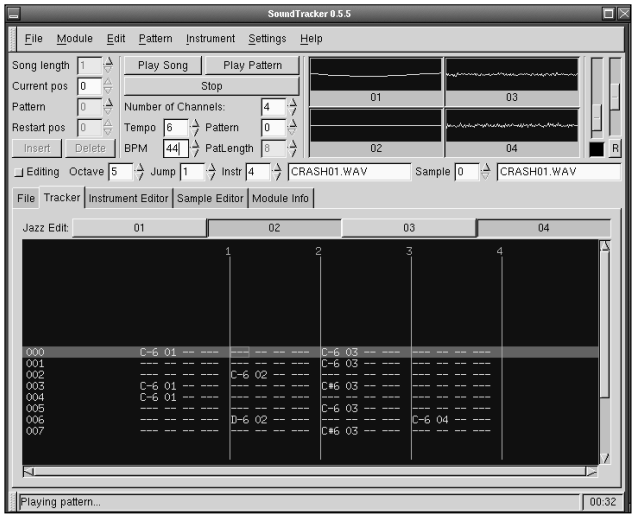
Ha kész a dal, a modult az alapértelmezett XM (Extended Module) formátumban menthetjük ki, de akár egyetlen WAV fájlt is készíthetünk belőle, ezt pedig MP3 formátumúra is alakíthatjuk, és így anélkül terjeszthetjük az Interneten, hogy bárki „ellophatna” belőle hangmintákat.

A Soundtracker további lehetőségei

Az *5. képen* egy basszushangszer hullámformáját látjuk, melyet a hangmintaszerkesztőbe töltöttünk be. Ezt a szerkesztőt a program írói kisebb feladatok elvégzésére tervezték, tehát csupán az alapvető lehetőségeket – másolás, kivágás, beillesztés, hurok- és hangerőbeállítás, a hang sztereó térben elfoglalt helye, finomhangolás – találjuk meg benne. Használhatunk 8 és 16 bites hangszereket is, és ezeket keverhetjük is egy modulon belül.

Saját hangmintáink rögzítéséhez először kattintsunk a *Monitor* gombra; ekkor a bemenő jelet hallgathatjuk. Ha a megfelelő helyre érkeztünk, a *Start Sampling* gombbal indíthatjuk a felvételt.

A *6. képen* a *Instrument Editor* (Hangszerszerkesztőt) láthatjuk,



7. kép A modul szerkesztése lejátszás közben (Jazz Edit Mode)

ezzel a hangminták hangerejének változását és térbeli helyét meghatározó burkológörbéket állíthatjuk be, akár a dal lejátszása közben is. Itt határozhatjuk meg azt is, hogy az egyes billentyűkhöz milyen hangszerek tartozzanak.

A hangszereket az *Instrument* (Hangszer) menüben vagy a hangszerkesztőben menthetjük, XI formátumban. Ez a fájlformátum a hurkok határait, a burkológörbéket és a vibrato beállításait is tárolja, így kifejezetten alkalmas a moduljainkban használt kedvenc hangszereink mentésére. A Sample Editorban megvághatjuk a hangmintákat, majd az Instrument Editorban a megfelelő változtatásokat elvégezve menthetjük azokat. Az XI formátumról a program doc könyvtárában található xi.txt fájlban részletes ismertetést olvashatunk.

A *Tracker* lapra váltva válasszuk az *Edit* menü *Jazz Edit Mode* (szerkesztés lejátszás közben) pontját, mire a Tracker lap tetején a modulban használt csatornák számával megegyező számú gomb jelenik meg. (*7. kép)* Ezekre egyesével kattintva jelölhetjük ki a Jazz Edit Mode üzemmódban használni kívánt csatornákat. Ha a lejátszás közben lenyomunk egy billentyűt, az annak megfelelő hang az elsőként kiválasztott csatornára kerül, majd a helyőr azonnal továbblép a következő olyan csatornára, melynek gombját bekapcsoltuk, és az új hangjegy már ide kerül.

A panel négy csatornájából a másodikat és a negyediket használjuk az azonnali szerkesztéskor. Miután a második sávra került egy hangjegy, a következőt már a negyedikbe fogjuk írni, a harmadikat ismét a másodikba stb. Az üzemmódot a TAB vagy a SHIFT+TAB billentyűkkel szakíthatjuk meg.

A Jazz Edit Mode igen hasznos segítőtársa lehet az alkotó zenészeknek. A 7. képen is láthatjuk, hogy az üzemmódban nem használt csatornákban lévő hangok zavartalanul szólnak, de közben folyamatosan szerkeszthetjük a többi csatorna tartalmát. Természetesen játék közben hangszert is válthatunk, vagy akár új hangmintát is betölthetünk. Lejátszás közben azon mezők tartalmát változtathatjuk meg, melyek nem váltanak át szürke színre.

A billentyűzetet könnyen átváltoztathatjuk „igazi” hangszerré: ehhez az *Instrument Editor Volume Envelope* és *Sustain* gombjaira kell kattintani. Ezután egy billentyű elengedésekor a hang nem hallgat el, hanem a burkológörbékkel meghatározott lecsengés szerint viselkedik. Ezen lehetőség használatakor érdemes az átmeneti tároló (audio puffer) méretét a lehető legkisebbre állítani. Ehhez válasszuk a *Settings* menü *Audio Configuration* pontját, majd a párbeszédablak tetején lévő *Editing Output* vezérlővel végezzük el a megfelelő módosítást. A billentyűzeten ettől fogva több szólamban játszhatunk, a szólamok számát a Jazz Edit Mode üzemmódhoz beállított csatornák száma határozza meg.

Kapcsolódó címek

Modulok

MODPlug Central (hatalmas modulgyűjtemény)
➲ http://www.modplugcentral.com/
United Trackers (minden, ami modul...)
➲ http://www.united-trackers.org/
MAZ Sound (rengeteg nagyszerű modul)
➲ http://www.maz-sound.com/
Linuxos zeneprogramok listája
➲ http://sound.condorow.net/
MOD Archive (óriási, jól szervezett gyűjtemény)
➲ http://www.modarchive.com/

Programok

MikMod (modullejátszó)
➲ http://mikmod.darkorb.net/
MODPlug bővítmény az XMMS-hez
➲ http://modplug-xmms.sourceforge.net/
SoundTracker (a létező legjobb linuxos modulszerkesztő)
http://www.soundtracker.org/
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.binaries.mods

Összefoglalás

A SoundTracker az általam ismert legfejlettebb képességekkel bíró linuxos modulszerkesztő, és jó hír, hogy fejlesztése a mai napig is tart. (A munkát a szerző vezeti, aki a program levelezési listáján szívesen fogad javaslatokat, hibabejelentéseket vagy akár kész programrészleteket is.) Az alkalmazás teljesítménye meggyőző, kezelése egyszerűen elsajátítható. A SoundTrackert a kezdő zenészek jól használhatják zeneírásra, a profikat pedig talán az fogja meg benne, hogy egy adott ritmusra villámgyorsan kipróbálhatnak új dallamokat, és így a program kiváló környezetet teremt az alkotó zenészek számára.

A szerző köszönetet mond Michael Krause-nak és „Mister X”-nek segítő megjegyzéseikért.

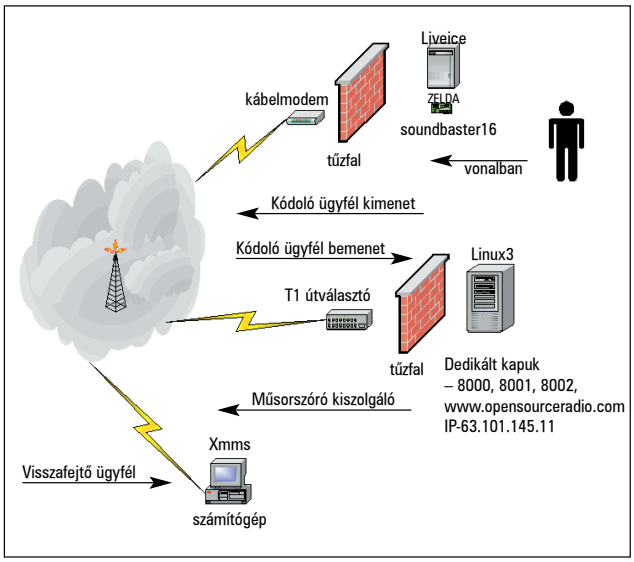
Dave Phillips
felügyeli a linuxos zeneprogramokkal foglalkozó Linux Music & Sound Applications honlapot, ő maga pedig több mint harminc éve zenél. Zeneprogramokkal 1985 óta dolgozik, Linuxot pedig 1995 óta használ. Részt vett a MIT Press kiadónál 2000-ben megjelent *The Csound Book* című könyv összeállításában, 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.

Internetes rádió nyílt forrású programokkal

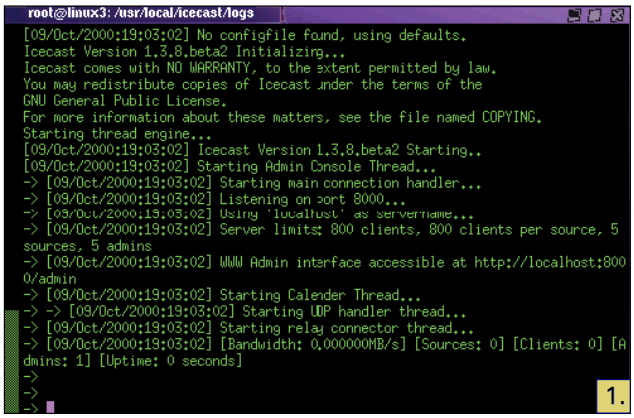
Hét Linux-őrült elhatározta, hogy saját internetes rádióállomás kiépítésébe kezd.

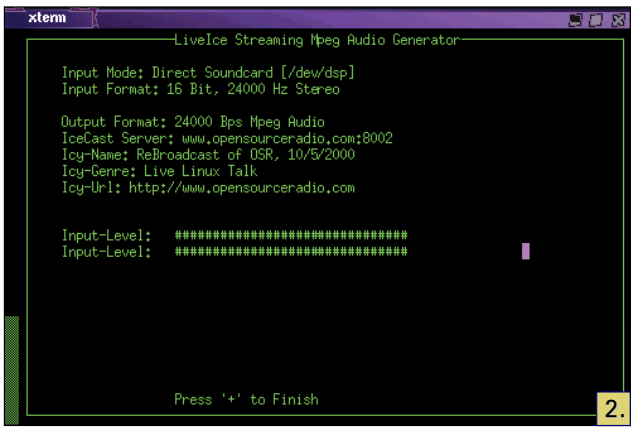
A történet úgy kezdődött, hogy öt barát kitalálta, az internetes rádióadás „tök jó”. A csapat hamarosan héttagúvá bővült, és a ➲ http://www.opensourceradio.com/ címen nemsokára meg is kezdődött az adás. Az állomás létrejöttében a Nyílt Forrás közösségnek óriási szerepe volt, hiszen az összes használt program nyílt forráskódú. Az Interneten történő sugárzáshoz két fő elem szükséges: a műsorszóró (broadcast server) és a kódoló ügyfél (encoding client). A kiszolgáló és az ügyfél futhat ugyanazon a gépen is, de a mi rendszerünk két, egymástól viszonylag távoli gépből áll.

A műsor tehát egy MP3-ba kódolt folyam. A LAME és a Liveice alakítja a hangkimenetet MP3 formátumra, s így a Winamp, az XMMS vagy bármely más lejátszó segítségével hallgatható az adás. Az MP3-ba kódolt anyagokra jelenleg nem vonatkozik semmilyen felhasználási szerződés, de várható, hogy a formátum jogait birtokló szervezet később begyűjti a 2001. évre esedékes díjakat. Ezen kérdésekről a ➲ http://www.mp3licensing.com/ címen olvashatnak az érdeklődők. Ha a jogdíjakat birtokló csoport végül is bevezeti a kötelező jutalékfizetést, akkor mi azonnal átállunk egy másik kódolási eljárásra. A közeljövőben a műsorszórón az Ogg Vorbis nevű, teljesen ingyenes és szabadalommentes hangformátumot fogjuk használni. Az Ogg Vorbisról a ➲ http://www.xiph.org/ címen és a Linuxvilág 2001. januári számában olvashatnak az érdeklődők. Még egy fontos megjegyzés: az MP3 szerződése nem érinti a sugárzott hanganyagra vonatkozó szerzői jogokat. Ha olyan anyagot szeretnénk a műsorba tenni, melynek szerzői jogait nem mi birtokoljuk, akkor mindenképpen engedélyt kell kérnünk a jogtulajdonostól.



Az internetes rádióadó hálózatának vázlata





1.-2. kép A Livice bejelentkező képernyője

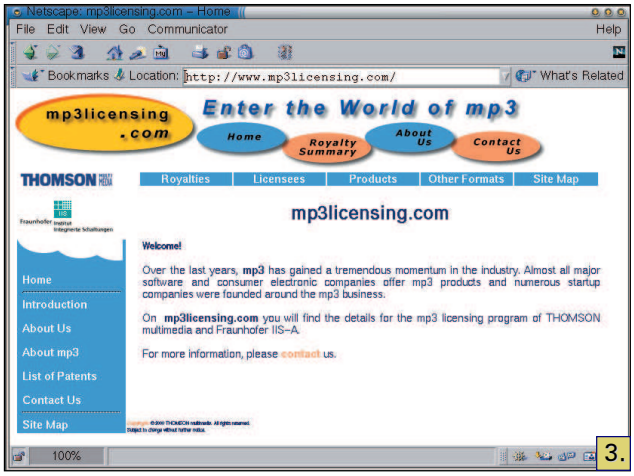
A kiszolgáló tulajdonságai

Szerettük volna a lehető leghamarabb elkezdeni a munkát, szóval vettünk egy tartománynevet, egy állandó IP-címet és egy nyílt kapukat használó kiszolgálót. Az első kettő nem feltétel, viszont így a hallgatóink mindig ugyanazon a címen érhetnek el bennünket, és ez fontos volt nekünk. A kiszolgáló T1-es kapcsolattal csatlakozik az Internethez. A nagy sávszélességre mindenképpen szükség van, hiszen csak így nyújtható a megfelelő adásminőség sok hallgató számára. A műsorszórót (Linux3) és az állandó IP-címet a ➲ www.doitwebcorp.com szolgáltatta számunkra. A kiszolgáló egy hagyományos, hálózatba kötött PC, melyen RedHat Linux 6.2 fut.

A cikk hátralévő részében a rádióállomásunk egyes alkatrészeinek beállításával foglalkozunk. A közben előfordult gondokra is kitérünk. Tisztában kell lennünk azzal, hogy egy internetes rádióállomás kialakításához sokféle módszer használható, mi azonban úgy igyekeztünk a lehetőségekből válogatni, hogy egyetlen gyártóhoz se kötődjön a rendszer. Az *ábrán* a hálózat vázlatos képe látható. A stúdióban használt mikrofonokat közvetlenül a Liveice-ot futtató kódoló ügyfélgép vonalbemenetére vezetjük. A Liveice rögzíti a hangfolyamot, és a LAME segítségével az analóg jeleket digitális formára alakítja. Ezt a Liveice a műsorszóróhoz továbbítja, melyen Icecast fut. Az Icecast a ➲ http://www.opensourceradio.com:8000/ címen teszi elérhetővé az MP3-folyamot, melyet ezután bárki meghallgathat egy arra alkalmas MP3-lejátszóval.

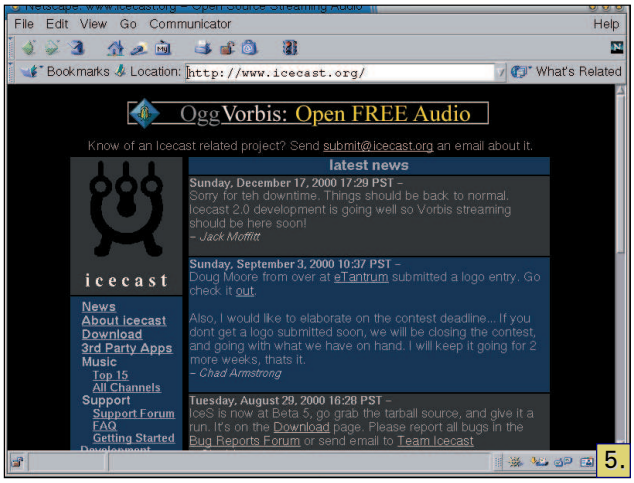
1. lista Az Icecast beállításfájlja

```
##############################################
# icecast.conf
# Olvassuk el az alábbi részt, úgy biztos,
# hogy semmi butaságot nem teszünk.
##############################################

###### A kiszolgáló helye és a felelős személy
       elérhetősége
# Itt a kiszolgálóval kapcsolatos kiegészítő
# adatok találhatók,
# ide mindenképpen írjunk valamit. Ezen adatok
# akkor jelennek meg, ha a
# könyvtárakat listázzák, illetve a folyam
# fejlécében is látható
# (ha a lejátszóprogram ezt lehetővé teszi).
##############################################

location A Marstól nyugatra...
rp_email dj@opensourceradio.com
server_url http://www.opensourceradio.com/

########## A kiszolgáló követelményei ##########
# Itt adhatjuk meg az egyidejű kapcsolatok
# (hallgatók) legnagyobb számát.
# Az érték elérése után a kiszolgáló nem
# engedélyezi további hallgatók csatlakozását.
##############################################

max_clients 100
max_clients_per_source 100
max_sources 10
max_admins 5
throttle 1.0

######## Az MP3 folyam kiegészítő adatai #######
# Ez itt egy nem mindig működő lehetőség. Ha nem
# működik,
# akkor eléggé összekavarja a hallgatók felé
# továbbított jeleket.
##############################################

use_meta_data 0
streamurllock 0
streamtitletemplate %s
#streamurl http://yp.icecast.org
streamurl http://www.opensourceradio.com:8000
nametemplate %s
desctemplate %

############### Könyvtárkiszolgálók #############
# A keresőkre ingyen bejegyezhetjük saját
# Icecast
# kiszolgálónkat, így több hallgatóra tehetünk
# szert.
# A touch_freq az adatok frissítésének
# gyakorisága
# (pl. hallgatók száma, a folyammal kapcsolatos
# adatok stb.)
##############################################

icydir yp.shoutcast.com
icydir yp.breakfree.com
icydir yp.musicseek.net
icydir yp.van-pelt.com
icydir yp.radiostation.de
directory yp.icecast.org
directory yp.mp3.de
touch_freq 5

# A kiszolgáló IP- és kapubeállításai (Fontos!)
# Ha nem adunk meg gépnevet, az Icecast az
# összes létező kapun figyel
# (minden bizonnyal ezt mindenki így szeretné).
# Ha mégis arra van szükségünk,
# hogy csak egy IP-címre figyeljen, akkor
# állítsuk be azt a hostname értékkel.
# A port (kapuszám) magáért beszél. Az Icecast
# 1.3-as változatában minden kapcsolat
# (a rendszergazdák, a kódoló, az ügyfelek stb.)
# a 8000-es kaput használja.
##################################################

hostname 63.101.145.44
port 8000
port 8001
port 8002
server_name www.opensourceradio.com

###### A kiszolgáló központi naplófájlja #######
# Az Icecast itt tárolja a kapcsolatokra
# vonatkozó adatokat és a hibaüzeneteket.
##################################################

logfile icecast.log
accessfile access.log
usagefile usage.log
logfiledebuglevel 0
consoledebuglevel 0

############### Fordított feloldás ##############
# Ezt állítsuk 1-re, ha az IP-ket fel akarjuk
# oldani (fordított feloldással), illetve
# hagyjuk 0-n, ha csak az IP-kre van szükségünk
# (ez így gyorsabb lesz).
##################################################

reverse_lookups 1
##################################################

# A fájl többi részét mi az alapértelmezett
# értékeken hagytuk, de azért olvassuk el,
# hátha tartalmaznak számunkra valami hasznos
# beállítási lehetőséget.
##################################################
```



3.

4.



5.

A kiszolgáló beállítása

Először is letöltöttük az Icecast kiszolgálót a ➲ www.icecast.org címről. Mi az 1.3.7-es mellett döntöttünk, hiszen akkor ez volt a legmegbízhatóbb változat. Az alapértelmezés szerinti telepítési helyet választottuk (/usr/local/icecast). Az Icecast beállítása nagyon egyszerű, mindössze egyetlen fájlban kell a megfelelő módosításokat elvégeznünk. Az icecast.conf-ban található megjegyzések nagyon sokat segítettek. A módosított szakaszok lehetővé teszik, hogy a kódoló ügyfél közvetlenül az adáskiszolgálóra küldhesse a hangfolyamot. Kiszolgálónk IP-címe 63.101.145.11, melyhez a ➲ www.opensourceradio.com tartománynevet jegyeztettük be. Az alapértelmezés szerinti icecast.conf fájl és az általunk használt icecast.conf összehasonlításából kiderül, hogy pontosan mely értékeket változtattuk meg. Itt most természetesen csak egy részletet közölhetünk ebből a hosszú fájlból.

Az Icecast indítása

Az Icecast nagyon egyszerűen futtatható. Lépjünk be azon a felhasználónéven, melyen az Icecastot szeretnénk működtetni, majd adjuk ki a `/usr/local/icecast/bin/icecast` parancsot. Ezzel belépünk a program konzoljára. A `?` paranccsal az Icecastban használható kapcsolókat jeleníthetjük meg. Itt jegyzem meg, hogy a konzolban azonnal látjuk, ha valaki a géphez csatlakozik. Ha az Icecastet a `-b` kapcsolóval indítjuk, akkor a konzol a háttérben fut. Az *1. és 2. képen* az Icecast indítása látható.

Az Icecast konzolja

Az Icecast konzol nagyszerű eszköz, ennek segítségével a kiszolgáló minden tulajdonságát vezérelhetjük. Például a `kick` paranccsal a nem kívánatos hallgatókat „rúghatjuk ki”. Egy másik hasznos parancs a `dump`, mellyel egy folyamot rögzíthetünk fájlba. A parancsok teljes listája az Icecast webes felületén érhető el.

A webes felügyeleti rendszer

Az Icecastnak van webalapú kezelőfelülete is, ezt a *http://gazdanév.tartomány:kapu/admin* címen érhetünk el, ahol a *gazdanév.tartomány* a kiszolgáló neve, a *kapu* pedig az icecast.conf fájlban beállított kapucím (lásd a listát). Alapértelmezés szerint a webalapú Icecast-beállító segédprogram mindenki számára elérhető, tehát mindenképpen védjük jelszóval. A felülethez tartozó súgóból mindent megtudhatunk a kezeléssel kapcsolatban. A témák között a felhasználói azonosítás beállítása és a webes felület kikapcsolása is szerepel. A webes felület egyik leghasznosabb lehetősége a források és a hallgatói kapcsolatok figyelemmel kísérését lehetővé tevő oldal. A kezelőfelületet tetszés szerint beállíthatjuk, erről is tájékoztatást kaphatunk a súgóban.

A kiszolgáló biztonsága

Minden Internetre kapcsolt kiszolgálót védeni kell. Beüzemelés előtt mindenképpen olvassuk el az Icecast kiszolgálóhoz kapott leírást. A kiszolgálót a „nobody” felhasználóként vagy más, különleges jogok nélküli felhasználóként futtassuk.

Gondok

Az Icecast beállítása során mi semmilyen hibával nem találkoztunk. Az Icecast leginkább egy webkiszolgálóra emlékeztet, és a beállítása is nagyon egyszerű.

Követelmények a kódoló géphez

Mi egy távoli kódoló ügyfélprogramot használunk, ezt a feladatot nem az Icecastot futtató gépre bíztuk. Az Icecasthoz továbbított adatokat a Liveice ügyfélprogrammal hozzuk létre. Ez a program egy hagyományos munkaállomáson (Zelda) fut, melyen Mandrake Linux 7.2 és egy SoundBlaster (ES1371) hangkártya található. A SoundBlaster mellett döntöttünk, hiszen ez egy széles körben támogatott kártya, és céljainknak tökéletesen megfelel.

A kódoló beállítása

A Liveice program telepítését a ➲ http://www.icecast.org/ címen található tar fájl letöltésével kezdtük (RPM fájlt, sajnos nem találtunk), az untar a helyi könyvtárba bontja ki a fájlokat. A Liveice futtatásához előbb a `make` paranccsal bináris fájlokat kell készítenünk. A README fájlban minden ehhez szükséges tudnivalót elolvashatunk. Az egyszerűség kedvéért a fájlokat a /usr/local könyvtárba

telepítettük. A liveice.cfg fájlt úgy módosítottuk, hogy a mi műsorszórónkra mutasson. Hasonlítsuk össze az itt közölt icecast.conf és liveice.cfg fájlokat az alapértelmezett fájlokkal, és így mindenki számára nyilvánvalóvá válik, hogy hol kell változtatásokat eszközölni. A Liveice ügyfél README fájljában az értékek szerepéről részletesen is olvashatunk. A legfontosabbak: SERVER, PORT, PASSWORD és USE_LAME3.
A SERVER az Icecast kiszolgáló neve, a PORT pedig az általa használt kapuszám. A PASSWORD mezőnek meg kell egyeznie az Icecast jelszavával, különben nem jön létre közvetlen kapcsolat a Liveice és az Icecast között. A USE_LAME3 mezőben határozhatjuk meg az analóg–digitális átalakításhoz használt kódoló nevét.
A mi rendszerünkben az alábbi beállítások működnek, de más beállításokkal is elérhetünk hasonló eredményt.

```
################################################
# liveice beállításfájl
################################################
SERVER www.opensourceradio.com
PORT 8002
NAME ReBroadcast of OSR, 10/5/2000
GENRE Live Linux Talk
URL http://www.opensourceradio.com
PUBLIC 1
ICY_LOGIN
SAMPLE_RATE 24000
STEREO
SOUNDCARD
FULL_DUPLEX
USE_LAME3 lame
BITRATE 32000
VBR_QUALITY 1
NO_MIXER
PLAYLIST playlist
DECODER_COMMAND mpg123
MIX_CONTROL_MANUAL
CONTROL_FILE mix_command
TRACK_LOGFILE track.log
#SAVE_FILE /osr/osr_10_5.mp3
```

A LAME nevű alkalmazást a ➲ http://www.sulaco.org/ címről töltöttük le és a /usr/bin könyvtárba csomagoltuk ki. Ha olyan könyvtárba szeretnénk helyezni, mely nem szerepel a $PATH változóban, akkor a liveice.conf fájlban meg kell határoznunk a kódoló teljes elérési útját. A kódoló ahhoz szükséges, hogy hangunkat digitális (MP3) formátumra alakítsa, melyet az Icecast kiszolgáló közvetítésével az Interneten elérhetővé tehetünk. Mi a LAME mellett döntöttünk, de természetesen bármilyen más kódoló szóba jöhet.

A Liveice indítása

A Liveice indításához lépjünk be a /usr/local/liveice/bin könyvtárba és adjuk ki a liveice parancsot. Ehhez a műsorszóróhoz kell tudnunk kapcsolódni. A *3. képen* a Liveice indító képernyője látható.

Gondok

Sem a LAME, sem pedig a Liveice beállításakor nem ütköztünk gondokba. Megfelelnek a nyílt forrású programokkal szemben támasztott követelményeknek és felállításuk is egyszerű.

A vonalbemenet

Vettünk egy hangkeverőt, mely több mikrofon jelét képes a hangkártya vonalbemenetére vezetni. Az xqmixer a hangkártyát bemenetként ismeri fel, s az onnan érkező jelfolyam a LAME-hez kerül átalakításra, mely a kimenetet a Liveice-hoz továbbítja. Ezt a Liveice ügyfél az Icecast kiszolgálóhoz küldi, végül az Icecast elérhetővé teszi az Interneten az MP3-lejátszók számára. A két legnépszerűbb lejátszóprogram az Xmms és a Winamp.
A keverő sokféle lehetőséget kínál számunkra. A hangbemenetekre köthetünk CD-ROM-ot, mikrofonokat, vagy más eszközt. A műsorban bejátszott felvételeket és mikrofonos stúdióbeszélgetéseket is sugárzunk, így igazi rádiós hangulat alakul ki az Internet közvetítésével. A keverőn hat hangbemenet található, és minden műsorvezetőnek saját mikrofonja van.

Gondok

A Liveice bizonyos hangminőséget követel meg. A jelet kapó hangkártyát az operációs rendszer eszközeivel vezérelhetjük. Mi a xqmixer segítségével állítjuk be a hangkártyát. A xqmixerben a rögzítési hangerő irányítja a Liveice-hoz küldött jelet. Ha a rögzítési hangerő túl alacsony, akkor nem hallunk semmit. Ha túl erős a jel, akkor az eszköz levágja a jelszintet (elhagyja azokat a jeleket, melyeket nem képes kibocsátani), ettől a minőség erősen romolhat. A hangerőt próbálkozással állítottuk be. Egyszerűen indítsuk el a szolgáltatást, és hallgassuk egy másik gépen. A dolog nagyon egyszerű: ha nem hallunk semmit, akkor emeljük a rögzítési hangerőt, ha pedig recseg vagy torz, akkor vegyünk vissza egy csöppet.

Összegzés

A ➲ www.opensourceradio.com műsora minden csütörtökön, keleti idő szerint este nyolctól tízig hallható. Az adásban a nyílt forrású fejlesztésekkel kapcsolatos híreket osztjuk meg hallgatóinkkal. Az internetes rádiózás olyan egyszerű, hogy bárki képes megvalósítani. A keverő, a számítógépek és a T1-es kapcsolat kivételével minden teljes egészében ingyenes. A rádióállomás működéséhez szükséges programokat bárki letöltheti az Internetről.

Szerzők: Andy Faulkner, Rich Smith, Brad Taylor, Jim Bailey, Paul Mack, Jim Lemaster, Tom Hartel.
Közösen megvalósították egy régi álmukat. A nyílt forrású rádióállomás házigazdái foglalkozásukat tekintve programmérnökök, akik éjjelente betyárkodnak. A műsorokban mindennapi tapasztalataikról, és a munkájuk során kapott hírekről is szó esik. A rádiót a ➲ http://www.opensourceradio.com címen hallgathatjuk, a készítőknek pedig a dj@opensourceradio.com levélcímre írhatunk.

Kapcsolódó címek

Icecat kiszolgáló, Liveice ügyfél és LAME kódoló *(5. kép)*
➲ http://www.icecast.org/

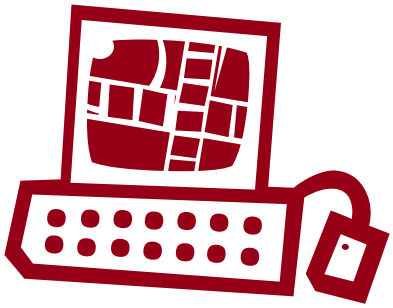
LAME *(3. kép)*
➲ http://www.sulaco.org/

Ogg Vorbis *(4. kép)*
➲ http://www.xiph.org/

Az MP3-mal kapcsolatos jogi tudnivalók
➲ http://www.mp3licensing.com/

Az OpenAL története

Nyílt forráskód és nyílt szabványok. A Loki egyik ingyenes programfejlesztésének áttekintése.



A Loki Entertainment Software-nél sokféle programmal foglalkozunk, a szabadon terjeszthetőtől kezdve a BSD-szerződéses forráskódon és a GPL, LGPL elvei szerint terjeszthető szabad programokon át a zárt forrású alkatrészmeghajtókig minden megfordul nálunk. Természetesen gépikódokat is írunk. A Linuxra átültetett játékaink nem mindegyike nyílt forrású, de saját, ingyen terjesztett munkánk a Fenris, a Setup, az SMPEG vagy az SDL programokban is megtalálható.
Miután sokszínű tevékenységünk során találkozunk a szabad és a nyílt forrású programok iránt elkötelezett alkotókkal is, így saját véleményünk és irányelveink is módosultak. Egyik fejlesztésünk például abból a felismerésből született, hogy a nyílt, jól leírt szabványokra van a legnagyobb szükség. Ez a fejlesztés az OpenAL volt, melynek története jó iskolapélda arra nézve, hogy milyen fontosak a szabványok – és hogy milyen nehéz megvalósítani őket.

Versenyhelyzet

A Heretic 2 és a Heavy Gear 2 linuxos változata esetében a Loki mérnökei új hanghatásokkal kezdtek el dolgozni, ezek messze felülmúlták a „balról vagy jobbról hallom?” típusú játékok hangjait. A Heretic 2 a hangkártyákat gyártó cégek számára is számos újdonsággal szolgált: az Aureal A3D-jének és a Creative EAX-jának támogatását is megkaptuk. Abban az időben mindkét gyártó túllépett a DirectSound3D lehetőségein, de teljesen különböző irányokban haladva kívánta megvalósítani a térbeli hangzást. A három API közé szorított windowsos játékfejlesztők (és akkor a Windows alatt elérhető számos hangfejlesztő környezetet még nem is vettük figyelembe) inkább nem okoztak maguknak további fejfájást, és kihagyták a térbeli hangzást a programokból (vagy esetleg egy-egy fejlesztőkörnyezet lehetőségeitől vérszemet kapva mégis belevetették magukat a téma kidolgozásába). Itt léptek közbe a gyártók: sokan közülük kiegészítőkkel (EAX, A3D) siettek a kártyák tulajdonságait jobban kihasználni igyekvő programozók segítségére. Ekkor került egyre több játék dobozára a „Térbeli hangzás” (Spatialized 3D sound) felirat, és a windowsos játékfejlesztők egy jó darabig nyugodtan dolgozhattak a meglévő fejlesztőkörnyezetek segítségével.
A Linux-felhasználók helyzete sajnos kedvezőtlenebb volt. A hangkártyák körében szabványosnak tekinthető SB16-hoz illeszkedő meghajtó már megbízhatóan működött, és a PCI-os hangkártyák is meglepően

jól működtek Linux alatt. A legfőbb hiányosság azonban az volt, hogy a kártyák legújabb lehetőségeit sem az OSS, sem pedig az ALSA alkalmazásával nem lehetett kihasználni.
Az Apple-felhasználók számára fejlesztő programozók hasonló helyzetben találták magukat: bár a Mac OS multimédiás API-jai összetettebbek és befejezettebbek voltak linuxos testvéreiknél, ennek ellenére a DirectSound szabványt még annyira sem támogatták, és az EAX, A3D lehetőségei is ismeretlenek voltak arrafelé. A windowsos fejlesztők számára fontos volt a programok hordozhatósága, egy csodálatos, rengeteg lehetőséggel bíró hangkörnyezetet kaptak, melyhez azonban nem tartozott hordozható API. Mivel a két másik résztvevő teljesen más utakat járt be, a dolgok kezdtek egyre rosszabbul alakulni – az időszakot jól jellemzi, hogy akkoriban a Microsoft nem jelentette meg a DirectSound3D NT-s változatát.
A hangkártyák piacvezetőjeként ismert Creative-nak ezzel egy időben kellett szembenéznie az Aureal nevű – a térbeli hangzás PC-s megvalósítását célul kitűző – céggel. Az Aureal a „hullámkövetés” (wavetracing) nevű eljárást helyezte fejlesztései középpontjába. Ennek lényege, hogy a tér tulajdonságait, amelyben a hang megszólal, átadjuk a hangkártya meghajtójának. Fontos megjegyeznünk, hogy a grafikával ellentétben a hangmódszerek a mai napig is főleg programból megvalósított megoldásokra épülnek. A fejlesztőkörnyezetek (Miles Sound System, QMixer) és az egymással versengő alkatrészmeghajtók jelentik a bizonyítékot erre. Az Aureal előnye a szimulációkkal, korai visszaverődések kezelésével és a hatásfok javításával kapcsolatos korábbi tapasztalataiban rejlett. Fejlesztéseit azonban hátráltatta a PC-s piac, ahol általánosan jellemző a „Na, vegyünk két ó’csó hangszórót is a masinához!” típusú hozzáállás. A minőségi hangfeldolgozás átköltöztetése egy olyan rendszerre, ahol hagyományosan sem a fejlesztők, sem pedig a felhasználók nem törődtek a térbeli hangzással, eleve vesztésre álló játszmának tűnt. Mindezek ellenére az Aurealnak sikerült valamit elérnie: felkeltette a felhasználók érdeklődését a jobb térbeli hangzású játékok iránt. Az első érdeklődők számára ez új hirdetési lehetőséget jelentett, és hamarosan minden játékfejlesztő megpróbált lépést tartani velük. A komoly fejlesztési tervekkel bíró Creative visszavágásképpen saját kártyáinak bővített lehetőségeit hangsúlyozta – ezek alapját egy egyszerűbb módszer, a kései visszaverődések valószínűségi zengetéssel megvalósított utánzása képezte.
A korai és kései visszaverődésekre alapuló megoldások kiegészítik egymást, az API-k és azok megvalósításai azonban teljesen különbözőek voltak. A Creative pedig jól választott: az EAX egyszerűbben használhatónak bizonyult, és a valószínűségi zengetés megfelelőbb megoldás volt az amúgy sem a csúcsminőségű hangrendszerekről híres PC-s piac számára.
Természetesen mindkét vállalat odafigyelt a Linuxra is. Az Aureal kidolgozott egy

A2D névre keresztelt rendszert, mely az A3D használatát tette lehetővé más gyártók kártyáin. Ez alapját képezhette volna akár egy nyílt forrású környezetnek is. Az Aureal kódbázisa hatékony volt, gépi kódban íródott, és ez a Linuxra történő átvitelt kissé megnehezítette volna. A Creative, linuxos programozók egy csapatával, meghajtók fejlesztésébe fogott, és az EAX linuxos megvalósításában látta a jövőt. Mindkét API a Microsoft DirectX és COM rendszereinél érvényes ajánlások figyelembevételével íródott, de az Aureal téralapú A3D API-jának tervezésénél az OpenGL-ből is vettek át ötleteket. Itt léptek a képbe a játékfejlesztők.

A nehéz kezdetek

Az EAX és az A3D előtt néhány játékfejlesztő egy hordozhatóbb hangrendszer megvalósításán törte a fejét. Az OpenGL-GameDev levelezőlista, melyen a hordozható kódok írásával foglalkozó fejlesztők fő fóruma 1998-ban egy új listát indított egy új, nyílt forrású hangkönyvtár megszületéséért. A lista neve – az OpenGL mintájára – OpenAL lett. Egyre több javaslat érkezett a listára, de hamar kiderült, hogy szinte mindenkinek másra lenne szüksége. Néhány listatagot csak a zene és a zeneszerzés érdekelt, ők főleg programból megvalósított hangképzéssel foglalkoztak azelőtt. Mások csak a térbeli hangzással foglalkoztak volna, mint a DirectSound3D idejében. Néhányan különleges, kifinomult OOP-megoldásokat szerettek volna az API-ban látni, míg mások beérték volna valami egyszerűbb, általánosabban használhatóval is. A kezdeti lelkesedés ellenére e véleménykülönbségek miatt csupán néhány fejlécfájl és egy, az elveket tartalmazó leírás készült el, és a fejlesztés félbeszakadt.

Talán nagyobb egyetértésre lett volna szükség, de a bukás fő oka az volt, hogy senki nem tudta igazán, hogy mi ennek az egésznek a célja. Ráadásul a hangkártyák tucatnyi új lehetőségét is kezelni kellett volna. Ez utóbbi végül is kétszer buktatta meg az OpenAL-t. 1999-ben a lista feltámadása látszott körvonalazódni: a Game Developer Magazine májusi számában *Jonathan Blow* cikke melletti oldalhasábon *Terry Sikes* (az OpenAL leírás szerzője) és jómagam az OpenAL céjait ismertettük. Akkoriban az Aurealnál egy javított, hordozható A3D kifejlesztésén gondolkodtak, mi pedig leírtuk, hogy az OpenAL képes lesz az OpenGL-lel való közvetlen együttműködésre, főleg a térgeometriai adatok feldolgozását illetően. Sem az Aurealnál, sem pedig az OpenAL levelezőlistán nem történt semmi változás a cikk hatására, és az állóvizet csak a Loki 1999 végén történő beszállása kavarta föl.

A Heretic 2 és a Heavy Gear 2 fejlesztése akkoriban kezdődött, és mivel nem voltunk (sőt, igazság szerint most sem vagyunk) abban a helyzetben, hogy az EAX-ot vagy az A3D-t támogassuk, nyilvánvalóvá vált,

hogy valamiféle megoldásra van szükség. Mindenképpen szerettük volna megvalósítani a DirectSound3D lehetőségeit (távolságérzékelés, Doppler-hatás, térbeli hangzás, hangmagasság- és iránykúpok) Linux alatt is. Ezenkívül természetesen célszerű volt a Creative-nál ás az Aurealnál érdeklődni a linuxos meghajtók felől. A Loki az eredeti OpenAL fórumon közelítette meg a fejlesztőket, felállított egy levelezőlistát, majd *Joseph I. Valenzuelát* megbízta az első elemek kifejlesztésével, majd mindhárman

munkához láttunk. *Michael Vance* (a Heavy Gear 2 fő programozója) és jómagam (a Heretic 2-n dolgoztam) elemeztük az eredeti OpenAL vitafórumot és a létező javaslatokat. Számos döntés meghozatalakor figyelembe vettük a régi hibákat, például legyenek-e OpenGL-szerű előírások, milyen legyen az API felépítése. Természetesen

a hang világa sok tekintetben különbözik a grafikától, de a felesleges egyezkedések elkerülése céljából sokszor a GL-ben alkalmazott eljárásokat vettük át. Az elején úgy terveztük, hogy az API nagy része az A3D-hez hasonló módon kezeli a térkörnyezetet, sőt, így okoskodtunk az OpenAL többi részével kapcsolatban is. A GL „utánzása” során eddig jutottunk: az OpenAL tulajdonképpen egy jelenetleíró API, mely számos, pontosan körülírt objektummal dolgozik. A GL-től más területeken viszont eltértünk: az OpenAL kevesebb belépési ponttal és több tokennel rendelkezik, ez azzal az előnnyel járt, hogy az API eljárásait úgy módosíthatjuk, hogy megőrizzük az ABI-val való megfelelőséget. A GL alrendszereihez (GL, GLX/WGL, GLU, GLUT) hasonlóan bevezettük az AL, ALC, ALU és ALUT alrendszereket, de szinte kizárólag az AL API-jával foglalkoztunk. Olyan régen vártunk már az OpenAL-ra, hogy nem bírtunk magunkkal és a Heavy Gear 2-be már be is építettük. Az OpenAL fejlesztésvezetőjének már ebben az első szakaszban komoly nehézségei támadtak az állandóan változó szabvány és maga a megvalósított könyvtár összehangolása terén. A Creative még jóval a San Joséban megrendezett 2000. évi Game Developer’s Conference (GDC) előtt értesített bennünket arról, hogy érdeklődik az OpenAL iránt. A saját linuxos meghajtójukkal kapcsolatos munkák kiegészítéseként, és tiszteletben tartva a hordozhatóságot és a széles körű elfogadottságot, a kártyagyártók számára egyre fontosabbá vált egy gyártóktól és operációs rendszerektől független hang API. A Microsoftról már régebben kiderült, hogy nem sok kedve van a DirectSound3D-t tovább fejleszteni, az Interactive 3-D Level 1 és 2 iránymutatásait közzétevő Interactive Audio Special Interest Group (IASIG) pedig nem kívánt az API-k szabványosításában részt venni. A Loki mindent elkövetett azért, hogy a szabvány és a megvalósítás ne csak a rövid távú igényeknek feleljen meg, mi pedig bíztunk a fejlesztés nyílt természetében (mi másért hívnák OpenAL-nak), de a Creative és más gyártók érdeklődése olyan váratlan meglepetés volt, mely megváltoztatta a szabályokat. Az OpenAL tehát segített bennünket abban, hogy a hangkártyák képességeit teljes mértékben kihasználó linuxos játékokat dobjunk piacra, majd a gyártók érdeklődése nyilvánvalóvá tette, hogy itt a lehetőség egy új szabvány megszületésére.

Beindult a dolog

A Loki és a Creative a GDC-n közölte szándéknyilatkozatát, mi pedig közzétettük *Michael Vince* szabványleírását. Azóta a követelmények némiképp megváltoztak, és az utolsó fél évben rengeteg kiegészítő munkára volt szükség. Bár eredetileg mi a térbeli hangzással kívántunk foglalkozni, de rájöttünk, hogy ugyanilyen fontos a sztereó és a többcsatornás formátumok támogatása, melyek közül egynéhány (például az MP3) nem igazán nyílt fejlesztés.
A tömörítés és a hangfolyamok nehéz témakörnek bizonyultak. E területeken nem is egy megoldás született már, és *Ian Ollman*, az OpenAL levelezőlista egyik tagjának javaslatára beépítettük az átmeneti tárolók sorkezelését (buffer queueing), ezzel lehetővé téve a hangfolyamok megvalósítását. Minél teljesebbé vált a kép, annál jobban izgultunk a végtermék jellege miatt.
A visszirányú csereszabatosság elvét nem egyszer, nem kétszer, hanem többször is megsértettük, régebbi játékainkhoz pedig bővítések kiadásával igyekeztük megoldani a régebbi, a szabvány új megvalósításából kimaradt lehetőségek támogatását. A Heavy Gear 2-t a Soldier of Fortune, majd a Sim City 3000 Unlimited követte. Manapság az Unreal Tournament is az OpenAL-t használja, és az UT-szerződésre épülő játékok is örökölni fogják e tulajdonságot. A Cognitoy csapat Mindroverje is OpenAL-t használ Linux alatt, és ők (sok más windowsos fejlesztőhöz hasonlóan) a Win32 OpenAL alkatrész-támogatással rendelkező meghajtóinak megjelenésére várnak. Mára szinte az összes OpenAL-t használó játék Linuxra íródott, de remélhetőleg nemsokára más felületek is bekapcsolódnak.
A Lokinak, illetve *Jean-Marc Jot*, *Garin Hiebert*, *Carlo Vogelsang* és mások révén a Creative-nak köszönhetően az OpenAL szabvány már az 1.0-s teljes változatnál tart, mely magában foglalja a DirectSound3D tulajdonságait, másrészt viszont eltér azoktól. Például az átmeneti tárolók (buffers) szigorúan elkülönülnek a forráskódtól: a kódok megosztva érik el azokat. Néhány változás inkább árnyalatnyi, például a viszonyítási távolságok közti különbségtétel, vagy a Doppler-számításokban használt viszonyítási sebesség pontos megfogalmazása. Néhány esetben, például a többcsatornás adatok kezelése során viszont szinte alig volt szükség változtatásra.
Ebben a szakaszban nem maga a szabványleírás számít igazi eredménynek, hanem amit a fejlesztés során tanultunk. Még mindig messze vagyunk attól, hogy befejezettnek tekintsük művünket, de a munka folyamata jól halad, létrejött a megfelelő párbeszéd és állandóan javítgatunk. Néhány dolgot az IETF-ből is kölcsönöztünk, no és az OpenGL bővítésének és módosításának működését is alapul vettük. A jelenleg általunk használt fő eszközcsoport – a DocBook DTD SGML változata és a megfelelő linuxos olvasó és formázó eszközök – elég jól bővíthető, és bár maga a leírás elég modulárissá vált, a kidolgozás és a leképezés még sok kívánnivalót hagy maga után. Számos további lehetőség beépítését tervezzük, de ezek megvalósítása még várat magára. Azon döntésünk, hogy az SGI OpenGL-t követjük, segített megőrizni az eredeti lendületet, mely a korábbi OpenAL próbálkozásokból mindig kiveszett az idő során. Az OpenAL név választása is egyértelmű: elkötelezett hívei vagyunk egy gyártófüggetlen, nyílt forrású rendszer kifejlesztésének – és azt hiszem, hogy eddig be is tartottuk az ígéretünket.
A következő mérföldkő az 1.0-s szabvány hivatalos lezárása lesz, melyet az OpenAL

windowsos, linuxos stb. változatainak megjelenése követ. Az OpenAL mindig hatékony meghajtótámogatást fog jelenteni, ehhez természetesen a Creative és más gyártók munkájára is szükség lesz. A Loki azt szeretné, ha a gyártók a nyílt forrású megoldásokat részesítenék előnyben, de a végső döntést maguk a cégek hozzák meg. Ez minden nyílt szabvány természetes velejárója – ahogy minden ingyenes programokat fejlesztő csapatnak joga van az OpenAL API kidolgozásához, a vállalatok ugyanígy saját változatok használata mellett is dönthetnek. A hatékony megvalósításokat és a fejlett eljárásokat mindenki fontosnak tartja, mégis eltart egy darabig, míg a nyílt forráskód elfogadottá válik.

A jelenlegi szabványleíráson túl az 1.1-es OpenAL-változathoz már egy sokkal letisztultabb tervünk van. Főként a Creative javaslatain alapuló IASIG I3DL2 előírások határozzák meg a használt lehetőségeket, ezeket gyártókra jellemző bővítésekként tervezzük megvalósítani, az 1.1-es változatban már gyártófüggetlenek lehetünk. Sok kérelmet, javaslatot át kell néznünk, és az OpenAL 1.0 esetében figyelmen kívül hagyott szolgáltatásokkal is foglalkoznunk kell. Ahhoz képest, hogy jelenleg a felhasználók száma meglehetősen csekély, meglepően sok váratlan javaslatot kaptunk. A környezetkezelő API-nak (ALC) több felületen is bizonyítania kell ahhoz, hogy (az OpenGL-lel ellentétben) egy teljes mértékben hordozható megoldást adjunk a fejlesztők és a felhasználók kezébe. Az API, melynél jelenleg csak a tokenek nagy számával oldható meg a kevés belépési ponttal való működés, talán jobban egyensúlyba kerül, ha a rendszer megbízhatónak bizonyul.

Mindeközben folytatjuk a játékok áthozását Linuxra. Bár mindegyik sajátos módon kezeli az erőforrásokat, úgy gondoljuk, hogy a jelenlegi API megfelelő lesz. A legújabb lehetőségeken (például a mikrofonos vezérlésen) kívül a munka fő részét mostanában a hangfolyamok kezelése és a további kodekek kifejlesztése jelenti. Talán kissé csúfondárosan hat, de a jelenlegi fejlesztési tervben nem szerepel a visszaverődések és más fejlett lehetőségek támogatása. Bár talán lenne értelme a visszaverődések kezelésének, de a játékok motorja általában sokkal pontosabban képes ezekre figyelni, és az OpenAL fejlesztésében egyre inkább azt az irányt követjük, hogy a programozó saját maga dönthesse el, hogy a jelenet miként „válaszol” az ehhez hasonló körülményekre. A fő szempont ezután az lett, hogy az OpenAL-t lehessen használni szerkesztésre, tehát nem a gyártójellemző lehetőségeken törtük a fejünket, és a cégek saját tulajdonú megoldásaiból sem akartunk mindenáron bizonyos elemeket átvenni. A Sensaura cég munkájának köszönhetően az AL-ból eredetileg kihagyott HRTF-ek is megjelentek, lehetséges bővítésként. Az Aureal fejlesztési tervében szereplő Dolby-támogatás is minden bizonnyal bekerül az OpenAL-ba.

A szórt műsorfolyam Khronos-féle, az SGI által támogatott megvalósítását az OpenAL-hoz kell még igazítanunk – az OpenGL-hez hasonlóan a jövőben az OpenAL-t is az együttműködés megkönnyítésére bővítésekkel látjuk el.

A tervezési és a programozási munka mellett az OpenAL foglalkozó fejlesztőcsoportot is át kell szerveznünk. Ami ma önkéntesek laza hálózata néhány vállalat támogatásával, annak holnap egységes, nem bevételközpontú szervezetként kell működnie. Ismét az OpenGL példájából kiindulva: létrejön egy elemző testület, egy szavazórendszer, és minden olyan elem, mely egy nyílt szabvány kialakításánál fontos lehet. Az OpenGL-lel ellentétben a példamegvalósítás és a próbakörnyezet nyílt forrású lesz. Az OpenGL-hez hasonlóan a bejegyzett védjegyet csak azok használhatják, akik átjutnak a hivatalos ellenőrzésen.

Mindennek a fontosságát a *Brian Paul* neve által fémjelzett Mesa is jelzi: ez a semmilyen más ingyenes fejlesztéshez nem hasonlítható letisztult projekt öt év alatt elvezetett a mai tökéletes 3D grafikával rendelkező Linuxhoz. A Mesa biztosította a keretet, a 3dfx linuxos Glide-ja pedig a meghajtót, melyek együttesen a linuxos játékokban is elterjedté tették a gépi leképezést. Az SGI-nek az OpenGL-en végzett gondos munkája és a Brian Paul számára nyújtott támogatásuk nélkül ma nem létezne a DRI nyílt forrású megoldás Linuxra, de valószínűleg az nVidia meghajtója sem. Ha kívánhatunk valamit, akkor az az, hogy az OpenAL ugyanazt a biztonságot jelentse a Linuxnak és más felületeknek, mint amit az OpenGL. A Loki számára most a legfontosabb, hogy az alkatrészek egyaránt támogassák a Linuxot és a Windowst. Végül az is nagy álmunk, hogy minden játékfejlesztő az új, nyílt forrású szabványokat részesítse előnyben a saját megoldások helyett.

Összegzés

Megérte vajon? Többet értünk el, mint amennyit eredetileg terveztünk, de ha az általunk elvégzett munka elősegítette egy újabb nyílt forrású API megszületését, akkor a válaszunk határozott igen. A Linuxnak csupán a közelmúltban kellett szembenéznie a szabványosítás nehézségeivel, és elmondhatom, hogy az IETF-fel szemben táplált tiszteletem megsokszorozódott az évek során. A Linuxnak szüksége lesz egy multimédiás API bővítésre és egy csak ezzel foglalkozó csoportosulásra azért, hogy a gépgyártók biztos alapokra építhessenek a jövőben – és akkor még nem is említettük azt a rengeteg ingyenes programot, melynek fejlesztői azért küzdenek, hogy műveik hordozhatók legyenek. A Linux-hívők valószínűleg alábecsülik a DirectX-nek a windowsos játékokra és magára a Windowsra gyakorolt hatását. Az is lehetséges, hogy sok linuxos fejlesztő alábecsüli az otthoni felhasználók igényeit. Azonban a Linux csak egy azon felületek közül, melyeknek egy átfogó, hordozható multimédiás API-ra lenne szüksége. Ha az OpenAL is hozzájárul ezen operációs rendszerektől független „OpenX” megvalósításához, akkor elmondhatjuk, hogy sokkal többet értünk el, mint amiért ezt az egészet elkezdtük.

Bernd Kreimeier játékfejlesztő, író és fizikus. A Loki programozójaként részt vett a Heretic 2, a Quake 3 és más híresség linuxos változatainak elkészítésében. Többek között ő felügyeli az OpenAL szabványleírását.

Kapcsolódó címek

Az OpenAL szabványleírása és a példamegvalósítás
➲ http://www.openal.org/

Az IASIG honlapja (I3DL1, I3DL2 ajánlások)
➲ http://www.iasig.org/

A Creative Labs fejlesztői oldala (EAX SDK)
➲ http://developer.creative.com/

Miles Sound System
➲ http://www.smacker.com/miles.htm

QSound Labs
➲ http://www.qsound.com/

Khronos Initiative
➲ http://www.khronos.org/

További honlapok a témában

A Loki Software nyílt forrású fejlesztései
➲ http://www.lokigames.com/development/

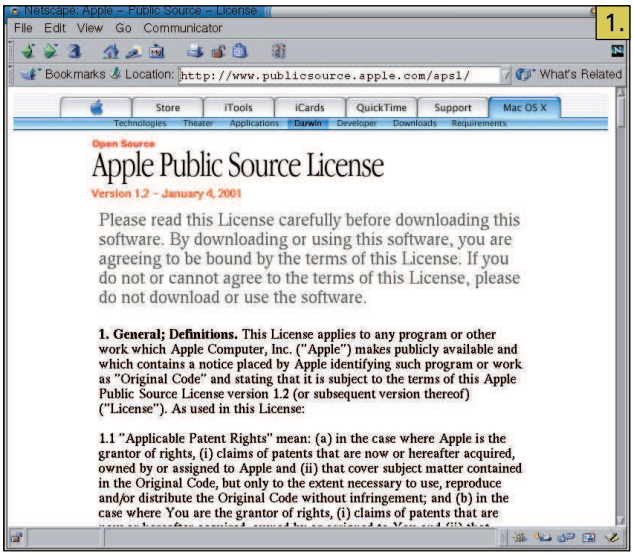
OpenGL ARB
➲ http://www.opengl.org/developers/faqs/arb_faq.html

Game Developer Magazine
➲ http://www.gdmag.com/

Folyamatos hang és kép a hálóra

Gondoljuk át, hogy milyen program- és gépigénnyel kell számolnunk, ha műsorszórásra adjuk a fejünket.

A korszerű megoldások nemhogy gyorsan, hanem robbanásszerűen alakulnak át, egyik napról a másikra. A különleges eljárások annyiféle új szakterület létrejöttéhez vezettek, hogy egy ember képtelen felfogni az egyes alrendszerek minden részletét, melyekből összeáll a végső kép, az éppen divatos új kifejezés. Az új felfedezések leírásához új szavakat ágyazunk egymásba, és ugyanígy évszázadok munkáját sűrítjük egyetlen elvont kifejezésbe, majd az elvont fogalmakat rendszerbe illesztve még összetettebb rendszereket alkotunk. Jogos a kérdés: szükségünk van-e egyáltalán arra, hogy e rendszerek legapróbb részleteinek működésével is tisztában legyünk? Nem lenne az egyszerűbb, ha a megfelelő dobozokat a kimeneti-bemeneti csatlakozók ismeretében csak összekapcsolnánk egymással, és már működne is a rendszer? A válasz egyértelmű nem. Nem kell a használt eljárások minden nyúlfarknyi részletét ismernünk, de ha a lehető leghatékonyabban kívánjuk kihasználni a rendelkezésünkre álló módszereket, akkor a látható daraboknál mélyebb szinten kell elemeznünk a rendszerek felépítését.



A multimédiás tartalom átvitele kifejezetten ez utóbbi szemléletmódot követeli meg. A multimédia kifejezés azt jelenti, hogy a tartalmat egyszerre két- vagy többféle típus (hang, mozgókép, kép, szöveg vagy bármilyen más, érzékszerveinkkel felfogható inger) alkotja. Cikkemben most a műsorszórással, azaz nem az időfüggetlen adatátvitellel, hanem az időfüggő, számítógépes multimédiás adatátvitellel foglalkozom, és bemutatom, hogy az egyes önálló, összetett eljárások együttes alkalmazásából hogyan áll elő a kívánt végeredmény. A műsor továbbításához szükséges lépéseket egyesével ismertetem, ezeket akár „fekete dobozokként" is tekinthetjük. Azonban a lépések tökéletes megvalósítása azt is igényli, hogy a folyamat minden egyes elemének működésével és szerepével is tisztában legyünk.

A műsorszóró rendszer áttekintése

A műsor továbbítása az adatátvitel egyik módja. Ez az adatközlés több forrásból több cél felé haladhat. Az adás célját fogadónak nevezzük. Természetesen bármelyik fogadó szolgálhat forrásként más fogadók számára. A források és a fogadók számától függően az adatközlés négy típusra bomlik: egy–egy, egy–több, több–egy és több–több. Mind a négy változat hasonló szolgáltatásokra épül, de a folyamatban szerepet játszhatnak az adott felállás hatékony működéséhez szükséges kiegészítő elemek is. A megfelelő protokollokat az Internet Engineering Task Force (IETF) fejlesztette ki. Ezek a műsorfolyamok minden lehetőségét magukban foglalják, az ipar azonban csak mostanában kezdi ezen elvek alapján megvalósítani a rendszereket. A gépi környezet oldaláról nézve minden rendszert úgy kell megtervezni, hogy legyen hely a későbbi bővítések számára. A csúcsminőségű berendezésekre az elején kiadott pénzekkel később lehet, hogy sokkal nagyobb összegeket takaríthatunk meg. Egy jellegzetes hiba, hogy a rendszer azon részeit fejlesztik, melyek nem befolyásolják a valóban jelen lévő, a teljesítményt akadályozó tényezőket. Írásomban az adatok létrehozásától indulok el, ismertetem a fogadó felé történő adatátvitel módját és igyekszem minden részletre kitérni.

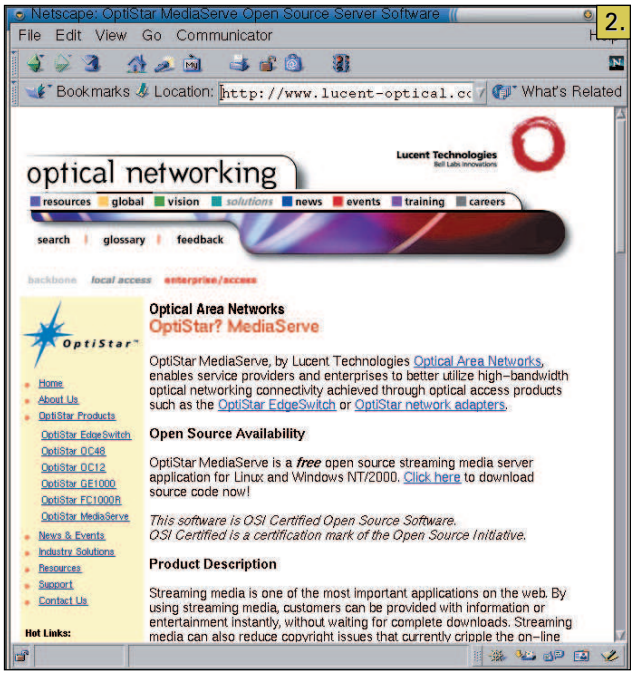
Az adatok létrehozása

Az adatok létrehozására egy egész ipar épül. Régebben „hollywoodi" minőségű anyag elkészítéséhez drága, csúcsteljesítményű munkaállomásokra volt szükség. A PC-forradalom eredményeképpen azonban mára a szerényebb anyagi lehetőségekkel rendelkező stúdiók, otthoni felhasználók is elvégezhetnek sok részfeladatot. Ezek eleinte főleg a Macintosh-felület felhasználói számára készültek el, később a Windows alatt is használhatóvá váltak, de ezek sokáig zárt, gyártófüggő rendszerek voltak.
A Linux nemrégiben kapcsolódott be ebbe a folyamatba. A Linux „ingyen sör" szellemisége mellett a felhasználók azért is szeretik ezt az operációs rendszert, mert függetleníti őket az egyes gyártók termékeitől. Az időfüggő tartalomkészítés a „hollywoodi" termelés sarokpontja, tehát egyik stúdió sem függhet egyetlen gyártótól sem. Saját üzletük védelmére képesnek kell lenniük az önálló munkavégzésre. A Visual Effects Society egy 24 vállalatból álló ipari csoportosulás, melynek tagjai a multimédiás adatok létrehozásához szükséges feladatokra alapozzák tevékenységüket. Ez a társaság bejelentette, hogy néhány éven belül átköltözik Linuxra, de ehhez az operációs rendszernek még fel kell nőnie. Szerencsére a műsorfolyamok Linux alatti megvalósítása és kezelése már régóta fejlődik, és a megfelelő úton jár.
Ha műsorszóró tevékenységet kívánunk végezni, a fő kérdések: Milyen típusú vállalkozást szeretnénk? Milyen adatok továbbítására lesz szükség? Élő mozgóképre? DVD vagy CD-ROM tartalomra? Tartalmaz-e a továbbítani kívánt folyam számítógéppel létrehozott grafikákat, vagy többféle típus összefonódásáról van szó? Hogyan fogjuk továbbítani a kívánt adattípusokat, esetleg elegyítjük egymással?
E kérdésekre először tervszinten kell válaszolnunk. Majd a válaszokat a rendelkezésre álló felszerelés, munkaerő, szakképzettség és anyagi erőforrások alapján elemezni kell. E tényezőket nemcsak az adatok létrehozása, hanem az azokhoz kapcsolódó szolgáltatások függvényében is figyelembe kell venni. Ha például azt tervezzük, hogy előre rögzített tartalmat szolgáltatunk fizető ügyfeleknek, akkor tisztában

kell lennünk célközönségünk igényeivel, természetével is. Minden egyes megtekintett adásért fizetni fognak, vagy csak az ingyenes adásra lenne igény? Milyen operációs rendszereket, ügyfélprogramokat (lejátszókat) kell támogatnunk? Ezek és más fontos kérdések mind befolyásolják az általunk beszerzendő gépeket és programokat.

Az adatok kódolása

Mi az adatkódolás és miért van erre szükség? A számítógépes adattovábbítás során a legfontosabb kódolási feladat az analóg–digitális átalakítás, más néven digitalizálás. Analóg világban élünk, a számítógépek viszont csak a digitális nyelvet beszélik. Mindennek, ami a rendszerbe kerül, egyesek és nullák sorozatából kell állnia. De nem csak digitalizálásra van szükség. A legtöbb anyag jogvédett és általános vélemény, hogy a szerzői jogokat meg kell védeni a törvénytelen másolástól és terjesztéstől. Éppen ezért a továbbítani kívánt adatokat az adó általában titkosítja, amit csak a jogosult felhasználók lejátszója fejthet vissza. Egy másik fontos tényező, hogy a hatékonyabb átvitelért az adatfolyamot a megfelelő eljárásokkal tömöríteni is kell. A kódoláshoz használni kívánt módszer a kódolóprogramtól és -alkatrésztől, a processzorteljesítménytől, a hálózat sebességétől és bizonyos adattárolási kérdésektől függ. Ha az adatokat közvetlenül az irányításunk alatt álló közönséghez továbbítjuk, akkor kérdéseink főleg a használni kívánt módszerekre irányulnak. Ha az általunk kiválasztott adattípust továbbítjuk a célközönség felé, akkor a szakmai kérdések mellett a szabványokról és a felhasználók igényeiről is beszélnünk kell.



2.

A kodek kifejezés a tartalom előkészítésekor és lejátszásakor szerepet játszó kódoló/visszafejtő eljárásra utal. A legtöbb kodek valakinek a tulajdonát képezi, de mivel ezek általában túl széles körben elterjedtek, ezért az ipar valószínűleg még jó ideig nem mozdul el a nyílt szabványok irányába.
A legnépszerűbb, céges kodekek a RealNetworks, a Microsoft és az Apple fejlesztőitől származnak. Egyre többen használják a Moving Picture Experts Group (MPEG) formátumait. Az MPEG formátumok előnye, hogy a legtöbben nemzetközi szabványként ismerik el őket, és ezek nagyon jó tömörítési arány mellett is viszonylag kicsi adatvesztéssel járnak. Jelenleg két változat, az MPEG-1 és az MPEG-2 van használatban, de az MPEG-7 2001-es várható megjelenéséig az MPEG-4-et is sokan alkalmazzák interaktív tartalom továbbítására. A legnépszerűbb változattal, az MPEG-2-vel kapcsolatos jogi tudnivalók a ➲ http://www.mpegla.com/ címen érhetők el. Bár a felhasználási szerződés nem ingyenes, a kódoló és visszafejtő eljárások bárki számára hozzáférhetők, és ezekre alapozva nyílt forrású megoldásokat is készíthetünk. Az MPEG LA szeptemberi bejelentésében azt közölte, hogy „szükség van arra, hogy egy felhasználási szerződés keretein belül bárki hozzáférhessen a nemzetközi MPEG-4 szabvány használatához szükséges szabadalmakhoz”.
Az MPEG-hez hasonló nyílt forrású szabványok legalább egy előnnyel járnak a műsoranyagaikat kódolni kívánó vállalatok számára: a jövőben soha nem kell visszatérniük egy gyártóhoz. Az MPEG-módszerhez annyi szabadalom kapcsolódik, hogy jelentős mérnöki teljesítmény volna, ha valaki képes lenne egy versenyképes, ingyenes választást felkínálni helyette.
Használatukhoz tisztában kell-e lennünk a kodekek működési elvével? Szerintem nem, de mindenképpen illik annyit tudnunk, hogy az egyes formátumok mekkora fájlméretet eredményeznek és mennyi kódolási/visszafejtési időt és processzorteljesítményt igényelnek.
A felhasználói igényeket is fel kell mérnünk, mielőtt eldöntenénk, hogy melyik kodeket kívánjuk használni. Azokhoz a kodekekhez, melyeknek tulajdonosa van, általában nem minden felületen érhető el ügyfélprogram (lejátszó). Azzal is tisztában kell lennünk, hogy a felhasználási szerződések pénzbe kerülnek, így a használt formátum a költségvetést is módosíthatja.

Az adatok tárolása, kiolvasása és továbbítása

Miután döntöttünk az alkalmazandó gépi környezetről és programokról, azt is meg kell határoznunk, hogy a felhasználói igényeket milyen típusú adattárolással elégíthetjük ki. Ha adataink jellege valós idejű folyam (például videokameráról érkező jel), akkor tudnunk kell, hogy a jelenleg használt kódolóegységek kártyánként egy élő bemenetet kezelnek le, emellett szükség van elegendő memóriára (RAM), melyben a kódolandó folyamot tároljuk, mielőtt az a hálózati kártyára kerül. Az egy–egy és egy–több továbbítás esetén ez magától értetődő. Minden egyes élő anyag átviteléhez egy-egy kódolókártyára, valamint számítógépre lesz szükség.
Ha az adatokat előre feldolgoztuk és az adás csak később vagy felhasználói igény szerint (Video-on-Demand, VOD) történik, akkor a szolgáltatásokhoz használt gép és program válhat a sebesség szempontjából a rendszer kényes pontjává. Következzen most egy példa a szükséges erőforrások felmérésére.
Tegyük fel, hogy tízezer nézőt szeretnénk kiszolgálni egy órán keresztül, folyamatos videoanyaggal, de csúcsidőben akár naponta két órán keresztül is. Tegyük fel, hogy egy átlagos felhasználónak 300 kb/mp sebességű folyam jelenti a remegésmentes, megszakítás nélküli mozgóképet. Azt is tételezzük fel, hogy egy 5000 filmből álló adatbázisból egyszerre átlagban százféle filmet néznek.
Az igény felső szintjét alapul véve kiderül az adattároláshoz és -kiolvasáshoz szükséges rendszerteljesítmény. A VOD nem engedélyezett, de a nézők választhatnak az előre betervezett programok közül.
Tegyük fel, hogy az útválasztónk képes úgynevezett üzenetszórásos (multicasting) üzemmódban működni, azaz egyszerre több ügyfél felé ugyanazt az adatfolyamot továbbítani. Így már elegendő adatunk van a tárolási és kiolvasási követelmények meghatározásához. A tárolási követelmény úgy számítható ki, hogy a filmek számát megszorozzuk a fájlok hosszával és a folyam átviteléhez igényelt sebességgel. Esetünkben ez 5000 film x 3600 másodperc x 300 000 bit/másodperc osztva nyolccal (egy bájt nyolc bitből áll) = 675 GB.
Milyen messze lehet a tárolóhely (a merevlemezek) az internetkapcsolattól a megfelelő átviteli minőség megvalósításához? Ennek kiszámítására az egyidejűleg kiolvasott fájlok számát kell az átlagos sebességgel megszorozni. Itt ez 100 film x 300000/8 = 3,75 MB/másodperc.

Ha a VOD-ot is engedélyeznénk, akkor ez a szám 8000-szeresére is növekedhetne, s így 30 GB-os másodpercenkénti kiolvasási igény is jelentkezhetne. Ezt a hatalmas mennyiséget minden bizonnyal csökkenteni kívánjuk valamilyen okos kezelési eljárással. Jó módszer, hogy egy új film csak minden egész percben kezdődhet. Ez még alig észrevehető a felhasználónál, de így a multicast sokat segíthet. Most a merevlemezek méretét és számát, az átviteli sebességet és a PCI csatoló legnagyobb teljesítőképességét kell meghatároznunk (a PCI csatolón az adatok kétszer is áthaladnak, egyszer a meghajtóról a gépre, majd egyszer a gépről a hálózati kapcsolat felé). Az biztos, hogy ekkora merevlemezt sehol nem fogunk találni, tehát valahogy meg kell osztanunk több lemez között a tárolandó adatmennyiséget. Azt is tudjuk, hogy a felhasználók bosszankodnak majd, ha a műsor a felénél megszakad, így mindenképpen fel kell készülnünk a lemezhibák kivédésére.

Hogyan érhetjük el az adatokat a leghatékonyabban a RAID 0 megoldásnál, ahol két lemez ugyanazt az adatot tartalmazza? Minden lemezvezérlőt a gyártó tulajdonában álló vezérlőprogram irányít, mely a lemezek közti adatforgalmat hangolja össze. Ez egy rendkívül összetett kérdés, melyre nem is igazán létezik tökéletes megoldás. Ha a lemezvezérlő két fájlt olvas be, akkor az egyiket olvashatjuk az első, a másikat pedig a második lemezről. De mi van akkor, ha eközben beérkezik egy harmadik fájlra irányuló kérelem? Azt se felejtsük el, hogy a merevlemezek az rákereséseket (seek) hajtják végre a leglassabban.

A rendszer összehangolásához tehát mindenképpen szükség van egy okos programra. Azt is megtehetnénk, hogy a népszerű fájlokat minden lemezre fölvesszük, és a ritkábban keresett fájlokat két másik lemezről olvassuk be. Ez a módszer igen hatékonyan segíthet a terhelés csökkentésében.

Minél többet tudunk a lemezek és a vezérlők működéséről, annál jobban összehangolhatjuk a rendszer működését. Ismét megjegyezném, hogy nem szükséges a tároló- és kiolvasórendszer minden részletével tisztában lennünk, de minél többet tudunk, annál hatékonyabban használhatjuk ki meglévő lehetőségeinket.

A rendszer és az Internet közti utolsó kapocs a hálózati kártya (vagy kártyák). Úgy építsük fel a környezetet, hogy a tartalomnak a lehető legrövidebb utat kelljen megtennie az Internetig. Gondolkozzunk el a hálózat és az egyes kártyák sávszélességén is, és ne felejtsük el: az, hogy egy gépbe több hálózati kártyát teszünk, még nem biztos, hogy megoldást jelent adatátviteli nehézségeinkre.

A hirdetésekben megjelenő elméleti teljesítményt kevés eszköz éri el a gyakorlatban. A rendszerhez szükséges alkatrészek becslésekor forduljunk szakértőkhöz, vagy látogassunk el az erre szakosodott honlapokra. Egy hatalmas rendszer vásárlása előtt építsünk egy kisebb mintadarabot és kísérletezzük ki, hogy hol várható teljesítménycsökkenés.

Az adatok továbbítása

Az Apple forgalmaz egy DARWIN Streaming Server nevű műsorszóró kiszolgálót, melynek nyílt forrású változata is elérhető, ehhez a forrásokban közölt helyre kell ellátogatnunk. A DARWIN Streaming Server „QuickTime Hinted" típusú fájlokat képes áramoltatni. Ez a formátum az Apple tulajdonában van, de mivel a kiszolgáló nyílt forrású, így a további formátumok támogatása egy kis módosítással megoldható. A Linux jelenleg nem támogatja a QuickTime formátumot az ügyféloldalon. A Lucent Technologies nemrég jelentette be, hogy Linuxra megjelent az OptiMedia MediaServe nevű műsorszóró kiszolgáló (lásd a Kapcsolódó címeknél) ingyenes változata. Állításuk szerint ez az ipari szabványnak számító Real-Time Streaming Protocolt (RTSP) használja, a számos formátum támogatása pedig több ügyfélfelületen is elérhetővé teszi.

A többi általam ismert megoldás zárt forrású. Néhány ezek közül (Entera, RealNetworks) Linux alatt is működik. Az Entera TeraCAST és TeraEDGE áramló kiszolgálók a szabványos RTP/RTCP protokollokat használják az RTSP-vel karöltve. A RealNetworks saját RDP protokollal is rendelkezik, mely az RTSP-t használva tart kapcsolatot a RealPlayer ügyfélprogramokkal. Ezzel szemben a Microsoft (még?) nem jelentette meg saját megoldásának linuxos változatát.

A szolgáltatás minőségének biztosítása

Nézzünk meg most néhány eljárást, melyeket mindenki rövidítéseikből ismer.

- QoS (Quality of Service) – A szolgáltatás minősége internetes szakkifejezés, mely minden valós idejű, tartalomszolgáltatás jelegű szolgáltatás minőségére, megbízhatóságára vonatkozik. Számos, műsorszórással foglalkozó cég e minőségi mutatókra hivatkozva tartja magát a legjobbnak. Legyünk óvatosak. A felhasználók számára nyújtott minőséget mindig a leggyengébb láncszem határozza meg.
- IP (Internet Protokoll) – Ez az Internet legáltalánosabban használt protokollja. A Dod Standard 1990-es közlése alapján az IP „teszi lehetővé az adatcsomagok átvitelét egyik gépről a másikra". Az Internet minden más protokollja az IP-n alapul.

- UDP (Unreliable Datagram Protocol) – Ez egy IP-re épülő protokoll, melyben nem biztos, hogy a csomagok megérkeznek a célig. Hasznos az RTP használata esetén is.
- RTP (Real-Time Protocol) – Valós idejű, az UDP-re épülő protokoll, melyet eredetileg a multicast típusú műsortovábbításra terveztek, de másra is használható.
- TCP (Transmission Control Protocol) – Arra tervezték, hogy megbízható, hibamentes adattovábbítást lehessen megvalósítani vele. Az RTP-vel azonos szinten működik, de ezt főleg időfüggetlen internetes kapcsolat során használják, vagy ha fontos a minél kisebb adatvesztés.
- RTSP – Alkalmazásszintű protokoll, mely az RTP alacsony szintű elemeinek használatával több folyamból álló műsortovábbítást tesz lehetővé. Ezek a folyamok több forrásból is származhatnak, melyek akár egymástól távol is elhelyezkedhetnek.
- RTCP (Real Time Control Protocol) – Itt a csomagok TCP-kapcsolaton keresztül mozognak, a minőségre RTP csomagok figyelnek.
- RSVP (Resource Reseration Protocol) – Ez szorosan kapcsolódik a QoS-hez. Az RVSP-t úgy alakították ki, hogy önálló folyamatként fusson, melytől az alkalmazások különböző minőségi szinteket kérhetnek. A kért adat útvonalán elhelyezkedő egységek elemzik a kérelmet, eldöntik, hogy a fogadó rendelkezik-e a megfelelő jogosultságokkal, majd a kért szint elérhetőségétől függően beleegyező vagy visszautasító választ adnak.

A fentiek lényege, hogy az RVSP egy nagyon okos protokoll, melylyel a gyártók nagyon egyszerűen növelhetik a szolgáltatás minőségét. Mielőtt kiválasztjuk a műsorszóráshoz használni kívánt programokat, érdeklődjünk a gyártónál, hogy csomagjuk hogyan kezeli a QoS-sel kapcsolatos gondokat, különös tekintettel az RSVP protokoll használatára.

Visszafejtés és lejátszás

Ezek már a felhasználó otthoni gépén történnek, de röviden említést teszek róluk, főleg azzal kapcsolatban, hogy mennyiben érintik a kiszolgálói oldalt. A szabványos internetprotokollok jellege miatt elvileg nem lenne szükség arra, hogy a kiszolgálóoldal kiépítésénél az ügyféloldalt is figyelembe vegyük. Sajnos, egy törvénytelen monopólium ereje elsöpörheti a nyílt szabványokat és a saját tulajdonában lévő módszereket erőltetheti a piacra. A nyílt forrású mozgalom erősödésével ezt a fajta „kézi vezérlést” egyre nehezebb lesz véghezvinni. Aki zárt forrású, fizetős formátumokkal szeretné a műsorszolgáltatását kialakítani, az a kiszolgálóoldalon jóval kevesebb lehetőség közül választhat. Üzleti döntésünk meghozatala előtt mindenképpen gondolkozzunk el azon: ha a szabadság szellemiségét nem is vesszük figyelembe, akkor is hátrányosabb a zárt szabványok használata, hiszen ezek megölik a szabad versenyt.

Kapcsolódó címek

APSL *(1. kép)*
➲ http://www.publicsource.apple.com/apsl/

OptiMedia MediaServe, Lucent Technologies *(2. kép)*
➲ http://www.lucent-optical.com/OAN/products/mediaserve.html

Entera
➲ http://www.entera.com

RealNetworks *(3. kép)*
➲ http://www.realnetworks.com

Streaming Media, Inc. *(4. kép)*
➲ http://www.streamingmedia.com

Összegzés

A műsortartalom iránti igény hatalmas piacot hozott létre az iparághoz egy kicsit is kötődő gép- és programgyártó cégek számára. Az iparág egyik legjobb adatforrása a Streaming Media, Inc.; ők magukat „az áramló multimédia otthonának” tartják. Akárhogy is döntünk, fogadjunk meg egy tanácsot. Egy néhány hétig tartó, az ajánlatok közötti keresgéléssel rengeteg pénzt takaríthatunk meg a későbbiekben.
Ha a Linux használata mellett döntöttünk, akkor jól vizsgáljuk meg, hogy az adott kereskedő hogyan viszonyul a nyílt forrású, nyílt szabványokhoz, a felhasználási szerződéséhez, és természetesen lehetőleg felkészült csapatot válasszunk. Vigyázzunk a „minden egyben” típusú termékeket forgalmazó cégekkel is: ha a „csodaszer” nem ad megoldást minden gondunkra, akkor máris felesleges kiadásokba bocsátkoztunk.

Frank LaMonica
az austini Texasi Egyetemen szerzett mérnöki diplomát, 18 éve dolgozik a számítógépes grafikai iparban. Dolgozott már Unixszal, Linuxszal és Windows NT-vel is. Frank volt a Precision Insight elnöke, azé a vállalaté, melynek az XFree86 4.0 részét képező és a nyílt forrású, gyorsított 3D grafika Linuxon való elterjedését elősegítő DRI közvetlen leképezési rendszert köszönhetjük. Frank ma a MultiMedia for VA Linux taktikai igazgatója, ez a cég vásárolta fel 2000. áprilisában a Precision Insightot. A nyílt forrású műsorszóró rendszerek melletti küzdelmen kívül jut ideje zenei pályájára is: klasszikus gitárkoncerteket ad.

Miért jó, ha proxyt használunk az otthoni internetezéshez?

Mivel a telefonhálózaton történő internetezés lassú, ezért mindenki szeretne megfelelő megoldást találni a kapcsolat gyorsítására. Erre használhatjuk a Squid vagy bármelyik proxykiszolgálót, a valós hálózati sebesség nem lesz ugyan gyorsabb, de ha többször is visszatérünk egy oldalhoz (ami bizony előfordul), akkor mindenképpen érezhető sebességnövekedést tapasztalhatunk. Ilyenkor a böngészőnk a proxykiszolgálótól kapja az adatokat, és csak az oldalon megváltozott elemeket? kell letöltenie. Gondoljuk csak át, mekkora sebességnövekedés érhető el, ha a honlapokon lévő képeket a saját számítógépünkön tároljuk, és nem kell minden egyes kapcsolatkor letölteni őket. A szerkesztőségben például a Linuxvilág kezdőoldalának letöltésénél mért idő, 64 k-s béreltvonalon:
➜ teljes letöltéskor 22 mp,
➜ a Netscape böngésző gyorstárazásával 10 mp,
➜ a Squid proxykiszolgáló használatával pedig 3 mp.
Ezekből az adatokból is látható, hogy az időnyereség nem elhanyagolható.
Nem minden számítógépen érdemes azonban használni ezt a megoldást, mert a proxykiszolgálók egyik tulajdonsága a nagyfokú memóriahasználat, így akinek kevés RAM áll rendelkezésére (32 MB) annak nem éri meg a használata.
A beállításokat a /etc/squid.conf fájlban tehetjük meg. (Lásd még a 74. oldalon lévő Squid proxykiszolgáló telepítése Linux alá című cikkünket.)

➲ http://www.squid-cache.org

A háromrétegű szerkezet (2. rész)

Szerzőnk további tanácsokat ad a háromrétegű felépítés használatához.

A múlt hónapban megkezdtük az ismerkedést a webes alkalmazásainkhoz illeszkedő háromrétegű felépítéssel. Egy középső objektumréteg segítségével különválasztjuk az adatbázis-kiszolgálót magától a webalkalmazástól, ezzel jelentősen leegyszerűsíthetjük az alkalmazás ügymenetét. Ezenkívül az alkalmazás és az adatbázis közötti réteg lehetővé teszi számunkra, hogy ugyanezt a középső réteget nem webes alkalmazásainkban is felhasználhassuk, sőt, az adatbázist anélkül módosíthatjuk, hogy ennek bármiféle hatása lenne az alkalmazásra.
Múlt havi írásom végére felépítettünk egy egyszerű középső réteget, mely képes kapcsolatot tartani a PostgreSQL adatbázisban létrehozott emberek és találkozók táblával. Most az ezen objektumok felhasználásával elkészíthető webes alkalmazásokkal ismerkedünk meg. Látni fogjuk, hogy a webes réteg soha nem éri el közvetlenül az adatbázist.

A webes alkalmazásréteg

Egy tökéletes világban a webes alkalmazásréteget bármely nyelven elkészíthetnénk, és a középső réteggel egy általánosan elfogadott protokoll biztosítaná a kapcsolattartást. A világ azonban nem elég fejlett ehhez, így a webes alkalmazásréteg kiválasztásakor a használt objektumréteg lesz a döntő.
Objektumainkat Perlben hoztuk létre, így e nyelv használatával kell kialakítanunk az alkalmazásréteget is. A CGI-programokkal kapcsolatos nehézségek elkerüléséért – mivel jelen esetben az Apache mod_perl-je sem hozna elegendő teljesítménytöbbletet – a már ismertetett Perl-alapú sablon- és alkalmazásfejlesztő környezetet, a Masont (Linuxvilág 2000. november, 59. oldal) fogjuk használni.
A Mason-elemek szükséges esetben a Perl-eljárásokba fordítódnak, ezek pedig újabb fordítással Perl opkódokká alakulnak. Ezeket az Apache mod_perl modulja gyorstárazza, és így sokkal gyorsabban elindíthatók, mintha CGI-t használnánk.

Személy hozzáadása

Első webalkalmazásos példánkban az adatbázist egy új személy bejegyzésével bővítjük. Ehhez két Mason-elemre van szükségünk: az egyik egy HTML kérdőív (ez lehet statikus is), a másik pedig magát a műveletet végzi el. A feladathoz a középső réteg emberek objektumát használjuk, mely az adatbázishoz kapcsolódva megkísérli új sor létrehozását. E két elem egyszerű változatát a *listán* is láthatjuk. Az összes lista túl sok helyet foglalna el, így azokat az ➲ ftp://ftp.ssc.com/pub/lj/listings/isue82 címről tölthetjük le.
A HTML kérdőív (add-person-form.html) a név–érték párjait az add-person.html-nek adja át, mely egy emberek nevű példányt készít belőlük, majd a new_person eljárás meghívásával új személy bejegyzését hozza létre:

```
my $success = $people->new_person
       (first_name =>$first_name,
       last_name => $las_name,
       country => $country,
       email => $email);
```

Ha $success igaz, akkor egy új személy került az adatbázisba, a $people->new_person-nak átadott értékekkel. Egyébként a meghívás sikertelen volt.
Ez azonban túl egyszerű módszer annak eldöntésére, hogy a művelet lezajlott-e vagy sem: a felhasználók számára nem egy igen/nem típusú visszajelzést célszerű adnunk. Sokkal ötletesebb megoldás, ha a látogatóval közöljük, hol rontotta el, és ennek megfelelően kijavíthatja a hibát. Ha az adatbázis belső hibája ugyanazt a hibaüzenetet juttatja el a felhasználóhoz, mintha egy újabb személy beillesztését kísérelte volna meg egy már használt levélcímre, akkor ezt nagyon nehéz kiküszöbölni. Ezért webes alkalmazásunk a bevitt adatokat még a középső réteghez való továbbítás előtt ellenőrzi. Minél több ellenőrző eljárást helyezünk el itt, és a rendszer minél több hibaüzenet megjelenítésére képes, annál jobb.
Az add-person.html összetevőnk két egyszerű ellenőrzést végez el. A Mason <%args> szakasza használatával minden adatot kötelező megadni. Ha egy kérdőív az adatait az add-person.html felé továbbítja, azok addig nem kerülnek feldolgozásra, míg a felhasználó az összes adatot be nem írja. Az értékeiket az add-person.html számára átadó HTML kérdőíveknek az összes elemet továbbítaniuk kell, különben a Mason megtagadja a kérelem teljesítését, és hibaüzenetet jelenít meg. Ezt nem az adatokat hiányosan beíró felhasználók fogják látni, hanem nálunk jelenik meg akkor, ha valamelyik <input> tagot kihagyjuk.
Amint a Mason-elemek működnek, biztosak lehetünk abban, hogy megkaptuk a megfelelő név–érték párokat. De vajon érvényes adatokat tartalmaznak-e? Az add-person.html fájl legelején lévő parancs segítségével ellenőriztük, hogy a $people->new_person meghívásához használt négy érték nem üres. Ha ezek bármelyike hiányozna, a látogatót felkérjük az adat begépelésére.
Azt is ellenőriztük, hogy a levélcímek érvényesek-e. A *listán* látható kifejezés nem illeszkedik minden levélcímre, de ehhez a példához megfelel. A szabálytalan levélcímmel próbálkozó felhasználók hibajelzést kapnak, ebből megtudhatják, hogy mit kell módosítaniuk.
Miután biztosak lettünk abban, hogy a kapott értékek értelmezhetők, meghívhatjuk a $people->new_person-t. Figyeljük meg, hogy az add-person.html mindezt anélkül teszi, hogy az adatbázissal közvetlenül kapcsolatot tartana. A DBI kétségtelenül nagy részt vállal a $people->new_person minden egyes meghívásában, de mindez a színfalak mögött történik, és Mason-elemeinknek nem kell foglalkozniuk ezekkel. Ha a people objektumot kijavítottuk, akkor biztos, hogy nem fogunk SQL-hibákba ütközni.

Az adatok változtatása

Miután áttekintettük, hogy miként illeszthetünk új személyt az adatbázisba a people objektum felhasználásával, próbálkozzunk valami bonyolultabbal: az update_first_name eljárással módosítsuk egy személy keresztnevét (a példák az ➲ ftp://ftp.ssc.com/pub/lj/listings/issue82/ címen érhetőek el, a fontosabb részeket a *3.* és *4. listában* találjuk). Ezt az eljárást csak akkor hívhatjuk meg, ha már kiválasztottuk a személyt, tehát a kérdőívünknek ezt lehetővé kell tennie.
Bár csábító lehetőség, hogy a személyek kiválasztását a név vagy a levélcím begépelésével oldjuk meg, ennek ellenére e módszert – a félre-

Lista add-person-form.html

```
<!-- -*- mmm-classes: mason -*- -->
<HTML>

<Head><Title>Add a new person</Title></Head>

<Body>
<H1>Add a new person</H1>

<Form method="POST" action="add-person.html">

<table>

  <tr>
    <td>First name</td>
    <td><input type="text" name="first_name"></td>
  </tr>

  <tr>
    <td>Last name</td>
    <td><input type="text" name="last_name"></td>
  </tr>

  <tr>
    <td>Address</td>
    <td><input type="text" name="address1"></td>
  </tr>

  <tr>
    <td>Address (line 2)</td>
    <td><input type="text" name="address2"></td>
  </tr>

  <tr>
    <td>E-mail address</td>
    <td><input type="text" name="email"></td>
  </tr>

  <tr>
    <td>City</td>
    <td><input type="text" name="city"></td>
  </tr>

  <tr>
    <td>State</td>
    <td><input type="text" name="state"></td>
  </tr>

  <tr>
    <td>Postal code</td>
    <td><input type="text" name="postal_code"></td>
  </tr>

  <tr>
    <td>Country</td>
    <td><input type="text" name="country"></td>
  </tr>

  <tr>
    <td>Comments</td>
    <td><input type="text" name="comments"></td>
  </tr>

</table>

    <input type="submit" value="Add this person">

</Form>
</Body>

</HTML>
```

gépelésekből adódó hibák elkerülése végett – nem ajánlom. A kiválasztást egy <select> lista segítségével végezzük inkább el, így kizárható annak lehetősége, hogy egy felhasználó olyan személy levélcímét, vagy bármely más adatát írja be, aki nem is szerepel az adatbázisban. A módosítani kívánt keresztnévhez tartozó személyt mindenképpen valamilyen egyedi adat segítségével kell azonosítanunk, de máris megjegyzem, hogy kicsit személytelennek tűnik, ha egyszerűen egy levélcímmel kell dolgoznunk. Megoldásként azt találtam ki, hogy az emberek objektumában (People.pm) új eljárást hozok létre (get_names_and_addresses, lásd az *5. listát* a már említett FTP-címen). Ez egy tömbhivatkozásokból álló listával tér vissza, melynek minden eleme névből és levélcímből áll. Az előbbit egyedi azonosítóként (az <option> tagban a „value” kapcsoló értékeként), az utóbbit pedig a megjelenítéshez használhatjuk. Átnézzük a levélcímeket és egy, az alábbihoz hasonló <select> listát készítünk belőlük:

```
<select name="email">
% # ftfutunk a neveken és az e-mail címeken és
kiírjuk azokat.
% foreach my $info (@names_and_addresses) {
  <option value="<% $info->[1] %>"><% $info->[0]
%>
% }
</select>
```

A felhasználók a többi személyi adatot is hasonló módon szerkeszthetik. Ha a személy azonosítására szolgáló adat valóban egyedi, akkor a személyhez tartozó bármelyik adatot egy hasonló kérdőívvel módosíthatjuk.

Találkozó hozzáadása

Miután áttekintettük, hogy miként használhatjuk a people objektumot az adatbázis tábláinak közvetett módosítására, térjünk rá a találkozók kezelésére, ezeket az appointments elem végzi. Ez az objektum lehetővé teszi, hogy adott személlyel, adott időpontban esedékes találkozót bejegyezzünk.
Ehhez ismét két elemre lesz szükségünk. Az első egy HTML kérdőívet készít, (add-appointment.html, a *6. lista* a már említett FTP-címen) ennek segítségével a felhasználók új találkozót jegyezhetnek be a rendszerbe. A személy kiválasztása egy <select> listáról történik. (Megjegyzés: ha ez „éles” feladat lenne, akkor a <select> listát külön egységként valósítanám meg, és más egységre bíznám, hogy a címjegyzékben lévő adatokat felkínálja.) Emellett tudnunk kell a találkozó kezdési és befejezési időpontját. Ismét a listából választást javasolnám, hiszen így elkerülhetjük a különböző dátumalakokból eredő félreértéseket.
Az itt látható Mason-kód a hónap, nap és év megadásához szükséges három <select> listát állítja elő. A @months és a @years előre meghatározásával a kódot olvashatóbbá tehetjük, és a rendszert könnyen felkészíthetjük a későbbi változtatásokra:

```
<select name="begin_month">
% foreach my $month (@months) {
        <option value="<% $month %>"><% $month %>
% }
</select>
<select name="begin_day">
% foreach my $day (1 .. 31) {
        <option value="<% $day %>"><% $day %>
% }
</select>
<select name="begin_year">
% foreach my $year (@years) {
        <option value="<% $year %>"><% $year %>
% }
</select>
```

A második összetevő (add-appointments.html; a *7. lista*) lehetővé teszi, hogy új bejegyzéssel bővítsük a találkozók naptárát. Egy <%args> szakaszban ellenőrzi, hogy az add-appointment-form.html-ből minden szükséges név–érték párt továbbítottunk, ezt követően pedig ugyanazokat az alapvető ellenőrzéseket hajtjuk végre, melyeket a többi elem esetében már bemutattunk.

Ez most tényleg három réteg?

Az eddigiekben láthattuk, hogy milyen egyszerűen készíthetünk háromrétegű webalkalmazásokat. Jogos a kérdés: a bemutatott példa valóban háromrétegű?

E szerkezet a régebbi modell, az „ügyfél–kiszolgáló” felépítés hiányosságainak hatására jött létre. A jelenlegi adatbázisok és webkiszolgálók tökéletes példái az ügyfél–kiszolgáló szerkezetnek. Ahogy ez a kifejezés két számítógépre utal, a háromrétegű szerkezet három különálló számítógépet jelent, tehát minden rétege egy-egy gépen található. Elmondhatjuk, hogy a példában vizsgált háromrétegű alkalmazást valóban három réteg alkotja, ezek önálló feladatokat végeztek, és lehetővé tették, hogy az alkalmazás és az adatbázis egy „középső réteg” közvetítésével tartson kapcsolatot egymással. Azonban az alkalmazás- és a középső réteg ugyanazon a gépen fut, és ezek elkülönítésére nincs is igazán lehetőség. Ha a webalkalmazásra nagyon sok kérelem kiszolgálásának feladata hárul, akkor egyszerűen egy vagy több azonos felépítésű Apache-ot futtató gépet kell illesztenünk a rendszerbe, de az alkalmazás- és a középső réteg objektumait nem helyezhetjük el külön gépeken.

Reményeim szerint sikerült bemutatnom a háromrétegű szerkezet előnyeit a szabványos API-kat igénylő programozó szemszögéből. A vázolt példa azonban nem tekinthető az elmélet tökéletes megvalósításának. Ehhez ugyanis távoli eljáráshívásokra (RPC) lenne szükségünk, melyek segítségével az egyik számítógép a másik gép valamelyik programelemét indíthatja el. Az RPC sokkal egyszerűbben használható, mióta megjelent a SOAP (Single Object Access Protocol, azaz egyszerű objektumelérési protokoll), de ez újabb gondokat szült, ráadásul újabb adatátviteli protokollal is meg kell barátkoznunk.

Míg a rétegek értelmezéséről elmélkedünk, talán figyelembe kellene vennünk azt is, hogy az itt bemutatott alkalmazás valóban három réteget használ, csak ebben a megfogalmazásban máshogy határozzuk meg a „réteg” fogalmát. Az „adatbázis, középső réteg, webkiszolgáló” helyett az „adatbázis, böngésző, webkiszolgáló” hármast is lehetne három rétegnek tekinteni – csak az a baj, hogy ezt bárki elmondhatja magáról, aki valaha írt már adatbázissal kapcsolatot tartó CGI-programot.

Sőt, ha már itt tartunk, újabb rétegeket is játékba lehetne hozni, csak hogy tovább bonyolódjon a helyzet. Mi van például az adatbázis tárolt eljárásaival, triggereivel és nézeteivel? Bár a tárolt eljárások nem tekinthetők fizikai rétegeknek, azért jól jönnek, ha egy adatbázissal dolgozó objektum- vagy alkalmazásréteget fejlesztünk. Meg merem kockáztatni, hogy a tárolt eljárások még jobbak is a középső rétegnél, hiszen közvetlenül az adatbázisban futnak le, és előre le vannak fordítva, tehát általában gyorsak.

A kódot futtathatjuk a webböngészőn is, például a JavaScripttel. Általában minden ügyfelem figyelmét felhívom arra, hogy ne használja ezt a hibáktól hemzsegő, megbízhatatlan és egyáltalán nem biztonságos nyelvet, ennek ellenére a böngészőkön kizárólag a JavaScript segítségével futtathatunk könnyedén programokat.

Ha webalkalmazásunk kapcsolati adatbázist, tárolt eljárásokat, középső réteget, webalkalmazás-réteget és ügyféloldali JavaScriptet is használ, akkor hány rétegünk is van? Valószínűleg még mindig három, de az igazság az, hogy teljesen mindegy, hogy minek nevezzük ezeket. Végtére is a tökéletes megoldás mindig az, ha a feladatnak leginkább megfelelő rendszert választjuk, és a jövőbeli bővítésekre is felkészülünk a tervezéskor. Így a rendszer anélkül is tökéletesen működik, hogy a legfrissebb, legmenőbb elnevezésekkel kellene dobálóznunk.

A háromrétegű szerkezet gondjai

Egyszerű példánk áttekintése után hadd szóljak a háromrétegű szerkezettel kapcsolatos gondokról. Nem állítom, hogy a rendszer alapjaiban véve rossz, de tudnunk kell, hogy nem maga a tökély. A legtöbb megoldáshoz hasonlóan csupán bizonyos körülmények között válik be igazán. Sok esetben, ha a tervezést és a megvalósítást külön munkatársakra bízzuk, az sokat könnyíthet a feladat nehézségén – ezt a webalapú határidőnapló rendszer esetében is láthattuk. Egy ember is képes megírni a szükséges kódot, de ketten hamarabb végeznek, ehhez azonban állandó kapcsolat kell.

Mint bármely mérnöki terv esetében, itt is engedményeket kell tennünk. A háromrétegű szerkezetnél az idővel lesznek bajok, hiszen egy ilyen rendszer megtervezése sokkal több időt emészt fel.

Bár a munkamegosztás megkönnyíti az egyes részterületek elkülönítését és kipróbálását, a „főpróbát”, azaz a teljes rendszer ellenőrzését eléggé megnehezíti. Ha mindenki egyetlen nyilvánosságra hozott és általánosan elfogadott API-t használna, akkor a dolog sokkal egyszerűbb lenne. Az előírás és a megvalósítás között azonban mindig ott tátong a szakadék, és teljes próba során ez általában ki is derül. Minél több réteg szerepel a tervben, annál összetettebb feladat a kipróbálás és az ellenőrzés.

Végül az adatbázissal való kapcsolatot biztosító középső réteg létrehozása is igen nehéz feladat. Az SQL sem tökéletes nyelv, viszont kevés paranccsal nagyon sokféle lekérdezést végezhetünk el.

Az SQL eltávolítása a Mason elemei közül, valamint egy adott API használatának kényszerítése azt eredményezi, hogy a programozó az adatbázis lehetőségeinek csak egy részét lesz képes kihasználni. Egy-egy újabb lehetőség iránti igény felmerülésekor a középső réteget kell bővítenünk. A programozó számára óriási segítség, hogy egy HTML sablonba (például egy Mason-elembe) közvetlenül beleépítheti az adatbázis-lekérdezést. Ezt a szabadságot nagy hiba lenne megvonni tőlük.

Összegzés

A nagy és összetett webalkalmazásokat fejlesztő programozók egyre célszerűbbnek vélik a régi ügyfél–kiszolgáló rendszer leváltását a háromrétegű szerkezetre. Ez ugyanis gyakran nagyon megkönnyíti a rendszergazdák és a programozók életét. A legfontosabb azonban az, hogy mindig valós szükségleteinknek megfelelően válasszunk.

Reuven M. Lerner
(reuven@lerner.co.il) egy izraeli web- és internet-tanácsadó cég tulajdonosa és vezetője. A Kovácsműhely rovat honlapja a ➲ http://www.lerner.co.il/atf/ címen érhető el.

linuxvilág

A magyar Linux-barátok magazinja.

www.linuxvilag.hu

Kiskapu Kft., 1081 Budapest, Népszínház u. 29. Telefon: 477-0443, Fax: 303-1619

Linuxos nyomtatókiszolgálás

Múlt hónapban a Microsoft idétlen rendszerleíró adatbázisával foglalkoztam. De hát nem ez az egyetlen terület, ahol komoly összeférhetőségi gondokkal kell szembenézniük a Windows-felhasználóknak: ott van például a hálózati nyomtatás. A „Microsoft-módszer” lényege, hogy minden ügyfélnek mindent kell tudnia arról a nyomtatóról, amit használni szeretne. A nyomtatókiszolgáló szerepe mindössze annyi, hogy a nyomtatási feladatokat sorba rendezze, az adatok formázásával nem törődik. Mi történik tehát akkor, ha a nyomtatót ki kell cserélnünk, például mert egy újabbat szeretnénk telepíteni? Nos, ilyenkor az akár több száz gépből álló ügyfélhálózat minden egyes gépén módosítanunk kell a meghajtó beállításait. A Linuxban is megtehetjük ezt a hatalmas butaságot, ha mindenképpen erre vágyunk – ekkor a nyers nyomtatófájlokkal kell dolgoznunk, szűrők nélkül. Ha azonban a nyomtatókiszolgálóként működő linuxos gépek szabványos PostScript fájlokból nyomtatnak, akkor a hálózati nyomtatás gyerekjátékká válik. Egyszerűen azt mondjuk a WordPerfectnek és társainak, hogy egy PostScript-meghajtót használjanak. A windowsos ügyfeleknél ilyen esetekben használhatjuk például az Apple 1200 PostScript meghajtót. Innentől kezdve a Unix/Linux háló bármelyik nyomtatójára kiküldhetjük az adatokat, de a PostScript fájlokat akár ismerőseinknek is elküldhetjük, olvasás, nyomtatás vagy PDF fájllá alakítás céljából. Nem kell ezernyi nyomtatómeghajtóval zsonglőrködni minden egyes ügyfélen. Az egyszerűség megint diadalmaskodik a Microsoft felett, hiszen kinek hiányzik egy újabb (szép nagy) adag fejfájás...?

SafetyNet

Ez a héjprogram gondoskodik arról, hogy azok a szolgáltatások, amelyeket mindig futtatni szeretnénk, folyamatosan működjenek. Ha nem futnak, újraindítja őket és küld egy levelet. Nincs több töprengés, hogy az összes kívánt szolgáltatás fut-e vagy sem. Ha valamilyen okból nem lehet újraindítani a szolgáltatást, a levélben ez is megjelenik. Szükséges: Bourne shell, ajánlott a cron.
➲ http://unixpimps.org/safetynet/

phpopenmonitor

Arra van szükségünk, hogy gyakran és gyorsan vizsgáljuk meg a hálózat egyes gépein működő szolgáltatásokat? Ez a kis segédeszköz nem fog az nmap helyére lépni, azaz nem kereshetünk vele nyitott kapukat. Ha azonban csak azt szeretnénk, hogy a kapuk egy bizonyos

ismert csoportjáról kapjunk tájékoztatást, akkor ez a program megfelelő: a kapuk általunk előre összeállított listáján lévő bármelyik szolgáltatásról azonnal képes adatokat szolgáltatni. Gyors, egyszerűen beállítható, alapértelmezés szerint ötpercenként frissíti az adatokat (ezt akár módosíthatjuk is). Kell hozzá egy webkiszolgáló, a PHP4 és egy színes megjelenítésre képes böngésző.
➲ http://www.edomex.net/phpopenmonitor/

HTAdmin

Amennyiben van némi tapasztalatod, a .htpasswd adatállomány kezelése nem okozhat gondot. Csak használd a htpasswd-t. Ha elfelejtetted, hogy hova helyezted az adatállományt, a „locate” parancs segíteni fog. Amikor rajtad kívül valaki másnak kell kezelnie egy nagy listát, akkor nem biztos, hogy az a számára is olyan egyszerű lesz. Belefáradtál már abba, hogy a sokadik alkalommal is elmagyarázd a htpasswd-t az embereknek? Akkor telepítsd fel ezt a segédprogramot a webkiszolgálóra, adj nekik hozzáférési jogot (talán egy másik .htpasswd adatállomány segítségével) és kezelési lehetőséget. Biztosíts összeköttetést a munkaasztalukon és (talán) nem hallasz felőlük minden öt percben… legalábbis nem a htpasswd adatállományról.
Szükséges: PHP-t támogató webkiszolgáló
➲ http://www.bilcag.net/hdogan/php/htadmin.php3

pkgbuild

Akinek gyakran kell RMP csomagokat építenie, annak nagyszerű segítséget jelenthet ez a grafikus felületet használó kis eszköz. Természetesen a csomagok összeállításában megszerzett tudásunkat nem kell miatta elfelejtenünk, de sok terhet levesz a vállunkról. Ha megfelelő sablonnal indítunk (amit a pkgbuild is szeret), onnan már egyszerű lesz a dolgunk. Nem lesz belőlünk RPM-mágus, viszont gondolkodásra késztet, hogy vajon jól építjük-e fel spec fájljainkat. Használatához a libm, libSM, libICE, libXext, libX11 és glibc könyvtárakra lesz szükségünk.
➲ http://www.linuxsupportline.com/~davin/

SQL-Ledger

Ha valakit érdekel, hogy hogyan is néz ki egy amerikai számlázóprogram, nosza, megtalálta! Ez a Perl segítségével elkészített SQL-adatbázison dolgozó program sok érdekességet tartalmaz. Vannak hiányosságai (például a POS-támogatás), de ezeket igyekeznek a további változatokban megvalósítani. Egy ilyen program mellett kinek kell a Quick Books? Használatához a PostgreSQL-re, egy webkiszolgálóra, a Perlre és a DBD-Pg, DBI nevű Perlmodulokra pvan szükségünk.
➲ http://www.sql-ledger.com/

multiscan

Mindenki tudja, hogy az nmap milyen nagyszerű segédeszköz. Sajnos, azonban nem túl gyors, emellett ráadásul egy kicsit „túlsúlyos” is szegényke. Ha a saját hálózatunkon szeretnénk villámgyors ellenőrzést végezni (például hogy az egyes rendszereken milyen kapuk vannak nyitva), akkor azonnal tapadjunk rá a multiscanra. Saját szememmel láttam, ahogy szempillantásnyi idő alatt végigsöpört egy egész C osztályú hálózaton. Az elérhetetlen gépek természetesen jócskán lelassították, az elérhetők viszont hihetetlenül gyorsan, 2 gép/másodperces sebességgel vallottak szégyenkezés nélkül önmagunkról, nyitott kapuikról, egyéb viselt dolgaikról. Bámulatos. Csupán a glibc kell hozzá.
➲ http://sourceforge.net/projects/multiscan/

pam_watch

Vágytak már valaha arra, hogy bekukucskáljanak a terminálok „mögé”? Vagy mondjuk szeretnék megmutatni valakinek a világ másik tájáról, hogy egy adott műveletet hogyan lehet a legegyszerűbben végrehajtani? Ez a segédprogram lehetővé teszi két felhasználó számára, hogy földrajzi helyüktől függetlenül ugyanazt a terminált használhassák. Ha bejelentkezési kapcsolatként használjuk, a pam_watch két csővezetéket hoz létre, egyet a kimenetek, egyet a pedig a bemenetek számára, melyhez valaki (általában a rendszergazda) csatlakozhat. Az egyetlen hiányossága, hogy az X és ssh kapcsolatokban használt pty-kel, valamint a terminálból indított kapcsolatokkal nem működik. Futásához a libpam, a libdl és a glibc csomagokra van szükségünk.
➲ http://frida.fri.utc.sk/~behan/devel/pam_watch/

tvguide

A hang és a mozgókép nem az én szakterületem. Természetesen szívesen hallgatok, mondjuk egy kis Pink Floydot munka közben, – kipróbáltam, a Comfortably Numb mellett kitűnően tudok összpontosítani egy feladatra. De szeretek például híreket nézni, amíg egy-két fordítás zajlik a háttérben. A tvguide gyorsan letölti a honlapról a műsort, melyben kereshetek is, és azonnal a megfelelő csatornára kapcsolhatok a tévévevő kártyámmal. Lehet, hogy ezt sokan az erőforrások pazarlásának vélik, de a gépem úgyis gyakran lustálkodik, akkor meg miért ne. Legalább nem maradok le a focimeccseimről. Használatához a Perlt kell telepítenünk.
➲ http://www.cherrynebula.net/tvguide.html

PHP DB űrlaplétrehozó

Ezen segédprogram jó kezdetnek bizonyul a MySQL adatbázis egyszerű és szép beviteli űrlapjának létrehozásához. Az alapelgondolás logikusnak, az űrlapbeállítás és -létrehozás egészen egyszerűnek tűnik. Bár nem egészen nyilvánvaló, ennek ellenére mégis van lehetőség adattáblázatok összekapcsolására. Jelenleg azonban nem lehetséges egyszerre több nézet kezelése és a jelentések még mindig a TODO listában szerepelnek. Ez a segédprogram nagyon kezdetleges állapotban van még jelenleg is, így a sokkal magasabb szintű változást a fejlődés hozza meg.
Szükséges: PHP és MySQL támogatású webkiszolgáló, MySQL, és webböngésző.
➲ http://sourceforge.net

CCC

A CCC nem használható általános számlázóprogramként – bár kisebb módosításokkal alkalmas lenne a feladatra –, de egy számítógép-kereskedés üzleti és egyéb nyilvántartásának vezetésére tökéletesen megfelel. Figyelemmel kísérhető a feladatok, a szerelők, a rendszerek állapota, még az ügyfelek számláinak vezetésére is alkalmas. Aki mindig is egyszerűen kezelhető, munkabeosztás- és számlakezelő programra vágyott, annak érdemes kipróbálnia. A MySQL, egy PHP és MySQL támogatással rendelkező webkiszolgáló, és egy böngésző kell hozzá.
➲ http://www.noguska.com/ccc/

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