Hírek

Tényleg billentyűzet?

Az *AlphaGrip AG-5* sokkal inkább egy konzol botkormányára hasonlít, mintsem egy klasszikus billentyűzetre. Közelebbről azonban szembe találjuk magunkat rengeteg gombbal. A tervezők minden szükséges billentyűt elhelyeztek rajta, sőt egy *négyirányú kar* is helyet kapott, ami az egeret hivatott kiváltani. A prospektus szerint *50 szó per perc* is elérhető, igaz, ehhez *30-60 óra gyakorlás* is szükséges. A periféria *USB* porton csatlakozik a *PC*-hez. Az ára elég magas, egy dollár híján *100 dollár*.
➲ *http://www.extremetech.com/article2/0,1697,1949084,00.asp*

A MINIX grafikus felületet kap

Andrew Tannenbaum 1987-ben kezdte a *MINIX* fejlesztését. A korai változatokat (1.x és 2.x) csak oktatási célokra ajánlott, míg a jelenlegi (3.x) változatokat már akár éles helyzetben is használhatjuk. Hogy képes lesz-e felzárkózni a 1991-től fejlesztett *Linux*hoz, az kérdéses, de sokat lendíthet rajta a hír, miszerint a *MINIX* mintegy 20 év után végre kap *X*-et. (A *MINIX* jelenleg mintegy 400 programot tartalmaz.)
➲ *http://www.minix3.org/*

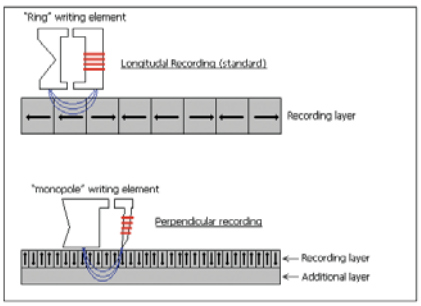
Hatvanmilliárd

Becslések szerint napi 60 milliárd emailt küldenek és fogadnak naponta a *Föld* lakói, ennek azonban nagy része kéretlen reklámlevél, vagyis spam.

Kritikus sebezhetőség a GPG-ben

Kritikus *sebezhetőséget* fedeztek fel a népszerű titkosító programban, a *GnuPG*-ben. A támadó számára a sebezhetőség lehetővé teszi, hogy extra információt juttasson az e-mailbe anélkül, hogy annak aláírt státusza megváltozna. Ez a sebezhetőség minden – *1.4.2.2-nél régebbi – GnuPG* esetén jelentkezik, tehát érdemes mihamarabb frissíteni.
➲ *http://secunia.com/advisories/19173/*

750 gigabájt



Nathan második törvénye szerint a szoftver – mint a gáz – kitölti a rendelkezésre álló teret. Természetesen a merevlemezgyártók sem tétlenkednek. A *Seagate* elkészítette az eddigi legnagyobb merevlemezt *750 gigabájtos* kapacitással, ezzel visszaszerezte a *Maxtortól (500 gigabájt)* az első helyet. Az új háttértár *merőlegesen* rögzíti a biteket, így nagyobb *sűrűséget*, *kapacitást* és *átviteli sebességet* érhet el a gyártó.

Linksys 802.11n accesspoint

A *Linksys* piacra dobta legújabb hozzáférési pontját (accesspoint), a *Linksys WRT300N*-t, amely nemcsak a *802.11a/b/g* szabványokat támogatja, de a *802.11n*-t is. Ez mintegy *tizenkétszer gyorsabb* adatátvitelt és kb. *négyszeres hatósugarat* biztosít. Ez a legelső termék, amely támogatja a *802.11n* szabványt. Természetesen az eddig megszokott dolgok sem maradtak ki: *WPA, DHCP szerver, VPN átjáró és tűzfal*. Az eszközt a már megszokott webes felületen konfigurálhatjuk.
A bolti ára 150 dollár, míg a *802.11n* szabványú *PCMCIA* kártya (*WPC300N*) ára 120 dollár. Azt egyelőre nem tudni, hogy vajon lesz-e a *PCMCIA* kártyához linuxos meghajtóprogram.
➲ *http://www.linuxdevices.com/news/NS2899697658.html*

Új felsőszintű tartomány?

Az *ICANN* egy új felsőszintű domain, a *.tel* bevezetését tervezi. Az ötlet szerint itt lehetne tárolni az adott tulajdonos elérhetőségeit, például a *www.linuxvilag.tel* esetén a *Linuxvilág*hoz tartozó telefonszámokat. Ez azonban magánszemélyeknél gondot jelentene, ugyanis pl. Kiss Jánosból rengeteg van. A megoldás például ez lenne: *www.1234.KissJanos.tel*, ennél viszont akkor már egyszerűbb egy on-line telefonkönyvben keresni a terv létjogosultságát megkérdőjelezők szerint.

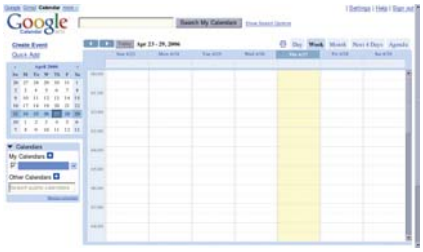
Hyperthread – Visszafele?

Az *AMD* egy új technikán dolgozik, amely lehetővé teszi, hogy a *kettő*, illetve *négymagos* processzorok *egymagosként* látszanak, szemben az *Intel Hypethread* megoldásával, ahol az egymagos processzor két magosnak látszik. Az *AMD* megoldásának nagyszerűsége abban rejlik, hogy nem kell a programozónak több processzorra kódolni, ugyanis a feladatok elosztását hardveresen végeznék.

Digitális sírkő

A holland *Henk Rozema* elkészítette az első digitális sírkövet, amely videókat, fotókat és hangot játszik le. A képi anyag megjelenítéséről egy 17 hüvelykes *LCD* képernyő gondoskodik. A „multimédia” sírkő csak akkor kapcsol be a mozgásérzékelőnek köszönhetően, ha arra jár valaki. Az energiaforrása napelem és beépített akkumulátor.

Google Calendar



Hosszú találgatás után a *Google* eszköztára egy újabb eszközzel bővült, a *Google Calendarral*. A naptár egységet alkot a *Gmail*-lel, illetve teljes *iCal* támogatással bír. Mint a legtöbb *Google* oldal, úgy ez is az *AJAX (Asynchronous Javascript and XML)* technológiát használja, vagyis kis adatforgalmat generál, tehát kis sávszélesség esetén is kényelmesen használható.

➲ http://calendar.google.com/
➲ http://www.gmail.com/

Linuxos kotta

Érdekes házasság köttetett a *Linux* és a zene között. A *FreeHand Systems* elkészítette a *MusicPad Pro Plus*-t, amely nem más, mint egy elektronikus kotta. Kinézetre egy *PDA*-ra hasonlít, ám annál sokkal nagyobb a kijelzője, *12 hüvelykes*, szemben a *PDA*-k 3-4 hüvelykes *LCD*-ivel. A művészek beszkennelhetik a meglevő – papíralapú kottáikat –, vagy akár az internetről is letölthetnek *JPG* vagy *PDF* formátumúakat. A kották *álló* vagy *fekvő* formátumban jeleníthetőek meg az eszközön és a képernyő megérintésével vagy a mellékelt pedállal lapozhatunk. Az LCD *háttérvilágítással* is rendelkezik, így akár sötét helyen is használható. A beépített akkumulátor 4 óra folyamatos megjelenítést tesz lehetővé. A burkolat alatt egy *400 Mhz-es AMD Alchemy (MIPS)* processzor, *256* megabájt *RAM* és *128* megabájt *flash* tároló található. Az készülék és a kiegészítők ára elég borsos. A készülék *1200 dollárba*, a lapozáshoz használható pedál pedig 70 dollárba kerül.

➲ *http://www.linuxdevices.com/articles/AT9665830722.html*

Linux az Apple iPod-on

Bizonyára sok Olvasónak van *Apple iPod*-ja. Nos, jó hír számukra, hogy az *iPod* sem kerülhette el a végzetét. Fut rajta a *Linux* mégpedig duálbootosan, vagyis megmarad a régi operációs rendszer is. Erről bővebben az *iPodLinux* projekt oldalán olvashatunk.

➲ *http://ipodlinux.org/Main_Page*

Medve Zoltán
(e-medve@e-medve.hu)

2001-ben kezdett „Linuxolni”, de már korábban is ismerkedett a szabad szoftverek világával.
Ha éppen nem a gép előtt ül, akkor fotózgat, olvasgat vagy bicajozik.

Mi újság a rendszermag fejlesztése körül

A *Raw Driver*, amit korábban a blokkeszközökön végrehajtott nem pufferelt I/O műveletek végrehajtására használtak már jó ideje elavultnak számít, egészen pontosan azóta, amióta az *open()* rendszerhívás támogatja az *O_DIRECT* opciót, ami pontosan ugyanerre szolgál. *Adrian Bunk* be is harangozta a meghajtó eltávolítását a rendszermagból, de aztán az utolsó pillanatban mégis meggondolta magát. Amint az a felhasználókkal közvetlenül kapcsolatban álló kernelszolgáltatásokkal már csak lenni szokott, az embereknek ilyenkor alapvetően két választása van: vagy találnak egy másik olyan magszolgáltatást, ami kiváltja a korábbit, vagy soha többé nem frissítik a kerneljüket. Az ilyen dilemmáknak aztán általában az a vége, hogy az elavult szolgáltatás végül mégis marad, miközben a fejlesztők régóta törekszenek arra, hogy megtisztítsák az efféle dolgoktól a felhasználói teret. Ugyanez történt a *Raw Driver* esetében is. Kiderült, hogy egyelőre nagyon sok felhasználó támaszkodik még rá, bár a legtöbbjük komoly erőfeszítéseket tesz annak érdekében, hogy mielőbb lecserélje az elavult kódrészleteket az új *O_DIRECT*-en alapuló rendszerhívásokra. Tekintettel azonban arra, hogy itt egy valóban nagyon elterjedt szolgáltatásról van szó, valószínű, hogy a *Raw Driver* még sokáig része lesz a hivatalos kernelnek.

Adrian folytatja az *OSS* hangmeghajtók kernelből való eltávolítását is, de ezzel is lassan halad. Továbbra is körülbelül 50 ilyen meghajtó van a kernelfában. Közülük egyesek olyan hardverelemekhez íródtak annak idején, amelyeket az *ALSA* rendszer immár tökéletesen támogat, így teljes biztonsággal eltávolíthatók a magból. Vannak azonban csupán befejezetlen, vagy hibásan működő *ALSA* meghajtókkal rendelkező hardverelemek is, ahol így előbb a megfelelő javításokat kell elvégezni, sőt olyan is akad, amihez még mindig nincs *ALSA* támogatás. *Adrian* már jó ideje nagy türelemmel dolgozik ezen az ügyön. Megkeresi minden egyes meghajtó íróját, begyűjti a hibajavításokat, illetve tartja a kapcsolatot a felhasználókkal, akiktől az értékes hibajelentések, illetve a hiányzó *ALSA* meghajtókkal kapcsolatos bejelentések érkeznek. Összességében igyekszik gondoskodni róla, hogy a kernelből csak a valóban tökéletesen elavult hangmeghajtók kerüljenek ki, olyan semmiképpen, amire valakinek még szüksége lehet.

Michael Hanselman írt egy javítófoltot egy olyan régi *ATI RADEON framebuffer* meghajtóhoz, amit 2002 óta nem frissített senki. Bár a régi meghajtót a fejlesztők már régen megjelölték, a lecserélésére írt kód még mindig nem tökéletes. Amint arra *David S. Miller* rámutatott, a képernyőtörlés megvalósításában van egy olyan hiba, ami bizonyos helyzetekben összezavarhatja az *X Window* rendszert. Mivel azonban nem csak *David*-nek, hanem még az olyan öreg motorosoknak is tetszik a *Michael* által benyújtott javítás, mint amilyen *Benjamin Herrenschmidt*, a régi meghajtó most már hamarosan távozik a kernelfából. Ugyanakkor *Andrew Morton* megjegyezte az üggyel kapcsolatban, hogy amennyiben lehetséges „szeretne elkerülni minden a felhasználókkal kapcsolatos gikszert”, amikor a régi meghajtót valóban eltávolítják.

Jeff Garzik közzétette két, korábban titkosított *SATA* vezérlő hardver leírását. Konkrétan a *Silicon Image 3114*-es és *3124*-es lapkakészletekről van szó. A *Silicon Image* – feltehetőleg hosszas személyes tárgyalás után – végül engedélyezte *Jeffnek* a dokumentációk nyilvánosságra hozását. Ez az új műszaki információ feltehetőleg lökést ad majd az *NCQ (Native Command Queuing)* támogatásának is, amit a nagy sebességű adatátvitelnél használnak. A hardvergyártók efféle gesztusait a nyílt forrású közösség mindig igen nagyra értékelte. Ne felejtsük el, hogy ma is számos olyan hardverelem létezik, amihez semmiféle nyilvános dokumentáció nem áll rendelkezésre. Ilyenkor a *Linux* fejlesztői számára két lehetőség van: vagy komoly munka árán visszafejtik a kérdéses eszköz működésének részleteit, vagy egyszerűen nem támogatják a használatát. Bár a *Watchdog* meghajtókat már jó ideje *Wim Van Sevroeck* tartotta karban, csak a közelmúltban járul hozzá ahhoz, hogy ez a kapcsolat hivatalossá váljon, és a neve bekerüljön a *MAINTAINERS* fájlba. A dolog kiváltója egyébként az volt, hogy *Kumar Gala* a fejlesztői listán nemrég rákérdezett a *Watchdog* fenntartójának kilétére, így *Arnd Bergmann* javaslatára *Wim* végül hivatalosan is felvállalta ezt a szerepet.

A kernel konfigurálása mindig is meglehetősen összetett terület volt ahhoz, hogy időről időre a legkülönbözőbb ötletek merüljenek fel az egyszerűsítésével kapcsolatban. Nemrég *Randy Dunlap* azzal az ötlettel állt elő, hogy a *SATA* eszközökkel kapcsolatos beállításokat ki kellene venni a *SCSI* ágból. A jelenlegi elhelyezés azért logikus, mert a *SATA* meghajtók valóban támaszkodnak a *SCSI* alrendszer bizonyos függvényeire, ezeket azonban máshol is meg lehetne valósítani, a *SCSI*-tól teljesen függetlenül. *Randy* szerint az átlagos felhasználónak semmi szüksége arra, hogy tudatában legyen ennek a *Serial ATA* és *SCSI* technológiák támogatása között fennálló ezoterikus kapcsolatnak. És bár maga *Randy* nyilván tisztában van vele, így személyesen nem érdekelt abban, hogy a javaslata meghallgatásra találjon, a kernelfejlesztők többsége érdekes módon mégis támogatja az ötletet. Ha tehát *Randy* részletesen kidolgozza a tervet, a módosítás megvalósulhat.

Linux Journal 2006. 145. szám

Zack Brown

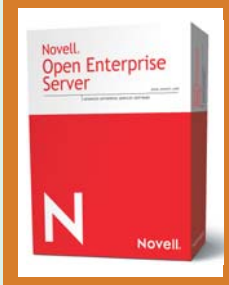
Új, költséghatékony munkacsoport megoldás a Novelltől

A Novell új munkacsoport megoldása, a Novell Open Workgroup Suite minden olyan szolgáltatást biztosít, ami egy korszerű irodai infrastruktúra kialakításához szükséges és a legfontosabb szerver oldali szoftverektől kezdve a teljes értékű irodai munkaállomásig az összes lényeges alkalmazást tartalmazza.

A fejlett hálózati és irodai szoftverek integrálásával létrejött *Novell Open Workgroup Suite* teljes körű integrált informatikai programcsomag, amely átfogó megoldást nyújt egy korszerű irodai szoftverkörnyezet kialakításához a kis-, közepes- és nagyvállalatok számára egyaránt.
A termék a legfontosabb szerver oldali szoftverektől kezdve az irodai munkaállomásig minden lényeges alkalmazást tartalmaz, többek között a *NetWare* és a *Linux* előnyeit ötvöző hálózati kiszolgálót, megbízható és biztonságos csoportmunka megoldást, hatékony rendszerfelügyeleti szolgáltatást, a legelterjedtebb irodai programcsomagok formátumait kezelő irodai alkalmazást és teljes értékű asztali operációs rendszert. A *Novell Open Workgroup Suite* kedvező árkonstrukciójának köszönhetően a vállalatok a legmodernebb technológiák használata mellett, kompromisszumok megkötése nélkül racionalizálhatják informatikai költségeiket, és hatékonyabban tervezhetik hosszú távú informatikai stratégiájukat is.

A Novell Open Workgroup Suite az alábbi termékeket tartalmazza: Szerver oldali szolgáltatások

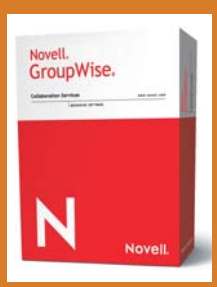
Novell Open Enterprise Server



A világ első vegyes forráskódú vállalati megoldása a tavalyi évben került bevezetésre, azóta a Novell több mint 8 millió *Open Enterprise Server* licencet szállított új, illetve meglévő ügyfeleinek. A *Novell Open Enterprise Server* biztosítja az ügyfelek számára a *NetWare* fejlett jogosultságkezelését, a megbízható fájl-, nyomtató- és címtárszerver szolgáltatásokat, az alkalmazásszerverként kiváló *SUSE Linux Enterprise Server*, pedig a nyílt forráskódú rendszerek rugalmasságát adja a termékhez, amely a felhasználók számára egy egységes felületet biztosít bármely operációs rendszer irányából.

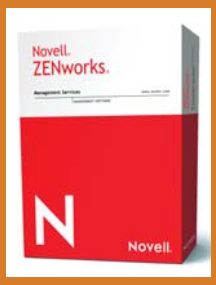
Novell GroupWise 7

A *GroupWise 7* a *Novell* vezető munkacsoportos és együttműködési szoftverének legújabb verziója.
A *GroupWise* leegyszerűsíti a kommunikációt: egyetlen helyen, az ún. *Univerzális Postafiókon* keresztül biztosít hozzáférést minden lényeges információhoz, beleértve az üzeneteket, határidőnapló-bejegyzéseket, dokumentumokat. A *GroupWise 7* támogatja a hagyományos- és webes klienseket, valamint a mobil eszközöket, ezzel biztosítva az elektronikus levélküldés,



azonnali üzenettovábbítás, feladatküldés és ütemezés lehetőségét, valamint a kapcsolatok és dokumentumok egyszerű elérhetőségét.

Novell ZENworks 7



A *Novell ZENworks 7* leegyszerűsíti az informatikai erőforrások telepítését, kezelését és karbantartását a ma használatos heterogén környezetekben. A *ZENworks* egy integrált többplatformos eszközkészletet nyújt, amely asztali, hordozható, szerver és kéziszámítógépek teljes életciklusán keresztül automatikussá teszi a felügyeletet. Ezáltal az ügyfelek és a rendszergazdák feladatai is csökkenhetnek, valamint az informatikai költségek terén is megtakarításokat érhetnek el. Ugyanakkor a felhasználók számára szükséges összes erőforrás – a személyazonossági beállításoknak megfelelően – időben jut el a megfelelő emberekhez.

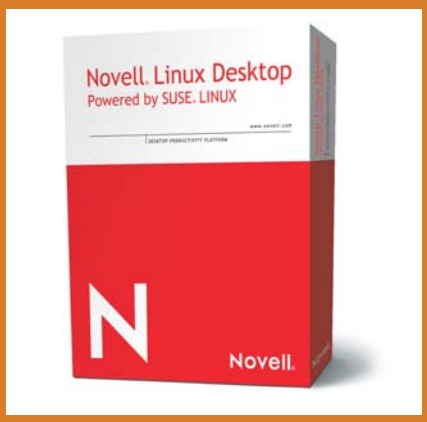
Programcsomagok irodai munkaállomásokhoz

OpenOffice for Windows

Az *OpenOffice.org* irodai alkalmazások gyűjteménye, mely táblázatkezelő, szövegszerkesztő, bemutatókészítő és rajzoló eszközökből áll. Az *OpenOffice.org Novell*-változata számos *Visual Basic* makrót kezel, és kiterjedt fájlformátum-kompatibilitást biztosít a *Microsoft Office* formátumaival, így a felhasználók hatékonyabban oszthatják meg fájljaikat a platformok között. Az alkalmazáscsomag felhasználói felülete magyar nyelvű és tartalmaz egy integrált, nyílt forráskódú, magyar nyelvű helyesírás-ellenőrző és elválasztó szótárt. Az *OpenOffice.org 2.0* teljes mértékben támogatja a dokumentumfájlok új, nyílt formátumát, az *OpenDocument* fájlformátumot.

Novell Linux Desktop, és legújabb verziója a SUSE Linux Enterprise Desktop

A *Novell Linux Desktop* termék egy hatékony végfelhasználói környezetet kínál, mely a vállalati



ügyfelek számára lehetővé teszi a *Linux* és a megbízható nyílt forráskódú megoldások biztonságos kihasználását. A *Novell Linux Desktop* azonnali alternatívát jelent a közigazgatástól kezdve az egészségügyi intézményeken és legnagyobb vállalatokon át a pénzintézetekig a különféle meghatározott funkciójú munkaállomásokon történő bevezetésre.
A *Novell Linux Desktop* tartalmazza a mindennapos üzleti feladatok elvégzéséhez szükséges összes fontos alkalmazást.
Azon ügyfelek, akik szoftverfrissítéssel együtt vásárolják meg az *Open Workgroup Suite*-ot, a *Novell Linux Desktop* következő verzióját, a *SUSE Linux Enterprise Desktop*-ot is megkapják a termék nyári bejelentése után. A *SUSE Linux Enterprise Desktop* olyan technológiai újításokat tartalmaz, mint a továbbfejlesztett energiagazdálkodás, a beépített asztali keresőrendszer, a nagy teljesítményű grafikus felületek és a különféle alkalmazások nyílt forráskódú fejlesztésekből származó újdonságai. Ennek köszönhetően a *SUSE Linux Enterprise Desktop* egyszerű használhatóságot, nagy teljesítményű, beépített irodai funkciókat és sziklaszilárd stabilitást kínál a termelékenység növelése mellett.

ELÉRHETŐSÉG

Az *Open Workgroup Suite* a *Novell* viszonteladói hálózatán keresztül 2006. július 30-ig bevezető áron kapható, akár 27.500 Ft felhasználónkénti áron is elérhető. A fenti ár ajánlott nettó végfelhasználói ár és az *Open Enterprise Server Linux* kernel alapú változatát tartalmazza.

KAPCSOLÓDÓ CÍM

További információ a
➲ http://www.novell.com/hungary/termekek/openworkgroupsuite/ weboldalon található.


Linuxvilág
www.linuxvilag.hu

Interjú a madárral (2. rész)

A nyár küszöbén és a Kisszikla peremén folytatódott interjúsorozatunk Pápaszemes Tóbiással, kedvenc pingvinhölgyünkkel.

H.L.: Igazán kedves Öntől, ***Tóbiás***, hogy hajlandó volt mikrofon elé állni, hisz úgy tudom, nemrég orvosi beavatkozáson esett át, megelőzendő a madárinfluenzát, és ennek okán kényszerpihenőre is küldték, illetve zárták. Khmm… izé… ha nem vagyok túl indiszkrét, hová bökték azt az injekciós tűt?

P.T.: Beszéljünk inkább másról – felelé a madár fájdalmas tekintettel, tapogatva testén egy bizonyos pontot. – Éppen most jutott eszembe, ma 3 éve, hogy itt, az Állatkertben két hónapos aláírásgyűjtő kampány indult az állatkínzás szabálysértésből bűncselekménnyé súlyosbítására… Maga aláírta?

H.L.: Természetesen. (E válaszomra mintha kissé szelídebb pillantásokkal mért volna végig, és abbahagyta masszírozását testén ama pontnak.) Láttam, hogy a környéken nagy a mozgolódás, újítják a ***Nagysziklát***, költöztetik az állatokat, a szárnyasok új röpdéket kaptak, egyesek ellenek, mások másállapotokba kerülnek, s rögvest anyáknapi koncerteket is adnak. Nem zavarja ez az Önök nyugalmát?

P.T.: Cseppet sem. Néha nem árt egy kis szórakozás. A medence partján napozgatva impozáns látvány, amint mindenféle ormányosmedvéket, pandákat, hiénákat, leopárdokat tologatnak erre-arra. Unalmas folyton perecdobáló embereket nézegetni. Így legalább felköszönthettük ***Johnnyt***, a rétisast, aki idén lett 36 éves. El is meséltük neki, az a hír járja, hogy a debreceni múzeumban kitömve is látogathatják rokonait.

H.L.: Azt olvastam valahol, hogy régmúlt időkben a pingvinek is képesek voltak repülni.

P.T.: Visszautasítom a múlt időt; igenis remekül szárnyalunk – csak éppen a víz alatt. Továbbá amit rosszindulatúan terjesztenek alkatunkról, tudniillik, hogy gömbölydedebbek lennénk itt-ott az átlagnál, az is csupán erőteljes mellizomzatunk miatt van – düllesztette ki erőteljes mellizomzatát. – Köztudottan évente több tízezer repkedő szárnyas sül halálra elektromos vezetékeken, mi óvjuk életünket az áramütéstől.

H.L.: Értem. És mit szól ***Ingrid*** és ***Juszuf*** újdonsült gyermekéhez, a kis ***Íriszhez***? Úgy hallottam, üknagyanyja még az ***Egri csillagok*** című nagy sikerű filmben is szerepet kapott.

P.T.: Ezek a kétpúpúak sokra viszik még! Egy ilyen tevegyerek közel 40 kilósan születik, pár hónap elteltével már egy mázsát nyom. Nem csoda, hogy a filmiparba is erőteljesen nyomulnak.

Madarunk is erőteljesebb nyomást érezhetett gyomra tájékán, mert szokásához híven szó nélkül faképnél hagyott, pontosabban egy fa szélénél, és erőteljes mellizomzatával a kerti medencébe vetette magát, hogy a túlparton az elsők között érhessen a heringesvödörhöz.

Halusz Léna

Nincs menedék?

Az ember alapvetően szabadnak születik. Aztán kap személyi számot, adószámot, napszámot, meg egyéb zablákat, amelyekkel mások korlátozhatják a szabadságában. Az itt következő két hír kerülhetett volna a rövid tudósítások közé is, de ott talán nem figyeltek volna fel rá annyian az olvasók közül. Persze még mindig eldönthetik, kíváncsiak-e rájuk. Szabadon...

A *Philips* bejelentette legújabb szabadalmát, mely megvalósítása esetén nem teszi lehetővé a csatornaváltást, amennyiben reklámot sugároznak az éppen nézett adón. A szabadalom lehetővé teszi továbbá azt is, hogy interaktív díjfizetés (például emelt díjas *SMS*) ellenében mégis lehetővé váljon a csatornaváltás. Ezt a szabadalmat az egyre rosszabb reklámozás hatékonysággal indokolta a Philips.

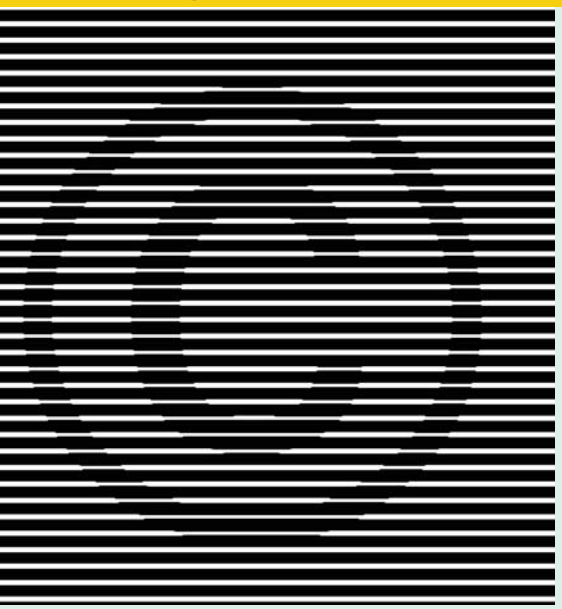
El az kezekkel a távirányítótól!

Nos járjuk körül a témát. A szabadalom ötlete nem új, hiszen számos műsoros *DVD*-t láttam már, aminek az elején volt egy-két 20-30 másodperces „kötelező” reklám, amit átugrani nem lehetett, jobb esetben is csak gyorsíthatjuk a lejátszást.
De hagyjuk az emóciókat, és vizsgáljuk meg a kérdést gazdasági oldalról. Mire is jók a reklámok? Bemutatnak egy-egy új terméket, vagy felfrissítik a régiekről az emlékeinket, egyfajta lenyomatot hagyva a tudatalattinkba, hogy legközelebb a boltban na az X márkájú joghurtot vegyük le a polcról, hanem Y-t. Azt nyilván mindenki sejti, hogy a reklámok ára be van építve a termék árába. És itt jön a költői kérdés: miért fizessünk kétszer is a reklámokért? Egyszer a boltban és még egyszer a *TV*-ben akár az időnkkel, akár mondjuk egy emelt díjas *SMS* formájában? Közelítsük meg egy másik oldalról is a dolgot.
Tegyük fel, hogy veszünk egy tévét több tízezer forintért. A kiadásoknak ilyenkor közel sincs vége, hiszen a kábeltelevíziós szolgáltatásért is fizetni kell. A legtöbb adón vannak reklámok. Film közben is. Ezek szerint tehát a tévé egyben hirdetési felület is. Ezek után talán nem nagy elvárás, ha azt szeretnénk, legalább a reklám legyen ingyenes, ha pedig nem tetszik, vagy elegünk van belőle, akkor szabadon – a *Nagy Testvér* közreműködése nélkül – átválthassunk egy másik csatornára.

Gratulálunk, ön nyert 203 ezer dollár jogdíjat!

Csekkel vagy készpénzben kívánja befizetni?

Írj szabad szoftvert és fizess 203 ezer dollárt a szabadalmak miatt. *Ben Jacobsen* – hobbi modellvasút építő – közzétett pár szabad forrású szoftvert, amely lehetővé teszi, hogy terepasztalunkat a *PC*-ről vezéreljük. A gond amiatt adódott, hogy a *KAM* cég is rendelkezik egy hasonló kereskedelmi programmal és ragaszkodik a szoftver szabadalmi jogaihoz. A 203 ezer dollár a hétezer letöltésből jött össze, amelyek – állítólag – darabonként 29 dolláros kárt okoztak a cégnek.



Nos próbáljunk analógiát keresni. Tegyük fel, hogy valaki kiad egy, a ***Szabad Kultúra*** című könyvhöz nagyon hasonló írást, persze nem ***Creative Commons*** jogállással? Követelhetne-e ez a másik kiadó a ***Kiskapu Kiadótól*** – uram bocsá' magától ***Lawrence Lessigtől*** – több százezer forintot (dollárt?), csak mert az ő eladásai az ingyenesen letölthető ***PDF*** változat miatt csökkentek? Reméljük nem.

Mármost miért kellene hogy máshogyan legyen ez a szoftverek világában? Ha valaki esetleg jobbat tud írni egy kereskedelmi programnál és ráadásul olcsóbban – netán ingyenesen – had tegye meg. Az emberi kreativitást amúgy sem lehet gátak közé szorítani. Nem olyan a természete.
A szoftverszabadalmak emlegetése óta sokan tartanak tőle, hogy egyszer majd a Microsoft is előáll valami olyan ötlettel, hogy a Linux fejlesztői és más szabad szoftverek fejlesztői fizessenek annyiszor 140 ezer forintot (a ***Windows XP*** és a ***Microsoft Office*** ára) elmaradt haszon címén, ahányan a ***Linux-OpenOffice.org*** kombináció mellett döntöttek. A józanul gondolkodók persze tudják, hogy ez nem különösebben reális elgondolás, de – ahogy azt maguk a jogászok is gyakran elismerik – a törvény és az állampolgárok jogérzéke néha nem fedi egymást. Mindazonáltal logikusan gondolkodva egy szoftverrel kapcsolatban három dolog lehetséges: megvesszük, nem vesszük meg, vagy írunk egyet – akár ingyen is.

Persze sok víznek kell még lefolyni a ***Dunán***, mire a szabadalmi hivatalokban dolgozók, a politikusok és a gazdasági szakemberek is rájönnek, hogy az ember alapvetően szabadnak születik, s így a fizikai és gondolati szabadság olyan alapjog, amit semmilyen gazdasági kényszer nem írhat felül.

Medve Zoltán
(e-medve@e-medve.hu)

2001-ben kezdett „Linuxolni", de már korábban is ismerkedett a szabad szoftverek világával. Ha éppen nem a gép előtt ül, akkor fotózgat, olvasgat vagy bicajozik.

KAPCSOLÓDÓ CÍMEK

- http://hardware.slashdot.org/article.pl?sid=06/04/18/2032219
- http://righttocreate.blogspot.com/2006/04/write-free-software-pay-203000-to.html

Többnyelvű programok létrehozása a GNU gettext rendszerrel

A legtöbb kereskedelmi és ingyenes programot manapság angol nyelven írják és dokumentálják. Csak az utóbbi időben tettek kísérletet arra, hogy ezen kívül más nyelven is próbáljon a számítógép kommunikálni a felhasználóval. A GNU gettext program megjelenése sokat javított a helyzeten.

A legtöbb *GNU* program ma már a *gettext* kerettel készül, amely lehetővé teszi a programok egyszerű fordítását. Ebben a rendszerben, ha rendelkezésre áll a fordítás, akkor a programoknál futás előtt, vagy akár közben állíthatjuk a nyelvet. A *gettext* egy olyan egyszerű módon oldja meg ezt, amely megkönnyíti a programozó és a fordítók munkáját is, mi több, külön, egymástól függetlenül tudnak dolgozni.
Ez az írás a *gettext* rendszert mutatja be. Létezik egy másik hasonló célú programcsomag is, a *catgets*, amelyik az *X/Open Portability Guide* ajánlása, de azt nem részletezzük.

Az első program

Kezdjük a *gettext* megismerését is a szokásos *„Hello, world”* programmal! Ennek a nemzetközi változata a következőképpen néz ki:

```
#include <libintl.h>
#include <locale.h>
#include <stdio.h>
int main()
{
  setlocale(LC_ALL, "");
  bindtextdomain
  ➥("hello", "/usr/
  ➥share/locale");
  textdomain("hello");
  printf(gettext("Hello,
  ➥world!\n"));  return 0;
}
```

Nyilván egy igazi programnál érdemes a visszatérési értékeket ellenőrizni. Mentsük el ezt a kis forráskódot *hello.c* néven, és fordítsuk le a programot a szokásos

```
gcc -o hello hello.c
```

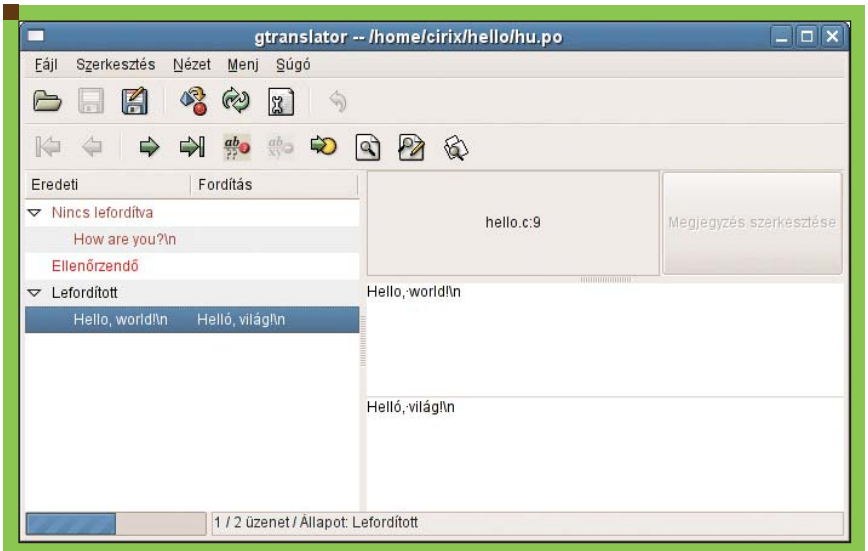
paranccsal. A programhoz kell szerkeszteni a *GNU libintl* könyvtárat; *GNU* rendszerekben ezt a *glibc* tartalmazza, máshol szükség lehet a `-lintl` paraméterre is.

A programozó feladata

A *hello* program a szokásos „Hello, world!” üzenetet írja a képernyőre. Néhány függvényhívás után, amelyeket csak a program indulásakor kell megtenni, az összes feladata a programozónak, hogy minden karakterláncot a `gettext(string)` függvénnyel feldolgoztasson kiíratás előtt. Megszokott ennek a rövidítése is, hogy még kevesebbet kelljen gépelni:

```
#define _(STRING)
➥gettext(STRING)
...
printf(_("Hello, world!"));
```

Nézzük a programot soronként. A *locale.h* néhány területi információt tároló C struktúrát határoz meg. Ez tartalmazza a `setlocale()` függvény prototípusát. A `libintl.h` pedig a *GNU gettext* függvényeket tartalmazza, a `bindtextdomain()`-t, a `gettext()`-et és a `textdomain()`-t. A `setlocale(LC_ALL,"")` függvényhívás beállítja a program területi paramétereit a környezeti változók, vagyis a felhasználó kérése alapján. A `bindtextdomain()` függvény megadja, hogy egy adott nyelv üzenet fájljai melyik könyvtárban találhatóak. A csomag neve általában megegyezik a program nevével.



1. ábra A gtranslator munka közben

Az üzenetkatalógusok alapértelmezés szerinti helye az */usr/share/locale* könyvtár (ebben az esetben egyébként el is hagyhattuk volna ezt a függvényhívást).
A `textdomain()` függvényhívás kiválasztja a ***hello*** nevű üzenetkatalógust.
A `printf("Hello, world\n");` sort pedig kicseréltük a `printf(gettext("Hello, world!\n"));` sorra. Ez a módosítás teszi lehetővé, hogy a fordító a programozótól külön dolgozhasson. A paramétere az angol nyelvű mondat. Visszatérési értéke pedig egy olyan karakterláncra mutat, amely a memóriában az adott nyelvre lefordított szöveget tartalmazza. Ha nem talál fordítást, az eredeti mondattal tér vissza. (Ezért látni a frissebb programokban elszórva angol szavakat.)

A fordítandó üzenetek

Kerestessük ki a fordítandó üzeneteket a forráskódból. Erre az `xgettext` program való, amelyet a következő paranccsal kell elindítanunk:

```
xgettext -d hello -s -o
➥hello.pot hello.c
```

Ez feldolgozza a ***hello.c*** forrást, és elkészíti a ***hello.pot*** fájlt (amelyet a `-o` paraméterrel adtunk meg, output). A `-s` hatására (sort) ABC szerint rendezi az üzeneteket a program, a `-d` pedig az üzenetkatalógus nevét (domain) állítja be, amelyet a `textdomain()` függvényhívásnak is meg kell adnunk.
A ***hello.pot*** (portable object template) fájl egy sablon, amely alapja a fordításokat tartalmazó fájloknak. A fordítás elkezdéséhez lemásolhatjuk a sablon fájlt egy másik néven, de a egyszerűbb módja ennek az ***msginit*** program használata, amely néhány változót egyből beállít a helyes értékre:

```
msginit -l hu_HU -o hu.po -i
➥hello.pot
```

Itt a `-l` paraméter (locale) adja meg a terület és a nyelv nevét, a `-i` és a `-o` paraméterek pedig a bemeneti és a kimeneti fájlokat (input, output). Ha csak egy sablon fájl van az aktuális könyvtárban, azt használja a program automatikusan. Az ***msginit*** megkérdezi a fordító e-mail címét.
A ***hu.po*** (portable object) fájlban az egyes bejegyzések formátuma a következő:

```
#  fordító megjegyzései
#. automatikus megjegyzései
#: hivatkozás
#, flag-ek
msgid "fordítatlan üzenet"
msgstr "lefordított üzenet"
```

A megjegyzések a `#` (kettőskereszt) karakterrel kezdődnek. Két fajtájuk van. Az első a kézzel, fordító által hozzáadott megjegyzések, amelyeknél a kettőskeresztet egy szóköz követi. A második pedig automatikus megjegyzés a ***gettext*** program által. Az `msgid` után a fordítatlan (általában angol nyelvű) üzenet van, az `msgstr` után pedig a lefordított üzenetet kell beírni.

A fordítás

A ***hu.po*** fájlt szerkeszthetjük a fordításhoz. Erre bármilyen szövegszerkesztő program alkalmas, ha az ember pontosan betartja a formátumot; érdemesebb azonban egy erre a célra való programot használni. Ilyenek a ***kbabel***, a ***gtranslator***, a ***poedit***; az ***Emacs*** is képes ***.po*** módban dolgozni.
Először a fejlécet töltsük ki. Ezt részben az ***msginit*** program elvégezte. A `Project-Id-Version` után írjunk annyit, hogy `hello 0.1`. A munka nagy része az egyes üzenetek fordítása. Minden `msgid` sor után írjuk be az `msgstr`-hez a magyar szöveget, idézőjelek között. Vagyis az msgid `"Hello, world!\n"` után `msgstr "Helló, világ!\n"` kerül. Figyeljünk arra, ha elmentjük a fájlt, a `Content-Type` sornál megadott karakterkészletet használjuk. Magyar nyelv esetén ez ***ISO-8859-2*** vagy ***UTF-8*** lehet. Az ***msginit*** program a kiválasztott nyelvkódtól függően állítja ezt be.

Az üzenetkatalógusok

A fordítás elvégzése után a ***.po*** szövegfájlt bináris formátumra kell hoznunk, hogy abban a ***gettext*** programok gyorsan meg tudják találni a karakterláncokat. Ehhez a következő parancsot kell használni:

```
msgfmt -c -v -o hello.mo hu.po
```

A `-c` opció részletesen ellenőrzi a ***.po*** fájlt. A `-v` hatására részletes üzeneteket kapunk. A kimeneti fájl nevét a `-o` opció után adhatjuk meg. Fontos, hogy a kimeneti fájl neve az üzenetkatalógus nevével egyezzen meg, amelyet a `bindtextdomain()` és `textdomain()` függvénynek is megadunk. A ***.mo*** (machine object) fájl a `bindtextdomain()` második paraméterében megadott könyvtárban kell legyen a program futásakor. A teljes elérési út a ***megadott_könyvtár/nn_OO/LC_MESSAGES/hello.mo***,

ahol nn a nyelv (hu) és OO az ország (HU). Így különböztetik meg például a kanadai franciát a franciaországitól: fr_CA és fr_FR. A *hello.mo* fájl szabványos helyre másolásához természetesen rendszergazdai jogosultság szükséges:

```
mkdir -p /usr/share/locale/
➥hu_HU/LC_MESSAGES
cp hello.mo /usr/share/locale/
➥hu_HU/LC_MESSAGES
```

(Ha enélkül szeretnénk kipróbálni a *gettext*-et, legyen a `bindtextdomain()` második argumentuma "" üres karakterlánc, de a *hello.mo* fájlt ebben az esetben is az aktuális könyvtárhoz képest a *hu_HU/LC_MESSAGES* helyre kell másolnunk!)

A felhasználó feladata

Ha az üzenetkatalógusokat megfelelően telepítettük, a rendszer bármelyik felhasználója kipróbálhatja a magyar nyelvű *Helló, világ* programot. Állítsuk be a megfelelő környezeti változót (bár ezt valószínűleg a *Linux* terjesztésünk már tartalmazza), utána pedig indítsuk el a programot:

```
export LC_ALL=hu_HU
./hello
```

Figyeljünk arra, hogy terminálunk vagy terminál ablakunk ismerje a megfelelő nem angol nyelvű betűket, különben a kimenet hibás lehet.

A program továbbfejlesztése – új üzenetek

Bővítsük kicsit az előbbi programunkat. Adjunk hozzá még egy mondatot:

```
...
printf(gettext("Hello,
➥world!\n");
printf(gettext("What's up?\n");
...
```

Ha csak ennyit változtatunk, legegyszerűbb lenne újra végigcsinálni a fenti lépéseket, és lefordítani mind a két üzenetet magyar nyelvűre. Egy nagy, több ezer üzenetet tartalmazó programnál nem ilyen egyszerű a helyzet. Jó lenne, ha csak az újonnan megjelent sorokat kellene újra lefordítanunk.

Keressük ki az *xgettext* programmal újra a fordítandó mondatokat:

```
xgettext -d hello -s -o
➥hello_uj.pot hello.c
```

Az *msgmerge* program egy régebbi, már lefordított *.po* fájlt tud összefűzni egy új .pot (sablon) fájllal:

```
msgmerge -s -U hu.po
➥hello_uj.pot
```

A `-U` opcióval (update) jelezzük, hogy a régi *hu.po* fájlt szeretnénk továbbfejleszteni. Az új sablon fájl tartalmazni fogja a régi fordításokat, az újonnan megjelent soroknál pedig üres karakterláncot. Fordítsuk le az új sort:

```
#. hello.c:11
msgid "What's up?\n"
msgstr "Mi a helyzet?"
```

Ha valamelyik *.po* fájl szerkesztő programot használjuk, akkor az külön mutatja a már lefordított és még feldolgozandó sorokat. Ezután pedig készítsük el a *.mo* fájlt, ahogy a fenti esetben:

```
msgfmt -c -v -o hello.mo hu.po
mkdir -p /usr/share/locale/
➥hu_HU/LC_MESSAGES
cp hello.mo /usr/share/locale/
➥hu_HU/LC_MESSAGES
```

Egyéb tudnivalók

Néhány dologra figyelnünk kell a fordítás közben. Legfontosabb, hogy ha a fordítandó üzenetünk C formátum karaktereket tartalmaz, azokat meg kell tartanunk, méghozzá a megfelelő sorrendben. Megfelelően fogalmazva a mondatot ez könnyen elérhető. Például:

```
#  helyes
msgid "Read %d lines from file
➥%s.\n"
msgstr "%d sor beolvasva a %s
➥fájlból.\n"
```

Helytelen viszont a következő sor:

```
#  helytelen
msgstr "%s fájlból beolvasva
➥%d sor.\n"
```

A C formátum karakterláncokat az *xgettext* megjelöli a `#, c-format` megjegyzéssel. (Ennek ellenkezője a `#, no-c-format`). A `fuzzy` jelzőt is ennél a sornál állíthatjuk be vagy törölhetjük. „Fuzzy" jelölést szokás tenni az ellenőrzendő vagy pontatlan fordítások mellé. Ezeket a szerkesztő programok külön kategóriába teszik.
Grafikus rendszerek gyakran _ (aláhúzás) karakterrel jelölik a gyorsbillentyűket. Ezen sorok fordítását fontolóra kell venni. Figyelni kell rá, hogy ne adjuk több gombnak vagy menünek ugyanazt a gyorsbillentyűt.
Az *xgettext* program a `-c` opcióval indítva az egyes karakterláncok elé megjegyzést is ír, amelyet a forráskódból másol át. Minden `msgid` elé az a megjegyzés kerül, amely a kódban a `gettext()` függvényhívást megelőzte. Ha ilyeneket helyezünk el a programunkban, az nagyban megkönnyítheti a fordító munkáját. Fontos azonban, hogy ez a megjegyzés csak *ASCII* karaktereket tartalmazzon, vagy adjuk meg azok karakterkódolását is a `--from-code` opcióval! Ellenkező esetben az *xgettext* nem fut le. Ráadásul a jelenlegi verzió (0.14.5) a `gettext()` függvénynek átadott karakterláncot jelzi hibásnak, nem pedig a megjegyzést!
Az *msgmerge* program #~ jelzéssel látja el azokat a sorokat, amelyek egy új programverzióban már nem szerepelnek. Ha nem a saját programunkat fordítjuk, érdemes ezeket figyelmen kívül hagyni; semmiképpen sem kitörölni, hátha egy későbbi verzióban megint szerepel majd.
A *gettext* csomag része egyébként a gettextize program is, amely felkészít egy programrendszert a *gettext* használatára, vagy egy újabb *gettext* verzióra. Ennek és az eddig említett programoknak a használatát a ➲ *http://www.gnu.org/software/gettext/manual/* oldal mutatja be.

Czirkos Zoltán
Jelenleg diplomatervező a Budapesti Műszaki Egyetem Elektronikus Eszközök Tanszékén. Kutatási területe az operációs rendszerek betörésvédelme és a P2P kommunikáció. 2005-ben a Tudományos Diákköri Konferencián II. helyezést ért el. Kedvencei a boszorkányos és a rózsaszín párducos filmek.

Adatbázis alapú webes alkalmazás mindössze 18 sornyi kóddal

Webes, adatbázis alapú alkalmazás készítése a nulláról indulva, nyolc könnyed lépéssel.

A *LAMP* összeállítás – *Linux*, *Apache*, *MySQL* és valamilyen programozási nyelv, általában *Perl*, *Python* vagy *PHP* – rendkívül sokoldalú megoldás. Aki készített már webes, adatbázis alapú alkalmazást, az, hogy az ismert fordulattal éljünk, mindet elkészítette. Programozói szemszögből a dolgok hamar ismétlődővé, unalmassá válnak.

Első webes alkalmazásomat a közelmúltban készítettem el. Természetesen linuxos alapokra építettem, *Apache* alatt futott, és *MySQL*-lel folytatott párbeszédet. Ragasztónyelvként, a többi nyelven készült összetevő összefogására szolgáló nyelvként *Perlt* használtam, és az egészet gazdagon megszórtam *CGI*-díszítéssel. Megírtam minden a *MySQL* egyik táblájának bővítéséhez, módosításához és frissítéséhez szükséges kódot, ahogy azt illik. Működött, szép és jó volt. A bajok akkor kezdődtek, amikor ugyanezeket a feladatokat az adatbázis többi táblájához is el kellett végezni. A *Perl* programozók körében régóta bevált szemléletet követve elkezdtem azon gondolkodni, hogyan lehetnék egyszerre lusta és termelékeny. Kellett lennie valami jobb megoldásnak.

Néhány hamvába holt kísérlet és némi keresgélés után rátaláltam a Maypole-ra.

A *Maypole* egy gyors alkalmazásfejlesztési keretrendszer webes alkalmazásokhoz. Eredetileg *Simon Cozens* hozta létre, karbantartását *Sebastian Riedel* vállalta fel. A *Maypole* honlapján azt az ígéretet olvastam, hogy akár 20 sornyi *Perl*-kóddal is teljes értékűen működő alkalmazásokat hozhatok létre. Túl jól hangzott ahhoz, hogy ne próbáljam ki.

Miután megismerkedtem a *Maypole*-lal, állíthatom, *Simon* és *Sebastian* nem hazudtak. Elég néhány sornyi kódot begépelnünk, máris sokoldalú alkalmazáshoz juthatunk. Némi telepítgetésre természetesen szükség van, de hadd védekezzek azzal, hogy az nem programozói munka. Ha a telepítéssel végeztünk, szinte onthatjuk az alkalmazásokat, mindegyik elkészítéséhez csupán egy pár sornyi kódot kell összeállítanunk. A továbbiakban végigvezetem a kedves olvasókat azon a folyamaton, amelynek végén – a *Maypole* közreműködésével – egy kész alkalmazást kapnak a kezükbe.

1. lépés: A Linux telepítése – ha szükséges

Az első lépésről külön cikket lehetne írni. Huszáros vágással intézzük el ennyivel: válasszuk ki a nekünk tetsző terjesztést, majd telepítsük.

Mivel nemrég új gépet vettem, letöltöttem a *Fedora Core 3*-at, majd egyedi telepítést választva mindent feltettem a gépre. Aki ezt a luxust nem engedheti meg magának, az az alábbi csomagokat mindenképpen válassza ki: *httpd*, *httpd-devel*, *mod_perl*, *mod_perl-devel*, *mysql* (ügyfél és kiszolgáló) és *Perl*.

2. lépés: Az Apache/mod_perl környezet előkészítése

A korszerűbb terjesztésekben már az *Apache 2*-es, valamint a *mod_perl 1.99*-es változata található meg, bár néhányan leragadtak az *Apache 1.3.x* kiadásainál. Szerencsére a *Maypole* az *Apache* bármely változatával képes együttműködni, és – ha a mod_perl nem áll rendelkezésre – *CGI* alapú üzemre is átállítható. Saját *Fedora* telepítésemben az *Apache 2.0.52*-es és a *mod_perl 1.99_16-3*-as változata szerepelt, így a továbbiakban ezek használatát feltételezem.

A *Maypole* levelezési lista tagjai számos különböző *Linux* változatra, köztük *SuSE*, *Debian* és *Red Hat* terjesztésre telepítették már sikeresen a keretrendszert. A *Maypole* az *Apple Mac OS X* operációs rendszere alatt is futtatható, sőt, némi erőfeszítéssel *Microsoft Windows* alatt is életet lehelhetünk belé.

Rootként módosítottam a *Fedora Apache* beállító fájlját (*/etc/httpd/conf/httpd.conf*), és megjegyzésbe tettem a `ServerTokens` utasítást. Gondoskodtam arról, hogy rendszerindításkor az Apache is elinduljon, majd az alábbi parancsokkal üzembe helyeztem a webkiszolgálót:

```
chkconfig httpd on
service httpd start
```

A kiszolgáló állapotát legkönnyebben a szöveges *lynx* böngészővel tudjuk ellenőrizni:

```
lynx -head -dump http://localhost/
```

Az eredmények – lásd a kimenet harmadik sorát – igazolják, hogy az *Apache* és a *mod_perl* üzemkész:

```
HTTP/1.1 403 Forbidden
Date: Wed, 17 Nov 2004 23:30:01 GMT
```

```
Server: Apache/2.0.52 (Fedora)
            mod_perl/1.99_16 Perl/v5.8.5 DAV/2
Accept-Ranges: bytes
Content-Length: 3931
Connection: close
Content-Type: text/html; charset=UTF-8
```

Minden rendben volt, ezért újra megnyitottam a *httpd.conf* fájlt, és kivettem megjegyzésből a `ServerTokens` utasítást, ugyanis nem túl szerencsés, ha webkiszolgálónk lapjaiba be engedjük tekinteni az esetleges támadókat. Ha már úgyis meg volt nyitva a fájl, a `ServerAdmin` utasításban megadtam a saját e-mail címemet, majd a `ServerName` utasításban a kiszolgálóm *DNS*-nevét. Egyúttal feljegyeztem a `DocumentRoot` utasításban megadott könyvtár elérési útját; esetemben ez a */var/www/html* volt.

3. lépés: A MySQL előkészítése

A kiválasztott terjesztéstől függően lehetséges, hogy a *MySQL* már telepítve van a gépünkre. Ha nem ez a helyzet, akkor a *MySQL*-t a terjesztésünkhöz tartozó letöltések közül vagy a *MySQL* weboldaláról szerezhetjük be. Saját, *Fedora* alapú gépemen – rootként bejelentkezve – a szokásos parancsokkal készítettem fel használatra a *MySQL*-t:

```
chkconfig mysqld on
service mysqld start
```

Miután elindult a *MySQL*, megadtam a hozzá tartozó felügyeleti jelszót:

```
mysqladmin -u root password 'jelszo'
```

4. lépés: A Maypole telepítése

A *Maypole* és az *Apache* között a *mod_perl* közreműködésével közvetlen párbeszéd folyik. Ha az *Apache 2*-es változatával szeretnénk dolgozni, akkor a *libapreq2* nevű fejlesztői könyvtárat is le kell töltenünk a *CPAN* gyűjteményből, majd telepítenünk kell a *Perl* csomagok közé. Jómagam a *libapreq2-2.04_03-dev.tar.gz* fájlt töltöttem le a *CPAN*-ról. A könyvtár telepítése előtt frissítettem a *Perlhez* alapesetben is hozzá tartozó `ExtUtils::XSBuilder` modult. Rootként elegendő az alábbi parancsot kiadni:

```
perl -MCPAN -e "install ExtUtils::XSBuilder"
```

Ha ez a *CPAN* héj első futtatása, akkor lehetőséget kapunk a helyi *CPAN* modul beállításainak megadására. Amikor rákérdez azoknak a moduloknak a letöltésére, melyek megléte a működés előfeltétele, akkor igyekezzünk odafigyelni. Miután frissítettem a modult, telepítettem a *libapreq2* könyvtárat; ennek során a szokásos, a Perl modulok telepítésére használt parancsokat adtam ki:

```
tar zxvf libapreq2-2.04_03-dev.tar.gz
cd libapreq2-2.04-dev/
perl Makefile.PL
make
make test
su
make install
<Ctrl-D>
```

A *Maypole* tényleges telepítése a *CPAN* héj rootként való meghívásával kezdődik:

```
perl -MCPAN -e "shell"
```

Mivel a *Maypole* működéséhez számos *CPAN* modul jelenlétére van szükség, a telepítés eltart egy ideig. Mielőtt a *CPAN* héjat a *Maypole* tényleges telepítésére utasítanánk, az alábbi parancsok kiadásával ellenőrizhetjük, hogy az időnként gondot okozó modulokkal is minden rendben van-e:

```
cpan> install CGI::Untaint::date
cpan> force install Class::DBI::mysql
```

Nálam a `Class::DBI::mysql` telepítését úgy kellett ráerőltetni a gépre, miután több ellenőrzés is sikertelen volt, illetve az önműködő telepítés is megszakadt. A telepítés erőltetésekor a sikertelen ellenőrzéseket a program figyelmen kívül hagyja, így végig tud futni a folyamat. Ha az előzetes feltételek teljesülnek, a *Maypole* telepítését a következő *CPAN* paranccsal kezdhetjük meg:

```
cpan> install Maypole
```

A parancs kiadása után a program különféle önműködő párbeszédeket folytat le a *CPAN* gyűjteménnyel. Figyeljünk, hogy mi történik, ugyanis bizonyos pontokon egyszerű kérdésekre kell válaszolnunk. Amikor végeztünk, pihegjük ki magunkat – szép munkát végeztünk, a *Maypole* legújabb (írásom születésekor 2.04-es) változata ott vár bennünket a gépünkön.

5. lépés: Az adatbázis és néhány tábla létrehozása

Visszatérve a *MySQL*-hez, bejelentkeztem mint rendszergazda, és az alábbi parancsok kiadásával az összes alapértelmezett fiókot töröltem:

```
mysql -u root -p
mysql> use mysql;
mysql> delete from user where User = '';
mysql> flush privileges;
```

Ezután létrehoztam egy új adatbázist, valamint egy felhasználót is, akit az adatok tulajdonosának szántam:

```
mysql> create database CLUB;
mysql> grant all on CLUB.* to manager identified
          by 'jelszo';
mysql> quit
```

A fenti parancsok hatására létrejön egy *CLUB* nevű adatbázis, valamint bekerül a rendszerbe egy *manager* nevű felhasználó. A példa kedvéért tegyük fel, hogy alkalmazásunk célja egy gyermek futballklub tagjainak nyilvántartása. A rendszer tárolja a játékosok személyes adatait, azt, hogy

melyik csapatba tartoznak, valamint egészségi állapotukat. Nézzünk néhány *SQL*-fájlt, ezek segítségével adtam meg a *CLUB* adatbázison belüli táblákat. Az első fájl (*create_player.sql*, játékos létrehozása) a *player (játékos)* táblát hozza létre:

```
create table player
(
    id                  int not null
                        ➥auto_increment
                            primary key,
    name                varchar (64) not null,
    date_of_birth       date,
    address             varchar (255),
    contact_tel_no      varchar (64),
    squad               int,
    medical_condition   int
);
```

A második fájl (*create_squad.sql*, csapat létrehozás) a *csapatok (squad)* kezdő listáját hozza létre:

```
create table squad
(
    id    int not null auto_increment
          ➥primary key,
    name  varchar (32) not null
);
insert into squad (name) values ('--');
insert into squad (name) values ('Under 8');
insert into squad (name) values ('Under 9');
insert into squad (name) values ('Under 10');
insert into squad (name) values ('Under 11');
insert into squad (name) values ('Under 12');
```

A csapat táblát megpróbáltam értelmes kezdő adatokkal feltölteni (például *„under 8” - „8 év alattiak”* stb). A harmadik, s egyben utolsó fájl, a *create_condition.sql* a lehetséges egészségi állapotok listáját hozza létre:

```
create table condition
(
    id    int not null auto_increment
          ➥primary key,
    name  varchar (64) not null
);
insert into condition (name) values ('--');
insert into condition (name) values ('Asthma');
insert into condition (name) values ('Epilepsy');
```

A `squad` táblához hasonlóan a `condition` táblába is beilleszt-tettem néhány alapadatot. A `squad` és a `condition` tábla adattételének neve `name` *(név)*. Ennek jelentőségét később még látni fogjuk.
Az *SQL* fájlokkal tudjuk létrehozni az adatbázison belül a táblákat:

```
mysql -u manager -p CLUB < create_player.sql
mysql -u manager -p CLUB < create_squad.sql
mysql -u manager -p CLUB < create_condition.sql
```

Bizonyára mindenki kitalálta már, hogy a *CLUB* adatbázis a játékosok adatait fogja tárolni. A játékosok egy csapathoz tartoznak, és van valamilyen egészségi állapotuk.

6. lépés: Az alkalmazás beállítása

Készen áll az adatbázis, a *Maypole* is telepítve, eljött tehát az alkalmazás beállításának ideje. Az *Apache httpd.conf* beállító fájljához egy új utasítást kell hozzáadni, amelynek alapján a *mod_perlre* bízzuk a *Maypole* alkalmazás kezelését. Én a következőket írtam a fájl végére:

```
<Location /Club>
    SetHandler perl-script
    PerlHandler ClubDB
</Location>
```

A fenti sorokból tudja meg az *Apache*, hogy amikor a */Club URL* iránti kérést kap, akkor azt a *ClubDB Perl* parancsfájlnak – ezt a következő lépésben írjuk meg – kell átadnia feldolgozásra. A következő parancsokat rootként kiadva adtam meg az *URL* helyét:

```
mkdir /var/www/html/Club
cd /var/www/html/Club
cp -r ~/.cpan/build/Maypole-2.04/templates/* .
cp maypole.css ../club.css
```

Először létrehoztam egy könyvtárat, az *Apache* gyökérkönyvtárán belül ez tartalmazza az alkalmazásom *URL*-jét, majd belemásoltam a *Maypole*-hoz mellékelt alapértelmezett sablonokat. A *Maypole CSS* fájlját is bemásoltam a webkiszolgáló *DocumentRoot* könyvtárába, majd az alkalmazásomnak megfelelően átneveztem. Az utolsó telepítési jellegű lépés egy beállító fájl létrehozása az *Apache /etc/httpd/conf* könyvtárán belül, melynek feladata az alkalmazás *MySQL* felhasználónevének és jelszavának tárolása. Ebbe a *ClubDB.conf* nevű fájlba a következő sorok kerültek:

```
[client]
user=manager
password=jelszo
```

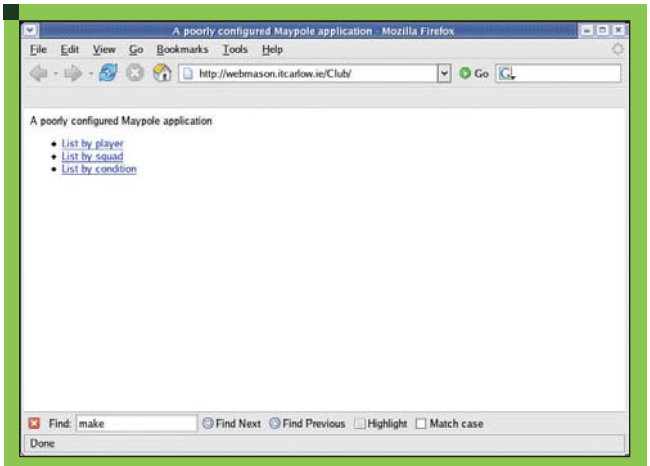
7. lépés: A 18 sornyi kód megírása

A labdarúgó klub adatbázis-kezelő kódja a *ClubDB.pm* fájlban található. Minden *Maypole* alkalmazás egy *Perl* névteret megadó `package` utasítással kezdődik. Bekapcsoljuk a *„szigorúságot” (strictness)*, majd használatba vesszük az `Apache::MVC` nevű *Maypole* alapmodult:

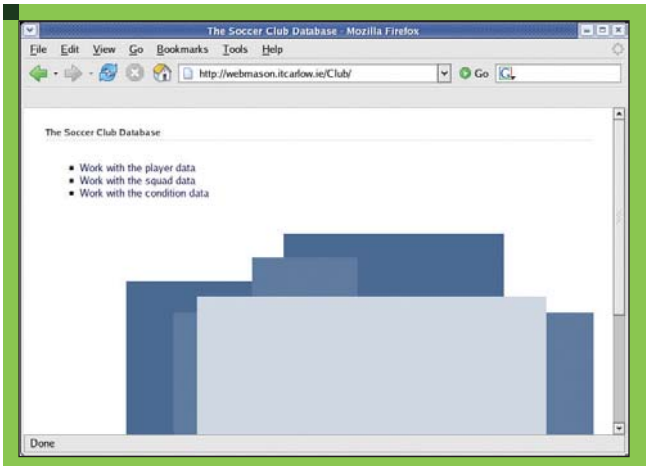
```
package ClubDB;
use strict;
use base 'Apache::MVC';
```

A kód ezután a megadott beállító fájlban szereplő felhasználónév és jelszó használatával kapcsolatot létesít az adatbázissal:

```
ClubDB->setup( "dbi:mysql:CLUB;
    mysql_read_default_file=
        /etc/httpd/conf/ClubDB.conf" );
```



1. ábra A Maypole alapértelmezett nyitóképernyője



2. ábra A labdarúgó klub testreszabott nyitóképernyője

A következő sorok tájékoztatják a ***Maypole***-t az alkalmazás kezdő webcíméről, továbbá megadják, hogy az adatbázis mely tábláinak elérésére van szükség. Mivel egyszerű programról van szó, a legjobb, ha minden tábla elérhetőségét biztosítjuk:

```
ClubDB->config->{uri_base} =
    "http://webmason.itcarlow.ie/Club/";
ClubDB->config->{display_tables} =
    [ qw[ player squad condition ] ];
```

A csapatokra térve: az alkalmazás lehetővé teszi a csapatnevek megtekintését, módosítását és törlését. Ennek megadása további pár sort tesz ki, ezek egyike újabb névteret ad meg:

```
package ClubDB::Squad;
sub display_columns{ "name" };
ClubDB::Player->untaint_columns(
    printable => [ "name" ] );
```

Az `untaint_columns` eljárás az oszlopban várt adat típusát adja meg, és itt jelezzük azt is a ***Maypole***-nak, hogy az oszlop a webes felületen keresztül módosítható. Az egészségi adatok kezelése hasonló módon történik:

```
package ClubDB::Condition;
sub display_columns{ "name" };
ClubDB::Condition->untaint_columns(
    printable => [ "name" ] );
```

A `player` táblához tartozó kód egy picit bonyolultabb, de csak picit. Újabb névteret adunk meg, majd a has_a eljárást kétszer meghívva létrehozzuk a `player` és a többi tábla közötti kapcsolatokat. A kapcsolatok kifejezése a korábban megadott névterekkel történik:

```
package ClubDB::Player;
ClubDB::Player->has_a(
    squad => "ClubDB::Squad" );
ClubDB::Player->has_a(
    medical_condition => "ClubDB::Condition" );
```

A `players` esetében a megjelenítendő oszlopokat a `display_columns` eljárás segítségével soroljuk fel. Ezzel a módszerrel a programozó szabályozhatja az oszlopok webes felületen való megjelenésének sorrendjét. Ha a `display_columnst` elhagyjuk, a ***Maypole*** ABC sorrendben jeleníti meg az oszlopokat, ami nem feltétlenül felel meg az igényeinknek. Az `untaint_columns` meghívásával határozzuk meg, hogy az oszlopokon belül milyen típusú adatokat szabad módosítani. A kód a ***Perlben*** jól ismert „`1;`” lezárással végződik, mely minden Perl modul elmaradhatatlan kelléke:

```
sub display_columns{ qw( name address
    date_of_birth contact_tel_no
        squad medical_condition ) };
ClubDB::Player->untaint_columns(
    integer    =>
        [ "squad", "medical_condition" ],
    printable =>
        [ "name", "address", "contact_tel_no" ],
    date       =>
        [ "date_of_birth" ] );
1;
```

Számoljuk meg a pontosvesszőket! Ha figyelembe vesszük, hogy a fenti kódrészleteket a nyomdai kötöttségeknek megfelelően kellett tördelni, akkor láthatjuk, hogy bizony összesen 18 sornyi kódot írtunk. Végső lépésünk a ***Perl*** modul olyan helyre másolása, ahol az ***Apache*** és a ***mod_perl*** is meg tudja találni:

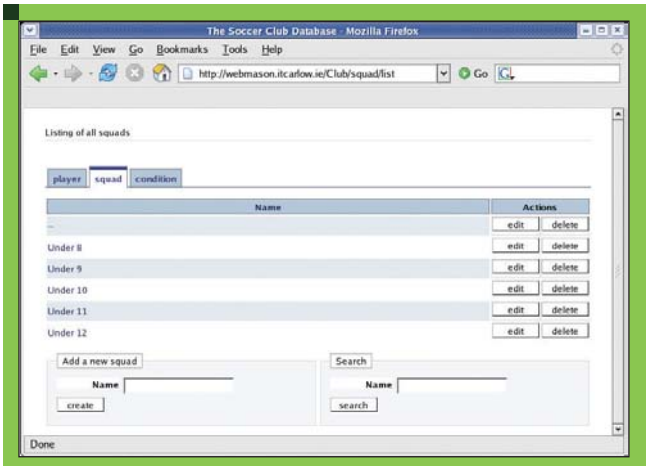
```
mkdir -p /etc/httpd/lib/perl/
cp ClubDB.pm /etc/httpd/lib/perl/
```

8. lépés: Rajta!

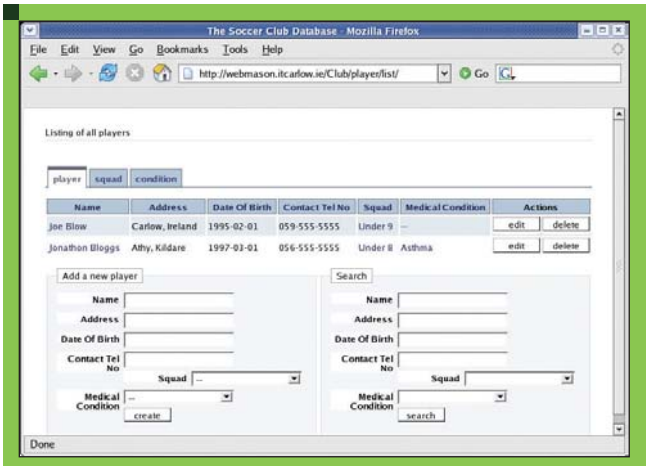
A ***Maypole*** alkalmazás használatba vétele előtt indítsuk újra az ***Apache***-ot:

```
service httpd restart
```

Beírtam a ***http://webmason.itcarlow.ie/Club/*** címet a ***Firefox*** címsorába, aminek hatására az *1. ábrán* láthatók jelentek meg. Jobb, mint a semmi, de nem tökéletesen az, amire számítottam.



3. ábra A Maypole felülete a squad táblához



4. ábra A Maypole felülete a player táblához

Ha valakinek nem volna nyilvánvaló: csinos *CSS* kimenetet vártam, nem nyers *HTML*-t. A hiba javításához megkerestem a 6. lépésben a webkiszolgáló alá másolt alapértelmezett sablonfájlokat. Ezek módosításával úgy tudjuk megváltoztatni alkalmazásunk megjelenését, hogy annak forráskódjába nem kell belenyúlnunk. Remélem, mindenki felfigyelt az utolsó mondatomra! A lényeg, hogy az alkalmazás megjelenését a *CSS* sablonok határozzák meg, működésének módját a kód adja, míg az adatokat a *MySQL* szolgáltatja. A feladatok elkülönítésével rendkívül termelékeny környezetet lehet teremteni, ugyanis az alkalmazás adott részének megváltoztatása semmilyen hatással nem lehet a többire.

A sablonok a *factory* nevű alkönyvtárban találhatók, ezt az alkalmazás *URL*-je – ez esetben *Club*/ – alatt találjuk. A *factory (gyári)* sablonok a *Maypole* alapértelmezett sablonjai, és egészen addig használatban maradnak, míg egy másik, *custom (egyedi)* nevű könyvtárba helyezett sablonokkal felül nem bíráljuk őket.

Miután létrehoztam a *custom* könyvtárat a *Club*/ *URL* alatt, a *factory* könyvtárból átmásoltam a *customba* a *fejrész (header)* fájlt, majd a vi segítségével átírtam. A */maypole.css* nevét */club.css*-re változtattam, valamint az A *„poorly configured" (rosszul beállított)* üzenet helyett is valami méltóbbat adtam meg. A *címlap (frontpage)* fájlt szintén átmásoltam a *factoryból* a *customba*, és megfelelőbb alkalmazásleírást adtam meg benne. Ezután módosítottam a *custom/frontpage* fájl hivatkozás címkéjét, az alapértelmezett *„List by player" (Listázás játékos szerint)* szöveget *„Work with the player data" (A játékosok adatainak kezelése)* szövegre változtattam. Miután végeztem a módosításokkal, rákattintottam a *Firefox* frissítés gombjára. A *2. ábrán* látható eredmény – azt hiszem, egyetértünk – jóval szebb volt.

A menüpontok bármelyikére kattintva csinos, a *3.* és a *4. ábrán* láthatóhoz hasonló beviteli képernyőt kapunk.

A 4. ábrán a két kitalált játékos adatainak bevitele utáni tartalom látható. Vegyük észre az „ingyen" kapott lehetőségeket. A táblákhoz tartozó fülek a képernyő tetején láthatók. A táblák adatait egyszerűen a megfelelő fülekre kattintva érhetjük el. Minden sorhoz tartozik egy szerkesztést és egy törlést lehetővé tévő gomb. Bármelyik oszlop fejlécére kattintva az adott oszlop adatai szerint rendezhetjük a listát, keresni vagy játékosokat hozzáadni pedig az e célra szolgáló űrlapokkal tudunk. A játékosok csapatát vagy egészségi állapotát a megfelelő mezőre kattintva, legördülő listából választhatjuk ki, ez az apró varázslat annak köszönhető, hogy korábban a has_a utasítással előírtuk, hogy minden játékosnak van csapata és egészségi állapota. Alapesetben a *Maypole* a hivatkozott tábla *name (név)* adatoszlopában szereplő adatok alapján állítja elő a legördülő listák tartalmát.

Mit is kaptunk? Teljes értékű webes felületet a háttérben futó adatbázishoz – mindössze nyolc egyszerű lépéssel. A *Maypole* még újdonságszámba megy, ám máris aktív közösség épült köré. A levelezési lista nemrég oszlott kétfelé, az egyik a fejlesztők, a másik a felhasználók eszmecseréjét szolgálja; weblapja pedig a *perl.org*-ra került.

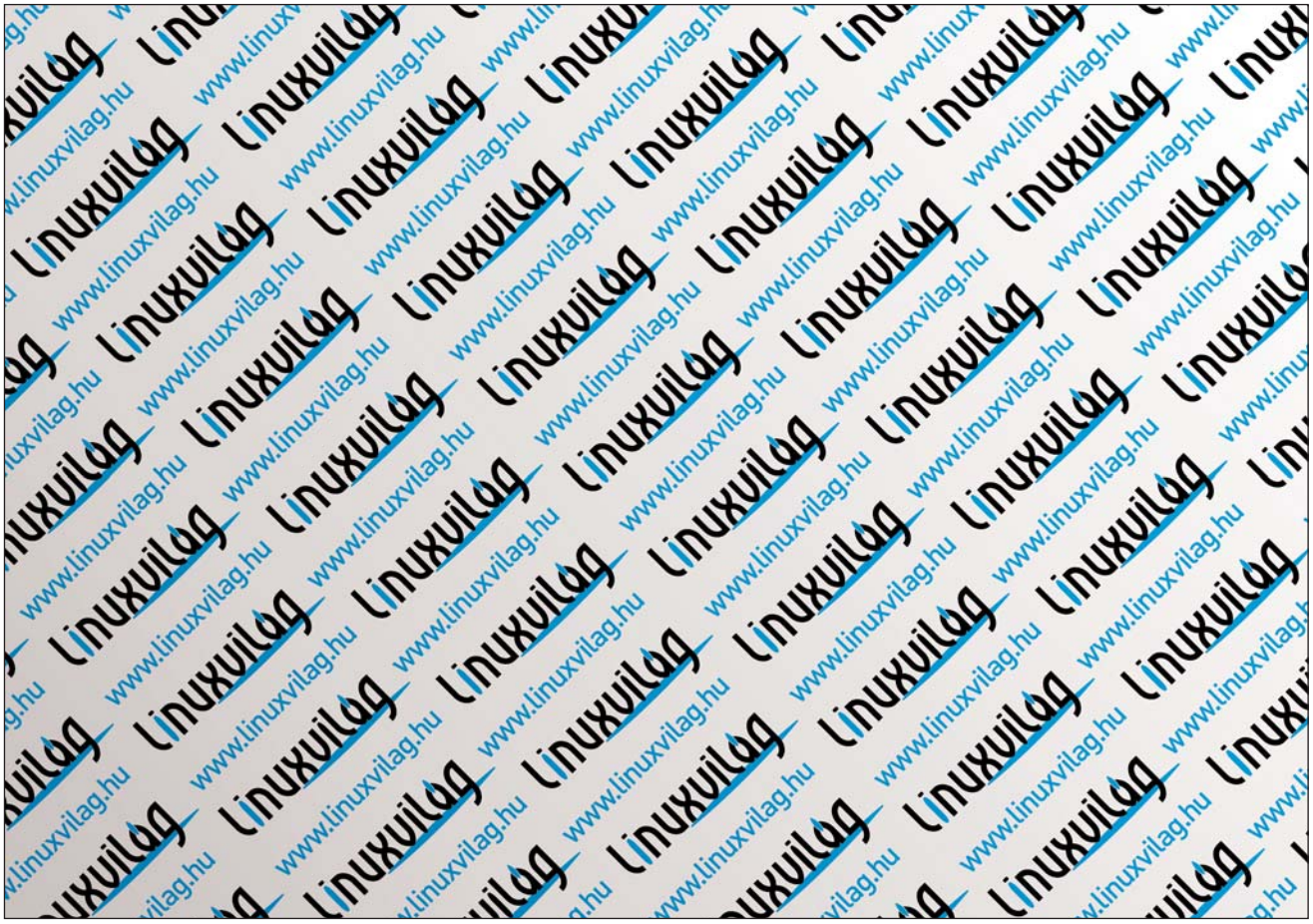
Remélem, sikeresen szemléltettem: a *Maypole* használata – ha túlestünk a telepítésen – gyerekjáték. A webes alkalmazásokhoz szükséges elemek túlnyomó részét készen kapjuk kézhez, ezeket szükség szerint tudjuk bővíteni. A *Maypole* nemcsak *MySQL*-lel, hanem bármely más *SQL* alapú adatbázis-kezelő rendszerrel is használható. További tájékoztatást a *Maypole* honlapján szereplő leírásokban lehet találni.

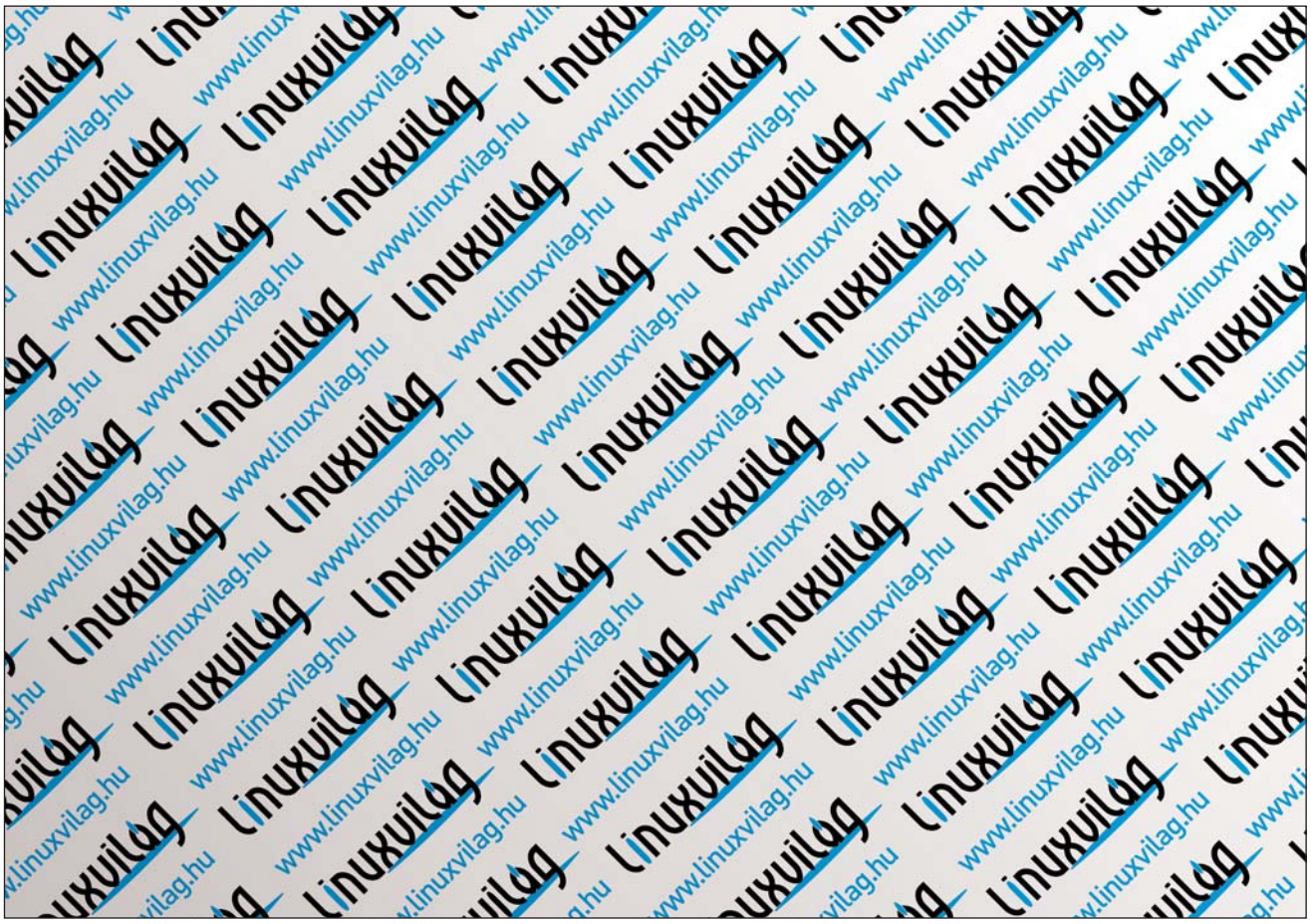
Linux Journal 2005. március, 131. szám

Paul Barry (paul.barry@itcarlow.ie)
Az írországi Carlow Műszaki Intézetében oktat. Az általa tartott tárgyakról, könyveiről és egyéb írásairól a glasnost.itcarlow.ie/~barryp weboldalon lehet bővebb információkat találni.

KAPCSOLÓDÓ CÍMEK

- maypole.perl.org
- search.cpan.org
- www.mysql.com





Adatbázis másolás Slony-I segítségével

Akár a magas elérhetőség érdekében, akár mentésként vagy leállás nélküli verziófrissítés miatt van szükségünk másolatkészítésre, ez a rugalmas eszköz mindent szinkronban tart.

Az adatbázis kezelő rendszerek jó néhány éve létfontosságú alkotóelemei a különféle megoldásoknak.
A *PostgreSQL* fejlett, objektum-relációs adatbázis kezelő rendszer, amelyet igen gyakran használnak összetett alkalmazások hátttereként. Bár ez az adatbázis kezelő rendszer közismerten stabil, két két nyílt forrású másolatkészítő rendszere az *rserv* és az *ERServer*, komoly problémákkal küzd és helyettesítésre szorul.
Szerencsére mostanában jelent meg egy ilyen helyettesítő rendszer. Nevezetesen *Jan Wieck PostgreSQL*-hez kifejlesztett trigger alapú *Slony-I*, egyetlen mester – több szolga elvű másolatkészítő rendszere. Ez az üzleti kategóriás másolatkészítő megoldás aszinkron elven működik és az adatközpontokban elvárható valamennyi kulcsképességgel rendelkezik. A *Slony-I* legfontosabb felhasználási területei:

- Adatbázis másolatkészítés a központi hivatal adatairól az ágak felé, csökkentve a sávszélesség felhasználást, vagy felgyorsítandó az adatbázis lekérdezéseket.
- Adatbázis másolatkészítés valamennyi példány terhelés-kiegyenlítése érdekében. Különösen hasznos megoldás jelentéskészítések vagy dinamikus weblapok esetében.
- Adatbázis másolatkészítés a magas elérhetőség biztosítása érdekében.
- Forró mentés készenléti kiszolgáló segítségével vagy átállás új *PostgreSQL* verzióra.

Cikkünk végigkíséri az olvasót a *Slony-I* telepítésének lépésein, majd másolatot készítünk a helyi gépen található adatbázisról. Bemutatjuk, miképpen használhatjuk a *Slony-I* rendszerét automatikus hibaelhárításra magas rendelkezésre állású megoldásokban.

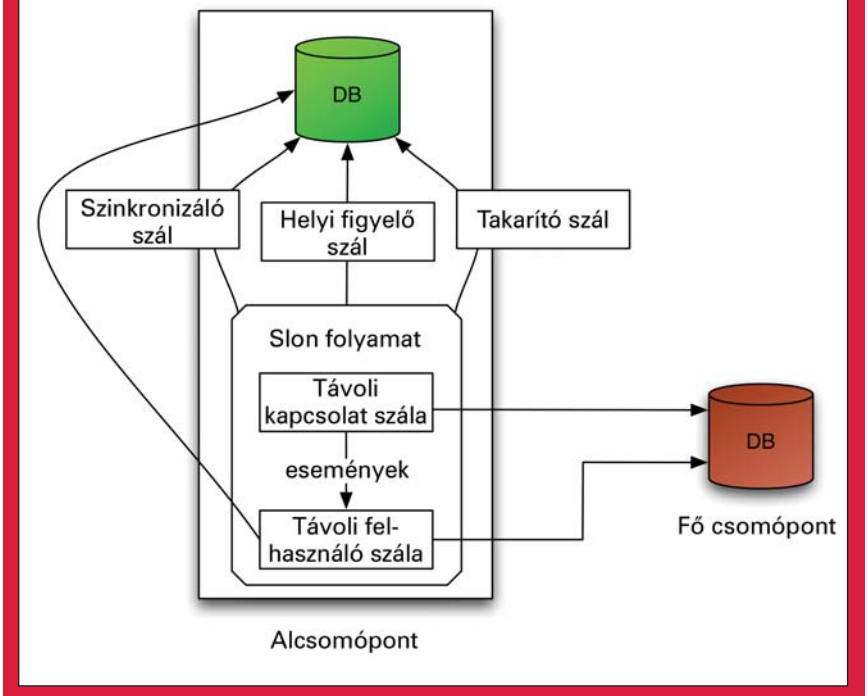
Slony-I telepítése

Amennyiben egy egyszerű adatbázison ki is szeretnénk próbálni a *Slony-I* rendszert először is telepítsük fel forrásból a *PostgreSQL*-t. A *Slony-I* a *PostgreSQL 7.3.2* vagy újabb (7.4. és 8.0) verziókat támogatja és fordítás közben szüksége lesz a *PostgreSQL* forrásfájára. Amennyiben szívesebben használjuk a kedvenc terjesztésünk *PostgreSQL* csomagjait, egyszerűen fordítsuk le őket a forráscsomagból és hagyjuk meg a forrásfát érintetlenül, így fel tudjuk majd használni a Slony-I fordításához. Szerezzük be a legfrissebb *Slony-I* kiadást, azaz az 1.0.5-öst, fordítsuk le és telepítsük.
Ezt a következő parancsokkal tehetjük meg:

```
% tar -zxvf slony1-1.0.5.tar.gz
% cd slony1-1.0.5
% ./configure \
--with-pgsourcetree=/usr/src/redhat/
➥BUILD/postgresql-7.4.5
% make install
```

Példánkban a *Slony-I* `configure` parancsfájlának megadtuk, hogy a */usr/src/redhat/BUILD/postgresql-7.4.5/* helyen kell keresnie a *PostgreSQL* forrást, ugyanis *Red Hat Enterprise Linux* ezt a könyvtárat használja a *PostgreSQL 7.4.5 RPM* állomány fordításakor. Az utolsóparancs lefordítja a *Slony-I*-t és telepíti a következő állományokat:

- *$postgresql_bindir/slonik*: a *Slony-I* adminiszrációs és beállító állománya. A *slonik* egyszerű eszköz és általában héjprogramba ágyazva a *Slony-I* másolatkészítő eszköz beállítására használják. Saját formátummentes parancsnyelve van amelyet a *Slonik Command Summary* dokumentum ismertet részletesen.
- *$postgresql_bindir/slon*: A központi másolatkészítő motor. Ez a többszálú rendszer használja fel a másolatkészítési vázlat információit és kommunikál a többi motorral létrehozva a megoszló másolatkészítő rendszert.
- *$postgresql_libdir/slony1_funcs.so*: C függvények és ravaszok (triggers).
- *$postgresql_libdir/xxid.so*: további adattípusok a tranzakció azonosítók biztonságos tárolásához.
- *$postgresql_datadir/slony1_base.sql*: Másolatkészítési vázlat.
- *$postgresql_datadir/slony1_base.v73.sql*.
- *$postgresql_datadir/slony1_base.v74.sql*.
- *$postgresql_datadir/slony1_funcs.sql*: másolatkészítő függvények.
- *$postgresql_datadir/slony1_funcs.v73.sql*.
- *$postgresql_datadir/slony1_funcs.v74.sql*.
- *$postgresql_datadir/xxid.v73.sql*: A korábban megadott adattípusok betöltését végző parancsfájl.



1. ábra A Slony-I másolatkészítő motor működése mester és szolga adatbázisokkal

egy `contactdb` adatbázist és engedélyezzük a ***plpgsql*** programozási nyelvet a frissen elkészült ***PostgreSQL*** adatbázisunkhoz a következő parancsok kiadásával:

```
% su - postgres
% createuser --pwprompt
➥contactuser
Enter password for user
➥"contactuser": (specify
➥a password)
Enter it again:
Shall the new user be allowed to
➥create databases?
(y/ n) y
Shall the new user be allowed to
➥create more new
users? (y/ n) n
% createdb -O contactuser
➥contactdb
% createlang -U postgres -h
➥localhost plpgsql \
contactdb
```


A ***$postgresql_bindir*** általában a */usr/bin/*, a ***$postgresql_libdir*** a */usr/lib/pgsql/* és a ***$postgresql_datadir*** pedig a */usr/share/pgsql/* könyvtárra mutat. Az `pg_config --configure` parancs használatával megjeleníthetjük a ***PostgreSQL*** fordításakor használt paramétereket és megtudhatjuk milyen megoldást használ a terjesztésünk. A teljes körű ***PostgreSQL*** másolatkészítő motorunk létrehozásához mindössze ezekre a fájlokra van szükségünk.

Amint azt az *1. ábrából* kitalálhatjuk, a ***Slony-I*** központi másolatkészítő motorja a ***slon*** jónéhány szálat használ. A szinkronizációs szál előre beállítható időtartamonként ellenőrzi, hogy van-e valamilyen másolható adatbázis aktivitás, és ha igen, akkor `SYNC` eseményt generál. A helyi figyelő szálak érzékelik az új beállítási eseményeket és ennek megfelelően módosítják a fürt beállításait valamint a ***slon*** folyamat a memóriában található paramétereit.

Mint a neve is mutatja, a ***tisztító (cleanup)*** folyamat a ***Slony-I*** vázlat karbantartását végzi, eltávolítva a régi eseményeket és kiürítve a táblákat. A távoli figyelő szál felkapcsolódik távoli csomópontok adatbázisához, hogy annak eseményküldőjétől eseményeket fogadjon. Amikor eseményt vagy jóváhagyást észlel, kiválasztja a megfelelő információt és a távoli dolgozók szálainak belső üzenetsorába tölti azokat. A másolatkészítési adat, tranzakciócsoportokból áll. A távoli dolgozó szál (minden csomóponton egy-egy) végzi a tényleges másolatkészítést, esemény tárolást és a visszaigazolások elküldését. A szolga minden időpillanatban pontosan tudja, milyen tranzakciókat dolgozott már fel.

Kisebb adatbázis másolata

Először is készítünk egy adatbázist amit majd másolhatunk. Az adatbázis egyetlen táblát és egy szekvenciát tartalmaz. készítsünk egy `contactuser` nevű felhasználót,

Ezek után létrehozzuk a szekvenciát és a táblát a másolandó adatbázisunkban majd némi információt szúrunk a táblába:

```
% psql -U contactuser contactdb
contactdb=> create sequence contact_seq start with 1;
contactdb=> create table contact (
 cid         int4 primary key,
 name        varchar(50),
 address     varchar(255),
 phonenumber varchar(15)
);
contactdb=> insert into contact (cid, name, address,
phonenumber) values ((select
➥nextval('contact_seq')),
'Joe', '1 Foo Street', '(592) 471-8271');
contactdb=> insert into contact (cid, name, address,
phonenumber) values ((select
nextval('contact_seq')),
'Robert', '4 Bar Roard', '(515) 821-3831');
contactdb=> \q
```

Az egyszerűség kedvéért készítsünk még egy adatbázist ugyanezen a rendszeren ahol az első (`contactdb`) adatbázisunk adatait fogjuk duplikálni. A következő parancsok segítségével hozzuk létre az adatbázist, vegyük fel a ***plpgsql*** programozási nyelv támogatást és vigyük fel az `contactdb` adatbázis vázlatát adatok nélkül:

```
% su - postgres
% createdb -O contactuser contactdb_slave
% createlang -U postgres -h localhost plpgsql \
contactdb_slave
```

1. lista cluster_setup.sh

```
#!/bin/sh
CLUSTER=sql_cluster
DB1=contactdb
DB2=contactdb_slave
H1=localhost
H2=localhost
U=postgres
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$H1
➥user=$U';
node 2 admin conninfo = 'dbname=$DB2 host=$H2
➥user=$U';
init cluster (id = 1, comment = 'Node 1');
create set (id = 1, origin = 1,
       comment = 'contact table');
set add table (set id = 1, origin = 1, id = 1,
        full qualified name = 'public.contact',
        comment = 'Table contact');
set add sequence (set id = 1, origin = 1,
➥id = 2,
        full qualified name =
        ➥'public.contact_seq',
        comment = 'Sequence contact_seq');
store node (id = 2, comment = 'Node 2');
store path (server = 1, client = 2,
      conninfo = 'dbname=$DB1 host=$H1
      ➥user=$U');
store path (server = 2, client = 1,
           conninfo = 'dbname=$DB2 host=$H2
           ➥user=$U');
store listen (origin = 1, provider = 1,
➥receiver = 2);
store listen (origin = 2, provider = 2,
➥receiver = 1);
```

```
% pg_dump -s -U postgres -h localhost contactdb | \
psql -U postgres -h localhost contactdb_slave
```

Miután létrehoztuk az adatbázisokat, már készen állunk az adatbázis fürt létrehozására, ahol egy mesterhez mindössze egyetlen szolga tartozik majd. Hozzuk létre a `cluster_setup.sh` parancsfájlt és futtassuk le.
A `cluster_setup.sh` parancsfájl tartalmát az *1. listában* mutatjuk be
Az 1. listában olvasható első `slonik` parancs (a `cluster name`) adja meg azt a névteret, ahol az összes ***Slony-I*** jellegű függvény, eljárás, tábla és szekvencia tárolódik. ***Slony-I*** rendszerében a csomópont egy adatbázis és `slon` folyamatok együttese, a fürt pedig csomópontok csoportja, amelyeket ösvények (`path`) kötnek össze. Megadjuk az 1-es és a 2-es csomópont kapcsolati adatait, majd az első csoportot alaphelyzetbe állítjuk (`init cluster`). Ezek után a parancsfájl létrehoz egy másolandó új készletet,

2. lista subscribe.sh

```
#!/bin/sh
CLUSTER=sql_cluster
DB1=contactdb
DB2=contactdb_slave
H1=localhost
H2=localhost
U=postgres
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$H1
➥user=$U';
node 2 admin conninfo = 'dbname=$DB2 host=$H2
➥user=$U';
subscribe set (id = 1, provider = 1, receiver
➥= 2, forward = yes);
```

amely tulajdonképpen a `public.contact` táblát és a `public.contact_seq` szekvenciát tartalmazza.
A készlet létrehozása után a parancsfájl felveszi a `contact` táblát és a `contact_seq` szekvenciát. A `store node` parancs segítségével alaphelyzetbe állítjuk a második csomópontot (`id = 2`) és felvesszük a fürtbe (`sql_cluster`). Ezek után a parancsfájl meghatározza a 2-es csomópont másolatkészítő rendszere milyen módon csatlakozik az 1-es csomóponthoz. Végül, a szkript mindkét csomópontot utasítja, hogy figyeljék a fürt többi elemének eseményeit (`store listen`).
A parancsfájl végrehajtását követően indítsuk el a ***slon*** másolatkészítő folyamatokat. A mester és a szolga csomóponton is szükségünk lesz egy-egy ***slon*** folyamatra. Példánkban a két szükséges folyamatot ugyanazon a rendszeren indítottuk el. Replikáció közben a ***slon*** folyamatoknak állandóan futniuk kell. Ha valamilyen okból le kellene állítanunk őket, egyszerű újraindítással onnan folytatják a munkát ahol abbahagyták. A másolatkészítő motor indítását a következő parancsok kiadásával végezzük:

```
% slon sql_cluster "dbname=contactdb
➥user=postgres" &
% slon sql_cluster "dbname=contactdb_slave
➥user=postgres" &
```

Következő lépésként fel kell iratkoznunk a frissen elkészített készletre. A készletre feliratkozva a második, azaz a feliratkozó csomópont, elkezdi az első csomópont `contact` táblájának és a `contact_seq` szekvenciájának másolását.
A *2. listában* a feliratkozó parancsfájl tartalmát olvashatjuk.
Az 1. listához hasonlóan a ***subscribe.sh*** is a fürt névtér és a két csomópont kapcsolati adatainak megadásával kezdődik. Ezek után a `subscribe set` parancs hatására az első csomópont ***slon*** folyamatokon keresztül elkezdi az egyetlen táblából és szekvenciából álló készlet másolását a második csomópontra.

3. lista compare.sh

```
#!/bin/sh
CLUSTER=sql_cluster
DB1=contactdb
DB2=contactdb_slave
H1=localhost
H2=localhost
U=postgres
echo -n "Comparing the databases..."
psql -U $U -h $H1 $DB1 >dump.tmp.1.$$
➥<<_EOF_
        select 'contact'::text, cid, name,
        ➥address,
          phonenumber from contact order by
          ➥cid;
_EOF_
psql -U $U -h $H2 $DB2 >dump.tmp.2.$$ <<_EOF_
        select 'contact'::text, cid, name,
        ➥address,
          phonenumber from contact order by
          ➥cid;
_EOF_
if diff dump.tmp.1.$$ dump.tmp.2.$$ >dump.diff ;
➥then
        echo -e "\nSuccess! Databases are
        ➥identical."
        rm dump.diff
else
        echo -e "\nFAILED - see dump.diff."
fi
rm dump.tmp.?.$$
```

Miután a ***subscribe.sh*** parancsfájl lefutott, csatlakozzunk a `contactdb_slave` adatbázishoz és vizsgáljuk meg a `contact` tábla tartalmát. Bármely időpillanatban azt kell látnunk, hogy az adatok pontosan lemásolódtak:

```
% psql -U contactuser contactdb_slave
contactdb_slave=> select * from contact;
 cid |  name  |   address    |  phonenumber
-----+--------+--------------+----------------
   1 | Joe    | 1 Foo Street | (592) 471-8271
   2 | Robert | 4 Bar Roard  | (515) 821-3831
```

Most csatlakozzunk a `/contactdb/` adatbázishoz és szúrjunk be egy sort:

```
% psql -U contact contactdb
contactdb=> begin; insert into contact
➥(cid, name,
 address, phonenumber) values
 ((select nextval('contact_seq')), 'William',
 '81 Zot Street', '(918) 817-6381'); commit;
```

Ha ismét megvizsgáljuk a `contacdb_slave` adatbázis `contact` tábláját, láthatjuk, hogy a sor itt is megjelenik. Most töröljünk egy sort a `/contactdb/` adatbázisból:

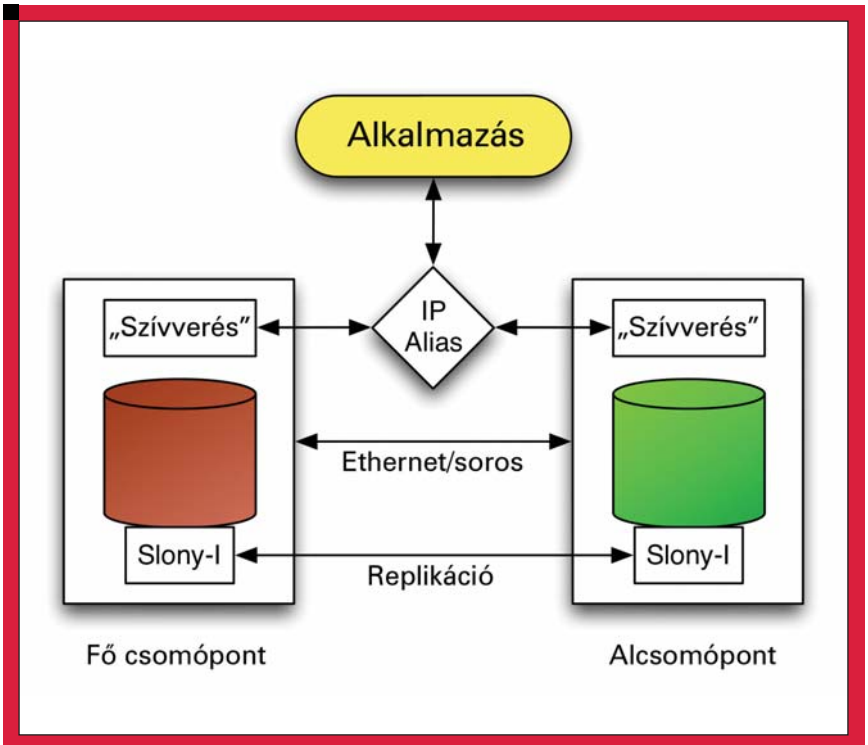
```
contactdb=> begin; delete from contact
 where cid = 2; commit;
```

Akárcsak az előbb, ha megnézzük a `contactdb_slave` adatbázis `contact` tábláját, láthatjuk, hogy a változások hűen követve a sor a szolga csomópontról is törlődött.
A `contactdb` és a `contactdb_slave` kézi összehasonlítgatása helyett egy egyszerű parancsfájl segítségével könnyen automatizálhatjuk ezt a folyamatot. A parancsfájl forrását a *3. listában* olvashatjuk. A parancsfájlt adott időközönként lefuttathatjuk és ellenőrizhetjük, hogy valamennyi csomópont szinkronban fut-e és figyelmeztethetjük az adminisztrátort amennyiben nem ez a helyzet.
Igaz, egyazon rendszeren nincs igazán sok értelme az adatbázis másolatok készítésének, azonban példánk jól szemlélteti milyen egyszerű kialakítani egy ilyen rendszert.
Amennyiben külön gépeken futó csomópontokkal szeretnénk kipróbálni a másolatkészítést, egyszerűen csak a `DB2`, `H1` és `H2` környezetei változókat kell módosítanunk az ***1-3 listákban***. Általában a `DB2` és a `DB1` azonos értékre mutatnak, íg az alkalmazás mindig azonos adatbázisnevet használ. A gazdagép (host) környezetei változókat a két csomópont teljes tartományvére kell állítani. Előfordulhat, hogy arról is meg szeretnénk győződni, hogy a slon folyamatok mindkét gépen futnak. Végül, nem árt, ha valamennyi gép óráját egyeztetjük az ntpd vagy hasonló eszköz segítségével.
Később, ha további táblákat vagy szekvenciákat szeretnénk felvenni az eredeti másolatkészítő rendszerbe, létrehozunk egy új készletet majd beolvasztjuk a `merge set slonik` parancs segítségével. Ezen kívül a készlet megosztásához használhatjuk a `set move table` és a `set move sequence` parancsokat. A ***Slonik Command*** összefoglalóban további információkat találunk ezekről a módszerekről.

Hibakezelés

A mester csomóponton probléma merül fel, például operációs rendszer összeomlás vagy alkatrész meghibásodás miatt, a ***Slony-I*** nem nyújt olyan lehetőséget, amellyel az egyik szolga gépet mesterré léptethetnénk elő. Ez gondot jelenthet, hiszen emberi beavatkozás szükséges a csomópont előléptetéséhez, így magas rendelkezésre állású adatbázisokat követelő szolgáltatásokat nem alapozhatunk rá. Szerencsére számos megoldás létezik amelyeket a ***Slony-I*** rendszerrel ötvözve automatikus hibakezelő rendszert kapunk. Az egyik ezek közül a ***Linux-HA Heartbeat***.
A *2. ábrán* egy mester és egy szolga csomópontot láthatunk ***Ethernet*** és soros kapcsolattal összekötve. Ebben a felállásban a ***Heartbeat*** ezen a két kapcsolaton ellenőrzi a kiszolgáló rendelkezésre állását. Az alkalmazás ***IP*** álnéven keresztül használja a ***PostgreSQL*** adatbázis szolgáltatást, amelyet a ***Heartbeat*** kezdetben a mester csomópontra állít. Amennyiben a ***Heartbeat*** a mester csomópont leállását érzékeli, az ***IP*** álnevet a szolga csomópontra állítja majd végrehajt egy ***slonik*** parancsfájlt, amely a szolga gépet mesterré lépteti elő.

A parancsfájl elég egyszerű. A *4. listában* olvashatjuk a ***slave.example.com*** címen futó szolga csomópontot előléptető parancsfájlt, amely ezáltal átveszi a ***master.example.com*** csomópont által eddig nyújtott adatbázis szolgáltatási feladatokat.
A 4. listában a `failover` ***Slonik*** parancs jelzi, hogy az `id = 1` jelzésű csomópont, azaz a ***master.example.com*** címen futó gép leállt, és a `id = 2` jelzésű csomópont veszi át a kimaradt gép valamennyi készletét. A második parancs a `drop node` feladata az `id = 1` jelzésű csomópont teljes eltávolítása a másolatkészítő rendszerből. Idővel persze szeretnénk a kimaradt csomópontot visszarakni a fürtbe. Ehhez szolgaként kell beállítanunk, majd utasítanunk ***Slony-I*** rendszert, hogy másolja le a hiányzó információkat. Végül az eredeti mester rendszerre visszaváltatáshoz először zároljuk a készletet (`lock set`), kivárjuk valamennyi esemény befejeződését (`wait for event`), átmozgatjuk a készletet az új forráshelyre (`move set`) majd megvárjuk amíg az utolsó parancs kiadásáról megérkezik a visszaigazolás. Ezekről a parancsokról a ***Slonik*** parancsösszefoglalóban találunk további információkat.



2. ábra Heartbeat a szolga csomópontra változtatja az IP álnevet a mester leállása esetén

Összefoglalás

A ***Slony-I*** segítségével viszonylag egyszerű az adatbázis másolatkészítés. A ***Linux-HA Heartbeat*** rendszerrel kombinálva magas rendelkezésre állást biztosíthatunk az adatbázis szolgáltatásainknak. Bár a ***Slony-I*** és a ***Linux HA-Heartbeat*** párosítása vonzó lehetőség, fontos megjegyezni, hogy nem helyettesítheti a kiszolgálóik jó minőségű alkatrészeit.

Igaz, vannak kisebb hiányosságai, hiszen nem tud vázlat változásokat vagy nagy objektumokat másolni, a ***Slony-I*** ma kiváló alternatíva lehet mind az ***rserv***, mind az ***ERServer*** rendszer helyett, sőt tulajdonképpen a ***PostgreSQL*** adatbázisok másolásához ez a leginkább javasolható megoldás.
A szinkronizált több mesteres másolatkészítést is támogató ***Slony-II*** már a tervezőasztalon fekszik.
Végszóként szeretnék köszönetet mondani ***Jan Wieck***-nek, a ***Slony-I*** szerzőjének e cikk lektorálásáért.

Linux Journal 2005. június, 134. szám

4. lista promote.sh

```
#!/bin/bash
CLUSTER=sql_cluster
H1=master.example.com
H2=slave.example.com
U=postgres
DB1=contactdb
DB2=contactdb
su - postgres -c slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$H1
➥user=$U';
node 2 admin conninfo = 'dbname=$DB2 host=$H2
➥user=$U';
failover (id = 1, backup node = 2);
drop node (id = 1, event node = 2);
```

KAPCSOLÓDÓ CÍMEK

- www.postgresql.org
- www.slony.info
- developer.postgresql.org/~wieck/slony1/download/slony1-1.0.5.tar.g z
- gborg.postgresql.org/project/slony1/genpage.php?slonik_commands
- gborg.postgresql.org/mailman/listinfo/slony1-general
- www.linux-ha.org/heartbeat

Ludovic Marcotte (ludovic@sophos.ca)

A Montreali Egyetem Informatika Karán szerzett mérnöki diplomát. Jelenleg a montreali központú Inverse, Inc., programtervezője.

K3b

CD/DVD írás KDE köntösben

A K3b egy linuxos CD/DVD-író alkalmazás. Segítségével készíthetünk adat-, videó- és zenei lemezeket, törölhetünk és másolhatunk lemezeket, képfájlokat készíthetünk ill. visszaírhatjuk ezeket. Összességében majd minden CD/DVD-lemezekkel kapcsolatos funkciót tartalmazó, könnyedén kezelhető alkalmazás.

Ismerkedés

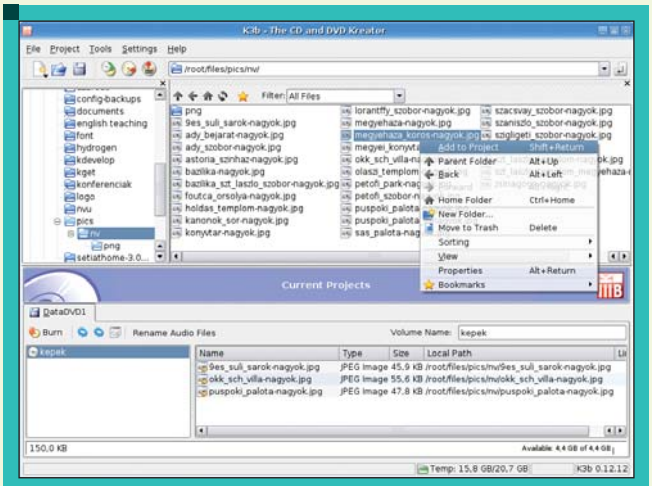
A *K3b CD/DVD*-lemezek kezelésének esetenként kényelmetlen megoldásaira nyújt egy vizuálisan és funkcionalitásában kielégítően sokoldalú alternatívát. Hasonló alkalmazások *Linux* alatt pl. a *GnomeBaker* vagy a *NeroLinux* (utóbbi kereskedelmi forgalomban kapható). *Linux* operációs rendszer alatt futó, *GPL* licenszű, ingyenes és szabad szoftverről van szó, amely noha *KDE* grafikus megjelenítési könyvtárakra épül, bármilyen *Linux*-os grafikus asztali környezet alatt használható. A legtöbb disztribúcióban *k3b* név alatt minden csomagkezelőben megtalálható.

A *K3b* úgy épül fel, hogy már létező, funkcionalitásban bizonyított *CD/DVD*-kezelő parancssori alkalmazásokra épül, ezek funkcióit használja fel a lemezek készítésében, másolásában, stb. Természetesen ezek használatáról, paraméterezéséről nem kell semmit sem tudnia a felhasználónak, mert a *K3b* egy olyan felhasználói felület mögé rejti ezeket,

1. ábra A K3b felülete indítás után

amely könnyűvé teszi különféle lemez-kezelési projektek létrehozását és folyamatos grafikus visszajelzést biztosít az írási/másolási/stb. Folyamatokról. Ezek az alkalmazások, amelyek a *K3b* mögött a tényleges munkát végzik, a következők:

cdrdao: audio/adat *CD*-k,
cdrtools (*cdrecord , mkisofs, readcd*): *CD/CDRW* írás
dvd+rw-format, growisofs, libdvdcss: *DVD-k, DVD+[-]R(W)*-k kezelése
normalize : audio hangerő normalizálás
sox: audió konverziók
transcode (ill. *tccat, tcdecode, tcextract, tcprobe, tcscan*): videóformátum-átkódolások
vcdxbuild, vcdrip: (S)VCD készítés



2. ábra A K3b felülete adat-lemez (itt DVD) projekt létrehozásakor

Ahhoz, hogy a *K3b* menüiben elérhető összes funkció működjön és minden funkciót használni tudjunk, ezeknek az alkalmazásoknak elérhetőknek kell lenniük rendszerünkön. A *K3b* telepítésekor csak a legszükségesebb csomagok települnek fel (ezek leginkább zeneformátum-kezelő könyvtárak), a többit nekünk kell utólag telepítenünk. Ez nem jelenthet gondot, mivel gyakorlatilag minden disztribúcióban megtalálható alkalmazásokról van szó. A *K3b* teljes képességlistája igen hosszú. Minden elemére nem is térünk ki, a fontosabb funkciókra fogunk koncentrálni, szót ejtve a kisebb, de sokszor ugyanolyan fontos elemekről is.

Beállítások

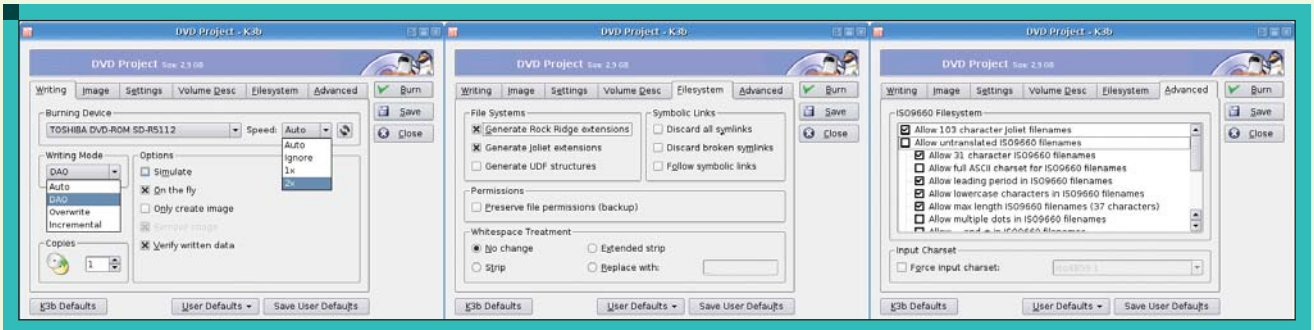
Első indításkor a *K3b Setup* dialógusával találkozunk (a későbbiekben a *Settings->K3b Setup* menüpontban találhatjuk). Itt azt állíthatjuk be, hogy milyen felhasználói csoportok kapjanak *CD/DVD*-írási jogosultságot, illetve azokat a könyvtárakat, ahol a *K3b* keresse a fentebb említett alkalmazásokat. A könyvtárak megadása után ezeket a *K3b* automatikusan megtalálja és többet nem is kell ezekkel foglalkoznunk.
A *K3b* alapértelmezett felülete az *1. ábrán* látható. További beállításokat az alkalmazás elindítása után a *Settings->Configure K3b* menüpontban eszközölhetünk. Itt beállíthatunk alkalmazás-témákat (a *K3b* felületének külalakja), visszajelzéshangokat (pl. az írási folyamat végéről, hibákról, stb.), átmeneti könyvtárakat, *CDDB* információkat (ahonnan audio *CD*-k esetén automatikusan letölthetők a lemez/sáv-információk).

Adatlemezek (CD/DVD) készítése

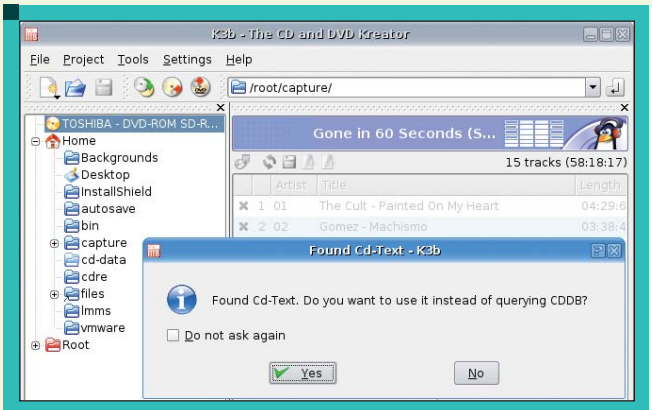
Adatlemezeket, amelyek az általunk összeállított fájlok gyűjteményéből áll össze, a *New Data CD/DVD Project* kiválasztásával hozhatunk létre (fájl menüben, vagy az eszköztáron). Ilyenkor a *K3b* felülete a *2. ábrán* láthatóra változik. Könnyen megkereshetjük a lemezre mentendő fájljainkat, és hozzáadhatjuk a lemez fájl-listájához. Az ablak státuszsorában folyamatosan láthatjuk mennyi helyünk van még a készülő lemezen.
A lemez tartalmának összeállítása után a *Burn* (*Project->Burn* menüpont, az eszköztár *Burn* eleme, vagy *CTRL-B* lenyomásával) indítja a lemez megírását. Ekkor, az írás megkezdése előtt az írás számos paraméterét beállíthatjuk (*3. ábra*). Ahhoz hogy jogosultságok megmaradhassanak és *Windows* operációs rendszereken is olvashatók legyenek a lemezek a *Rock ridge* és a *Joliet* kiterjesztéseket kell alkalmaznunk, ezen kívül számos paraméterhez férhetünk hozzá az *Advanced* fülön.
Természetesen újraírható lemezek törlésére/formázására is van lehetőségünk, a *Tools->Erase CD-RW* ill. *Tools->Format DVD+/-RW* menüpontokban.

Audio CD-k készítése és rip-elése

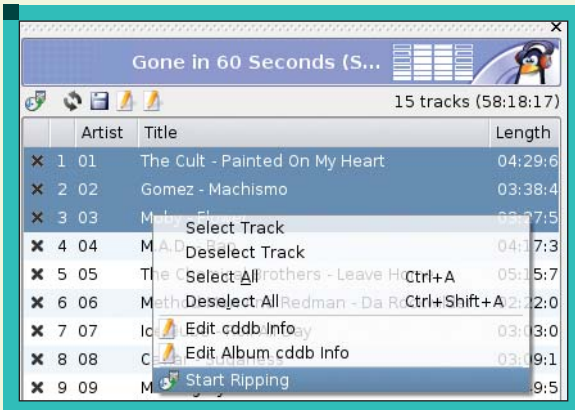
Audió *CD*-ket a következőképpen *rip*-elhetünk, azaz készíthetünk az audió lemez sávjairól mentéseket különféle fájlformátumokba. Helyezzük az audió *CD*-t a meghajtóba, majd a *K3b*-ben kattintsunk a meghajtóra, amelyikben a lemez található. A *K3b* felismeri a lemezt, és ha talál a lemezen *CD Text*.-es sávleírásokat, akkor felajánlja, hogy azt használjuk vagy kérdezzük



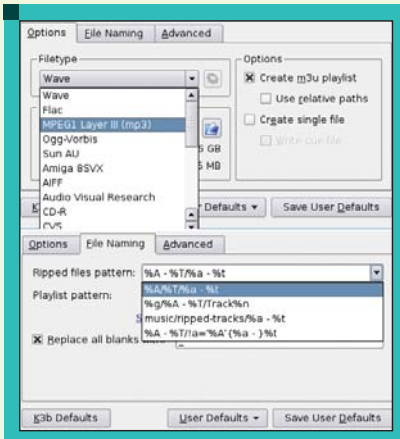
3. ábra Írás paramétereinek beállítása



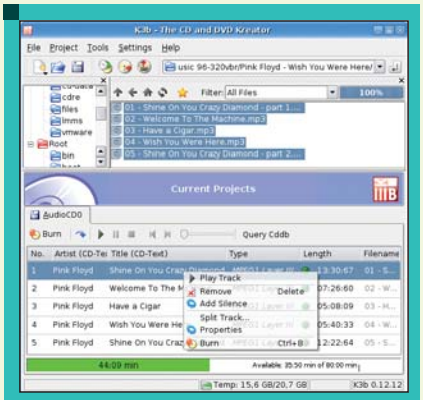
4. ábra Audió CD rip-elésre készülve



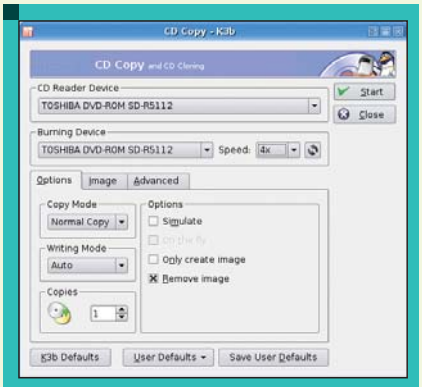
5. ábra Sávok mentése



6. ábra Audió-sáv mentések paraméterei



7. ábra Audió CD készítése zenefájlokból



8. ábra CD másolási dialógusa (Tools.->Copy CD)

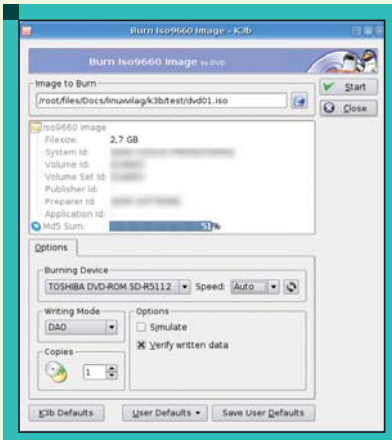
le a *CDDB* adatbázisból a lemez sávadatait (*4. ábra*). Ha nincs *CD Text* információ, akkor is felkínálja a *CDDB*-s letöltést, ami természetesen opcionális.

Az egyes sáv-címekre kattintva helyben szerkeszthetjük azokat. Majd kiválasztva sávokat, jobb klikkel elkezdhetjük a *rip*-elést (*5. ábra*). Ekkor a *rip*-elési paraméterek ablaka ugrik elő, ahol a létrehozandó hangfájl-ok formátumát, másrészt a mentési nevet, útvonalat, *playlist* létrehozást állíthatjuk be (*6. ábra*).

Audió *CD*-k készítése is hasonlóan egyszerű. A *File->New Project->New Audio CD Project* kiválasztása után megkeressük az audió *CD*-re mentendő zenefájlokat és egyszerűen behúzzuk a projektbe (az ablak alsó része, *7. ábra*). A *Burn* kiválasztásakor beállíthatunk még néhány paramétert (pl. *CD Text*.es sávleírásokat adhatunk meg, sebességet állíthatunk be, stb.) és megírhatjuk a lemezt.

Videólemezek készítése videó-fájlokból

Első lépésként hozzunk létre egy *New Video CD Project*-et. Ezután az audió *CD*-hez hasonlóan csak be kell húznunk a kiválasztott videófájlokat a projektbe. *VCD* esetén *MPEG1*, *SVCD* esetén *MPEG2* formátumú videókat adhatunk hozzá a projekthez. Ha a videóink nem ebben a formátumban vannak, akkor át kell konvertálnunk. Sajnos maga a *K3b* nem végzi el ezt a konverziót helyettünk, de ezt könnyen orvosolhatjuk. Ha tegyük fel van egy valamilyen formátumú *.avi* videónk, amiből *VCD*-t szeretnénk készíteni, akkor parancssorban adjuk ki a következő parancsot: *ffmpeg -i videónév.avi -target vcd videónév.mpg*, majd az elkészült fájlt húzzuk be a projektbe *K3b*-ben. *SVCD* esetén *MPEG2*-be kell konvertálnunk, ekkor a szükséges parancs: *ffmpeg -i videónév.avi -target svcd videónév.mpg*, majd az elkészült fájlt adjuk hozzá a projekthez *K3b*-ben. Ezután a *Burn* kiválasztásával rövidesen szabványos *(S)VCD* lemezhez jutunk.



9. ábra Képfájl megnyitása lemezre íráshoz

DVD videólemezek készítésekor a *K3b* a *DVD* lemez *AUDIO_TS* és *VIDEO_TS* könyvtáraiba behúzott fájlokat kiírja a lemezre. Ha nem *MPEG2 VOB* fájlokat adunk hozzá, akkor viszont nem jutunk szabványos, lejátszható *DVD*-hez. Ezért *DVD* videólemezek létrehozásához vagy kézzel elkészítjük a szükséges videó-fájlokat, vagy a *DvdSyler*-hez hasonló alkal-

mazásokkal összeállítunk egy teljes *DVD* videó-lemezt, menükkel, hangokkal, stb. majd az elkészült *DVD* képfájlt *K3b*-vel ráírjuk egy lemezre.

Lemezek másolása, képfájlok készítése és írása

Ha lemezeket szeretnénk másolni, akár *CD* akár *DVD* esetén, a *Tools* menü *Copy CD/DVD* pontját kell választanunk. Ekkor a *8. ábrához* hasonló dialógus jelenik meg, ahol az átmeneti könyvtár, a forrás és a cél kiválasztása után már indulhat is a másolás.
DVD-k másolása esetén ugyanez történik. Videó *DVD*-k esetén a *K3b* nem végez átkódolást a videóban, ezért csak ugyanakkora lemezre tudunk másolni.
Ha videó *DVD*-ről másolatot szeretnénk készíteni, akkor vagy az előbbi módon másoljuk, vagy lemezre másoljuk a *DVD* tartalmát majd a *Tools->Encode Video* menüpontot használva *.avi* formátumba kódolhatjuk át a lementett videót.
Ha nem másik lemezre akarunk másolatot készíteni, hanem csak egy képfájlt (*image*-et) szeretnénk létrehozni a *CD/DVD* lemezről, ennek a menete csupán abban tér el *rip*-elési ill. másolási műveletektől, hogy a *Burn* kiválasztásakor a paraméterablakban ki kell választanunk az *Only Create Image* opciót.
Ha elkészült vagy kézzel létrehozott (*mkisofs, growisofs*), vagy *K3b*-vel (vagy bármilyen más alkalmazással, pl. *DvdStyler*) létrehozott képfájlokat szeretnénk lemezre írni, akkor azt a *Tools->Burn CD/DVD Image* menüponttal tehetjük meg.

Végszó

Összességében a *K3b*-ről elmondható, hogy sikere abban rejlik, hogy az amúgy grafikus felület nélkül is elérhető – de általában körülményes, hosszadalmas, vagy csak egyszerűen kényelmetlen – *CD/DVD*-írással kapcsolatos eszközök használatához egy olyan könnyen kezelhető, letisztult, egyszerű és mégis nagy tudású grafikus felületet biztosít, amely nagyban megkönnyíti és kényelmesebbé teszi a *CD/DVD* írással kapcsolatos feladatokat. Teszi ezt úgy, hogy egyes kereskedelmi forgalomban kapható hasonló tudású alkalmazást is maga mögött hagy átláthatóságban, kezelhetőségben, egyszerűségben, mindezt szabad, nyílt forrású szoftverként. Reméljük sikerült az olvasók érdeklődését legalább egy próba erejéig felkeltenünk. A *K3b*-vel kapcsolatos hírekre, további fejlesztésekre, dokumentációkra találhatnak a *K3b* honlapján.

Kovács Levente
(leventek@gmail.com)

26 éves informatikus- és villamosmérnök. Évek óta használ különféle Linux disztribúciókat. Fontosnak tartja a nyílt forrású szoftverek és fejlesztés előnyeinek megismertetését az emberekkel.

KAPCSOLÓDÓ CÍMEK

- www.k3b.org
- k3b.plainblack.com/about
- dvdstyler.sourceforge.net

Háttéradatok kigyűjtése extract és libextractor segítségével

Ne csak találgassuk a fájlok tulajdonságait kereséskor. Használjunk célzott kibontó bővítményeket a pontos fájladatbázis előállítására.

Modern állományformátumok leíró adatokban tárolják a fájlra vonatkozó információkat. Ezeket a fejlesztéseket az az igény hajtotta, hogy a fájlokat egyszerű állományneveken felül összetettebb módon is lehessen szervezni. Az ilyen háttéradatokkal az a probléma, hogy nem szabványos módon tárolódnak a különféle formátumokban. Ez sajnos megnehezíti az erősen formátumfüggő programok dolgát (ilyenek például a fájlkezelők vagy fájlmegosztó alkalmazások) ha hozzá szeretnének férni ehhez az információhoz. Következésképpen az adatok kibontására rengeteg elemző program látott napvilágot, ilyen jellegű többek közt az ***AVInfo***, az ***id3edit***, a ***jpeginfo*** és a ***Vocoditor***.

Cikkünkben a ***libextractor*** könyvtárat és kibontó eszközét mutatjuk be. A ***libextractor Project*** célja egységes felületet biztosítani különféle fájlformátumok háttéradatainak kisbontásához. A ***libextractor*** programot jelenleg az ***evidence***, a következő ***Enlightenment*** verzió fájlkezelője, valamint a ***GNUnet***, ez az név nélküli ellenőrizhetetlen peer-to-peer fájlmegosztó rendszer használja. Az ***extract*** eszköz a könyvtárhoz tartozó parancssoros kezelőfelület. A ***libextractor*** a ***GNU General Public License*** védelme alá tartozik.

A ***libextractor*** sok hasonlóságot mutat a népszerű ***file*** eszközzel, amely a fájl első bájtját használja a ***MIME*** típus kiderítéséhez. A ***libextractor*** ugyanakkor különbözik is a ***file***-tól, hiszen az egyszerű ***MIME*** típuson kívül sok egyéb információt is megpróbál kigyűjteni. A ***libextractor*** által kikeresett egyéb adatok közt találjuk az állomány létrehozására használt program nevét, az állomány szerzőjét, leírását, az albumcímeket, képméretet vagy éppen a film hosszát.

A ***libextractor*** ezeket az információkat a népszerű formátumokhoz írt értelmezők segítségével szerzi meg. A listán jelenleg ott találjuk az ***MP3***, ***Ogg***, ***Real Media***, ***MPEG***, ***RIFF (avi)***, ***GIF***, ***JPEG***, ***PNG***, ***TIFF***, ***HTML***, ***PDF***, ***PostScript***, ***Zip***, ***OpenOffice.org***, ***StarOffice***, ***Microsoft Office***, ***tar***, ***DVI***, ***man***, ***Deb***, ***ELF***, ***RPM***, ***asf*** formátumokat valamint az olyan általános megoldásokat mint a ***MIME***-típus érzékelése. Persze sok más formátum is létezik, de a népszerűbb formátumok közül csak néhány üzleti formátum nem támogatott.

Az új formátumok támogatása könnyen beépíthető, hiszen a ***libextractor*** bővítmények segítségével gyűjti össze az adatokat. A ***libextractor*** bővítmények tulajdonképpen megosztott könyvtárak amelyek általában egy adott formátum értelmezését végzik. Cikkünk végén azt is bemutatjuk, miképpen tudjuk egy új formátum támogatását könyvtárba szervezni.

A ***libextractor*** összegyűjti a különféle bővítményektől kapott háttéradatokat és az ügyfeleknek osztályozást és karaktersorozatot tartalmazó adat-pár listát ad vissza. Az osztályozási adatot a háttéradatok kategóriákba (cím, szerző, téma, leírás) szervezésére használjuk.

A libextractor telepítése és használata

A ***libextractor***-t legegyszerűbben úgy telepíthetjük, ha a terjesztésekben található bináris csomagot használjuk fel. ***Debian*** alatt az ***extract*** eszköz külön csomag, ***extract*** néven. Ha más alkalmazást szeretnénk ***libextractor***-hoz fordítani a ***libextractor0-devel*** csomagban találjuk a szükséges fejléceket. Amennyiben forrásból szeretnénk lefordítani a ***libextractort***, szokatlanul magas memóriaterületre lesz szükségünk: 256 MB rendszermemória nagyjából a minimum, ugyanis a ***GCC*** körülbelül 200 MB-ot használ az egyik bővítmény fordításánál. Egyébiránt a fordítás a megszokott utat követi, amint azt az *1. listában* láthatjuk.

A ***libextractor*** telepítése után, az ***extract*** eszköz segítségével gyűjthetjük ki a dokumentumok háttéradatait. Alapértelmezés szerint az ***extract*** eszköz adott bővítménykészletet használ, amely a ***libextractor*** jelenlegi verziójának valamennyi fájlformátum-jellegű bővítményét tartalmazza, a ***MIME***-típus érzékelő bővítménnyel együtt. A ***Linux Journal*** honlapról kapott kimenet a *2. listában* olvasható.

A ***BibTeX*** felhasználók számára valószínűleg jól jön a `-b` kapcsoló, amely a dokumentumokból automatikusan háttéradatokkal kiegészített ***BibTeX*** bejegyzéseket készít, amint azt a *3. listában* láthatjuk.

Másik érdekes kapcsoló a `-B LANG`. A kapcsoló az egyik nyelvfüggő, ám formátumfüggetlen bővítményt tölti be. Ezek a bővítmények megpróbálnak egyszerű szöveget keresni a dokumentumban kartersorozatokat összehasonlítva egy szótárállománnyal. Amennyiben furcsának tűnt a 200 MB-os memóriaigény a ***libextractor*** fordításakor, a választ ezek a bővítmények meg. a gyors szótárkeresés érdekében egy ***bloomfilter***-t hoz létre amely gyors valószínűség alapú egyezés keresést tesz lehetővé; a létrejövő adatstruktúrát a *GCC* némiképp nehezen nyeli le.

1. lista A libextractor fordítása 200MB memóriát igényel

```
$ wget
http://ovmj.org/libextractor/download/libextrac
tor-0.4.1.tar.gz
$ tar xvfz libextractor-0.4.1.tar.gz
$ cd libextractor-0.4.1
$ ./configure –prefix=/usr/local
$ make
# make install
```

2. lista Háttéradatgyűjtés HTML-ből

```
$ wget -q http://www.linuxjournal.com/
$ extract index.html
description - The Monthly Magazine of the Linux
Community
keywords - linux, linux journal, magazine
```

3. lista BibTeX bejegyzések készítése igen egyszerű, ha a dokumentum számos hátéradattal rendelkezik

```
$ wget -q
http://www.copyright.gov/legislation/dmca.pdf
$ extract -b ~/dmca.pdf
% BiBTeX file
@misc{ unite2001the_d,
    title = "The Digital Millennium Copyright
    ➥Act
    of 1998",
    author = "United States Copyright Office
    ➥- jmf",
    note = "digital millennium copyright act
    circumvention technological protection
    ➥management
    information online service provider
    ➥liability
    limitation computer maintenance competition
    repair ephemeral recording webcasting
    ➥distance
    education study vessel hull",
    year = "2001",
    month = "10",
    key = "Copyright Office Summary of the
    ➥DMCA",
    pages = "18"http://www.netpincer.hu/
    ➥index.php
}
```

4. lista A libextractor néha még ismeretlen formátum esetén is használható információkat nyerhet ki

```
$ wget -q http://www.bayern.de/HDBG/polges.doc
$ extract -B de polges.doc | head -n 4
unknown - FEE Politische Geschichte Bayerns
Herausgegeben vom Haus der Geschichte als Heft
der zur Geschichte und Kultur Redaktion Manfred
Bearbeitung Otto Copyright Haus der Geschichte
München Gestaltung fürs Internet Rudolf Inhalt
im.
unknown - und das Deutsche Reich.
unknown - und seine.
unknown - Henker im Zeitalter von Reformation
          ➥und Gegenreformation.
```

A `-B` kapcsoló még nem dokumentált vagy támogatott formátumok esetén jöhet jól. A ***printable*** bővítmények általában a dokumentum teljes szövegét sorban kinyomtatják. A ***4. lista Microsoft Word*** dokumentumra alkalmazott ***extract*** futási eredményét mutatja be.
A németül értők láthatják, hogy a kapott anyag elég jó leírása a szövegnek. A támogatott nyelv még a ***Dán (da)***, ***Német (de)***, ***Angol (en)***, ***Spanyol (es)***, ***Olasz (it)*** és a ***Norvég (no)***. Más nyelvek támogatása mindössze a megfelelő karakterkészlettel beillesztett ingyenes szótárkészleten múlik.
A többi kapcsolót az ***extract*** kézikönyvoldala ismerteti; lásd a `man 1 extract` parancsot.

libextractor saját projektjeinkben

Az *5. lista* egy ***libextractor*** könyvtárat használó minimális programot mutat be. A ***minimal.c*** fordításához a `-lextractor` kapcsolót kell megadnunk a ***GCC***-nek. Az `EXTRACTOR_KeywordList` egyszerű láncolt lista, amelyben a kulcsszavak és a kulcsszó típusokat találjuk. A részleteket, bővítmények betölt és a kulcsszólisták kezelését végző további függvények leírását a ***libextractor*** kézikönyvoldalán találjuk: `man 3 libextractor`. A ***Java*** programozóknak érdemes tudni, hogy a ***libextractorral JNI*** segítségével kommunikálni képes ***Java*** osztály is létezik.

Bővítmények készítése

A ***libextractor*** bővítmények készítésekor a legnehezebb feladat az adott formátumhoz tartozó értelmező megírása. Mindazonáltal az eljárás lényegében szinte mindig ugyanaz. A bővítmény könyvtárat ***libextractor_XXX.so*** néven kell elkészíteni, ahol az ***XXX*** a bővítmény által kezelt fájlformátumot jelzi. A könyvtárból exportálni kell a ***libextractor_XXX_extract*** metódust a *6. listában* bemutatott definíció szerint.
A `filename` paraméter a feldolgozandó állomány nevét adja meg. A `data` a fájl általában ***mmap***-olt formában tárolt adataira mutat, végül a size adja meg az állomány méretét. A legtöbb bővítmény (plugin) nem használja fel az állomány nevét és közvetlenül az adatokat elemezi, rögtön az első lépésben ellenőrizve, hogy az állomány fejléce egyezike az adott formátum által igényelt alakkal.

5. lista minimal.c egybegyűjtve mutatja be a legfontosabb libextractor függvényeket

```
#include <extractor.h>
int main(int argc, char * argv[]) {
  EXTRACTOR_ExtractorList * plugins;
  EXTRACTOR_KeywordList   * md_list;
  plugins = EXTRACTOR_loadDefaultLibraries();
  md_list = EXTRACTOR_getKeywords(plugins,
  ➥argv[1]);
  EXTRACTOR_printKeywords(stdout, md_list);
  EXTRACTOR_freeKeywords(md_list);
  EXTRACTOR_removeAll(plugins); /* unload
  ➥plugins */
}
```

A `prev` az eddigi bővítmények által kigyűjtött kulcsszavak listája. A függvénynek a kulcsszólista frissített változatát kell visszaadnia. Amennyiben a formátum nem esik egybe a bővítmény által várttal, a `prev` adódik vissza. A legtöbb bővítmény az `addKeyword` vagy hasonló függvényekkel bővíti a listát (*7. lista*).
Az `addKeyword` leggyakrabban a ***MIME*** típus felvételére használják miután a fájl fajtája már kiderült. Például a ***JPEG-extractor*** (*8.lista*) ellenőrzi a ***JPEG*** fejléc első bájtjait és vagy megszakítja futását vagy úgy dönt az állomány valóban ***JPEG*** típusú. A kódban található `strdup` fontos hiszen a karaktersorozat memóriafoglalása később megszűnhet (általában az `EXTRACTOR_freeKeywords()` hívás során). A támogatott kulcsszó osztályokat, azaz példánkban az `EXTRACTOR_MIMETYPE` típust, az `extractor.h` fejlécállományban találjuk.

Összefoglalás

A ***libextractor*** egyszerű bővíthető ***C*** könyvtár, amely képes összegyűjteni a dokumentumok háttéradatait. Bővítményalapú szerkezete és széleskörű formátumismerete kiemeli a formátumfüggő eszközök közül. A ***libextractor*** ugyanakkor korlátozott a tekintetben, hogy a nem képes módosítani a háttéradatokat, amelyet egyébként a specializálódott eszközök általában képesek megtenni.

Linux Journal 2005. június, 134. szám

6. lista A libextractor bővítményből exportálandó függvény definíciója

```
struct EXTRACTOR_Keywords *
libextractor_XXX_extract
   (char * filename,
    char * data,
    size_t size,
    struct EXTRACTOR_Keywords * prev);
```

7. lista A bővítmények egyszerű láncolt listaként adják vissza a háttéradatokat

```
static void addKeyword
   (struct EXTRACTOR_Keywords ** list,
    char * keyword,
    EXTRACTOR_KeywordType type)
{
  EXTRACTOR_KeywordList * next;
  next = malloc(sizeof(EXTRACTOR_KeywordList));
  next->next = *list;
  next->keyword = keyword;
  next->keywordType = type;
  *list = next;
}
```

8. lista A fájl fejlécének értelmezése után a jpegextractor.c felveszi a MIME típust a listába

```
if ( (data[0] != 0xFF) || (data[1] != 0xD8) )
  return prev; /* not a JPEG */
addKeyword(&prev,
           strdup("image/jpeg"),
           EXTRACTOR_MIMETYPE);
/* ... további értelmező kód ... */
return prev;
```

KAPCSOLÓDÓ CÍMEK

- gnunet.org/libextractor
- getid3.sf.net
- evidence.sf.net
- ovmj.org/GNUnet
- www.wotsit.org
- dublincore.org/documents/dcmi-terms
- dmoz.org/Computers/Software/Typesetting/TeX/BibTeX
- enlightenment.org
- ovmj.org/GNUnet/download/bloomfilter.ps

Christian Grothoff
2000-ben diplomázott a Wuppertali Egyetemen matematika szakon. Jelenleg PhD hallgató a Purdue Egyetemen, ahol a statikus programanalízis és biztonságos peer-to-peer hálózatkezelés témakörét tanulmányozza. 1995 óta Linux felhasználó, több nyílt program fejlesztésében is segédkezett, jelenleg a GNUnet karbantartója és a libextractor magcsapatának tagja. Honlapja a grothoff.org/christian címen érhető el.

Oszd meg és uralkodj – QTparted

Bizonyára mindenki, aki a számítástechnikával egy kicsit mélyebb szinten foglalkozik, hallott már a Partition Magic nevezetű programról. Ez a Microsoft operációs rendszerek alatt futó szoftver a merevlemezeken található partíciók kezelésére ad lehetőséget. Nagyon sok fájlrendszert kezel, viszonylag könnyű és biztonságos a használata, ám két problémánk is adódhat vele. Az egyik, hogy nem tudjuk futtatni kedvenc operációs rendszerünk alatt, a másik pedig, hogy fizetős programról van szó.

Szerencsére *Linux* alatt sem kell nélkülöznünk az ehhez hasonló programokat, ha szétnézünk az interneten találkozhatunk néhány ilyen jellegű alkalmazással. Az egyik ilyen program a *QTParted*, amely a parancssoros `parted` alkalmazásnak a grafikus felülete. Ezen programok segítségével lehetőségünk van partíciókat létrehozni, törölni, mozgatni, akár átméretezni és megformázni, mégpedig néhány kattintással. Kiegészítő könyvtárakkal nem csak linuxszos fájlrendszereket képes kezelni, hanem akár *NTFS*-t is.
A *QTparted* jelenleg a 0.4.5 verziónál tart, amit ez év augusztusában tettek közzé. A tapasztalataim szerint nem a legjobb választás, mivel akadnak gondok a telepítéssel éppúgy, mint a későbbi használattal. Szerencsére elérhető ez előző verzió is, amely sokkal stabilabb és a telepítési módja is sokkal egyszerűbb és tisztább.
Mivel a két verziót nem egyformán kell telepíteni, azért mindkét változatot ismertetni fogom, így mindenki el tudja dönteni, hogy számára melyik a használhatóbb.
Első lépésként be kell szereznünk a programot, amihez menjünk el a ➲ *qtparted.sourceforge.net* oldalra és itt kattintsunk a baloldali menüben a *File list* menüpontot, ami elvezet minket a projekt összes állományát tartalmazó listához. Itt a legelső változattól kezdve a legutolsóig mindegyik le tudjuk tölteni. Találhatunk *Debian* csomagot is, valamint *RPM* változatban is elérhetjük a lefordított programot. Javaslom a forrásból való telepítést, mivel így a rendszerünket saját magunk alakíthatjuk ki. Töltsük le a *qtparted-0.4.4.tar.bz2* vagy a *qtparted-0.4.5.tar.bz2* programot (vagy programokat) és csomagoljuk ki a

```
tar -xjvf qtparted-
➥0.4.4.tar.bz2
```

vagy értelemszerűen a

```
tar -xjvf qtparted-
➥0.4.5.tar.bz2
```

paranccsal. Létrejön egy könyvtár, amibe belépve a fordítás a már megszokott módon történik, legalábbis a 0.4.4-es verzió tekintetében.
Mielőtt azonban ennek nekikezdenénk, szükség lesz néhány egyéb program telepítésére is.
Ezek szinte minden disztribúcióban megtalálhatók, így telepítésük nem okozhat túl nagy gondot. Az alábbiakra lesz szükségünk: *qt*, *qt3*. *qt3-devel*, *qt-devel*, *libjpeg*, *libjpeg-devel*, *e2fsprogs*, *e2fsprogs-devel*, *libparted*, *progreiserfs* (amennyiben *ReiserFS* támogatást szeretnénk, ez a *libreiserfs-progs* csomagnév alatt érhetjük el), *ntfsprogs* (az *NTFS* fájlrendszerű partíciók kezelésére), *ntfsprogs-devel*, illetve a 0.4.5 verzió fordításához az *unsermake* csomagokra. Ezeknek a csomagoknak a telepítésére nem térek ki, mert meghaladná a cikk kereteit, viszont a disztribúciónk csomagkezelőjével egyszerűen elvégezhetjük.
A *SuSE Linux* rendszerrel rendelkezőknek érdemes felvenni, persze amennyiben lehetőségük van rá, a *Novell FTP* oldalának a megfelelő könyvtárszerkezet és, mint telepítési forrás, mivel a *CD*-ről és a *DVD*-től is hiányoznak csomagok, amelyek az *FTP*-n viszont megtalálhatók.
Ha ezek telepítésével végeztünk, fogjunk neki a fordításnak. Első lépésként ellenőrizzük le, hogy létezik-e a `QTDIR` változó és megfelelő-e a tartalma. Ehhez használjuk az

```
env |grep QTDIR
```

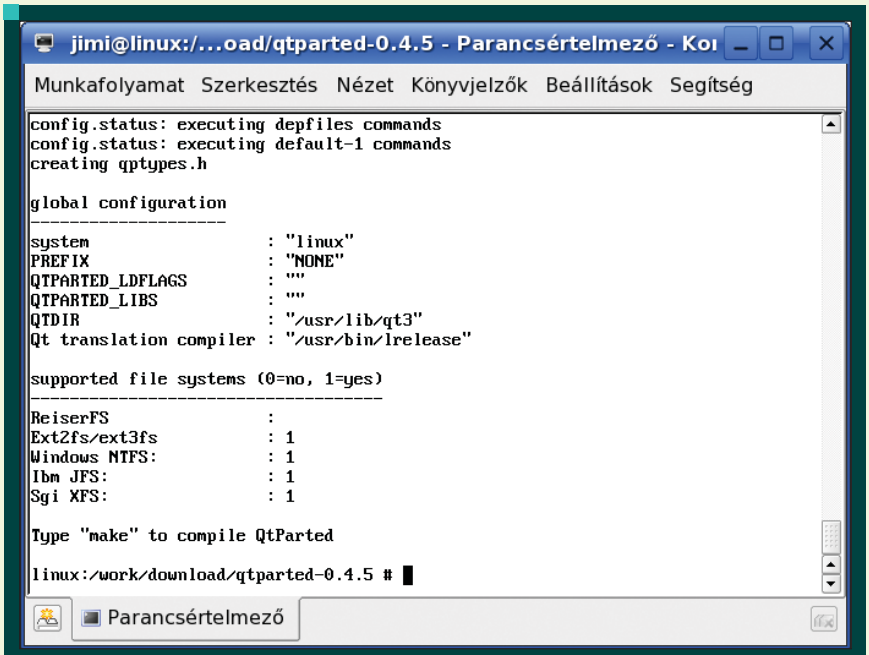
parancsot, aminek eredményeként meg kell jelennie a változónak és a hozzá tartozó útvonalnak.
Ha nem jelenne meg, akkor adjuk meg kézzel az

```
export QTDIR=/usr/lib/qt3
```

paranccsal. Most már tényleg nekiláthatunk a fordításnak. A

```
./configure --help
```

paranccsal kérhetünk segítséget a fordítási opciókról. Tanulmányozzuk át a listát, majd a kapcsolókat



1. ábra A ./configure befejezését záró sorok

használva adjuk ki a

```
./configure
```

parancsot. Általában elegendő lehet ezt kapcsolók nélkül kiadni, hiszen az alapbeállítások szinte mindent fontos opciót tartalmaznak, amire szükségünk van.
A parancsnak hiba nélkül le kell futnia. Ha `Error` üzenetet tartalmazó sort látunk, akkor nézzük meg, hogy mit hiányol a program és telepítsük fel azt a csomagot a rendszerünkre, majd újra adjuk ki a `./configure` parancsot. Ha hibamentesen lefutott, akkor egy összesítést kell látnunk a programba fordított opciókról, valamint fájlrendszerekről. Ezt mutatja az *1. ábra*. Ez a lépés a két verziónál teljesen megegyezik.
Következő lépés a tényleges fordítás. Itt már meg kell különböztetnünk a ***0.4.4*** és a ***0.4.5*** verziószámú programok telepítését. Elsőként nézzük az újabb lépéseit, hiszen ez lesz a bonyolultabb, valamint sokan gondolják úgy, valljuk meg, jogosan, hogy az újabb programverziók stabilabbak, kiforrottabbak és fejlettebbek, mint a megelőzők. A fordítást az `unsermake` paranccsal kell végeznünk a `make` helyett, aminek eredményeként minden bizonnyal egy hibaüzenettel találhatjuk szembe magunkat (*1. lista*).
A hibát szerencsére könnyen orvosolhatjuk, bár egy kicsit be kell kézzel segítenünk a fordításba.
A hiba az alábbi részben van:

```
ret=$? if test "$ret" = 0;,
```

Az `if test` előtt egy `;` kellene legyen, de mint látható, nincs ott. Jelöljük ki a kettős idézőjelek között lévő részt és másoljuk át a vágólapra (jobb kattintás, másolás). A parancssorba illesszük be, majd javítsuk ki az említett hibát. A helyes parancs a következőképpen néz ki (*2. lista*).

Ha ütünk egy ***Entert***, a parancssort vissza kell kapnunk hibaüzenet nélkül. Újra adjuk ki az `unsermake` parancsot, aminek eredménye egy újabb hiba. Az ok ugyanez, tehát ismét kijelölünk, másolunk, beillesztünk és javítunk. Ezt a két lépést addig kell folytatnunk, míg az `unsermake` be nem tudja fejezni a fordítást hibaüzenet nélkül. Következő lépés a telepítés, amihez használjuk az

```
unsermake install
```

parancsot. Ez bemásolja a futtathat fájlt a */usr/local/sbin* könyvtárba, ahonnan a

```
./qtparted
```

paranccsal futtathatjuk. Ezt root jogokkal tegyük meg, aminek eredményeként el kell indulnia a grafikus felületnek. Sajnos tapasztalataim szerint ilyenkor könnyen találhatjuk magunkat szembe egy újabb hibaüzenettel: ***No device found. Maybe you're not using root user?***, ami annyit tesz, hogy nem található eszköz, talán nem root jogokkal futtatjuk a programot. Persze nem ez a hiba, esetleg a javításhoz a naplófájlok nézegetése adhat támpontot, illetve ellenőrizhetjük, hogy minden kernel modul, ami az alaplapi vezérlőhöz tartozik, be van-e töltve.

1. lista

```
/bin/sh: -c: line 0: syntax error near unexpected token `then'
/bin/sh: -c: line 0: `/usr/lib/qt3/bin/uic -L  -nounload -i
➥qp_ui_config.h /work/download/qtparted-0.4.5/src/
➥qp_ui_config.ui > ./src/qp_ui_config.cpp ; ret=$? if test
➥"$ret" = 0; then echo '#include "qp_ui_config.moc"' >>
➥./src/qp_ui_config.cpp; else rm -f ./src/qp_ui_config.cpp;
➥exit $ret; fi'
Error creating ./src/qp_ui_config.cpp. Exit status 2.
```

2. lista

```
/usr/lib/qt3/bin/uic -L  -nounload -i qp_ui_config.h /work/
➥download/qtparted-0.4.5/src/qp_ui_config.ui > ./src/
➥qp_ui_config.cpp ; ret=$? ; if test "$ret" = 0; then echo
➥'#include "qp_ui_config.moc"' >> ./src/qp_ui_config.cpp;
➥else rm -f ./src/qp_ui_config.cpp; exit $ret; fi
```

A 0.4.4 verzió lefordítása és telepítése, ahogy már említettem sokkal egyszerűbb. Adjuk ki a

```
make
```

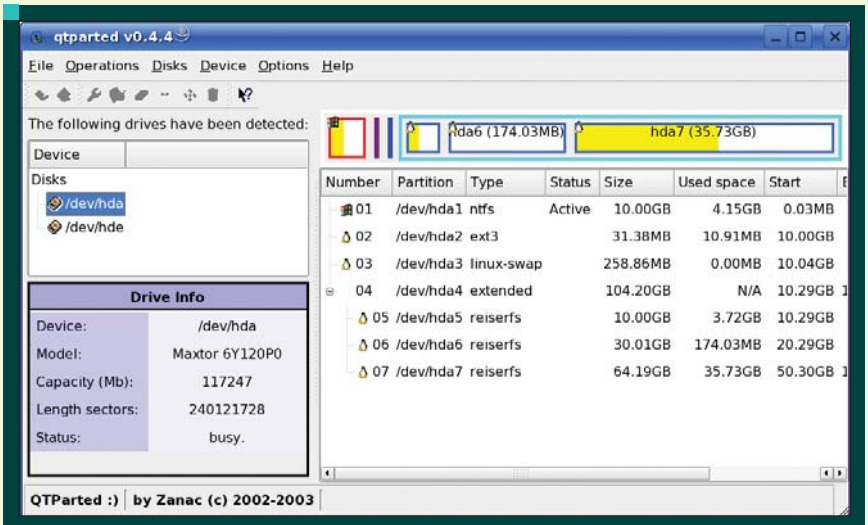
majd amikor befejeződött, a

```
make install
```

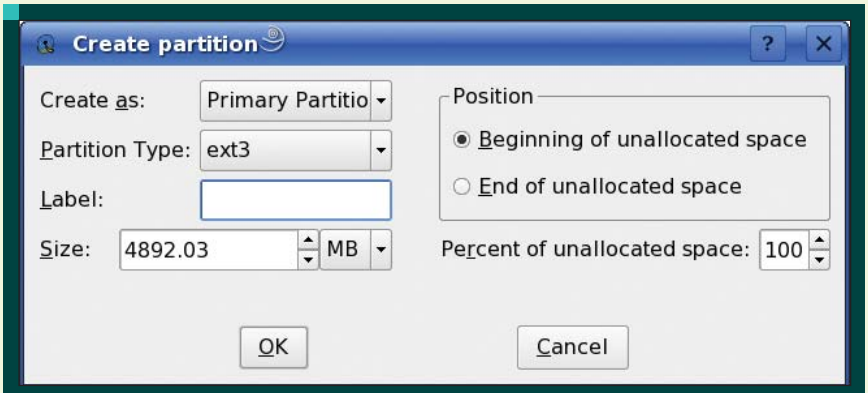
parancsot. A futtatható fájl bekerül a */usr/sbin* könyvtárba, szintén *qtparted* néven. Ezt is rendszergazdaként kell futtatni, de ennél az előbb említett hibaüzenettel még sohasem találkoztam. Megjelenik a grafikus ablak, amelyet a 2. *ábrán* láthatunk. Az ablak három jól elkülöníthető funkciójú részből áll. A baloldalon a felső részben láthatjuk az eszközöket, amelynek partíciós tábláját a program beolvassa, mihelyt kijelöljük. Az alsó részben a kiválasztott eszköz adatait láthatjuk, az ***eszköz nevét (Device)***, a ***merevlemez típusát (Model)***, ***kapacitását (Capacity (MB))***, a ***szektorok számát (Length sectors)*** és az ***állapotát (Status)***. Ez utóbbi a befűzött lemezeknél ***foglalt (busy)***, így az ezen található partíciókat nem lehet szerkeszteni. Amikor kiválasztjuk az eszközt, akkor a jobboldali részben láthatjuk a partíciós tábla tartalmát, vagyis a ***partíciók kezdetét (Start)***, ***végét (End)***, ***fájlrendszerét (Type)***, valamint a ***méretét (Size)*** és a ***foglalt hely méretét (Used space)***. A lista felett grafikusan is láthatjuk a partíciók helyzetét.

A program menürendszere egyszerű és világos. A ***File*** menüben vissza tudjuk vonni a változtatásokat (***Undo***), vagy el tudjuk menteni a módosításokat (***Commit***), illetve ki tudunk lépni a programból (***Quit***). Az ***Operations*** menüben tudunk a partíciókkal dolgozni. Meg tudjuk nézni a tulajdonságait (***Property***), létre tudunk hozni új partíciót (***Create***), meg tudjuk formázni (***Format***), át tudjuk méretezni (***Resize***), el tudjuk mozgatni (***Move***) és törölhetjük is (***Delete***). Szintén ebben a menüben tudjuk a partíciót rendszerindításra kijelölni (***Set Active***), illetve el is rejthetjük (***Hide***).

Ha több merevlemezünk van, akkor a ***Disks*** menüben tudjuk ezek közül kiválasztani azt, amellyel dolgozni szeretnénk. Ugyanezt érjük el, ha a jobboldalon kiválasztjuk a megfelelő eszközt.



2. ábra Végre elindult a QTParted



3. ábra Partíció létrehozása

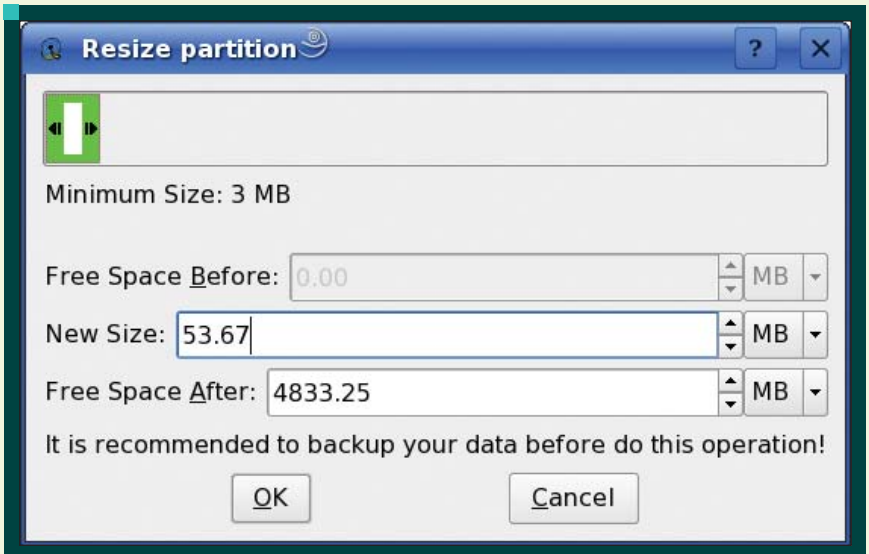
Az ***Options*** menüben lévő ***Configuration...*** menüponttal tudjuk a programokat beállítani, illetve megadhatjuk, hogy a kiegészítő alkalmazások, amelyeket feltelepítettünk a *QTParted* telepítése előtt, milyen útvonalakon érhetők el. A fordítás során ide már a helyes útvonalak kerültek bele, tehát csak akkor kell módosítani, ha a *QTParted* fordítása és telepítése után változtattunk ezen programokon.

Partíció létrehozása

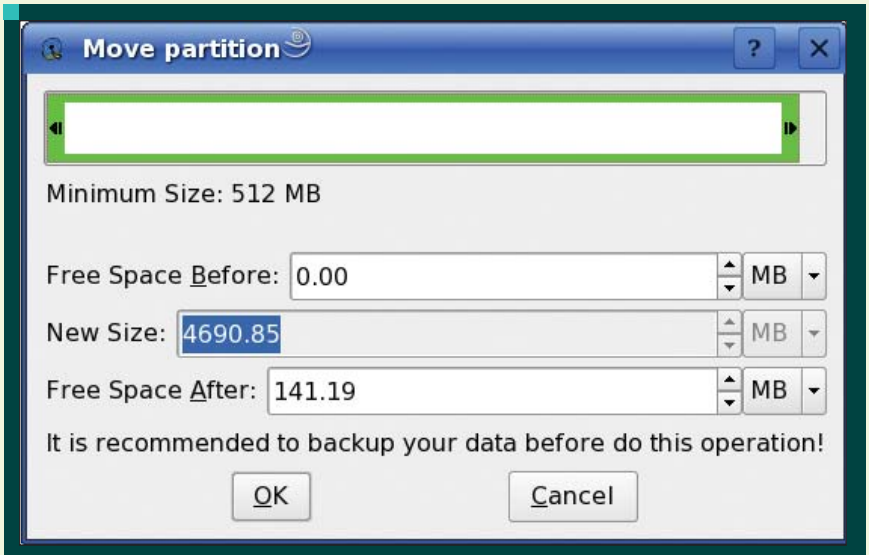
Az egyik leggyakrabban használt művelet az új partíció létrehozása (***Create***) Ehhez szükség van a lemezen egy szabad területre, mire jobb gombbal kattintva kiválaszthatjuk a ***Create*** menüpontot. Ennek hatására megjelenik az alábbi ablak, ahol egy listából kiválaszthatjuk, hogy milyen partíciót szeretnénk létrehozni (***Create as***), mi a ***partíció típusa (Partition Type)***, milyen ***nevet*** adunk a partíciónak (***Label***) és mekkora legyen a ***mérete (Size)***. Lehetőségünk van a szabad helyhez viszonyítottan is megadni a partíció méretét (***Percent of unallocated space***), valamint ezt is eldönthetjük, hogy a szabad hely elején (***Beginning of unallocated space***), vagy a végén helyezkedjen el a partíció (***End of unallocated space***). Amikor elmentjük a változtatásokat, előfordulhat, hogy hibát jelez a program, illetve utána nem a kiválasztott lesz a partíció típusa. Nincs probléma, mivel a ***formázás (Format)*** művelettel ezt majd helyre tudjuk hozni.

Partíció átméretezése

Átméretezni csak olyan partíciót lehet, amely előtt és/vagy után szabad terület van. Kiválasztjuk a partíciót, a jobb egérgombbal rákattintunk, majd kiválasztjuk a ***Resize*** menüpontot. Megjelenik egy ablak, amelyben láthatjuk,



4. ábra Partíció átméretezése



5. ábra Partíció mozgatása

hogy mennyi hely van a partíció előtt (*Free Space Before*), mekkora az új mérete (*New Size*) és mekkora hely van a partíció után (*Free Space After*). Az átméretezéssel óvatosan kell bánni, mert könnyen előfordulhat adatvesztés, ami semmiképpen sem szerencsés, ezért a partíciók szerkesztése előtt csináljunk biztonsági mentést a fontos állományainkról. Miután beállítottunk mindent, az *OK* gombra kattintva a *QTParted* elvégzi a kívánt méretezést. Persze ezt még el kell mentenünk (*Commit*).

Partíció mozgatása

Abban az esetben, ha a partíció mérete megfelelő, de szeretnénk áthelyezni azt a merevlemez egy másik szektorcsoportjára, akkor használjuk a mozgatás (*Move*) parancsot. Ezt sajnos nem minden fájlrendszernél tudjuk megtenni, de a *FAT32*-nél igen. Kiválasztjuk a partíciót, ami előtt vagy után van szabad hely és abba az irányba el tudjuk mozgatni. A program folyamatosan mutatja az aktuális partíció előtti szabad hely méretét (*Free Space Before*), a partíció méretét (*New Size*), ami kicsit félreérthető lehet, de ha figyelmesen megnézzük nem aktív ez a numerikus mező, tehát nem lehet az értékét megváltoztatni. Természetesen a partíció utáni szabad hely mérete is látható (*Free Space After*).

Partíció törlése

Ez a művelet nagyon egyszerű, a törlendő partícióra jobb gombbal kattintunk és kiválasztjuk a *Delete* menüpontot. Minden módosítás után el kell menteni a művelet eredményét, amihez használhatjuk a *File* menü *Commit* menüpontját, vagy az eszköztáron a mentés ikonjára kattintunk. Ennek eredménye egy figyelmeztetés

Partíció formázása

Miután megvannak a partíciók, létre is kell hozni rajtuk a megfelelő fájlrendszert. Ezt megtehetjük abból az operációs rendszerből, amely képes teljes körűen kezelni az adott fájlrendszert, illetve a *Linux* alatti fájlrendszereket a *QTParted* segítségével is létrehozhatjuk. Ehhez kiválasztjuk a partíciót, majd a gyorsmenüben a formázás (*Format*) menüpontot. A fájlrendszer megadása után a formázás folyamatát egy állapotjelző sávban nyomon követhetjük. Tapasztalataim szerint a naplózó fájlrendszereket nem szereti a program, sokszor hibaüzenettel leáll és nem hajlandó a formázást elvégezni.
Mint láthatjuk a program meglehetősen egyszerűen kezelhető, kis gyakorlással minden művelet gyorsan és pontosan, de ami talán még fontosabb, látványosan elvégezhető. A program mindig figyelmeztet, ha valamilyen művelet nem engedélyezett, vagy nem sikerült neki maradéktalanul elvégezni, ilyenkor gondoljuk át, hogy mi is lehet a hiba oka. Azt se felejtsük el, hogy a partíciós táblával végzett munka mindig kockázatokat rejt magában, bármikor lehet egy áramszünet, vagy hibázhat az írásnál, ami az adataink elvesztését jelentheti. Éppen ezért mindig legyen biztonsági mentésünk, amit a rendszerünktől külön tároljunk és ne ugyanazon a merevlemezen.
Sok sikert kívánok ennek a remek kis programnak a használatához.

Markó Imre
(linux@akribisbt.hu)

Hardvermérnök és mérnöktanár végzettsége van. Saját cégében Linux rendszerek tervezésével és üzemeltetésével foglalkozik. Ezen kívül egy főiskolán oktat, elsősorban hardveres tantárgyakat.

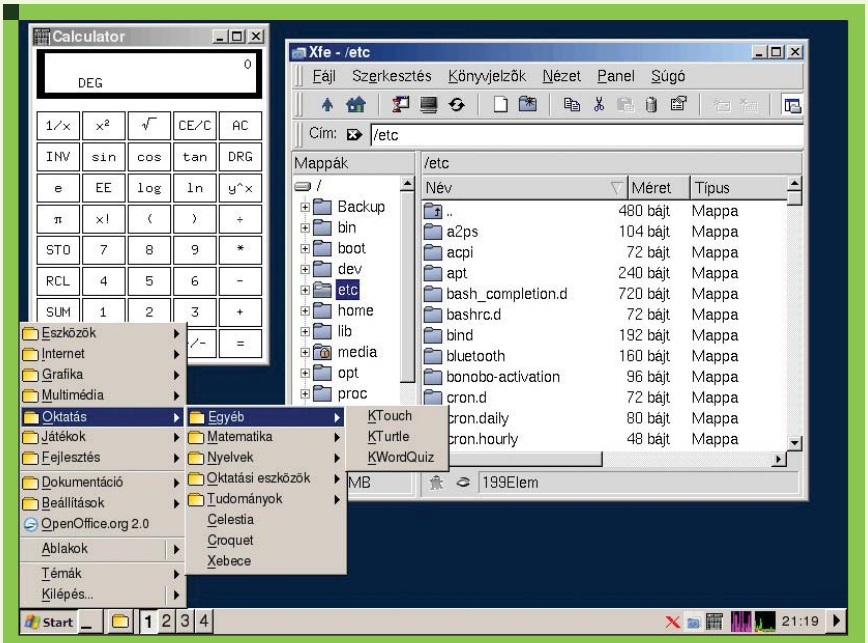
IceWM – Egy népszerű, kis méretű ablakkezelő

A közelmúltban sokat foglalkoztam az IceWM ablakkezelővel, és úgy gondoltam, tapasztalataimat megosztom a Linuxvilág olvasóival. Ebben a cikkben úgy próbálom áttekinteni ennek a rendszernek a használatát, hogy főleg az érdekességekre, a dokumentációban nem szereplő dolgokra koncentrálok.

Az *IceWM* megalkotásánál a cél az volt, hogy egy kicsi, gyors, de mégis a megszokottakhoz hasonló ablakkezelőt hozzanak létre az *X Window* rendszerre. Az eredeti tervek szerint korábbi operációs rendszerek kinézetét emulálta volna a program, de azóta már több, korszerűbb kinézet is elkészült hozzá, mivel kinézete a hozzáértők számára könnyedén átszabható. Az *IceWM* beállításai is jól testre szabhatóak, azonban minden beállítás alapból csak konfigurációs állományokon keresztül lehetséges. Ez a nagy hátránya előnnyé válik akkor, amikor belegondolunk abba, hogy a fejlődés útja nem csak az egyre bonyolultabb, a profik számára minél több funkciót felvonultató, hanem a „bolond” felhasználók elől minél többet elrejtő alkalmazásokon át is vezet.

Hiszen bizonyára mindenkivel előfordult már, hogy szépen beállított asztali környezetét pillanatok alatt romba döntötték a hozzá nemértők, mondjuk egy egész panel eltüntetésével. Képzeljük el milyen zavaró lehet ugyanez egy internet kávézóban, egy kiállításon vagy egy sok gépes munkahelyen! Rendszergazdaként az *IceWM* használatával könnyedén lehetőségünk nyílik arra, hogy olyan rendszert adjunk a felhasználók kezébe, amely csak a számukra szükséges alkalmazásokat teszi elérhetővé, megkönnyítve ezzel mindkét fél munkáját.

Az *IceWM* sebessége és csekély mérete miatt ideális ablakkezelő régebbi, lassabb gépeken, ott ahol kevés hely áll rendelkezésre, illetve távoli grafikus bejelentkezéshez.



1. ábra Az IceWM igyekszik a megszokott kinézeteket és működést biztosítani, hogy megkönnyítse az átállást

Beszerzés, fordítás, telepítés

Az *IceWM* a legtöbb *Linux* disztribúcióhoz letölthető csomagban, és az ott megszokott módon telepíthető. Ha azonban az ablakkezelő legújabb verziójára és annak minden funkciójára vagyunk kíváncsiak, akkor érdemes lehet forrásból telepíteni. Ez szerencsére nem túl bonyolult.

Először is töltsük le a program forrását a *sourceforge.net/project/icewm (icewm.sf.net)* vagy a *www.icewm.org* címről (Egyből a letöltésre: *prdownloads.sourceforge.net/icewm*). A forrás installálása általában minden rendszeren probléma mentes kell legyen (függőségek: *libxpm*, *gcc*, *make*). Először ki kell bontanunk a forrás állományokat, majd a kibontott könyvtárba lépve konfiguráljuk, lefordítjuk és installáljuk a programot (xx a verzió helye):

```
tar xzf icewm-0.9.xx.tar.gz
cd icewm-0.9.xx
./configure
make
make install
```

2. ábra Bejelentkezés egy másik gépről, miközben egy modernebb kinézetet látunk

1. táblázat *Az IceWM beállítófájljai*

`keys`	A gyorsbillentyűk leírása.
`preferences`	Az egész ablakkezelőre vonatkozó megjelenés, viselkedés, funkciók állítása. Minden beállítás jól le van írva, átállítása végtelenül egyszerű.
`prefoverride`	A témák beállításainak felülbírálása (általában nem használt)
`startup`	Az indításkor mindig végrehajtásra kerülő szkript. Futtathatónak kell lennie.
`menu`	A menüszerkezet leírása. UHU Linux esetében a rendszer által generált menü lesz betöltve, de írhatunk sajátot menüt is. A nem létező alkalmazások automatikusan kimaradnak.
`toolbar`	A menüvel megegyező módon írja le a panel ikonjait és almenüit.
`winoptions`	Leírja az egyes ablakok, alkalmazások viselkedését.

(Az utolsó parancs root jogokkal indítandó!)
Ha a fordítás elsőre sikerült, akkor bonyolíthatunk a dolgon, adjuk ki a

```
make clean
```

parancsot (ez törli a fordításnál keletkezett fájlokat, így újrakezdhetjük a folyamatot), majd kérdezzük le a lehetséges fordítási opciókat a

```
./configure --help
```

paranccsal.

Amint látható, az *IceWM* valójában sokkal többre képes, mint alapkiépítésben. Van benne például *antialiasing* és *Gnome* menü támogatás, sőt létezik egy *„lite”* verziója is. Ha kedvünk tartja válasszuk ki a kívánt opciókat és az első példa alapján fordítsuk le ismét. A legszebb *IceWM* témákhoz (például *silverXP*) külön foltokat (*patch*) is mellékelnek, amelyekkel újra kell az ablakkezelőt fordítanunk.

Az IceWM indítása

A legtöbb ember bejelentkezés kezelőből indítja az *IceWM*-et, amit ha csomagból installáltuk, akkor a bejelentkezés kezelőben is egészen biztosan megtaláljuk. Ha azonban mi magunk fordítottuk és előtte nem volt ilyen csomagunk, akkor ezzel a résszel gondunk lehet. A *KDM* bejelentkezés kezelő használata esetén is szerencsénk van, ugyanis az *IceWM*-et automatikusan felismeri, *GDM* és *XDM* esetén viszont a */etc/X11/* könyvtár alatt érdemes szétnéznünk. Ha nem használunk bejelentkezés kezelőt, vagy mint haladó *Linux* felhasználók *X* szerver nélküli parancssoros üzemmódban dolgozunk és hirtelen szükségünk van egy grafikus felületre, akkor az *IceWM* indításához írjuk be:

```
startx icewm
```

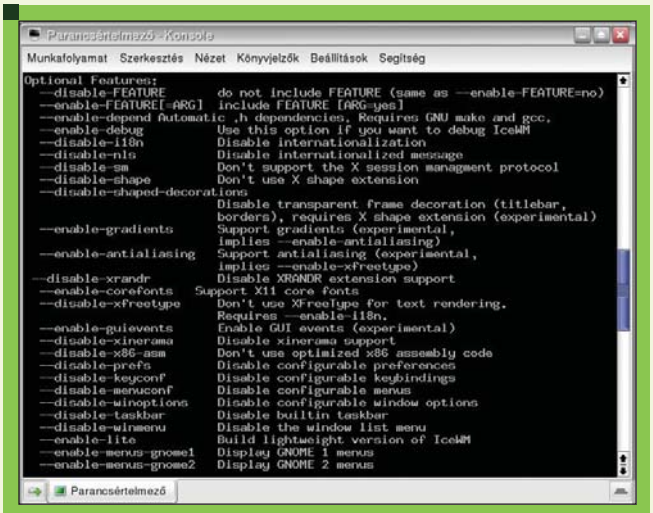
Előfordul, hogy munka közben valamilyen okból kellene még egy grafikus környezetet nyitnunk, ekkor indítsunk egy újabb, minimális erőforrásigényű *X* képernyőt az *IceWM* ablakkezelővel, amit akár egy *CTRL+ALT+BACKSPACE* gombkombinációval is pillanatok alatt becsukhatunk:

```
startx icewm -- :1
```

Az `icewm` parancs helyett lehet használni az `icewm-session` parancsot is, a különbség csak annyi, hogy az előbbi csupán az ablakkezelőt és a panelt indítja, az utóbbi viszont beállítja a hátteret (`icewmbg` parancs) és a indítja a rendszertálcát is (`icewmtray` parancs).

Az IceWM beállítása

Mint említettem az *IceWM* beállítása ránézésre nem egyszerű feladat, mivel alapból grafikus beállító programot nem tartalmaz, de az interneten hosszasan keresgélve több ilyen grafikus felületű beállító programot is találhatunk. Az *ICEpref*, *ICEpref2*, *ICEcc*, *IceWMConf*, valamint a legjobban ajánlott *IceWM Control Panel* mind az ablakkezelő beállítására szolgáló grafikus alkalmazások. Azonban helyzetet bonyolítja, hogy maga az *IceWM* honlapja sem tartalmazza mindegyiket, egyes linkek nem élnek, a beállító programok már elavult *IceWM* verziókra íródtak, nem futnak rendszerünkön vagy honlapjuk enyhén szólva az őskáosz állapotában leledzik... Tapasztalatom



■ ***3. ábra*** Számtalan lehetőségünk van a fordításnál

■ ***4. ábra*** Az IceWM Control Center azaz ICEcc beállítóprogram

2. táblázat ***A winoptions beállításai***

`Firefox-bin.tray: 2`	Csak a rendszertálcán jelenjen meg, ha nyitva van
`Firefox-bin.workspace: 1`	Az 1-es munkaasztalon legyen
`tvtime.geometry: 320x240+100+150`	Az ablak mérete, helyzete
`tvtime.dMinimize: 0`	Ne legyen minimalizáló gomb
`tvtime.dMaximize: 0`	Ne legyen maximalizáló gomb
`tvtime.dBorder: 0`	Ne legyen keret
`tvtime.layer: OnTop`	Mindig felül legyen
`tvtime.allWorkspaces: 1`	Minden asztalon megjelenjen
`gkrellm.Gkrellm.layer: Below`	Mindig a többi ablak alatt legyen
`Gkrellm.Gkrellm.ignoreTaskBar: 1`	Ne legyen látható a tálcán
`rxvt.icon: xterm`	Az alkalmazás ikonjának beállítása

szerint ezeknek a beállító programoknak a próbálgatásába ölt idő nem térül meg, ellenben a konfigurációs fájlok szerkesztése pofon egyszerű feladat.
Az eredeti konfigurációs állományok megtalálhatóak az ***icewm*** könyvtárban. Ha magunk fordítottuk a rendszert,a kkor ez általában a*/usr/local/share/icewm* könyvtár, ha csomagból tettük fel, akkor a */usr/share/icewm*. A biztonság kedvéért azért tehetjük őket a */etc/icewm* vagy a *$HOME/.icewm* könyvtárba is, így újratelepítés esetén is megmaradnak. A konfigurációs állományok olyan egyszerűen átláthatóak, hogy általában dokumentáció nélkül is boldogulunk velük.

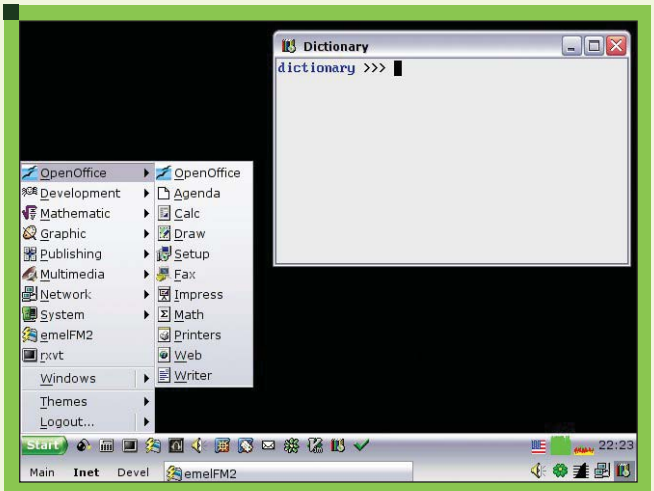
A ***winoptions*** állománynál álljunk meg egy szóra, mert egyrészt ez az ablakkezelő egyik legjobb funkciója és idáig ez az egyetlen terület, amellyel a túl kevés megadott példa és a dokumentáció hiányossága miatt gondunk lehet.
Segítségképen leírok néhány beállítást, egyben ízelítőt adva a lehetőségekből (2. *táblázat*).
Felvetődik a kérdés, hogy honnan tudjuk, hogy mi az aktuális alkalmazásablak neve, amit vezérelni szeretnénk, vagyis honnan tudtuk meg, hogy a ***Firefox*** ablak neve *„Firefox-bin”*. Ennek a lekérdezéséhez terminálból használjuk az `xprop` parancsot, amit elindítva megváltozik egér kurzorunk alakja, majd a lekérdezni kívánt ablakra kattintva egy lista jelenik meg az ablak paramétereivel.
A keresett ablak nevét itt találjuk:

```
WM_CLASS(STRING)=XXXYYY
```

Ezzel a funkcióval átláthatóbbá, szebbé, könnyebben kezelhetővé tudjuk varázsolni munkaasztali környezetünket, hiszen gondoskodhatunk arról, hogy az állandóan futó alkalmazások ne jelenjenek meg feleslegesen a tálcán (taskbar), kinevezhetünk munkaasztalokat egy adott feladatra és beállíthatjuk, hogy minden ezzel kapcsolatos alkalmazás ott jelenjen meg.

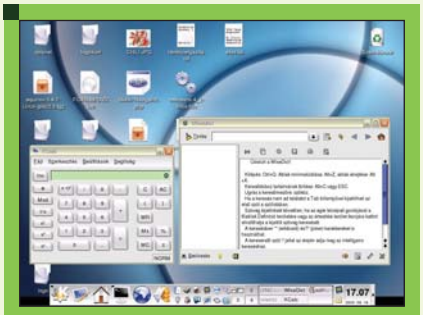
Kinézetek, hátterek, ikonok és trükkök

Az ***IceWM*** könyvtárában a konfigurációs fájlok mellett találunk három könyvtárat is. Az ***icons*** könyvtár az ikonok tárolására szolgál, a ***wallpapers*** a hátterek, a ***themes*** a különböző kinézetek (témák) könyvtára. Érdemes elmondani, hogy kinézeteket legnagyobb számban és legjobb minőségben érdekes módon nem az *icewm.org* honlapon találunk, hanem a *themes.freshmeat.net* oldalon.
Itt megszámlálhatatlan, jobbnál jobb minőségű téma közül válogathatunk, amelyek installálása abból áll, hogy a témák könyvtára alá kitömörítjük őket. Megfigyelhetjük, hogy az egyes kinézetek beállító állományai az *icewm/themes/kinézet_neve/default.theme* fájlban találhatóak, ahol minden beállítás a *preferences* fájlhoz



■ *5. ábra* Egy szépen beállított dupla méretű panel

■ *6. ábra* IceWM, háttérben a KDE munkaasztallal ikonjai



■ *7. ábra* A KDE IceWM ablakkezelővel

■ *8. ábra* Az IceWM, háttérben a Gnome munkaasztallal

■ *9. ábra* Az iDesk ikonkezelő és IceWM

hasonlóan jól el van látva megjegyzésekkel. Itt vannak meghatározva az adott téma színei, betűkészletei és a háttér is, amelyet az említett `icewmbg` parancs elindításával tudunk kirajzoltatni.
Ha már idáig eljutottunk, miért ne lépjünk egy kicsit tovább, és feszegessük az *IceWM* lehetőségeinek határait. Mivel az *IceWM* csupán ablakkezelő, készítésénél nem volt szempont a munkaasztal ikonjainak kezelése. Erre azonban mégis lehetőségünk nyílik, de így az egyik legfőbb előnyéről, a kis erőforrás igényéről kell lemondanunk. Érdekességképpen tehetünk egy kísérletet a *KDE* munkaasztal bekapcsolásával, amely természetesen a munkaasztali ikonok kezelésének lehetőségét is magával hozza. Indítsuk el az alábbi parancsot:

```
kdesktop
```

Ennek éppen a fordítottját csinálják sokan, akik a *KDE* munkaasztal környezetet preferálják, de nincsenek kibékülve ablakkezelőjének sebességével és kinézetével. Ha *KDE*-ben a *kwin* helyett a sokkal gyorsabb *IceWM*-et szeretnénk használni, akkor a következőképpen indítsuk el:

```
export KDEWM="icewm"
startx
```

Ebben az esetben célszerű az *IceWM* eszköztárat (*toolbar*) a *preferences* fájlban kikapcsolni, mert különben az is ábrázolódik, e fölött jelenik majd meg a *KDE* panel. Visszatérve a munkaasztali ikonokra, sokan a *Gnome* munkaasztal kezelését szeretik, ezt is előhívhatjuk az alábbi parancs segítségével:

```
nautilus -n
```

Lehetőség van más programokkal is ikonokat varázsolni az asztalunkra. Az egyik ilyen program a *DFM*, azaz *Desktop File Manager*.
Ezt a *www.kaisersite.de/dfm/* címen találjuk meg, jelenleg a *0.99.9*-es verziónál tart, azonban kinézete még nem az igazi. Létezik egy olyan megoldás is az asztali ikonok kezelésére, amit kifejezetten az *IceWM*-hez hasonló kis erőforrásigényű ablakkezelőkhöz fejlesztenek, és a kinézetével sem lehetnek problémáink. A program neve *iDesk*, megtalálható az *idesk.sourceforge.net* honlapon, mostanában jött ki újabb stabil verziója.
Az *IceWM* használatához sok sikert kíván a szerző.

Somorjai Ede
(info@gazlap.hu)
Budapesten él családjával. Érdeklődési köre leírhatatlanul széles, de leginkább a Linux körében forog. 10 éve találkozott először vele, két éve mindenre csak ezt használja (épp csak náthára nem). Kedvence az UHU Linux, amelyre csomagokat is készít.

Karakteres böngészők

A 2005 augusztusi Linuxvilágban Marcel Gagné elfeledett terminálokról ír. Ennek továbbgondolására vállalkoztam azzal, hogy a karakteres webböngészőknek eredtem a nyomába. Még a mai gyors-grafikus-mindentudó böngésző-áradatban is hasznunkra lehet időnként egy karakteres felületű program, több okból is.

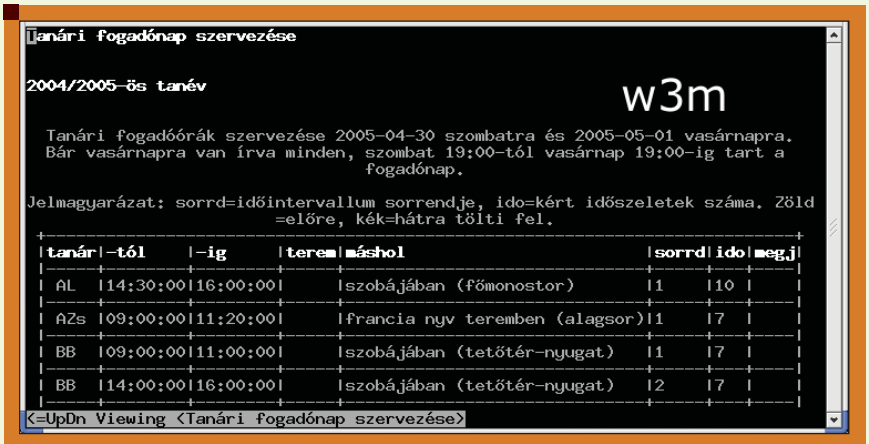
A munkám során (iskolában) különböző számítógépes környezetekkel dolgozom. Informatika órán a diákok hálózatára kapcsolt gépet látok, s innen csak *SSH*-n keresztül érhető el a tanári hálózat és az azon futó intranetes webszerver. Ez utóbbi azért fontos, mert ezen fut az elektronikus osztályozónapló. (Lehetne persze az *SSH* `-X` vagy `-Y` kapcsolóit is használni, de ezekhez egyrészt a rendszergazdát is igénybe kellene venni, másrészt orvosolni kellene a felmerülő biztonsági és sávszélességbeli problémákat.) Ennél egyszerűbb egy testhezálló karakteres böngésző használata – legalábbis olyan célra, amire eddig nekem szükségem volt.
Lehetnek persze más, ennél általánosabb okai is annak, hogy a grafikus böngésző helyett egy „fapados” karakteres változatot használjunk.

- A *pine* vagy *mutt* levelezőklienst használók széles tábora (mint jómagam) gyakran kap *HTML* formátumú vagy más bosszantó mellékletet hordozó levelet. Ilyenkor jól jön a karakteres böngésző (és még néhány egyéb segédprogram, amikre még visszatérünk).
- Lássuk be: az igazán fontos információk szöveges formában jelennek meg. Ha valakit nem a puszta szórakozás vezet, akkor általában nem a multimédiás lehetőségeket keresi a weben, és főleg nem a beérkező leveli között. Vannak olyan honlapok, amelyeken kifejezetten zavaró a sok kép.
- A webes fejlesztők számára is kiváló lehetőség a karakteres böngésző arra, hogy oldaluk használhatóságát teszteljék az *„any browser”* elv értelmében.
- A karakteres böngésző kíméli a szemet és a helyi erőforrásokat. Nagy segítség ez annak, aki órákon át fejleszt, vagy elavult hardver használ.
- A karakteres böngésző gyorsan elindul. Nem véletlen, hogy a *Midnight Commander* is ilyent indít alapértelmezésként még akkor is, ha grafikus felületen futtatjuk. (Elismerem, vannak ma már kellően gyors grafikus böngészők is.)
- Mobiltelefonokra vagy kéziszámítógépre nem is nagyon lehet más böngészőt telepíteni, és a vakok illetve gyengénlátók számára is a karakteres böngészőkön keresztül vezet az út a web felé.

A számítógép hálózatok kialakulásának idején divatos volt mindenféle állatokról elnevezni a különböző protokollokat (a *Gopher* például hörcsögöt jelent). Ekkor született meg a *Kansasi Egyetemen* a *Lynx* (vagyis Hiúz). Leszármazottai közé tartozik az *Alynx* (*Amiga-lynx*), *MacLynx*, majd (a kiejtést megtartva, de szójátékkal a nevében a linkekre utalva) a *Links* és számos egyéb változat. Használható a *w3m* is, ami a japán *„WWW-wo-miru”* (*„nézd meg a www-n”*) kiejtését követi.
Ígéretesnek tűnik a jelenleg béta állapotú *Netrik*, amelynek fejlesztői nem kisebb célt tűztek maguk elé, mint azt, hogy programjuknak az összes változat közül a legjobb lesz a felhasználói felülete.
Nem karakteres böngésző ugyan, de azokhoz hasonlóan igen kevés erőforrást igényel a *Dillo*. Nem véletlen, hogy a *DamnSmallLinux* is ugyanezt szeretné elérni.
Nézzük tehát az egyes versenyzők előnyeit és hátrányait.

w3m

Azért vettem előre ezt a programot, mert kicsit kilóg a sorból hiszen a többiek történetileg kicsit közelebb állnak egymáshoz.
A program japán eredetű, szerzője *Akinori Ito (aito@fw.ipsj.or.jp)*.
E sorok írásakor a 0.5.1 a legfrissebb verziója (ami már másfél éves).
Sajátos, számomra nem könnyen kiismerhető billentyűkombinációkkal és kapcsolókkal vezérelhető. Indításkor például mindenképp meg kell adni a meglátogatni kívánt URL-t vagy fájlt. A kapcsolók egy része megjeleníthető `w3m -help` paranccsal (vagy a paraméter nélkül hívott `w3m` paranccsal); az igazán érdekesek pedig a `w3m -show-option` kapcsoló hatására bukkannak elő.
Tartalmaz képmanipuláló lehetőségeket is, ugyanis megfelelő módon beállítva, alkalmas terminálon képeket is meg tud jelenteni külső program segítségével (bár akkor már inkább a *Firefox*...). Különös módon itt egy másik böngészőprogram is megadható, ami kicsit értelmetlennek tűnik.
Egy hivatkozást úgy lehet követni,



1. ábra A w3m

2. ábra A Lynx

hogy kurzorral odalépegetünk, majd Entert ütünk. A kurzornak sajnos közben át kell haladnia minden karakteren – azokon is, amik nem tartalmaznak hivatkozást, viszont egérrel megfelelő sebességgel használható. *SSL* kapcsolatot is tud követni, és kereteket („frame”-eket) is meg tud mutatni, táblázattá konvertálva őket. Érdekessége, hogy kézikönyvlapok olvasására is használható (`w3mman`).
Az *Emacs* is használja a *w3m.el Emacs LISP* modulon keresztül. Gyors *HTML* böngészést tesz lehetővé az *Emacs*-on belül, hogy a lassabb *Emacs/W3*-mal ellentétben, amely a weboldalak értelmezését és kijelzését is maga végzi, a *w3m.el* csak a kijelzéssel foglalkozik, az értelmezést a *w3m*-re bízza.
Egy továbbfejlesztése a w3mmee (➲ *http://pub.ks-and-ks.ne.jp/prog/w3mmee/index.shtml.en*), mely nevében a *„Multi-Encoding Extension”*-ra utal. *Boehm-GC* könyvtárat is igényel a telepítése, és e sorok írásakor *w3mmee-p24-22* a legfrissebb verziója.

Lynx

Bátran mondhatjuk, hogy a legkiforrottabb karakteres böngésző.
A *Lynx 2.8.5rel.2* a legfrissebb 2005 augusztusában. Szerzői: *Lou Montulli*, *Michael Grobe*, *Charles Rezac*, *Garrett Blythe*.
Használata parancssorból (ahogy az összes többié is):

```
lynx [kapcsolók] [elérési út vagy URL]
```

segítséget a

```
lynx -help
```

ad.
Ügyesen használja a kapcsolókat a Midnight Commander beállítási fájlja (*UHU-Linux* alatt a */usr/share/mc/extfs/sfs.ini* ; a név a *Single FileSystem*-re utal), amelyben megadható, hogy a különböző fájltípusokkal hogyan kell elbánni – nevezetesen az *uhtml*-lel így:

```
lynx -force_html -dump %1 > %3
```

az egyéb *URL*-lel pedig így:

```
lynx -source `echo "%2" | sed 's-|-/-g'` > %3
```

A csatolt fájlként e-mail-ben érkező *HTML* fájlok átolvasását elősegítendő (*pine*, *mutt* alatt) célszerű a *.mailcap* fájlban elhelyezni:

```
lynx -dump %s; copiousoutput; nametemplate=%s.html
```

valamint a szívszomorító *.doc* illetve *.rtf* csatolmányok megnézésére:

```
antiword %s;
```

valamint

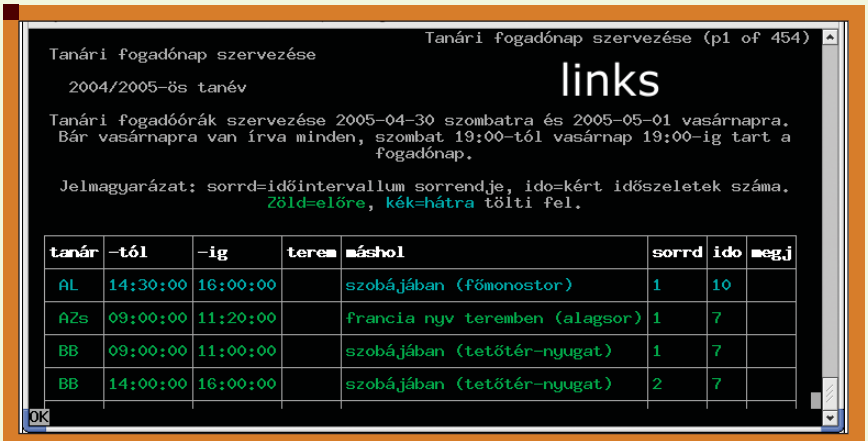
```
unrtf –text %s
```

(A részletekért látogassunk el a következő webhelyre: *http://www.go.dlr.de/pdinfo_dv/pine.html, http://jriddell.org/converting-from-pine-to-mutt.html*).
A *Lynx* hihetetlen mennyiségű *URL* sémát, protokollt támogat: *http*, *https*, *telnet*, *tn3270*, *rlogin*, *gopher*, *file*, *ftp*, *wais*, *news*, *nntp*, *snews*, *newspost*, *newsreply*, *snewspost*, *snewsreply*, *mailto*, *finger*, *cso*, *bibp* (*bibliographic protocol*), *lynxexec*, *lynxprog*, *lynxcgi*, *internal*. Ezek közül a *Lynx* saját moduljait használó, belterjes protokollok nyilván feledésbe merülnek idővel.
Maga a *Lynx* dokumentáció is kéri, hogy „álljunk ellen a kísértésnek”, és ne használjunk egzotikus modulokat. (➲ *http://www.subir.com/lynx/lynx_help/lynx_url_support.html*)
A *Lynxben* a navigálás a kurzorbillentyűkkel történik, értelemszerűen.
Ha valaki ezt megszereti, akkor be lehet állítani a *Midnight Commanderben* való mozgás megvalósítására is, nevezetesen a

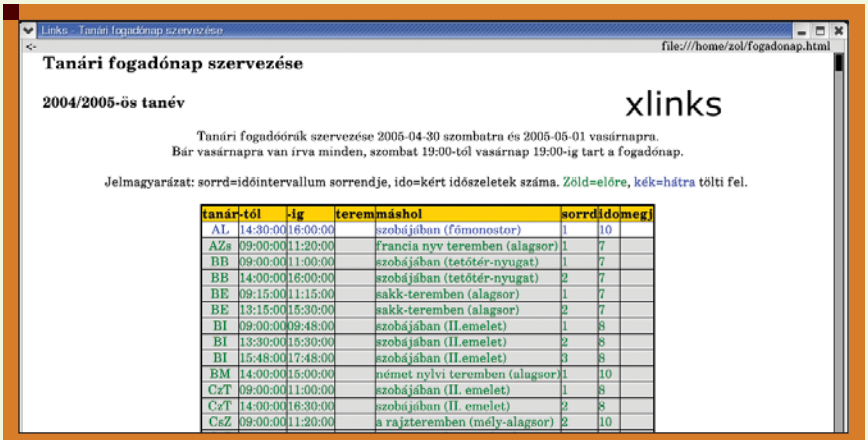
```
navigate_with_arrows=1
```

megadásával a *.mc/ini*-ben.

(*?* és *h*: Súgó, *o*: Beállít, *p*: Nyomtat, *m*:Főablak, *q*:Kilép, /:Keres, *Del*:Előző, \: oldal forrása.)
Felhívnám a figyelmet a *p* jelentőségére, ami a „fájlba nyomtatást”



3. ábra A Links



4. ábra Az xlinks

– magyarul a mentést – is takarja. Ezekre van lehetőség itt: mentés helyi fájlba, küldés levélben, nyomtatás a képernyőre (vagyis a szabványos kimenetre), nyomtatás a *vt100*-as terminálhoz csatlakozó nyomtatóra.
Itt érdemes megemlíteni, hogy a *links* ezen túl még „formázott dokumentumot” is tud „nyomtatni”: ekkor csak a szövegtest kerül átadásra, mindenféle formázási elő- és utótag nélkül. Az aktuális oldalról és aktuális linkről szóló információkat az „=” lenyomásával lehet megszerezni: *URL*, cím, karakterkészlet, kiszolgáló, dátum, utolsó módosítás, tulajdonos, méret, mód.
A *g* gombbal (*„go”*) ugorhatunk egy *URL*-re, a *G* betűvel pedig az aktuális *URL*-t átszerkesztve kérhetjük egy-egy oldal betöltését. *d* indítja a letöltést (*„download”*), a beállításokat pedig az *o* (*„options”*) gombbal kért űrlapon változtathatjuk.

Előnye még, hogy a (munkám során időnként előkerülő) iszonyatosan nagyméretű *HTML* táblázatokat is pillanatok alatt megmutatja – ilyen feladatra semmilyen más böngésző nem lenne képes (grafikus meg pláne nem, „polinom idő alatt”).
Nagy hátránya, hogy nem támogatja a *JavaScriptet* (meg persze a *Javat* és ennél komolyabbakat sem), valamint a kereteket („frame”-eket) sem – esetleg csak különkülön tudja őket megmutatni (bár az is igaz, hogy egyre kevésbé szalonképesek a „frame-es” weboldalak).
E hátrányok kiküszöbölésére született a *Links*.

Links

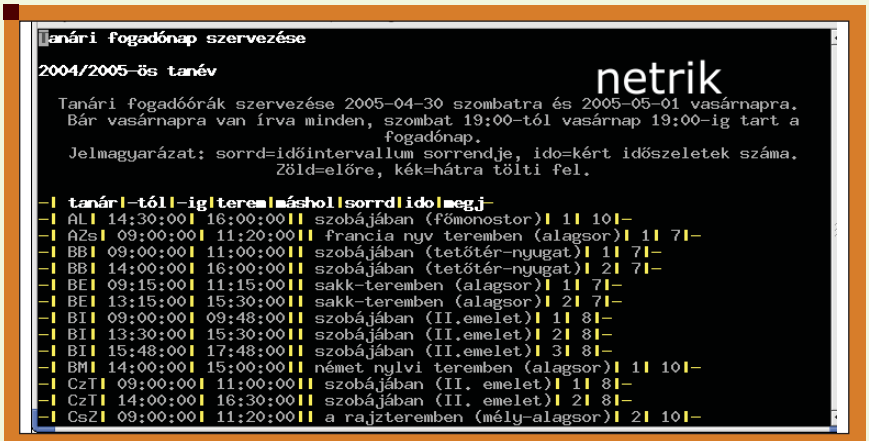
A *w3m*-mel és a *Lynx*-szel ellentétben a *Links* a grafikus felülethez szokott felhasználókat célozza meg (menük, felugró ablakok).
Előnye, hogy a kért oldalt olvasás közben már értelmezi (rendereli) és meg is mutatja. A magam részéről ennek a kettes verzióját tudom (és szeretem) a legjobban használni a karakteres böngészők közül: támogatja a kereteket, a *JavaScriptet*, az *SSL*-t (az *OpenSSL* révén).
Links 2.1pre15 a legfrissebb verziója ezen sorok írásakor.
A program önmagát *„Lynx-típusú szöveges WWW-böngésző”*-ként definiálja.
A ➲ http://links.sourceforge.net weboldalon már ennél markánsabban fogalmazódik meg, hogy ez „A” szöveges webböngésző (Ezzel ellentétben a *w3m* weboldalán a *Lynx*-et illeti ez a megtisztelő kiemelés).
0.98 a legfrissebb stabil verziója, és létezik pre-99-es verzió is.
Az eredeti projekt vezetője *Mikulás Patoeka*, (segítői: *Petr Kulhavy*, *Karel Kulhavy*, *Martin Pergel*).
A hivatalos honlap a ➲ http://artax.karlin.mff.cuni.cz/~mikulas/links címen található.
Az elágazásokat mások vették kezükbe.

Elinks

A két szerző *Petr Baudis* és *Jonas Fonseca* (➲ *http://elinks.or.cz/*). Ellentétben az eredeti *Links*-el, ez még mindig aktív fejlesztés. (A *links-lua* is ebbe épült be.) Érdekessége, hogy a Javascript támogatást külső programból (például *SpiderMonkey*: ➲ *ftp://ftp.mozilla.org/pub/mozilla.org/js/js-1.5-rc6a.tar.gz*) lehet belefordítani: igen jó és működő leírást ad ennek menetéről a programmal letölthető *ecmascript.txt*.
Ennek elolvasását annak is ajánlom, akit nem érdekelnek a karakteres böngészők: jó példa arra, hogy hogyan érdemes megírni egy olyan dokumentációt, ami egy nem triviális műveletsor (parancssori) elvégzését járja végig, a végén sikerélményt adva a vállalkozónak. Az igazsághoz hozzátartozik sajnos, hogy ezzel együtt az *Elinks* még mindig jóval kevesebb *JavaScript* parancsot ért meg, mint a *Links(2)*.

Links(2)

A *Links-0.92*-ből ágazott el, felhasználva a kísérleti jellegű *wb0* (➲ *http://atrey.karlin.mff.cuni.cz/~clock/twibright/wb0*) grafikus böngésző jellemzőit.



5. ábra A Netrik

(➲*http://atrey.karlin.mff.cuni.cz/~clock/twibright/links*). A *Twibright Labs* a prágai *Károly Egyetem* csoportja – *Mikulás Patoeka* is tagja volt. Nekik általában is igen érdekes hardveres és szoftveres projektjeik vannak. Ilyen például `pdf2html`; a zajos csatornán való adatátvitelt megvalósító `brutalcopy`, illetve a *Schizzors*, ami speciális fájlfeldarabolást végez: egy fájlból két (törvényesen is bárhol tárolható) véletlenszerű fájlt állít elő, amelyeket *XOR*-ral lehet egyesíteni az eredetivé.

A *Links(2)* – némiképp érdekes fejleményként – már grafikus felülettel is rendelkezik (➲ *http://links.twibright.com/features.php*). Az *UHU-Linuxban* mint *links-x11* szerepel, és az `xlinks` paranccsal indul.

Links-hacked

Ez a változat úgy született meg, hogy *Szergej Karpov* egyesítette az *Elinks* néhány tulajdonságát a *Links(2)*-évél.(➲ *http://xray.sai.msu.ru/~karpov/links-hacked*). Ez az oldal csak jelszóval közelíthető meg, de az *SDL-Linux* letölthető iso-képfájljai (amelyek híresek kis méretükről: 50MB) is tartalmazzák.

Escape vagy *F9* hatására a *Links(2)* menürendszere jelenik meg.

A *Fájl* menüben a szokásos menüpontokon kívül érdekes még a *Gyorsítótár kiürítése*, az *Erőforrás- és memória-információ*, valamint a parancssor kérése. A *Nézet* menüben találjuk a szokásos nézetfajtákat és keresést, valamint itt adhatjuk meg a *HTML* beállításokat.

A *Link* menüben állítható be a nyelv, a kódlap, a terminálkezelés. Itt adhatók meg a hálózati beállítások, a *JavaScript* kezelése, a különböző hozzárendelések és a fájlkiterjesztésekkel kapcsolatos információk. Ne felejtsük el menteni őket ah változtattunk valamin (legalsó menüpont). A Súgó menüben az itt megszokott dolgokon kívül a színkalibráció is helyet kapott.

A gyorsbillentyűk teljesen megfelelnek a *Lynx* viselkedésének, ugyanakkor az előddel ellentétben itt alapértelmezésben a helyi fájlrendszeren is érvényes a billentyűk jelentése.

Netrik

Ez a program a legkisebb méretű, de még annyira béta állapotú, hogy lényegében élvezhetetlen.

A vállalkozó kedvűek kipróbálhatják a ➲ *http://netrik.sourceforge.net/?intro.html* címről indulva.

Dillo

Eme böngésző is kilóg a sorból, mert nem karakter-alapú, viszont kis mérete és gyorsasága miatt említést érdemel. A cikk írásakor a 0.8.5-ös verzió a legfrissebb stabil változat. 5 éve folyik a fejlesztése, kizárólag C nyelven. A forrás kevesebb, mint 420 kB – a binárist lásd az alábbi táblázatban. Célul tűzi ki a kis méret, stabilitás, használhatóság és felhasználó-barátság összeegyeztetését. Egyelőre *GTK+* az alapja (tehát nem kell a futtatásához még *GNOME* sem), de a következő kiadás FLTK2-alapú lesz. Szívesen használják beágyazott rendszerekben vagy „kisméretű terjesztésekben” (Lásd a *2005 szeptemberi Linuxvilágban Marcel Gagné* cikkét).

Egyelőre a kereteket, a *Javat* és a *JavaScriptet* nem támogatja. Kedves jelenség, ahogy – terminálból indított *Dillo* esetében – fájlletöltéskor látszik a terminálon a *wget* bitkolbásza.

A projekt vezetője *Jorge Arellano Cid*, a fő fejlesztők még *Sebastian Geerken* és *Luca Rota*. Honlapjuk a ➲ *http://www.dillo.org* címen található.

Összehasonlítás

Nézzünk egy táblázatot a fájlméretekre (byte-ban), valamint egy nagy méretű (869 kB-os), soktáblázatos, valódi életből vett *HTML* fájl különböző böngészők általi feldolgozási idejének összehasonlítására:

```
0375891   0.908s   netrik
1095544   0.371s   lynx
1137164   1.628s   w3m
1183244   1.207s   links(2)
2907916   5.414s   xlinks.bin
3510889   1.384s   dillo
4492575   2.049s   elinks
9703260   4.062s   firefox-bin
```

Látható, hogy a *lynx* messze a leggyorsabb, az *xlinks* a leggyengébb (és valóban, ha már képeket is akarunk látni, akkor használjunk rendes grafikus böngészőt – de legalábbis egy *dillo*-t), viszont, ami nehezebben fejezhető ki számokkal, a *links(2)* igen szép, vonalakkal határolt táblázatokat mutatott fel, úgyhogy én leginkább azt szeretem (és tudom *SSH* alól) ilyen prózai munkacélokra használni. És mi tagadás: a *Firefox* futási idejével sem kell nagyon szégyenkezni!

Szabó Zoltán
(szz@freemail.hu)

Három gyermekével és feleségével Pannonhalmán él. Tíz éve kísérletezik a Linuxszal. Matematikát és informatikát tanít, diákotthonban keseríti a rábízottak életét. Szívügye a PHP és a PostgreSQL.

KDE alkalmazások (4. rész)
CD/DVD borító – Kover

A CD írás elterjedésével igény merült fel a szép CD borítók nyomtatására. Igazán szép borítóhoz grafikai program, DTP szoftver kell, de egyszerűbb igényeket egyszerűbb programok is ki tudnak elégíteni.

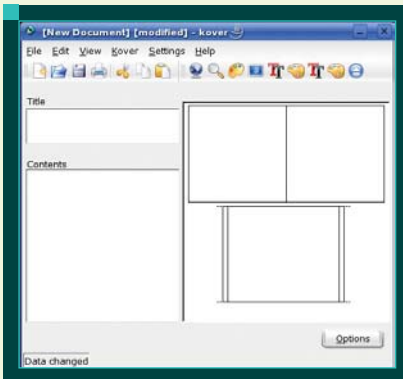
Mi kell a borítóhoz?

Egy átlagos *CD* tokba kell egy félbehajtandó előlap, illetve a hátlap, amelyen lennie kell két felhajtható fül, ezek látszanak a *CD* oldalán. Az előlapot nem fontos félbehajtani, de úgy stabilabban megmarad a tokban. Gyakorlatilag a *Kover* program kezdő oldalán láthatjuk ezeket a lapokat (*1. ábra*). A programot a legtöbb terjesztés tartalmazza, de akár a *KDE-Apps* oldaláról is le tudjuk tölteni, azonban ekkor le kell fordítanunk, hiszen a forrását kapjuk kézhez.

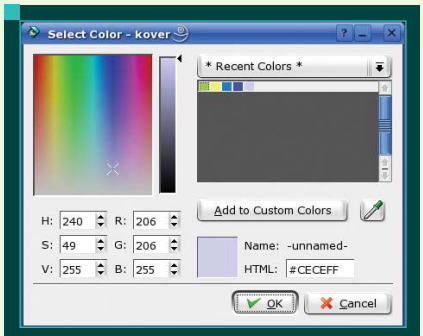
A program menürendszere és ikonjai saját magukért beszélnek, bár egyelőre magyarítás még nem történt, így angol nyelven kell keresgélnünk. Első körben érdemes megnézni a három paletta ikont, amelyek rendre a háttér, a cím, illetve a tartalomjegyzék színe (*2. ábra*). Természetesen háttérszín helyett lehetőségünk van háttérképet is a borítóra tenni, amelyet a háttérszín melletti ikonnal tudunk kivitelezni (*3. ábra*). Szám szerint három képet tudunk kitenni: az előlap bal és jobb (esetleg egyszerre mindkettő) oldalára, illetve a hátlapra. Ez utóbbi esetben a képet tehetjük csak a hátlapra, de akár az oldalsó felhajtandó fülekre is. A képeket tehetjük középre az eredeti méretében, csempeként egymás mellé, illetve nyújtva is (*4. ábra*).

A borítóra kétféle szöveget tudunk írni: címsort és tartalmat. A címsor fog a jobb oldali előlapon és a felhajtható füleken megjelenni, túl sok mozgásterünk nincs, ugyanis az előlapon középre igazítva, a füleken pedig a *CD* tok állásának megfelelően oldalra fordítva fog megjelenni a beírt szöveg (*4. ábra*). A program ránk bízza a megfelelő összhangot a háttérkép és a szöveg színe között, amely egy színekkel teli képen kissé nehézkes, így lehetőleg kerüljük az olyan háttérképet, mint amit az 4. ábrán láthatunk, mivel az előlap és a hátlapon található kék-fekete-zöld-barna színkombinációkon gyakorlatilag egyik szövegszín sem lesz olvasható vagy esztétikus.

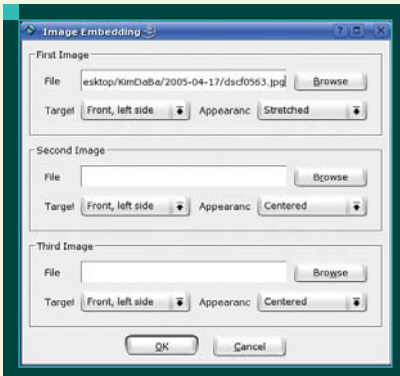
Ahogy az *5. ábrán* láthatjuk, a beírt tartalom balra igazítva a megadott betűtípussal jelenik meg, amely nem túl szerencsés, de azért használható eredményt tud adni egy kis törődéssel. Elsőként a betűtípust kell beállítanunk,



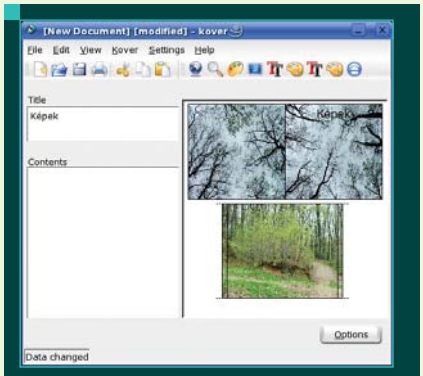
1. ábra Kover kezdő oldala



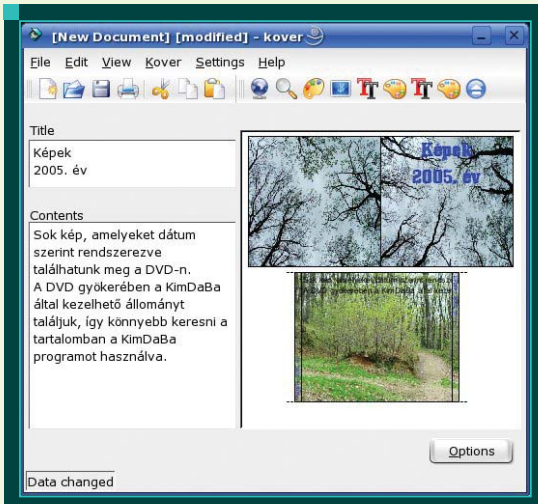
2. ábra Háttérszín beállítása



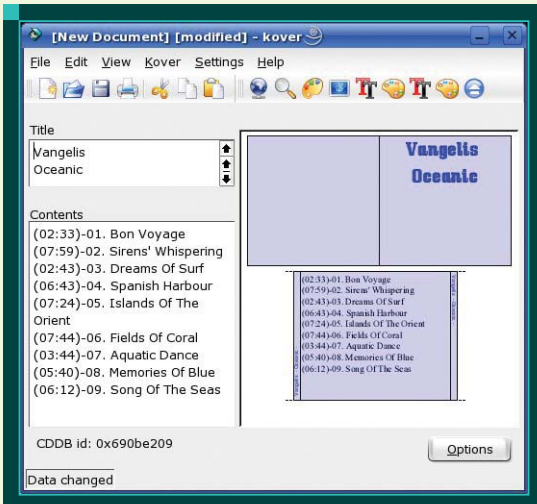
3. ábra Háttérképek megadása



4. ábra Háttérképek és címsor a borítón



5. ábra Tartalom a hátlapon



7. ábra CDDB lekérdezés eredménye

majd a megfelelő méret meghatározása után a sorok végén egy ***Enter*** billentyűt kell ütnünk, hogy a hátlapon is új sor kezdődjön. Módosítás esetén ezt újra el kell végeznünk, de sajnos a program csak ennyit tud.
Ha bármikor kattintunk a borító előnézetén, akkor a program egy külön ablakban megmutatja eredeti méretben a nyomtatási képet, amely csak akkor lesz eredeti méretű, ha jól állítottuk be a monitorunk paramétereit (*6. ábra*).

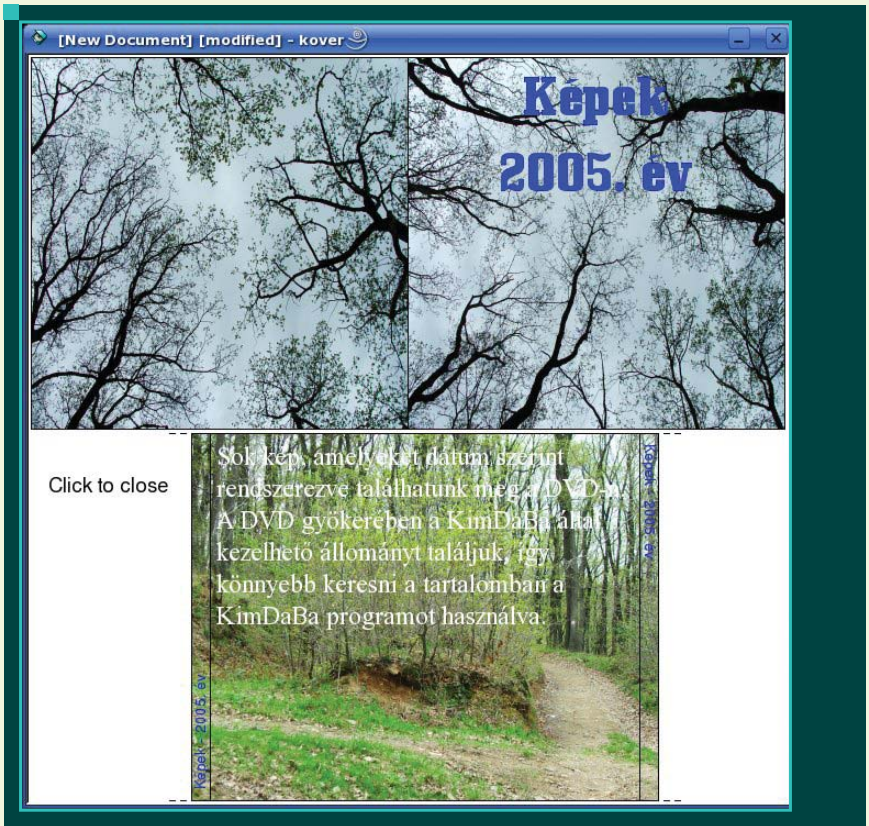
A borítót kinyomtatva egy két oldalas nyomatot kapunk, amely mérethelyesen, kivágható formában tartalmazza a CD borítóját, azokkal a szövegekkel és színekkel, amelyeket megálmodtunk.

CDDB lekérdezés

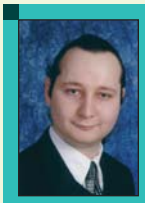
Zenei ***CD*** borítóját az eredeti hiányában (vagy lapolvasó nélkül) egy ***CDDB*** letöltéssel tudjuk pótolni, ekkor a program „hálózattal ellátott földgömb” ikonján kell kattintanunk, amely a gépünk első ***CD/DVD*** meghajtójában lévő ***CD*** adatait lekérdezi a publikus ***CDDB*** adatbázisból, s ki is tölti a borító mezőit, amelyet – természetesen – módosíthatunk utólag (*7. ábra*).

Zárszó

Bár a program nem sok mozgásteret ad, egyszerűbb borítókat el lehet vele készíteni mérethelyesen, s ezt könnyedén meg tudjuk tenni. Ha ez kevés, akkor a ***GIMP*** nyújt megfelelő hátteret az összetett grafikai műveletekhez, s a szövegeket az ***OpenOffice.org*** rajzoló modulja kényelmesen kezelhetővé teszi. A programot fogadjuk megértéssel, s ne várjunk túl sokat: amire készült, arra tökéletes.



6. ábra A borító eredeti méretben



Auth Gábor
(auth.gabor@enaplo.hu)

Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

KDE projekt oldala:
➲ http://www.kde.org

A Kover oldala:
➲ http://kde-apps.org/content/show.php?content=9968

Linux alapú , több terabájt kapacitású háttértár építése otthon

Építsünk alacsony költségű, Linux alapú, terabájt méretű háttérkiszolgálót digitális képeink, audio és videóanyagaink tárolására.

A terabájt méretű, vagy ennél is nagyobb otthoni háttértárak építése a *Linux* felhasználók számára immár elérhető alternatíva az adatmentésben. Ebben a cikkben egy ilyen rendszer építésével, beállításával és használatával kapcsolatos lehetőségeket fogjuk áttekinteni.

A kiszolgáló megtervezése

Manapság a kifejezetten nagy kapacitású merevlemezek szinte elképesztően alacsony áron szerezhetők be, legalábbis ha a mindössze néhány évvel ezelőtti árakhoz hasonlítjuk a mai költségeket. Ezzel párhuzamosan, köszönhetően elsősorban a *Linux* multimédiás képességek területén tapasztalható fejlődésének, egyre több *Linux* felhasználó foglalkozik zenei CD-k tömörítésével, digitális fényképezéssel és filmezéssel, filmvágással és feldolgozással (például a *MythTV* segítségével). Így érthető, hogy egyre nagyobb az igény a nagy kapacitású háttértárakra is, hiszen az elkészült adatok biztonságos mentése és tárolása immár kritikus elvárás egy ilyen célú rendszer esetében. Ha egy merevlemez meghibásodása miatt elveszítjük a gyermekeinkről készült képeket vagy filmeket – vagy egyszerűn csak a nagy gonddal összeválogatott zenegyűjteményünket – az egyenesen katasztrófa. Szerencsére ma már egész egyszerűen megépíthetünk egy a biztonsági mentéseket tároló nagy kapacitású háttérkiszolgálót linuxos szoftverekkel, és egyszerű, bármelyik számítástechnikai szaküzletben megvásárolható olcsó merevlemezekből. Gyakorlatilag bármilyen otthoni felhasználásra szánt *PC* alkalmas lehet arra, hogy egy ilyen kisegítő háttérkiszolgáló szerepét betöltse. Ha viszonylag ritkán kell adatainkat menteni, és egy-egy frissítés nem igényli különösebben sok anyag átvitelét, akkor a számítógép sebessége feltehetőleg nem lesz akadály. Ugyanakkor győződjünk meg róla, hogy a hálózatunk képes a rendelkezésre álló idő alatt átvinni a szükséges adatmennyiséget. Az öregebb gépek esetébe a szűk keresztmetszetet leginkább maguknak a lemezeknek az adatátviteli sebessége (30-1510 Mbps a technológiától függően) jelentheti.

Ügyeljünk rá, hogy számos, általános felhasználásra tervezett számítógép hűtési kapacitása csupán két merevlemezhez elegendő. Persze a legtöbb alaplap eleve legfeljebb négy merevlemez csatlakoztatását teszi lehetővé. (Ez elsősorban a csak *ATA*/*IDE* lemezeket támogató rendszerekre érvényes, manapság azonban egyre gyakoribb a két *ATA*/*IDE* plusz két *SATA* csatlakozó megléte.) Használhatunk külső, *USB* portra csatlakoztatható lemezeket is. Öregebb géphez, amiben csak *USB1*-es csatlakozó van érdemes venni egy *PCI USB2* kártyát, ami tízszer gyorsabb.

A *SCSI* rendszereknél sokkal kevesebb korlátozással kell számolnunk, viszont ezek kifejezetten drágák, illetve maga a technológia kicsit összevissza fejlődik, rengeteg változattal és kitérővel. A lemezek számát legegyszerűbben egy úgynevezett *HBA* kártya (*Host Bus Adaptor*) segítségével növelhetjük. Ilyeneket gyárt például a *Promise Technology*. A *HBA* egy olyan *PCI* sínre csatlakozó lemezvezérlő, amelynek rendszerint semmiféle kiegészítő szoftverre nincs szüksége a működéséhez, és mivel saját *BIOS*-a van, így a *PC BIOS*-ában esetleg meglévő korlátok (például bizonyos méretnél nagyobb lemezek kezelésének képessége) sem gátolják. Egy *HBA* kártya segítségével tehát olyan gépben is használhatunk 120 GB-nál nagyobb lemezeket, amelyek elvileg nem lennének képesek ezek kezelésre,

1. ábra Egy öreg kiszolgálóból épített lemeztömb (kilenc IDE lemezt tartalmaz, amelyek közül öt egy SCSI RAID tagja) A kiegészítő IDE csatolók egy Promise HBA kártyán találhatók.

1. táblázat *Néhány lehetséges összeállítás a kisegítő fájlszerverre, az egy gigabájtra vetített költséggel együtt*

Típus	Konfiguráció	Kapacitás (TB)	Ár/Gigabájt ($)
ATA/SATA lemezek	Belső lemez	0.4	0.56
Linux asztali gép*	Három belső lemez	1.2	0.84
Linux asztali gép*	Három belső plusz két külső USB lemez	2.0	0.73
LaCie 2TB Storage	Speciális célú hálózati kiszolgáló	2	1.15
Linux kiszolgáló**	Hat belső lemez	2.4	1.21
Linux kiszolgáló**	Hat belső és két USB-s külső lemez	3.2	1.08

** Intel Celeron D 478 325 2.53 GHz, 256 MB RAM.*
*** Intel SC5275 ház, Intel ATX alaplap, két darab 3 GHz-es Xeon CPU, 2 GB RAM.*

ATA-33-ról ***ATA-150***-re fejleszthetjük a rendszer képességeit, illetve egyszerre használhatunk ***ATA*** és ***SATA*** lemezeket.

Megfontolhatjuk egy dedikált fájlkiszolgáló beszerzését is. Egy hat lemez fogadására képes úgynevezett barebone kiszolgálót (összeszerelve, de lemezek és operációs rendszer nélkül) körülbelül 1500 dollárért megvehetünk. Ezzel a viszonylag jelentősebb kezdeti beruházással gigabájtonként körülbelül 0,8 dollárért bővíthetjük a rendelkezésre álló lemezterületet, illetve ha elfogyott a hat hely, akkor használhatunk külső ***USB***-s lemezeket is. Ha eldöntöttük, hogy hány lemezre lesz szükségünk, a következő lépés a helyigén, a zaj és a szükséges hűtés felmérése. Az *1. ábrán* egy olyan otthoni fájlkiszolgálót láthatunk, aminek az alapját egy kiöregedett otthoni gép szolgáltatta. A kapacitása jóval 1 gigabájt fölött van.

Még ha úgy döntünk, hogy saját kezűleg építjük meg a fájlkiszolgálónkat, akkor is egészen alacsonyan tartható az egy gigabájtra vetített költség. Ennek persze leginkább az az oka, hogy a merevlemezek ára az elmúlt időben drasztikusan esett. Az *1. táblázatban* felsoroltam néhány lehetséges konfigurációt, a hozzájuk tartozó egy gigabájtra vetített költséggel együtt. (Az árak természetesen csak becsült értékek, és nem tartalmazzák a szállítási költséget illetve az adókat.) Amint látható, egy 2 terbájtot is meghaladó fájlkiszolgáló egy gigabájt tárterületre vetített költsége nem haladja meg az 1,50 dollárt, miközben egy ekkora lemeztömbön már igazán jókora mennyiségű filmet, digitális képet és zeneszámot tárolhatunk.

A kiszolgáló megépítése és a lemezek behelyezése

Jómagam az elmúlt néhány évben több Red Hat Linux 9 alapú háttérkiszolgálót építettem, de természetesen használhatunk bármilyen más terjesztést is. Az én választásom azért esett éppen a Red Hat-re, mert stabil, szabadon hozzáférhető, és – a Fedora Legacy Projectnek köszönhetően – folyamatos a frissítése. Telepítése és beállítása is egészen egyszerű. Aki új számítógépet vásárol, annak persze érdemesebb valamilyen ennél frissebb terjesztéssel próbálkoznia. Az alacsony költségvetéssel készülő fájlkiszolgálókon én nem szoktam ***RAID***-et használni, mivel itt az ár – értelemszerűen – elsődleges szempont. Ugyanakkor érdemes ezen a lehetőségen is elgondolkodni.

Ami a szoftvert illeti, egy linuxos háttérkiszolgáló igényei nyugodtan nevezhetők minimálisnak. Amire – az alapokon kívül – biztosan szükségünk lesz, az az alapvető hálózati adminisztráció, beleértve az ***SSH***-t is, valamint az ***rsync***. Utóbbi egy gyors, inkrementális másoló/szinkronizáló segédprogram, amit a legtöbb ***Linux*** terjesztés eleve tartalmaz. Az ***SSH*** és az ***rsync*** segítségével gyakorlatilag bármilyen alapvető adatmentési feladat elvégezhető. Hasznos, ha a kisegítő kiszolgáló egyben fájlkiszolgálóként is működik, így én mindig telepítem rá a ***Samba***-t is. Azért ezt a megoldást használom, mert az ***MS Windows*** ügyfelek egyrészt alapértelmezésként ***SMB***-n keresztül kommunikálnak, másrészt a megfelelő ***Samba*** ügyfél segítségével az így létrehozott hálózati megosztások bármilyen ***UNIX*** alapú rendszerről (még a ***Mac OS X***-et is beleértve) elérhetőek. Akinek homogén ***UNIX***-os hálózata van, az használhat ***NFS***-t is, de ezt a lehetőséget itt részletesen nem tárgyalom.

Új partíciók létrehozása

Ha a `hdb` eszközhöz tartozó lemezen akarunk új partíciókat kialakítani, adjuk ki a következő parancsot:

```
fdisk /dev/hdb
```

Az `fdisk` promptjánál gépeljük be az `m` parancsot, amely a súgót jeleníti meg. Itt is látható, hogy új partíciót az `n` segítségével hozhatunk létre, előbb azonban válaszolnunk kell néhány kérdésre:

```
Command action
   e   extended
   p   primary partition (1-4)
p
```

Ha az egész lemezen egyetlen partíciót akarunk kialakítani, használjuk a `p` parancsot:

```
Partition number (1-4):1
```

A program ekkor megkérdezi az elsődleges partíció sorszámát. (Ha egyetlen partíciót hozunk létre, akkor a válasz 1 kell legyen.) Határozzuk meg a partíció méretét az első ás utolsó cilinder megadásával. Tekintve hogy mi most a teljes lemezt akarjuk fölhasználni, nincs más dolgunk, mint elfogadni az alapértelmezett értékeket.

```
First cylinder (1-48641, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-48641, default
➥48641):
Using default value 48641
```

A partíciós tábla frissítéséhez adjuk ki a w parancsot. Ezzel létrehoztuk a /dev/hdb1 eszközt, amely egy a teljes lemezt elfoglaló partíciónak felel meg.
Ezután válasszuk ki a céljainknak megfelelő fájlrendszertípust (én az *ext3*-mal szoktam használni), majd formázzuk meg a lemezt az `mkfs` parancssegítségével:

```
mkfs -t ext3 /dev/hdb1
```

Aztán hozzuk létre a csatolási pontot (esetben `/data1`)

```
mkdir /data1
```

majd csatoljuk be az új lemezrészt:

```
mount -t ext3 /dev/hdb1 /data1
```

Ellenőrizzük az írhatóságát és olvashatóságát, végül pedig szúrjuk be az automatikus befűzéséhez szükséges sort a */etc/fstab* fájlba:

```
# Device   mountpoint  fstype   options      freq   pass_no
/dev/hdb1  /data1      ext3     defaults       1        2
```

A lemezek csatlakoztatása előtt először is győződjünk meg róla, hogy van elegendő tápcsatlakozónk és (*IDE*/*SATA*/*SCSI*) adatkábelünk. Esetleg érdemes azt is ellenőrizni, hogy a beépítendő lemez kompatibilis-e a ***Linuxszal*** (a legtöbb igen). Kapcsoljuk ki a gépet, és húzzuk ki a tápkábelt is. Csatlakoztassuk a lemezeket, majd kapcsoljuk be újra a rendszert. A ***Linuxnak*** automatikusan fel kell ismernie az új elemeket. Ha ez valamiért mégsem történik meg, akkor vagy speciális meghajtóra lesz szükségünk, vagy a kérdéses hardverelem nem használható ***Linux*** alatt. A bootolás közben megjelenő üzeneteket a `dmesg` paranccsal hívhatjuk elő újra. Egy sikeresen felismert IDE lemezzel kapcsolatos üzenet valahogy így néz ki:

```
hdb: ST3400832A, ATA DISK drive
```

Minden *IDE*/*ATA* (sőt néhány *SATA*) merevlemezhez egy `hdx` névszerkezetű eszközön keresztül férhetünk hozzá, ahol x az ábécé valamelyik betűje (esetünkben b). Hasonlóan az ***USB***, *SCSI* és egyes *SATA* merevlemezek csatlakoztatása után a bootolás közben megjelenő üzenetek között egy `sdx` eszközre vonatkozó jelenik meg, ahol x ismét valamilyen betű.
A legtöbb ***Linux*** terjesztés általában tartalmaz valamilyen grafikus felületen át használható lemezkezelő programot. Ezekkel az alkalmazásokkal felosztathatjuk (particionálhatjuk) a merevlemezt, illetve meg is formázhatjuk az így kialakított lemezrészeket (jómagam minden háttérkiszolgálóba helyezett lemezen egyetlen partíciót használok), sőt rendelhetünk hozzájuk befűzési pontot is (például */**data1***, */**data2***). Mindezt természetesen parancssorból is megoldhatjuk az fdisk és egyéb parancsok segítségével.

A szoftverkomponensek beállítása – rsync és SSH

Az *rsync* program a legtöbb ***Linux*** terjesztésnek része. Erre és az *SSH* rendszerre az ügyfélen és a kiszolgálón egyaránt szükségünk lesz. Azt, hogy az *rsync* telepítve van-e, a legegyszerűbben úgy ellenőrizhetjük, ha a parancssorban kiadjuk az `rsync` parancsot, vagy átvizsgáljuk a telepített csomagok listáját. Ha esetleg nem

SSH kulcspár létrehozása

Jelentkezzünk be a megfelelő felhasználó nevében arra a gépre, amelyen a kulcspárral majd hitelesíteni szeretnénk magunkat (jelen esetben tehát Bob-nak kell bejelentkeznie a bar szerverre), majd adjuk ki a

```
ssh-keygen -d
```

parancsot. Ha a kulcsot más felhasználók is használják, vagy olvashatják, akkor adjunk meg jelszót is, ellenkező esetben az ezzel kapcsolatos kérdésre válaszoljunk az ***Enter*** megnyomásával. Lépjünk be a ***.ssh*** könyvtárba, és másoljuk be az engedélyezett kulcsok listájába a most létrehozott nyilvános kulcsot:

```
cd ~/.ssh
cp id_dsa.pub authorized_keys2
```

A titkos kulcsot másoljuk át annak a gépnek a ***.ssh*** könyvtárába, amelyről be fogunk jelentkezni (jelen esetben tehát a `foo` gépen a Bob könyvtára alatt található ***.ssh*** könyvtárba). A kiszolgálóról (bar) egyidejűleg töröljük a titkos kulcsot:

```
scp id_dsa root@foo:~/.ssh/id_dsa
rm id_dsa
```

A bejelentkezéshez használt gépen indítsuk el az ***SSH*** ügynököt, és vegyük fel a most átmásolt titkos kulcsot a listájába (ha az imént megadtunk jelszót is, akkor az ügynök meg fogja azt kérdezni):

```
eval `ssh-agent`
ssh-add
```

És ezzel készen is vagyunk. Ettől kezdve ***Bob*** a ***foo*** gépről a bar kiszolgálóra jelszó nélkül jelentkezhet be:

```
ssh bob@foo
```

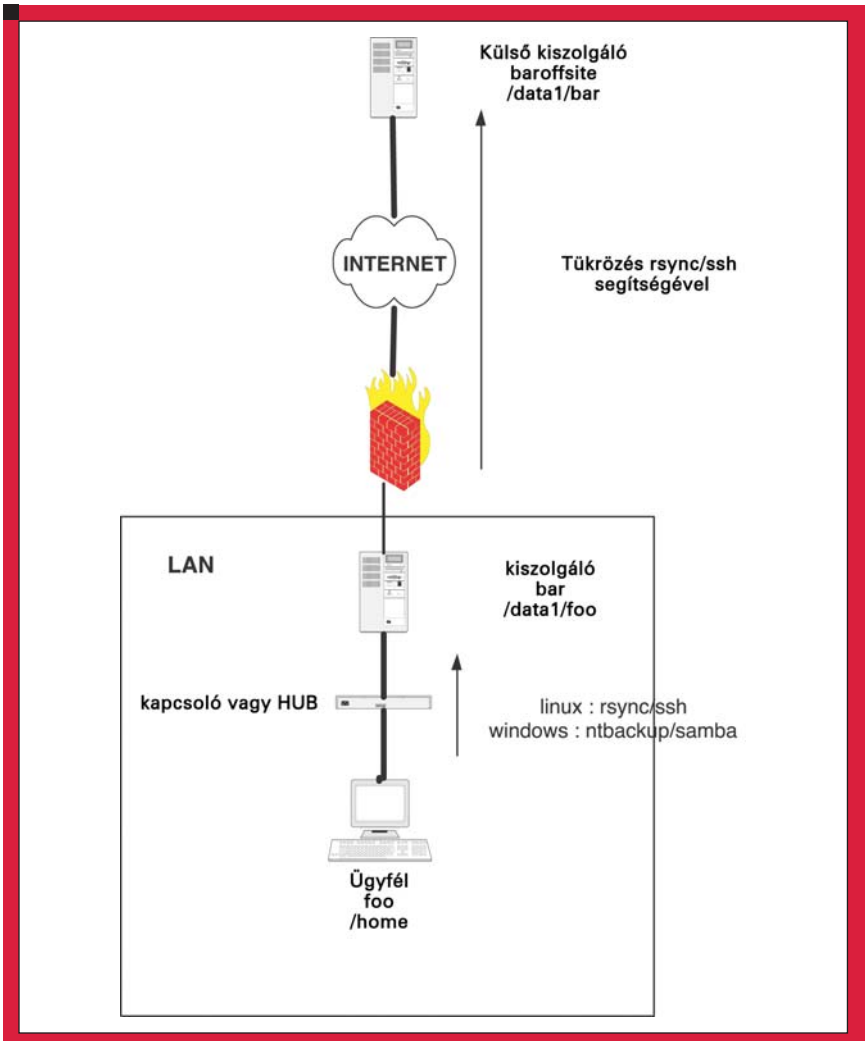
találjuk az ***rsync*** csomagot sem a rendszeren, sem a telepítőlemezen, akkor letölthetjük a forrását is a program hivatalos weblapján található linkeket követve (lásd a kapcsolódó címeket). Az ***rsync*** biztonságos futtatásának legegyszerűbb módja az, ha önálló alkalmazásként indítjuk, a titkosításhoz pedig az ***SSH***-t használjuk. A program ugyan futtatható démonként is, sőt így a szolgáltatásainak köre is valamivel bővebb, de erre esetünkben nem lesz szükség. A következőkben a parancssori program használatát fogom bemutatni. A példákban az ügyfelet ***foo***-nak, a kiszolgálót pedig ***bar***-nak hívják.
Ha a ***foo*** nevű, ***Linuxot*** futtató gép ***/home*** könyvtárát akarjuk átmásolni a bar kiszolgáló ***/data1/foo*** könyvtára alá az `rsync` és ***SSH*** kombinálásával, gépeljük a következőt:

```
rsync -az  /home  -e ssh
➥bob@bar:/data1/foo
```

A rendszer kérni fogja bob jelszavát, majd a `foo` gépen található `/home` könyvtárait átmásolja a `bar`-on található `/data1/foo/home` könyvtárba. (Ehhez Bobnak természetesen rendelkeznie kell egy felhasználói fiókkal a kiszolgálón, illetve írási joga kell legyen a ***/data1/foo*** könyvtárra.)
Ha el akarjuk kerülni, hogy minden egyes alkalommal be kelljen gépelnünk a kérdéses jelszót, hosszunk létre egy jelszó nélküli nyilvános/titkos kulcspárt, és azt használjuk az ***SSH*** hitelesítéshez.
A mentést természetesen nem csak kézzel végezhetjük el, hanem írhatunk rá egy szkriptet is, ami a ***foo*** gépen lefuttatva elvégzi a mentett anyag frissítését. Ehhez azonban előbb célszerű elolvasni az ***rsync*** dokumentációját, hiszen a program több mint 70 parancssori kapcsolója van. Különösen a `-delete` opcióhoz tartozó részt olvassuk el figyelmesen, ennek nem megfelelő használata ugyanis katasztrofális következményekkel járhat. Az *1. Lista* egy hét napos inkrementális mentést megvalósító

1. Lista *Teljes és inkrementális mentés az rsync segítségével*

```
#!/bin/sh
# Ez a szkript menti a foo gép
kijelölt anyagait a bar
kiszolgálóra.
# A mentés módja: 7 napos,
inkrementális.
# A program Andrew Tridgeld
eredeti kódja alapján készült.
# A menteni kívánt könyvtár
BDIR=/home
# Célkönyvtár a kiszolgálón
BACKUP_HOME=/data1/foo
# A mentés során használandó
felhasználói fiók
BACKUP_LOGIN=bob
# A kiszolgáló neve
BSERVER=bar
BACKUPDIR=`date +%A`
OPTS="-force -ignore-errors
➥-delete -backup
-backup-dir=$BACKUP_HOME/
➥$BACKUPDIR -av"
export PATH=$PATH:/bin:/usr/
➥bin:/usr/local/bin
# A kimenetet egy fájlba
irányítjuk
date > /var/log/
➥backup.$BACKUPDIR.log
# Ez a sor törli az előző héten
végrehajtott
# inkrementális mentésekhez
tartozó könyvtárat
[ -d /tmp/emptydir ] || mkdir
➥/tmp/emptydir
rsync -delete -a /tmp/emptydir/
BACKUP_LOGIN@$BSERVER:
➥$BACKUP_HOME/$BACKUPDIR/
rmdir /tmp/emptydir
# Itt kezdődik a tényleges
átvitel
rsync $OPTS $BDIR BACKUP_
➥LOGIN@$BSERVER:$BACKUP_HOME/
➥current >>
/var/log/backup.$BACKUPDIR.log
```



■ *2. ábra* Példa külső telephelyen elhelyezett adatmentő kiszolgálóval. A helyi ügyfél foo, a helyi adatmentési kiszolgáló bar, a külső telephelyen elhelyezett tükörszerver pedig baroffsite.

szkriptet mutat. Azok a fájlok, amelyeket módosítottunk vagy töröltünk az adott napnak megfelelő könyvtárba kerülnek (utóbbit a `-backup-dir` kapcsolóval lehet beállítani), a legfrissebb mentést pedig mindig a *current* nevű könyvtárban találjuk.
Aki tömörített formában szeretne mentést készíteni, az természetesen a *tar*-t is használhatja a hálózati mentéshez a következőképpen:

```
tar cvfz - /home | ssh bob@bar
➥dd of=/data1/foo/
➥current.tar.gz
```

Inkrementális mentéshez ilyenkor a `-newer` kapcsolót kell használnunk. Mindazonáltal az *rsync* jóval hatékonyabb, mint a tar, mert előbbi csak a jelenlegi és az előző állapot közti különbségeket másolja át.

Összességében bár az *rsync* és az *SSH* segítségével kiválóan elboldogulhatunk még *MS Windows* környezetben is, azért valós körülmények között sokkal célszerűbbnek tűnik egy mentési célokra szolgáló fájlkiszolgáló üzembe helyezése. Ez különösen akkor célszerű, ha a hálózatban sok windowsos ügyfél található. Ilyen esetben nyilván olyan szkriptet kell írnunk, amely a *Samba* kiszolgáló egy megadott megosztására írja a menteni kívánt adatokat.

Samba fájlkiszolgáló üzembe helyezése

Ha az általunk használt *Linux* terjesztés támogatja az *SMB*-n keresztüli fájlmegosztást, akkor a *Samba* rendszer minden valószínűség szerint alapértelmezésként települt. Ha mégsem így lenne, vagy valamiért a *Samba* legfrissebb kiadását szeretnénk használni, töltsük le a forráskódját, és fordítsuk le. Mind megannyi más programnál, itt is rendelkezésre állnak különböző bináris csomagok is (lásd a kapcsolódó címeket). A fordítás és a kezdeti beállítás menetéről a dokumentációban olvashatunk.
Amint feltelepítettük a csomagot, a *Samba* kiszolgáló valamennyi beállítását az *smb.conf* megfelelő átszerkesztésével végezhetjük el. Ez a fájl rendszerint a */etc/samba/smb.conf*, vagy a */usr/local/samba/lib/smb.conf* helyen található. Léteznek grafikus felülettel rendelkező beállítóprogramok is. Ilyen például a *SWAT*, amit a legtöbb *Linux* terjesztés szintén tartalmaz. A *Samba* indításával és leállításával kapcsolatos műveletekről a program dokumentációjában olvashatunk. A legegyszerűbb nyilván az, ha úgy állítjuk be rendszerünket, hogy a gép indulásakor a *Samba* kiszolgáló is automatikusan elinduljon.
Visszatérve korábban említett mentési példánkhoz, a bar kiszolgálón – amelyen immár a *Samba* is telepítve van – hozzunk létre egy egyszerű *smb.conf* fájlt, vagy másoljuk be egy már létező végére a következő szakaszt:

```
[bob]
   comment = foo backup account
   path = /data1/foo
   valid users = bob
   public = no
   writable = yes
```

A következő lépésben hozzunk létre egy `bob` nevű *Samba* felhasználót egy kellően biztonságos jelszóval ellátva. (Ne felejtsük el azt sem, hogy bobnak egy közönséges felhasználói fiókkal is rendelkeznie kell, illetve írási és olvasási joga kell legyen a */data1/foo* könyvtárra:

```
smbpasswd -a bob
New SMB password: somepassword
Retype new SMB password:
somepassword
Added user bob
```

A *Windowst* futtató ügyfeleken hozzunk létre egy a *\\bar\bob* megosztásra mutató hálózati meghajtót (network drive) természetesen a bob bejelentkezési névvel és bob *Samba* fiókjának jelszavával „felszerelve”. Én általában

2. Lista *Adatmentés külső telep helyre az rsync segítségével*

```
#!/bin/sh
# Mirror /data1 on bar to
/data1/bar on baroffsite.
#Backup directory on bar
BACKUP=/data1
#Backup directory on baroffsite
BACKUP_OFF=/data1/bar
# Give the day of week as name
of backup
BACKUPNAME=`date +%A`
# Offsite server
BSERVER=baroffsite
# Backup account on backup
server
BAC_ACC=backup
date >  /var/log/
➥backup.$BACKUPNAME.log
/usr/bin/rsync -avz -delete -e
➥ssh $BACKUP
➥$BAC_ACC@$BSERVER:$BACKUPOFF
>>  /var/log/
➥backup.$BACKUPNAME.log
# Email the log to
administrator
cat /var/log/
➥backup.$BACKUPNAME.log | mail
➥-s 'Mirror Check'
backup_guy@mycompany.com
```

3. Lista *Egy egyszer időbélyeg- és lemezterület-megfigyelő program*

```
#!/bin/sh
# Szabad hely lekérdezése
minden partíción
# Időbélyegek megjelenítése
időrendi sorrendben
BACKUPS=/data1      # Az
ellenőrizendő könyvtárak
azonosítása
# A mentés neve a megfelelő nap
lesz
BACKUPNAME=`date +%A`
#Timestamp
date >  /var/log/
➥backup.$BACKUPNAME.log
# Lemezterület a partíciókon
df -k > /var/log/
➥backup.$BACKUPNAME.log
echo ' ' >>  /var/log/
➥backup.$BACKUPNAME.log
# Az adatmentő kiszolgáló
időbélyegeinek listája
# Az ls -lRt parancs sokkal
beszédesebb
ls -lt $BACKUPS/* >>  /var
➥/log/backup.$BACKUPNAME.log
# Levélben elküldjük a naplót
a rendszergazdának
cat  /var/log/
➥backup.$BACKUPNAME.log | mail
➥-s 'Backup Check'
backup_guy@mycompany.com
```

a szabadon használható `ntbackup` program segítségével szoktam a kiszolgálóra felmásolni a *.bkf* fájlokat. Az `ntbackup` egyébként része a *Windows 2000* és *Windows XP* rendszernek is, futtatható automatikusan, időzített módon, így minden probléma nélkül megoldható vele a *Windows 2000* és *XP* ügyfelek időszakos mentése. A dolognak ráadásul az az előnye is megvan, hogy ezek a mentések a rendszer teljes állapotát is tartalmazzák, beleértve a regisztrációs adatbázis (registry) is.
A *UNIX* különböző változatait, illetve *Mac OS X*-et futtató ügyfelek mentése szintén zökkenőmentesen megoldható *Samba* megosztások segítségével.
A *Mac OS X*-en alapértelmezésként telepítve van az *smb* ügyfél, a különböző *Linux* terjesztéseknél pedig vagy ugyanez a helyzet, vagy csomagból kell telepítenünk ezt a szoftvert. Ha ez megtörtént, akkor az *smb* megosztást a *foo* gép */backup* könyvtára alá kell becsatolnunk:

```
mount -t smbfs -o username=bob,
➥password=somepassword
➥//bar/foo /backup
```

Ha ezt a műveletet automatikusan szeretnénk elvégeztetni a rendszerrel már a bootolás során, akkor a következőképpen kell módosítanunk a */etc/fstab* fájl tartamát:

```
//bar/data1/foo  /backup  smbfs
➥rw,username=bob,password=some
➥password  0 0
```

Biztonságos adatmentés külső telephelyre

Ha még nagyobb biztonságban szeretnénk tudni adatainkat, készíthetünk róluk másolatot egy olyan külső telephelyen található kiszolgálóra is, amely kellően messze van mind a munka közben használt gépünktől, mind pedig a helyi kisegítő kiszolgálótól. Megtehetjük például azt, hogy a belső, adatmentésre használt kiszolgáló tartalmát hetente egyszer áttükrözzük erre a külső telephelyre.
Ha a belső gépekkel valamiféle katasztrófa történik, például leég az iroda, még mindig lesz egy teljes másolatunk az adattartalomról.
Egy ilyen hálózat elvi vázlatát mutatja a *2. ábra*.
A *2. Lista* egy olyan egyszerű szkriptet mutat, ami a belső kiszolgáló tartalmát áttükrözi a külsőre az `rsync` segítségével. Az ilyen mentéseket nyilván adott időközönként automatikusan érdemes végrehajtani. Szintén fontos, hogy a mentés folyamatáról napló készüljön, illetve hogy ennek tartalmát rendszeresen ellenőrizzük is.

Az adatmentő kiszolgáló megfigyelése

Egy a cikkben vázolt adatmentési rendszerhez mindenképpen tartoznia kell egy megfigyelő programnak is, ami egyrészt ellenőrzi, hogy az ütemezett mentések valóban a tervnek megfelelően történnek-e, felügyeli magát a folyamatot, és megfigyeli a mentés céljaként használt gép állapotát (például hogy nem telt-e be a merevlemez). A *3. Lista* egy ilyen megfigyelést végző kódot mutat, ami a `cron` segítségével szintén szabályos időközönként fut le, és jelentést küld a végrehajtott mentési műveletekről, illetve a pillanatnyilag rendelkezésre álló lemezterületről.

Linux Journal 2006. 141. szám

Duncan Napier
Brit Columbiában, Vancouver területén dolgozik számítógépes és műszerezési tanácsadóként.

KAPCSOLÓDÓ CÍMEK

A cikkhez tartozó források a következő címen találhatók:
➲ www.linuxjournal.com/article/8635

Hagyományos adatbázis lecserélése LAMP-ra

Néhány hagyományos adatbázis-alkalmazás szinte kínálja magát, hogy cseréljük le LAMP alapú webes alkalmazásra. Richard Hulse elmeséli, hogy az Új-Zélandi Rádió hogyan hajtott végre egy ilyen átállást.

Az ***Új-Zélandi Rádió*** egy nyilvános rádióadó, és mint minden ilyen, hatalmas zene- és műsortárral rendelkezik. 1987-ban egy új, számítógépes könyvtár-katalogizáló rendszerre bíztuk a könyvtáradatok tárolását, ez volt a ***BRS***. A ***BRS***-t konkrétan a hanglemezek – és később a CD-lemezek –, a szalagok, az élő koncertfelvételek, az interjúk és a klasszikus zenei gyűjtemény adatainak tárolására használtuk. A rendszert eleinte egyszerű terminálokon, később PC-ken futó terminálemulátorokon keresztül értük el. Munkatársaink ugyancsak ezt a rendszert használták a ***Concert FM*** – a cég klasszikus zenei hálózata – műsorainak szerkesztésére és követésére.

A ***BRS*** egy zárt katalógusprogram volt, a ***Maxwell Online Inc.*** árulta. ***UNIX***-on futott, és hosszú életet élt meg. 16 év alatt átesett néhány hardverbővítésen, míg szoftveres oldalon néhány újabb adatbázistábla hozzáadásával elégítettük ki az újabb típusú adatok tárolására vonatkozó igényeket. A ***BRS*** – minden ezzel ellentétes várakozástól függetlenül – a legkisebb gond nélkül élte túl a 2000-ik évet, és valamikor 2003-ban kezdtünk komolyan foglalkozik lecserélésének gondolatával.

Csináld magad!

Korábban egy ilyen jellegű átállási feladat végrehajtását alighanem külső cégre bízták volna. Tapasztalati tény ugyanakkor, hogy ilyenkor az eredmény egy zárt forrású, egyedi alkalmazás, valamint a módosítások és a frissítések terén a kötöttség valamely céghez. Amikor aztán ezek a cégek megszűnnek, alkalmazottaik pedig szétszélednek, a nélkülözhetetlen alkalmazás árván marad, az adatokat pedig roppant nehéz egy új alkalmazás alá áttelepíteni. A saját kivitelezésű munkák viszont nem mindig eredményesek, ezért gondosan mérlegeltünk minden szempontot. Az adatok és az alkalmazás kritikus szerepe, illetve a házon belüli erőforrások rendelkezésre állása alapján végül úgy döntöttünk, hogy magunk vágunk bele.

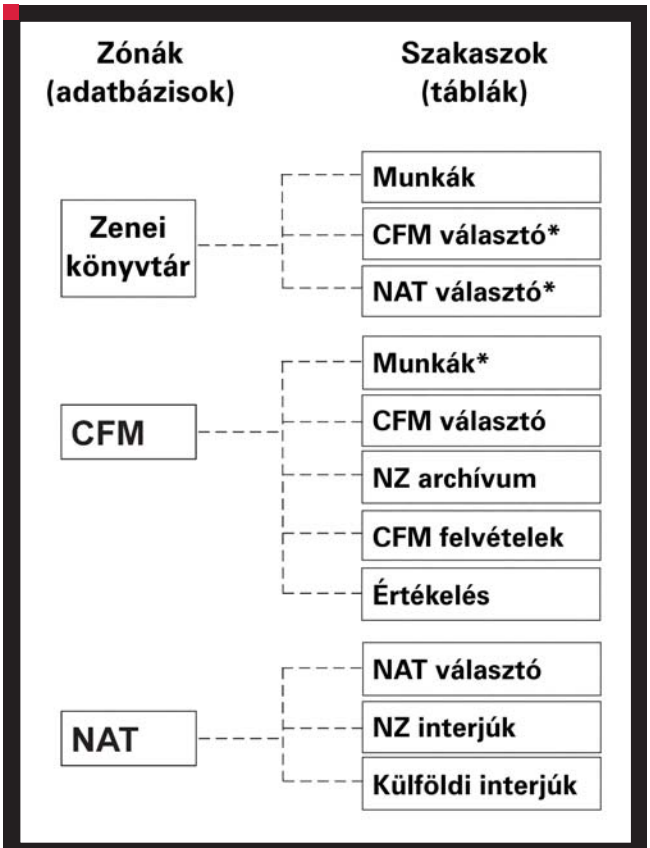
Helló, BRAD

Informatikai részlegünk vezetője, ***Bruce Intemann*** lett a tervezet vezetője, ő állított össze egy kísérleti rendszert egy asztali PC-n, melyen ***Red Hat Linux 8***, ***Apache*** webkiszolgáló, ***MySQL*** és ***PHP*** (együttesen ***LAMP***) futott. ***Bruce*** kitalálta, hogyan lehet ***BRS*** alól egyszerű szöveges formátumba kimenteni az adatokat, valamint készített egy egyszerű, a ***MySQL*** teljes szöveges indexére alapuló keresőt. Az adatok egy kisebb részét kézzel átalakította. A hozzáférést egy normál webkiszolgálón keresztül biztosítottuk. Nagyjából ekkor fejeztem be egy ***PHP***-s, webes munkát a cég egy másik részlegének, és felajánlottam segítségemet az új tervezet végigviteléhez. Amikor elértünk oda, hogy nevet kellett adni a rendszernek, arra gondoltunk, hogy a ***B*** és az ***R*** betűt meg kellene tartani, ezzel emlékezve a „szülők” nevének kezdőbetűire. Feleségem a ***BRAD*** névvel állt elő, munkatársaim egyike pedig kitalálta, hogy a név bizonyára a ***Bruce and Richard's Audio Database (Bruce és Richard zenei adatbázisa)*** elnevezés rövidítése. A név ezzel meg is ragadt.

A próbarendszert hamarosan elfogadtuk, ezért írtam egy rövid ***Perl*** parancsfájlt, melynek feladata az összes adat feldolgozása – körülbelül kétszázezer rekordról van szó – és a ***MySQL*** adatbázisba való beillesztése volt. A feladat meglehetősen összetett volt, mert a globális keresések segítése céljából a kisebb adatbázisok közül sokat beolvasztottak a fő, ***Works*** nevű adatbázisba. Szerencsére egy külön mezővel jelölték meg az eredeti adatok helyét, forrását. A ***BRS*** által tárolt adatok formátumára az első kódrészlet mutat példát. Miután sikerült áttöltenünk az adatokról egy teljes pillanatfelvételt, objektumorientált ***PHP***-ben újraírtam ***Bruce*** kódját. Felhasználtam a másik tervezethez készített kereső osztályomat is, amit hírek helyett zenei adatok megjelenítésére módosítottam.

A nagyjából kész bemutató rendszert egy fejlesztési kiszolgálóra telepítettük, majd megkértük a többieket, hogy mondják el róla véleményüket. Válaszaik alapján úgy döntöttünk, hogy a visszajelzések szerint folyamatosan fogjuk fejlesztgetni a már üzembe helyezett rendszert. A kettősséggel biztosítható a munkatársak hozzáférése a működő rendszerhez, valamint lehetővé válik a két megoldás által adott keresési eredmények összehasonlítása. Szintén fontos volt, hogy az új rendszer elnyerje a személyzet bizalmát.

Mivel az adatokat az eredeti csoportokra akartuk szétosztani, egy összetettebb parancsfájlt is készítettünk, mely kihámozta az eredeti adatforrásokat a ***BRS***-ből. A kapott halmazokat különböző adatbázisokba és táblákba



1. ábra A BRAD adatforrásainak elrendezése. A csillaggal jelölt tábla egy a jelenlegi zónán kívül található táblához vezető álnevet jelez.

helyeztük, ez látható az *1. ábrán*. A *BRAD* alatt a cég minden részlege egy-egy zónának számít, továbbá minden adatforrást külön részként kezelünk. Bármelyik zóna rendelkezhet álnévvel a más zónákba tartozó részekhez, illetve lehetőséget kaphat tetszőleges táblákban való keresésre, függetlenül azoknak a rendszerben elfoglalt helyétől.
A *BRS* adatbázisa egyszintű, nem relációs volt, és az évek során sokféle személy sokféle formátumú adatot rögzített benne. Amikor átnéztem az új rendszerbe kerülő pillanatfelvételeket, a *Perl* parancsfájl módosításával megszüntettem néhány, elsősorban a dátum mezőket illető rendellenességet. Előfordult például, hogy az utolsó frissítés dátumának szöveges mezőjét valaki kézzel múltbéli időpontra módosította – a *BRAD* alatt ez egy a rendszer által karbantartott datetime, típusú mező.
A rekordokat újabb, a létrehozás időpontját tároló mezővel is bővítettük.

A BRAD és a nyílt forrás találkája

A *BRAD* fejlesztését egy *LAMP*-ot futtató kiszolgálón végeztük, és magától értetődőnek tűnt, hogy a fejlesztés során nyílt forrású *PHP* osztályokat használjunk. Az adatbázis elérésére, az űrlapok előállítására és feldolgozására, valamint az alapszintű hibakezelésre *PHP Extension and Application Repository (PEAR)* modulokat alkalmaztunk. Egy meglévő hibakezelő osztályt úgy módosítottunk, hogy figyelmeztessen a hibákra, de a teljes hibaüzenetet ne jelenítse meg a felhasználóknak.

1. kódrészlet Egy BRS alól származó rekord eredeti szöveg formátumban (Original Text Format)

```
*** BRS DOCUMENT BOUNDARY ***
..Document-Number:
      000080019
..TI:
      Ode for the centenary of Trinity College
      Dublin, Great parent, hail to thee (Z327)
..MA:
      Hyperion
..CA:
      CDA 66476
..ME:
      cd
..RA:
      The King's Consort
..CF:
      vocal - ode
..CP:
      PURCELL
..CD:
      Robert King
..SO:
      Gillian Fisher, Evelyn Tubb (sopranos),
      James Bowman, Nigel Short (counter-tenors),
      Rogers Covey-Crump (high tenor), John Mark
      Ainsley (tenor), Michael George, Charles
      Pott (basses)
..ST:
      T12-21
..AT:
      Purcell - Complete odes and welcome songs
      vol 5
..DU:
      002419
..PT:
..RD:
      2-4 Jan 1991
..RE:
..AD:
..SR:
..RO:
..CY:
      1694
..LI:
      Nahum Tate
..OR:
..LN:
..PU:
..RI:
..ED:
..LP:
..LQ:
      cp
..LQ:
..NO:
..IS:
..LD:
      921012
..LU:
      leander
*** BRS DOCUMENT BOUNDARY ***
```

1. táblázat *A BRAD által használt nyílt forrású modulok*

PEAR::DBM	Adatbázis-hozzáférés
PEAR::HTML:QUICKFORM	A szerkesztő felület űrlapjai
PATUSER	Felhasználókezelés és a szerkesztői hozzáférések szabályozása
Hibakezelő osztály	Erősen módosítottuk, hibaüzenetek helyett az oldal elsődleges tartalmát jeleníti meg.
Lapozó	Az eredmények lapozását teszi lehetővé. Módosításával lehetővé tettük az URL osztályon belüli feldolgozását.

Amikor szükségem volt valamilyen függvényre, először mindig megpróbáltam egy nyílt forrású modult keresni, csak ezt követően fogtam bele saját kód írásába. Ezzel a módszerrel rendkívüli mértékben felgyorsult a fejlesztés. (A ***BRAD*** által használt modulok listáját az *1. táblázat* tartalmazza.)

Várakozások és eredmények

Mivel a ***BRS***-t hosszú időn keresztül megtartottuk, munkatársaink komoly rutint szereztek a használatában. A ***BRS*** sokoldalú és gyors keresésre volt képes. Képes volt például bármely szó bármely a felhasználó által megadott mezőben végzett keresésére. Voltak persze nyűgjei, például bizonyos szavakat nem indexelt. Emiatt előfordult, hogy együttesek teljes nevét figyelmen kívül hagyta, ilyen volt például a ***The Who*** nevű társulat. Az ilyen esetekben, ha például a ***The Who*** nevéhez kötődő anyagokat akartuk kikeresni, akkor valami mást is tudnunk kellett róluk, például ismernünk kellett az egyik tag nevét (***Pete Townsend***), vagy valamilyen általuk írt dolgot (***Tommy***). Természetesen egyik megoldás sem volt tökéletesen megbízható.

A parancssoros felület időnként kiszámíthatatlan viselkedése és használatának körülményessége azt eredményezte, hogy munkatársaink jó része a zenekönyvtárost kérte meg az általa kívántak kikeresésére, egyszerűen egy kézzel írt listát adva át neki a kért anyagokról. Az új rendszerrel szembeni elvárások között tehát szerepelt a legalább egyenértékű, de inkább jobb keresési lehetőség, valamint a felületet is úgy kellett elkészíteni, hogy minimális betanulás után bárki használni tudja.

A ***BRS*** egyik leghasznosabb szolgáltatása az volt, hogy a keresési kifejezéseket adott mezőkre lehetett korlátozni. Például:

```
Mozart.cp. zongora
```

A fenti keresés minden olyan anyagot kiad, melynek szerzője (`cp`, `composer`) Mozart, rekordjának bármely más mezőjében pedig szerepel a zongora. Úgy határoztunk, hogy a BRAD alatt is megtartjuk ezt az írásmódot, így a felhasználók a korábban megszokott módon végezhetik el a kereséseket. Eleinte egy speciális keresési oldalt is akartunk készíteni a BRAD-hez, ám a fenti írásmód annyira rugalmasnak bizonyult, hogy végül a külön oldal szükségtelenné vált.

Túllépve a korlátokon

A tervezet megvalósítása során számos kihívással szembesültünk. A legtöbb kapcsán elvárásaink szerint kellett módosítani a ***MySQL*** alapértelmezett működését. Az első probléma az volt, hogy a ***MySQL***-t rá kellett vennünk az adatokban túlságosan gyakori előfordulásuk miatt nem indexelt szavak figyelembe vételére. Esetünkben ugyanis minden egyes szó fontos.

MySQL alatt a tiltott szavakat úgy távolíthatjuk el, hogy – még mielőtt bármit is bevinnénk az adatbázisba – az alábbi sort adjuk a ***MySQL*** beállító fájljához:

```
ft_stopword_file = ""
```

A második gond az volt, hogy a ***MySQL*** által megszabott négy karakteres határérték alatti hosszúságú szavak alapján végzett kereséseket is lehetővé kellett tenni. A ***BRS*** rendszer minden szót indexelt, függetlenül annak hosszúságától, kivéve persze a tiltott szavakat, és a tiltott szavak eltávolításával a megadott feltételeknek jobban megfelelő keresési eredményeket kaphattunk.

Ezt a kérdést két lépésben oldottuk meg. Először az alábbi sort a beállító fájlhoz adva három karakterre csökkentettük az indexszavak hosszát:

```
set-variable = ft_min_word_len=3
```

A kezelt adatmennyiséget is figyelembe véve ez a beállítás még elfogadható teljesítményveszteséggel járt.

Második lépésként készítettünk egy intelligens, a lekérdezést a megadott legrövidebb szótól függően átalakító, majd a ***MySQL***-nek továbbküldő keresőmotort. Segítségével a keresési kifejezések hosszától teljesen függetlenül lehet teljes szöveges keresést végezni.

Utolsó teendőnk az volt, hogy minden keresést alapesetben is ***ÉS*** kereséssé alakítsunk. A ***MySQL*** teljes szöveges boolean módja, ha nem használunk módosítókat, ***VAGY*** keresést végez. Alapesetben egy + karakterrel tudunk ***ÉS*** keresést indítani, tehát a lekérdező motorra bíztuk, hogy – egyéb módosító hiányában – beillessze a szükséges + karaktereket.

A lekérdezésfordító

A ***BRAD*** lelke egy kifejezés-feldolgozó és egy lekérdezésfordító. A kifejezés-feldolgozó átveszi a lekérdezést, feltördeli, az összetevőket pedig egy-egy tömbbe helyezi. Minden tömb tartalmaz egy ***MySQL*** módosítót (+, -, <, >, ~); egy atomot, vagyis a lekérdezési kifejezés egy részét, mely lehet egy szó vagy egy mondat; továbbá egy elhagyható mezőnevet.

A kifejezés-feldolgozó – ha nem talál egyéb módosítót – minden atomhoz önműködően hozzáadja a + karaktert, amivel alapesetben minden keresés ***ÉS*** kereséssé válik. Egyértelműen ez a jobb megközelítés, hiszen a felhasználók is azt várják a kereséstől, hogy minél több szót megadva egyre kifinomultabb eredményt kapjanak.

Az elhagyható mezőt speciális keresések megvalósítására használjuk, ezeknél meghatározott szavaknak meghatározott mezőben kell szerepelniük. ***BRAD*** alatt is megtartottuk

a már említett . karaktert, mint mezőkeresési jelölést. Amikor megkezdünk egy normál keresést, a lekérdezésfordító a keresés hatókörébe eső táblák mindegyikét lekérdezi, majd eredményként ezek teljes szöveges mezőit adja vissza. Ezek alapján kerül összeállításra egy az összes teljes szöveges mezőre kiterjedő lekérdezés.

A lekérdezésfordító teljes szöveges, általános kifejezéseket és nem teljes szöveges, mezőspecifikus kifejezéseket vegyesen is képes kezelni. A lekérdezésfordító segítségével a *BRAD* adatforrásai gyakorlatilag korlátok nélkül bővíthetők, az előállított lekérdezések pedig dinamikusan alakíthatók – szükségtelenné válnak tehát az adatbázis valódi objektumait képviselő objektumosztályokhoz tartozó statikus alaplekérdezések.

A felhasználók – egy normál weboldalról – a keresés minden jellemzőjét képesek szabályozni. (2. *ábra*) A felhasználók annyi zónát vagy táblát választhatnak ki, amennyit csak akarnak. A testreszabási lehetőségekkel maradéktalanul meg lehet felelni a cég és a felhasználók minden igényének.

A BRAD bővítése

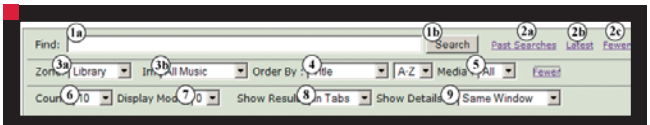
A *BRAD* fejlesztése során különös figyelmet fordítottunk a tetszőleges irányú bővíthetőségre. A keresések bármely típusú adatra kiterjeszthetők, illetve az egyes adattípusokon speciális kereséseket is lehet végezni.

A cégnél súlyos problémát jelentett, hogy a különböző feladatokra különböző adatbázisokat használtunk, és az adathalmazok különféle alkalmazásokba szóródtak szét. Két rádióhálózatunk a *Selector* nevű alkalmazást használja az adásba kerülő műsorok ütemezésére. A *Nemzeti Rádió* egy körülbelül tízezer zeneszámot tartalmazó adatbázissal dolgozik, a *Concert FM* pedig öt, együttesen körülbelül százezer anyagot tároló adatbázisra támaszkodik. A *Concert FM* adatbázisát azért kellett ötfelé osztani, mert a Selector csak korlátozott adatmennyiséget képes kezelni.

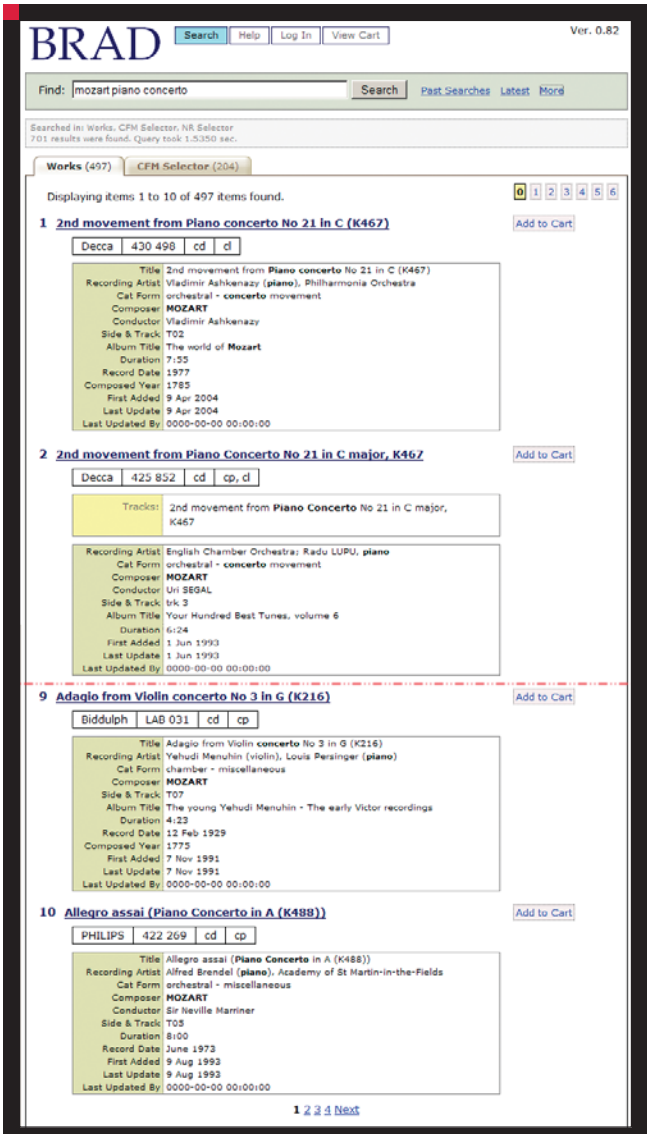
Ha valaki adott zeneszámra szeretett volna rákeresni, akkor mindhárom alkalmazással – *BRS*, a *Nemzeti Rádió Selectora* és a *Concert FM Selectora* – meg kellett próbálkoznia. Mindhárom rendszerben nem lehetett egyszerre keresni. Szerencsére a *Selectorhoz* volt egy az adatok *XML* formátumba való kimentésére alkalmas segédprogram. Igaz, leírás nem volt hozzá, és nem is sikerült szerezni, *Bruce*-nak sikerült rájönnie működésének módjára. Az adatokat így ki tudjuk menteni, illetve egy windowsos munkaállomásról *FTP*-n keresztül fel tudjuk tölteni a *BRAD* kiszolgálójára. A műveletet minden reggel elvégezzük, a *BRAD* kiszolgálóján pedig egy *Perl* parancsfájl gondoskodik az adatok beemeléséről. A *Concert FM* öt adatbázisát egyetlen táblába egyesítettük, ugyanis az adatok egyediek.

Az eredeti keresőmodult kibővítettük, így már egynél több táblában is képes keresni, illetve a mezők számától és típusától függetlenül képes visszaadni az eredményeket.

Az eredmény lapozós stílusban jelenik meg. (3. *ábra*) Az ábrán a *Works* táblából származó első találatok láthatók. A másik fülön a *Concert FM Selector* táblájából származó eredmények száma látható. A keresés hatókörétől és a találatok számától függően tetszőleges számú lap megjelenhet. A műsorszerkesztők így végre egyetlen felületen keresztül bármilyen típusú zenére rá tudnak keresni. Az első változat elkészítése óta munkatársaink kérésére számos további szolgáltatással bővítettük a rendszert. Ilyen például a keresési történet



2. ábra A BRAD keresőfelületének ismertetése (lásd: 2. táblázat)



3. ábra Az eredményeket tartalmazó lap egy részlete

és a kosár. A kosárba bármely tábla elemeit be lehet helyezni. A kosarakat el lehet menteni és elő lehet keresni, illetve létrehozás után a kosár számát el lehet küldeni elektronikus levélben a könyvtárosnak. Így nincs szükség hosszas listák nyomtatására vagy átküldésére, elég csupán a kosár számát közölni.

Hazai zeneszámok keresése és hossz alapján végzett keresés

Legújabb szolgáltatásunk az álnév alapú keresés. Egy álnév egy gyakran használt, összetettebb kifejezést helyettesít. Ilyen például az ***Új-Zéland***, vagyis az olyan zene, amelynek előadói vagy szerzői között új-zélandi művész is szerepel. Fontos jellemző, ugyanis mindkét adóhálózat meghatározott arányban hazai zeneszámokat sugároz.

2. táblázat *A keresőfelület mezői*

1a: Keresési terület	Ide kell beírni a keresési kifejezést.
1b: Search (keresés) gomb	Rákattintva indíthatjuk el a keresést.
2a: Past Searches (korábbi keresések)	Egy a korábbi kereséseket megjelenítő területet nyit meg. A megjelenő elemekre kattintva az egyes kereséseket újra el lehet végezni.
2b: Latest (utolsó)	A Works, a CFMS és a NATS adatforrások legutolsó bejegyzéseit jeleníti meg. Mindegyik forrásból legfeljebb 250 találatot ad vissza, a lista elejére a legújabb bejegyzést helyezi.
2c: Fewer (kevesebb)	Ez a hivatkozás, illetve a következő sorban szereplő a More (több) és a Fewer (kevesebb) beállítás között vált; csökkenti vagy növeli a BRAD által megjelenített keresési lehetőségeket.
3a: Zone (zóna)	A BRAD a cég különféle részeihez kötődő zónákra osztja adatait. A zónáknak számos különböző adatforrásuk van. A zónaválasztó segítségével megszabhatjuk, hogy a cég melyik részére vonatkozzon a keresés.
3b: In (Ebben)	Segítségével kiválaszthatjuk, hogy a zóna melyik adatára szeretnénk rákeresni. A kereséseket általában egy-egy zóna összes adatára vagy egyetlen adattípusára végezzük.
4: Order by (Rendezési szempont)	–
5: Media (Adathordozó)	A BRAD képes az adott típusú adathordozón található anyagokra korlátozni a kereséseket.
6: Count (számláló)	Az oldalanként megjelenített tételek száma.
7: Display Mode (Megjelenítési mód)	A megjelenítési módok a régi BRS rendszerhez kötődnek, segítségükkel a felhasználó kiválaszthatja, hogy az eredményt milyen összegzésben szeretné látni. A megjelenítési módok testreszabhatók, ha tehát egy felhasználó valamilyen különleges formátumot igényel, akkor megoldható annak hozzáadása.
8: Show Results (eredmények megjelenítése)	Lapokon vagy listaként. A BRAD normál módja a keresések eredményét lapozós stílusú felületen jeleníti meg. Lista módban egy fejlécekkel ellátott listát jelenít meg. A lista mód főleg nyomtatásra vagy elektronikus levélben való továbbításra szánt listák előállítására használható.
9: Show Details in (részletek megjelenítési helye)	Segítségével a felhasználó megadhatja, hogy adott rekord teljes adattartalmát – a rekord hivatkozására kattintva – új vagy az aktuális ablakban szeretné látni.

Az évek során a cégnél megforduló emberek számtalan adatbevitelt végeztek, ennek során a fő ***Works*** adatbázisban különféle mezőket és azonosítókat használtak az új-zélandiság jelzésére. Az ***NZ Music*** (mint ***New-Zealand***, új-zélandi zene) álnév – VAGY keresést eredményezve – automatikusan hozzáadja a lekérdezéshez a szükséges kifejezéseket és mezőket. Mindehhez egy új osztályt készítettünk, mely a kifejezés-feldolgozó felett helyezkedik el, és annak segítségével bontja ki a lekérdezésben szereplő álneveket. Ezután a feldolgozó hozzáadja a szükséges átadott értékeket a lekérdezésfordító által kezelt lekérdezésveremhez. Nézzünk néhány példát a BRAD álneveinek használatára. A

```
Mozart @nza
```

lekérdezésnél minden `Mozart` vagy `NZ` tartalmú mezőt megkapunk. A tényleges lekérdezés így néz ki:

```
SELECT * FROM brs.works WHERE (cf REGEXP
➥'[[:<:]]local[[:>:]]' OR cf
REGEXP '[[:<:]]nz[[:>:]]' OR lq REGEXP
➥'[[:<:]]nz[[:>:]]') AND MATCH
ti,ra,cf,cd,cp,so,at,notes,lq AGAINST ('+Mozart'
➥IN BOOLEAN MODE) ORDER BY ti asc LIMIT 1000
```

A hossz alapján végzett keresés szintén a szerkesztők kérésére került be a rendszerbe, segítségével adott időtartamhatárok közötti zeneszámokat lehet találni. Sokszor előfordul ugyanis, hogy valamely szerzőtől nagyjából meghatározott hosszúságú művet kell előkeresni. A ***BRAD*** alatt a szögletes zárójelek közé illesztett számokat hosszmegszorításokként kezeljük. A ***BRAD*** képes a közelítő keresésre, illetve az adott hossztartományokon belüli keresésekre. További példák a széljegyzetben találhatók.

Példák a BRAD alatti hosszra keresésre

Rövidebb mint:

```
brahms [<20]
```

Között (hossztartományt is meg lehet adni). Az alábbi lekérdezéssel a ***Mozarthoz*** kötődő, 20 perc, 30 másodperc és 30 perc, 15 másodperc közötti hosszúságú zeneszámokat kapjuk meg:

```
mozart [20:30-30:15]
```

Közelítő keresések – a megadott időtartam keresése +/- 10 százalék eltéréssel. A c a „circa”, a körülbelül rövidítése:

```
mozart [ c 24 ]
```

Időtartományt is hozzáadhatunk. Az alábbi lekérdezés például a 24 perc, plusz/mínusz 1 perc időtartamú zeneszámokat adja vissza:

```
mozart [ c 24 r 1 ]
```

Összetett időtartam-keresések – az alábbi lekérdezéssel Beethoven 20 és 22 perc közötti időtartamú műveire kereshetünk rá:

```
beethoven.cp [20-22]
```

Az utolsó lekérdezéshez tartozó, lefordított lekérdezés a következő:

```
SELECT * FROM cfm.cfms WHERE (du <= 1320)
  AND (du >= 1200)
  AND MATCH ti,ca,ma, ra,cd,cp,so,at,notes AGAINST
    ('+beethoven' IN BOOLEAN MODE)
  AND MATCH cp AGAINST ('+beethoven' IN BOOLEAN
  ➥MODE)
  ORDER BY ti asc LIMIT 1000
```

A *Concert FM Selectorának* adatai közt – már említettem őket – mindenhol helyesen meg van adva az *NZ* előadói és az időtartam mező értéke, így ezeket az álneveket biztonságosan használhatjuk a teljes adatkészleten. Mivel a *Works* adataiban keverednek az elemtípusok, esetében csak az érvényes időtartammal rendelkezőkre terjed ki a keresés. Mivel korábban a *Works* bejegyzései között egyáltalán nem lehetett időtartamra keresni, ez is előrelépésnek mondható.

A jövő

Írásom születésekor éppen megkértek, hogy illesszem bea *BRAD* alá a céges telefonkönyvet, illetve a hírrészleg számára nemrég készült egy kísérleti kiejtési útmutató.

Összefoglalás

Tervezetünk révén újabb rendszerre tudtuk cserélni az *Új-Zélandi Rádió* kulcsfontosságú katalógusát, illetve újabb szolgáltatásokat bocsátottunk a munkatársak rendelkezésére – mindezt alacsony birtoklási összköltség mellett. Az új és a régi adatok közös tárolórendszert kaptak.

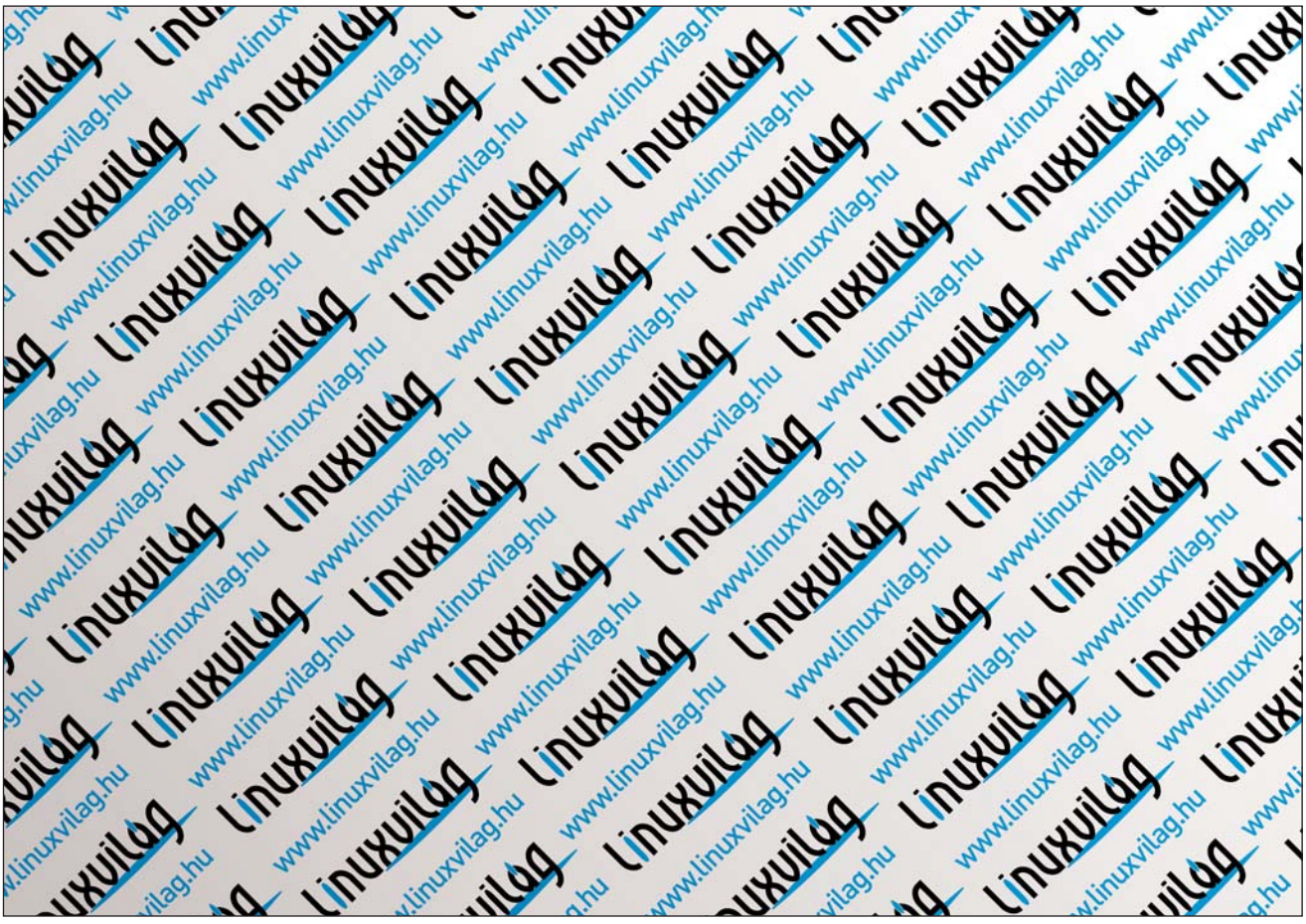
A továbbiakban valószínűleg meg fogjuk oldani, hogy a műsorszerkesztők az egy-egy szerzővel vagy művésszel kapcsolatos zeneműveket, interjúkat és archív anyagokat egyetlen lekérdezéssel megkereshessék. Az eredmények között valószínűleg szerepelni fog az illető telefonszáma és nevének helyes kiejtése is.

A *BRAD* minden bizonnyal fejlődő rendszer marad, mi pedig újabb és újabb alkalmazásokat fogunk kitalálni hozzá. Ez az egyik előnye a csináld magad mozgalomnak – a rendszer a sajátunk, akkor és úgy bővítjük vagy módosítjuk, ahogy és amikor csak akarjuk.

Linux Journal 2005. március, 131. szám

Richard Hulse

Vezető hangmérnök az Új-Zélandi Rádiónál. Jelenleg több informatikai tervezeten is dolgozik, ezek egyike a rádió weboldalának továbbfejlesztése (www.radionz.co.nz).



Solaris 10 (1. rész)
Út az ismeretlen felé

Mikor a téma gondolata megszületett, elhatároztam, hogy magát a cikket az első betűtől az utolsó képig az új rendszer alatt fogom összeállítani. Szerencsére nem volt felelőtlen gondolat. No de ne szaladjunk ennyire előre, egyáltalán mi is ez a Solaris? Legyünk precízek, kezdjük a definícióval!

Egy kis történelem

A *define:solaris* keresőszóra a *Google* felkutatja a kívánt meghatározást: Operating system used on *SUN Systems*. A cikk írásának pillanatában ezt találta a legrelevánsabbnak a kereső, bár a pontossága megkérdőjelezhető. A *Solaris* ma egy többfeladatos, többfelhasználós *UNIX* alapú operációs rendszer, amely a *SUN Sparc*-ján kívül *x86* és *x64* rendszerekre is elérhető. Honnan is indult a rendszer? Tekintsük át a főbb mérföldköveit a fejlesztésnek! 1982-ben alapították meg a *Sun Microsystems*-et és ezzel egyidejűleg megjelent a *SunOS 1.0* is. 1992-ben már *Solaris 2.0*-nak hívták a rendszert, ami a *System V 4*-es kiadásán alapult. Az 1997-ben piacra került *Solaris 2.6* már támogatta a szálakat (*threads*) és a 64 bites folyamatokat (64 bit processes). 2004-ben kiadták a *Solaris 10*-et, mely mára az 1/06 -os változatnál jár. A *Solaris 10* nem nyílt forráskódú, de szabadon elérhető program, valamint a rendszer egy részének az *OpenSolaris* projekt keretein belül elérhető a forrása is. A letöltött szoftvert bármilyen célra felhasználhatjuk. Amíg technikai támogatásra nincsen szükségünk, addig csupán a letöltés és a nyersanyag költségébe kerül a Solaris.

A kezdetek

Mi is szükséges ahhoz, hogy rendszerünkön *Solarist* használjunk? Először is némi kísérletező kedv. Feltételezhető, hogy aki ezt az újságot a kezében tartja, nem riad vissza új operációs rendszerek felfedezésétől, hiszen a legtöbb ember nem úgy kezdte a számítástechnikával való ismerkedését, hogy leültették egy root prompt elé. A rendszer ugyan *UNIX* alapú, tehát a megrögzött *Linux* felhasználóknak nem lesz teljesen idegen, de amint nem csak szövegszerkesztésre és internetezésre akarjuk majd használni, máris előjönnek a különbségek. Ha a kalandvágy adott, akkor a következő lépcsőfok a telepítő *CD*-k illetve *DVD*-k beszerzése.
A ➲ *http://www.sun.com/software/solaris/get.jsp* címen kiválogathatjuk, hogy magán a rendszeren kívül milyen kiegészítőket akarunk még letölteni. Elsőre elég ha csak az operációs rendszert, a Solaris 10 1/06-ot töltjük le. Az alaprendszer egyetlen *DVD*, illetve 5 *CD* (nyelvi *CD* nélkül eggyel kevesebb) formájában töltődik le egy rövid regisztrációt követően. Az adatmennyiség mindkét esetben nagyjából ugyanannyi, körülbelül 2300 Mb. Ezt 1 Mbites *ADSL* kapcsolattal bő öt óra alatt le lehet tölteni. Ha nincs szélessávú internet kapcsolatunk, akkor a ➲ *http://hu.sun.com/buy/* címről kiindulva megrendelhetjük a telepítőkészletet. Miután beállítottuk, hogy a rendszer a *CD*/*DVD* meghajtóról próbáljon elsőnek indulni, semmi más dolgunk nincs, mint berakni a lemezt és újraindítani a rendszerünket. Én a cikkírás erejéig egy külön merevlemezt használtam, amit a *Solaris* egyedül használt, de ha a meglévő rendszerünket nem akarjuk megbolygatni, semmi akadálya annak, hogy például *VMware* virtuális gépre telepítsük fel a rendszert. Magáról a telepítés folyamatáról részletesen nincsen értelme értekezni, hiszen a ➲ *http://shots.osdir.com/slideshows/slideshow.php?release=279&slide=1* címen megnézhetünk egy képernyőkép-túrát a telepítésről. Sokkal inkább az előkészületeken múlik a telepítés sikeressége, semmint azon, hogy hogyan néz ki a telepítő ablaka. Igen, a *Solaris 10* grafikus telepítővel érkezik, sőt mi több, az 1/06 -os kiadásban a *GRUB* indítja a rendszert egy ízléses grafikus menü segítségével.
Mivel valószínűleg az olvasótábor nagy része *x86* vagy *x64*-es architektúrán fog nekivágni a telepítésnek, ezért valamely platformhoz kötődő részek erre értendőek. Mielőtt a frissen írott lemezünkkel nekivágnánk a nagy ismeretlennek, győződjünk meg arról, hogy nagy kudarcok nem érhetnek minket. Attól függően, hogy szöveges vagy grafikus telepítést szeretnénk, 256 MB, illetve 512 MB RAM-ra lesz szükségünk. A feltelepített rendszernél a Sun 256 MB-ot minimumként, 512 MB-ot ajánlott memóriaméretként tüntet fel a telepítési útmutatóban. Amennyire nagy igényei vannak a memóriát tekintve a *Solarisnak*, processzorból

állítólag akár 120 MHz-essel is megelégszik, ezt azonban nem volt alkalmam kipróbálni és mást se nagyon tudok arra bíztatni, hogy egy 120 MHz-es számítógépen próbálja ki teszem azt a *StarOffice 7*-et. A merevlemezen szükséges tárhely 2 GB és 6.7 GB között mozog a feltelepített összetevőktől függően.

Ha a rendszerünk kellőképpen izmosnak találtatott, nézzük meg a *Solaris* ízlését a hardverekre vonatkozóan, mert az *x86/x64*-es platformon még jóval válogatósabb, mint egy friss kernellel felszerelt *Linux* rendszer. A ➲ *http://www.sun.com/bigadmin/hcl/* címen a legfontosabb rendszerösszetevőket érdemes leellenőrizni. Ha ezt a fordulót is sikerrel vette a házi kedvencünk, akkor itt az idő, hogy összeállítsunk egy rövid listát azokról az információkról, melyekre telepítés közben szükségünk lesz.

Tudnunk kell, hogy a rendszerünk hálózatba van-e kötve, valamint akarunk-e *DHCP*-t használni, vagy sem. Ha a második kérdésre nem a válasz, akkor az IP címet és a hálózati maszkot is írjuk fel. Az *IPv6* használatáról is döntenünk kell. A számítógép hálózati neve (host name) sem árt, ha kéznél van. Lehetőség nyílik *Kerberos* beüzemelésére a telepítés közben.

A következő lépcső a névfeloldás lesz, a *NIS+*, *NIS*, *DNS* és *LDAP* közül választhatunk. Itt általában a *DNS*-re van szükségünk és ha ezt jelöljük be, meg kell adnunk az elsődleges és másodlagos névszervert. Ezután az alapértelmezett átjárót(default route) adhatjuk meg. Ezzel a hálózati beállítások végére értünk. Az időzóna, a rendszergazdai jelszó, a nyelvi támogatás (van magyar locale is) kiválasztása után a telepítendő szoftverekről dönthetünk. Dönthetünk az alapértelmezett telepítés mellett, ami mindent felrak, vagy a különféle előre összeállított csoportokból válogathatunk.

Ha ez még nem elég finomhangolás, akkor csomagról csomagra szemezgethetünk, bár ekkor érdemes tisztában lenni a csomagfüggőségi viszonyokkal.

A merevlemez felosztását, ha csak a Solaris lesz a gépen, nyugodtan rábízhatjuk a telepítőre, ellenkező esetben ki kell találnunk egy nekünk

1. ábra Java Desktop System for Solaris

megfelelő elrendezést. Az itt használt partíció-felfogás eltér a *Linux* alatt megszokottól: a partíciókon belül x86 rendszereken slice-okat kell létrehozni és a slice-ok tartalmazzák az egyes fájlrendszereket. Egy partíciót 16 slicera oszthatunk. A telepítő nagyjából ennyi információval be is éri, aztán már semmi teendőnk nincs. A gépünk erősségétől függően kicsit vagy nagyon kell türelmesnek lennünk, aztán egy újraindítás után meg is pillanthatjuk a grafikus felületet, ahol bejelentkezhetünk.

Otthon, édes otthon

A bejelentkezés után rögtön döntenünk kell: *Java desktop* vagy *CDE*. A *Java desktop* többé-kevésbé a *Gnome*-ot fedi, válasszunk ízlés szerint. Én a *Java desktopra* voksoltam, a továbbiak ez alapján íródtak.

Miután bejelentkeztünk, van még tennivaló annak érdekében, hogy egy magyar cikket el tudjunk készíteni és a fontosabb dolgok életre keljenek. A *locale* rendes magyar beállítása nem vonja maga után azt, hogy a billentyűzetkiosztás magyar.

A *Solaris 10 x86/x64* platformos verziója *Xorg*-ot használ, így a magyar billentyűzetet éppen úgy kell beállítani, mint bármely más rendszer alatt, ahol az *Xorg* látja el az grafikus rendszer feladatát. Először is hozzunk létre a */etx/X11/xorg.conf* fájlt. Ez a telepítés során egyáltalán nem is jön létre. Mintának azért találunk egy */etx/X11/.xorg.conf* fájlt, ez jó kiindulási alapnak. Két sorral kell kiegészíteni, hogy le tudjuk írni azt a szép példamondatot, hogy öt szép szűz lány őrült írót nyúz (egyébként ugye mennyivel kellemesebb, minthogy árvíztűrő tükörfúrógép?):

```
Option "XkbModel"
➥"sun_type6_euro_usb"
Option "XkbLayout"  "hu"
```

Mindezt a billentyűzet részéhez természetesen. Szükségünk lesz a monitorunk néhány adatára is. Legkönnyebb egy már meglévő *Xorg.conf*-ból átemelni a helyes beállításokat. No igen, szövegszerkesztés: vagy a `vi`-t használjuk vagy a `gedit`-et. Rosszabb esetben az `ed`-et. Később ezen majd segítünk, mert talán nem mindenkinek e három közül kerül ki a kedvenc szerkesztője. Ha elvégeztük a szerkesztést és újraindul az *Xorg* (egy kijelentkezéssel elérhető), akkor jó esetben máris gépelhetünk magyarul. Az *Evolution*, a *StarOffice 7*, a *Mozilla* és a *Gimp* és még sok egyéb is azonnal a rendelkezésünkre áll, így aztán nem is érezzük magunkat nagyon elveszettnek. Ha a telepítés során jól adtuk meg a hálózati tulajdonságokat és rendelkezünk internetes kapcsolattal, akkor mindezeknek működnie kell. De még mielőtt nagyon belefeledkeznénk a felfedezésbe,

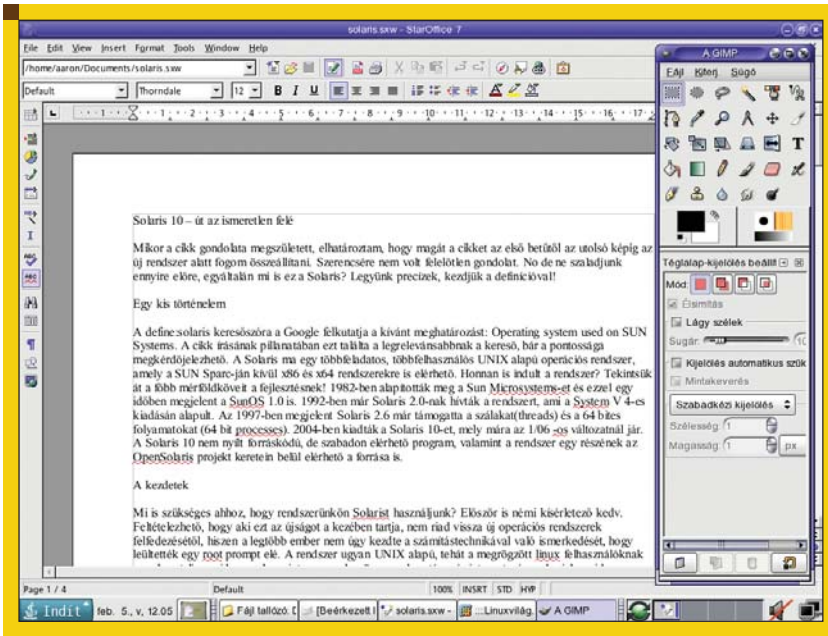
vegyük észre, hogy a telepítéskor csak a root felhasználói fiók jött létre, így nyilván azzal vagyunk bejelentkezve. Ezen változtatni kell! Használhatjuk nyugodtan az `useradd` parancsot.
A *Solaris home* könyvtár konvenciója eltér a *Linuxban* megszokottól: a */home* az `automount` fennhatósága alatt van, a sima *home* könyvtárakat az */export/home* alatt kell létrehozni:

```
useradd  -m -d /export/
➥home/valaki valaki
passwd valaki
chown valaki /export/
➥home/valaki
```

Csak zárójeles megjegyzés, hogy ezt a konvenciót fel lehet rúgni büntetlenül, működni fog a rendszer.
Ezután jelentkezzünk ki és vegyük birtokba személyes használatú fiókunkat, amit már otthonossá tehetünk, hisz nem a root felhasználóhoz tartozó beállításokat módosítjuk vele.
Ha akarjuk, a jól bevált *Bash* héjra is átnyergelhetünk, hisz *Solaris* alatt a *Bourne Shell* az alapértelmezés.

Jajok és mosolyok

Az *Evolution* és úgy egyáltalán az egész *Gnome* környezet magyarul beszél hozzánk. A rendszer kellőképpen sebesen működött egy 768 MB RAM-mal és egy 1400 MHz-es *AMD* processzorral ellátott rendszeren.
A *Java desktop* attól *Java desktop system* és nem *Gnome*, hogy így jobban hangzik a sajtóbejelentésekben. Akkor az előző mondat helyesen: attól nem *Gnome*, hogy a java a telepítés során felkerül a rendszerre és számtalan praktikus *Java*-s programot találunk a *Gnome* szoftverparkja mellett.
Itt feltétlenül meg kell említeni *Kertész Csabát*, akinek a *jDictionary* programja belekerült a rendszerbe. Így aztán egyből van angol-magyar és német-magyar szótárunk is.
A *StarOffice 7*-be nem került bele az *OpenOffice.org* magyarítása, legnagyobb meglepetésemre azonban rengeteg, köztük a magyar nyelv helyesírás ellenőrzője viszont igen.
Beraktam egy *DVD* lemezt a meghajtóba, a Totem nem volt hajlandó lejátszani. Egyáltalán nem volt hajlandó szinte semmilyen filmet lejátszani.
A hangkártyámat telepítés után nem ismerte meg a rendszer, de miután a *Solaris HCL* szerint egy kereskedelmi szoftverrel (*Open Sound System*) működésre lehet bírni, letöltöttem, feltelepítettem az *OSS Solarisos* (➲ *http://www.opensound.com/solaris-x86.html*) verzióját. Bár ne tettem volna! Ez okozta az első és egyetlen kernelpánikot.
És ami még rosszabb: a *Solaris* alapértelmezett fájlrendszere az *UFS*, így az újraindítás után vad `fsck` futtatásba kezdett. Szükség volt a fájlrendszer módosítására is. Szerencsére eztán gond nélkül elindult a rendszer.
Jó tudni, hogy néhány hangkártyához beépített támogatást kínál a *Sun*, ott nem fordulhat elő ilyen kínos malőr külső forrásból származó eszközmeghajtó miatt. A *Sun Update Manager* biztosítja, hogy a rendszerünkre felkerüljenek a különféle frissítések, ez a szolgáltatás kifogástalanul üzemelt a teszt alatt. Ez is egy *Java* alapú program (a rendszerhez közvetlenül kapcsolódó felügyeleti szoftverek jó része ilyen), de az általam használt rendszeren soha nem éreztem problémának az alkalmazások futási sebességét.
A *Gnome* is teljesen akadozás nélkül futott, bár ez ugyanezen a gépen egy lassabb merevlemezen lévő *Debian GNU/Linux* alatti *KDE* felületre is igaz. Sok weboldalhoz szükség van *Flash* lejátszóra, ez elérhető szerencsére – bár éppúgy egy elavult verzió, mint *Linuxra*. A *Gimp* szintúgy jól működik, mint *Linux* alatt.
Visszatérve a cikk bevezetőjében említett elhatározásra, nem kellett csalódnom, a levelezésem az *Evolution* képében, a webböngésző a *Mozillától*, a *StarOffice 7* és a *Gimp Solaris 10* alatt is szolgáltatták azt a megszokott felületet, ami egy cikk megírásához A-tól Z-ig szükséges. Viszont a hangkártya néma maradt, így rákényszerültem arra, hogy a magyar kereskedelmi rádiókat hallgassam az internetes rádiók helyett munka közben, valamint a *Solaris 10* webes *Flash* bemutatójának a hangját sem hallhattam. Ellenben letöltve és feltelepítve(mindkét szoftvernek felhasználóbarát telepítője van) a *Sun Studiot* és a *NetBeanst*, *C* és *C++* fordítót (a telepítés során nem kerül fel), valamint integrált fejlesztői környezetet kaptam. Itt még mindig ugyanott tart a dolog, sőt kicsit hátrébb, mintha egy átlagos *Linux* terjesztéssel kezdtem volna, hiszen a hang nem működik és még videózni sem lehet.
Ez utóbbin egy forrásból telepített *xine* vagy *MPlayer* segített volna. Ez esetben a folyamat majdnem úgy megy, mint a pingvines rendszeren, viszont `make` helyett `dmake` parancs használatos. Viszont sok program csak a *GNU make*-kel, sőt mit több, a *GCC*-vel hajlandó lefordulni. A rendszeren én a `joe` szövegszerkesztőt, az `mc` fájlkezelőt és a `wget` letöltéskezelőt tudtam elsőre, minden gond nélkül a *Sun* által biztosított fejlesztőeszközökkel lefordítani.
Visszatérve az *Xorg* konfigurálásánál



2. ábra StarOffice 7 és a helyesírás szigorú őre bevetés közben

tett ígérethez: a *Sun Studióval* jó eséllyel lefordul a kedvenc szövegszerkesztő. Aki nem akar a forráskódokkal és a fordítókkal bűvészkedni, annak a ➲ *http://sunfreeware.com/indexintel10.html* oldalon és a *Solaris 10 1/06 Software Companion Packages* kínálatában a sun.com oldalon bőséges választék áll rendelkezésre. Ezeket a binárisokat a pkgadd paranccsal üzemelhetjük be.

Az új rendszeren sajnos nem tudtam pontos sebességméréseket végezni, mert a Debian, melyet nap, mint nap használok, egy jóval lassabb merevlemezen lakik, így semmi értelme nem lett volna különféle alkalmazások sebességét méricskélni.

Látható, hogy kisebb-nagyobb kényelmetlenségekkel éppúgy belakható asztali rendszernek a *Solaris 10*, mint bármi más, ha nem használunk egzotikus hardvereket. Ezt elolvasva talán alábbhagy a kalandvágyunk, pedig épp most jön a kaland!

Hajtsátok uralmatok alá!

Természetesen a *Solaris* rendszerről van szó. Eddig az alapértelmezett telepítéssel dolgoztunk, átállítottuk a billentyűzetkiosztást, szoftvereket használtunk és helyeztünk üzembe; ez nem valami nagy kihívással járó munka. A *Solaris Management Console* és a *Java Web Console* igazán hatékony eszközöket ad az elcsigázott rendszergazdának, aki úgy érzi, hogy kicsúszik kezei közül a rendszer irányítása. A *Service Management Facility-vel (SMF)* a rendszer szolgáltatásait vezérelhetjük. Megismerhetjük a *DTrace*-t, mellyel könnyedén felderíthetjük, hogy mitől vánszorog csiga módjára leterhelt rendszerünk. Ha ennyi még nem volna elég, zónákat állíthatunk fel, megvédve ezzel az egyes szolgáltatásainkat. Ezekről lesz szó a következő számban, addig is a *Solaris* annyi felfedezni valót tartogat, hogy biztosan nem lehet mellette unatkozni!

Addig egy kis ízelítő linuxosoknak, hogy ne érezzék magukat teljesen elveszve az ismeretlenben.

A `top` helyett a `prstat` használatos. Ha ki akarjuk adni a `ps aux`-ot, hogy átfogó képet kapjunk a folyamatainkról, akkor ezt `ps -Af` adja nekünk. Ha a különféle szolgáltatások (*services*) után kutakodunk, az `svcs` ad róluk egy listát, utána az `svcadm disable svc:/eleresi_ut` utasítással leállíthatjuk azokat, amiket fölöslegesnek ítélünk. Természetesen itt csak önkényesen kiemeltem pár számomra fontos tudnivalót kedvcsinálónak, de szerencsére a ➲ *http://docs.sun.com* címen bőséges angol nyelvű dokumentációra lelhetünk mind a telepítésről, mind pedig a rendszer üzemeltetéséről. Tehát a következő alkalommal a *Management Console*, a *DTrace*, szolgáltatáskezelés, zónák és tűzfalak körül nézelődünk, hogy megismerjük, mit tartogat a Solaris a rendszergazdáknak.

Novák Áron
(aaron@szentimre.hu)

BME-VIK-es hallgató, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal, amelyeket meg lehet szólaltatni Linux alatt.

Fájlszinkronizálás Unisonnal

Több gépen egyszerre szinkronban tartani a könyvtárakat nem könnyű feladat. Az egyik lehetőség amivel egyszerűbbé tehetjük életünket, az Unison használata.

Az ***Unison Linux***, ***UNIX*** és ***Microsoft Windows*** környezetben futó fájlszinkronizáló program. Akik már használtak ***IBM Lotus Notes*** vagy ***Intellisync Mobile Suite*** programokat nagyjából tudják miért is jobb a szinkronizálás mint az olyan egyirányú tükrözési megoldások mint az ***rsync***. Elképzelhető például, hogy a cég dokumentum könyvtárát a laptopunkra tükrözzük, majd módosítunk egy-két dokumentumot. Mások ugyanebben a könyvtárban más állományokat módosíthattak, mire visszaérünk. Az ***rsync*** esetében kézzel kellene elsimítanunk a két könyvtár közötti különbségeket vagy kockáztatjuk, hogy felülírjuk valakinek a munkáját. Az ***Unison*** ki tudja válogatni mi és hol változott, terjeszti a megváltozott állományokat sőt, ha megmondjuk miképpen tegye, akár az azonos fájlok változásait is képes összeolvasztani.

Az ***Unisont*** úgy is elképzelhetjük, mint egy kis változáskezeléssel fűszerezett kétutas ***rsync*** programot. Leggyakoribb felhasználási területe a helyi és távoli saját könyvtár vagy egy különféle környezetekben gyakran használt könyvtár szinkronizálása. Az ***rsync*** algoritmusával tartja alacsonyan a hálózati forgalmat és ***SSH***-n keresztül vihető át a megbízhatatlan hálózatokon. Mindez többletmunkát nem jelent számunkra: egyszerűen a ***ssh:***// kifejezést kell megadnunk amikor a helyi könyvtárat bejegyezzük. Ugyanakkor az ***Unison*** használata során elég szép mennyiségű extra helyre lesz szükségünk, ugyanis a szinkronizáló algoritmusnak tudnia kell, hogyan néztek ki az állományok a legutóbbi futtatáskor.

Unison beszerzése, fordítása és telepítése

Az ***Unison*** honlapja az ***Pennsylvania Egyetemen*** található; a projekt vezetője, ***Benjamin C. Pierce***, a ***Számítástechnika és Információtudomány Tanszékének*** professzora. Az ***URL***-t a források közt találjuk.

Az ***Unisont*** nem alkalmazzák olyan széles körben mint az ***rsync***-et, így előfordulhat, hogy a terjesztésünkhöz nem találunk előre fordított csomagot. Az ***Unison*** honlapról letölthető binárisok azonban valószínűleg a legtöbb embernek megfelelőek lesznek .

Ha forrásból szeretnénk lefordítani, ezt is megtehetjük. Igaz ez esetben pár körrel többet kell futnunk, ugyanis az ***Unison*** a nem túl gyakori ***Ocaml*** nyelven készült.

Ha terjesztésünk nem tartalmaz kézhez álló csomagot, a források közt megtaláljuk.

1. lista .unison/default.prf

```
# Az Unison beállító állomány
merge = diff3 -m CURRENT1 OLD CURRENT2 > NEW
backup = Name *
maxbackups = 10
log = true
logfile = /home/knan/.unison/unison.log
rshargs = -C
```

Az ***Unison*** fordítása és telepítése egyszerű; adjuk ki a `make UISTYLE=xxx` parancsot. A ***GTK*** felhasználói felülethez további ***OCaml GTK*** bővítményekre lesz szükségünk, ezért cikkünkben a szöveges felületre szorítkozunk. A `make UISTYLE=text` vagy `make UISTYLE=gtk` parancs segítségével kapjuk meg az ***Unison*** végrehajtható állományt. Egyszerűen másoljuk az állományt valamilyen elérhető helyre mindkét szinkronizálandó gépen.

Cikkünkben ha külön nem említjük, az ***Unison*** legfrissebb 2.9.1-es stabil változatát használjuk. Amennyiben 2GB-nál nagyobb állományokat szeretnénk szinkronizálni, jelenleg a béta változatot kell alkalmaznunk.

Általában a fejlesztői verziók is jól működnek. A fejlesztők is ezeket használják saját értékes adataikon. A kalandvágyóak feliratkozhatnak az ***Unison*** programozók levelezőlistájára is. ***Jerome Vouillon***, ***Benjamin C. Pierce*** és ***Trevor Jim*** általában itt vitatják meg a fejlesztéseket. Gyakran láthatunk küldeménynaplókat is, úgyhogy nyomon követhetjük a történéseket.

Unison beállítása és használata

Az ***Unison*** saját beállításait és munkafájljait a ***.unison*** könyvtárban tartja a saját könyvtárunkon belül vagy ahová éppen tenni szeretnénk. Az ***UNISON*** környezeti változót beállítva más helyet is megadhatunk.

Az alapértelmezett beállításokat a ***.unison/default.prf*** állomány tárolja. Az *1. lista* teszteléshez használható egyszerű beállítást mutat. Két könyvtár szinkronizálása ezek után mindössze ennyiből áll:

```
$ unison /nfsmount/dir1 /home/me/dir1
```

2. lista .unison/home.prf

```
# Unison beállítás állomány
root = /home/erik
root = ssh://remotehost/home/erik
# vagy pontosan kettő "root" sor lehet,
# vagy egy sem
ignore = Name *.mp3
# az összes .mp3 állományt figyelmen kívül
# hagyjuk
ignore = Path .unison
# nem foglalkozunk az elérési útjukban .unison
# szót tartalmazó # állományokkal
include default
# az alapbeállításokat a default.prf-ből
# olvassuk
```

Az ***Unison*** ezek után megkérdezi, van-e különbség a könyvtárak között majd elfogadható alapértékeket kínál fel. Beletelhet kis időbe míg hozzászokunk az ***Unison*** gondolkodásmódjához. Fontos, hogy az ***Unison*** nem helyettesíti a mentéseket. Például nyugodt szívvel tovaterjeszti az egyik másolatban bekövetkezett fájltörléseket, ami hidegzuhanyként érheti a ***CVS***-hez szokott programozókat. Például:

```
rm dir1/* ; unison ssh://server/dir1 dir1
```

egyáltalán nem úgy viselkedik mint a következő sortól várjuk:

```
rm dir1/*; cvs update dir1
```

A törlés ugyanolyan művelet, amit le kell másolni szinkronizáláskor. A példaparancs ennek megfelelően mindkét oldalon törli a dir1 könyvtár tartalmát.
Miután kicsit belerázódtunk, érdemes az `auto = true` utasítást elhelyezni az ***Unison*** állományába. Így az ütközést nem okozó változásokra nem kérdez rá a rendszer de a végén lehetőséget ad a visszalépésre.
Érdemes áttanulmányozni az ***Unison*** kézikönyvét. Tisztán és világosan fogalmazott és jól rávilágít mi fog történni az egyes határesetekben .

Sajátkönyvtárak szinkronban tartása

Amikor a felhasználó megbarátkozott az ***Unisonnal*** gyakran felmerül a sajátkönyvtárak gépek közötti szinkronizálásának kérdése, például a laptop és az asztali gép között. Ezt nagyon könnyen megvalósíthatjuk. A *2. listában* olvasható egyszerű beállítások ezt a feladatot oldják meg, bár valószínűleg bővíteni akarjuk majd. A 2. lista, például, figyelmen kívül hagyja az MP3 állományokat és az ***Unison*** saját állományait valamint bemutatja az `include` működését mellyel valamennyi profilon azonos beállításokat érvényesíthetünk.
Új profilunkat a következőképpen próbálhatjuk ki:

```
$ unison home -testserver
```

Meghívása pedig így történik:

```
$ unison home -batch
$ unison home
```

A `-batch` futtatás kezeli az összes egyszerű esetet visszakérdezés nélkül, ment és naplóz szükség szerint, majd a második futtatásnál az összes trükkös dologra rákérdez – például az összeolvasztásokra.
A `root =` sorok elhagyhatók ha inkább a parancssorban szeretnénk megadni a szinkronizálandó állományokat.
A sorok a következő parancshívásnak felelnek meg:

```
$ unison home /home/erik
➥ssh://remotehost/home/erik
```

Ütköző változások összeolvasztása

A három utas összeolvasztás kivitelezéséhez be kell kapcsolnunk a mentéseket. Alapértelmezés szerint a háttérmentés nincs engedélyezve, az ***Unison*** csak az ellenőrző összegeket és metaadatokat (például a jogosultságokat) tárolja tehát nincsen felhasználható módosítatlan változata. Az ***Unison 2.9.1***-es változatában, ha az ütközés feloldása érdekében kiadott összeolvasztás művelet kézi beavatkozás nélkül is sikeres volt, a változások azonnal szétterjednek, ami nem ad lehetőséget a visszavonásra. Ezért, ha van elég helyünk, javaslom a `maxbackups` értéket hagyjuk legalább 5-ön az alapértelmezett 2 helyett, így megadjuk az esélyt, hogy az esetleges hibás összeolvasztásokat helyre tudjuk állítani. A mentési könyvtár összeolvasztás után a következőképpen néz ki:

```
$ ls -1 .unison/backup/
shared.txt          összeolvasztott verzió ("NEW")
shared.txt.1.unibck távoli változat ("CURRENT2")
shared.txt.2.unibck helyi változat ("CURRENT1")
shared.txt.3.unibck régi verzió ("OLD")
```

A legfrissebb ***Unison*** béta változat (írásunk születésekor a 2.10.3) eltérő állományokhoz eltérő összeolvasztó programokat képes meghívni. ***XML*** állományok összeolvasztásához például valószínűleg szívesebben használnánk ***3DM***-et míg a ***Berkeley*** adatbázisokhoz a valamilyen adatbázis olvasztó eszközt. Ez a képesség még új és képlékeny. A projektvezető meg is jegyezte, hogy az összeolvasztó képességet mindenképpen újra kellett írni és nem működött igazán jól a 2.9.1 és 2.9.20 változatban. Ezért ha sok olvasztást szeretnénk végezni, jobban járunk ha a legfrissebb változatokból szemezgetünk.

Linux Journal 2005. április, 132. szám

Erik Inge Bolso UNIX tanácsadó és épée vívó a Norvégiai Molde-ban, és 1996 óta futtat Linuxot. Másik hobbijáról a „balrog genealogyról" egy Google keresés segítségével kaphatunkinformációt, őt magát pedig a ljcomment@tvilsom.org címen érhetjük el.

KAPCSOLÓDÓ CÍMEK

A cikk forrásai: ➲ www.linuxjournal.com/article/8059

Házi stúdió Linux alatt

Audacity

Saját zenéink megörökítéséhez Linux alatt nem kell több, mint egy hangszer, némi tudás, egy számítógép hangkártyával felszerelve, továbbá egy program, mely képes rögzíteni művészetünket. Tegyük fel, hogy a feltételek első része adott és foglalkozzunk az utolsóval.

Természetesen azért ennyire nem egyszerű a dolgunk, de alaposan ki fogok térni mindenre. A hangszerhez és a hangszeres tudáshoz nem fűznék különösebb kommentárt. Akusztikus hangszernél vagy énekhangnál nyilván szükségünk van mikrofonra, ami pluszként beköltözik a feltételeink közé.

A tudásról is csak annyit, hogy bármilyen hangszeres tudással bátran fogjunk hozzá a felvételek készítéséhez, mert ebből lehet a legjobban tanulni. A felvétel nem hazudik, minden hibát brutális őszinteséggel tükröz. Ha gitározunk, és a játék hevében mellényúlunk, az talán fel sem tűnik. Viszont, ha egy felvételt hallgatunk vissza saját magunkról, akkor igazán meglepődhetünk a felmerülő bakik hallatán. Talán a beszédhangunk visszahallgatásához tudnám hasonlítani, ami általában mindenkinek furcsa elsőre. A játékunkat is meg kell szokni, és rengeteg felvételt kell készíteni ahhoz, hogy megszokjuk, fejlődjünk, majd egy saját stílust alakítsunk ki.

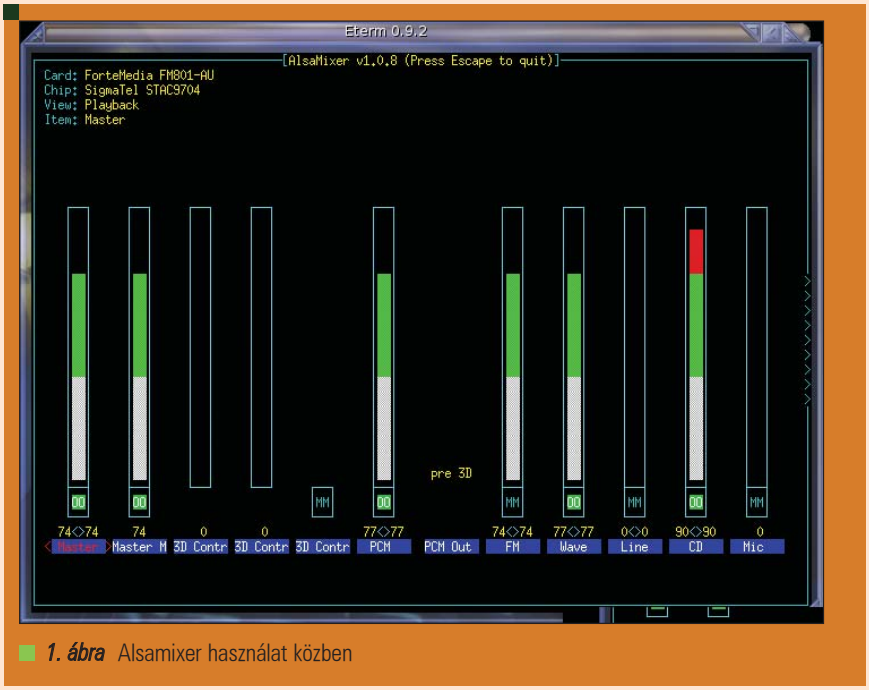
Jómagam gitározom már csaknem 13 éve. Általában hetente szoktam egy-egy nyúlfarknyi felvételt készíteni nem csak az utókornak, hanem saját magamnak is okulásképpen. Bármiről szólhat a felvétel: akár egy skála tisztán eljátszva, egy akkordmenet, vagy egy saját ötlet, ami a legjobb. Észrevehetően a legnagyobb örömöt az okozza, ha másoknak is tetszik a munkánk. Nos, ennyi bevezető után – stílszerűen – csapjunk a húrok közé. *Linux* rendszereken régebben igen nehezen és fáradságos munkával lehetett csak működésre bírni hangkártyáinkat. Azóta már jócskán javult a helyzet, de akadnak még manapság is olyan – általában régi típusú – hangkártyák, melyekkel meggyűlhet a bajunk.

Hangkártyák beüzemelése

Célunk egy működő hangkártya, *Line-in* vagy *Mic* bemenettel. Ha egyik sem található rajta és mégis szeretnénk hangfelvételt készíteni, akkor célszerű beruházni egy új hangkártyába. Első lépésként meg kell győződnünk arról, hogy a hangkártyánk kernelmodulja (driverje) megfelelően be van-e töltve. Mivel rengeteg gyártó és hangkártyatípus létezik, ezért nem térnék ki ennek a műveletnek a részletezésére. Bármely internetes linuxos fórumon lehet kérni tanácsot ehhez. Megjegyzem, az újabb disztribúciók már nagyon fejlettek, fel vannak vértezve hardverfelismerő programokkal, úgyhogy az esetek 90%-ban a hangkártya meghajtójának betöltése kihagyható.

Linux alatt az *ALSA* (➲ http://alsa-project.org) nevű program végzi a hangkezelést. Léteznek ugyan más hangkezelő szervek is, de a legfőbb ez. Amennyiben megfelelően működik, úgy a hangkártyánk is tökéletesen látja el a feladatát. Nyomatékosítom újfent, hogy a legtöbb mostani *Linux*-disztribúcióban már szinte semmilyen beállításra nincs szükségünk hangkártyánk igénybevételéhez.

Egy szó, mint száz, lássunk most egy demonstrációt, hogy *Debian GNU/Linux* (➲ http://debian.org) alatt hogyan is kell egy hangkártyát működésre bírni, amennyiben a kernelmodulja minden hiba nélkül betöltődött.



1. ábra Alsamixer használat közben

Legelőször feltelepítjük az ***ALSA*** megfelelő csomagjait az

```
apt-get install alsa-base
```

és az

```
apt-get install alsa-utils
```

parancsokkal.
Ezek után hozzáadjuk a felhasználónk csoportját az ***audio*** csoporthoz az

```
addgroup felhasználónév audio
```

paranccsal, amely a legjobb tudomásom szerint csak ***Debian*** alatt szükséges. A „felhasználóbarát" terjesztések ezt a lépést elvégzik helyettünk.
Ebben a pillanatban, ha minden jól ment, a hangkártyánk bevetésre készen áll, csupán egy apró dolog van hátra, mégpedig a mixer (például hangerő) beállítása az

```
alsamixer
```

paranccsal. Ezt célszerű konzolon végrehajtani , bár léteznek grafikus ***alsamixer***en alapuló keverőprogramok is, mint például az ***Aumix, GTK*** felülettel.
A keverő beállítására egy hatékony módszer a következő: indítsunk el egy zenefájlt, mielőtt nekikezdenénk. Nem kell megijedni a teljes csend hallatán, a potméterekkel tökéletesen beállítható a kívánt hanghatás.
Az ***alsamixer*** funkcióbillentyűi: ***M*** – némítás, illetve a némítás megszüntetése, fel-le nyilak – potméterek állítása, jobbra-balra nyilak – potméterek közötti mozgás, ***Esc*** – kilépés.
Ha mindent megfelelően beállítottunk és kiléptünk az ***alsamixer***ből, adjuk ki az

```
alsactl_store
```

parancsot, amely elmenti beállításainkat.

Zenefelvevő és zeneszerkesztő programok

Számtalan zenefelvevő és zeneszerkesztő alkalmazás létezik ***Linux*** operációs rendszerre. Egy részük kiváló, másik részük sok jóindulattal sem kerülhetne a gyenge kategóriába. Van karakteres, grafikus, könnyen kezelhető, nehezen kezelhető, és bizony létezik olyan is, melynek futtatásához elég izmos gép kell. A ➲ http://linux-sound.org oldalon csaknem az összes linuxos zenei alkalmazás megtalálható, köztük az úgynevezett ***harddisk-recorder***ek is. Lényegében ezen cikk is erről fog szólni. Úgy képzeljük el, mint egy virtuális stúdiómagnót.

Harddisk recorderek

Aki már egy kicsit kalandozott a számítógépes zenerögzítés területén, annak ismerősen csenghet a ***CoolEdit*** név, melyet ***Windows*** alatt majdnem minden valamirevaló zenész használt. Nagyszerűsége az egyszerűségében és a minőségében rejlik.
A ***harddisk-recorder*** nevében is benne van, hogy a „merevlemezre veszi fel" a zenét. Több sávon dolgozhatunk, programja válogatja, hogy mennyin. Szerkeszthetjük a felvett részeket, vághatunk, majd összeilleszthetünk darabokat. Használhatunk beépített effekteket, kihasználhatjuk a teljes sztereó panorámát, „lekeverhetjük" a zenét, tehát mintha csak egy stúdióban lennénk.
Sokáig kizárólag a ***CoolEdit*** program miatt használtam ***Windows***t, miközben buzgón kutattam a ***Linux***os megfelelőjét több-kevesebb sikerrel. Tucatjával próbáltam ki a különféle programokat, melyek nagy része már a telepítésnél elhasalt. Már csaknem feladtam a keresgélést, amikor egy véletlen folytán belebotlottam az ***Audacity***be.

Audacity

Egy olyan programot szeretnék bemutatni, ami a kiváló kategóriába tartozik, mindemellett egyszerű a kezelése és kicsi az erőforrásigénye, tehát minden szempontból megfelelő.
Az ***Audacity*** (➲ http://audacity.sourceforge.net/) egy multiplatformos, nyílt forráskódú ingyenes zenefelvevő és szerkesztő program ***GNU/Linux***, ***MacOSX*** és ***Windows*** operációs rendszerekhez. Jelenlegi stabil verziója az 1.2.4., de már elérhető az 1.3.0. bétaverzió is. A fejlesztők sok munkát fektettek az alkalmazás egyszerűségébe és átláthatóságába. A ***wxWidgets-GTK*** felület ilyen téren nyerő választás. Ellentétben a ***Qt***-alapú programokkal, kicsi az erőforrásigénye. Szinte bármilyen kis gépen elfut, ráadásul homogén és tetszetős.

Telepítés

Az alkalmazás telepítése pofonegyszerű. ***Debian*** alatt elég hozzá kiadni az

```
apt-get install audacity
```

2. ábra Bemeneti forrás választása Audacityben

parancsot. Más disztribúciókhoz (***Alt Linux, Fedora Core, Ubuntu, Gentoo, PLD Linux***) is találhatunk csomagokat a ➲ http://audacity.sourceforge.net/download/linux oldalon, ahonnan a forráskód is letölthető. Mivel alapvetően kis programról van szó, ezért bátran használjuk az előregyártott csomagokat a fordítgatás helyett.

Az Audacity jellemzői

Előszóban annyit, hogy a program legfőbb erénye a használhatóság, az áttekinthetőség és a sokrétűség. Másrészt pedig szabad szoftver és nyílt forráskóddal rendelkezik. Akár itt be is fejezhetném, de azért nézzük az alkalmazás fontosabb részeit bővebben.

Felvétel

Az ***Audacity*** képes bármilyen hangot rögzíteni mikrofon vagy mixer segítségével. Mit is jelent ez? A hétköznapi hangkártyákon általában 3 ***mini-jack*** aljzat van: ***Mic***, ***Line-In***, ***Line-Out***. Mi most a ***Mic*** és ***Line-In*** bemenetekkel fogunk foglalkozni. Lényeges dolog a felvételhez, hogy egyik se legyen némítva. Ezt az ***alsamixer***ben tudjuk ellenőrizni (a potméterek alatti kis négyzet, amiben *MM* van, rikító zöld). Ezután kiválasztjuk a kívánt bemeneti forrást, illetve a felvétel alaphangerejét megadjuk. A ***Record*** gombra kattintva rögtön megkezdődik a felvétel. A program sokszínűségét mi sem bizonyítja jobban, mint az, hogy nem csak ***mikrofon***, illetve ***Line-In*** bemenetekkel dolgozhatunk, hanem például *CD*, vagy ***video***bemenetet is megadhatunk forrásnak.

Egyszerre akár ***16 sáv***ra rögzíthetünk egy időben, amennyiben rendelkezünk ennek megfelelően komoly hangkártyával. Külön-külön azonban rengeteg sáv áll rendelkezésünkre, legalábbis összeszámolni még sosem tudtam őket (mivel a végtelen nem megszámlálható). Annyi biztos, hogy bőven elég akármilyen komplex hangszerelésű zene rögzítéséhez. Sztereó és monó sávokat egyaránt használhatunk.

A sávok hangerőszabályzása fantasztikusan van megoldva, mivel külön is lehet őket állítani, akár a felvétel közben is. Így már tényleg úgy érezhetjük magunkat, mintha a világ legkényelmesebb stúdiójában lennénk (a saját szobánkban).

Fájlok importálása és exportálása

Ez egy nagyon fontos fejezet az ***Audacity*** jellemzői között. A program képes arra, hogy meglévő zenéinket importálhassuk, majd szerkesszük őket. Például kedvenc zeneszámunkat betöltjük, kivágjuk belőle a gitárszólót, majd rájátszunk egy sajátot. Ennél szabadabban nem is engedhetne bennünket az alkalmazás.

Még ennél is magával ragadóbb, hogy egyenesen ***wav***, vagy ***mp3*** formátumba exportálhatunk, azaz menthetünk el komplett projekteket. Igazából a ***wav*** nem is olyan fontos, mert majdnem mindegyik zeneszerkesztő képes ***wav***-ba menteni, viszont az ***mp3*** funkció egy merőben új újítás. Ehhez természetesen szükségünk van a ***LAME*** nevezetű ***mp3-kódoló***ra, amit a program nem tartalmaz. Ennélfogva el kell látogatnunk a ➲ http://lame.sf.net weblapra, hogy letölthessük.

További támogatott fájlformátumok: ***aiff***, ***au***, ***ogg***. A program jelenleg nem támogatja a ***lossless formátum***okat (***flac***, ***shn***, stb.), ahogy a ***Windows-formátum***okat (például ***wma***) sem.

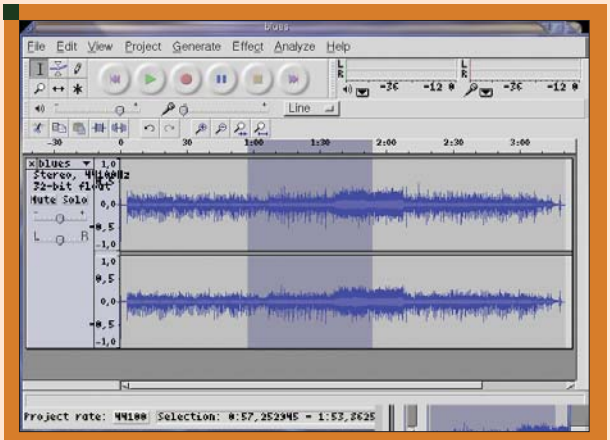
Szerkesztés

Az összes megszokott szerkesztőeszköz rendelkezésünkre áll: kivágás, másolás, beillesztés, törlés. Kijelölni sávrészleteket az egérrel tudunk. Létezik nagyító-kicsinyítő funkció is a programban, ami megkönnyíti a hosszú zenefájlok áttekintését. Természetesen ennek a fordítottja is igaz, mivel akár az utolsó ***Hz*** rezdülésig is nagyíthatunk egy sávot, így „milliméter pontossággal” dolgozhatunk. Korlátlan mennyiségben visszavonhatunk bármilyen elrontott mozdulatot, mert ***Undo-Redo*** lehetőség is van a programban. Ez nagyon hasznos tud lenni egy véletlen törlésnél vagy rossz helyre történő beillesztésnél. Amint már említettem, a sávok száma is korlátlan, melyeket akár egybe is tudunk keverni az intelligens ***mixer*** segítségével. A ***fade-in*** (előtűnés, most felerősödés) és a ***fade-out*** (eltűnés, jelen esetben elhalkulás) funkció is fontos szerepett kapott a programban, mivel elég népszerű dolog, mióta világ a világ.

Az ***Audacity*** (akár a *CoolEdit*) projektekkel dolgozik, melyeknek ***.aup*** a kiterjesztése. Ha a munkánkat projektbe mentjük, akkor a legközelebbi megnyitásnál onnan folytathatunk mindent, ahol abbahagytuk előzőleg. Célszerű projektbe menteni az olyan zenedarabokat, ahol a sávok még nincsenek keverve, illetve vannak kihagyott vagy befejezetlen részek.

Effektek

Kicsit bajban vagyok ezzel a résszel, mivel én kevés beépített effektet használok, sőt, őszinte leszek: semennyit. Multieffekten keresztül szoktam játszani, aminek semmi köze az



3. ábra Hangsáv szerkesztése

*Audacity*hez. Hogy kis fogalma legyen az embernek erről, elmondom, hogy multieffekt bármilyen hangszerhez kapható, illetve programokba is bele van építve, ahogy ebbe is. Sok effekt együtt használva – talán ez lehet a legjobb meghatározása a multieffektnek. Persze nem feltétlenül kell a különféle effekteket együtt használni, de így ideális. Aki nem rendelkezik „kézzelfogható" effektekkel, annak valamilyen effektszimulációs programot kell keresnie, illetve az *Audacity*ben alapból elhelyezett effekteket használhatja. No, lássuk miből élünk!

Szóval most, hogy kicsit utánajártam, igazán nem kell szégyenkeznie annak, aki nem rendelkezik tényleges „fémből és műanyagból készült lábbal kapcsolgatható" effektekkel.

Az *Audacity*ben jelentős effektpark található. Nézzünk ebből néhányat. Vegyük először az *Echo* nevű effektet. Ez visszhangot jelent, ami nélkülözhetetlen eleme az összes zenének. Enélkül a gitár, a dob, az ének száraz lesz, egyszóval nem életszerű. A gépben nincs levegő, nincsenek tárgyak, amik visszaverhetnék a hanghullámokat ezzel teret adva a zenének. Tehát az *Echo* kötelező darab.

Egy nem túl kötelező, ámde annál érdekesebb effekt a *Wah-wah* névre hallgató. Biztosan mindenki ismeri, még az is, akinek így névről nem ugrik be. *Jimi Hendrix Voodoo Chile*-ja, vagy a *Guns N' Roses Sweet Child O' Mine*-jának gitárszólója örök érvényű darab. Ezekben mind fellelhető a *Wah-wah* effekt.

Talán a legfontosabb effekt a programban a *zajzár*, amivel el lehet tüntetni mindennemű háttérzajt zenénkből, s ezzel teljesen polírozottá lehet tenni azt. Másik érdekesség, melynek muszáj e cikkben szerepelnie a következő: létezik egy olyan beépített effekt a mi szeretett ingyenes programunkban, ami bizony még a fizetős, nagy testvéreiben sem található meg mindig. Ez egy olyan funkció, amit a legtöbben tanulásra használhatnak. Adva van egy effekt, mely képes egy bizonyos zenerészletet lelassítani anélkül, hogy az a hangmagasságából bármit is veszítene. Vagy akár ez fordítva is eljátszható, felgyorsíthatunk egy lassú részt anélkül, hogy egyre magasabb és vékonyabb lenne a hang. Példánk legyen a következő: egy kezdő gitáros tanonc meg van őrülve a *Metallica* gitárosáért, *Kirk Hammett*-ért, ezért elhatározza, hogy olyan jól fog gitározni, mint ő. Begrabbeli az *Enter Sandman* című dalt a *Fekete Album*ról, majd betölti az *Audacity*be. Kivágja a szólóját, majd ezzel az effekttel lelassítja, hogy ezzel megkönnyítse annak elsajátítását.

További rengeteg effekt található a programban, amit mindenki felfedezhet saját magának kényekedve szerint (például *phaser*, *flanger*, *delay*, stb).

Hangminőség

Alkalmazásunk támogatja a *16-bit*et, *24-bit*et és a *32-bit*et (lebegőpontos) is, akár szerkesztés, akár felvétel céljából. A frekvenciát *8KHz*-től *96KHz*-ig állíthatjuk, annak megfelelően, hogy mit szeretnénk. Ez különbözően van fokozatokba szedve: *telefonminőség*, *beszédminőség*, *kazettaminőség*, *CD-minőség*, *DAT-minőség* (*Digital Audio Tape*, minden idők legjobb minőségű hanghordozója). Ezen felül a program képes a különböző minőségő sávokat egybekeverve valós időben automatikusan átalakítani.

Archiválás, digitalizálás

Sokszor előfordul velünk az, hogy valamilyen fontos családi emlék, vagy ritka koncertfelvétel csak *kazettá*n, vagy *bakelit lemez*en van meg. Mindkettőről tudjuk, hogy az idő múlásával jelentősen romlik az ezen tárolt hangsávok minősége. Többet nem kell emiatt aggódnunk, mivel az *Audacity* kiváló digitalizáló eszköz is egyben. Csatlakoztassuk a *magnó*, vagy *lemezjátszó* kimenetét a hangkártyánk *Mic* bemenetébe, indítsuk el a lejátszást, s közben ezzel egyidejűleg indítsuk el a felvételt az *Audacity*ben. Nincs más dolgunk, mint a „megszokott módon" hátradőlni és megvárni, amíg az ideiglenes „csere-bere fogadom" megtörténik.

Remélem sikerült minden érdeklődő figyelmét felkeltenem legalább annyira, hogy kipróbálják a programot. Természetesen sem az idő, sem a hely nem elég ahhoz, hogy minden egyes funkcióról írjak, ezért bátran ajánlom az on-line dokumentációját az alkalmazásnak. Ez elérhető a ➲ http://audacity.sourceforge.net/manual-1.2/ címen. Alapos leírás angol nyelven mindenre kitérően (az általam kihagyottakra is, például menüszerkezet, bővítmények, stb).

Összefoglaló

Álljon itt néhány személyes véleményem a programmal kapcsolatban:

- Rendkívül könnyen használható
- Nem igényel különösebb beállításokat használat előtt
- Megbízható, pontos, átlátható
- Kezdők és haladók egyaránt használhatják

Apagyi György, (killall)
(killall@linuxforum.hu)

24 éves, jelenleg az ELTE programozó matematikus szakán másodéves hallgató. Hobbija a zene (gitározás), az olvasás (Stephen King) és a számítástechnika (Linux, Unix, VMS).

KAPCSOLÓDÓ CÍMEK

Egy az Audacityvel készült kis szösszenetem:
➲ http://valhalla.eof.hu/audacity/blues.mp3

A Blender használata (8. rész)
Az animációkészítés alapjai

Nem ígérem, hogy a cikk végigolvasása után Hollywood-ot megszégyenítő animációs filmeket tudunk majd készíteni, de abban biztos vagyok, hogy más szemmel nézzük majd a Jégkorszakot, a Szörny Rt-t vagy a Csodacsibét, ha ráébredünk: nem is olyan egyszerű dolog ez az animáció.

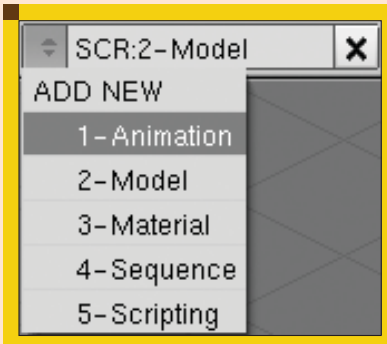
Ha mozgóképet akarunk készíteni, sok-sok képet kell egymás után raknunk. Ahhoz, hogy az eredmény folyamatos kép legyen, legalább 25-30 képkocka kell másodpercenként, ami egy fél perces animációnál is 750-900 képkocka. Ha ezeket mind kézzel kellene elkészítenünk, bizony hamar megunnánk, arról nem is beszélve, hogy néhány évtizedig biztos eltartana egy animációs film elkészítése. Szerencsére itt van nekünk a ***Blender***, ami meglehetősen jó képességekkel rendelkezik az animáció terén is, így sikeresen felveszi a versenyt a drága, zárt forráskódú társaival szemben. Nem csak egyszerű modellező/renderelő program, de létrehozhatunk vele animációt, amit aztán szerkeszthetünk, megvághatunk, szinkronizálhatunk, majd az egészet exportálhatjuk a többféle videó formátum egyikébe.
A ***Blenderben*** alapvetően négyféleképpen tudunk animálni:

- Egy teljes objektumot mozgathatunk a térben úgynevezett ***KeyFrame***-ek, és ***IPO Görbék*** segítségével. Ezt mindenképp érdemes elsajátítani, a többi módszer ugyanis erre épít.
- Használhatjuk a ***Constraints*** panelt, amivel egy objektumot dinamikusan egy másik felé fordíthatunk, vagy megadhatunk neki egy görbét, amin az animáció közben végighalad. Vagy esetleg az előző kettőt egyszerre szeretnénk? Kiválóan alkalmazható például a kamera mozgatására, ha egy objektumot akarunk követni.
- Objektumunknak megadhatunk egy csontvázat, amit aztán animálva az egész objektumot deformálhatjuk. A ***Character Animation*** – ahogy a ***Blender*** hívja – eléggé összetett folyamat, így vele csak a következő számban foglalkozunk.
- ***Vertex Key***-ek segítségével a meshünk vertexeit is animálhatjuk. Erre a módszerre nem fogunk kitérni, mivel nagyon hosszadalmas és precíz munkát igényel, és mivel az esetek többségében a csontvázak alkalmazása jóval célravezetőbb.

Az első két módszerrel az objektumot mozgatni, míg a második, illetve harmadik módszerrel deformálni tudjuk. Sajnos egy cikkbe nem férne el a lehetőségek teljes körű bemutatása, így most csak az objektum animációról lesz szó.

Előkészületek

A cikksorozat 2. vagy 3. részében már szó volt róla, hogy a ***Blender*** felhasználói felülete tetszőlegesen átalakítható a különböző igényeknek megfelelően. Mindezidáig erre nem nagyon volt szükség – ami azt illeti, most sem kötelező – de nagyban megkönnyítheti a munkánkat, ha átváltunk az ***Animation*** elnevezésű ablak elrendezésre. Ezt pedig a következő képen tehetjük meg: A ***Blender*** képernyőjének

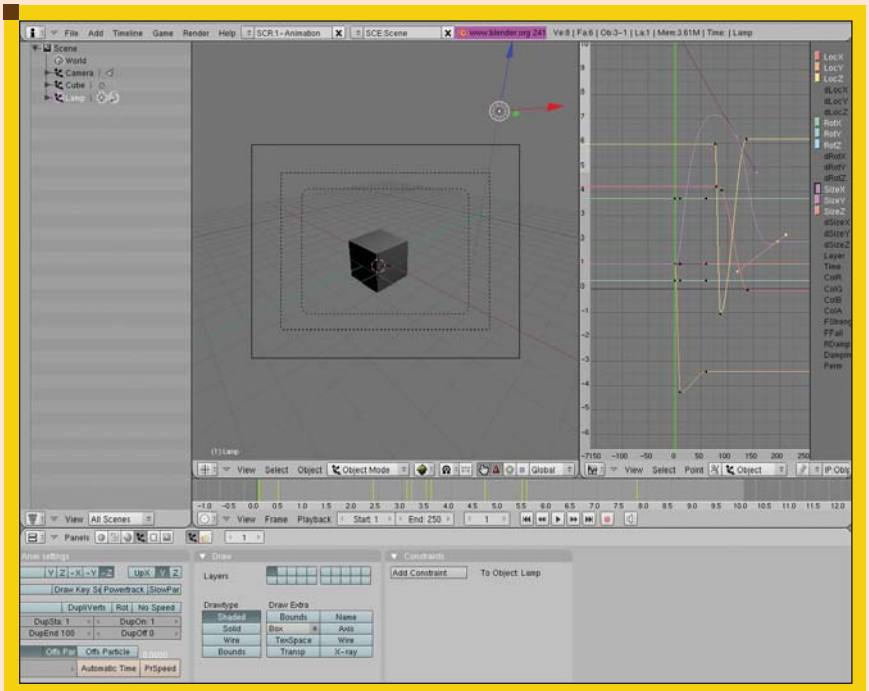


1. ábra Átváltás az Animation elrendezésre

tetején, a menüktől jobbra található egy ***SCR:2-Model*** elnevezésű mező (*1. ábra*). Ezt *„**1-Animation**”*-ra átállítva egy teljesen új elrendezést kapunk (*2. ábra*). Jobbra fent látható az ***Outliner*** nevű ablak, ebben az objektumokat látjuk szép, hierarchikus elrendezésben. Középen a megszokott ***3D View***, jobb oldalon pedig az ***IPO Curve Editor*** található. Róla később részletesen lesz szó. Gondolom az alsó ***Buttons Window*** sem idegen senkinek, viszont a felette található vékonyka ***Timeline*** méretével ellentétben egészen hasznos kis ablak. Nem árt megismerkedni vele.

Képkockák

Frame?... Az animáció képek egymásutánjából áll. ***Blenderben*** ezt a képkockát ***Frame***-nek hívjuk. Azt az értéket, ami megadja az egy másodperc alatt látott frame-ek számát ***Frs/sec***-nek, röviden ***fps***-nek (***Frames Per Second***) nevezzük. Minél nagyobb ez az érték,



2. ábra Az Animation nevű ablak-elrendezés

3. ábra A Frame Button

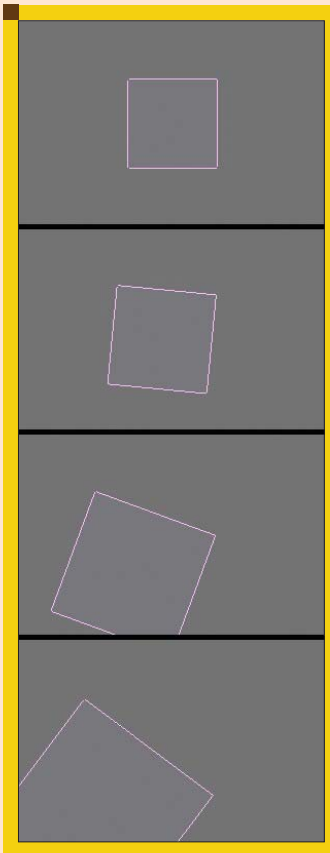
4. ábra Insert KeyFrame

az animáció annál „simább” lesz, tapasztalataim szerint azonban az előre beállított 25-ös érték tökéletesen megfelel. Ne feledjük el, hogy minden egyes képkockát le fogunk renderelni, ami igencsak erőforrás-igényes művelet. Az animáció alapegysége tehát ***Blenderben*** a ***Frame***. A ***Buttons Window*** fejlécén találhatunk egy árválkodó gombot, mindenféle jelzés nélkül, amin csak egy szám található (*3. ábra*). Akinek még mindig fogalma sincs mi lehet az, megsúgom: az való a frame-ek közötti váltásra. Kényelmesebb azonban a billentyűzeten található nyilak használata. A Jobbra és Balra nyilakkal egy-egy, míg a Fel és Le gombokkal tíz-tíz frame-et mozoghatunk előre és hátra, illetve ha a fentebb leírtak alapján átváltottunk az ***Animation*** nevű felosztásra, a ***Timeline*** -on kattintva tetszőleges frame-re ugorhatunk. Megjegyzem, a ***Timeline***-on kijelzett számok az időt jelölik, nem pedig a frame-eket, ez sokat segíthet.

KeyFrame

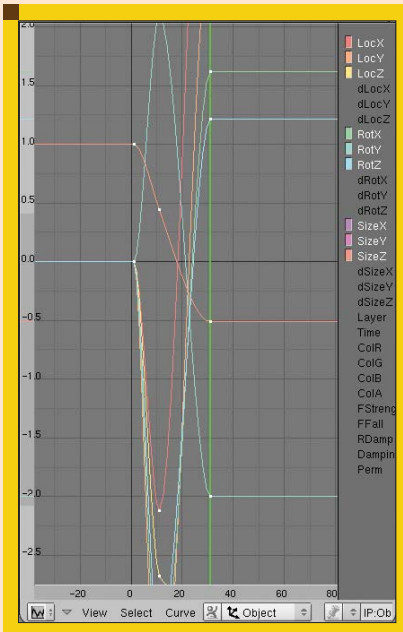
Rögtön próbáljunk is ki valamit: váltsunk át a tizedik frame-re, ott mozgassunk el egy objektumot, majd váltsunk vissza az elsőre. Az objektumot az új helyén találjuk. Most mozgassuk vissza az eredeti helyére, és menjünk újból a tizedik frame-re. Az objektum ott is az eredeti helyén van. Valami tehát nincs rendben. A ***Blendernek*** mozgatnia kellene az objektumot, miközben az első képkockáról a tizedikre megyünk. Vagy mégsem? Az animáció nem történik meg automatikusan akármilyen két frame között. ***KeyFrame***-eket (kulcs frame-eket) kell létrehoznunk, amik a mozgássorozat egy-egy fix pontját adják meg – például a két végét – és a ***Blender*** csak ezen frame-ek között hajlandó mozgatást végezni.

Mozgassunk vissza mindent az eredeti helyére, menjünk az első frame-re, és fussunk neki újra. Mielőtt bármit tennénk, nyomjuk meg az ***I*** billentyűt. A *4. ábrán* látható menü van előttünk. Itt választhatunk a ***Loc*** (Location, vagyis hely), ***Rot*** (***Rotate***, vagyis forgatás) vagy ***Size*** (vagyis méret) menüpontok közül, tehát animálni tudjuk az objektumunk helyét, helyzetét (elforgatását), méretét, vagy mindezt egyszerre. Most válasszunk ki a ***LocRotSize*** menüpontot, tehát mind a helyet, a méretet és az elforgatást animálni fogjuk. Az ***IPO Editor*** ablakban megjelenő vonalakkal egyelőre ne foglalkozzunk. Váltsunk át az ötvenedik frame-re, mozgassuk/forgassuk el, és méretezzük át az objektumot, majd ismét hozzunk létre egy keyframe-et az ***I*** billentyűvel. Ha mindent jól csináltunk, a ***Timeline***-on két függőleges



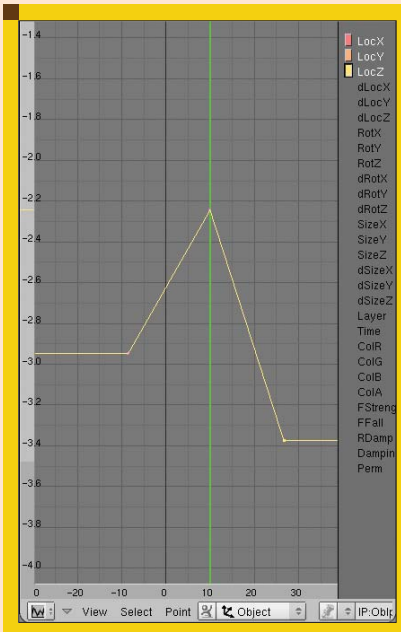
5. ábra Első animációnk

sárga vonal is látható, és a frame-ek közötti váltással az objektumunk mozogni fog. (*5. ábra*) Több ***KeyFrame*** hozzáadásával egész bonyolult mozgásokat is létrehozhatunk.

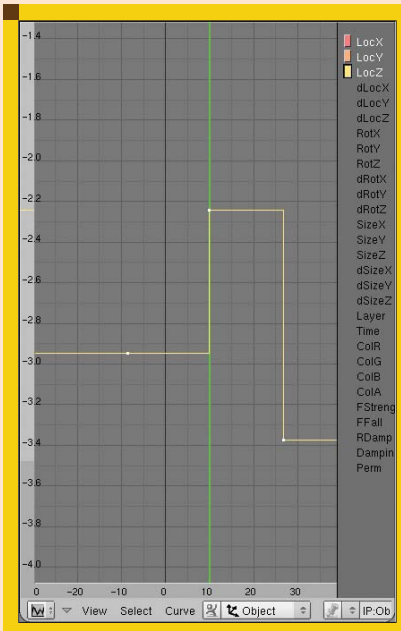


6. ábra Az IPO Curve Editor ablak

7. ábra Az interpoláció típusának beállítása



8. ábra Lineáris interpoláció



9. ábra Konstans interpoláció

Az IPO Curve Editor

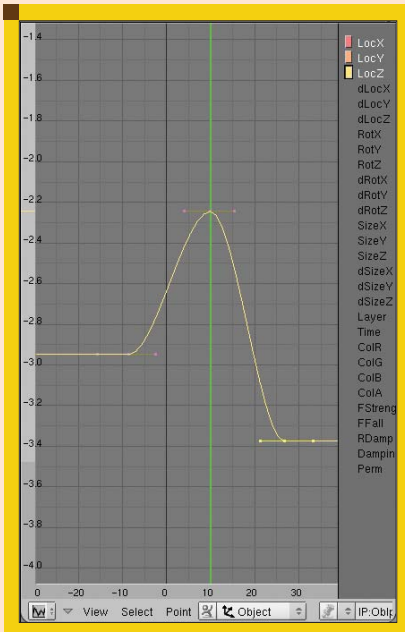
Ha kicsit eljátszottunk a *KeyFrame*-ekkel, biztos észrevettük, hogy a jobb oldali ablakban különböző görbék jelentek meg. (*6. ábra*) Ezek az ***IPO (InterPOlation)*** görbék, és ezek adják meg az objektum helyét, helyzetét és méretét az idő (pontosabban a frame-ek) függvényében. Minden objektumhoz saját görbék tartoznak. Három görbe adja meg egy objektum X, Y és Z irányú helyét, másik három az elforgatását, három pedig a méretét. Ezen kívül sokféle speciális görbe létezik, relatív elmozdulásra (***dLocX***, ***dLocY***, ***dSizeX***, ***dSizeY***, ***Time***, stb.) és egyéb tulajdonságok módosítására. Az ***IPO Editor*** jobb oldalán felsorolva látjuk ezeket a görbéket, mellettük láthatjuk a színüket, és hogy épp melyiket jelöltük ki. Akárhányszor, amikor egy keyframe-et hozunk létre, a görbe egy új ponttal bővül. Néhány megszorítástól eltekintve a görbéket ugyanúgy – legalábbis hasonlóan – szerkeszthetjük, mint az objektumokat:

- Van ***Görbe mód***. Itt a görbéket teljes egészében mozgathatjuk, használhatjuk a ***G (Grab)*** és ***S (Scale)*** billentyűket.
- Ha megnyomjuk a ***Tab***-ot, a görbe pontjait egyesével szerkeszthetjük, természetesen csak két dimenzióban.
- Mivel a görbének egyértelmű hozzárendelésnek kell lennie (emlékezzünk mit tanultunk az iskolában a függvényekről), nem tudunk olyan görbét létrehozni, ami egy időpillanathoz egyszerre két helyet/helyzetet/méretet rendel.
- A görbe a „semmiből jön” és a végtelenbe tart, tehát mindkét irányba van folytatása, amit minden görbénél külön beállíthatunk.

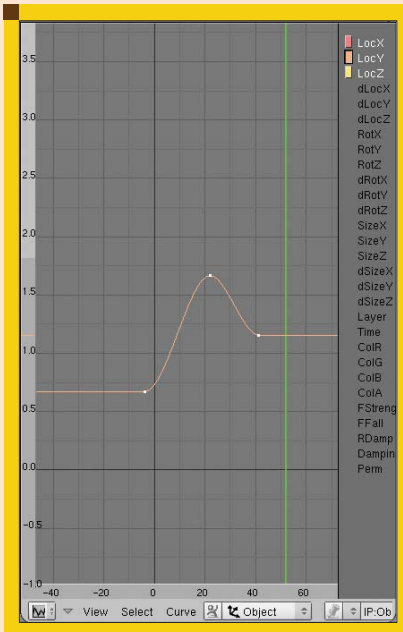
A fentieken kívül két fontos tulajdonsága van még az ***IPO*** görbéknek.
Az első az interpoláció típusa, vagyis az a mód, ahogy a ***Blender*** átmenetet képez a görbe fix pontjai között.
A ***T*** billentyűvel állíthatjuk be (*7. ábra*).
Ha a ***Linear*** módot választjuk, a pontok között egyszerű szakaszok jelennek meg. Ilyenkor a mozgás is állandó sebességű lesz a pontok között (*8. ábra*).
A ***Constant*** mód egyáltalán nem interpolál, az objektumok egész egyszerűen az adott frame-nél odaugranak a helyükre (*9. ábra*).
A legtöbb szabadságot a ***Bézier*** mód adja. Ilyenkor a görbét, mint bézier görbe szerkeszthetjük, így igaz rá minden, amit a görbék szerkesztésénél tanultunk. A ***H*** billentyűvel válthatunk a ***Free*** és A***ligned handle*** között, a ***SHIFT+H*** kombinációval ***Auto***, a ***V*** billentyűvel pedig ***Vector Handle*** állítható be (*10. ábra*).
A görbék másik fontos tulajdonsága az ismétlődés lehetősége. Mint feljebb írtam, ezek a görbék végtelenek, így mindkét irányba folytatódnak. Beállítható azonban, hogy a görbe ismétlődjön, így könnyen animálhatunk például rezgőmozgásokat, vagy folyamatos helyben forgást. Az ***IPO Editor*** fejlécén található ***Curve*** menü ***Extend Mode*** almenüjében állíthatjuk be.
Az alapértelmezett beállítás a ***Constant***. Ha ezt állítjuk be, a görbe az utolsó értéket felvéve vízszintesen folytatódik (*11. ábra*).
Az ***Extrapolation***-nál a görbe az utolsó keyframe-nél megadott irányban halad tovább (*12. ábra*).
A ***Cyclic*** nemes egyszerűséggel ismétli az első és utolsó ***KeyFrame*** közötti részt (*13. ábra*), míg a ***Cyclic Extrapolation*** úgy ismétli, hogy az utolsó ***KeyFrame*** illeszkedik az elsőre (*14. ábra*).
A *15. ábrán* található görbe például egy folyamatos forgatást ad meg a Z tengely körül.

Constraints

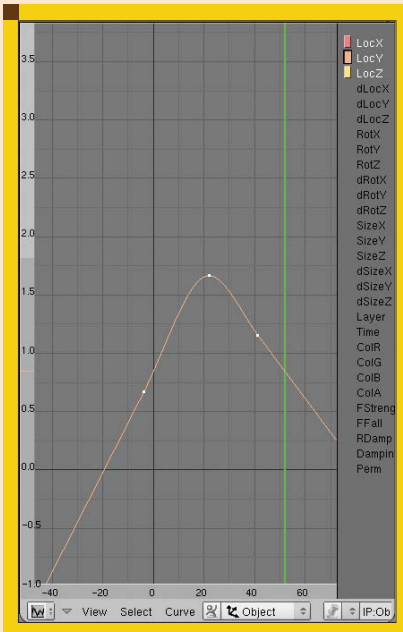
Bár a görbékkel való animálás egész pontos, egyes esetekben mégis hosszadalmas, vagy túlságosan bonyolult használni. A ***Blender*** nyújt számunkra egy eszközkészletet, amivel az ***IPO*** animációt kiegészítve egészen összetett



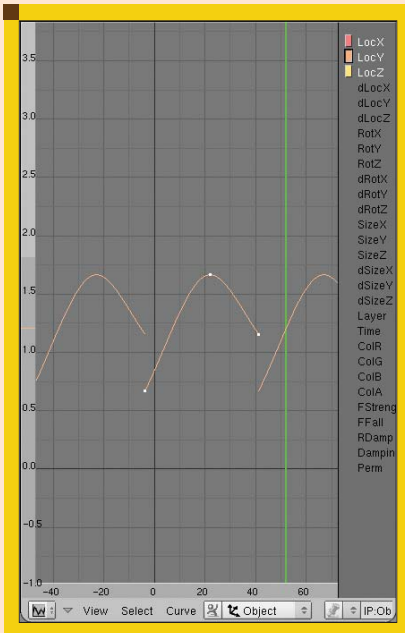
10. ábra Bézier interpoláció



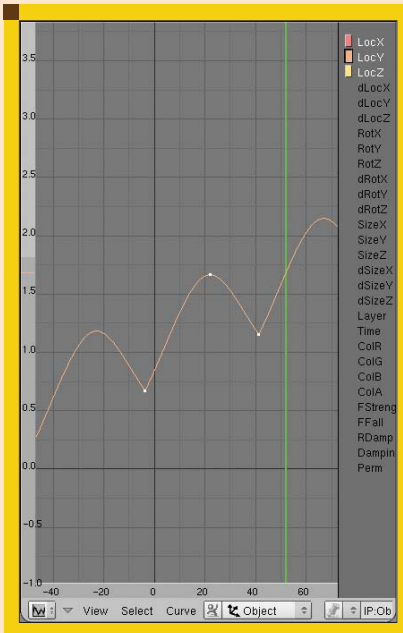
11. ábra Constant



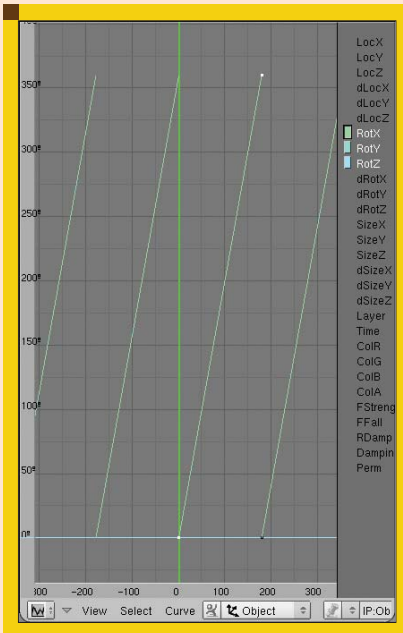
12. ábra Extrapolation



13. ábra Cyclic



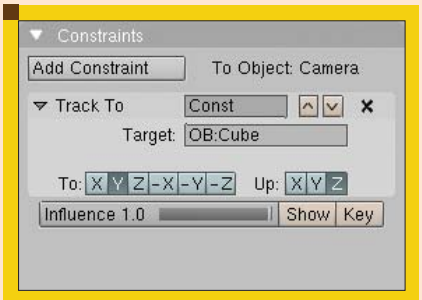
14. ábra Cyclic Extrapolation



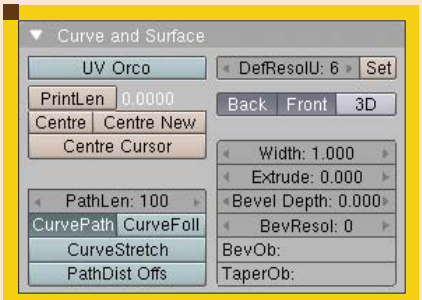
15. ábra A folyamatos forgatás IPO görbéje

mozgásokat is képesek vagyunk könnyen lemodellezni. Nézzünk egy példát. Szeretnénk, ha a kamera elhaladna egy objektum mellett úgy, hogy közben mindig az objektum felé fordul. Hagyományos módszerrel elég nehéz lenne ezt megoldani. Keressük meg a ***Constraints*** panelt (*16. ábra*), majd adjuk hozzá a ***Track To*** opciót-t, és a 16. ábrán látható módon írjuk be annak az objektumnak a nevét, amire szeretnénk, hogy nézzen a kamera. Figyeljünk, hogy a nevet pontosan adjuk meg, kis és nagybetű is számít. Meg kell adnunk, hogy az objektum melyik oldala nézzen a másik objektum felé, és ki kell választanunk egy irányt, ami felfelé mutat. Kameránknál az előbbi a -Z, utóbbi az Y irány. Ezek után akárhogy mozgatjuk a kamerát, az mindig a megadott objektum felé fog nézni. Egy másik hasznos dolog, hogy a ***Follow Path***-el egy görbéhez illeszthetünk egy objektumot, így az animáció közben a görbén fog mozogni. Ehhez a görbén be kell kapcsolnunk ***CurvePath*** opciót, amit a ***Curve and Surfaces*** panelen találunk, (*17. ábra*) majd hozzáadni egy tetszőleges objektumhoz egy ***Follow Path*** bejegyzést, és ott megadni a görbe nevét (*18. ábra*). Ilyenkor az objektum ugyan a görbén mozog, azonban nem követi annak vonalát, nem fordul együtt a görbével. A ***CurveFollow*** gombbal ezt kapcsolhatjuk be. A fenti kettőn kívül még számos ***Constraint*** található a ***Blenderben***. Közülük néhány:

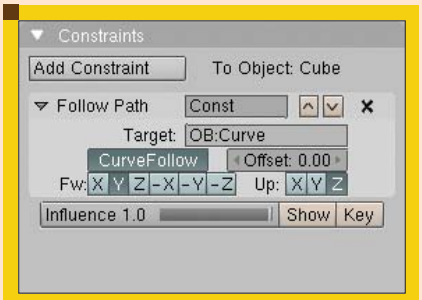
- ***Copy Location*** – Egy másik tárgy helyét „másolja”, tehát követi az adott tárgyat.



16. ábra Track To Constraint



17. ábra Görbe beállítása útvonallá



18. ábra Follow Path Constraint

- ***Copy Rotation*** – Egy másik tárgy helyzetét „másolja”, vagyis úgy táncol, ahogy a másik tárgy fütyül.
- ***Floor*** – Ezzel egy másik objektum X, Y vagy Z síkjához ragaszthatjuk az objektumot, így az csak az adott síkban, vagy afelett lesz képes mozogni.

Nem árt odafigyelni ez utóbbi technika használatakor, különben könnyen azon kaphatjuk magunkat, hogy a ***Blender*** értelmetlennek tűnő helyre teszi, vagy rossz irányba forgatja a kameránkat. Beállíthatunk ugyanis egyszerre több opciót is a fent leírtak közül, ilyenkor viszont nem mindegy a sorrend. A leggyakoribb eset, hogy kameránkat egy görbe mentén szereznénk mozgatni úgy, hogy közben mindig egy adott tárgyra néz. Ilyenkor előbb a ***Follow Path***-t állítsuk be, csak utána a ***Track To***-t. Hiszen ha elmozdítjuk a kamerát, az irányát újra irányba kell állítanunk. A sorrend tehát mozgatás, majd forgatás.
Bár megtanultuk az objektumok mozgatásának legalapvetőbb mozzanatait, a ***Blender*** eszköztára jóval nagyobb annál, mintsem egy pár oldalas cikkben azt le lehetne írni, ráadásul gyakorlás nélkül semmi sem megy (olyan jól).

A következő számban már deformálni fogjuk objektumokat, csontvázat adunk nekik, és bemutatom, hogy készíthetünk videót a ***Blenderben*** készített animációkból.
Reszkess ***Hollywood***!

Szalai András
(sly87@freestart.hu)

Jelenleg középiskolába jár, ahol informatikát tanul. Jövőre érettségizik. Hobbija a programozás és a biztonságtechnika, és a továbbtanulási szándékai is ilyen irányúak.

Örök háború

A cikkben két igen nagy sikerű és élvezetes kultuszjáték kerül gyors bemutatásra, melyek a második világháborúba kalauzolnak el bennünket.

A történelmünkben folyton tetten érhető hatalmi harcok megnyilvánulásai az idők folyamán számtalan háborúval sújtottak minket. Ezek szinte mindegyikének összecsapásait, viselő feleinek életét rengeteg regény és film dolgozta már fel. Nem hiába, hiszen sokunkat vonz a háborús történelem megismerése. Bizonyára akad olyan is közöttünk, akit kifejezetten taszít a harcok világa. Jómagam azonban az első csoporthoz tartozom.

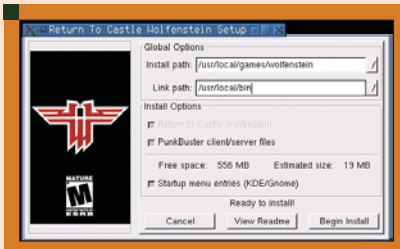
Leginkább az oly sok talány és ellentmondás „övezte" második világháború lefolyása érdekel: ennek történelmét már hosszú évek óta bújom, így bátran állítom, a téma valójában kimeríthetetlen mind a két oldalon. Ezt a jelenséget felismerve, a *PC*-s játékprogramok már kezdetektől fogva profitálnak a témából. Két részesre tervezett írásom első felében a modern, „*3D* korszak" két igen kiforrott és népszerű, küldetéseken alapuló *FPS* játékát szeretném megmutatni, melyek természetesen rendelkeznek linuxos binárissal.

Return to Castle Wolfenstein

A program (mely a *Gray Matter* / *Nerve* csapatok munkája) az *id Software* gondozásában jelent meg, akik a nagy sikerű *Wolfenstein* sorozat ezt megelőző részeivel anno a játékipar nagyjait adták ki kezeik közül. A címben említett epizód természetesen méltó a sikersorozathoz: ötletei a publikálását megelőző két évre is visszanyúlnak, így a sokáig „érlelt" elképzelések az alapul használt *Quake3* motorjával együtt igen meggyőző elegyet alkotnak. A *Win32* verzió 2001-es megjelenése után röviddel megszületett a linuxos bináris, így Szabad Rendszerünk alatt is megélhetjük a virtuális világban életre keltett harcokat.

A játék intrója látványosan vázolja fel a kerettörténetet: a háború idején a német hadvezetés igen nagy erőket mozgósítva kereste a *Szent Grált*. A katonák eközben felélesztették kősírjából (a múlt egyik legnagyobb párviadalában foglyul ejtett) démoni *Heinrich* grófot. Miután a szövetséges hadvezetés tudomására jutott, hogy a német oldal több-kevesebb sikerrel saját hasznára tudja fordítani a a felszabadult erőt, így az ellenséges vonalak mögé juttattak minket, a birodalom tényekedésének megakadályozására és szabotálására. A történetből következően a német katonák mellett zombik, élőhalottak és „félgépek" egyaránt ellenfeleink lesznek. Élvezetet fokozandó, az említett karakterek az esetek többségében egymás ellenfelei is, így sokszor taktikázásra késztethet a kérdés, kire támadjunk először.

Az ellenséges erők jól mozognak, csapatmunkára képesek, a zombik pajzsukat maguk előtt tartva védekeznek lövéseink ellen, vagy lidérceket idéznek. Ezen felül természetesen minden

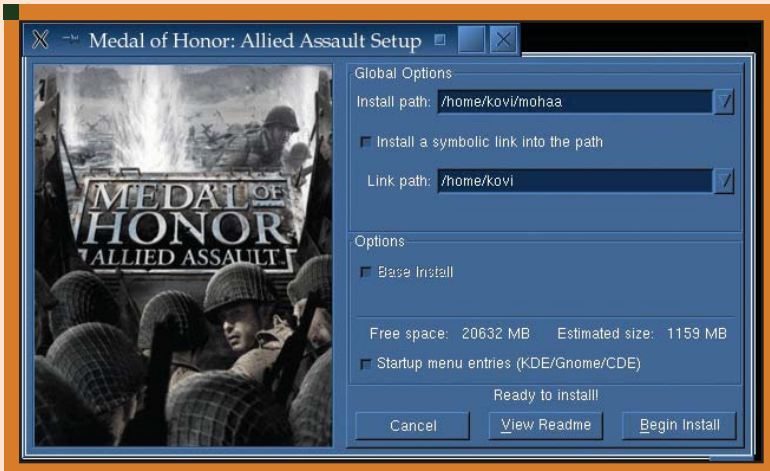


1. ábra A RTCW telepítője

2. ábra A RTCW világa

3. ábra A RTCW végjátéka

játékbeli karakter saját életet él, melyet rövid időre érdemes „lesből" követni: virtuális ellenfeleink roppant élethűen mozognak. A játék történelmi helyszínekre kalauzol el minket, ahol a realitást igen ügyesen keveri a legendával, így könnyen képernyő elé „ragasztja" a műfajra fogékony játékosokat. A legenda jelleg játékmenetben, környezetben, fegyvereinkben (például ***Tesla Cannon***) egyaránt fellelhető, a kézzel fogható történelmi tényeken felül.
Egy rövid (és idevágó) kitérőt engedjetek meg: az igazsághoz tartozik, hogy a jelen pártatlan kutatásai szerint a ***Führer*** valóban kereste a ***Grált*** (emellett a sokat említett ***„Végzet Lándzsája"*** a bunker eleséséig német kézben volt). A birodalmi hadsereget (a kiváló tábornokok mellett) olyan technikai fejlettség segítette, mely a mai modern haditechnika alapját képezi.
A játékban látható helyből felszálló repülő, az infrás keresővel ellátott puska, a ***V2*** rakéta, az ***MP40*** mind létező eszközök voltak a 40-es évek ***Németországában.*** Nem beszélve azokról a tényekről, melyek szerint pl. az ***USA*** a ***Horten*** fivérek repülőinek prototípusaiból és terveiből fejlesztette rettegett „lopakodó" repülőit, vagy arról, hogy a ***V2*** megtorló rakéta tervezője, ***Von Braun*** a ***NASA*** holdprogramjának technikai vezetőjeként szolgált a háború után. Tehát a látott és tapasztalt, misztikusnak tűnő dolgok ezen összetevőit tessék nagyon komolyan venni. Bizonyos részei a valóság egy igen hiteles (de legalább ennyire kevéssé közismert) képét tükrözik.
A küldetésalapú egy játékos üzemmód ötletekben igen gazdag, egyszerű továbbjutáson felül szabotázs akciókat, lopakodó bevetéseket, információszerzést egyaránt feladatként fogunk kapni. A programban (elődeihez hasonlóan) fellelhetőek titkos pályarészek, ahol aranyrögök, egyéb kincsek várnak ránk. A hanghatásokról és a dinamikus zenéről csak jókat mondhatok, a hatás kedvéért ezt fejhallgatón keresztül javaslom megtapasztalni. A több játékos részleg is élvezetes, taktikus jellege által „hús-vér" csapatunkkal valószínűleg hosszú időt fogunk zászlólopással, területbirtoklással, egyéb küldetésekkel tölteni.
A linuxos kliens és szerver állományok az ➲ ***ftp://ftp.idsoftware.com/idstuff*** címen érhetőek el, ***wolf.linux.verzió.run*** formában, telepítőbe ágyazva. Ezt a fájlt szuperfelhasználóként indítva, bináris tartalma alapértelmezésben a ***/usr/local/games/wolfenstein/*** útra kerül. A játék telepített linuxos motorja mellett természetesen szükségünk lesz az eredeti ***RTCW Win32*** verziójának adatállományaira is. Ezt egy ***WINE wrapper*** segítségével telepített játékból nyerhetjük ki legegyszerűbben, a virtuális környezetben installált programból.
A ***fake*** úton található ***sp_pak1.pk3, sp_pak2.pk3, pak0.pk3, mp_pak0.pk3, mp_pak1.pk3, mp_pak2.pk3*** állományokat mozgassuk a natív telepítési út ***/usr/local/games/wolfenstein/main/*** mappájába! Ezek után a ***WINE*** telepítette ***RTCW*** törölhető. (Megoldás lehet egy ***Win32*** platformon telepített programból kinyert adathalmaz is,



4. ábra MOHAA telepítő „Loki módra”

ha a *wrapperek* üzembe állítása problémát jelentene az olvasó számára).
A *pk3* adatokra ezek után olvasási jogot kell biztosítani mindenki számára (`chmod a+r`). Miután a telepítő az elérési útra készített `wolfsp` (*single player*) és `wolfmp` (hálózati *multiplayer*) kötést is, így a natív program felhasználóként terminálra gépelt ugyanezen parancsokkal indítható. Kompromisszummentes futtatásához szükséges hardverigénye 1 GHz körüli (teljes értékű) *x86* processzort és geometriai gyorsító feladatra képes, *OpenGL API*-n programozható videokártyát jelent. A kód kiforrott, gyakorlatilag mentes a bugoktól. Adataival együtt mérhető teljes helyfoglalása megközelítőleg 700 Mbyte. Mentéseit, képernyőképeit személyes mappánkban tárolja. *Linux* alatt bármely *pk3* állomány tartalma (textúrák, zenék, stb.) a használt tömörítés okán *Midnight Commanderrel* is böngészhető. Mindezzel óvatosan, jogvédett anyag! Aki esetleg nem ismerné ezt a programot (vagy még nem próbálta *Linux* alatt), a ➲ *http://www.idsoftware.com* honlapról kiindulva elérheti a program demóját is, így előzetes betekintést nyerhet ebbe a háborús környezetbe. Mint ahogy írásomból leszűrhető, ez a játék az egyik kedvencem. A cikk írása előtt nem sokkal kerültek bejelentésre a második rész készülő munkálatai. Addig is hódoljunk ezzel a résszel műfajnak: szenzációs játék, mely az elmúlt öt év alatt sem veszített fényéből. *„Multis”* részét szándékosan nem ecsetelem, ezt egy kiegészítő projekt, az *Enemy Territory* bemutatásával szeretném megejteni egy későbbi cikkben.

Medal of Honor – Allied Assault

A *MOHAA* a háborús környezet egy másik igen jelentős virtuális megvalósítása. Az előzőleg említett programmal közel egy időben jelent meg, motorja szintén *Quake3 engine* származék. Témája a *RTCW*-től eltérően minden legenda jelleget nélkülöz. Sokan ezért a merev történelmi hűségért szeretik leginkább: megjelenésekor a *„Ryan Közlegény Megmentése”* című, hitelesnek ható szuperprodukció egyik számítógépes megvalósításának volt titulálva a legtöbb szakmai sajtóban. Nem hiába: a program egy játékos módja az *USA* gyalogsági erejének egyik tagjaként játszható végig, a történelmi helyszínek és bevetések pontosan másolt világában, akár a partraszállást is „személyesen” átélve. Ellenfeleink erősen taktikusak, itt inkább ezen van a fő hangsúly, mint a mozgáson: csakúgy, mint az igazi háborúban, az elsődleges veszélyforrást a rejtőző ellenfeleink, illetve a nehezen rálátható területről érkező támadások jelentik. A játék természetesen küldetésalapú, mely bizonyára egy-két igen nehezen megoldható feladattal fogja szembesíteni még a gyakorlott megszállottakat is. Bevetéseink során kapunk majd személyi kísérő, lopakodó, gépágyús, tankos küldetéseket egyaránt. Nem ritkán ezeket időre kell teljesítenünk, ami által az egy játékos szál hosszú játékidőt ígér. A több játékos üzemmód nem kevésbé élvezetes: természetesen mindkét fél oldalán beszállhatunk a küzdelmekbe, több kaszt közül választva egyszerű felderítő egységen át, medikuson keresztül, akár taktikai mesterlövész szerepet is betölthetünk.
A program a *2015* csapat munkája, akik sajnos nélkülözték a linuxos támogatás implementálását, így a kód hivatalosan nem rendelkezik natív binárissal. Hála az *Icculus* gárda tevékenységének (elsősorban *Ryan Gordon* szakértelmének) az alapul szolgáló, módosított *Quake3* motorjának nem hivatalos portolása azonban folyamatban van. A projekt a cikk írásakor már jó ideje béta 3 fázisban lelhető fel: képi világa és fizikája egy-két jelentőséggel

5. ábra Omaha Beach, 44.06.06.

6. ábra Komoly utcai harcok

nem bíró helyen még hibázik, hangszolgáltatása pedig „alfateszt" állapotú. A Szabad Rendszer ***Medal of Honor***-jának „fejlesztése" sajnos igen lassan halad, bár a folyamat vélhetően gyorsulni fog a ***Quake3*** motor közelmúltbéli szabaddá tétele által. A linuxos, disztribúciótól független béta bináris elérhető a ➲ ***http://icculus.org*** oldalon „csupasz archívként", illetve a ➲ ***http://liflg.org*** linken ***Loki*** alapú telepítőbe ágyazva. Első esetben, a kódot egyszerűen a letöltött ***mohaa.linux.verzió.tar.bz2*** archív tetszőleges helyre történt kicsomagolásával lehet üzembe állítani. Mivel a ***RTCW*** esetében ismertetett módon itt is szükségünk lesz a teljes program adatcsomagjaira, így a ***Win32 MOHAA*** telepítőlemezekről installálnunk kell a játékot egy ***wrapper*** segítségével, annak ***„fake c"*** útjára. Az ott megjelent ***/main*** könyvtárat almappástul be kell emelnünk a natív bináris útjára, közvetlenül a futtatható állomány mellé.

Ezek után a ***wrapper*** által telepített ***Win32*** játékverzió törölhető (***WINE megoldás*** helyett természetesen itt is használható a valódi ***Win32*** környezetben történő beüzemelés, ahonnan szintén elérhetjük a kívánt adatokat). A beemelt ***/main*** mappára olvasási jogot kell biztosítanunk mindenki számára, rekurzív módon. Ezek után a játék a program gyökerében található indító állománnyal, terminálra gépelt `mohaa_lnx` parancs által futtatható.

Az említett második esetként használható a ***LIFLG*** csapat ***Loki*** alapú ***mohaa.verzió.run*** telepítője: szuperfelhasználóként indítva a bináris tartalma (amely megegyezik a ****.bz2*** állományban foglaltakkal) automatikusan a kívánt helyre kerül, miután a bekért ***MOHAA Win32*** telepítő korongjairól megpróbálta a szükséges adatokat a helyére tenni. Kérésünkre indító linket készít az elérési utakra, illetve a grafikus interfészünk menüjébe is beépül. A program hardverigénye valamivel magasabb, mint a ***RTCW*** esetén: itt már elkél egy testesebb ***x86*** központi egység is, a ***3D*** grafikus hardveren felül.

Az adatállományok jogvédett tartalma ebben az esetben is böngészhető.

Fontos ismét megjegyeznem: a játék natív portja béta fázisban van. Ennek ellenére ez a tény valószínűleg nem fogja visszatartani a műfaj rajongóit. Megnyugtatásként mondom: a ***MOHAA*** próbaverziós binárissal is végig játszható, ***Linux*** alatt. Köszönjük, ***Ryan***!

Nekem pedig ismét csak annyi dolgom maradt hátra, hogy megköszönjem a figyelmeteket, és tartalmas időtöltést kívánjak.

Kovács Zsolt (kovi@linuxforum.hu)

Quake fanatikus. Négy éve a debreceni linuxosok egyike. Töretlenül hisz a Slackware terjesztésben.


Linuxvilág
www.linuxvilag.hu