Beköszöntő

Linuxvilág

Íme új korszakunk harmadik lapszáma.
Novemberben ismét több új hazai szerző bukkan fel.
Közülük ***Radics Péter*** a ***Gambas*** fejlesztőrendszert mutatja be, ami várhatóan a ***Visual Basic*** linuxos megfelelőjévé növi majd ki magát. A cikkből megtudhatjuk, hogy bár fejlesztése még eléggé az elején jár, szolgáltatásai már most is olyan eszközzé teszik, amivel filozófiájának megfelelően „könnyen, gyorsan, hatékonyan" hozhatunk létre akár parancssori, akár grafikus felülettel rendelkező alkalmazásokat.
Szurmai László a ***Konqueror*** böngészőt mutatja be, ami egyben fájlkezelő, komplex megjelenítő amúgy pedig tetszés szerint testreszabható mindenes.
Apagyi Györgynek (killall) rögtön két cikke is megjelenik ebben a lapszámban. Az egyikben néhány tanácsot olvashatunk arról, hogyan lehet az internetes fórumok segítségével hatékonyan megoldani azokat a problémákat, amelyek például egy új rendszer telepítése után bukkannak elő.
Gyuri maga is egy fórum moderátora, tehát érdemes megszívlelni a tanácsait. Jellegénél fogva ez a cikk nyilván elsősorban a kezdőknek szól, de ne bízzák el nagyon magukat a haladók sem: a témáról a közelmúltban egy egész könyv jelent meg, amit nem kifejezetten amatőrök írtak...
Karai Csaba azon magyar fejlesztők egyike, akik részt vesznek valamilyen nemzetközi nyílt forrású projektben. Csaba történetesen a sokak által kedvelt ***Krusader*** fájlkezelőt fejlesztő csapat tagja. Cikkében „beszervezésének" történetét meséli el.
Magazin rovatunkban igazi szenzációt olvashatunk ***Halusz Léna*** tolmácsolásában. Új magyar Nobel-díjassal büszkélkedhetünk, aki a nemzetközi elismerést ráadásul linuxos témáért kapta: egy pingvinben átvitel közben uralkodó nyomásviszonyokat tanulmányozta, különös tekintettel az output megvalósítására. Érdekes, tanulságos írás...
Folytatódik néhány sorozatunk is.
Garzó András már az ***IPv6*** bemutatásánál tart, ***Auth Gábor*** újabb szépségeket varázsol ***PovRay*** segítségével, ***Fülöp Balázs*** pedig ezúttal egy egyszerű játékot fejleszt ***Javaban***. ***Szalai András Blenderről*** szóló sorozatának harmadik részében a világítás használatát mutatja be. ***Novák Áron*** szintén a harmadik résznél tart az ***SVG*** világát bemutató sorozatával, amelyben az alkalmazási területeket tárgyalja.

Kellemes olvasgatást, jó szórakozást kíván a Linuxvilág stábja!

Hírek

Nyitás vagy távozás?

Az *USA Massachusetts* állama nemrég hivatalosan is bejelentette, az állami irodák 2007 januárjától kezdődően kizárólag nyílt dokumentumformátumokat használhatnak. Ahogy az állam informatikai vezetője elmondta, lépésük – amely viszonylagos távoliságával megfelelő felkészülési időt biztosít – egyetlen gyártó ellen sem irányul, ám amelyik gyártó nem képes megfelelni a feltételeknek, annak termékeit fokozatosan eltüntetik a hivatalokból. Az állam ugyanakkor támogatja az *OpenDocument* és a *PDF* formátumot, ezek lesznek az irodai dokumentumoknál alkalmazott szabványos formátumok. Csak remélhetjük, hogy a hasonló lépések egyre inkább rákényszerítik majd a zárt formátumokat alkalmazó gyártókat, hogy vagy támogassák a nyílt szabványokat, vagy teljes mértékben nyissák meg a közönség előtt a saját specifikációikat. Amíg hasonló ösztönzéseket nem kapnak, addig ezt – saját érdekeiket szem előtt tartva – aligha fogják megtenni.

Kezes bárány

A *QCD Microsystems* újabb taggal bővítette az *InterStructures* kiszolgáló-felügyeleti rendszer moduljainak sorát – a *Samba PDC* nevű beépülő modul segítségével a rendszergazdák immár a *Samba* kiszolgálók felügyeletét is elvégezhetik a Microsoft termékekből ismert felügyeleti konzol (*MMC*) alkalmazásával. Az *InterStructures* rendszer ezzel minden alapszolgáltatás – *Samba* fájlkiszolgálók, *Active Directory* felügyelete, *DHCP* kiszolgáló, *Apache HTTP* kiszolgáló, *Squid* webproxy és egyéb *Linux* alapú szolgáltatások – kezelését támogatja. A *QCD* terméke kiválóan alkalmas arra, hogy a *Windows* és *Linux* alapú kiszolgálók közötti különbségeket elfedje, a windowsos rendszerekhez szokott rendszergazdák számára is könnyen kezelhetővé tegye a linuxos gépeket. Az *InterStructures Fedora Core*, valamint a *Red Hat* és a *SuSE* vállalati terjesztései alatt futtatható, árát a gyártó gondosan titkolja.

➲ *www.interstructures.com*

Gombok helyett LCD

A *Sony Cyber-shot DSC-N1* névvel jelentette meg legújabb digitális fényképezőgépét. Az *N1* 8 megapixeles felbontással dolgozik, a külvilágra 3-szoros közelítésre alkalmas *Carl Zeiss* lencséken keresztül tekint ki, a képeket pedig saját 26 MB-os memóriájában vagy *Memory Stick Duo* kártyákon tudja tárolni. Igazi érdekessége azonban az, hogy hátoldalán csak a közelítés-távolítás gombjának és néhány apró jelzésnek szorítottak helyet, ettől eltekintve az egészet elfoglalja a „hatalmas" méretű, 3"-os érintőképernyő – értelemszerűen ennek segítségével lehet minden műveletet elvégezni. A gép jelenleg csak előrendelhető, ára 500 dollár.

➲ *www.sonystyle.com*

SuSE 10

A *Novell* bemutatta a *SuSE Linux 10*-es változatát, a *SuSE* első olyan kiadását, amely az augusztusban indított *OpenSuSE* közösségi modell szerint, a *SuSE Linux Pro* által adott alapokra építkezve készült. A terjesztés a *Gnome 2.12*-es és a *KDE 3.4.2*-es változatát tartalmazza, szolgáltatásaival a tapasztalatlan és a profi felhasználók igényeinek egyaránt megpróbál megfelelni. Ugyancsak a csomag része az *OpenOffice.org 2.0* kiadásra szánt változata, a *Xen 3* virtualizációs megoldás, az *iFolder 3* általános fájlelérési szolgáltatás, valamint a *Mozilla Sunbird* naptár. A 10-es *SuSE* kétféle változatban készült el, az egyikben található néhány zárt jellegű alkalmazás is, például az *Adobe Acrobat Reader 7*, *RealPlayer* és *Flash Player*, míg a másik összeállítás kizárólag nyílt forrású összetevőkből épül fel. A *Novell* a terjesztést ingyenesen letölthetővé tette. Ehhez a változathoz értelemszerűen semmilyen támogatás nem jár, akik azonban hajlandóak kifizetni a dobozos változat 60 dolláros árát, azok telepítési támogatást és kézikönyvet is kapnak.

Szabad a dal

Ingyenes lett az *Opera* böngésző. Ennyi a hír, az *Opera* a továbbiakban licencdíj megfizetése nélkül is úgy használható, hogy közben nem kell hirdetési csíkokat bámulni. Aki eddig szerette az *Operát*, az most bizonyára örül. Aki szerette és megvásárolta, most a földhöz vághatja a kalapját. Aki eddig nem szerette, az most kapott egy indokot: próbálja ki. Az ingyenessé tétel okáról a cég weboldalán csak valami homályos utalás van arra, hogy végre elég nagyra nőtt a felhasználói tábor ahhoz, hogy megtehessék ezt a fontos lépést. A norvég böngésző 27 nyelven, 8 operációs rendszer alatt áll a felhasználók szolgálatára.

➲ *www.opera.com*

Számlázás webről

A kanadai ***Simian Systems SiteInvoice*** névvel barátságos felületű, nyílt forrású számlakezelő rendszert mutatott be. A web alapú ***SiteInvoice*** elsősorban a kisebb vállalkozások, egyéni vállalkozók számára nyújt megoldást a számlák egyszerű kezeléséhez, a kinnlevőségek követéséhez és olyan feladatok elvégzéséhez, amit ezek a vállalkozók általában nem szeretnek elvégezni. A ***SiteInvoice*** segítségével egyszerűen lehet ***PDF*** formátumú számlákat készíteni és elektronikus levélben elküldeni, a lejárt tartozásokról szintén e-mailben érkezik az értesítés. A program az adók terén széles körű testreszabhatóságot biztosít, így – megfelelő honosítással – elvileg akár itthon is használható. A program honlapja sajnos nemcsak e tekintetben szűkszavú, az egyik képernyőképen a pénznemek között szereplő Forint mindenesetre reménykedésre ad okot.

➲ ***www.sitelliteforge.com/siteinvoice***

APC ujjlenyomat-olvasó

Az ***American Power Conversion*** új, ***PCMCIA*** felületű ujjlenyomat-olvasó kártyát mutatott be hordozható gépekhez. A ***Password Manager*** a cég egyéb hasonló termékeinek sorába illeszkedik: a kínálatban ***USB*** csatolós és egérbe épített ujjlenyomat-olvasó egyaránt található.

A kiegészítő kártya használatából fakadó előnyök kézenfekvők: a sokfelé hurcolt gépeket egy további védelmi vonallal ruházza fel, ami a rajtuk található adatok sokszor bizalmas jellegét tekintve cseppet sem túlzás; ugyanakkor használatának kevéssé praktikus jellege, a gép oldalából kilógó kártya sérülékenysége is legalább ennyire szembetűnő. A kártya – különféle ***Windows*** változatok alatt futtatható – kezelőprogramja legfeljebb 20 ujjlenyomatot képes megjegyezni, ami egy hordozható gép esetében bőségesen elegendőnek tűnik. Egy érdekes adat: egy átlagos felhasználónak állítólag 17 jelszót kell fejben tartania, és a céges helpdeskeknek is gyakorta kell az elfelejtett jelszavak problémájával megbirkózniuk. A kártya ára 150 dollár, ami talán egyetlen megspórolt jelszóelvesztés alkalmával is megtérülhet.

➲ ***www.apc.com***

Zenei kiadás

A ***Nokia*** – éppen csak orrhossznyival maradva le a ***Motorola*** mögött – bemutatta első kifejezetten a zenebarátoknak szánt mobiltelefonját. A már korábban bemutatott ***6630***-as jelű készülék, megújított, ***Music Edition*** változata egyrészt zenelejátszó, másrészt ***3G***-s okos telefon, a ***Motorola ROKR*** készülékéhez képest azonban egészen más filozófiát követ. A ***6630***-as nem rendelkezik beépített zeneletöltő ügyféllel, így tulajdonosának nem sok esélye van arra, hogy út közben újabb dalokkal gyarapítsa a gyűjteményét – igaz, ez egyben azt is jelenti, hogy egyetlen szolgáltatóhoz vagy zeneműbolthoz sem kell kötődnie, illetve maga a telefon sem csak bizonyos szolgáltatóktól lesz megvásárolható.

A zenemobil általános jellemzői nagyjából az eredeti modellre emlékeztetnek, a 127 grammos készülék 1,23 megapixeles kamerát és 176 x 208 képpontos kijelzőt, támogatja az ***USB 2.0*** kapcsolatokat, illetve 256 MB-os memóriakártyával szerelték fel.

A vásárlók kapnak hozzá egy ***USB***-s ***MMC/SD*** kártyaolvasót, valamint átalakító kábeleket is, utóbbiak segítségével normál fejhallgatót is csatlakoztathatnak a telefonhoz, illetve akár asztali berendezésekkel is összekapcsolhatják. Utóbbi eszközöket a ***Nokia*** külön csomagban is elérhetővé tette, a ***Music Pack*** a cég számos más telefonjával együtt is használható.

➲ ***www.nokia.com***

Medgyesi Zoltán
(mz@rettesoft.hu)
A Linuxvilág hírszerkesztője. Szabadidejét legszívesebben a barátnőjével tölti, szeret autózni és bográcsban főzni.

Mi újság a rendszermag fejlesztése körül

Hosszú, szenvedésekkel teli életútja végén a *DevFS* immár végképp eltávozott a *Linux* kernelből. A *Richard Gooch* által írt modul évekig része volt a magfejlesztésnek, és kétségkívül komoly próbálkozásnak tekinthetjük a kissé elburjánzó */dev* könyvtár megzabolázására. A *DevFS* fejlesztése ugyanakkor mindvégig árral szemben haladt, hiszen számos bírálója akadt a közösségen belül. Ennek ellenére kijelenthetjük, hogy *Richard* egy nagyon hasznos eszközt adott a kezünkbe. Végül mégis azok a kritikusok győztek, akik „megoldhatatlan versenyhelyzetekre" és egyéb hasonló hibákra hívták fel a figyelmet, s így *Richard* mostanra teljesen eltűnt a kernelfejlesztők közül. *Greg Kroah-Hartman* és mások már korábban elkezdték fejleszteni az *udev* rendszert, amely így a *DevFS* utóda lett. Bár a 2.6-os fa stabilitása utáni kósza vágy a döntést kissé ellentmondásossá teszi, ennek a tábornak a befolyása nem lesz elegendő a dolog megfordításához. Így hát ég veled *DevFS*, derék próbálkozás voltál.

A közelmúltban számos helyről érkeztek hibajelentések, amelyek szerint a 2.4-es kernelt nem lehet lefordítani a *GCC 4*-es változatával. Bár ennek hatására érkeztek is olyan foltok, amelyekkel a probléma orvosolható, *Marcello Tosatti* szerint már túl késő ahhoz, hogy ezek bekerülhessenek a 2-4-es fába. A 2.6-os rendszermagtól eltérően a 2.4-es, 2.2-es és 2.0-ás fa fenntartói nem csatlakoztak az új irányzatokhoz, és továbbra is a stabilitás fenntartását vélik munkájuk legfőbb céljának. Ennek ellenére *Marcello*, aki az első 2.6-os rendszermag megjelenése óta koordinálja a 2.4-es fejlesztését egészen komoly változtatásokat is megengedett például az *IDE* alrendszerrel kapcsolatban. Számos új hardverhez készült meghajtó és sok olyan, egészen mélyreható folt is bekerülhetett a fába, ami nem föltétlen vált használára az áhított stabilitásnak. És mivel a 2.6-os változat fejlesztése csak nem akart lelassulni, *Marcellora* komoly nyomás nehezedett: egyre többen egyre több szolgáltatás beépítését várták tőle azok közül, akiknek ezzel párhuzamosan a 2.4-es mag stabilitására is szükségük volt. A *w.x.y.z* fa megjelenésével aztán ez a nyomás enyhült, így *Marcellonak* végre esélye volt arra, hogy szigorítson a szabályokon.

A *git* változatkezelő rendszer tovább növekszik és erősödik. Az *Andrew Morton* által fejlesztett *-mm* fa hamarosan elérhető lesz *git* formátumban is, bár magának *Andrew*-nak nincsenek tervei azzal kapcsolatban, hogy elkezdene használni bármiféle változatkövető rendszert a folyamatban levő fejlesztésekhez. Az *ALSA* projekt, akárcsak a *libata* már átállt *git*-re. *Marcello Tosatti* szintén ezzel a változatkezelő rendszerrel folytatja majd a 2.4-es fa fenntartását. *Linus Torvalds* még mindig meglehetősen aktív részese a projektnek, és bár a levelezési lista forgalma némileg visszaesett a viharos első hetek után, ez valószínűleg egyszerűen azzal magyarázható, hogy a tagok többsége már tisztába jött a rendszer működésének alapjaival, illetve hogy az újonnan jövőknek már nem kell az alapoktól indulva mindent elmagyarázni.

A változatkezelő rendszerek körül keletkezett fölfordulásban nem igazán lehet pontosat tudni az új *w.x.y.z* fa stabilitásáról. Mindazonáltal néhány fejlesztő, köztük *Jeff Garzik* és *Alan Cox* úgy vélik, ez a fa tényleg képes lesz egy stabil rendszermagot létrehozni a 2.6-os fa folyamatos, nagyszabású fejlesztései közepette is. A *w.x.y.z* fa két fő fejlesztője, *Greg Kroah-Hartman* és *Chris Wright* meglehetősen szigorúan irányítja a folyamatot. Nem egyszerűen csak begyűjtik és alkalmazzák a foltokat, hanem szigorúan ragaszkodnak a *Linus Torvalds* által megfogalmazott alapszabályokhoz azzal kapcsolatban, hogy mi, mikor milyen formában kerülhet be a fába. Bár ez a munka valószínűleg nem annyira érdekes, mint a valódi magfejlesztés, *Chris* és *Greg* kitartanak, a dolog haszonélvezői pedig mi felhasználók leszünk. *Martin J. Bligh* összeállított egy olyan automatikusan működő szkriptcsomagot, amely minden főbb kernelváltozatot (beleértve a *w.x.y.z* és a *-mm* fát is) lefordít és kipróbál 15 perccel a kibocsátása után. Ha az új rendszermaggal sikeres a bootolás, akkor *Martin* szkriptjei intenzív tesztelésnek vetik alá a „jelöltet". A játék végén a fordítás és a bootolás részleteiről napló, a teljesítménytesztekről pedig grafikonok készülnek, és az egész anyag azonmód kikerül a *kernel.org* megfelelő lapjaira. Ez tehát egy olyan projekt, ami önmagában nem tud megoldani problémákat, de képes fölhívni a figyelmet számos triviális hibára, illetve segít beazonosítani azokat a teljesítmény-problémákat, amelyek az egyes változatok fejlesztése során fölbukkannak. Segítségünkre lehet olyan nehezen felfedezhető problémák megtalálásában is, amelyekkel az átlagos felhasználók csak ritkán találkoznának.

A kernelfoltok kezelési rendszerébe a közelmúltban bevezetett *Signed-Off-By* bejegyzés jelentősen megkönnyíti annak nyilvántartását, hogy melyik módosítás kitől érkezett. Ennek igazából akkor lesz jelentősége, ha ismét kitör egy a *SCO* perhez hasonló perpatvar, hiszen pillanatok alatt meg lehet találni azt, aki a vitatott kódsorokat begépelte. *Linus Torvaldsnak* eredetileg mindössze ennyi volt a szándéka ezzel az újítással, ám bevezetésekor az egész dolog meglehetősen amorfnak tűnt, hiszen a legtöbb elképzelés nem volt kellően kidolgozva. Azóta viszont olyan új ötlet került elő, amelyek növelik a rendszer használhatóságát. Az egyik legfrissebb közülük a *From* fejléc bevezetése, ami mostantól a foltot tartalmazó e-mail törzsének első sora. Korábban a folt szerzőjének azt tekintették, aki a legalsó *Signed-Off-By* fejlécben szerepelt, ami egyrészt zavaró, másrészt nem is mindenki járt el e szerint. A *From* fejléc célja az, hogy semmiféle kétség ne merülhessen fel egy-egy folt szerzőjének kilétével kapcsolatban.

Zack Brown

Új termékek

Monarch Furia AMD Athlon 64 X2-es processzorokkal

A *Monarch Computer* bejelentette a *Monarch Furia* sorozatot *AMD Athlon 64 X2* kétmagos processzorral felszerelve. Az új *Furia* munkaállomások és asztali gépek 32-bites és 64-bites alkalmazásokat is képesek futtatni. Az *on-die* kétmagos *x86 PC* processzorokkal felszerelve a *Monarch* új munkaállomásai processzor magok közötti kommunikációs lehetőséget biztosítanak *CPU* sebességgel, valamit direkt hozzáférést a memóriavezérlőhöz és a *HyperTransport* technológiához.
➲ *www.monarchcomputer.com*

Nokia 770 Internet Tablet

A *Nokia 770 Internet Tablet*, egy internet böngészésre, és e-mail kommunikációra alkalmas zsebméretű eszköz. A *Nokia 770* nagy felbontású, *800x400*-as szélesvásznú, nagyítható kijelzője, és kijelzőn található billentyűzete, kimondottan alkalmassá teszi online tartalmak *Wi-Fi*-n keresztüli megjelenítésére. A *Wi-Fi*-től eltekintve, az eszköz képes egy kompatibilis mobiltelefonon keresztül, *Bluetooth* vezeték nélküli technológiát használva, csatlakozni az internetre. A 770-es, *Linux* alapú *Nokia Internet Tablet 2005 Software Edition* szoftvert futtat, amely sok népszerű szabad forráskódú technológiát tartalmaz. A *Nokia 770* megjelenésével összhangban, a *maemo* fejlesztési platform (➲ *www.maemo.org*) már elérhető, hogy segédeszközöket és lehetőséget biztosítson a szabad szoftver fejlesztőknek arra, hogy a *Nokiával* együttműködve hozzák létre a jövő eszközeit és *Internet Tablet* operációs rendszereit.
➲ *www.nokia.com*

PL-01025 1U Beágyazott Fejlesztési Platform

A *WIN Enterprises, Inc.*, bemutatta a *PL-01025*, nagy teljesítményű, rackbe helyezhető, *1U* magas beágyazott fejlesztési platformját amelyet az Internetre/hálózati eszközökre tervezett. A *SafeXcel 184x* társprocesszorral kiegészített számítási kapacitással a *PL-01025* képes kiszolgálni egy *Pentium M* processzort, és akár 8 GB *DDR* memóriát. Az eszköz *CompactFlash*, *Gigabit Ethernet*, és *PCI-X* csatlakozó helyekkel is rendelkezik. További csatlakozási felületként tartalmaz tíz *Gibabit Ethernet (10/100/1000)* és négy *Ethernet (10/100)* csatlakozást, valamint digitális *I/O* (négy be, négy ki) soros interfészt és egy *IDE* csatlakozót a 2.5\200\235 *HDD* részére.
➲ *www.win-ent.com*

HW400c/2 Kommunikáció Vezérlő

A legújabb darabja az *SBE HighWire* távközlési eszköz sorozatnak, a *HW400c/2*, egy intelligens *PICMG 2.16 CompactPCI I/O* processzor. Kiegészítve egy 1 GHz-es *PowerPC* processzorral, akár 1GB *SDRAM*-mal, két *PCI Telecom Mezzanine Card (PTMC)* csatlakozó hellyel, *Gigabit Ethernet* csatolóval, és *H.110* felülettel. Olyan kritikus telekommunikációs infrastruktúra alkalmazásokra tervezték, mint a média

átjárók, softswitchek, és a távoli csomópont kontrollerek, a két kiegészítő helyet a *PTMC 2*-es és *5*-ös konfigurációnak az általános *PMC* lapokon felüli kiszolgálására szánták. A központi számítási architektúra a *HW400c/2*-ben egy 1GHz-es *Freescale MPC7447A PowerPC* processzorra és *Marvell Discovery III* rendszer vezérlőre van alapozva. Akár 1GB *ECC DDR* memória is támogatott és ezen felül egy lapra integrált *Disk-on-Chip Flash* fájlrendszer tároló.
➲ *sbei.net*

Heroix Longitude

A *Heroix Longitude* rendszere egy ügynök nélküli, több platformon elérhető, az operációs rendszer és a rajta futó alkalmazások alkalmazás, megfigyelésére alkalmas ellenőrző és jelentéskészítő rendszer. Szolgáltatásai, esemény megjelenítés, grafikus mérőműszerek, nézetek, teljesítmény jelentések, grafikonon megjelenített információk a teljes rendszerről, így az informatikusok képesek kezelni a teljesítmény és kapacitás ingadozásokat, mielőtt az *IT* szolgáltatás minőségére hatással lenne. Ipari szabványokra alapozva, a *Longitude* 100%-osan web-kompatibilis, tartalmaz több mint 250 előre elkészített mérendő paramétert, melyekkel monitorozza a *Windows*, *Linux* és *UNIX* rendszerek teljesítményét, valamint az alkalmazások, web, adatbázis, és üzenetküldő szerverek teljesítményét is. A több mint 125 előre elkészített jelentés és az intuitív mérőműszerek lehetőséget biztosítanak a felhasználóknak, hogy egy átfogó képet állapítsanak meg a rendszer egyes teljesítmény problémáiról, és a probléma részleteit is feltárják. A *Longitude*-hoz konfigurálás nélkül kevés tudás is elegendő, 15 percen belül éles környezetben beállítható és futtatható.
➲ *www.heroix.com*

Linux Journal 2005. 137. szám

HTML és CSS – Webszerkesztés stílusosan

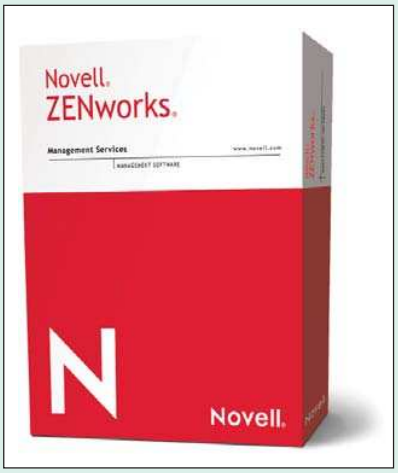
Egy weboldal a tartalom, a szerkezet és a megjelenés hármasából áll össze. A *CSS* megjelenése előtt mindhárom komponenst a *HTML* forrásban kellett elhelyezni. Ennek a viszonylag szerencsétlen állapotnak a megszüntetése hozta életre a *CSS* (rangsorolt stíluslapok) használatát. A *HTML* és a *CSS* fogalma mára már összekapcsolódtak, és együtt alkotják a weblapot. A *HTML* forrásban található a tartalom és a szerkezet, míg a megjelenés, mint összetevő átkerült a *CSS* hatáskörébe – azaz csak részben, mivel a mai napig is inkább olyan weblapok születnek, amelyeknél a *HTML* is tartalmaz a megjelenésre vonatkozó információt.
A *HTML* és *CSS* – *Webszerkesztés stílusosan* című könyv alapvetően a weblapszerkesztésben járatlan olvasóknak készült. Ennek fényében nem egy *CSS* referenciáról van szó, és nem is csak a *CSS* használatát mutatja be, hanem végigvezet az *XHTML* alapú weblapszerkesztés alapjain ötvözve azt a *CSS* alkalmazásával. A hangsúly végig a szabványosságon van: olyan oldalakat tanulhatunk meg létrehozni, amely megfelel a hivatalos ajánlásoknak és előírásoknak. A módszereket a jelenleg elérhető legújabb változatokon az *XHTML 1.0*-án és a *CSS 2*-n keresztül mutatja be.
A szabványos dolgok azonban nem mindig működnek, de a szerző nem csak felhívja erre a figyelmet, hanem megoldást is ad a problémára. A könyv növekményesen dolgozik, ami azt jelenti, hogy mindent az alapoktól kezd, egymás után mutatja be az egyes *XHTML* elemeket, tulajdonságaikat és azok *CSS*-en keresztül történő formázását, célszerű tehát az elejéről olvasni, folyamatosan, s minden szakasz végén szemügyre venni a *CD* mellékleten található példákat, esetleg megpróbálni azokat újra megvalósítani. Rendkívül hasznos, hogy egymás mellett mutatja be a két szabvány használatát, mivel az a tapasztalatom, hogy az összekapcsolódó dolgok könnyebben megragadnak, ha azokat együtt tanulom.
A szerző a *HTML*, *XHTML*, *CSS* fogalmával, szerepével, nyelvtanával vezeti be a könyvet. Az tapasztaltabb olvasó már-már hiányolni kezdi a *CSS* valódi erejének, szerkezetének bemutatását, de a második fejezetben az is megérkezik, mi több az egész szakasz a *CSS* bemutatására lett szánva. Ezek után egy kis weblapszerkezeti elmélet következik, amelyen keresztül megismerjük a weboldalak logikai felépítését. Ez a fejezet már a „párhuzamosság" jegyében épül fel, s ezt a felépítést a továbbiakban minden fejezet követni fogja. A fejezetek első felében mindig az adott feladat megoldásához szükséges *XHTML* elemekkel ismerkedhet meg az olvasó, amelyet a fejezet második felében a hozzá tartozó *CSS* ismeretek követnek.
A negyedik fejezettől kezdődik az egyes *XHTML* összetevők bemutatása, betartva a már emlegetett felépítést. Minden fejezet gazdag képernyőképekben. A bemutatott források után minden esetben ott szerepel, hogy hogyan is fog az kinézni a böngészőablakban – ez megint csak a gyorsabb elsajátítást segíti.
Az *XHTML* elemek bemutatása mellett a szerző nem feledkezett meg az egyéb járulékos ismeretekről sem, mint például a weblapot alkotó fájlok elhelyezéséről, a szükséges könyvtárstruktúra kialakításáról, és az alapvető weblap-elméleti kérdések (például abszolút és relatív hivatkozások közötti különbség) tisztázásáról.
A könyv végén arról is információt kapunk, hogy az elkészült weblapokat hogyan kell közzétenni, internetes kiszolgálókon elhelyezni. Mindezt olyan módon, hogy még a teljesen járatlanok is elboldoguljanak. Az utolsó előtti fejezet az igazi gyakorlati alkalmazásra irányul. Egy publikus webnapló oldalon járva elkészíthetjük saját webnaplónk egyedi kinézetét *CSS* segítségével. Mindez úgy működik, hogy a webnapló megengedi minden felhasználónak, hogy stíluslapot rendeljen a saját kis naplójához, s ennek a stíluslapnak az elkészítésében vezet végig ez a szakasz.
A könyv végén újra az elméleti síkot járjuk: megtanulhatjuk, hogyan helyezzük el weboldalainkon helyesen az egyes összetevőket. Képernyőképeken keresztül mutatja be a szerző, hogy mi szép ill. helyes és mi nem az, mire kell leginkább ügyelni, hogy a honlapunk valóban azt a célt szolgálja, amire terveztük: a figyelem felkeltésére, az információ hatékony átadására.
Virginia DeBolt célkitűzése alapvetően a tanítás volt, amikor megírta ezt a könyvet, a teljesen járatlan webrajongók felkészítése a szabványos oldalak készítésére, úgy is mondhatnám, hogy ők a célcsoport. A jól felépített ismeretanyagon érződik a szerző komoly oktatási tapasztalata. Ez utóbbi tény ugyanakkor hátrány is: ami élvezetes egy kezdő számára, az általában nem csak unalmas, de haszontalan a témában járatos, de a *CSS*-t kevéssé ismerő olvasó számára. Ennek fényében a könyvet inkább a kezdőknek ajánlom, nekik azonban nem is nagyon tudnék ennél jobbat mutatni.

Komáromi Zoltán

Bemutatkozott a Novell ZENworks 7

A Novell Magyarország sajtótájékoztató keretében jelentette be a ZENworks 7-es verziójának megjelenését, amely jelentős újdonságokat tartalmaz az előző változathoz képest. A ZENworks az egyik vezető erőforrás-kezelő eszközkészlet, mely egyaránt támogatja a NetWare, Open Enterprise Server, Linux, és Windows platformokat. A szoftver 2004-ben, és 2005-ben is elnyerte a LinuxWorld „Legjobb rendszerfelügyeleti eszköz” díját.

A jelenleg alkalmazott heterogén vállalati környezetben erőforrás-kezelő, és menedzsment szoftverek nélkül szinte lehetetlen kézben tartani a rendelkezésünkre álló különböző eszközöket, a kiszolgálóktól kezdve egészen a tenyérnyi számítógépekig. Míg ezidáig egy hálózatban többnyire ***Windows*** alapú számítógépek voltak, addig mára jóval elterjedtebb a ***Linux*** operációs rendszer munkaállomásként való alkalmazása. Jogos igényként merül fel tehát, hogy a két operációs rendszert egy központi helyről tudjuk menedzselni. Az informatikai infrastruktúra fenntartása jelentős emberi erőforrást igényel, a térben elszórtan elhelyezkedő eszközök kezelése, nyilvántartása pedig a távoli menedzselés lehetősége nélkül szinte megoldhatatlan. Ezzel párhuzamosan az egyre növekvő biztonsági igények, vírusokkal szembeni védelem, a lehetséges biztonsági rések felderítése is problémát jelent az informatikus munkatársak számára.
Egy az *IDC* által nemrégiben készített jelentés szerint a ***ZENworks*** használatával jelentősen leegyszerűsödnek a rutinszerű felügyeleti feladatok, így az azok elvégzéshez szükséges idő akár 90%-kal is csökkenhet.
Ez azt jelenti, hogy az informatikai infrastruktúra fenntartási költségének jellemzésére manapság használt ***TCO (teljes birtoklás költsége)*** ezzel a szoftverrel felszerelkezve alacsonyan tartható. A kezdeti befektetés kevesebb mint 100 nap alatt térül meg, három év alatt pedig teljes a megtérülés.
A fenti adatok egy 500-1000 főt foglalkoztató vállalatra vonatkoznak, de mivel a ***Novell ZENworks*** moduláris felépítésű, a rendszer egyes elemeit gazdaságosan használhatjuk már 10-20 munkaállomás esetén is. 50-100 munkaállomás esetében ugyanakkor már a teljes rendszer használata ajánlott.
A ***ZENworks 7*** természetesen sok újdonságot tartalmaz elődeihez képest. Ebből számunkra talán a legérdekesebb, hogy a a teljes életciklus-felügyelet immár ***Linux*** rendszerekre is elérhető.
A ***ZENworks Linux Management*** az egyetlen olyan megoldás, amely házirend alapú automatizálást használ a ***Linux*** erőforrások telepítési, felügyeleti és karbantartási feladataihoz.
A modul segítségével lehetővé válik a munkaállomások zárolása, lemezképek készítése, távoli irányítása, leltár, és szoftverfelügyelet ***Linux*** operációs rendszerek alatt, bár egyelőre csak a ***Red Hat*** és ***SUSE Linux*** rendszerek támogatottak.
A ***ZENworks Asset Management*** eszköz-, és készletgazdálkodási lehetőségeket nyújt, mely segíti a licenckezelést, valamint szoftverhasználati és trendelemző képességekkel is rendelkezik.
A ***ZENworks Control Center*** egy web alapú adminisztrációs felületet biztosít, mely a központi webes irányítást tesz lehetővé. Ezen felül az előző verzióhoz képest továbbfejlesztett tenyérgép



támogatást is nyújt az új 7-es verzió.
A ***ZENworks Patch Management*** egy hatékony automatikus hibajavítás-felügyeleti megoldás mely a szükséges hibajavítások és frissítések egyre növekvő számával képes megbirkózni.
A ***ZENworks Handheld Management*** segítségével a ***Palm*** vagy ***Windows CE*** alapú tenyérgépeink konfigurációját van lehetőségünk egy központi helyről vezérelni, illetve biztonsági mentéseket készíthetünk az eszközön található adatokról, melyeket azután egy központi helyen tudunk tárolni.
A ***Novell ZENworks*** használatával elért megtakarításokat pedig mi sem bizonyítja jobban, mint hogy az ezt a rendszert használó ***T-Mobile Magyarország Rt.*** közel 2500 fős csapatát egy mindössze 7 fős help-desk csoport szolgálja ki. ■

SFD2005
Szabad Szoftverek Világnapja Szegeden

Szegedet nemegyszer nevezték már Magyarország második szabad szoftveres fővárosának. Talán nem véletlenül? A Szabad Szoftverek Világnapját (Software Freedom Day) idén is Szegeden ünnepelhettük meg.

Szeptember 25-én, vasárnap a magyarországi szabad szoftveres közösség figyelme ismét *Szegedre* irányult, hisz már másodjára került itt megrendezésre előadássorozat a *Szabad Szoftverek Világnapja* alkalmából. Számos neves előadó tartott előadást az ország különböző részeiből érkezvén, a vendégek között is sokan csak ezért utaztak *Szegedre*. Minden előadó igyekezett úgy megragadni a témáját, hogy az érdekes legyen az egyszerű érdeklődőktől kezdve egészen a vérprofikig mindenkinek. Voltak olyan előadók, akik szakmai értékük mellett személyes kisugárzásukkal is rátettek egy lapáttal az amúgy is közvetlen és derűs hangulatra. Az előadások sokszínűségét jól példázza, hogy szó esett a szabad szoftverek egyéni, közösségi, gazdasági és jogi létjogosultságáról is. Senki sem távozott el üres kézzel. A szervezők *SFD* feliratú pingvines tollal kedveskedtek minden látogatónak, és a szerencsésebbek tombolanyereményeket (póló, bekeretezett *XaoS* fraktálkép, szerverhely bérlet, könyvutalvány) is hazavihettek, amelyeket a támogatók ajánlottak föl. Végül, de nem utolsósorban, a szervezők gondoltak az időközben megéhező vendégekre is. A menü idén open-source palacsinta volt. A bináris változat ugyan népszerűbbnek bizonyult, de a forráskódot is mellékelték a szervezők. A rendezvényt a neten keresztül is lehetett követni a *RádióE* biztosította lehetőségnek köszönhetően. Várható, hogy az előadásokból letölthető videó-összeállítás is készül a – helyszínen kint volt – jópár miniDV kamera anyagából. Az ez iránt érdeklődőknek érdemes időnként felkeresni a rendezvény honlapját. Azt mondják egy egyszeri rendezvény hagyományteremtő. Két rendezvény már hagyomány. Lesz harmadik? Reméljük igen...

Medve Zoltán és Havasi Ferenc

KAPCSOLÓDÓ CÍMEK

RádióE
➲ http://www.radioe.hu/

Szabad Szoftverek Világnapja (főoldal)
➲ http://www.softwarefreedomday.org/

Szegedi SFD oldala
➲ http://www.inf.u-szeged.hu/opensource/sfd/

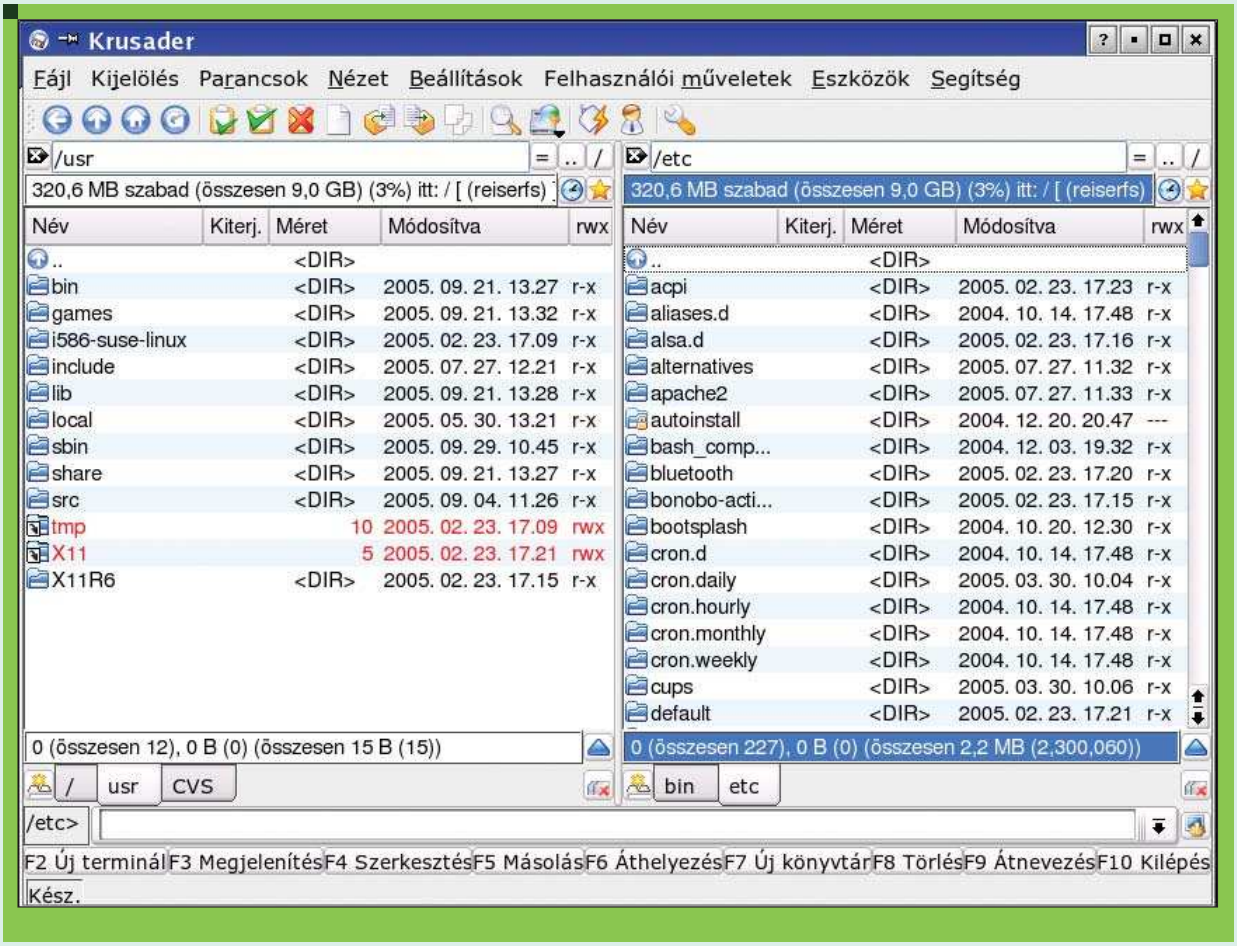
Szegedi Open Source Laboratory
➲ http://www.inf.u-szeged.hu/opensource/

Xaos
➲ http://xaos.sf.net/

A Krusader csapat

avagy hogyan válik az ember linuxos szoftverfejlesztővé.

A Linuxos pályafutásom 2000-ben kezdődött. A munkahelyemen sokszor kellett ***Unix*** rendszereken alkalmazásokat futtatni, így lassan megbarátkozhattam az alapvetően más, kezdetben gyakran barátságtalannak tűnő világukkal. Egy év múlva már az otthoni gépemre is felkerült a ***Linux***. Számomra azért volt fontos az átállás, mert zavart, hogy nem jogtiszta kereskedelmi termékeket használtam és megvásárolni sem akartam őket, mert olyan vállalatokat támogattam volna, amelyek visszaélnek gazdasági erejükkel és hatalmukkal.
A régi programjaim közül legjobban a ***Total Commander*** (akkor még ***Windows Commander***) kétpaneles fájlkezelő hiányzott. Rengeteg szoftvert megnéztem, de egyik sem tudta tökéletesen helyettesíteni. Végül az internetet böngészve eljutottam a ***Krusader*** nevű programhoz és letöltöttem annak az 1.10-es változatát. Ez kinézetben nagyon hasonlított a ***TC***-re, de instabil volt és nagyságrendekkel kevesebbet tudott. Arra gondoltam, hogy a következő verziókban majd javítják a hibákat, de ez ne nem történt meg. Jött az 1.11, ami semmi újat sem hozott, majd az 1.20 is, melyben a fejlesztőknek még azt is sikerült elrontani, ami addig működött. Ekkor meglátogattam a honlapjukat, ahol észrevettem, hogy szoftverfejlesztőt keresnek...
Írtam egy levelet, hogy bár a ***Qt*** programozásához nem értek, a ***KDE***-t nem ismerem és angolul sem tudok tökéletesen (középszint), de azért szívesen besegítenék a hibák javításába. Meglepetésemre azonnal felvettek és írásjogot is kaptam. Magam sem gondoltam volna, hogy néhány hét alatt még a ***Qt*** és a ***KDE*** használatát is sikerül elsajátítanom.
A nyelvtudásom hiányosságai nem jelentettek problémát, mert még a két izraeli szerző (***Shie Erlich*** és ***Rafi Yanai***) sem anyanyelvi szinten beszélte az angolt. Egyedül a fordítóknak okozott néha fennakadást, hogy kitalálják, mit szerettünk volna angolul leírni. Később csatlakozott hozzánk egy venezuelai úr (***Richard Holt***), aki azóta kizárólag a nyelvi hibák javításával foglalkozik.
A ***Krusader*** írása ciklikus folyamat: a fejlesztési szakaszban bővítjük a program tudását, új funkciókat adunk hozzá (mindenki azt, amit akar), majd egy bizonyos idő eltelte után befagyasztjuk a kódolást. Ekkor következik a hibajavítási fázis. Kiadunk egy, vagy több béta változatot (instabil), amit a felhasználók velünk együtt tesztelhetnek, végül, amikor már elég megbízhatónak találjuk a szoftvert, kiadjuk a végső, stabil verziót. Aztán újra kezdődik a fejlesztési szakasz.
Mivel csak hobbiprogramozók vagyunk, így teszteket egyáltalán nem írunk. Nincs rá idő. A programot a felhasználók tesztelik le és jelzik, ha nem megy.
Az egyik legszebb dolog a nyílt forrású fejlesztésben, hogy mindenki azt csinál, amit akar. Kezdetben, amikor beléptem a projektbe, a két szerző inkább csak új dolgokat valósított meg, én a hibákat javítottam, egy német fiatalember (***Dirk Eschler***) a honlapot adminisztrálta, egy másik német társunk (***Heiner Eichmann***) pedig ***FreeBSD***-re portolta a rendszert, vagy besegített a fejlesztésbe.
Később egy belga férfi (***Frank Schoolmeesters***) csatlakozott hozzánk dokumentációt írni. A program addigi dokumentációja hiányos és gyenge volt, de az igazat megvallva egyikünknek sem volt kedve javítani rajta. ***Frank*** fantasztikus munkát végzett. Megtanulta az ***SGML***-t a dokumentálás kedvéért és ezerszer jobban megcsinálta egymaga, mintha nekünk kellett volna kínkeservvel végigszenvedni az egészet. A program ikonjai ugyanígy, külső segítséggel születtek, a fordításokról nem is beszélve.
A ***Krusader*** jelenleg 24 nyelven beszél. Érdekesség, hogy arab nyelvekre nem fordították le. Egy iszlám felhasználó elmesélte, hogy azért, mert a ***Krusader*** az angol ***Crusader***, azaz keresztes lovag szóból származik. A keresztes lovagok pedig nem örvendenek nagy népszerűségnek iszlám földön. A projekt névválasztása ezért nem volt a legsikeresebb. A szerzők kezdetben nem gondoltak arra, hogy ez a szó zavarhat más nemzeteket.
A legfrissebb fejlesztő tagja a csapatnak szintén német származású, ***Jonas Bähr***. Amikor megismertük, egy elképesztő ötlettel állt elő. A ***Krusadert*** tetőtől talpig konfigurálhatóvá szerette volna tenni, hogy tetszés szerint testre szabható legyen és képes legyen felhasználói műveletek definiálására. Az elképzelése senki tetszését sem nyerte meg a csapatból. Ezt azzal indokoltuk, hogy hatalmas munka lenne és feleslegesen túrná fel a stabil, jól működő kódunkat. A ***Total Commander*** fejlesztőjének is megírta ezt az ötletet, ahol hasonló fogadtatásra lelt. Egy hónap derekas, kitartó küzdelem után legvégül sikerült rávennie ***Shie***-t, a projekt vezetőjét, hogy véghezvigye az elképzeléseit. Nagyon jól jártunk vele, hogy megcsinálta. Még újságcikk is született a megoldásáról. Az ő nevéhez fűződik egyébként a ***Mac OS*** portolás is.
Több fejlesztő nem csatlakozott hozzánk, mert egy ekkora projektre már



elegen vagyunk. Meg kell még említenem a szlovén származású ***Matej Urban***-ot, aki szintén nemrég jött és a program marketingjével foglalkozik. Újságcikkeket ír, disztribútorokkal tárgyal, egyszóval reklámoz minket. Nagyon fontos feladatot lát el.

A legjobb ötleteket általában kívülről kaptuk. A ***Total Commanderből***, ***Midnight Commanderből*** sokat merítettünk, a fórumon felhasználók hozzászólásai szintén pótolhatatlanok voltak és máshonnan is érkeztek igazán jó elképzelések.

Egyik ilyen ötlet a ***Firefox*** böngészőben jelent meg először, mely lehetővé tette több lap egyidejű böngészését (***Tabbed Browsing***). ***Shie*** elhatározta, hogy ugyanezt megvalósítjuk a ***Krusaderben*** is. Az ötlet olyan jól bevált, hogy még ***Christian Ghisler***, a ***Total Commander*** szerzője! is átvette, persze jelentős módosításokkal. A ***Ghisler*** megoldás természetesen jobb lett, így nekünk is akadt mit visszavenni belőle.

A példából megérthető, hogy miért káros a szoftverötletek szabadalmaztatása. Kit illetne meg igazából ebben az esetben a szabadalmi jog? A ***Firefoxot***? Lehet, hogy ő is máshonnan vette...

A ***Krusader*** ötleteinek legalább a 90%-át mások találták ki és nem mi. Pont ezért voltunk képesek az elmúlt három év alatt olyan hihetetlen nagy fejlődésre, hogy sok tekintetben utolértük a *TC*-t, ami véleményem szerint egy igazán jó zárt forrású program.

A nyílt forrású fejlesztés sebessége azért rendkívüli, mert felhasználjuk egymás kódját és nem kell állandóan újra feltalálni a kereket. Mondok egy példát: egyik felhasználónk jelezte, hogy jó lenne, ha a ***Krusader*** képes lenne belenézni ***ISO*** fájlokba. Rákerestem a neten, hogy írt-e már valaki hasonlót és meg is találtam egy magyar programozó nevét. Levelet küldtem neki, hogy ha nem bánja felhasználnánk a kódját. Azt válaszolta, hogy tegyük, de említsük meg a nevét a szerzők között. Így jutottunk néhány óra alatt! egy működőképes, letesztelt, új komponenshez. A legnagyobb elismerés számunkra az volt, amikor ***Christian Ghisler*** (a *TC* szerzője), a fórumán a ***Krusadert*** ajánlotta ***Linux*** alá. Másnap hálából felraktuk a honlapunkra, hogy mi ***Windows*** alatt a ***Total Commander*** használatát javasoljuk :-).

A három év programozás alatt egy dolgot alaposan megtanultam: egyedül nem lehet igazán jó szoftvert írni. Rendkívül nagy szükség van társakra: felhasználókra, programozókra, akik észrevételeikkel, ötleteikkel, munkájukkal segítik a projekt fejlődését. Bátorítok mindenkit, hogy merjen csatlakozni linuxos projektekhez, ossza meg a tudását másokkal, így közösségre talál és felejthetetlen élményekben lesz része.

Karai Csaba
(cskarai@freemail.hu)

Informatikus vagyok egy telefontársaságnál, szabadidőmet legszívesebben a jövendőbelimmel töltöm, emellett szeretek táncolni, focizni, görkorizni...

Egyedül nem megy... avagy egy moderátor tanácsai a kérdezőknek

Az ember egyedül rendszerint nem boldogul az életben. Többé-kevésbé társas lények vagyunk, akik problémáikat néha tudatosan, néha tudat alatt hárítják át egymásra. A cél azonban mindkét esetben közös: a problémát meg kell oldani. A kezdő Linux felhasználók az első napokban gyakran esnek abba a hibába, hogy rögtön mindent szeretnének tudni a vadonatúj és merőben szokatlan rendszerükről...

Lássuk be: ez egyszerűen lehetetlen. Egy operációs rendszert, akárcsak egy embert nem lehet egy nap alatt (mi több percek alatt) megismerni. A megismeréshez idő kell. Kinek több, kinek kevesebb. Bár néhányan már a kezdet kezdetén is haladónak tartják magukat, a megfelelő tudás hiánya elég gyorsan kiderül.
Aki azonban kellő alázattal viseltetik a probléma iránt, annak több lehetősége is akad a gyors segélykérésre ha éppen „csapdába esett". Az alapvető eszköz természetesen maga az internet.
Néhány terjesztést (***Mandriva***, ***SuSE***, ***UHU*** hogy csak a nagyobbakat említsem) ha megvásárlunk, a termékhez úgynevezett on-line támogatást is kapunk. Ez azt jelenti, hogy a vásárlástól számítva bizonyos ideig (ez 30 és 90 nap között mozog általában) segítséget kaphatunk a forgalmazótól, a telepítéssel kapcsolatban felmerülő kérdésekben.
Természetesen másképp, teljesen ingyenesen is hozzá lehet jutni a ***Linux*** terjesztésekhez. Le is tölthetjük a telepítőkészletet a hivatalos disztribútor honlapjáról, illetve annak tükreiről (például ➲ ***www.opensuse.org***), de akár újságok mellékleteként is hozzájuthatunk egy-egy ***Linux CD***-hez, vagy ***DVD***-hez.

Ilyenkor persze nem jár a hivatalos segítség, más módon kell megszereznünk az információt felmerülő problémáinak orvoslására.
A sok lehetőség közül a két legfontosabbat (és egyben leggyorsabbat) emelném ki: keresők, illetve fórumok.

Keresők

A legfontosabb dolog a Linux világában a keresők pontos ismerete, használata. Enélkül úgyszólván tehetetlenek vagyunk. A kezdő linuxosok általában abba a hibába esnek, hogy azt képzelik, problémáik teljesen egyediek és önerőből megoldhatatlanok. Nos, ez nincs így. A tapasztalat azt mutatja, hogy csaknem minden kezdő ugyanazokon a gyermekbetegségeken esik át, ugyanazokat a hibákat követi el, amiből az is következik, hogy ezek

megoldása igen nagy valószínűséggel dokumentálva van a világhálón. Csak meg kell találni, hogy hol, a keresőket pedig pontosan erre találták ki.

Egy példa

Szeretnénk kézzel indítani a grafikus felületet parancssor alól a `startx` paranccsal, de a program nem fut le, és többek közt ezt a hibát kapjuk:

```
(II) Primary Device is: PCI
➥01:00:0
(--) Assigning device section
➥with no busID to primary
➥device
(EE) No devices detected.

Fatal server error:
no screens found
```

Ilyenkor mindössze annyi a teendőnk, hogy a teljes hibaüzenetet (nem tévedés) bemásoljuk a keresősávba. A kapott találatok között szinte biztosan ott leljük a megoldást.
Mindig pontos hibaüzenetekre érdemes keresni, így a kapott találatok is minél pontosabbak lesznek. Ha túl hosszú egy hibaüzenet, akkor érdemes az `error` szót tartalmazó sor „környékét" (plusz-mínusz 3-4 sor) bemásolni, és arra keresni.
Szöveget ***Linux*** alatt a következőképp másolunk: egérrel kijelöljük a másolandó részletet (ezáltal már vágólapra

is helyeztük), majd középső egérgombbal beillesztjük a kívánt helyre. Egyes rendszereken a kétgombos egerek a bal és a jobb gomb egyszerre történő lenyomásával emuláljak a középső gombot.
Fontos dolog a keresők használata közben az angol nyelv ismerete is. Legalább a számítástechnikai „konyhanyelvet" el kell sajátítanunk, enélkül ugyanis nagyon nehéz lesz bármit megoldani.
A legjobb keresők, ahol szinte bármely problémára megoldást találunk:

➲ http://google.com
➲ http://google.com/linux

Ha csak angol, illetve más nyelvű találatokat kapunk és nem értjük az ott leírtakat, vagy egyszerűen nem hozott értékelhető eredményt a keresésünk, akkor felkereshetünk egy magyar nyelvű szakirányú fórumot.

Fórumok

Az úgynevezett „internetes fórumok" azért jöttek létre, hogy az emberek egy bizonyos témáhpz kapcsolódóan, illetve kötetlenül beszélgessenek, és önzetlenül, saját szabadidejüket feláldozva segítsenek a kezdőknek, és mindenkinek, aki kérdéssel fordul hozzájuk.
Megkönnyítvén a segítők, illetve a segítségre szorulók dolgát, itt is érdemes betartani néhány íratlan (néha írott) szabályt.
Hozzászólásunkat (angolul: post) a megfelelő gyűjtőtémában (topic) hozzuk létre. Tehát ha például ***SuSE Linux*** alatt nem tudunk filmet lejátszani, akkor azt ne az ***UHU-Linux*** témában közöljük.
Mivel a legtöbb fórum rendelkezik keresővel, ezért a későbbi kereshetőséget elősegítendő a témánk címe legyen kerek, és fedje le a benne leírtakat. Tehát ***„SEGÍTSETEK PLS!!!"*** nem a legmegfelelőbb cím, mert nem derül ki belőle gyakorlatilag semmi azzal kapcsolatban, hogy kinek és mivel volt (van) baja. Ugyanakkor jó cím például az „Mplayer lefagyott".
Lehetőleg ne sürgessük a választ. Amint egy hozzáértő „benéz" a témánkba, választ fogunk kapni. Ha probléma merül fel, előbb a fórum keresőjét, illetve a Google-t használjuk. Szinte biztos, hogy már szerepel a megoldás a régebbi hozzászólások között. Ha mégsem, akkor viszont kérdezzünk bátran.
Fogalmazzunk és írjunk helyesen, egyértelműen. Ez sok időt spórol meg a segítségnyújtók számára, és mi is előbb jutunk a megoldást hozó válaszhoz.
Ezeket szabályokat betartva részesei lehetünk a közösségnek, és minden kérdésünkre érdemleges választ kaphatunk.
A két legelterjedtebb linuxos fórum Magyarországon:

➲ http://linuxforum.hu
➲ http://hup.hu

Apagyi György, (killall)
(user@killall.eof.hu,
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).

Oklevél és Nobel-díj a pingvinekért

Az elmúlt hetekben két jeles, hazai és nemzetközi vonatkozású esemény kívánkozik gazdagítani a pingvin-históriát.

Szeptember 14-én sajtótájékoztató vaku- és kereszttüzében immáron hivatalos okirat tanúsítja: ***Tóbiás*** a ***Linuxvilág*** (egyenetlen ági) leszármazottja.
A magazin megújulásának főszerkesztői tájékoztatója és a ***Budapesti Állatkert*** cukrászdai sóspogácsája között átadásra került az ***„Állatkerti Nevelőszülői oklevél”***. E papirost ***Tarnóczai Odette*** alapítványi titkár nyújtotta át ***Szy Györgynek*** kedves és gondolatébresztő szavakkal. (Abba tán jelenleg ne menjünk bele, mily gondolatokat ébresztett csemeténk férfineve és női neme.) A tapsorkánt a magazin olvasói, írói, szerkesztői, meghívott újságírók, vendégek, a háttérből pedig ***Pongo pygmaeusok*** biztosították. ***Tóbiás*** – bár meghatottsága felől egészen bizonyosak vagyunk – továbbra is tojásain guggolva viselte családra találásának eme megindító pillanatait.

A budapesti ceremóniával szinte egyidőben készülődtek a világ túlsó felén, ***Bostonban*** a 2005-ös év ***IgNobel-díjainak*** átadására. Az idén magyar kutató, ***Gál József*** (karöltve ***Victor Benno Meyer-Rochow*** brémai egyetemi professzorral) vehette át a „kissé kattant” vizsgálódásokért járó megtisztelő díjat nem másért, mint a pingvinek székletürítésekor uralkodó nyomásviszonyok tanulmányozásáért...
Nos, gondolom a díj indoklásához elkelne némi magyarázat...
Történt pedig, hogy valamikor ‘93 környékén ***Victor Benno Meyer-Rochow***, e játszi elmével megáldott kutató az ***Antarktiszon*** szokásos madárleső helyén ülve arra lett figyelmes, mily szaporán suhan röppályáján egy pingvin széklete. Hogy elméje melyik bugyrában (illetve pontosan milyen gyógyszer hiányában) fogant meg az a gondolat, hogy e megfigyelés nyomán nagy dolgokat vigyen végbe, s ebben ne végezzen szerényen titkolt és elszigetelt félmunkát – ez egyelőre a megválaszolatlan kérdések számát gyarapítja.
Mindenesetre állhatatosságának és ***Gál József*** kollégájának köszönhetően az emberiség nem maradt tudatlanságban e madarak bélsárkilövellési nyomásviszonyaival kapcsolatban.
Ahogy tehát Ferenc József mondotta volt egy szoboravatáson: ***hulljon le a pel***. Ime: 1 db pingvin 1 db székletet 1 db székelési folyamat alatt 40 cm-re lövell farától, nem kevesebb, mint 60 kilopaszkál nyomást kifejve mindeközben. Külön figyelemreméltó pontja a kutatásnak, hogy e nyomásadat éppen négyszerese az emberi fajra jellemzőnek, vagyis bár kétségkívül mi jelentjük a teremtés csúcsát, azért fölényünk nem teljesen általános.
Talán akadnak most némelyek, akik azt gondolják, a köznek ügyein munkálkodva nem feltétlen szükséges felfedni az összes rejtélyt. Mások amellett, hogy bizonyítékot, kézzel foghatót kérnek számon kutatóinkon, úgy vélik, a verejtékes munka mégiscsak megérdemli a jutalmat. (Szerény véleményem szerint enyészetnek indult, nem örök életű dolgoknak járna a tisztes és azonnali elhantolás, de azért legyünk büszkék magyar vonatkozásban is az emberi alaposságra...)
Mivelhogy határa nincs az esztelen ész kalandozásainak, e hirtelen fellépő és gyors lefolyású munkásság elnyerte, mit megérdemelt: az idén tizenötödszörre megrendezett, neves koponyák jelenlétével emelt ünnepélyen átadott alternatív Nobel-díjat. Érdekesség, hogy a díjkiosztón olyan ***Nobel-díjasok*** osztogatták az ***IgNobel-díjakat*** – közben papírrepülőkkel dobálózva – mint ***Dudlex Herschbach***, ***William Lipscomb***, ***Sheldon Glashow*** és ***Robert Wilson***.
Az ember mindig tanul... Levonva a tanulságot azon leszek tehát, hogy ***Tóbiást***, ha netalántán a falánkság bűnébe esne, fél méteres távban érjem tetten legközelebbi látogatásomkor...

Halusz Léna

Kávéfőzés lépésről lépésre

(6. rész)

Négy labda, két egérgomb, és egyetlen cél – minél több pontot gyűjteni. Útmutató a világ legidegesítőbb játékának létrehozásához.

A múlt hónapban lerántottuk a leplet a grafikus felületek létrehozásának mikéntjéről. Szó esett arról, hogyan hozhatunk létre egy ablakot, és miként tölthetjük fel vezérlőkkel. A kedves Olvasónak lehetősége nyílt továbbá arra, hogy megismerje az eseménykezelés módszerét. Mindez egy bájosan trükkös alkalmazás példáján keresztül került bemutatásra, amely a grafikus kezelőfelület ellenére gonosz módon épp a felhasználóbarátság tökéletes ellenpéldájának bizonyult.

Be kell, hogy ismerjem, annak ellenére, hogy a *GUI* programozást a felhasználó kényeztetése címén harangoztam be, a most bemutatásra kerülő alkalmazás célja úgyszintén az idegek borzolása lesz. Egy valódi játékot fogunk írni, mely stílusát tekintve egészen a gyökerekig fog visszanyúlni. Azokba az időkbe, amikor egy őrületbe kergető egyszólamú muzsika mellett kellett elhasználni három botkormányt, különben a nagy piros kör megette a kis kék négyzetet.

Nem csalás, és nem is ámítás, valódi játékot készítünk. Ez első hallásra nem tűnhet akkora kihívásnak, viszont gondoljunk bele, mennyi feladatot foglal magába egy ilyen alkalmazás írása. Szükség van animációra, aminek objektumközpontú környezetben egyenes következménye a szálkezelés. Bizonyos fokú fizikát is meg kell valósítanunk annak érdekében, hogy a piros kör élethűen fogyassza el a kék négyzetet. Végül ne feledkezzünk meg a beviteli eszközök kezeléséről sem, ami itt eseménykezelést jelent. Ezek már önmagukban elgondolkodtató problémákat vetnek fel akkor is, ha a játék szabályait még nem is részleteztük. Mi több, nem szóltunk olyan további lehetőségekről, amelyek egyfajta játékélmény elengedhetetlen forrásait jelentik. Ide sorolhatjuk a zenét, a hálózaton keresztüli játékot, vagy a mesterséges intelligenciát. Ezekre első játékunk írása során nem térünk ki, csupán érzékeltetni szeretném azt, hogy játékprogramot írni már csak azért is komoly kihívás, mert szinte elképzelhetetlen olyan terület, amit ne érintene.

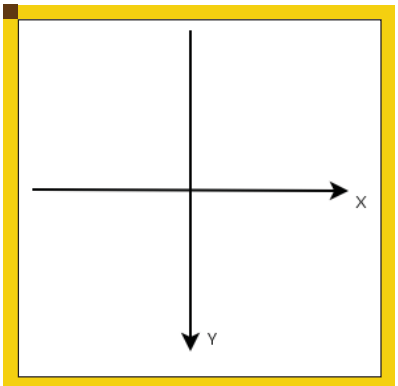
Sok munka áll előttünk, és már most szólok, hogy bármennyi szabadideje és türelme van a kedves Olvasónak, biztos, hogy mindre szüksége lesz. Természetesen megteheti, hogy a gondolkodás terhétől megszabadulva vadul begépeli az itt látható kódokat, és azonnal beleveti magát a játékba. Viszont még ha így is tesz, kérem, hogy ha valamilyen szerencsés véletlen folytán alábbhagyna a játékszenvedélye, tegyen egy kísérletet a program működésének elemzésére. Nem is szaporítom tovább a szót, vágjunk bele!

Mire lesz szükségünk?

A játék igen egyszerű. Egy ketrecben négy pattogó labda van, két piros és két fekete. A játékos úgy szerez pontot, ha sikeresen rákattint valamelyikre. Minden labda egyenlő sebességgel mozog, viszont minél több pontot szerez a játékos, annál gyorsabbak lesznek a labdák. A piros labdákra csak a bal, míg a fekete labdákra csak a jobb egérgombbal történő kattintással lehet pontot elérni. A labdák csak a fallal ütköznek, egymással nem, így ha két azonos színű fedésbe kerül, és sikeres a kattintás, dupla pont jár.

Az alkalmazás 6 osztályból építkezik, ezeket fogjuk egyesével elkészíteni. Minden azonosítónál, legyen az akár osztály-, vagy változónév, az eddig megszokottaknak ellent mondva angol neveket használtam. Azért döntöttem így, mert bizonyos elemek esetén sokkal kényelmesebb a szakirodalomban is elterjedt jelöléseket alkalmazni. Jó példa erre egy személyes tagváltozó beállító, illetve lekérdező függvénye, amelynél a szokásos `set`, illetve `get` nevektől csak körülményes szóhasználattal lehetne eltérni.

Elsődleges célom itt is a következetesség megtartása volt, ezért nem alkalmaztam az egyes körökben elterjedt kevert megoldást, például `getSzin`, `setSzin`. A megjegyzések továbbra is magyar nyelvűek, és a leírás remélhe-



1. ábra A koordinátarendszer

tőleg kellő magyarázattal szolgál azoknak is, akik még kevés jártassággal rendelkeznek az angol nyelvű irodalomban.
Lássuk, melyek azok az osztályok, amiket meg kell írnunk:

- `Ball` (Labda)
- `Cage` (Ketrec)
- `Game` (Játék)
- `Main` (Fő)
- `Score` (Eredmény)
- `Speed` (Sebesség)

A `Main` jelenti az alkalmazás belépési pontját. Ez a legegyszerűbb osztályunk, hiszen egyetlen feladata, hogy létrehozzon egy `Game` objektumot, és elindítsa a játékot. A `Game` építi fel az alkalmazás ablakát, példányosítja a többi osztályt, és lekezeli a felhasználói beavatkozásokat. A `Speed` egy sebességvektort nyújt a `Ball` számára, ami a játék egyik legfontosabb szereplője. A `Cage` a játéktér határait biztosítja. A `Score` pedig a játékos pontszámait tartja nyilván.

Speed

Kezdjük egy kis fizikával. Mint az a játék leírásából kitűnik, a labdák egyenlő sebességgel mozognak.
Ez nagy könnyebbséget jelent. Ennek következtében ugyanis élhetünk egy apróbb egyszerűsítéssel, és ezzel komoly munkát takaríthatunk meg.
A `Speed` szigorú fizikai megközelítésben nem sebességvektort ad, hanem csak irányvektort. A labdák valódi sebességét nem a modell, hanem a megjelenítés szintjén kezeljük.
Ez egy takaros játékprogramban szentségtörésnek minősülne. Már a cikk írásának pillanatában látom az ablakom alatt a dühödt tömeget, és mintha már az akasztófát is elkezdték volna építeni. Viszont ne felejtsük el, hogy legelső játékunkban nem a tökéletesség a cél. Jogos kérdés, hogy az osztály az említettek ellenére miért viseli a `Speed` nevet. Azt szeretném, ha ebből a példaprogramból az Olvasóban egy általános kép maradna meg a kisebb játékok működéséről, és nem konkrét megvalósítási kérdések bosszantanák. Írjunk be magunknak egy rossz pontot ezért a csalásért most, és a következő programban küszöböljük ki.
Térjünk vissza a labdák mozgásához. Az osztályban külön tároljuk az X, illetve Y irányú összetevőt. Mi több, a későbbi számolások megkönnyítése érdekében az összetevőknek a nagyságát, azaz abszolút értékét, illetve az irányát, vagyis az előjelét is külön változókban tároljuk. Ez négy tagváltozót jelent, nevezetesen: `abs_dx`, `dir_dx`, `abs_dy`, `dir_dy`.
A következő példához tekintsük az *1. ábrán* látható koordinátarendszert. Elsőre furcsának tűnhet az Y tengely fordított helyzete, azaz, hogy a Y tengely negatív tartománya az X tengely felett, pozitív tartománya pedig az X tengely alatt helyezkedik el. Számítógépes grafikáknál legtöbbször kényelmes ilyen rendszerben számolni, mivel az esetek nagy részében a megjelenítés ezt várja el tőlünk. Ez a Java esetében sincs másként.
Ebben a rendszerben adott egy sebességvektor: (-3,2). Ez azt jelenti, hogy időegység alatt az objektum 3 egységnyi utat tesz meg balra, és 2 egységnyit lefelé. A `Speed` osztályban bevezetett változókkal ez így írható fel:

```
abs_dx = 3
dir_dx = -1
abs_dy = 2
dir_dy = 1
```

Nevezzük mozdulatnak az időegység alatt történő teljes elmozdulást.
A mozdulatot több körben tesszük meg. Egy körben 1-1 egységnyi lépést teszünk azon összetevők által meghatározott irányokban, amelyek abszolút értéke által előírt mennyiséget még nem értük el korábbi körökben. Visszatérve a példára, a vektor alatt értendő mozdulatot 3 körben tesszük meg:

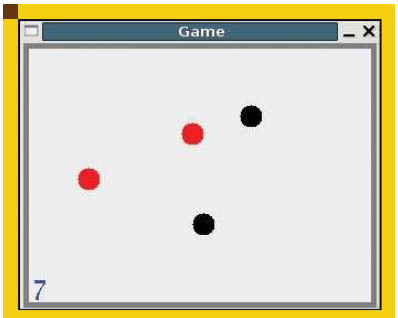
- 1-et lépünk balra, 1-et le
- 1-et lépünk balra, 1-et le
- 1-et lépünk balra

Ezek után újabb mozdulat következhet. Az adott irányba már megtett lépések nyilvántartására bevezetünk két újabb változót: `stepX`, `stepY`. Ezek kezelésére később visszatérünk.
Az idáig vázolt modell nem sérti a sebességvektor definícióját. A csalást ott fogjuk elkövetni, hogy a különböző objektumoknak egyenlő hosszú köröket fogunk osztani. Így a teljes mozdulat nem egységes idő alatt történik. Emiatt függetlenül a sebességvektor abszolútértékétől, ugyanolyan gyors objektumokhoz jutunk. Például egy (1,1) és egy (2,2) vektor ebben a programban (sajnos) egyenlő.
Lássuk az osztály forráskódját (*1. kód*)! A tárgyalt tagváltozók felsorolása után a konstruktort láthatjuk. Ez a paraméterként kapott vektor alapján tölti fel értékekkel a létrejövő objektum tulajdonságait. Mindkét összetevő esetén először az előjelet dönti el, majd az ezzel szorzott összetevőt tekinti abszolút értéknek. Így, ha negatív volt, -1-el szoroz, ha pozitív, 1-el, tehát helyes a leképezés. A `stepX` és `stepY` változók nulla kezdőértéket kapnak.
Azok az objektumok, amelyeket mozgatni szeretnénk a programban, rendelkeznek majd egy `Speed` típusú tulajdonsággal. Mozgatásuk úgy fog történni, hogy előbb lekérdezik, hogy az adott körben az egyes tengelyek mentén lépniük kell-e, majd a lépést követően ezt jelentik.
A `step` metódusból látszik a `stepX` és `stepY` értelme. Ennek a függvénynek a meghívásával jelentheti egy objektum, hogy megtette a szükséges lépéseket. Ha mindkét változó túlszaladt, akkor lenullázza őket, egyébként mindkettőt növeli 1-el. Itt egy újabb csalást érhetünk tetten. A metódus mindkét változót növeli, függetlenül attól, hogy az elérte-e már a hozzá tartozó összetevő abszolút értékét, vagy sem. Ennek a látszólagos hanyagságnak az okára és árára hamarosan rátérünk.
Két lekérdező metódus következik, melyekkel a mozgó objektumok megtudhatják, hogy kell-e lépniük egy adott tengely mentén. Ezek csak akkor

adják vissza a megfelelő összetevőhöz tartozó irányt, ha a vonatkozó `step*` változó nem szaladt túl. Egyébként nullát adnak vissza. Ez a feltétel vizsgálat javítja az előző metódus hanyagságát. Ez időben nem jelent veszteséget, viszont rövidebb és áttekinthetőbb kódot biztosít.
A további négy metódussal az irányváltás oldható meg. Ezek egészen egyszerűen az irányokat jelentő tagváltozók átírását teszik lehetővé.

Ball

Következzen főhősünk, a labda. Egy labdát középpontjának X és Y koordinátája, sugara, színe és sebessége határoz meg. Ennek alapján tekintsük a forráskódot. (*2. kód*)
A tagváltozók nevei kellően beszédesek ahhoz, hogy eltekintsünk a tárgyalásuktól. A konstruktor ezeknek ad kezdőértéket. Látható, hogy létrehozunk egy `Speed` objektumot a sebességvektor jellemzéséhez.
A többi tulajdonság lekérdezhető a megfelelő `get` metódus meghívásával. Ne felejtsük el, hogy egy objektumközpontú környezetben érdemes minden tagváltozót személyesre állítani, és csak azokhoz biztosítani lekérdező, illetve beállító metódust, amelyekhez ez valóban szükséges.
A `move` metódus végzi a labda mozgatását. Röviden összefoglalva az eddigieket, ebben a modellben a labdák mozgatják saját magukat. Viszont a jelenlegi pozíciójukon kívül mást nem tudnak, ezért segítséget vesznek igénybe egy `speed` objektumtól. Amikor valaki mozgásra kényszerít egy labdát, azaz meghívja a `move` metódusát, az módosítja saját pozícióját úgy, ahogy a `speed.getX()` és a `speed.getY()` előírja. Ezután becsületesen jelenti a `speed` objektumnak, hogy a mozgás megtörtént. Így a `speed` nyilván tudja tartani, hogy a labda hányadik körnél tart, és az egyes tengelyek mentén kell-e még lépni.
A `move` ezért először hozzáadja a `speed` által szolgáltatott lépéseket saját koordinátáihoz. Ezután jelent, majd ellenőrzi, hogy történt-e ütközés a ketrec valamelyik falával. Az ellenőrzésnél figyelembe veszi saját középpontját, és sugarát. Ütközés esetén elfordul.



2. ábra A játék

Az utolsó metódus kissé szokatlan lehet, mert egy `Graphics` objektumot kap paraméterül. Ez nem lesz más, mint annak a rajzfelületnek a referenciája, amelyre ki kell rajzolni a labdát. A `paint` feladata tehát megjeleníteni a labdát a kapott felületen. Ezúton újabb megrovásban részesítem magam, mert egy komoly alkalmazásban külön kell választani a vezérlést a megjelenítéstől. Ezt legtöbbször úgy szokták elérni, hogy a megjelenítést végző kód külön osztályban kap helyet.
A működéséhez szükséges információ például egyszerű öröklődéssel nyerhető. Ebben az esetben viszont egyetlen kétsoros metódus miatt nem tettem meg külön osztálynak a labda grafikus változatát.
A *Java API*-ban a `Graphics` osztály tanulmányozásával láthatjuk, hogy milyen rajzolási műveletek állnak rendelkezésünkre egy ilyen objektum esetében. A mi `paint` metódusunkban előbb a `setColor` segítségével beállítjuk a rajzolás színét. Ezután egy kitöltött ellipszist rajzolunk. A `fillOval` metódusnak kissé furcsa a paraméterezése. Azt a téglalapot kell leírnunk, amely magába foglalja a kívánt ellipszist. Meg kell adnunk a téglalap bal felső sarkának X és Y koordinátáját, a szélességét és a magasságát.
Ez egy kör esetében nyilvánvalóan egy négyzet lesz, a paraméterek pedig a fent látható módon adódnak.

Cage

Lássuk most a játékteret határoló ketrec felépítését. Ezt falának vastagsága, színe, valamint szélessége és magassága jellemzi. (*3. kód*)
Az említett tulajdonságok leírását követően a konstruktor látható.
Ez ismét tartogat egy meglepetést, jelen esetben a `Dimension` osztály személyében. Ezzel az egyszerű könyvtári osztállyal egy 2 dimenziós méret fejezhető ki. Ne törődjünk most azzal, honnan fogjuk ezt megkapni a `Cage` példányosításához, pusztán tegyük fel, hogy egy ilyen objektum referenciájával hívjuk meg a konstruktort.
Először a `thick` és `color` tagváltozóknak adunk kezdőértéket. Ezután a `Dimension` típusú objektumnak a megfelelő metódusaival lekérdezzük a szélességet és magasságot, és ezeket átadjuk a `width` és `height` változóknak. A típuskényszerítés azért kötelező, mert ezek a metódusok dupla lebegőpontos értéket adnak vissza, nekünk pedig egész számokra van szükségünk.
A konstruktort az osztály `paint` metódusa követi. Ketrecet legegyszerűbben úgy tudunk rajzolni, hogy előbb a teljes rajzfelületet kitöltjük egy téglalappal, majd ebből kivágunk egy ablakot. A `fillRect` a `setColor` által meghatározott előtérszínnel kitöltött téglalapot rajzol, a `clearRect` pedig a háttérszínt használja. Mindkét metódusnak a bal felső sarok X és Y koordinátáját, valamint a téglalap szélességét és magasságát kell átadni.
Végül négy lekérdező függvény látható, melyekkel a falak belső oldalának megfelelő koordinátái kérdezhetők le. Nyilvánvalóan minden oldal esetében csak egy koordináta értelmes, hiszen például a plafonnak nincs X pozíciója.

Score

Következő osztályunk a `Score`, amely a játékos pontszámának tárolásáért és megjelenítéséért felel. Tulajdonságai a pontszám, a megjelenítéshez használt szín, betűtípus, és a helyének X és Y koordinátája. (*4. kód*)
A tagváltozók felsorolását megszokott módon a konstruktor követi. Ez paraméterként egy X és Y pozíciót, valamint egy színt kap. Ezeket kezdőértékként átadja a megfelelő változóknak, a pontszámot pedig 0-ra állítja. A betűtípus meghatározásához készítünk egy `Font` objektumot. Ennek létrehozásakor megadjuk a betűtípus nevét, stílusát és méretét. A név jelen esetben nem konkrét betűtípusra vonatkozik, hanem egy családot határoz meg, melynek minden virtuális gépet

futtató rendszeren biztosan található példánya. A stílus félkövér, a méret pedig 24 pont.
Az `increment` metódus a pontszámot növeli eggyel, a `getPoints` pedig visszaadja ezt. A `paint` előbb beállítja a színt, majd a betűtípust, végül meghívja a `Graphics drawString` tagfüggvényét, mellyel egy szövegfüzért lehet kirajzolni. Miután ez `String` típusú objektumot vár, az `Integer` osztály segítségével előbb át kell alakítanunk az `int` típusú pontszámot szövegfüzérré.

Game

Eddig a nagy képnek csak különálló részletein dolgoztunk. Hogy hogyan lesz ebből egy nagy műalkotás, már biztosan foglalkoztatja az Olvasót. Elérkezett az idő, hogy elkezdjük használni eddigi munkánk eredményeit, és lássuk, ahogy a dolgok összeállnak. Következzen a `Game.java` (*5. kód*).
A sorozat előző részében létrehozott grafikus felülethez használt osztályok közül több visszaköszön itt is. Viszont első ránézésre sok az újdonság. Az elemzést kezdjük a konstruktor vizsgálatával. Az első sorban az ismert módon létrehozunk egy ablakot, *„Game”* címmel. A második sorban állítjuk be az ablak elrendezéskezelőjét. A `BorderLayout` egy olyan elrendezéskezelő, ami öt részre vágja az ablakot. Van egy nagy középső rész, és négy kisebb a négy égtájnak megfelelően. Mi csak a középső részt használjuk, így az elhelyezésre kerülő egyetlen vezérlő kitölti az ablakot.
Ez a vezérlő a `Canvas` lesz, ami nem több, mint egy rajzvászon. Ezzel nem becsmérelni akartam a képességeit, csupán azt akartam jelezni, hogy egy nagyon alacsony szintű ***AWT*** elemről van szó. Gyakori, hogy nem is használják közvetlenül, hanem egy saját osztály terjeszti ki. Viszont arra a feladatra, amire nekünk kell, elegendő lesz önmagában is. A `Canvas` példányosítása után beállítjuk a méretét. Itt látható, hogy egy új `Dimension` objektumot adunk át a `setSize` metódusnak, így határozzuk meg a vászon szélességét és magasságát.
A következő sorban az ablak tartalmához hozzáadjuk a vásznat, és jelezzük, hogy középen szeretnénk elhelyezni. Ezután életre keltjük az elrendezéskezelőt, beállítjuk, hogy az ablakot ne lehessen átméretezni, továbbá, hogy az ablakkezelőn keresztül kezdeményezett bezárás hatására lépjen ki a program.
A következő sorban meghatározzuk, hogy dupla-pufferelt üzemmódban szeretnénk használni a rajzvásznat. A dupla-pufferelés azt jelenti, hogy két lapunk van. Egy amit a felhasználó lát, és egy, amin mi dolgozhatunk. A kettőt egy művelettel kicserélhetjük. Így a sok időt emésztő rajzolást a háttérben végezhetjük, és csak akkor mutatjuk meg a felhasználónak a képet, ha az elkészült. Ha nem használnánk ezt a módszert, az animációnk biztosan villogna, ami nem túl szép.
A pufferkezelő stratégia objektumával tudjuk a háttérlaphoz tartozó `Graphics` objektumot lekérdezni, illetve a cserét végrehajtani. Ezért ennek a referenciáját a következő sorban lekérdezzük, és eltároljuk egy személyes tagváltozóban. Fontos, hogy a dupla-puffer létrehozása, és a stratégia lekérdezése csak azután történhet, hogy a vásznat hozzárendeltük egy ablakhoz, és az elrendezéskezelő is befejezte tevékenységét, és így a vászonnak biztosan van valódi mérete. Ezen megfontolások miatt az utóbbi két sor nem előzheti meg a `frame.pack()` hívást!
Az ezt követő részben hozzuk létre azoknak a hozzávalóknak az objektumait, amiken idáig dolgoztunk. Már itt érezhetjük, milyen boldogító érzés saját osztályainkat felhasználni. Egyetlen jól irányzott sorral 4 helyett 5 labdánk lehetne a játékban, mi több, semmi sem állíthat meg bennünket még több és több labda létrehozásában.
A `gameSpeed` tagváltozó a labdák sebességében fog szerepet játszani.
A konstruktor utolsó két sorában az egérrel kapcsolatos események lekezelését bízzuk a létrejövő `Game` objektumra, és láthatóvá tesszük az ablakot. Ezek alapján látható az osztály egyes tagváltozóinak szerepe. Egyedül a `gameSpeed` tulajdonság nem tisztázott, ám erre is hamarosan fény derül. „Jó, de hogyan mozog a labda?” – kérdezheti teljes joggal az Olvasó. Végtére is minden adott, már csak az isteni szikra hiányzik a gépezet beindításához. A labdák mozgása lényegében egy végtelen

ciklus eredménye. Ez a ciklus nem csinál mást, csak frissíti a labdák helyzetét, azaz megmozdítja őket, majd újrarajzolja a teljes vásznat. Majd megint mozdít, újrarajzol. Egy ilyen végtelen ciklust viszont nem tehetünk közvetlenül a programunkba.
Ez a lépés azt eredményezné, hogy teljesen elveszítjük a vezérlést a programunk felett. Egy alkalmazás ugyanis számtalan eseményre kell, hogy reagáljon. Ezek egy részét tudjuk csak programozni ***Java***-ban, például az egérkezelést. A többség jelzések kezeléséből áll, amelyeket ilyen magas szinten nem is tudunk befolyásolni. Egy végtelen ciklus megölné a programot, nem tudna ellátni számos adminisztratív teendőt.
A megoldás egy új szál bevezetése a programba. A szálat szokás könnyűsúlyú folyamatnak is hívni, mert igen hasonlóan kezeli egy program a szálait ahhoz, ahogy az operációs rendszer kezeli a folyamatokat. Ezeket mellesleg az előbbivel szemben szokás nehézsúlyú folyamatoknak is nevezni. Ha tehát egy önálló szálra bíznánk ezt

a végtelen ciklust, azzal megoldanánk a problémát. Az egyetlen kérdés ezek után, hogy mennyire nehéz feladat a szálkezelés *Java*-ban.
A válasz természetesen az, hogy mint minden, ez is roppant egyszerű. Minden olyan objektum lehet új szál egy programon belül, ami egy `Runnable` interfészt megvalósító osztály példánya. Az interfész megvalósításán túl lehetőség van a `Thread` osztály kiterjesztésére is. Ennek részleteiért lásd a *Java API*-ban a `Thread` leírását.
A `Runnable` interfész a `run` metódust megvalósítását írja elő. Ennek a metódusnak a tartalma jelenti a szál kódját. A `Game` osztályban ez egy azonnal szembetűnő végtelen ciklus, pont az, amire szükségünk van. A ciklus hasa három lépést tartalmaz. Először annyi `update` hívást végez, amennyi a `gameSpeed`. Másodszor meghívja a `repaint` függvényt. Végül elalszik 10 millimásodpercre.
Az első lépésben érvényesítjük azt a csalást, amiről a sebességvektor tárgyalásánál ejtettem szót. Úgy gyorsítjuk a labdákat, hogy kevesebbszer rajzolunk, más szóval, több számolásra jut csak egy rajzolás. A `repaint` utáni alvás azért kell, mert még szálkezeléssel együtt is 100%-os processzorkihasználtságot eredményezne a végtelen ciklusunk, ha nm függesztené fel bizonyos időközönként a futását. A `try - catch` pedig azért kell, mert a `Thread.sleep`, ami a jelenlegi szálat altatja el, `InterruptedException` kivételt dob. Ezt most üres `catch` ággal kezeljük, de ez normális esetben nem gond.
A kellően kielemezett `run` után következik annak két segítő függvénye, az `update` és a `repaint`. Az `update` feladata a labdák léptetése. Egy ciklusban végigmegy a labdák tömbjének elemein, és egyesével meghívja a `move` metódust. Ennek átadja a ketrec referenciáját, így a labda le tudja kérdezni a ketrectől, hogy hol vannak a szélei, és így képes ellenőrizni az ütközéseket.
A `repaint` újrarajzolja a teljes pályát. Ehhez előbb lekérdezi a vászon pufferkezelő stratégiájától a háttérlap `Graphics` objektumát. Ezután ezt átadja a ketrec rajzoló metódusának, majd az összes labda rajzoló metódusának, végül a pontszám rajzoló metódusának. Végezetül kicseréli a két puffert, és így előtérbe hozza azt a lapot, amire idáig a háttérben rajzoltunk.

Az osztály utolsó metódusa az egérkattintás eseményét kezelő `mouseClicked`. Ez két részből áll. Egyrészt ellenőrzi, hogy megfelelő kattintás történt-e labdán, majd a játékos pontszáma alapján állítja a játék sebességét.
Az első részben egy ciklussal végigmegy az összes labdán. Mindegyiknek lekérdezi a középpontját és a sugarát, majd a kör egyenletét felhasználva kiszámolja, hogy a kattintás a kör területén belül történt-e. Ez esetben még azt is megnézi, hogy a megfelelő gombbal kattintott-e a játékos, és ha minden rendben, odaítél egy pontot.
A második részben a pontszám növekedésével arányban gyorsítja a játékot.

Main

Végül, de nem utolsó sorban futtathatóvá tesszük fáradozásaink eredményét egy burkoló osztály segítségével. Íme:

```
/**
* Az osztaly az alkalmazast
* inditja.
*/
public class Main {

   /**
   * A jatek.
   */
   private Game game;

   /**
   * Letrehozza a jatek
   * objektumat
   * es elinditja a szalat.
   */
   public Main() {
      game = new Game();
      Thread t = new
      ➥Thread(game);
      t.start();
   }

   /**
   * A jatek inditasa.
   */
   public static void
   ➥main(String[] args) {
      new Main();
   }

}
```

A belépési pontot jelentő `main` tagfüggvény egy `Main` típusú objektumot hoz létre. Ennek a konstruktora így létrehoz egy `Game` típusú objektumot. Ezután a `game` referenciával létrehoz egy új szálat. A `Thread` ezen konstruktora egy `Runnable` interfészt megvalósító objektumot vár. Végül elindítja a szálat a `start` metódussal.

Hogyan terjesszük?

Ha a képen látható játékélményt mással is szeretnénk megosztani, kényelmetlen lehet a 6 `.class` fájlt kezelni. Szerencsére a `jar` parancs segítségével saját csomagot hozhatunk létre. Így elég egyetlen állománnyal bíbelődnie annak, aki a programot futtatni szeretné. Ám ehhez nem elég csupán a `.class` fájlokat összecsomagolni, azt is meg kell mondani, hogy melyik osztály tartalmazza a belépési pontot. Ezt egy úgynevezett `manifest` állománnyal írhatjuk le. Készítsünk egy `Manifest.txt` fájlt az alábbi tartalommal:

```
Main-Class: Main
```

Itt a kettőspont után álló `Main` vonatkozik arra, hogy a `Main` osztály tartalmazza a belépési pontot. Ezután adjuk ki a következő parancsot abból a könyvtárból, ahova idáig dolgoztunk:

```
$ jar cmf Manifest.txt
Balls.jar *.class
```

Ezzel el is készítettük a csomagot. Elég a `Balls.jar` fájlt elküldeni ismerőseinknek. A programot ezek után az alábbi paranccsal lehet elindítani:

```
$ java -jar Balls.jar
```

Jó játékot!

Fülöp Balázs
(bigwig42@gmail.com)
21 éves, imádja a Túró Rudit, a Debian Linuxot és a teheneket. Kedvenc írója Slawomir Mrožek. Leginkább a számítógépes hálózatok biztonsága érdekli. A BME VIK műszaki informatikus szak hallgatója.

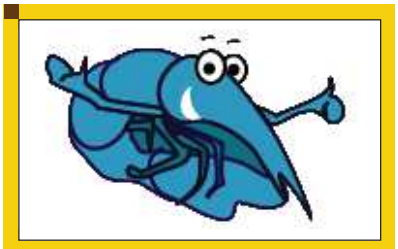
KAPCSOLÓDÓ KÓDOK

A kódok a Linuxvilág honlapjáról tölthetők le.

Gambas – Visual Basic Linux alatt (1. rész)

A 90-es évek elején jelent meg a Visual Basic mint szkriptnyelv. Bár a „szkript” kifejezés jelentésén bőven túltett, hiszen alkalmas volt rendszerprogramok írására is. Kényelmes de még bőven hatékony eszközzé vált. Felbukkanása után nagy népszerűségre tett szert, mivel nagyon könnyen készíthettünk vele grafikus felületű alkalmazásokat. A mai napig az egyik legnépszerűbb negyedik generációs programnyelv, komolyodott, bővült a palettája. Jó tulajdonságai miatt, a nyelv Linux alatti megvalósítására volt már több próbálkozás...

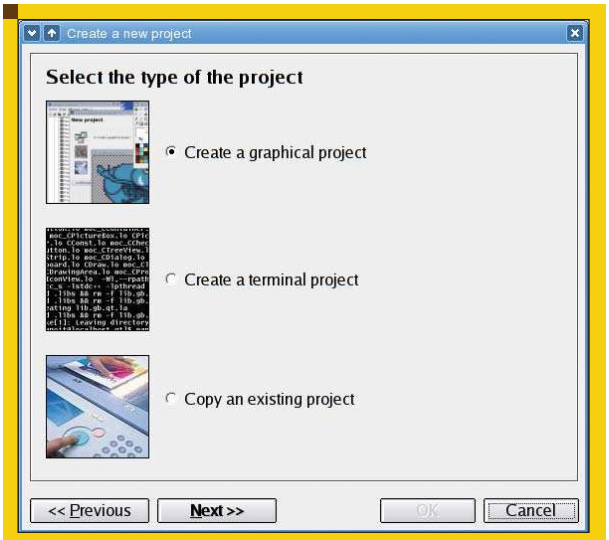
Az egyik ilyen megvalósítás a *Gambas*, egy egyszerű objektum orientált fejlesztőkörnyezet, mely egy *BASIC* interpreteren alapszik. Az alkotás egyáltalán nem kompatibilis a *Visual Basic*-kel sőt soha nem is szeretne az lenni, viszont jó tulajdonságait mind egy szálig hordozza a szkriptnyelvnek. Az, hogy a kompatibilitás szempontját tudatosan figyelmen kívül helyezték, a projekt sikerességét jelentősen segítette. Az eddigi próbálkozások a *Visual Basic*-nek a *Linux* alatti „honosítására”, nagyjából mindig a kompatibilitás megtartása miatt vesztek el.



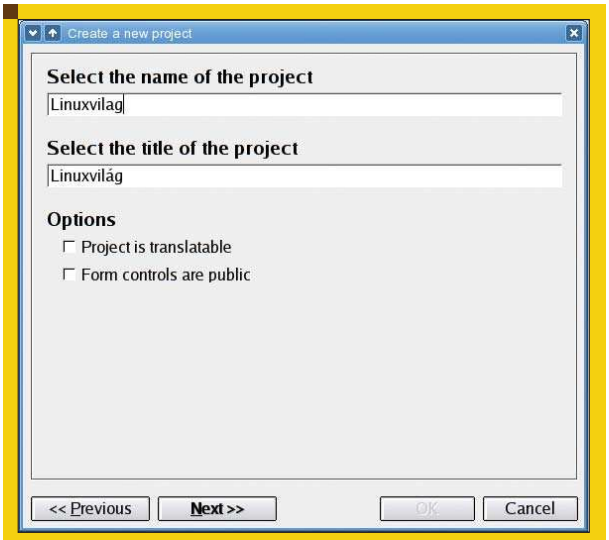
1. ábra Az egyik próbálkozás: A Gambas

2. ábra A fogadtatás...

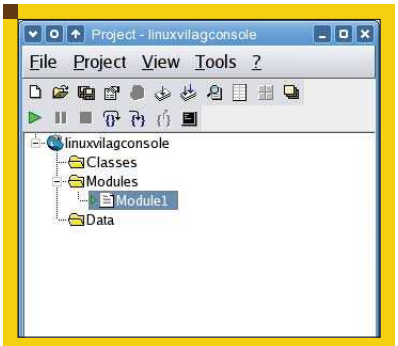
A *Gambas* filozófiája: *„Könnyen, gyorsan, hatékonyan”*. Mindenek előtt az eszköz egy *BASIC* nyelv objektum kiterjesztésekkel. Egy program több fájlból áll, minden fájl egy osztályt ír le, az objektum orientált programozás elveinek megfelelően. Ezek az osztályok fordulnak le és indulnak el a fordító által. Ebből a szemszögből egy kicsit hasonlít a *Java* nyelvre. A *Gambas* több részből tevődik össze: fordító, interpreter, archiváló, *GUI*, fejlesztő környezet. Egy projektet egy könyvtárban tárol. A fordító ebből a könyvtárról készít egyetlen egy futtatható állományt vagy akár installációs csomagot. Telepítése nem igényel különösebb beállításokat. A csomag függőségei a mai disztribúciók többségében megtalálható. Fordítás előtt azért ellenőrizzük az alábbi csomagok jelenlétét: *X11*, *KDE 3.x*, *Qt 3.x*, *PostreSQL*, *MySQL*. A forráscsomag a projekt honlapjáról (➲*http://gambas.sourceforge.net*) szerezhető be. Nagyon sok *Linux* terjesztés alatt probléma nélkül fut. A honlapon egy táblázatban megtalálhatók az egyes instrukciók, ha esetleg valamely *Linux* fajta alatt makacskodna a program. 64 bites *Linux* támogatottsága még nem teljesen alakult ki, *FreeBSD* alatt viszont gond nélkül fut, valamint *Solaris* alatti megjelenésére sem kell már sokat várni. Sikeres fordítás, telepítés után, máris letámadhatjuk a fejlesztő környezet képességeit. Lássunk hogyan is működik az újdonsült *IDE*. Indítás után – a modern fejlesztő környezetektől elvárt módon – kényelmesen választhatunk a különböző, projektekkel kapcsolatos tevékenység közül. Itt találhatjuk a legutóbb megnyitott projektek listáját is. Munkánkat végigkíséri *„Gambas”* a kék homár, akit itt ott megjelenik a képernyőn és különböző tevékenységeinket (fordítás, futtatás) más és más animációval kíséri. Természetesen *Gambast* át lehet lehet helyezni bárhová és ki is lehet kapcsolni ha már nagyon akadályoz minket a programfejlesztésben. Új projekt választása esetén egy varázsló segít minket a készítendő alkalmazás típusától függően a beállításokban. Készíthetünk grafikus felületű és konzolos alkalmazásokat egyaránt, vagy lemásolhatunk egy már meglévő projektet, majd szerkeszthetjük tovább.
A *Gambas* felülete egyébként szintén *Gambasban* készült, csak hogy lássuk



3. ábra A projekt típusának kiválasztása

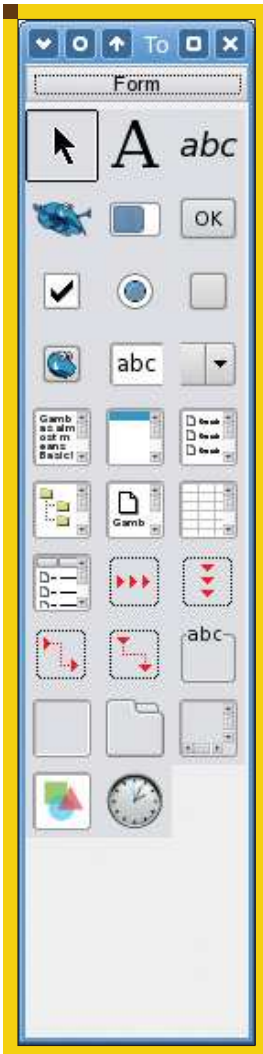


4. ábra A projekt adatainak megadása

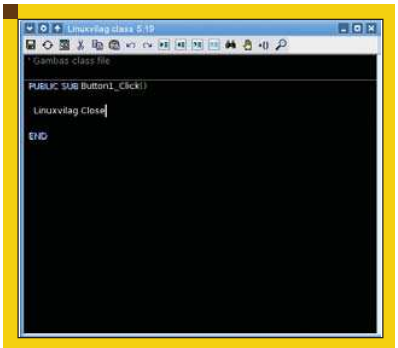


5. ábra A Projekt ablak

6. ábra Telepítőcsomag készítése

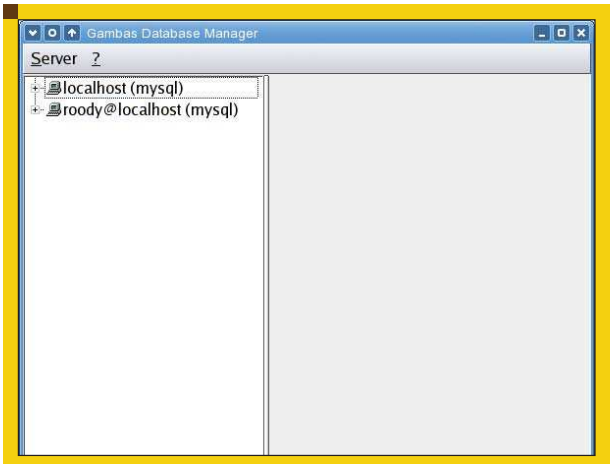


8. ábra Az eszközök listája

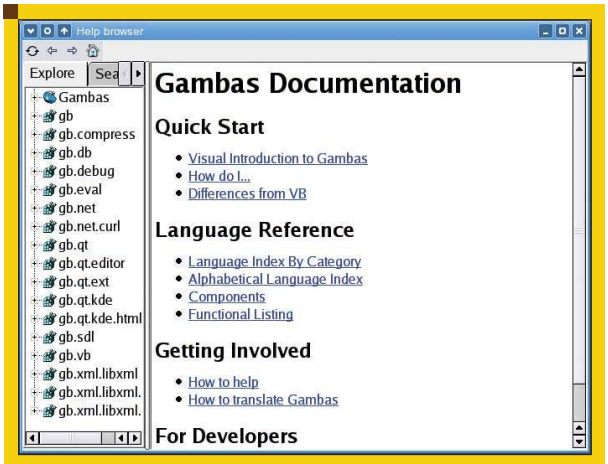


7. ábra Kódablak

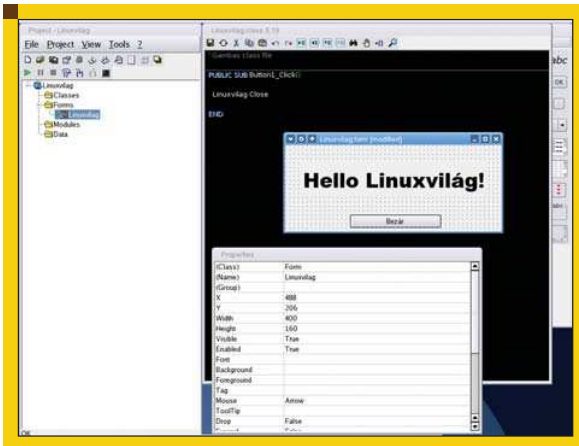
mire is képes ez az egyszerű kis fejlesztő környezet. Miután sikeresen beállítottuk projektünk paramétereit, egy sokak számára bizonyára ismerős környezet tárul elénk.
Baloldalon a ***Projekt*** ablak, jobb oldalon a komponensek listája. A grafikus elemek hozzáadása után középen, lent jelenik meg a komponensek tulajdonságait tartalmazó ablak. Természetesen úgy rendezzük a felületet ahogyan csak szeretnénk, de a fejlesztő környezetek közötti „konvenció” a fenti elrendezést preferálja. Ami újat jelenthet és érdekességnek számíthat azt a Projekt ablak rejti. A legalapvetőbb – a mai fejlesztő környezetekből nélkülözhetetlen – dolgokon kívül, mint a debug, fordítás, összes fordítása, hierarchia táblázat, pár hasznos dolog még helyet kapott a Projekt ablakban. A ***Projekt*** menüben lelhető fel a ***„Make Installation Package”*** opció, ami egy varázslón keresztül egy egyszerű installációs csomagot készít alkalmazásunknak. Ezen csomagok fajtáit disztribúciókhoz szabhatjuk. Említésre méltó még a ***„Make source archive”*** menüpont, ami programunk forráskódjából készít csomagot. Hasznos lehet a fejlesztések során, ha egy



9. ábra Az adatbázisok kezelője



10. ábra A súgó



11. ábra A Gambas bevetés közben

12. ábra Egy konzolos alkalmazás

egy állapotot el szeretnénk menteni a projektből. A kódablak is a szokásos módon áll kézre. Az automatikus kódkiegészítést és az elengedhetetlen szintaxis kiemelést is támogatja. Ha egy egy komponensre duplán kattintunk máris előbukkan a hozzátartozó kódrészlet és írhatjuk is az eseménykezelőt.

A eszközök listája minden alapvető kis dolgot tartalmaz ami elengedhetetlen egy grafikus felületű alkalmazás készítéséhez, kiegészítve a komponensek pontos elrendezését biztosító boxokkal. Természetesen helyet kapott az adatbázisok, mint a ***PostgreSQL*** és a ***MySQL*** támogatása is. Ezek kezelését a ***Projekt*** ablak ***„Tools”*** menüjében lévő ***„Database Manager”*** könnyíti meg számunkra.

A munkánkat egy nagyszerű súgó segíti, ami még sajnos csak félkész állapotban létezik, de már így is nagyon sok gyötrődéstől szabadíthat meg minket. Az egyes komponensek nagyon részletesen le vannak írva és szinte mindegyikhez mellékeltek példákat a használatra. A objektumok, komponensek szokásos fastruktúrában vannak elhelyezve, de természetesen egyszerűen rákereshetünk kulcsszavakra is.

Mindent összevetve a ***Gambas*** egyszerűsége ellenére, igen komoly alkalmazások fejlesztésére ad lehetőséget. Ötletes, egyszerű eszközökkel végre ***Linux*** alatt is élvezhetjük a ***Visual Basic*** nyújtotta egyszerűséget és praktikát.

A fejlesztése folyik, tehát várhatók még újdonságok.

Kis hátránya – amit a jövőben valószínűleg orvosolnak – hogy kevés a felhasználható komponens. Főleg az adatbázissal kapcsolatos eszközök listája kevés, de már így is szépen lekezelhetők a különböző adatbázisok. Ha nem is profi programokat, de azért komoly segédprogramokat könnyen elkészíthetünk vele, melyek megkönnyíthetik a mindennapos munkavégzést ***Linuxos*** gépünkön, ***„Könnyen, gyorsan, hatékonyan.”***.

A cikk folytatásában egy konkrét projektet mutatok majd be: egy egyszerű képnézegető alkalmazást fogunk készíteni a ***Gambas*** segítségével, amely során részletesebben is megismerkedünk az egyes komponensek főbb tulajdonságaival.

Radics Péter
(peter.radics@gmail.com)
Az ELTE-n tanulok programtervező matematikus szakon. Hobbim a kosárlabda, autóvezetés, web-design, programozás. Főleg webes alkalmazások fejlesztése érdekel. 4 éve megrögzött Linux felhasználó vagyok.

Adatbázis-fejlesztés könnyedén, a Rekall segítségével

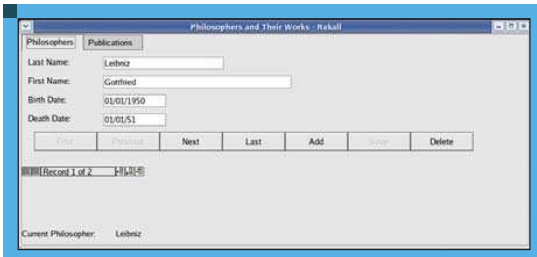
A Rekall segítségével rövid idő alatt fejleszthetünk ki Linux vagy Microsoft Windows alatt futó adatbázis-alkalmazásokat. Nézzük, következő munkánk során hogyan vehetjük hasznát ennek az ingyenes fejlesztőeszköznek.

A többféle géptípuson vagy operációs rendszeren is futtatható alkalmazások témaköre a következő néhány évben minden bizonnyal egyre nagyobb szerephez jut majd, ahogy a vállalatok – különösen a kis- és középméretűek – egyre nagyobb hányada ismeri fel a *Linuxra* való áttérés előnyeit. Ezek a vállalatok várhatóan fokozatosan, egyszerre mindig csak néhány géppel fogják végrehajtani a váltást. Ha a piac vertikális lefedésére törekedve rövid idő alatt tudunk az ügyfél által használt alaprendszerek mindegyikén futó alkalmazásokat fejleszteni, akkor tanácsadóként fontos előnyre tehetünk szert.
A *Rekall* és a *PostgreSQL* párosa itt és most biztosítja számunkra ezt a lehetőséget.
Tavaly a főbérlőm munkahelyéről felkértek, hogy segítsek lecserélni egy régi, Microsoft Access alapú alkalmazást.
A hasonló megbízásokat korábban különféle eszközök segítségével teljesítettem, például *Glade*, *C++* és *Sybase ASE*, később pedig *Apache*, *PHP* és *PostgreSQL* alkalmazásával.
Ez alkalommal többféle feltételt is teljesítenem kellett: a jelentéseket előállító programnak *Microsoft Windows* és *Linux* alatt egyaránt működnie kellett, illetve rendelkeznie kellett helyi „vastag” ügyféllel.
A háttérben futó adatbázis kiválasztásával nem sokat vacakoltam, a *PostgreSQL* mellett döntöttem. Az, hogy a *PostgreSQL 8*-as változata *Linux* és *Windows* alá egyaránt elérhető, nemcsak a szabad programok erejének nagyságát bizonyítja, de egyben kiváló alapot szolgáltat robusztus, többféle operációs rendszeren vagy géptípuson is futtatható alkalmazások készítéséhez. Megfelelő felülettel teljesen mindegy, hogy a kiszolgáló *Linux* vagy *Windows* alapú, vagy a munkaállomások mekkora hányadán fut *Linux* vagy éppen *Windows*. Mindez főként azoknak a vállalatoknak fontos, amelyek alkalmazottaik egy részénél vagy mindegyikénél az asztali munkaállomások *Linuxra* való átállítását tervezik.
Felületként olyan fejlesztői környezetet akartam, amellyel gyorsan meg lehet tervezni az űrlapokat, a jelentéseket és azokat az adatbázisokat, amelyekhez kapcsolódnak. A többféle környezet támogatása alapkövetelmény volt, ugyanis a megrendelő az alkalmazottak egy részének számítógépét *Linuxra* akarta átállítani, míg a zárt, csak *Windowsra* elérhető alkalmazásokat futtató munkatársak továbbra is maradtak volna a *Windowsnál*.
Miközben keresgéltem, számos a *Microsoft Accesshez* hasonlítható, csak annál jobbnak nyilvánított alkalmazással találkoztam. Az eszközök mindegyike többféle adatbázis elérésére is képes volt, valamint támogatták az *ODBC* forrásokat is. Mindegyikük biztosított parancsfájlkészítési lehetőséget is, *BASIC*, *JavaScript* vagy *Python* nyelven. Néhány ezek közül az eszközök közül: *Kexi*, *OpenOffice.org Base*, *Kylix*, *Knoda*, *Rekall* és *Glom*.
A széles választék ellenére csak két eszközt nyilvánítottak termelésre késznek, a *Kylixot* és a *Rekallt*, így részletesebben csak ezekkel foglalkoztam. Miután rájöttem, hogy a *Kylix* önmagában nem támogatja a jelentések előállítását, egyedül a *Rekall* maradt.

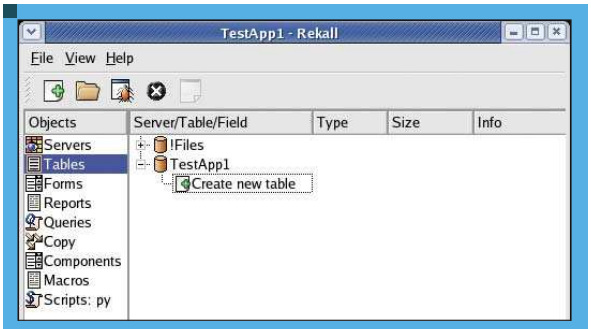
Ismerkedés a Rekallal

A *Rekallt* egy brit cég, a *Series One Consulting* fejleszti. *Mike Richardson*, a *Series One* vezető tanácsadója 2001-ben kezdte írni a *Rekallt* – egész egyszerűen zavarta, hogy nincsenek adatbázis-fejlesztői eszközök *Linux* alá.

Mike-hoz később csatlakozott *John Dean* is, aki a *Windows* és a *Macintosh* alatti fejlesztésért vállalta a felelősséget, illetve megírta az *Oracle* és a *DB2* illesztőprogramokat.
A *Rekallt* eleinte a *TheKompany* terjesztette, mint keresztplatformos, kereskedelmi fejlesztőeszközeinek egyikét. 2003 végén a terjesztési szerződés lejárt, és a *Series One* úgy határozott, a továbbiakban maga terjeszti a *Rekallt*. Aki tehát a *Rekall* megismerése mellett dönt, az a *TotalRekall* vagy a *RekallRevealed* webhelyről töltse le, ezek ugyanis a tevékenyen támogatott változatok.
A *Rekall* kétféle, *GPL* és zárt szerződés hatálya alatt érhető el. Ennek oka az, hogy *Qt* alapú, és amikor első változatai megjelentek, a *TrollTech* még nem *GPL* hatálya alatt kínálta a *Qt* windowsos változatát. A *Rekall* linuxos változatát tehát forráskód formájában szabadon letölthetjük a *RekallRevealed* webhelyről, de dönthetünk úgy is, hogy fizetünk 25 fontot (körülbelül 9200 forintot), ekkor a linuxos, a windowsos és a *Macintosh* alá készült futtatható változatot egyaránt megkapjuk. A *Series One ODBC*, *Oracle* és *DB2* adatbázis-illesztőprogramokat és futási idejű csomagokat is kínál – felár ellenében.
A *Rekall* következő kiadása a 2.4-es lesz, várhatóan több különböző változatban lesz elérhető. A *GPL*-es változat mindig forrás formájában lesz beszerezhető. A *Series One* úgy tervezi, hogy egy *Professional* változatot is megjelentet, mely két további szolgáltatást lesz képes biztosítani:



■ *1. ábra* A kész filozófusnyilvántartó program



■ *2. ábra* A tervezet főablakáról lefelé lépkedve juthatunk el a táblák összeállításáig és az űrlapok és a jelentések létrehozásáig

Titkosítás – Segítségével úgy terjeszthetjük alkalmazásainkat, hogy nem kell forráskódjuk lemásolásától tartanunk. A *Rekall Professional* változata által kínált titkosításnak köszönhetően alkalmazásainkat az ügyfelek számára személyre szabott védelemmel adhatjuk majd át.

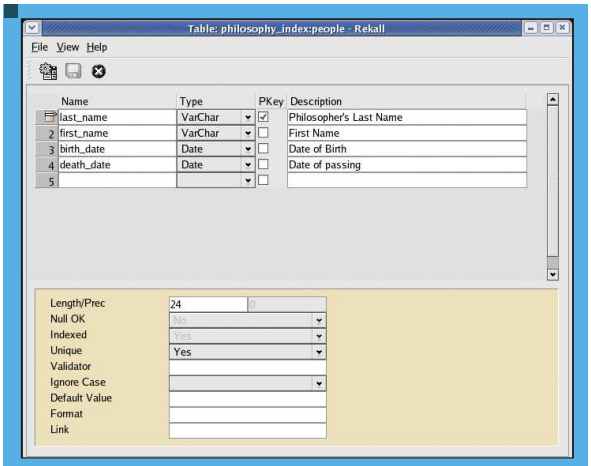
Webalkalmazások készítése – Segítségével bármilyen *Rekall* alkalmazásból *LAMP* alapú webalkalmazást készíthetünk – lásd a *RekallRevealed* webhelyét.

A Rekall hátrányai

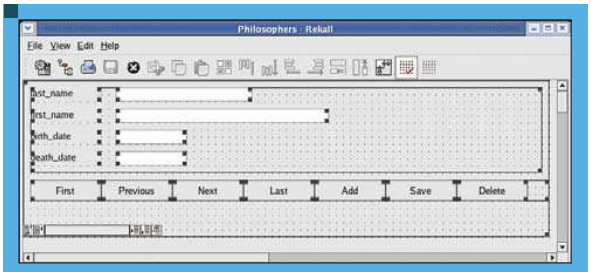
Használata során felfedeztem a *Rekall* néhány hiányosságát is. Először is, nem biztosít egyszerű eljárást menüsorok létrehozására, másodszor pedig az általa előállított alkalmazások nincsenek titkosítva.
A *Rekall*-alkalmazásokban található egy szabványos menüsor és eszköztár, amellyel a végfelhasználó – egyéb műveletek mellett – például lekérdezéseket indíthat el. Mivel szolgáltatásai meglehetősen széles körűek, ha korlátozni akarjuk a felhasználók számára elérhető lehetőségeket, akkor teljesen le kell tiltanunk a menüt és az eszköztárakat. Aki nem fél az *XML* fájlok kézi szerkesztésétől, az korlátozni tudja a megjelenő gombok és menük körét, illetve sajátokat is létrehozhat. Ugyanakkor ez a használat egy nem támogatott módja, tehát jó volna, ha a szerzők a következő változatot kibővítenék ezzel a lehetőséggel.
A kód teljes egésze, illetve az alkalmazásokban használt űrlapokat leíró *XML* szöveges formátumban tárolódik a fájlrendszerben vagy az adatbázisban. Ha tehát olyan alkalmazást fejlesztünk, amelytől valamilyen bevételt várunk, akkor érdemes megvárnunk a titkosítási és webalkalmazás-készítési lehetőség megjelenését; esetleg megismernünk más megoldásokat, mint az alkalmazás szolgáltatásként való rendelkezésre bocsátását lehetővé tévő *FreeNX*. Egy az alkalmazással együtt terjesztett futási idejű könyvtárat megvásárolva korlátozhatjuk a végfelhasználók által végrehajtható műveletek körét, ugyanis a futási idejű könyvtárak nem tartalmazzák a fejlesztői eszközöket. Ezzel a megoldással azonban nem akadályozhatjuk meg a hozzáértőbb felhasználókat abban, hogy letöltsék a *Rekall* teljes változatát, kimásolják kódunkat az alkalmazásból, majd felhasználják saját céljaikra, esetleg a fájlok módosításával saját szolgáltatásokat valósítsanak meg. A *Rekall Professional* változatának kiadása után az ügyfelek számára személyre szabottan, titkos kulcsra alapuló eljárással tudjuk majd elfedni az alkalmazások belső világát.
Természetesen annak is vannak előnyei, ha mindent nyílt *XML*-ben hagyunk. Nemrég észrevettem, hogy van egy sorozatnyi hibás beállításokkal ellátott összetevőcsoportom, más néven blokkom. Szerencsére semmi szükség nem volt arra, hogy kivág-beilleszt módszerrel javítsam ki, esetleg újratervezzem az egyes blokkok mind a 11 mezőjét, ehelyett egyszerűen átírtam az *XML*-t, a blokkokat a lekérdezések helyett a megfelelő táblákra irányítva, és minden gond nélkül működött.

A Rekall beszerzése

Miután egy ideig dolgozgattam a *Rekall* forrásként letöltött linuxos változatával, a windowsos telepítőkészletként vásárolt változattal is kipróbáltam az alkalmazásomat. Az eredmény meggyőző volt, az alkalmazás mindkét környezetben azonos módon működött. Az átültetés *Windows* alá gyerekjáték volt: egyszerűen készítettem egy biztonsági mentést a linuxos gépemen futó *PostgreSQL* adatbázisról, majd átmásoltam ezt a fájlt,



■ ***3. ábra*** Új tábla létrehozása a Table ablakban



■ ***4. ábra*** Az űrlapvarázsló a munka túlnyomó részét elvégzi helyettünk, mégis hatalmas rugalmasságot biztosít az űrlap tervezésében

valamint a *Rekall* tervezetfájlját a windowsos gépre. Ott visszaállítottam a mentési fájlt a *PostgreSQL 8.0.2* windowsos változata alá, majd a *Rekall* windowsos változata alatt futtattam a tervezetfájlt.

A windowsos telepítés mérete nagyjából 7 MB. Ügyeljünk arra, hogy a *Rekall* telepítése előtt a *Python* megfelelő változatát is telepítsük. A *Rekall 2.2.3*, tehát a jelenlegi kiadás használatához a *Python* a 2.3-as sorozatba tartozó változatainak valamelyike szükséges.

A *Linux* alatti lefordítás roppant egyszerű. *CentOS 3* alatt a fordítás és a telepítés egyaránt hiba nélkül zajlott le. *CentOS 4*-re frissítés után ugyan fordítási hibákat kaptam, ám az alkalmazás telepítése így is lezajlott. Viszonylag régi, 900 MHz-es *Athlon* processzorral és 512 MB memóriával ellátott gépemen a fordítás körülbelül egy órát igényelt.

Első futtatásakor a *Rekall* jó néhány párbeszédpanelt jelenít meg, ezek segítségével a fejlesztői környezet különféle beállításait adhatjuk meg, ilyen például a rekordfrissítések és a törlések ellenőrzése, illetve a fejlesztői környezet elrendezése. A beállításokkal kezdő megadásuk után többé nem kell foglalkoznunk, hacsak le nem töröljük a *Rekall* beállító fájlját.

Példaalkalmazás

Szemléltetni szerettem volna a *Rekallban* végzett fejlesztői munka könnyűségét, ezért összeállítottam egy apró, filozófusok publikációinak nyilvántartására használható alkalmazást. Segítségével mind a fejlesztés egyszerűségét, mind a *Python* alapú parancsfájlkészítési lehetőségeket be tudom mutatni. Célom az *1. ábrán* láthatóhoz hasonló alkalmazás készítése volt, egy kisebb adatbeviteli ablak két lappal, alul pedig egy jelentés- és egy állapotsor, mely az éppen kiválasztott filozófust adja meg.

Első lépésként módosítanunk kell a *PostgreSQL* beállításait, hogy megfelelően együttműködjön a *Rekallal*. Ehhez a helyi alkalmazások által indított kapcsolatok támogatására van szükség, amelynek engedélyezését az A *PostgreSQL* beállítása című széljegyzet foglalja össze. Ha ezzel végeztünk, adjuk hozzá a *philosophy_major* felhasználót, adjunk neki jelszót, valamint hozzuk létre a *philosophers* adatbázist. Mindezeket a lépéseket még az előtt kell elvégeznünk, hogy a *Rekallal* először csatlakoznánk, ugyanis a *Rekall* nem képes felhasználók és adatbázisok hozzáadására. A *PostgreSQL* kiszolgáló újratöltése után létrehozhatjuk az első kapcsolatot.

A következő teendőnk a tervezet létrehozása. Nyissunk meg egy terminálablakot, és hozzunk létre egy könyvtárat a tervezet számára, majd a `rekall` paranccsal indítsuk el a *Rekallt*. Telepítés után a *rekall* futtatható fájljának a */usr/bin* könyvtárban kell lennie.

A tervezet létrehozására szolgáló varázsló első képernyőnjén adjuk meg az adatok tárolására kiválasztott könyvtárat és a tervezet nevét, majd adjuk meg az űrlapok, a jelentések és a többi *XML* adatszerkezet tárolási helyét. Ezek az elemek egy különleges *Rekall Objects* táblában az adatbázisban is tárolható,k de fájlok formájában a fájlrendszerben is elhelyezhetők. Miután kiválasztottuk a tárolási helyet, adjuk meg, hogy milyen adatbázist használunk. Külön-külön illesztőprogramok segítségével több adatforrással is dolgozhatunk, de a most látható párbeszédpanel a fő adatbázisra vonatkozik.

A következő két párbeszéden az adatbázist futtató állomást, a kapcsolatok fogadására használt kaput és a kapcsolatok létrehozásánál használt felhasználónevet és jelszót kell megadnunk. Ha a csatlakozás sikeres, kiválaszthatjuk, hogy melyik adatbázist kívánjuk használni.

Az adatbázis kiválasztása után a *2. ábrán* láthatóhoz hasonló képernyő jelenik meg. Ezen a ponton megkezdhetjük az alkalmazás összeállítását. Ennek első lépése a táblák létrehozása, melynek elvégzéséhez kattintsunk az *Objects (Objektumok)* fa *Tables (Táblák)* elemére. Válasszuk ki a megfelelő kiszolgálót, majd kattintsunk duplán a *Create new table (Új tábla létrehozása)* parancsra. Megjelenik a *3. ábrán* is látható táblaépítő.

A *Rekall* a már meglévő táblákat nem tudja használatba venni – érdekes feladat volna egy adatbázissémákat *Rekall* definíciós fájlokká alakító segédprogramot írni. Ha a *Tables* fában rákattintunk az egér jobb gombjával a tervezetünk nevére, akkor megjelenik egy menü, melyből módunk nyílik táblameghatározás beemelésére. A meghatározásnak *XML* fájlban kell lennie, ilyet viszont *SQL* táblameghatározásból hozhatunk létre.

Itt az ideje, hogy létrehozzunk néhány űrlapot a korábban megadott adatbázissémák alapján. Szerencsére a *Rekall* erre is biztosít néhány könnyen használható eszközt. Ha rákattintunk az *Objects* fa *Forms (Űrlapok)* elemére, majd kibontjuk a tervezetünk után elnevezett

elemet, akkor varázsló segítségével vagy a nélkül hozhatunk létre űrlapot. Még ha lapokra osztott oldalakat tartalmazó, összetett alkalmazásokat készítünk is, érdemes először mindegyik lapot külön űrlapként létrehozni, aztán másolni ezeket az objektumokat, majd beilleszteni őket az űrlap megfelelő blokkjaiba.

A *4. ábrán* követhető, hogy a varázsló segítségével hogyan hoztam létre egy űrlapot a filozófusok táblája alapján. Az űrlapok felett széles körű ellenőrzést gyakorolhatunk, még a varázsló használatakor is. Nemcsak a használni kívánt táblát és mezőket választhatjuk ki, de megadhatunk oldalanként több rekordot, valamint meghatározhatjuk a mezőformátumokat és az oldalhoz önműködően hozzáadandó eszközöket is. A 4. ábrán látható képernyőn a varázsló a képernyő alján látható gombokat és a legalul megjelenő, apró navigációs eszközt adta hozzá. A ***Nav. Toolnak (Navigációs eszköznek)*** nevezett apró kiegészítő segítségével rekordról rekordra lépve navigálhatunk az adatbázisban.

A ***Philosophers (Filozófusok)*** és a ***Publications (Publikációk)*** űrlapját létrehozásuk után (*5. ábra*) közös ablakba kell egyesítenünk. Itt mutatkozik meg az űrlapok kézi létrehozásának előnye, így ugyanis kívánságunk szerint tudjuk hozzáadni az összetevőket. Mielőtt rátérnénk erre, meg kell ismerkednünk a blokk fogalmával.

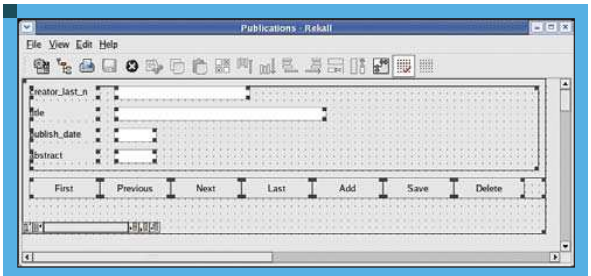
A ***Rekall*** űrlapjaiban az adatokat a blokkoknak nevezett egységek képviselik. Számos blokktípus létezik, a legegyszerűbb a tábla blokk, de van lekérdezés és ***SQL*** blokk is, utóbbiak egy adott lekérdezésből származó adatokat jelenítenek meg. Egy űrlap tetszőleges számú blokkot tartalmazhat.

A blokkok a képernyőn gyakorlatilag tetszőleges elrendezésben megjeleníthetők, illetve noteszablakba, lapokra elosztva is elhelyezhetők. Mivel csak egy ablakot akartam, én is ilyen megoldást választottam.

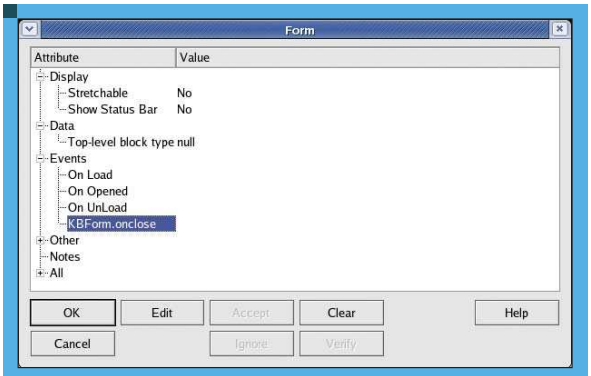
A noteszablak létrehozásához készítettem egy új űrlapot, de a varázsló nélkül. A parancs kiválasztása után a 6. ábrán látható ***Form Attribute (Űrlap jellemzői)*** ablak jelent meg. Mint látható, alapszintű ablakot választottam. Ezt nem lehet átméretezni, nincs állapotsora, valamint legfelsőbb szintű blokkjának típusa null – a kiválasztási párbeszédben ez a ***menu (menü)*** blokk nevet viseli. A legfelsőbb szintű blokk típusaként menüt, avagy nullt választva válik lehetővé, hogy a blokkokat és a vezérlőelemeket tetszőleges módon rendezzük el az űrlapon.

Rájuk kattintva menjünk végig az összes jellemzőn, tegyük meg a szükséges választásokat, illetve kattintsunk rá a párbeszéd alján található ***Accept (Elfogadás)*** gombra. Ezt követően a ***Form Attribute (Űrlap jellemzői)*** oldalhoz sokban hasonlító ***Block attribute (Blokk jellemzői)*** ablak jelenik meg. Mivel most egyszerű menü blokkról van szó, az alapértelmezett értékek túlnyomó részét nyugodtan elfogadhatjuk, a legtöbb érték úgyis csak a tábla vagy a lekérdezés blokkokra érvényes. Ezzel egy üres lapot kapunk, amit a kívánt nagyságúra méretezhetünk, majd hozzáadhatjuk a lapozó vezérlőelemet.

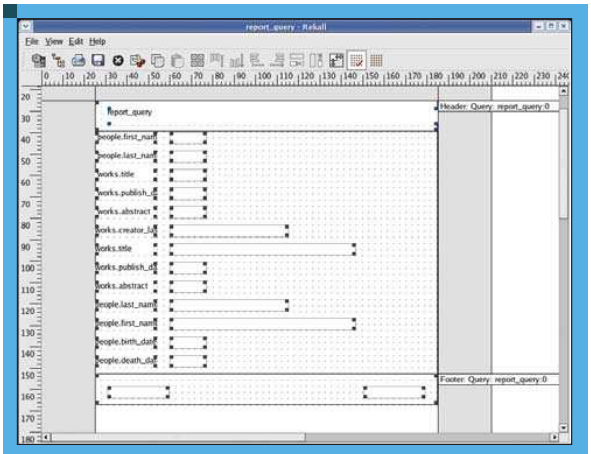
Mindegyik lapon létre kell hoznunk egy a megfelelő táblához tartozó tábla blokkot. Ez után másoljuk át a ***Philosophers*** űrlapon található blokk tartalmát a lapokra osztott ***Philosophers*** ablak megfelelő blokkjaiba. A ***Publications*** lappal ismételjük meg a műveletet. Ha ez is



■ ***5. ábra*** A publikációk űrlapja egy noteszablak része lesz, a 4. ábrán látható fő Philosopher űrlappal együtt



■ ***6. ábra*** Rekall űrlap létrehozása null típusú legfelsőbb szintű blokkal



■ ***7. ábra*** A jelentéskészítő ablakban a jelentések összeállítása legalább olyan egyszerű, mint az űrlapok megtervezése

megvan, nevezzük el a mezőket, módosítsuk a ***Publications*** oldal ***Abstract (Kivonat)*** mezőjének méretét – és végeztünk is, legalábbis ami a noteszablakot jelenti.

Most át kell váltanunk a ***Philosophers*** oldalra, és meg kell adnunk a publikációk kereséséhez használt kulcsot. Erre a fő űrlap alján, a lapozó vezérlőelem alatt látható állapotcímke használható. A címke szöveges értéke minden adatbázisbeli keresésnél vagy új bejegyzés létrehozásánál beállításra kerül. A beállításra akkor kerül sor, amikor az *1. kódrészletben* található kódot a ***Philosophers*** blokk ***On Display (Megjelenítéskor)*** eseményével, visszahívással futtatjuk. A ***Publications*** blokk megjelenítésekor

1. kódrészlet Philosophers blokk:

```
def eventFunc (block, row) :
    someMainForm = block.getForm();
    currBlock = block;
    dataLabel =
someMainForm.getNamedCtrl("current_philosopher");
    dataLabel.setText(currBlock.getNamedCtrl
    ➥("last_name").getValue());
```

2. kódrészlet Publications blokk:

```
def eventFunc (block, row) :
    mainForm = block.getForm();
    currBlock = block;
    dataLabel = mainForm.getNamedCtrl
              ➥("current_philosopher");

currBlock.setUserFilter(dataLabel.getValue());
```

megtörténik az ***On Display*** esemény meghívása a blokkhoz, amely egy felhasználói szűrőt állít be a blokkban megjelenő adatokra. A felhasználói szűrő kódja a *2. kódrészletben* található.
Végül kell valamilyen megoldás az adatbázisban szereplő filozófusok kilistázására. A *7. ábra* a ***Rekall*** jelentéskészítő szolgáltatását szemlélteti.
A ***Rekall*** számos további összetevőt ismer, például jelentéseket, lekérdezéseket és adatmásolókat. Mindegyik összetevő hasonló könnyedséggel hozható létre, és ugyanolyan sokoldalúságot kínál, mint az űrlapok.

Összefoglalás

Mint remélem, sikerült bemutatnom, a ***Rekall*** és a ***PostgreSQL*** párosával – ***Linux*** alatt – bármilyen adatbázis-programozási feladatot rövid idő alatt elvégezhetünk, miközben a számos tanácsadó által igényelt több rendszeren való működés képességét is biztosíthatjuk. Ahogy a vállalatok egyre nagyobb számban térnek át a ***Linux*** használatára a munkaállomásokon, a ***Rekallhoz*** hasonló termékek iránti igény nagyságrendekkel fog bővülni.

A PostgreSQL beállítása

Ha alapértelmezett beállításokkal telepítjük, a ***PostgreSQL 8.0.2*** a felhasználókat linuxos azonosságuk ellenőrzésével hitelesíti. Ha biztonságosabb alkalmazást akarunk írni, állítsuk át az adatbázist jelszó alapú hitelesítésre.
Ezt az alábbi lépéseket követve tehetjük meg.
Először módosítsuk a ***postgres*** adatbázis-felhasználó jelszavát, így akkor is be tudunk jelentkezni, ha jelszót kell megadnunk:

1. Írjuk be a parancssorba a `su` parancsot, majd adjuk meg a root jelszót.
2. Adjuk ki a `su postgres` parancsot.
3. A `psql template1` paranccsal indítsuk el a ***psql*** monitort.
4. A jelszót az `alter user postgres with password 'pgjelszo89'` paranccsal változtathatjuk meg, természetesen az általunk kívánt jelszót használva.
5. A monitorból a `\q` parancs kiadásával és az ENTER lenyomásával léphetünk ki.

Második teendőnk a ***pg_hba.conf*** fájl módosítása, amelyet követően az adatbázis az ***md5*** jelszavakat is el fogja fogadni az összes kapcsolathoz. (Alapesetben az adatbázis-kezelő a hitelesítést az aktuális ***Linux*** fiók alapján végzi.) Alapértelmezett telepítésnél a fájl a ***/var/lib/pgsql/data*** könyvtárban található. Keressük meg benne az alábbiakhoz hasonlító sorokat:

```
# "local" is for Unix domain socket connections
# only
local   all     all                             trust
# IPv4 local connections:
host    all     all     127.0.0.1/32            md5
# IPv6 local connections:
host    all     all     ::1/128                 md5
```

A jelszavak engedélyezéséhez a `local` kezdetű sorban a `trust` kulcsszót módosítsuk `md5`-re, majd mentsük el a fájlt. Indítsuk újra a ***PostgreSQL***-t, a ***Red Hat*** jellegű rendszereken ezt a `/sbin/service postgresql reload` paranccsal tehetjük meg.
Ezt követően a felhasználókat és az adatbázisokat a ***PostgreSQL*** beépített eszközeivel vagy külső eszközökkel, például a ***PgAdminIII*** segítségével hozhatjuk létre. A ***PostgreSQL*** webhelyén mindezekkel a témakörökkel kapcsolatban kiváló leírásokat lehet találni.

Linux Journal 2005. július, 135. szám

Joshua Bentham
Hamarosan filozófiai bakkalaureátusi fokozatot fog szerezni az Ohio állambéli Bexleyben található Capital Universityn. A Linuxot az 1.2.8-as rendszermag megjelenése óta használja. Weblogja a www.globalherald.net/jb címen található, ő pedig a jb42@globalherald.net címen érhető el.

KAPCSOLÓDÓ CÍMEK

Rekall: ➲ www.totalrekall.co.uk

Rekall Revealed: ➲ www.rekallrevealed.org

A hódító Konqueror

A Konqueror egy webböngésző, egy fájlkezelő, egy komplex megjelenítő, egy teljes FTP-kliens és egy teljesen személyre szabható program egyben.

A *Konqueror* név egy a korábbi böngészőgenerációk szójátékára vezethető vissza. Először volt a *Navigator* a *Netscape*-től, majd a felfedező, alias *„Explorer”*, és most a tulajdonképpen angol hódító *„conqueror”*, *K* betűvel írva. A *K* a *KDE* felületre utal. (Tapasztaljuk ezt nagyon sok *KDE*-s programnál).
A következőkben *SUSE 9.2 Professional* operációs rendszer, *KDE 3.3* grafikus felület alatt mutatom be a *Konqueror* használatát.
A *Konqueror* indítása után megjelenik a bemutatkozó oldal (*1. ábra*), ahol egy háromoldalas gyorstalpalót olvashatunk. További linkeken keresztül új ismeretekkel lát el bennünket, például elérhető a *SUSE HelpCenter*-e, vagy hogyan oszthatjuk meg a képernyőnket vízszintesen illetve függőlegesen. Vegyük sorra a képességeit.

■ ***1. ábra*** A Konqueror indítása után megjelenik a bemutatkozó oldal

Konqueror mint webböngésző

A *Konqueror KHTML* modult használ. (Ugyanezt a modult használta fel a *Safari*.) Ez képessé teszi a *Konquerort a HTML 4.01* értelmezésére (*2. ábra*), van benne beépített *Java*, *CSS2* és kétirányú írás támogatás(ilyen például az arab). Támogatja továbbá a Java kisalkalmazásokat (applet) valamint az olyan *Netscape* bővítményeket (*Plug-In*), mint a *Flash*, a *Real Audio*, a *Real Video*, a *PDF*, az *rpm* és az *SSL*.

Űrlapkitöltés használata során megjegyzi az oldalakhoz tartozó beviteli mezők értékeit, mellyel nagyban meggyorsítja az oldalakra való bejelentkezést, vagy adott esetben a rendeléseknél szereplő címek ismételt megadását, de ez csak egy pár eset a felhasznált területek közül.
A menüsort az *ALT* billentyű és a kiemelt betűk lenyomásával érhetjük el. Így a *„Cím”* menüt az *Alt + C* együttes lenyomásával jeleníthetjük meg. Az ablakra vonatkozó műveleteket az *1. Táblázat* foglalja össze. Lehetőségünk van az adott webcím linkként, illetve fájlként való közvetlen elküldésére levélben. Ehhez meghívja az alapértelmezett levelezőszoftvert segítségül. (nálam *KMail*) Ebben a menüben található a weboldal mentése és nyomtatása is, továbbá a más böngészővel megnyitása. (például *Opera*, *Mozilla*, *Firefox*) Ezenkívül a feltelepített HTML- szerkesztők sorakoznak

■ *2. ábra* A Linuxvilág weblapja

■ *3. ábra* Hogyan oszthatjuk meg ablakunkat vízszintesen és függőlegesen

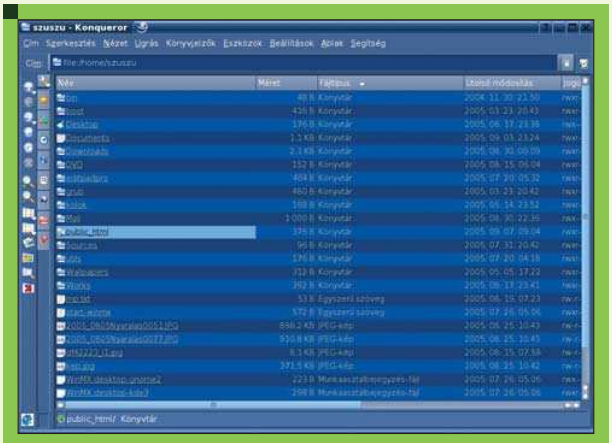
fel, mint webszerkesztők. (***Bluefish***, ***Quanta plus***, ***KVim***) Láthatjuk a menü utolsó sorában a kilépés gyorsbillentyű kombinációját. (***CTRL + Q***)
A *„Szerkesztés”* menüben a szokásos parancsokat találjuk, úgy mint (ha ki van jelölve egy szövegrész) a ***„Másolás”***, ***„Minden kijelölése”***, ***„Keresés”*** és a ***„Következő keresése”***. A hozzájuk tartozó gyorsbillentyűk a megszokottak. Rendre ***CTRL + C***, ***CTRL + A***, ***CTRL + F*** és az ***F3***.
A *„Nézet”* menüpontban a *„Nézetmód”* alatt a beágyazott modulokat találjuk. Elsődlegesen természetesen a ***KHTML-t***, hiszen így látjuk rendesen a weboldalt. A modulok sorát a beágyazott kódszerkesztők a ***Kate*** és a karakteres ***Vim*** követi. A feltüntetett utolsó modul A ***KimageMapEditor,*** melynek segítségével az oldalon található képeket nézegethetjük. Ebben a menüpontban nyílik lehetőségünk a betűméretek nagyítására/kicsinyítésére (***CTRL + +/CTRL + -***), ami segíti az olvashatóságot. A ***„Biztonság...”*** menüben az ***SSL*** titkosításokat tudjuk kezelni. Ha nincs bekapcsolva az automatikus frissítés az ***„Eszközök”*** menüben, akkor ***F5*** lenyomásával frissíthetjük a weboldalainkat.

Az ***„Ugrás”*** menüben az eddig megjelenített oldalaink között vándorolhatunk, egyesével vagy a lent megjelenő listából kiválasztva közvetlenül. Az ugrás az alkalmazásokra, beállításokra és eszközökre egy-egy speciális címet takar. Ezekről később még lesz szó.
Az ***„Automatikus indítás”*** megjeleníti a ***.kde/Autostart/*** alkönyvtárt, ahova létrehozhatjuk azokat a például programokra mutató linkeket, amiket a ***KDE*** betöltése során automatikusan el szeretnénk indítani.
„A leggyakrabban felkeresett oldalak” menüpont neve azt hiszem magáért beszél.
A ***Könyvjelzők*** menüben a fontosnak tartott linkjeinket tárolhatjuk. Adott linkre új könyvjelzőt a ***CTRL + B*** billentyűkombináció segítségével tehetünk. Előbb utóbb annyi lesz, hogy logikailag érdemes őket csoportosítani, ehhez nyújtanak segítséget az ***„Új könyvjelzőmappa”*** és ***„A könyvjelzők módosítása”*** menüpontok.
Az *„Eszközök”* menüpont alatt találhatjuk többek közt az ***„Automatikus frissítés”*** menüt. Ez nagyon jó lehet egy online sportközvetítés során például sakk, vízilabda stb. A ***„HTML-beállítások”***-nál engedélyezhetjük a ***Java***, ***JavaScript***, ***Cookie***-k, Bővítőmodulok használatát, a képek automatikus betöltését, (régen lassabb internet kapcsolat esetén nagy hasznunkra vált) a ***Proxy***, illetve a gyorstár működését. ***„A weboldal lefordítása”*** akkor válhat számunkra nagy segítséggé, ha bizonyos információkat olyan oldalakon találhatunk meg, amelyeknek nyelvét nem ismerjük. Angolról, franciáról, németről keresztbe fordít, azaz bármelyikről bármelyikre elérhető a fordítás. Angolról 14 nyelvre képes fordítani. ***„A weboldal learchiválása”*** annyiban különbözik a ***„Mentés másként”***-től, hogy az oldalhoz tartozó képeket is lementi. Ha bizonyos oldalak egyfajta böngészőre vannak optimalizálva például ***Internet Explorer***, és helytelenül, vagy egyáltalán nem jelennek meg, akkor ***„A böngészőazonosító megváltoztatása”*** során rá tudjuk kényszeríteni az oldalt a megjelenítésre.
A ***Beállítások*** menüben érhető el a ***„Eszköztárak”*** megjelenítése négy különböző eszköztárral. (***Fő***, ***Extra***, ***Címmegadási*** és ***Könyvjelző*** -eszköztár)
Teljes képernyős módra a ***CTRL+Shift+F***-fel válthatunk, míg a menüsort a ***CTRL+M***-mel tüntethetjük el.
Váltogathatunk a nézetprofilok között, úgymint ***„Egyszerű böngésző”***, ***„Lapozós böngésző”***, ***„Fájlkezelés”***, ***„KDE-s fejlesztés”***, ***„Midnight Commander”***, és a ***„Webböngészés”***.
A ***„KDE-s fejlesztés”*** megjelenít egy több lapból álló oldalstruktúrát, ahol dokumentációk

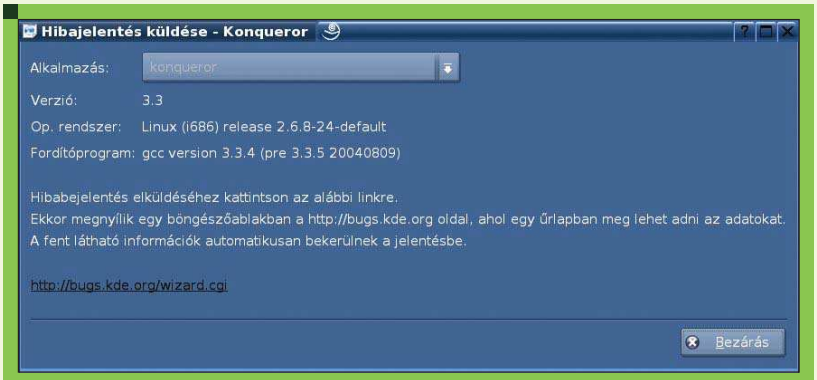
1. táblázat *Ablakra vonatkozó műveletek*

CTRL + N	Új ablak
CTRL + SHIFT + N	Új lap
CTRL + D	Ablak duplikálása
CTRL + O	Cím megnyitása (ami gyorsabban is kivitelezhető az Alt + M alkalmazásával)

■ ***4. ábra*** A Navigációs panel és a Parancsértelmező megjelenése a fájlkezelőben



■ ***6. ábra*** A Fájlkezelő



■ ***5. ábra*** Hibabejelentés küldése

2. táblázat ***A Konquerorral használható protokollok***

http:// vagy https://	weboldal
ftp://	FTP -Kliens
file://	normál fájlkezelő
fish://	SSH -Kliens
smb://	Samba megosztások
lan://	Lan megosztások
imap://	IMAP Server hozzáférés
ldap://	LDAP Server hozzáférés
settings:/	Rendszerkonfiguráció (olyan mint a vezérlőpult, csak Explorer szerűen megjelenítve)
programs:/	Az alkalmazásokra mutat
service:/	SLP-Kliens
info:/	Infooldal
man:/	Manual oldalak

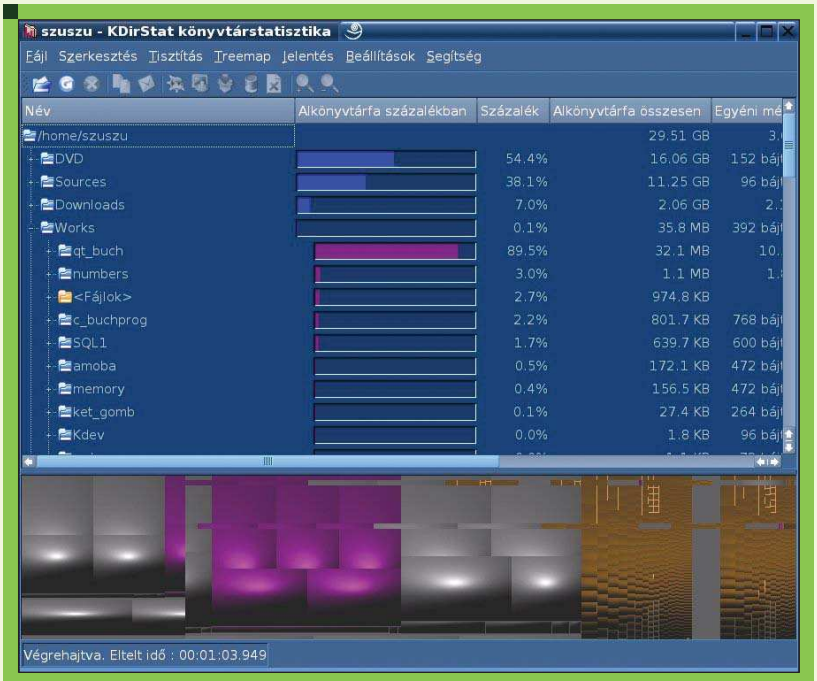
olvashatók a ***KDE*** fejlesztésről, a ***Trolltech***-es ***Qt***-ről, és a ***gcc -ről***. Hasonlóan többlapos oldalstruktúra a ***„Lapozós böngésző”***, de itt a ***kde.org, kde-look.org, kde-app.org,*** és a ***dot.kde.org*** jelenik meg.

„A billentyűparancsok beállítása” hatására megjelenik egy olyan szerkesztő, ahol az összes menüpont fel van sorolva a hozzátartozó alapértelmezett billentyűparancsokkal. Ezeket megváltoztathatjuk esetleg újakat definiálhatunk. ***„Az eszköztárak beállítása”*** teljesen hasonló mint más programokban. Az adott eszköztárra felvehetünk illetve eltávolíthatunk menüket.
A ***„Beállításokban”*** az utolsó menüpont ***„A(z) Konqueror beállítása”***, erről majd később lesz szó.
Az ***„Ablak”*** menüben találunk néhány hasznos funkciót. Ilyenek például a ***„Nézetelválasztás függőlegesen”*** és a ***„Nézetelválasztás vízszintesen”.*** Egy extrém helyzetet mutat a *3. ábra* Mindig az az ablakrész az aktív, ahol a bal alsó sarokban lévő ***RadioButton*** zölden világít. A fent említett képen a felső ablak. Az aktív ablakrészt a ***CTRL+SHIFT+R***-rel zárhatjuk be.
Az ***„Ablak”*** menüben is megtalálható az ***„Új lap”*** menü és egy kicsit másként az ***„Az ablak duplikálása”***, mégpedig ***„Az aktuális ablak duplikálása”*** néven. A különbség csak annyi, hogy itt új lapot nyit meg ugyanabban az ablakban, míg a ***„Cím”*** menüben szereplő, új ablakot is.
Parancsértelmezőt az ablak alatt, Navigációs panelt az ablak bal oldalán jeleníthetünk meg. (*4. ábra*)
Az utolsó menüponthoz értünk, ami nem más mint a ***„Segítség”***. A szokásos menüpontokkal megtűzdelve: ***„Konqueror kézikönyv”***, ***„Mi ez?”***, ***„Névjegy: Konqueror”***, ***„Névjegy: KDE”***, ***„A Konqueror bemutatkozó oldala”***. (már találkozhattunk ezzel

az oldallal a leírás elején 1. ábra)
A *„Hibabejelentés...”* alkalmazása során segítséget nyújthatunk a fejlesztőknek a hibák feltárásában. (*5. ábra*)
A címsorba írt előtagok segítségével a ***Konqueror*** más és más képességét érhetjük el. A lehetséges protokollokat a *2. Táblázat* foglalja össze.

Konqueror mint fájlkezelő

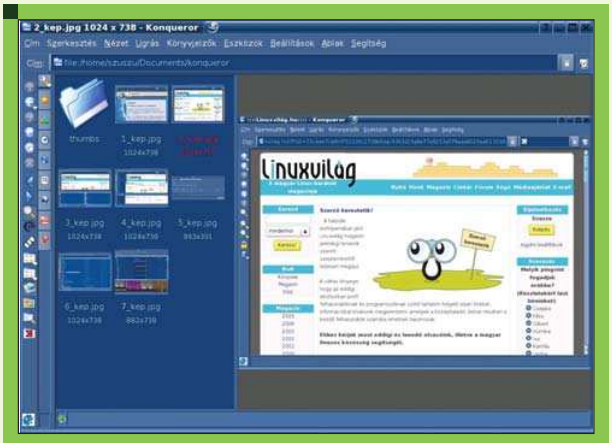
Alapesetben a ***Konqueror*** egy egypaneles fájlkezelő (*6. ábra*), de mint fentebb a webböngésző részben már írtam, itt is használhatjuk a *„Nézetelválasztás függőlegesen”* vagy a *„Nézetelválasztás vízszintesen”* menüket, ami után már mint kétpaneles fájlkezelőt használhatjuk. Lehetőség nyílik a *„Drag and Drop”*-ra, ami nemcsak a panelek között működik, hanem az ablakok között is.
A *„Cím”* menüben megjelenik két új lehetőség. Az egyik a ***KDirStat*** (*7. ábra*). Ez nem más mint egy statisztikai elemzés alkönyvtárszerkezetünkről, rengeteg hasznos információval megspékelve. Az egyik legfontosabb, hogy melyik alkönyvtár mennyi helyet foglal el a winchesterünkön. A könyvtárstatisztika számos lehetőséget tár elénk, de ez már egy másik alkalmazás másik történettel. A másik új lehetőség a ***Cervista***. Fejlesztések során, verziókövetésnél van szerepe. Aki dolgozott már nagyobb projekten, annak nem kell ecsetelni a jelentőségét.
A *„Szerkesztés”* menüben megjelent az *„Új elem létrehozása”*. Létrehozhatunk alkönyvtárat, fájlt, és eszközt. Fájlok közül pl.: prezentációt, szöveges dokumentációt, egyszerű szöveget, táblázatot, ***HTML*** -t és mindenek előtt alkalmazásra mutató linket. Eszközök lehetnek például a háttértárak. Mivel manapság egyre jobban terjednek a digitális kamerák, így ez sem maradhatott ki az eszközök közül, menüpontot tekintve *„Digitális kamera...”*.
Hasznos segítségnek számít a másolásoknál, áthelyezéseknél, törléseknél a *„Kijelölés”* használata. Vigyázat a kijelölés érzékeny a kis- és nagybetűkre, vagyis a ***JPG*** és a ***jpg*** különbözik!
Ha már van(nak) kijelölt állományaink azt lehet átmásolni, áthelyezni, átnevezni, beilleszteni és törölni. Kétféleképpen törölhetünk. Az első lehetőség egy átmeneti tárolóba történő áthelyezést takar, ez *„A szemétkosárba dobás”*, a második fizikai törlés is egyben. A törlés egy olyan művelet, ami megerősítést kér arról, hogy biztosan ezt akarjuk -e csinálni. Nem jó dolog, ha véletlenül olyan állományt vagy alkönyvtárat törlünk le, amit nem szerettünk volna. Azonban lehetőség nyílik ennek a megerősítési kérelemnek a kiiktatására.
A fájlokhoz, alkönyvtárakhoz tartozó tulajdonságokat az ***ALT + Enter***-rel nézhetjük meg. Itt nem csak a fájl, alkönyvtár jellemzője jelenik meg, hanem a jogosultságok, és alkönyvtárak esetében a megosztás is.
A *„Nézet”* menü teljesen kicserélődött a webböngészőéhez képest. A nézetmódban *„Ikonos”*, *„Lista”*, *„Fastruktúra”*, *„Lista (rövid)”*, *„Lista (részletes)”*, *„Szöveges”*, *„Gwenview Image Browser View”*, *„Fájlméret-nézet”*, *„KFileReplace”*, *„CVS-kliens”* -t találunk. A nézetmódokban egyesével kapcsolhatjuk ki/be a megjelenített mezőket. Ezeket érdemes otthon kipróbálni. Mindenkinek más és más nézetmód a szimpatikus. Hasznos, ha tudjuk, hogy ebben a menüben kapcsolhatjuk be a rejtett fájlok megjelenítését.
A *„Nézet”* -ben utolsó menüpont *„A háttér beállítása”*. Használhatjuk a szokásos színválasztó rendszert, vagy képeink közül válogathatunk és helyezhetünk be belőle háttérnek. Mindkettőt könnyen kezelhetjük.
Az *„Ugrás”* és a *„Könyvjelzők”* menük nem változtak.
Az *„Eszközök”* -ben megjelenik a *„Fájlkeresés...”* menüpont. Kereshetünk fájlra, alkönyvtárra, tartalomra, tulajdonságra. Használhatjuk a helyettesítő karaktereket, úgymint * és ?. Vannak olyan esetek, amikor egy alkönyvtárban rengeteg állomány van, számunkra meg csak bizonyos állományok érdekesek. Ekkor jut szerephez



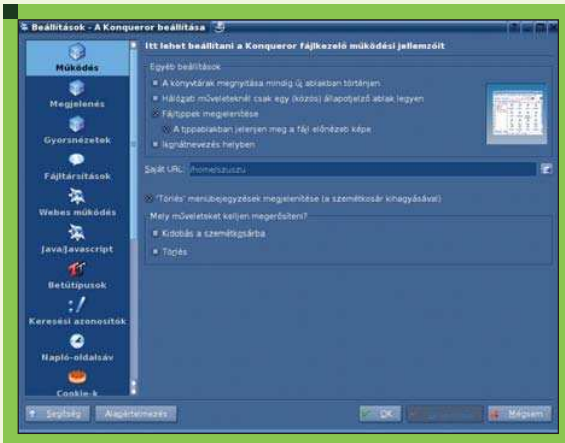
■ *7. ábra* KDirStat

3. táblázat ***Kijelölési műveletek***

CTRL + +	Kijelölés hozzáadása
CTRL + -	Kijelölés törlése
CTRL + *	Kijelölés invertálása
CTRL + U	Kijelölés megszüntetése
CTRL + A	Mindent kijelöl



■ *8. ábra* Qwenview Image Browser View



■ *9. ábra* A Konqueror beállítási lehetőségei

a *„Nézetszűrő”*. Alkalmazása során a nekünk tetsző, nekünk fontos állományainkat tudjuk kilistázni. Rögtönzött weboldalt tudunk létrehozni képeinkből a *„Képbemutató létrehozása...”* menüvel. Lehetőségünk nyílik nemcsak az adott alkönyvtár, hanem az ő alkönyvtárainak hozzá vételére is. Rekurzió mélységet állíthatunk be, hogy milyen alkönyvtármélységig dolgozza fel a struktúrát. Alapesetben végtelenre van állítva. Beállíthatjuk a betűtípust, betűméretet, a képek számát soronként, hogy csak egy pár lehetőséget említsek. A *„Beállítások”* -ban aktív lett *„A könyvtárnézet mentése”* menü, ami arra szolgál, hogy a *Konqueror* megjegyezze az aktuális könyvtárhoz tartozó tulajdonságokat pl.: nézetmód, háttér. Az utolsó két menü *„Ablak”* és *„Segítség”* teljesen megegyezik a webböngészőnél találhatóakkal.

Konqueror mint komplex megjelenítő

A *Konquerornak* beépített fájlmegjelenítő szolgáltatása van. Azaz képes a képeket, dokumentumokat, bizonyos fájlokat anélkül megjeleníteni, hogy valamilyen külső segédprogramot kelljen elindítanunk hozzájuk. A képek megjelenítésére a *KView* -t, a szöveges dokumentumok megjelenítésére a *KWrite* -ot, (szövegkiemeléssel) a *PostScriptekhez* a *KGhostview* -t használja. *DVI* megjelenítő a *KDVI*. Minden *KOffice* dokumentumot képes megjeleníteni a hozzátartozó alkalmazással. A képek megjelenítésére olyan szinten felkészítették, hogy külön nézetmódot is kapott. *„Gwenview Image Browser View”* (*8. ábra*) E nézetmód használata során az eszköztáron megjelenik az előre és hátra nyíl, a forgatás és a diavetítés ikon.

A Konqueror beállítási lehetőségei

A *„Beállítások”* -nál találhatjuk *„A(z Konqueror beállítása...”* menüpontot. Rákattintva a menüpontra megjelenik egy 17 pontból álló menürendszer (*9. ábra*). A *„Működés”* -nél a fájlkezelő működését állíthatjuk be. Itt találkozunk a művelet megerősítés ponttal. Korábban ugye volt szó arról, hogy a törlés műveletét meg kell erősíteni, hát itt kikapcsolhatjuk. :) *„Megjelenésnél”* a fájlkezelő jellemzőit szabályozhatjuk. pl. betűtípus, szín. *„Gyorsnézetek”* szolgál arra, hogy egyes fájlokba már ikonnézetbe belepillanthassunk. *„Fájltársítások”* -nál rendelhetjük össze a fájltípusokat az alkalmazásokkal. *„Webes működés” -el* a webböngésző részeinek működését szabályozhatjuk. Például az automatikus űrlapkitöltés engedélyezése vagy tiltása, az egér működése az oldalon, linkek aláhúzása vagy a képek automatikus betöltése, hogy egy párat kiemeljek a lehetőségek közül. *„Java/Javascript”*: *Java* és *Javascript* beállításai. *„Betűtípusok”* a webböngésző betűtípusainak beállításaira vonatkoznak. *„Keresési azonosítók”*-ról olvashattunk már a bemutatkozó oldalon. Itt szabhatjuk személyre azokat a címrövidítéseket, keresőazonosítókat amiket a címsorba írunk. például: ha a *Google* -n szeretnénk *Linux* után keresni, akkor elég a címsorba beírni a *gg: Linux* címet. Ezáltal nem kell először behozni a *Google* -t és a kereső mezőbe beírni a Linuxot, hanem egyből megkapjuk a keresés eredményét. Saját rövidítéseinkkel bővíthetjük a listát, így nagymértékben meggyorsíthatjuk kedvenc oldalaink elérését. *„Napló-oldalsáv”*, *„Cookie-k”*, *„Gyorstár”*, *„Proxy”* azt takarja, amit a neve hordoz. *„Stíluslapok”* tükrözik a *CSS* használatát. A *„Titkosítás”* és a *„Böngészőazonosító”* -ról pedig volt szó a Konqueror mint webböngésző részben. Egy beállítási lehetőséget említenék még meg itt, ami könnyíti, és gyorsítja a fájlkezelő működését, ez a *„Gyors másolás és mozgatás”* A vezérlőpult, *KDE*-komponensek, fájlkezelő alatt érhető el. Arra szolgál, hogy a *„Másolási műveletek”*, *„Mozgatási műveletek”* során mennyi könyvtárat jegyezzen meg. Alapértelmezett értéke 5, ami még nem zavaró és általában elégnek is bizonyul. A *Konqueror* egy rendkívül összetett és nagy tudású, számos hasznos lehetőséggel ellátott, mégis könnyen kezelhető programrendszer. Kipróbálásához jó időtöltést kívánok. Biztos vagyok benne, hogy a program senkinek sem okoz csalódást.

Szurmai László
(laszlo.szurmai@hu.bosch.com)
4 éve foglalkozom Linuxszal. Leginkább az adatbázis-kezelés érdekel, ezzel foglalkozom már 7 éve hol mellék hol főállásban. Asztaliteniszezem a Kispest III csapatában. Játszani a Diablo II -vel a Battle.net-en szoktam.

KDE alkalmazások (1. rész)
Időzítők

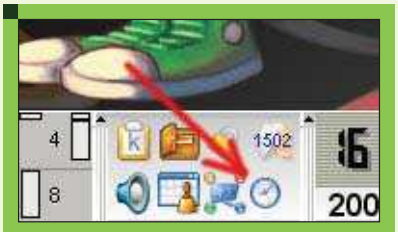
A KDE ablakkezelő rendszer sok olyan apróságot tartalmaz menürendszeren és hivatkozáson kívül, amelyek kellemessé tudnak tenni a munkát és a szórakozást. Sorozatom célja, hogy ezen kis programocskákat részletesen bemutassam. Elsőként kettő olyan mütyürt gyűjtöttem kis csokorba, amelyek az akut időzavart képesek enyhíteni.

KTimer

Általában a segédprogramok között találjuk meg ***KTimer*** néven, ha mégsem itt foglalna helyet és a menürendszerben máshol sem találjuk, akkor elindíthatjuk `ktimer` néven parancssorból is. Ha nem találunk ilyen nevű programot, akkor az általunk használt Linux terjesztés nem tartalmazza alapértelmezésben (pedig már egy ideje a ***KDE Utils*** csomag része). Ez utóbbi esetben meg kell szereznünk a forráskódját és a fordítás-telepítés után már használhatjuk is.

A ***KTimer*** program olyan esetben segít sokat, amikor egy-egy program elindítását időhöz szeretnénk kötni, de nem akarjuk a munkát a `cron` programra bízni (mert nehézkes a használata, illetve nincs szükségünk rendszeres ismétlődésre), esetleg az `at` parancs sem áll kézre. Ritka igény, hogy másodpercre pontosan időzítsünk valamilyen eseményt, amelyre se az `at` se a `cron` nem képes, mivel ezen programoknál a legkisebb beállítható időegység egy perc.

A program beépül a tálcára is, illetve ablakot is nyit az első futtatáskor. Ha el szeretnénk tüntetni a megnyílt ablakot, akkor a tálcán lévő kis óra ikonon jobb egérgombbal kattintva a *„Minimalizálás”* menüpontot kiválasztva háttérbe küldhetjük, s ilyenkor az ablaklistában sem jelenik meg. A program ablaka olyan egyszerű, hogy azonnal el lehet igazodni a felületén. Láthatunk rajta egy táblázatot, amely



1. ábra KTimer a tálcán

az eddig használt időzítéseket mutatja, amelyeket az *„Új”* nyomógombbal tudunk készíteni. A kijelölt időzítéseket az „Eltávolítás” gomb segítségével tudjuk törölni. Ha rákattintunk egy sorra, akkor a jellemzői megjelennek az alsó *„Beállítások”* feliratot viselő területen, s itt tudjuk beírni a szükséges adatokat. Az első mezőbe tudjuk a futtatni kíván program nevét a teljes parancssorral együtt. Itt képesek leszünk tetszőleges programot futtatni, amelynek tudjuk a nevét. A késleltetéssel tudjuk beállítani azt az időt, amelynek nulláig kell csökkennie, hogy a kívánt program elindulhasson, ide maximálisan 99999 másodperc írható, amely nem egy kis szám. Ha ismételtetni szeretnénk az időzítőt, akkor be kell pipálni ezt a kis jelölőnégyzetet is, amelynek hatására a számláló nem áll meg a program futtatásakor, hanem újraindul a késleltetésnél megadott értékkel. Ha óvatlanok vagyunk, akkor egy kis időközzel megadott ismétlés esetén a futtatni kívánt program sok tíz példányban is futni fog. Ezt tudjuk megelőzni egy pipával a *„Csak egy programpéldány”* jelölőnégyzetben.

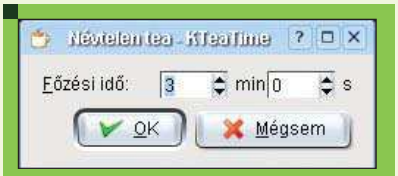
Az időzítőt a jobb oldalán látható *„Állapot”* feliratú területen található nyomógombokkal tudjuk vezérelni. Itt a szokásos ikonokat láthatjuk, amelyek az időzítést elindítják, szüneteltetik vagy megállítják. A csúszkával a számláló aktuális értékét tudjuk befolyásolni, természetesen a nulla és a késleltetés által meghatározott határok között.

2. ábra KTimer beállításai

■ *3. ábra* KTeaTime a tálcán

■ *4. ábra* Visszalévő idő

■ *5. ábra* Kész a tea



■ *6. ábra* Gyors teakészítés

■ *7. ábra* Beállítások

KTeaTime

Teafüggő egyéneknek kötelező program a tálcán, nélkülözhetetlen segédeszköz a jó tea elkészítéséhez (de akár tojásfőzéshez vagy úgy általában főzéshez is használható).

A teafőző alapesetben a tálcán egy kis kék bögre képében jelenik meg, amely csendben várja a teaidőt. Ha rákattintunk bal egérgombbal, akkor egy alapértelmezett listát kapunk a főbb teatípusokról és azok elkészítési idejéről. Ebből a listából ki tudjuk választani a megfelelő időtartamot, és már vissza is térhetünk a megszokott tevékenységünkhöz, a ***KTeaTime*** figyelmeztetni fog az elkészült teára (*5. ábra*).

A tea elkészítésének folyamatát egy kis kördiagram mutatja, így nyomon tudjuk követni, hogy mennyi időnk van még a filter vagy a teatojás áztatására. Ha az előre megadott időtől különböző időtartamot kellene megadnunk hirtelen, akkor a *„Névtelen...”* menüpontot választva már meg is adhatjuk gyorsan a szükséges időt (*6. ábra*).

Ha gyakran teáznánk olyan teafűvel, amelyet 5 percnél tovább kell vízben áztatni (például ha szeretjük a jól kioldódott kesernyés teát), akkor érdemes a kék bögrén jobb gombbal kattintva a „Beállítások...” menüpontot választani, ahol fel tudjuk venni a gyorsmenüre a kívánalmainkat.

A megnyíló ablak bal felső részén a teáinkat tudjuk felsorolni és az alsó részen az ikonokkal kezelni (új létrehozása, törlés, pozíció módosítása).

A jobb oldalon a tea idejét állíthatjuk be, amelyet a program kilépés után is megjegyez. Hozzá tudjuk rendelni a tea elkészülését egy ***KDE*** rendszereseményhez, amely felirattal és hangjelzéssel is jár, így biztos nem kerüli el a figyelmünket a kész tea, amelyet immár a gőzölgő kék bögre jelez a tálcán... egészségünkre!)

■ *8. ábra* Gőzölgő finomság

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

Halló, itt az internet – VoIP alkalmazások Linux alatt

Olvasóink közül néhányan talán emlékeznek még azokra az időkre, amikor egy vonalas telefon „birtoklása”nem annyiból állt, hogy az ember előfizetett a szolgáltatásra, hanem egyfajta kiváltságnak számított. Aztán jött a méregdrága mobil, majd az olcsó mobil, mostanra pedig ha telefonálni akarunk, előfordulhat, hogy se előfizetés, se a hagyományos értelemben vett készülék nem kell hozzá...

Mi a VoIP?

A ***VoIP (Voice over IP)*** nem más, mint beszédhang átvitele az ***Internet Protokoll*** segítségével. A ***VoIP*** telefonhívások tehát nem – vagy nem végig – a klasszikus telefonvonalakon közlekednek, de ugyanolyan nagy távolságokat tesznek meg az internet közegét használva.

Mostanság, amikor egyre több embernek van otthon ***szélessávú*** (***ADSL*** vagy kábeltévé) internetkapcsolata, kezdenek igazán elterjedni az alternatív szolgáltatók, amelyek havidíj fizetése nélkül teszik lehetővé hogy hívást kezdeményezzünk vagy fogadjunk. Kezdetben csak netről netre lehetett hívást kezdeményezni adott ***VoIP*** szolgáltatón belül, de manapság már bárhonnan lehet hívni bárkit, akár vonalas készüléket is.

Ez például az én esetben azt jelenti, hogy minimális percdíjért tudom felhívni szüleim normál vonalas telefonját az ország túlsó végében. És ehhez egy szélessávú internetelérés, egy mikrofonos fejhallgató, egy a ***VoIP*** szolgáltatónál bejegyzett felhasználói fiók (account) no és persze rendelkezésre álló egyenleg szükséges.

És ez még nem minden, a dolog ugyanis fordítva is működik. Ők is tudnak engem hívni, feltéve, hogy az általa futtatott kliensprogram aktív, vagyis rá vagyok kapcsolódva a szolgáltató rendszerére. Előfordulhat persze az is, hogy nem vagyok gépközelben. Az ilyen esetekre a legtöbb szolgáltató ***hangpostafiók***ot (üzenetrögzítőt) biztosít.

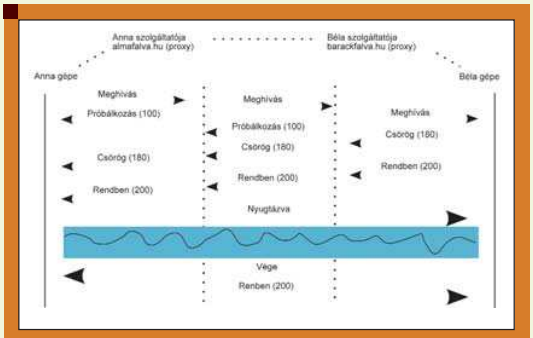
VoIP hívást tehát persze nem csak netről lehet indítani, de jelen cikkben ezt az esetet nem tárgyalom.

A rendszer lelke a SIP (Session Initiate Protocol)

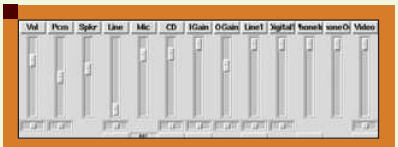
A legtöbb ***VoIP*** szolgáltató a ***SIP*** szabványt használja, ami rugalmasan bővíthető, alakítható a felmerülő igényeknek megfelelően.

Ez csak az alapréteg, akár hangot, akár videót átvihetünk, sőt lehetővé teszi a konferenciabeszélgetést is. Erre épül következő rétegként a hangátvitel. Az hogy a kliensünk éppen milyen hangkódolást alkalmaz változó, sőt számos kliens megadja a választás lehetőségét is.

A ***SIP*** működési elvét a mellékelt diagramon követhetjük nyomon. (Az ábra az ***RFC3261***-es dokumentáció 4. bekezdéséből származik.) Ha valaki a hivatkozott ***RFC***-be beleolvas, jól látható, hogy a bevált ***HTTP*** kódokat alkalmazzák, így aki szeretne saját ***SIP*** klienst írni, minden további nélkül megteheti.



1. ábra A SIP protokoll működési elve



2. ábra A mixer

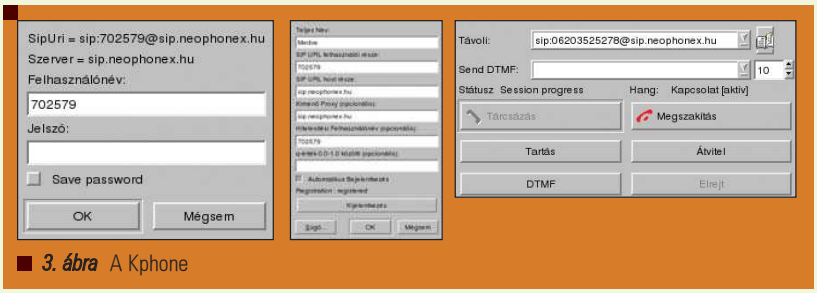
NeophoneX

Akik nem írni, csak használni szeretnének ilyen programokat, azoknak három, természetesen ***Linux*** alatt futó változatot mutatok be. Ezek a ***Kphone***, az ***SJPhone*** és az ***Xten-Lite***.

A mellékelt ábrákon látható beállítások a magyar ***NeophoneX*** szolgáltatás hálózatával használhatóak. Hogy miért éppen a ***NeophoneX*** – az ***Euroweb*** szolgáltatása – lesz a példa?

Az ***Euroweb*** már korábban is lehetővé tette a normál telefonalról ***VoIP*** hívás kezdeményezését. Idén februárban azonban elindult a ***nettelefon*** szolgáltatás és ez azóta több mint 18000 – ebből körülbelül 14000 aktív – felhasználót szerzett.

A szolgáltatás sikerét jól mutatja, hogy segítségével eddig két és fél millió percnyi hívást bonyolítottak. Fontos megjegyezni, hogy a ***NeophoneX***



3. ábra A Kphone

másodpercalapú a szolgáltató, vagyis itt teljesen kizárt, hogy például 12 másodperces hívásért 60 másodpercnyi díjat fizessünk. Kiváló hangminőséget nyújt és a feltöltőkártyái (1000, 2000 és 5000 forint – 6 hónapon belül fel kell használni) az ország számos pontján elérhetőek. Én történetesen például a kedvenc újságosomnál szoktam megvásárolni, ugyanott ahol a *Linuxvilágot* is. Feltöltőkártyára természetesen csak akkor van szükség, ha nem kizárólag netes ismerősöket szeretnék felhívni, sőt ahhoz sem szükséges egyenleggel rendelkeznünk, hogy az interneten kívülről tudjunk hívást fogadni.

A regisztráció egyszerű, a cég honlapján végezhető el. Kapunk egy hatjegyű számot (ez lesz a virtuális telefonszámunk), illetve fontos még megjegyeznünk a bejelentkezési nevünket (login) és a jelszót, amit megadtunk. Ezekre természetesen szükségünk lesz.

Az interneten kívülről – vagyis például egy normál telefonról – úgy lehet egy *NeophoneX* ügyfelet elérni, hogy tárcsázzuk a helyi hívásnak minősülő 888-181-es telefonszámot majd utána a hívott fél hatjegyű virtuális telefonszámát. Amennyiben az illető nincs a rendszerre kapcsolódva, úgy a hívó fél üzenetet hagyhat, amit a címzett elektronikus levélben *wav* formátumban megkap.

Számkijelzés is van, akárcsak a mobiloknál. Ha a hívó fél oldaláról a telefonközpont képes elküldeni a telefonszámot, a hívott félnél a kliensprogram kiírja azt, s eldöntheti, felveszi a virtuális kagylót vagy sem. A dolog egyetlen árnyoldala, hogy a számkijelzés visszafelé nem működik. Ha tehát valakit *NeophoneX*-ről hívok, hiába támogatja a hívott fél telefonja a *hívószám-kijelzést*, az nem fog megjelenni a készülékén. Az ok egyszerű: a *VoIP-átjáró*, vagyis az a gép amelyen keresztül a netről kimegy a hang a normál telefonra, nem támogatja a hívószámküldést.

Tesztkörnyezet

A tesztelést egy 900 MHz-es *Celeron* processzorral szerelt, 192 megabájt memóriát tartalmazó *Debian GNU Linuxot* futtató gépen végeztem. Korábbi tapasztalataim ugyanakkor azt mutatják, hogy akár egy 2-300 MHz-es géppel is lehet „IP-telefonálni”. A tesztgépben egy *SB Live 5.1*-es hangkártya volt. A típus igazából nem fontos, a lényeg, hogy rendelkezzen *full duplex* képességgel, vagyis egyszerre tudjon felvenni és lejátszani. Ezt a legegyszerűbben úgy lehet tesztelni, hogy az *xmms*-ben elindítunk egy webes rádiót és ezzel egyidejűleg az *audacity* hangfeldolgozó programmal megpróbálunk felvenni mondjuk mikrofonról. Ha megy, akkor *full duplex* kártyánk van.

Fontos még, hogy ne felejtsük el beállítani a keverőt úgy, hogy a mikrofon legyen a felvételi forrás, ezt ugyanis nem minden *VoIP* program teszi meg helyettünk. Ami az igényelt sávszélességet illeti a minimálisan ajánlott sebesség a 128 kbit/s mind a fel-, mind a letöltésre. (Ez tehát azt jelenti, hogy az *ISDN* vonal sajnos nem minden hangkódolás esetén elég.) Aki tényleg csak a minimálisan ajánlott átviteli sebességgel rendelkezik, annak ajánlott a netes telefonálás alatt felfüggeszteni minden egyéb, a sávszélességet foglaló tevékenységet, különben „darabos” lesz a beszéd.

Ennyi bevezető után lássuk tehát az alkalmazásokat.

Kphone

Weblapja a ➲ *http://www.wirlab.net/kphone/* helyen található.

Előnye, hogy könnyen átlátható, és létezik *Debian* csomag formájában is. Ugyanakkor az egyszerűsége helyenként már-már puritán külsőt kölcsönöz neki.

Személyes beállításainkat a *Fájl* menüt tartalmazó csíkra kattintva tudjuk bevinni. Hívást a kamera ikontól balra lévő telefon ikonnal lehet kezdeményezni. Fontos megjegyezni, hogy ez a program nem kéri a *Neophone* regisztrációkor megadott bejelentkezési nevet, csak a telefonszámot és a jelszót.

X-Ten Lite

Weblapja a ➲ http://www.xten.com helyen található.

Előnye a grafikailag szép kivitel, illetve hogy létezik Windows és Macintosh alá is. Ugyanakkor a PDA-kra készített változat verzió sajnos fizetős. Szintén hátrány, bár nem túl jelentős, hogy a cikkben bemutatott változat is úgynevezett „lite” verzió, a teljes verzióért fizetni kell.

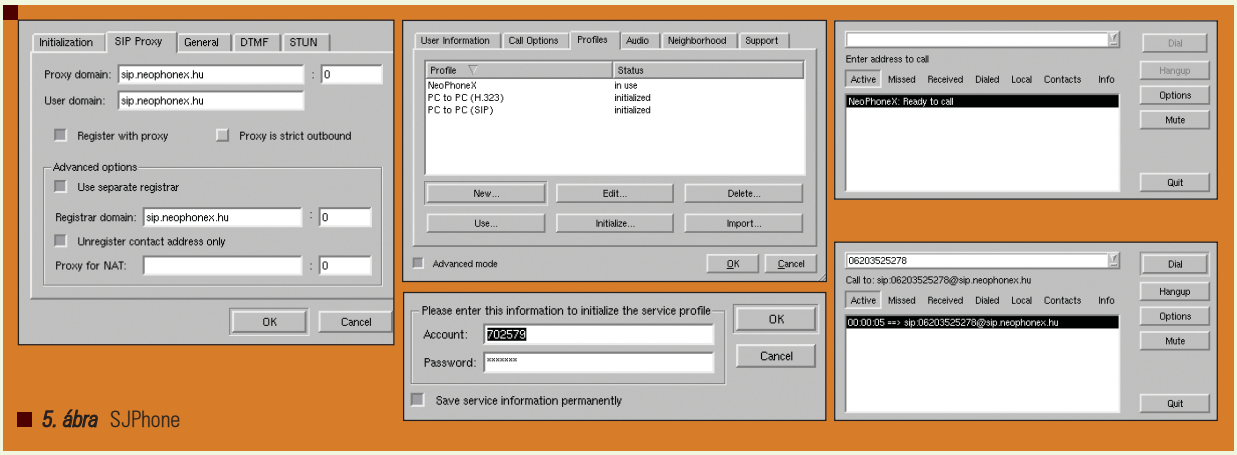
Annak ellenére, hogy a *„Lite”* jelző a csökkentett képességeket jelzi, a program teljesen jól használható. Első induláskor azonnal megjelenik a beállító menü. Ha valamit elfelejtünk, természetesen később is pótolhatjuk a hiányt. A beállítópanelt a *Menü -> System settings -> SIP Proxy -> Default* menüpont alatt találjuk.

Maga a beállítómenü *CLEAR* gombtól jobbra található. Hívást kezdeményezni egészen egyszerűen lehet: meg kell nyomni a *CLEAR* feletti számok (1,2,3) egyikét, be kell gépelni a telefonszámot, majd a végén a zöld telefongombot. A hívást befejezni vagy a bejövő hívást elutasítani természetesen a piros telefongombbal lehet. Aki szeret kísérletezni, próbálja ki a különböző hangkódolásokat. Az ügyfélprogram *GSM* feliratú részénél három kódolást lehet beállítani.

4. ábra X-Ten Lite

Összefoglalás

Mindhárom program képes ***NAT-olt (Network Address Translation)*** hálózat mögül port átirányítás ***(portforward)*** nélkül is hívást indítani és fogadni. Bármelyiket is választjuk mindegyik szabadon használható, vagy van ilyen változata is, ami pedig a percdíjakat illeti, nos a ***VoIP***-pal ezen a téren nem könnyű versenyezni. Nekem személy szerint az ***SJPhone*** a kedvencem, de az Olvasónak azt javaslom, próbálja ki mindhármat, csak aztán döntsön. Ha valamelyik program beállításával



5. ábra SJPhone

Aki viszont egyszerűen csak használni szeretné a programot, annak azt ajánlom, hagyja ezeket a beállításokat az alapértelmezett értékeken.

SJPhone

Weblapja a ➲ ***http://www.sjlabs.com*** helyen található.
Itt is előnyként említhető, hogy a programnak nem csak ***Linux***, hanem ***Windows***, ***Mac*** és ***PocketPC*** rendszerekhez is létezik változata. Bár egy ilyen áttekintő értékelésben a cikk írójának nyilván az a feladata, hogy az előnyöket és hátrányokat egyaránt kiemelje, utóbbiakról igazán nem tudok nyilatkozni. Talán azt megemlíthetem, hogy a linuxos változatnak kissé puritán a kinézete a ***PDA***-n futóhoz képest, de ez a használhatóságot egyáltalán nem befolyásolja. A beállítások megadása és a tárcsázás az előbbi két programhoz teljesen hasonlóan történik.

Lehetséges hibaüzenetek

Amennyiben ***Forbidden (403-as)*** hibaüzenetet kapunk a rendszertől, három dolgot érdemes megnézni.

A kezdeti időszakban ilyenkor gyakran jelszóváltást tanácsolt az ügyfélszolgálat. (Nagyon rugalmas a ***Neophone*** e-mail-es ügyfélszolgálata, általában 24 órán belül válaszolnak.) Előfordulhat, hogy egyszerűen elfogyott az egyenlegünk, és emiatt nem tudunk külső hálózatra (vezetékes vagy mobil) hívást kezdeményezni. A harmadik lehetőség kicsit problémásabb: ellenőrizzük nem vagyunk-e „tűzfal mögé zárva"? Ez különösen cégeknél fordulhat elő, ahol a rendszergazda gyakorlatilag hivatalból szenved heveny üldözési mániában. Ez persze nem baj, különösen ha jó viszonyban vagyunk vele, és kiengedi az adatforgalmunkat.

gondok merülnének fel, és a képernyőképek sem segítenek, amennyire időm engedi, e-mailben megpróbálok segíteni. Persze ehhez érdemes hozzátenni, hogy a ***NeophoneX***-nek fóruma is van, ami szintén tele van hasznos ötletekkel.

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

RFC (leírás) a SIP protokollról: ➲ http://www.faqs.org/rfcs/rfc3261.html
NeophoneX: ➲ http://www.neophonex.hu
Kphone: ➲ http://www.wirlab.net/kphone/
SJPhone: ➲ http://www.sjlabs.com/
X-Ten Lite: ➲ http://www.xten.com/

Az Ubuntu Linux telepítése és beállítása (2. rész)

Akik az előző számban velem tartottak az Ubuntu Linux telepítésénél, észrevehették, hogy a cikk megjelenése után nem sokkal (október 13-án) már meg is jelenik az újabb verzió: a Breezy Badger. Hát igen, ennek a disztribúciónak a fejlesztése gőzerővel folyik, csak győzzük kapkodni a fejünket. Persze a verziók között nem a gyökeres változtatások vannak a túlsúlyban, hanem a finomabb eltérések.

Így a *Hoary* telepítése szinte teljesen megegyezik a *Breezy* telepítésével. De bizonyára vannak, akik már az új verzió megjelenése után telepítenek, számukra azért szeretnék néhány kisebb változtatást bemutatni.

HP laptoppal rendelkezők számára fontos információ lehet, hogy számukra külön Ubuntu telepítő létezik, amit a ➲ *http://www.ubuntu.com/support/custom/hplaptops* címről tölthetnek le.

A *Breezy* telepítésekor már nem nem kapunk vissza „fekete képernyőt”, ahol a kiadott parancsok listája futna (emlékezzünk: az előző számban a csomagok telepítése így zajlott), most már ezt is „magához ragadta” a telepítő, így az éppen telepített csomagokat egyesével írja ki és egy folyamatjelzőt is láthatunk. Particionáláskor már létrehozhatunk átméretezhető partíciókat is (ez az úgynevezett *LVM*).

Indításkor pedig a megszokott szöveges információk mellett egy elegáns úgynevezett *„splash”* grafikát is megjelenít. Mivel haladnunk kell a korral, talán megbocsátják olvasóim, ha a következőkben a *Breezy* alapján folytatom az *Ubuntut* bemutató sorozatomat, ami persze nem fog jelentősen eltérni attól, mintha a *Hoary* lenne a minta. A következő számban bemutatom azt is, miként frissíthetnek a régebbi *Ubuntu* tulajdonosok az új verzióra.

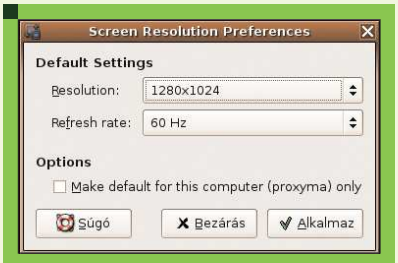
1. ábra Felhasználók és csoportok beállítása

Előfeltételek a beállításokhoz

Miután sikeresen telepítettük, érdemes egy kicsit még saját igényeinkhez megfelelően igazítani bármely operációs rendszert. Az is lehet, hogy egyikmásik szolgáltatás nem működik. Ezeket ilyenkor kézileg kell beállítanunk. De ha szeretnénk jobban megérteni egy operációs rendszer „lelkivilágát”, akkor is érdemes a beállításokkal bajlódnunk. Ehhez egy *Linux* kiváló társnak fog bizonyulni. Az *Ubuntu* pedig olyan rendszer, amely mind a kezdőknek, mind a haladóknak megfelelő lehet.

Mielőtt azonban belekezdenék a beállítások magyarázatába, fontosnak tartok megemlíteni néhány dolgot, amire szükségünk lehet bármely linuxos rendszer konfigurálásához. Sajnos arra nincs lehetőségem, hogy mindent elmagyarázzak, a kezdőknek érdemes részletesebben is utánanézni a következő dolgoknak.

Sokan riasztónak találják egy *Windows* (vagy akár *MacOS*) után, hogy *Linuxban* gyakorta ajánlják a grafikus eszközökön kívül a terminált, ami elég „fapadosnak” tűnhet az ablakok kényelmes megoldásai után. Egy idő után azonban rá fogunk jönni, hogy egyrészt a modern *Linux* disztribúciók igyekeznek minden alapvető beállításhoz grafikus alternatívát adni, másrészt a terminálos, azaz szöveges vagy más néven karakteres beviteli lehetőség nem hátránya, hanem éppen ellenkezőleg, erőssége a Linuxnak.



■ *2. ábra* A képernyő felbontásának beállítása

Akinek egy kis affinitása van a számítógép használatához, könnyen rászokhat a lassabb „egerezés” helyett a gyorsan átírható szöveges állományok használatára. És mint egy-két példánál látni fogjuk, hogy ezek általában könnyen átláthatóak, csupán azt kell tudni, milyen beállítás hol tárolódik. Az ***Ubuntu*** a ***Debian Linuxra*** épül, teljes egészében annak struktúráját használja, így az itt szerzett ismeretek teljes egészében vagy kis módosítással a ***Debianra*** is érvényesek.

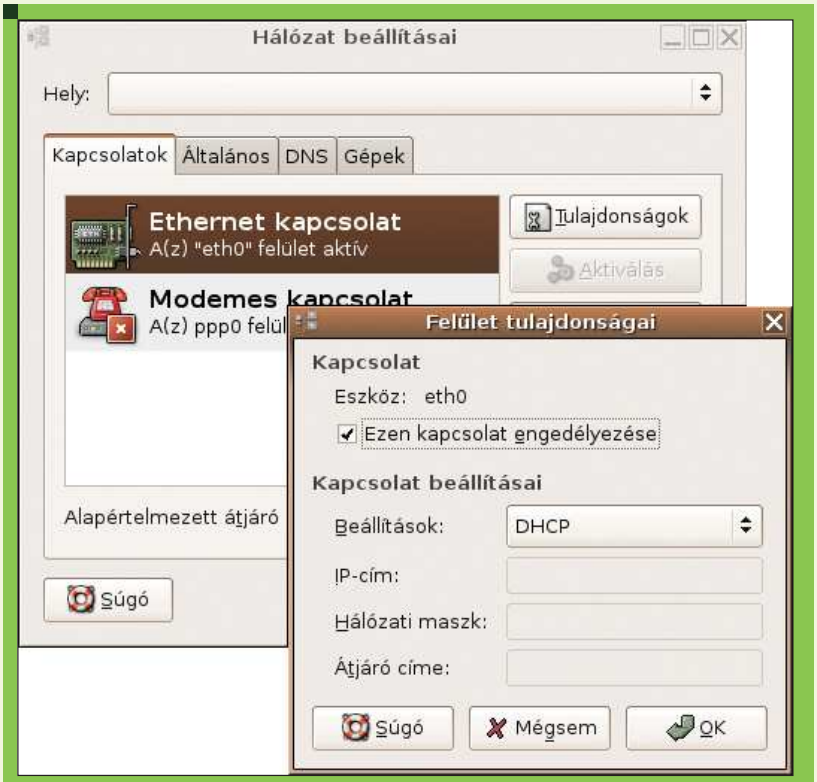
Ha éppen valamely probléma miatt a grafikus rendszer nem működik, akkor nincs is más lehetőségünk, mint a karakteres felületet használni, hogy „felélesszük” az ablakozó rendszert. Amikor gépünk könyvtárszerkezetében dolgozunk, sose felejtsük el, hogy bármely ***UNIX*** alapú rendszerben megkülönböztetjük a kis- és nagybetűket. A könyvtárrendszerben a beállítások leggyakrabban a */etc* könyvtár alatt tárolódnak. Saját könyvtárunk pedig a */home* könyvtáron belül található, a felhasználónként változó beállításokat is itt fogjuk megtalálni.

A kezdő „/”-jel pedig a gyökérkönyvtárat jelenti, azaz hogy a megadott elérési út onnan kiindulva értendő. Terminálként az ***„Alkalmazások / Kellékek”*** menüben lévő Terminált nyissuk meg, ez a ***Gnome*** ablakkezelő alapértelmezett terminál programja lesz. Szükségünk lesz egy jól használható karakteres szövegszerkesztőre is. A profi felhasználók közül bizonyára legtöbben a ***vi***-t ajánlják erre, de ennek használata kezdésnek akár elég furcsa is lehet. A ***nano*** editor is mindig rendelkezésünkre áll, mi ezt fogjuk használni: könnyen megtanulható és a képernyő alján mindig láthatóak a leggyakoribb billentyűparancsok. Sokan szeretik az ***mcedit***-et (a ***Midnight Commander*** része) is felhasználóbarát működése miatt, de az alapértelmezésben ez nem települ fel. A ***Linuxban*** szerencsére ugyanarra a feladatra általában többféle lehetőségünk is van.

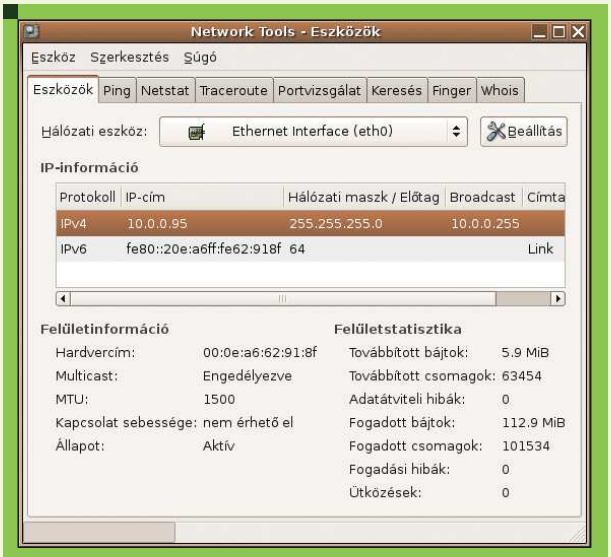
Felhasználók, jogosultságok

Mielőtt bármit is tennénk, érdemes megismerkedni a felhasználói rendszerrel, illetve hogy mihez milyen jogosultságokkal rendelkezünk. Telepítéskor már megkérdezte a telepítő tőlünk, milyen néven kívánunk bejelentkezni. A ***Linux*** többfelhasználós rendszer, minden egyes felhasználónak saját könyvtárai, beállításai vannak, valamint mindegyik külön jogosultsági rendszerrel is rendelkezik. Mindig létezik egy kitüntetett felhasználó, a root vagy superuser, aki mindenhez hozzáférhet, bármit csinálhat. Ez bizonyos esetekben nem túl biztató. Ha például egy program képes root jogokkal futni, teljesen tönkre is teheti rendszerünket (ez ismerős lehet más, nem unixos operációs rendszereknél).

Ezért az ***Ubuntu*** egyik alapelve, hogy ne is lehessen root-ként belépni, így máris egy gonddal kevesebb. Ez így nagyon okos megoldás, de mi van ha olyan dolgot kell csinálnunk, ami root jogokat igényel (például telepíteni szeretnénk)? Ekkor a Linux képes arra a feladatra odaadni számunkra ezeket a jogokat, miközben minden máshol szimpla felhasználóként dolgozunk. Ezt a `sudo` paranccsal (vagy grafikus társával a `gksudo`-val) tehetjük meg. A ***„sudo”*** jelentése: ***„super user do”***, azaz a legfőbb felhasználó csinálja. Pontosabban: a legfőbb felhasználó nevében csináljuk. A `sudo` azonban csak azokra a parancsokra, alkalmazásokra működik, amelyekre a rendszergazda engedélyt adott a számunkra. Ez a */etc/sudoers* fájlban van eltárolva. Ezt most nem elemezném, csupán annyit érdemes tudnunk, hogy az ***Ubuntu*** beállításai szerint az `admin` csoportban lévő felhasználók bármit futtathatnak a superuser nevében. Telepítéskor a megadott felhasználót automatikusan hozzárendelte ehhez a csoporthoz, az általunk esetleg később létrehozott felhasználókkal viszont már nem ez a helyzet, ők semmit sem fognak tudni rendszergazdai jogosultságokkal futtatni, hacsak erre engedélyt nem adunk nekik. A rendszergazdai jogok `sudo`-val való átvételekor természetesen a saját



■ *3. ábra* Hálózatunk DHCP-vel működik



■ ***4. ábra*** Hálózati információk megjelenítése

■ ***5. ábra*** Bejelentkezés egy Windows hálózatba

jelszavunkat kell megadnunk. Tapasztalni fogjuk, hogy bizonyos beállításoknál vagy programoknál meg kell adni jelszavunkat: ilyenkor az alkalmazás ugyan rendszergazdai jogokat igényel, de mivel nekünk lehetőségünk van azokat átvenni, saját jelszavunkkal továbbléphetünk. Ha nem grafikus módon, hanem terminálban szeretnénk ilyet tenni, először ki kell adnunk a `sudo` parancsot, utána pedig megadni a futtatni kívánt alkalmazást. Erre majd fogok mutatni példát, amikor a gyakorlatban is használni kell a terminált.

Az *1. ábrán* látható a ***„Felhasználók és csoportok”*** párbeszédablak. Itt jól áttekinthetően beállíthatjuk rendszerünk felhasználóit és azok csoporttagságait is. Ezt a ***„Rendszer”*** menüből érhetjük el (és természetesen csak superuser jogosultságokkal, így a jelszavunkat is meg kell adnunk). Itt vehetünk fel újabb felhasználókat vagy módosíthatjuk azokat. Ha átkattintunk a ***„Csoportok”*** fülre, ott a legfontosabb csoportok listáját láthatjuk. Ha az ***„admin”*** csoport tulajdonságait nézzük meg, láthatjuk, hogy saját magunk már benne van, tehát rendelkezünk adminisztrátori, más néven rendszergazdai jogosítványokkal (természetesen a `sudo`-n keresztül). Ha felveszünk egy másik felhasználót, neki csak akkor lesznek ilyen jogai, ha őt is bevesszük ebbe a csoportba. Egyébként a felhasználó tulajdonságainál, a „Felhasználó jogosultságok” fülre kattintva tudjuk finomhangolni, mit is tehet és mit nem. Ha mondjuk nem adjuk meg neki a ***CD-ROM***-ok kezelésének lehetőségét, hiába illeszt be egy lemezt a meghajtóba, azt nem fogja tudni olvasni. Ha egy-egy felhasználóhoz ennél is komplexebb jogosultságrendszert szeretnénk hozzárendelni, úgy azt már grafikus felületen nem tudjuk megtenni.

Minden egyes felhasználónak saját munkaasztala, önálló beállításai, saját, egyéni könyvtárai lesznek a ***/home*** könyvtáron belül.

A felhasználókezelés részleteivel, a hozzáférések pontos beállításával nincs helyünk részletesebben foglalkozni, koncentráljunk a következőkben arra, ami a szemünk előtt van.

Látvány

A kép minősége az egyik első dolog, amit telepítés után észreveszünk. Ez vagy tökéletesen az, amire számítottunk vagy pedig javítani kell rajta (persze ha nem fizikai okai vannak a rossz képminőségnek, azaz például nem monitorunk, videokártyánk romlott el vagy esetleg túl régi az éles, nyugodt képhez). A telepítés vége felé jelölhettük be hogy, a telepítő által felismerteken kívül milyen felbontásokat tud szerintünk még monitorunk és videokártyánk együtt. Ha ezek jó értékek, most valószínűleg a megadottak közül a legnagyobb felbontást állította be az ***Ubuntu***. Ha rosszul adtuk meg, elképzelhető hogy képet sem látunk. Amennyiben csak felbontást szeretnénk váltani, a ***„Rendszer / Beállítások”*** menü alatt a „Képernyő felbontása” párbeszédablakban tehetjük meg (*2. ábra*). Itt a képfrissítési frekvenciát is megadhatjuk, ha a kép érezhetően vibrál, az arra utalhat, hogy alacsony ez az érték, próbáljuk meg nagyobbal (75 ***Hz*** vagy afelett javasolt katódsugárcsöves monitoroknál. Az ***LCD*** monitoroknál a képfrissítési érték nem játszik túl nagy szerepet). Ha ez nem elég, vagy egyáltalán nincs képünk, esetleg indulás után csak a karakteres képernyőt kapjuk vissza, bizony kézileg kell a dolgot kijavítanunk. Az ***Ubuntu*** grafikus ablakkezelője az ***XOrg*** rendszert használja, így beállításai a ***/etc/X11/xorg.conf*** fájlban találhatóak. Ha tudunk, nyissunk egy terminált. Ha nincs képünk, nyomjuk le egyszerre a ***Ctrl+F1*** billentyűket, ekkor megkapjuk a karakteres képernyőnket. Itt be kell jelentkeznünk. Ezek után írjuk be:

```
sudo nano /etc/X11/xorg.conf
```

(jelszavunkat természetesen bekéri a `sudo`). Keressük meg a ***„Section monitor”*** sort! Nekem itt a következő áll:

```
Section "Monitor"
        Identifier  "Prestigio"
        Option           "DPMS"
        HorizSync        28-64
        VertRefresh      43-60
EndSection
```

A monitor beállításai a `Section` és `EndSection` sorok között találhatóak. Az `Identifier` egy név, amit a telepítő olvasott ki a monitor lekérdezése után. Ez csupán tájékoztató adat. Az `Option` sorokból több is lehet, jelenleg a `"DPMS"` érték van beállítva. A modern monitorokra jellemző ez a `Display Power Management Signaling` tulajdonság, ami lehetővé teszi, hogy az automatikusan kikapcsoljon, amikor egy ideje nem használjuk. A `HorizSync` és `VertRefresh` értékek már misztikusabbak: az első a vízszintes, a második a függőleges frekvenciát takarja, mégpedig egy minimum és egy maximum értékek közötti tartományt. Nem szükséges ezek műszaki hátterét ismernünk, viszont monitorunk gépkönyvében biztosan szerepel ez a két adat. Nézzük meg és javítsuk ki a helyes értékekre. Miután ezzel készen vagyunk, nyomjuk meg a ***Ctrl+O*** (mentés) billentyűket, majd a ***Crtl+X*** billentyűket (kilépés). Ha működik a grafikus rendszer, most újra kell indítanunk azt például úgy, hogy kijelentkezünk, majd újra bejelentkezünk. Karakteres felületen el kell indítanunk a ***gdm***-et (a ***Gnome*** grafikus bejelentkezés kezelőjét) a következő módon:

```
sudo /etc/init.d/gdm restart
```

Ha monitorunk beállításával volt a baj, most be kell tudnunk lépni a grafikus rendszerbe, majd be tudjuk állítani a képernyő felbontását és a jó frekvenciát a fent említett módon. Ha ez sem megoldás, a `Screen` szekcióval is próbálkozhatunk. Itt lehetőség van az alapértelmezett színmélység (`DefaultDepth`) megváltoztatására (bitben). Ha ez például 24 vagy 36 értéken áll, akkor csökkentsük le 8-ra vagy 16-ra (a 8 bit 256 színt jelent, a 16-os 65536-ot, a 24-es több, mint 16 milliót). Mentsük el és próbáljuk most ki a grafikus rendszert ahogy említettem. Ennél mélyebben a képmegjelenítés hibáiba és azok megoldásába most nem megyek bele.

Behálózva

Ami a képminőség után a legégetőbb dolog szokott lenni, az a hálózat beállítása. Lehetünk belső hálózatban akár egy kis otthoni hálózatban vagy egy nagyobb, céges környezetben is, amiben lehetnek más ***Linuxot*** és ***Windows***-t használók. Kapcsolódhatunk az internetre közvetlenül modemmel vagy egy útválasztón (***router***) keresztül is. Lehet vezetéknélküli (***wireless***) és infravörös (***Irda***) kapcsolatunk is. Akár így, akár úgy, az a célunk, hogy kommunikálhassunk. Az ***Ubuntu*** a telepítés során megpróbálta felderíteni hálózati kapcsolatainkat is, de ez nem biztos hogy sikeres volt. Ahogy telepítésnél említettem, a hálózati beállításokat ***DHCP*** protokoll segítségével próbálta meg lekérdezni és ha sikerül, minden bekapcsolás után újra lekérdezi a ***DHCP*** szervert. De ha nincs ilyen szerver a hálózatban, bizony hiába próbálkozik. Ekkor kézzel kell megadnunk a legfontosabb értéket. Természetesen a ***Linuxnak*** fel kell ismernie hálózati kártyánkat vagy modemünket. Ha ez nem sikerül, a hálózati elérés is eleve kudarcra van ítélve.

6. ábra Egy megosztás beállítása

Először feltételezzük, hogy van hálózati kártyánk, amit az ***Ubuntu*** felismert és az internetet, belső hálózatot egy útválasztón (router) keresztül érjük el. Útválasztónk pedig helyesen van konfigurálva, valamint ez az útválasztó ***DHCP*** segítségével ki is osztja a hálózati erőforrásokat. Ez a legideálisabb állapotok egyike, telepítés után általában semmi gondunk sincs a hálózattal. A ***„Rendszer / Adminisztráció”*** menü alatt a ***„Hálózat”*** párbeszédablakban a *3. ábrán* lévő állapotot látjuk. Az „Alkalmazások / Rendszereszközök” menüpont alatt található ***„Hálózati segédeszközök”***-el tudjuk megtekinteni, milyen értékeket kaptunk így meg (*4. ábra*). (Az ***„Eszközök”*** fülre kattintva ***„Hálózati eszköz”***-re az ***„Ethernet interface”***-t válasszuk ki.) A képen az én beállításaim láthatóak, ezek szerint a ***10.0.0.95***-ös IP-címmel rendelkezem, az alhálózati maszk ***255.255.255.0***, a ***broadcast*** cím pedig a ***10.0.0.255***. Az útválasztási információkat pedig ugyanezen a párbeszédablakon, de a ***„Netstat”*** fület kiválasztva tekinthetjük meg.

Ha nincs ***DHCP*** szerver a hálózatban, akkor kézileg kell a hálózati beállításoknál megadni az IP-címünket, alhálózati maszkunkat és az alapértelmezett átjáró címét, sőt a ***DNS*** fülön a névkiszolgálók címeit is fel kell vennünk. Az értékeket kérdezzük meg a hálózat rendszergazdájától (ha esetleg az olvasó maga lenne a hálózat egyedüli karbantartója, akkor ismernie kell ezeket az értékeket). Debiant ismerők jobban általában szeretik karakteres felületen beállítani a hálózati kártyához rendelt értékeket. Ezt mi is megtehetjük, ha a ***/etc/network/interfaces*** fájl szerkesztjük. Igen részletes és kimerítő leírását adja a hálózat beállításának ennek a fájlnak a súgólapja (***man interfaces***), csak győzzük elolvasni. (A ***Linux*** híres jól dokumentáltságáról – az egyes kézikönyvoldalakat a `man` paranccsal érhetjük el, utána írva a megismerni kívánt parancs, állomány nevét).

Előfordulhat, hogy rendszergazdánk proxy szervert is üzemeltet, és csak ezen keresztül érhetjük el az internetet. Ennek beállítására szolgál

■ *7. ábra* A távoli asztali kapcsolat beállítása

a *„Rendszer / Beállítások”* menü alatt a *„Hálózati proxy”* párbeszédablak. Itt akár kézileg is beállíthatjuk a proxy címeket, de megadhatunk egy olyan címet is, amely automatikusan konfigurációt tesz lehetővé. A lehetőségekről tájékozódjunk hálózatunk rendszergazdájánál.

Ha modemet használunk, akkor valószínűleg vagy kapcsolt vonali internetelérésünk van (a betárcsázós forma) vagy pedig *ADSL*-, esetleg kábelmodemmel rendelkezünk. Az első esetben a már bemutatott hálózati beállításoknál a modemkapcsolatot kell megfelelően beállítanunk (telefonszám, azonosító, jelszó, modem port, egyéb beállítások) és aktiválnunk. A modem portnál általában elég a */dev/modem* eszközt megadnunk, ha ezzel nem megy, kiválaszthatjuk a megfelelő soros portot is (*Windows* felhasználóknak: a */dev/ttyS0* a *COM1*-et, a */dev/ttyS1* a *COM2*-őt jelenti és így tovább). Itt megpróbálkozhatunk az *„Automatikus felismerés”* gombbal is. A modem beállításánál alapértelmezés, hogy az legyen az alapértelmezett átjáró az internet felé és hogy az internetszolgáltatónk névkiszolgálóját alkalmazzuk (egyébként a *DNS* fülön kézzel meg kell adnunk a névkiszolgálókat).

Ha *ADSL*-ünk van és közvetlenül csatlakozunk az *ADSL* modemhez, akkor terminálban írjuk be a

```
sudo pppoeconf
```

parancsot! Ez először megkeresi az *ADSL* modemet, majd ha megtalálta, megkérdezi az eléréshez szükséges adatokat, s reméljük be is állítja az *ADSL* kapcsolathoz szükséges dolgokat.

Kábelnet esetén, ha közvetlenül csatlakozunk a kábelnet modemhez, úgy kell eljárnunk mint ahogy az elején említettem. A kábelneten található egy útválasztó, amely *DHCP*-vel osztja ki hálózati erőforrásainkat. Vezetéknélküli (wireless) hálózatokkal, valamint a hálózati kártyákkal egy későbbi részben foglalkozom részletesen. Nézzük még meg, hogyan tudjuk karakteres felületen lekérdezni legfontosabb hálózati beállításainkat! Az

```
ifconfig eth0
```

parancs kiadására hálózati kártyánk minden adatát láthatjuk, ezek közül a leglényegesebb az IP-cím és az alhálózati maszk. Az útválasztó táblázatot tudjuk lekérdezni a

```
route
```

paranccsal.

Windows hálózat

Céges környezetben gyakorta szükségünk lehet, hogy egy *Windows* szerver vagy más windowsos gépek megosztott erőforrásait érjük el (hát igen, bármennyire is szeretjük a *Linuxot*, a *Windows* egy ideig még biztosan igen elterjedt marad). Az *Ubuntu* a linuxos körökben igen népszerű *Samba* alkalmazást használja a *Windows* hálózatokban szokásos *Netbios* emulálására.

A megosztások eléréséhez a *„Helyek”* menüben található „Kapcsolódás kiszolgálóhoz” párbeszédablakot használjuk. Válasszuk ki a szolgáltatás típusát: *Windows megosztás*. Sok mindent kitölthetnénk, én azonban azt javaslom, csupán a *„Kiszolgáló:”* mezőt adjuk meg, itt egy IP-címet vagy a kiszolgáló, megosztott gép *Netbios* elnevezését és kattintsunk a *„Kapcsolódás”* gombra!

Ekkor megpróbálja elérni az illető címmel megadott számítógépet, majd megjelenik az *5. ábrán* látható ablak, ahol megadhatjuk azon a gépen érvényes azonosítónkat, jelszavunkat, a tartomány nevét, valamint hogy miként tárolja a megadott jelszót a rendszer. Ha bejelöljük, hogy *„Jelszó megjegyzése erre a munkamenetre”*, akkor amíg ki nem jelentkeznünk vagy újra nem indítjuk a számítógépet, a jelszót még egyszer nem kell beírnunk. Ha a *„Jelszó mentése a kulcstartóra”* opciót bejelöljük, akkor a bejelentkezéskor használt jelszót egy titkosított állományba, az úgynevezett *„kulcstartóra”* rakja el. Legközelebbi bejelentkezéskor pedig innen olvassa ki.

Ez akkor hasznos, ha több kiszolgálót is felveszünk, mivel a kulcstartót is védi jelszó. Ekkor csupán egy jelszót kell megjegyeznünk minden szerverhez. Ez természetesen nemcsak a windowsos megosztásokra, de bármely hálózati bejelentkezésre vonatkozik.

Ha sikerül elérni a megosztott erőforrást, annak ikonja megjelenik mind a munkaasztalon, mind a *„Helyek”* menüben, valamint *Fájlböngészőben* is, ha ott megjelenítjük a „helyek”-et az oldalsávban és úgy használhatjuk, akár bármelyik helyi mappánkat.

Bejelentkezés más gépekre, sajátunk megosztása

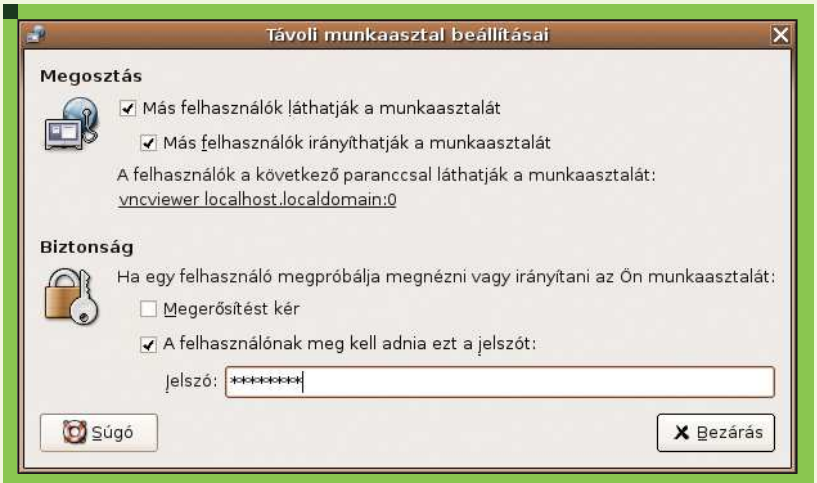
A *„Rendszer / Adminisztráció”* menüben a *„Megosztott mappák”*-kal tudjuk saját megosztásainkat adminisztrálni.

A két lehetséges megosztási mód a *Samba*, azaz *windowsos* megosztás annak unixos megvalósítása segítségével és az *NFS*, azaz a *Network File System*. Első használatkor megkérdezi, melyiket választjuk (legalább az egyiket, de lehet mind a kettőt is). A kiválasztott protokollokat automatikusan telepíti is. Ha most

nem jelöljük be az egyiket, később már csak kézileg telepíthetjük. Megosztandó mappáinkat egy – most még üres – listához kell hozzáadnunk, kiválasztva melyik – *Samba* vagy *NFS* metódus szerint osztjuk meg, és kinek engedélyezzük (kiszolgáló hozzáadása). Ez utóbbit vegyük nagyon komolyan, a hozzáférők körét mindig a lehető legszűkebbre vegyük, hogy elkerüljük az illetéktelen behatolást. Ha lehet, minden egyes engedélynél jelöljük be a *„Csak olvasható”* opciót, így kevésbé leszünk kiszolgáltatva szándékos vagy tudatlanságból eredő rongálásoknak. Az engedélyezett gépek körét négyféle módon adhatjuk meg:

- Gépek az `eth0` hálózatban (ha több hálózati kártyánk van, mindet felsorolja): ekkor azok férhetnek hozzá a megosztáshoz, akik a hálózati kártya (itt `eth0`) beállításaiban megadott, azzal egyező hálózati szegmensben vannak
- Gépnév: egy megadott nevű gép
- IP-cím: egy megadott című gép
- Hálózat: a megadott hálózaton lévő gépek. Ha ez a hálózat a `"0.0.0.0"`, akkor bárki hozzáférhet, de ezt lehetőleg ne alkalmazzuk.

Az engedélyezett gépeket egy listába vehetjük fel, így akárhány gépnevet, IP-címet, hálózatot hozzárendelhetünk egy megosztáshoz. Minden sorhoz külön megadhatjuk, csak olvasható módon férhessen hozzá vagy ha ezt nem jelöljük be, akkor teljes joggal.
A *Windowst* futtató gépek csak a *Samba* megosztásokhoz fognak hozzáférni, míg a linuxos vagy más unixos gépek (beállításuktól függően) mindkét módon képesek hozzánk csatlakozni (*6. ábra*)
Nemcsak mappákat, nyomtatókat, de akár munkaasztalt is megoszthatunk más gépekkel, illetve azok számunkra. Ez számtalan lehetőséget rejt magában, a legkomolyabb üzleti alkalmazásoktól (ahol a felhasználók bárhonnan tudják használni megszokott környezetüket) csak egyszerűen az otthoni gép eléréséig vagy lehetővé teszi akár rendszergazdák távoli adminisztrációját is. A *Windows* rendszereken elterjedt a *„Távoli munkaasztal”* (*Remote Desktop*), ahol a megfelelő jogosultságokkal rendelkező felhasználók hálózaton keresztül bejelentkezve, saját virtuális munkaasztalt kapnak. Ennek egy másik megoldása a *VNC* (*Virtual Network Computing*), amely ugyan nem olyan gyors, mint a Távoli munkaasztal, de nyílt forráskódú, szabad szoftver.



8. ábra Munkaasztalunk megosztása

Mindkettőt kiválóan elérhetjük egy *Ubuntuval*. Ehhez nyissuk meg az *„Alkalmazások / Internet”* menüben lévő *„Távoli asztali kapcsolat”* alkalmazást. Itt meg kell adnunk a távoli gép IP címét vagy nevét, majd hogy milyen protokollal kívánjuk elérni. Ha a távoli gép egy windowsos Távoli asztal szolgáltatást nyújt, akkor ez az *RDP* (az *RDP 5*-ös verzióját *Windows 2003*-as szerverrel alkalmazhatjuk). Ha a távoli gépen egy *VNC* szerver fut (lehet ez linuxos és windowsos egyaránt), akkor a *VNC* protokollt válasszuk.
Megadhatunk még felhasználónevet, jelszót, esetleges tartománynevet, saját gépnevünket, mekkora képernyőt szeretnénk látni, milyen színmélységben, a hangok átjöjjenek-e a helyi gépünkre a távoli gépről, miként kezelje a billentyűkombinációkat, és még néhány apróságot (*7. ábra*).
Saját gépünket asztalát megoszthatjuk másokkal a *„Rendszer / Beállítások”* menüben a *„Távoli munkaasztal beállításai”* párbeszédablakban (*8. ábra*). Ez tulajdonképpen egy *VNC* szervert állít be és indít el gépünkön, amennyiben bejelöljük, hogy *„Más felhasználók láthatják a munkaasztalát”*.
Ez csak megfigyelésre jó, mondjuk gyermekünk netezésének figyelemmel kísérésére, ha a távolból érdemben használni is szeretnénk a számítógépet, akkor a „Más felhasználók irányíthatják a munkaasztalát” is be kell jelölnünk. Ez a működő munkaasztal megosztására vonatkozik, tehát ebben az esetben nem új virtuális munkaasztalról van szó. A biztonság nagyon fontos, hiszen nem engedhetjük meg, hogy bárki is elérhesse engedélyünk nélkül a gépünket, ezért bejelölhetjük, hogy egy távoli felhasználó csatlakozásakor kérjen-e megerősítést, illetve hogy kelljen-e jelszó a kapcsolathoz. Én ajánlom mindkettőt megadni.
Ha idáig eljutott a kedves olvasó, nyugodtan hátradőlhet, és elégedetten elszopogathat jól működő *Ubuntuja* tiszteletére egy pohár *Villányi Chardonnay*-t. A következő számban az indítással, a kernellel, a fájlrendszerrel, a frissítésekkel, a csomagok telepítésével és az alkalmazások menüvel foglalkozunk.
Addig is a bor mellett ne feledjék az *Ubuntu* jelszavát: *„Emberség másokkal szemben!”*

Molnár Norbert
(molnar.norbert@gmail.com)
34 éves, rendszergazdaként dolgozik, 5 éve foglalkozik Linuxszal. Főként a szabad szoftverek és a számítógépes biztonság érdekli. Budapesten él feleségével és 2 éves kisfiával. Hobbija a csillagászat és a filozófia – lehetőleg jó vörösbor mellett.

Kis csevej – XChat

Minden kezdő linuxosban előbb vagy utóbb felmerül a kérdés a „nagy váltás” után: megtalálja-e kedvenc programjai linuxos megfelelőit? Jelen esetben a közkedvelt windowsos mIRC linuxos testvérével, az XChat-tel fogunk megismerkedni.

A váltás mindig rizikós, ám ezennel bátorkodom eloszlatni a kételkedők minden aggodalmát: igen, a legtöbb windowsos programnak *van* linuxos megfelelője. Néha jobb is nála, néha kevésbé, de van. Lássuk be, ez tulajdonképpen törvényszerű. Ilyen az élet. Ráadásul a *Linux* alatt futó alkalmazások puritán kinézetük ellenére is stabilabbak.

Az *XChat* határozottan kényelmes és felhasználóbarát grafikus felülettel rendelkezik, amely a *GTK+*-on alapul. Rendelkezik csaknem az összes *mIRC*-ben megszokott szolgáltatással, sőt a „hardcore” linuxosok által használt konzolos *IRC*-ügyfelekkel (*BitchX*, *ircii*, *irssi*, *epic4*) is felveszi a versenyt.

Az *XChat* gyakorlatilag minden *Linux* terjesztésnek része.

Ha *GNOME* ablakkezelőt használunk, akkor biztosra vehetjük, hogy a menüben ott lesz az *XChat* pont is. Természetesen forrásból is telepíthető, de a kezdő felhasználóknak inkább a bináris csomagot javaslom. *SuSE* alatt a *Yast*, *UHU* alatt az *apt-get* segítségével telepíthetjük, hogy csak a két legnépszerűbb terjesztést említsem.

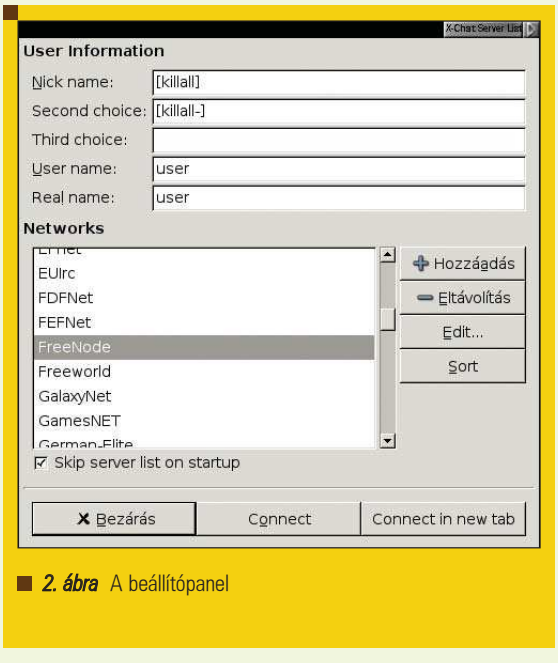
Az *XChat* amúgy nem csak *Linux* alatt működik. Támogatja a *FreeBSD*, *NetBSD*, *OpenBSD*, *Solaris*, *AIX*, *IRIX*, *DEC/Compaq Tru64 UNIX*, *HPUX 10.20* és *11*, *MacOS X* valamint a *Windows 98/ME/NT/2000/XP* rendszereket is.

Az *XChat* jelenlegi stabil verziója a *2.4.4.*, amit a ➲ *http://xchat.org/donwload/* oldalról tölthetünk le akár forráskód, akár előrefordított csomag formájában. Csomag a következő terjesztésekhez létezik: *Fedora*, *SuSE*, *Mandriva*, *Gentoo*, *Slackware*. Szintén innen érhető el a *MacOS X*-re, a *FreeBSD*-re és a *Windowsra* készült változat is.

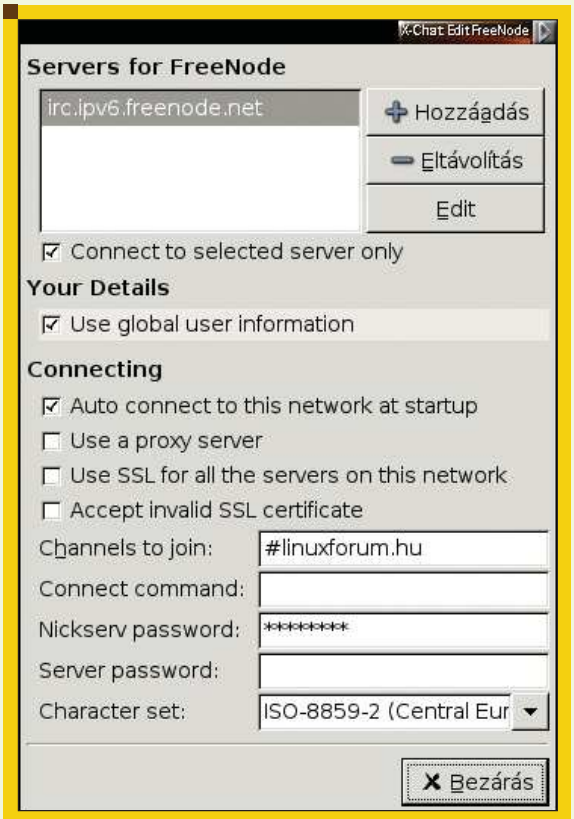
A *dpkg* alapú terjesztések (ilyen például a *Debian* és az *UHU*) felhasználói se keseredjenek el. Mindkét disztribúció tükörszerverein elérhető az *XChat*, és egy egyszerű

```
apt-get install
➥xchat
```

1. ábra XChat használat közben



2. ábra A beállítópanel



3. ábra A kapcsolat...

paranccsal telepíthető. (Tulajdonképpen a legtöbb nagyobb disztribúcióban ez a kliens az alapértelmezett *IRC* alkalmazás, vagyis csak kivételes esetben lesz szükségünk telepítésre.)

Az XChat használata

Amint elindítottuk a programot, létre fog jönni a *home* könyvtárunkban egy rejtett könyvtár (*~/.xchat2/*) amely az *XChat* konfigurációs fájljait tartalmazza. (*Linux* alatt a rejtett fájlok és könyvtárak neve ponttal kezdődik, ettől lesznek „rejtettek”. Ahhoz, hogy ezt a könyvtárt lássuk, be kell állítanunk kedvenc fájlkezelőnkben a rejtett könyvtárak/fájlok megjelenítését. Szerencsére e könyvtár tartalmának bármiféle módosítására csak különleges esetben lehet szükség, hiszen a grafikus felületen szinte minden beállítás elvégezhető. Ez pedig egy kezdő linuxos számára igen fontos szempont lehet.)
Lássuk tehát a lényeget, vagyis hogy miként tudunk minél gyorsabban kapcsolódni egy *IRC* kiszolgálóhoz.
Első lépésben meg kell adnunk a következő adatainkat: becenév (nicknév), felhasználónév, valódi név.
El kell döntenünk, hogy melyik kiszolgálóhoz szeretnénk csatlakozni.
A beállítópanel eleve felkínál egy bőséges listát, tehát van miből válogatni. És ennyi.
A *Connect (Csatlakozás)* gombra kattintva a program máris csatlakozik a kiválasztott helyhez.

Lássuk mindezt a gyakorlatban!

Tegyük fel, hogy *UHU Linuxot* használunk és még életükben nem láttak *IRC* ügyfelet. Szeretnénk segítséget kérni az *IRCNet*-en lévő *#uhulinux* csatornán rendszerünk finomhangolásához.
Az imént felsorolt adatok beállítása után válasszuk szerverként az *IRCNet*-et.
Célszerű egy hozzánk topológiailag közel eső kiszolgálóhoz, vagyis egy magyar *IRCNet* szerverhez csatlakozni. Ehhez fel kell vennünk legalább egyet a listába.
A kiszolgálók listájában az *IRCNet* szervercsoportot kell szerkesztünk (lásd a képet). A *newserver* feliratú sor a *Hozzáadás* gombra kattintva jelenik meg. Itt kell megadnunk az új elem paramétereit. A végén egyszerűen zárjuk be az ablakot (erre a cikk végén kitérek majd.

Miután csatlakoztunk a szerverhez, a

```
/join #uhulinux
```

paranccsal máris beléphetünk az *#uhulinux.hu* csatornára. Aki nem szeret gépelni, az használhatja az ábrán is látható *„channels to join”* mezőt is.

Az XChat testreszabása

Általában is igaz, hogy az *IRC*-ügyfelek testre szabhatók, de az *XChat* e tekintetben majdnem mindegyiken túltesz.
Beállíthatunk benne átlátszóságot (lásd a képet) időbélyegek használatát, naplózást, és számos egyéb szolgáltatást. Külön gombokat is létrehozhatunk a gyakori parancsoknak, így egyetlen kattintással helyettesíthetünk egy begépelendő sort.
A beállításokat a *Settings/Preferences* menüben találjuk, ráadásul saját *Perl* nyelvű szkriptekkel is gazdagíthatjuk a szolgáltatások tárházát.

Apagyi György, (killall)
(user@killall.eof.hu,
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

A magyar IRCNet kiszolgálók nevei:
hub.irc.hu
extra.irc.hu
elte.irc.hu
irc.atw.hu

Magyar disztribúciófüggő/független linuxos irc-csatornák, ahol segítséget kaphatsz (zárójelben a szer ver neve):
#debian.hu (IRCNnet)
#linux.hu (IRCNet)
#debian.hu (FreeNode; irc.freenode.net)
#gentoo.hu (IRCNet)
#slackware.hu (IRCNet)
#uhulinux (IRCNet)
#linuxforum.hu (FreeNode)

IRC dokumentációk (FAQ, parancsok, tudnivalók):
http://irc.lap.hu
http://irc.hu

Hivatalos XChat honlap:
http://xchat.org

A linuxos biztonság jövője

Az, hogy a Linux más rendszereknél biztonságosabb is lehet, nem jelenti azt, hogy a saját Linuxunk is az. Vajon a fejlesztők és a terjesztések készítői hogyan segíthetik a jövőben a rendszergazdákat?

Ki gondolta volna, hogy már öt éve írok rendszeresen biztonsági témájú cikkeket? És micsoda eseménydús öt év volt ez! Láthattuk, hogy a legnagyobb korábbi vetélytársai felkarolják a ***Linuxot***, amely fokozatosan utat talált az asztali gépek felé.
A ***Linux*** biztonsága terén szintén komoly fejlődés tanúi lehettünk. A ***Linux*** tűzfalszolgáltatásai olyan magas szintre fejlődtek, hogy mára számos beágyazott tűzfalkészülék alapjául szolgálnak, a biztonsággal távolabbi kapcsolatban álló eszközök megszámlálhatatlan sokaságát nem is említve. A ***Linux*** elképesztő mennyiségű biztonsági eszköz támogatására képes, és a biztonsági auditorok és tanácsadók kedvenc eszközévé vált. Külön büszkeségre ad okot, hogy a ***Linux*** adta több kivételesen biztonságos, ***szerep alapú hozzáférés-vezérlést (ultra-secure role-based access control, RBAC)*** alkalmazó operációs rendszer alapját, mint például az ***NSA SELinuxa***.
De vajon milyen lesz a linuxos biztonság jövője? A jelen és a múlt linuxos biztonsági kérdéseiről sokat témáztunk, de a jövőről nem sok szó esett, leszámítva a ***Richard Thieme***-mel készített interjút. Ebben a hónapban szeretnék egy kicsit elrévedezni, és kifejteni, hogy véleményem szerint merre tart a linuxos biztonság, illetve merre ***kellene*** tartania.

Mi a baj a jelenlegi állapottal?

Az utóbbi időben egyre többen ismerik fel, hogy egy átlagos linuxos rendszer nem sokkal biztonságosabb egy átlagos ***Microsoft Windows*** rendszernél. Mielőtt méltatlankodó levelek árasztanák el a postaládámat, hadd fejtsem ki, mire gondolok. Először is, személy szerint úgy vélem, hogy a ***Linuxot*** biztonságosabbá lehet tenni, mint a ***Windowst***, és ezt a véleményemet e helyütt már többször megfogalmaztam az elmúlt évek során. A felhasználók egyszerűen nagyobb hatalmat kapnak linuxos rendszerük működésének ellenőrzéséhez, mint egy azzal egyenrangú windowsos rendszer esetében kapnának.
A baj az, hogy a ***Linux*** felhasználók, hasonlóan a ***Windows*** használókhoz, energiájuk egyre nagyobb hányadát arra fordítják, hogy gépük képes legyen érdemi feladatait ellátni, és túlságosan nagy bizalommal viseltetnek rendszerük beépített, alapértelmezett biztonsági beállításaiba. Az is vitathatatlan, hogy amikor egy-egy programhiba napfényre kerül (márpedig az ilyen esetek elkerülhetetlenek), akkor hatása jóval kiterjedtebb, mint amekkora a megfelelő óvintézkedések megtételével lehetett volna.
Ha például ***BIND v9*** névszolgáltatást akarok futtatni, belekerül némi munkámba és kutakodásomba, mire működni kezd. A ***BIND chroot jailben*** való futtatása és a kiszolgáló fájlrendszerének a ***named*** folyamat számára elérhető részének korlátozása ennél is több munkát igényel. Éppen ezért a ***BIND*** használóinak jelentős része ***nem chroot jailben*** futtatja a ***BIND***-et. Ha ismertté válik a ***BIND*** valamilyen sebezhetősége, akkor a ***BIND***-ot használók többsége jóval több kellemetlenséggel kénytelen szembesülni, mintha megküzdött volna a ***chrootos*** bebörtönzéssel. Lehet, hogy végül semmivel sem járnak jobban, mintha egy microsoftos, a ***BIND***-hez képest akár szerényebb tudású névkiszolgálót futtattak volna.

Röviden szólva, azt akarom kifejezni, hogy a *Linux* biztonsági szolgáltatásainak jelentős részét egyszerűen nem veszik igénybe a felhasználók. A végeredmény – legalábbis profi behatoláspróbákat végző barátaim szerint – az, hogy egy átlagos *Red Hat Enterprise* rendszerbe nem sokkal nehezebb betörni, mint egy átlagos *Windows 2003 Server* rendszerbe.

Szerencsétlen végkifejlet, és talán egy kicsit meglepő is. Kódbázisának teljes nyíltsága ellenére a *Linux* továbbra is ugyanolyan programhibákat tartalmaz, mint a *Windows*, nagyságrendileg hasonló mennyiségben, és ezek felbukkanási gyakorisága is közel áll egymáshoz. Jobban belegondolva, miért is lenne ez másként?

A *Windowshoz* hasonlóan a *Linux* is egy elképesztően összetett, több száz ember munkája nyomán létrejött kódhalmazt jelent. Minél több a kód, annál több a hiba is, nemde?

Nemrég készített velem egy interjút a *SearchSecurity.com*, a téma egy a *Microsoft* támogatásával a *Security Innovation, Inc* által készített tanulmány volt. A tanulmány végső következtetése az volt, hogy a *Windows* biztonságosabb a *Linuxnál*. Véleményüket főként a biztonsági hibák megjelenésének gyakoriságára és a foltok megjelenéséig átlagosan eltelt idő hosszára alapozták.

Azt gondolom, korrekt kritikát adtam a tanulmányról, amely a biztonságnak kizárólag ezeket a könnyen számszerűsíthető értékeit vette figyelembe, a *Linux* más a biztonság terén jelentkező előnyeiről, mint a testreszabhatóság és a bővebb szoftverválaszték, egyszerűen elfeledkezett. Fogalmazhatnék úgy is, a tanulmány inkább az alapértelmezett telepítésekre volt érvényes, és nem vette figyelembe, hogy az egyes operációs rendszerek mennyi lehetőséget adnak használóiknak a biztonság fokozására.

Minél többet gondolkodom ezen, annál tisztábban fogalmazódik meg bennem az a gondolat, hogy az adott rendszerben rejlő biztonsági lehetőségek mit sem érnek, amíg a rendszert futtató gépek nem használják ki őket. Itt nem kizárólag a végfelhasználókra gondolok, nem a rendszergazdákat szeretném bírálni. Később még kifejtem részletesebben is, de úgy vélem, a *Linux* fejlesztőinek és terjesztőinek továbbra mindent meg kell tenniük annak érdekében, hogy a biztonsági szolgáltatások mindent átszőjenek, átlátszók, könnyen beállíthatók és egyszerűen használhatók legyenek.

Egyébként, ha már összehasonlítom a *Linuxot* és a *Windowst*, meg kell említenem, hogy a *Windows* is túlságosan sok a használói által parlagon hagyott biztonsági szolgáltatással rendelkezik.

Rendben, alapállapotában a *Linux* és a *Windows* egyaránt rosszabb biztonságot ad, mint amire képes volna, és mindkettő megnyerhetetlen versenyfutásba kényszerít bennünket a programhibák és a biztonsági foltok között. Mivel kell még megküzdenünk?

Nos, mindkét operációs rendszer a meglehetősen primitív önkényes (vagy önkéntes) hozzáférés-vezérlési modellt használja, ahol a biztonsági beállítások egész osztályainak használata, illetve ezek viselkedése kiegészítő, elhagyható jellegű. Ebben a modellben van egy szuperfelhasználói fiók – *Linux* alatt a root, *Windows* alatt a rendszergazda –, mely istenségszerű hatalommal rendelkezik a teljes rendszer felett, ide értve más felhasználók fájljait is. Mindkét operációs rendszerben csoporttagságokkal lehet különféle hozzáférési szinteket létrehozni, más szóval a szuperfelhasználói jogokat átadni.

A gyakorlatban a legtöbb rendszerre szuperfelhasználóként kell bejelentkeznünk, illetve ideiglenesen az ő szerepét kell felvennünk, ha valamilyen fontosabb műveletet akarunk elvégezni.

Ha tehát egy linuxos vagy windowsos rendszer felett korlátlan hatalmat akarunk szerezni, akkor csak egy a szuperfelhasználó jogaival futó folyamat biztonságát kell megtörnünk. Hogyan? A fontosabb démonokat jogosultságok nélküli felhasználók nevében futtatjuk, így hiába találnak hibákat ezekben a démonokban, a rendszer biztonsága nem sérülhet. Vagy mégis? Nem, közvetlenül valóban nem, de az egyéb szoftverekben található hibák miatt egy alacsonyabb jogosultságokkal futó program is szert tehet szuperfelhasználói jogokra. Tegyük fel például, hogy van egy *Apache* alapú webkiszolgálónk, és egy napon egy támadónak sikerül kihasználnia egy még befoltozatlan, az *Apache*-ban található puffertúlcsordulásos hibát, amivel sikerül héjhozzáférést szereznie a kiszolgálón. Ezen a ponton a támadó a webfelhasználó jogaival élhet, ugyanis az Apache ennek a felhasználónak a jogosultságaival fut. Tegyük fel, hogy ugyanezen a rendszeren a rendszermagnak is van egy befoltozatlan hibája, ami lehetővé teszi a helyi jogosultságok megemelését.

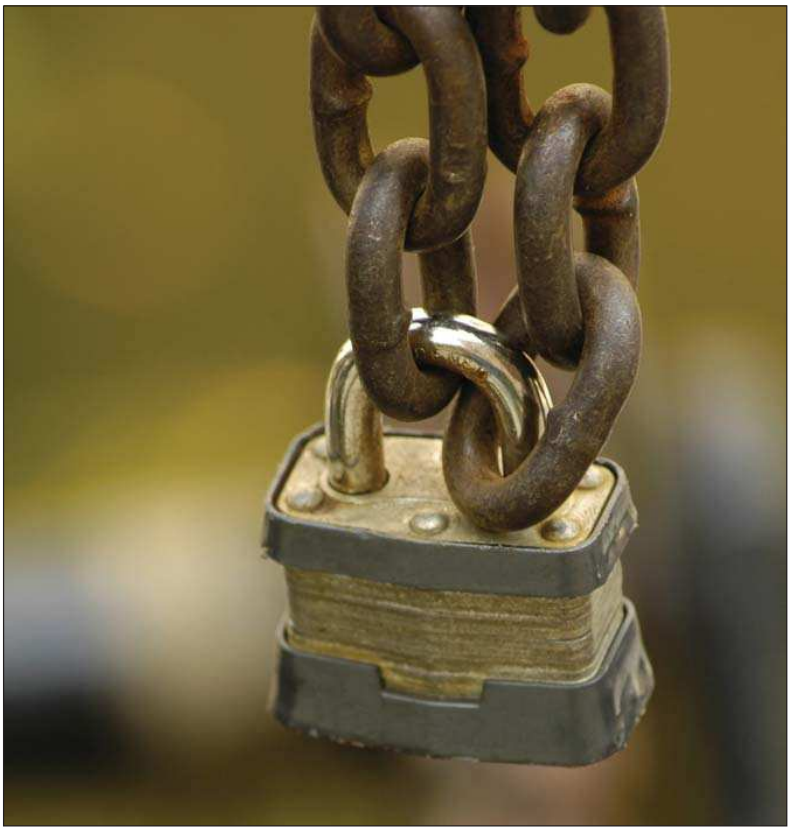
Lehet, hogy a rendszergazda tud is a hibáról, de nem foglalkozott a javításával, hiszen csak helyből használható ki, és rajta kívül senki nem rendelkezik héjhozzáféréssel a rendszerhez – és persze ki akarná újraindítani a gépet csak a rendszermag megfoltozása miatt? Csakhogy mostanra a távoli támadó helyi héjhozzáférést szerzett, sikeresen kihasználja a rendszermag sebezhetőségét, és máris rootként garázdálkodhat a gépen! Nem is gondolnánk, milyen gyakori ez a forgatókönyv, és kiválóan szemlélteti, hogy a hibák önmagukban is sok fejtörést okoznak, de a root mindenhatóságára épülő biztonsági modellel kombinálva a bajok megsokszorozódnak.

Nagyjából ez jellemzi a linuxos biztonság jelenlegi állapotát.

A *Linux* biztonságossá tételéhez komoly erőfeszítéseket kell tennünk: ki kell használnunk az alapesetben legtöbbször nem engedélyezett, időnként meglehetősen bonyolult biztonsági szolgáltatásokban rejlő lehetőségeket, és minden rendelkezésünkre álló biztonsági foltot telepítenünk kell – mindezt a Linux egyszerű biztonsági modellje által biztosított keretek között. De ne aggódjunk, jó társaságba keveredtünk: napjaink operációs rendszerei ugyanezekkel a korlátokkal és kihívásokkal küzdenek.

Kötelező hozzáférés-vezérlés

Már utaltam rá, hogy a *Linux*, az általános *UNIX* és a *Windows* rendszerekben látható hozzáférés-vezérlés és a fájleléréseket szabályozó megoldás önkényes, ami biztonsági modellként gyengének számít. Mi a helyzet az *SELinuxszal*? Az *RBAC*-k és a *típus-*



szabályok (type enforcement, TE) vajon jó példát adnak a kötelező hozzáférés-vezérlésre? Igen. Tartok azonban attól, hogy a linuxos biztonságnak nem ez a jövője, pontosan azért, amiért az *SELinux* sem tudott számottevő szerepet szerezni e téren.

Az *RBAC*-k gondosan körülírt szerepek alapján korlátozzák a felhasználók viselkedését és a rendszer erőforrásaihoz való hozzáférését. Ezek a szerepek hasonlóak a hagyományos *UNIX* csoportokhoz, bár jóval messzebbre hatnak. A típusszabályok hasonlóan, előre megadott műveletartományok alapján korlátozzák a folyamatok által végrehajtható műveletek körét. Az *RBAC*-k és a *TE*-k használatával végül elválasztott silók (én legalábbis így hívom őket) jönnek létre, ezekben tevékenykednek a felhasználók és a folyamatok, és a silók között csak korlátozott párbeszédet engedünk meg.

Ez egy elegáns és hatékony biztonsági modell, csakhogy az emberek túlnyomó része számára az *RBAC*-k, a *TE*-k és az egyéb hozzáférés-vezérlési módszerek túlságosan bonyolultak, és túlságosan sok felügyeleti többletmunkát okoznak. Sokak számára ezek a tényezők az *SELinuxot* és a hozzá hasonló operációs rendszereket elismert, de elkerült megoldásokká teszik – olyan operációs rendszerek ezek, melyek bizonyos csoportok igényeinek megfelelnek, széles körben azonban nem tudnak elterjedni. Bár magam is csodálom az *SELinux* biztonsági architektúráját, sőt, rajongok az *RBAC*-k használatának ötletéért, nem hiszek abban, hogy az általuk megvalósított kötelező hozzáférés-vezérlésnek esélye volna a linuxos biztonság megújítására.

Hiperfelhasználók és virtuális gépek

Ha az *RBAC*-k és a *TE*-k használata az operációs rendszer szintjén túlságosan kényelmetlennek bizonyul a betörések hatókörének korlátozására, a hiperfelhasználók (hipervizorok) és a *virtuális gépek (virtual machine, VM)* egy magasabb szinten talán megoldást kínálnak. A virtuális gépeket már két területről is ismerhetjük: egyrészt a virtuális futtatási környezetek révén, ilyet használnak például a Java programok, másrészt a virtuális gépek által, ilyen a *VMware*, a *plex86* és a *VirtualPC*; ezek virtuális hardverkörnyezetben teljes operációs rendszerek futtatására képesek.

A *Java* virtuális gépet figyelemre méltó biztonság szolgáltatásokkal ruházták fel, ezek közül a legfontosabb talán a *Java* homokozó. Általában véve a Java biztonsága abból fakad, hogy a Java kisalkalmazások a nyers, valós rendszererőforrásoktól elkülönítve futnak, minden művelet a Java virtuális gép közvetítői segítségével történik. Ez is kiváló biztonsági modell, és a programozók és a végfelhasználók számára egyaránt könnyen használható. A *Java* – több okból – napjainkra széles körben elterjedt.

A virtuális számítógépek eggyel tovább viszik ezt az elgondolást, és nemcsak programok, de teljes operációs rendszerek között is képesek közvetíteni. A biztonsági alrendszer esetükben még kevésbé kiforrott, mint a Java virtuális gép esetében. A biztonság kezelését túlnyomó részt a virtuális környezetben futó vendég operációs rendszerre bízzák. Egy *VMware*-ben futó *SUSE Linux* tehát se többé, se kevésbé nem biztonságos, mint egy a saját hardverén futó.

A hiperfelhasználós technológia az azonos hardveren futó virtuális gépek elkülönítésének és a párbeszédek korlátozásának gondjára kínál megoldást, illetve általa megelőzhető, hogy az egyik virtuális gép biztonságának sérülése a többi gépet is érintse. Az *IBM sHype* névvel már készített egy hiperfelhasználós biztonsági rendszert. Létezik egy nyílt forrású hiperfelhasználókra és virtuális gépekre alapozó fejlesztés is, ez *Xen* néven fut.

Bár a hiperfelhasználók alkalmazásának fő oka az, hogy az azonos fizikai gépen futó virtuális gépek – például a megosztott hardveres erőforrások kisajátításával – ne zavarhassák meg egymás működését, az önmagában sem rossz gondolat, hogy kellene valamilyen magasabb szintű, a rendszerek felügyeletére képes intelligenciát biztosítani. Ez a hagyományos *behatolásészlelő rendszerek (intrusion detection system, IDS)* működését legalábbis javítaná, de lehetséges, hogy át is venné azok szerepét a rendszer biztonságának megsértésére irányuló kísérletek felismerése és kibontakozásának megakadályozása terén.

A kötelező hozzáférés-vezérlés és

a hiperfelhasználókra és virtuális gépekre alapuló megoldások nem zárják ki egymás alkalmazását. Ugyanakkor az én meglátásom – nem kis részben barátom és biztonsági elemző munkatársam, ***Tony Stieber*** hatására – az, hogy a hiperfelhasználós megoldásnak jóval nagyobb lehetősége lesz a linuxos biztonság jövőjének alakítására, mint a kötelező hozzáférés-vezérlésnek. Természetesen együttes használatuknak sincs akadálya. Képzeljünk el egy nagyméretű, nagy teljesítményű rendszert, mely nagyszámú virtuális gépet futtat, és ezeket egy hiperfelhasználó felügyeli. Az egyik virtuális gépen egy általános célú operációs rendszer – például ***Linux*** – mint webkiszolgáló üzemel. A másik virtuális gép érzékeny adatokat kezelő adatbázisként szolgál, erre valamilyen kötelező hozzáférés-vezérlést alkalmazó operációs rendszer kerülhet, például ***SELinux***. Mindkét virtuális gép részesül a hiperfelhasználós modell által biztosított előnyökből, miközben az ***SELinux*** a maga játékterén további védelmet is nyújt.

Rendellenesség alapú behatolásészlelés és vírusvédelem

A kötelező hozzáférés-vezérlés és a hiperfelhasználós megoldás mellett egy további, már létező, de a jövőben valószínűleg a jelenleginél nagyobb szerepre érdemes technológia a rendellenesség alapú behatolásészlelés.
A rendellenesség alapú ***IDS***-ek ötlete egyszerű: meg kell adni, hogy mi jellemzi a rendszer és a hálózat normál működését, és váratlan vagy rendellenes viselkedés észlelésekor riasztást kell küldeni.
Ha maga az ötlet egyszerű, a megvalósítás pedig már megvan, akkor miért nem használjuk szélesebb körben is? Azért, mert nem olyan egyszerű és kiforrott, mint az aláírás-egyeztetés. Az aláírás alapú ***IDS***-eket már ismerjük: rendelkeznek a támadások aláírásainak adatbázisával, és összevetik ennek tartalmával a hálózati csomagokat, illetve a csomagok sorozatait. Ha egy csomag megegyezik egy az adatbázisban szereplővel, akkor támadás részeként kezeljük, és riasztást bocsátunk ki.
Ennek a megoldásnak az előnye, hogy könnyen használható, és kevés hamis riasztást ad ki. Az aláírás alapú rendszerek akkor mondanak csődöt, ha újfajta támadás éri a rendszert, esetleg a támadás túlságosan bonyolult ahhoz, hogy megfelelő aláírás tartozhasson hozzá az adatbázisban, és ezért nem sikerül felismerni.
A rendellenesség alapú megoldásnál ezzel szemben minden olyan új támadás felismerhető, amely kellően megkülönböztethető a normál működéstől. Ennek ára az, hogy az ***IDS*** rendszergazdájának be kell tanítania a rendszert (ennek során mutatja meg neki, hogy mi számít normál üzemnek), és a betanítást rendszeres időközönként meg is kell ismételni. Amíg a viszonyítási alapot nem sikerül finomhangolni, addig viszonylag gyakori hamis riasztásokra kell készülni.
Még 1999-ben volt szerencsém részt venni ***Marcus Ranum*** egy előadásán, ahol a rendellenességek felismerését a behatolásészlelő rendszerek jövőjeként jellemezte. Mondanom sem kell, ez a jövő még nem érkezett el, bár már vannak ilyen termékeket kínáló gyártók, például a ***Lancope*** és az ***Arbor Networks***. Remélem, előbb-utóbb valaki kitalálja, hogyan lehet ezt a feladatot valamilyen egyszerűbb módon elvégezni, a jelenlegieknél egyszerűbb és könnyebben használható rendszereket alkotva. Úgy látom, ezzel olyasfajta hálózati hiperfelhasználó jönne létre, aki hasonló intelligenciával ruházná fel a hálózatokat – álljanak azok akár virtuális, akár valós gépekből –, mint amilyet a virtuális számítógépeknek is biztosít.
Megjegyzem, a víruskeresők legalább annyira ki tudnák használni a rendellenességek felismerését, mint a behatolásészlelő rendszerek. Mi sem igazolhatja ezt jobban, mint az a tény, hogy azok a szervezetek, amelyek korszerű, de kizárólag vírusaláírásokra támaszkodó víruskereső programokat használnak, rendszeresen áldozatául esnek a nagyobb vírus-, trójai- és féregkitöréseknek. Egyértelmű, hogy a jelenlegi, aláírásokat alkalmazó víruskereső eszközök hatékonysága elégtelen.

Összefoglalás és búcsú

Nagyjából ezek a ***Linux*** biztonságával kapcsolatos gondolataim. Amíg nem sikerül előbbre lépni, használjuk tovább a meglévő, a jelen rovatban is sokat tárgyalt eszközöket: a tűzfalakat, a víruskeresőket, az önműködő foltozó és programfrissítő eszközöket, a virtuális magánhálózatokat és az egyedi biztonsági alkalmazásokat, mint a chroot jailek és a naplók.
Ezzel én búcsút intek, nemcsak erre a hónapra, hanem bizonytalan időre. Legalább egy kis ideig más dolgokra kell fordítanom a figyelmemet, és a rovatban új hangokat is hagynom kell megszólalni. Biztonsági szerkesztőként továbbra is ott leszek a háttérben, és ebben a szerepben a jövőben is segíteni fogom a ***Linux Journalt*** abban, hogy a biztonság témakörében kimagasló színvonalú tartalmat biztosíthasson. Remélem, hogy alkalmanként további írásokat is tudok majd készíteni, a rovat kizárólagos szerzőjeként azonban ez az utolsó cikkem. Köszönöm mindenkinek az elmúlt öt év támogatását, bátorítását és segítségét: írásaimban soha nem véthettem úgy hibát, hogy valaki ki ne javítson – mindig baráti hangnemben.
Nagyszerű öt év volt, hálás vagyok a lap fantasztikus gárdájának és olvasóinak, hogy részem lehetett benne!

Linux Journal 2005. 136. szám

Mick Bauer
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban. Az O'Reilly & Associates nemrég, 2005 januárjában jelentette meg Linux Server Security című könyvének második kiadását. Mick „indusztriális polka" zeneszerző, ám van annyira jó ízlésű, hogy ne nagyon játssza el saját szerzeményeit.

KAPCSOLÓDÓ CÍMEK

sHype ➲
www.research.ibm.com/secure_systems_department/projects/hypervisor

Xen ➲
www.cl.cam.ac.uk/Research/SRG/netos/xen

Hálózatok (21. rész) – Az Ipv6

Az IP protokoll ma használt változata, az IPv4 felett eljárt az idő. Előbb-utóbb elfogynak a még ki nem osztott IP címek, és egyre kevésbé tud megfelelni a 21. század igényeinek. Hamarosan eljön tehát az IPv6 kora.

Már a 90-es évek elején, az internet robbanásszerű terjedésének kezdetekor látható volt, hogy kevés az a 32 bit, amelyen az *IP* címeket tárolják. Ugyan a *CIDR* bevezetésével (lást előző rész) egy kis idő nyerhető, de előbb vagy utóbb el fognak fogyni a még szabadon kiosztható *IP* címek.

Az internetbe kapcsolt gépek száma még ma is nagy léptékben növekszik, és ez várhatóan a jövőben sem fog változni. Viszont az új gépek nagy része valószínűleg nem újabb számítógépek bekötését fogja jelenteni (habár az Interneten jelenlévő számítógépek száma továbbra is exponenciálisan emelkedni fog). Az internet ma már új iparágak célpontja, mint például a telekommunikáció vagy a szórakoztató ipar, amelyek hihetetlenül nagyszámú eszközt szeretnének az internethez kapcsolni. Gondoljunk például az interneten keresztüli telefonálásra, vagy olyan televíziókészülékre, amely egyben internetes gépként is működik. Ezeknek az új eszközök bekötéséhez nem áll elegendő *IP* cím rendelkezésre.

A túl kicsi címtéren kívül más problémák is vannak az *IPv4*-el. Először is nem foglalkozik kellőképp a szolgálat típusával, minden csomagot ugyanúgy kezel, tartalmazzon az akár egy e-mailt, vagy egy valósidejű médiafolyamot.

Ezenkívül nem tartalmaz semmiféle hitelesítési és titkosítási szolgáltatást. Egy gép sosem lehet biztos abban, hogy a csomag tényleg attól érkezette, aki feladóként szerepel, ráadásul az átmenő csomagok tartalmába az néz bele, aki csak akar. Ha ez ellen tenni szeretnénk valamit, akkor azt csak alkalmazás szinten tehetjük meg, azaz olyan programokat kell használnunk, amelyek saját maguk gondoskodnak az átvitel titkosságáról.

Szükség volt még az *IPv4* csomagok fejlécének egyszerűsítésére is. A bonyolult, sok mezőből álló fejléc feldolgozása komoly teljesítményt igényel, így az útválasztóknak tovább tart feldolgozniuk egy-egy csomagot.

Világos tehát, hogy szükség volt egy új protokollra, amelynél nem állnak fent ezek a problémák. Ezért már 1990-ben elkezdtek dolgozni az új *IP* protokollon, amelynek eredménye az először 1993-ban publikált ***SIPP (Simple Internet Protocol Plus)*** lett, amit ma már csak ***IPv6*** néven emlegetnek.

(A logika azt diktálná, hogy az *IPv4* következő generációjának *IPv5* legyen a neve, de ez a név már sajnos akkor foglalt volt, egy kísérleti folyamalapú protokoll birtokolta). Minden bizonnyal ez a protokoll lesz az, amely egy nap kiváltja majd az *IPv4*-et (habár egy a váltás a gépek nagy száma miatt nem egyik napról a másikra fog bekövetkezni), ezért mindenképp érdemes közelebbről is megismerkednünk vele.

Prefix	Use
0000 0000	Reserved
0000 0001	Unassigned
0000 001	Reserved for NSAP Allocation
0000 010	Reserved for IPX Allocation
0000 011	Unassigned
0000 1	Unassigned
0001	Unassigned
001	Unassigned
010	Provider-Based Unicast Address
011	Unassigned
100	Reserved for Geographic-Based Unicast Addresses
101	Unassigned
110	Unassigned
1110	Unassigned
1111 0	Unassigned
1111 10	Unassigned
1111 110	Unassigned
1111 1110 0	Unassigned
1111 1110 10	Link Local Use Addresses
1111 1110 11	Site Local Use Addresses
1111 1111	Multicast Addresses

1. ábra Az IPv6-os címtartomány felosztása

Az IPv6 legfontosabb tulajdonságai

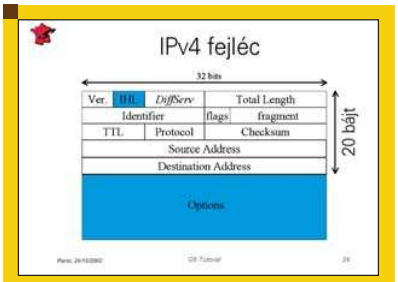
Először is, az ***IPv6*** 128 bites címekkel dolgozik. Ez elegendő ahhoz, hogy jó időre, hacsak nem örökre megszabaduljunk a kiosztható címek hiányából fakadó problémáktól.
Az ***IPv6*** fejléce egyszerűsödött. Míg az ***IPv4*** csomagok fejléce 13 mezőt tartalmaz, addig az ***IPv6*** fejléc csak 7-et. A legtöbb mező opcionális lett, azaz csak akkor szerepel a csomagban, ha tényleg szükség van rá.
Ez a megoldás elősegíti a többletterhelés csökkentését, mivel az útválasztók könnyen átugorhatják azokat az információkat, amelyek nem érdeklik őket. Az ***IPv6*** fejléc csupán a duplája az ***IPv4*** fejlécnek, ami nem mondható rossznak, ha figyelembe vesszük azt, hogy a forrás- és a célcím mérete megnégyszereződött.
Az ***IPv6*** az elődjénél sokkal jobban odafigyel arra, hogy mit szállít a csomag, azaz mi a szolgálat típusa. Igaz, erre 8 bit az ***IPv4***-ben is rendelkezésre állt, manapság azonban ez nem tűnik valami soknak, a jövőben meg biztosan nem lesz elegendő.
A legutolsó, ám egyáltalán nem elhanyagolható újítás a biztonság terén történő előrelépés. Végre a hálózati réteg szintjén is lehetőség van a hitelesítésre és a titkosításra.

Címzés az IPv6-ban

Mint már említettük, az ***IPv6*** 32 bites címek helyett 16 bájtos, azaz 128 bites címeket használ. 16 bájton rendkívül sok cím ábrázolható, olyan sok, hogy gyakorlatilag soha többé nem kell szembesülnünk a szabad címek hiányával. Az sem jelent problémát, hogy a címek elég gazdaságtalanul, hierarchikus módon kerülnek kiosztásra. Még így is a ***Föld*** minden négyzetméterére több ezer cím jut.
Az ***IPv6*** címeket különböző kategóriákba soroljuk az első bájtjuk, az úgynevezett ***format prefixük*** alapján.
Az *1. ábrán* látható táblázat mutatja, hogy milyen csoportokra is bontjuk a 16 bájtos címtartományt. (Fontos megjegyezni, hogy a címtérnek körülbelül csak a 16%-a van lefoglalva, a többi tartalékként, későbbi kihasználásra várnak).
Az ***IPv6*** két címkiosztási stratégiát definiál, és mindkét módszer számára fenntart egy-egy címtartományt. Ezek közül az elsők a 010-val

2. ábra Az IPv6 csomagok fejléce

kezdődő úgynevezett ***Provider based unicast address (szolgáltató alapú egyesküldésű címek)***. Ez a kiosztás némileg hasonlít arra, ahogy a telefontársaságok kiosztják ügyfelei között a telefonszámokat.
Az összes vállalat rendelkezik a telefonszámok egy bizonyos tartományával, amelyeket hozzárendel az előfizetők vonalaihoz. A telefonszámok első pár számjegye azonosítja a szolgáltatót, a többi pedig az ügyfelet. A gyakorlatban a szolgáltató azonosítása két lépcsős: először magát az országot kell azonosítani az előhívószámmal, majd utána következik a szolgáltató kódja.
A szolgáltató alapú címkiosztás is hasonló módon épül fel. A prefix utáni 5 bit meghatároz egy nyilvántartót, amely a hozzá tartozó címtartományt felosztja a szolgáltatók között. Ezután a szolgáltatót azonosító bitek következnek (hogy erre hány darab bit szolgál, az csak a nyilvántartótól függ). A maradék bitek pedig magát az interfészt azonosítják.
Ez a fajta kiosztás szöges ellentétben áll a mai ***IPv4***-es címkiosztással, amelyben a szolgáltatók nem játszanak főszerepet. A másik csoportot, a ***Geographic Based Unicast Addresses (Földrajzi alapon történő egyesküldésű címek)*** a „hagyományos”, ***IPv4***-es módszer szerint kiosztott címek számára van fenntartva. Az ***IPv6 unicast*** címek ily módon történő felosztásának köszönhetően mindkét fajta címkiosztás lehetséges.
A ***helyi használatú címek (local use addresses)*** a belső hálózatok számára vannak fenntartva, amelyek elszigetelten, tűzfalak árnyékában élik életüket. Az ilyen címek felé menő csomagokat egy útválasztó sem továbbítja a kommunikációs alhálózat felé.
Kétféle helyi használatú cím létezik. Az első a ***kapcsolati lokális cím (Link Local Address)***. Ezt akkor használják, ha a hálózaton lévő összes gép egyetlen ***LAN***-on van. Ilyenkor mind a 118 bit rendelkezésre áll az interfész azonosítására. A másik típus a ***hely lokális cím (Site Local Address)***, amely egy belső intranet kialakítására szolgál. Ilyenkor hálózatunk több különálló ***LAN***-ból áll, amelyeket egymással útválasztókkal kapcsolunk össze. Ebben az esetben a cím első pár bitje szolgál az egyes ***LAN***-ok



3. ábra Az IPv4 csomagok fejléce

azonosítására.
A ***többesküldésű (multicast)*** címek mindig nyolc darab egyessel kezdődnek. Ezek a címek ***unicast*** címek egy csoportját azonosítják. Az erre a címre küldött csomagot a csoport összes tagja megkapja (lásd az előző részt).
Az ***IPv6*** rendelkezik egy új funkcióval, mégpedig a ***bárhova küldéssel (anycasting)***. Ilyenkor szintén interfészek egy csoportjának küldünk egy csomagot, viszont közülük csak az egyik kapja meg (általában az, amelyik a legközelebb helyezkedik el a géphez).
Ha például van több, ugyanazzal a tartalommal ellátott (úgymond tükrözött) fájlszerverünk, és ezekhez rendelünk egy ***anycast*** címet, akkor a gépek ezen a címen mindig kapcsolatba léphetnek valamelyikükkel. Hogy a több szerver közül ki lesz az a kiválasztott, aki a csomagot megkapja, azt az útválasztók döntik el. Az ***anycast*** címek tartományát hiába keressük az 1. ábrán, ugyanis ilyen nem létezik. Az ***anycast*** címek az ***unicast*** címek közül kerülnek kiválasztásra, így elvileg lehetetlen egy címről eldönteni, hogy az ***unicast***, vagy ***anycast*** cím-e.
(Az ***unicast*** és ***anycast*** címek közötti különbség gyakorlatilag annyi, hogy míg az ***unicasthoz*** csak egyetlen interfész van rendelve, addig az ***anycasthoz*** egyszerre több).
Az ***IPv6*** címeket nyolc darab négy hexadecimális számmal jelöljük, amelyeket egymástól kettősponttal választunk el. Például:

0100:0000:0000:0123:4567:89AB:CDEF:
➥1234

Ezt a címet azonban rövidebben is leírhatjuk, például a ***0123***-at ***123***-ként is jelölhetjük. A cím méretét csökkenthetjük azzal is, hogy a ***0***-t nem írjuk ki, sőt, ha több 0 van egymás mellett, akkor az egészet két kettősponttal helyettesítjük. Az előbbi cím tehát így nézne ki rövidebben:

0100::123:4567:89AB:1234.

Az IPv6 csomagok fejléce

A 2. *ábrán* látható az ***IPv6*** csomagok fejlécének felépítése. Most gyorsan áttekintjük, hogy melyik mező mire szolgál.
A prioritás mező segít az útválasztóknak, hogy könnyebben megbirkózzanak a torlódásokkal. A csomagok a prioritás mezőjük szerint két csoportba oszthatók: a 0 és 7, illetve a 8 és 15 közöttiekre. Az első csoportba azok a csomagok tartoznak, amelyek olyan forrásból származnak, amik képesek a forgalomszabályozásra (torlódás esetén visszavehetnek az adás sebességéből). A másik csoport tipikusan valósidejű forgalmat szállít, például hangot vagy mozgóképet. Az ilyen csomagokat kibocsátó alkalmazások sosem változtathatnak az adás tempóján, még akkor sem, ha fennáll a veszély, hogy egyes csomagok útközben elvesznek.
E két csoportot tovább bonthatjuk fontosabb és kevésbé fontosabb csomagok osztályára. Az első csoportot véve példaként, az 1 értékű prioritású mező egy olyan csomagot jelöl, amelyet az útválasztók nyugodtan késleltethetnek, hiszen senki sem fog megbotránkozni, ha pár másodperccel később éri el a célját. Vannak azonban olyan, többnyire interaktív forgalmat szállító csomagok, ahol minden másodperc késlekedés a felhasználókban komoly indulatokat szülhet. Tipikusan ilyen a ***telnet*** vagy az ***ssh***. Az ilyen csomagokhoz legalább 6-os prioritási szintet érdemes rendelni.
A prioritás mező használata lehetőséget ad a beleszólásra, hogy az útválasztók miként kezeljék csomagjainkat, ám van amikor ennél többre van szükségünk. Például két alkalmazás kommunikációjához szükség van valamiféle speciális késleltetésre. Erre nem tudjuk az útválasztókat rávenni kizárólag a prioritás mező használatával.
Erre találták ki az ***IPv6*** folyamcímke (***flow label***) nevű szolgáltatását, amely jelenleg még csak kísérleti jellegű. Segítségével a forrás és a cél között egyfajta álösszeköttetést, úgynevezett folyamot hozhatunk létre, amely különböző tulajdonságokkal rendelkezik (például sávszélesség). A folyamon keresztül haladó csomagok egyedi bánásmódot igényelnek az útválasztóktól. Az útválasztók úgy továbbítják ezeket a csomagokat, hogy ne sértsék meg az adott folyam előírásait. Ha például elő van írva egy meghatározott sávszélesség, akkor az útválasztóknak garantálniuk kell, hogy ez a paraméter ne sérüljön. Ez a szolgáltatás tulajdonképpen nem más, mint hogy virtuális áramkör alapú alhálózat „szimulálása" egy datagram alapú alhálózaton. Ez egyfajta egyesítése a két típusú alhálózatnak úgy, hogy nem sérül a datagram alapú alhálózatok rugalmassága, viszont kiegészül a virtuális áramkör által nyújtott garanciákkal.
Hogy egy csomag melyik folyamba tartozik, azt a folyamcímke mezője határozza meg. Ha egyik folyamhoz sem tartozik, akkor a mező értéke mindig nulla. Az útválasztók a folyamokat egyértelműen azonosítani tudják a forrás és célcím, illetve a folyamcímke mező alapján.
Az ***IPv6*** fejlécét úgy sikerült egyszerűsíteni, hogy azokat a mezőket, amelyekre nincs minden csomagnál szükség, opcionálissá tettek. Az opcionális fejrészeket csak akkor kell a fejléchez csatolni, ha valóban szükség is van rájuk. A 2. ábrán az ***IPv6*** fejlécnek csak a kötelező 40 bájtos részét tüntettük fel. Az opcionális mezők ezután következnek, ezek méretét a ***Payload Length*** nevű mező tartalmazza. Az opcionális részek a 40 bájtos kötelező rész után következnek. Hogy ezek közül melyik az első, azt a ***következő fejléc (next header)*** mező mondja meg. Az utolsó fejléc következő fejléc mezője pedig azt mondja meg, hogy a csomag tartalmát melyik szállítási rétegbeli protokoll (***TCP*** vagy ***UDP***) kezelőjének kell átadni.
Hat típusú opcionális fejrész létezik, ezek nagy részével az útválasztók számára állíthatunk be bizonyos paramétereket. Ezenkívül itt adhatjuk meg a hitelesítésre és a titkosításra vonatkozó paramétereket. Ezek használatához a kommunikáló feleknek először meg kell egyezniük egy vagy több titkos kulcsban

(hogy ezt miként teszik, azzal az *IPv6* nem foglalkozik).
Hitelesítés esetén nincs szükség a csomag tartalmának titkosítására. Ilyenkor csak ki kell nullázni a fejléc azon adatait, amelyek útközben változhatnak (például ugrásszámláló), majd a csomaghoz fűzni a titkos kulcsot, és így elkészíteni a csomag lenyomatát például az *MD5* algoritmussal.
Az *IPv6*-ban a titkosítás is algoritmusfüggetlen, lényegében csak az adón és a vevőn múlik. Mindenesetre, az átjárhatóság érdekében, a *DES-CBC* nevű algoritmus használata a javasolt (a titkosító algoritmusokról sorozatunk egy későbbi részében részletesen is beszélünk).
Az *ugráskorlát (hop limit)* mező akadályozza meg a halhatatlan, az alhálózaton az idők végezetéig bolyongó csomagok létezését. Ez a mező teljesen megegyezik az *IPv4* élettartam nevű mezőjével.
Érdemes egy rövid összehasonlítást végezni az *IPv6* és az *IPv4* csomagok fejléce (3. *ábra*) között. Először is eltűnt az *IHL* és a *Protokoll mező*, mivel az *IPv6* fejlécek rögzített méretűek, másrészt a *Next Header* mezőből kinyerhető, hogy a csomag tartalma milyen szállítási protokollhoz köthető. Eltűntek továbbá az IP csomagok darabolására vonatkozó részek is.
Ez azzal magyarázható, hogy az *IPv6* útválasztók nem darabolják a csomagokat. A szabvány szerint minden útválasztónak kezelnie kell az 576 bájtos csomagokat, viszont az ennél nagyobbakat el kell dobniuk, és hibaüzenetet kell küldeniük a küldő állomás felé. Ha mégis nagyobb méretű csomagokat szeretnénk küldeni, akkor a darabolásról a forrásnak kell gondoskodnia.
Ez a gépek szempontjából visszalépésnek számít, viszont az útválasztók munkája egyszerűsödött, ezáltal gyorsabban dolgozhatják fel a beérkező csomagokat.
Szintén nem találjuk sehol az *ellenőrző összeg (checksum)* mezőt. Ennek oka az, hogy az *IPv6* nem is számol ilyet, mivel a felsőbb rétegbeli protokollok úgyis ellenőrzik az adatok sértetlenségét.

Az áttérés

Az *IPv6* egy rugalmas és gyors protokoll, amely bőséges címtartománnyal rendelkezik. Ezen tulajdonságai alkalmassá teszik, hogy kiváltsa az Interneten jelenleg uralkodó *IPv4* protokollt. Ez az átállás azonban nem egyik pillanatról a másikra fog megtörténni. A már telepített *IPv4*-es gépek nagy száma miatt lehetetlen megoldani, hogy vasárnap éjszaka leállítani az egész Internetet, majd hétfő hajnalban újraindítani úgy, hogy mindenki az *IPv6*-ot használja.
Az átállás folyamatosan, kisebb lépésenként fog történni. Már ma is rengeteg hálózat használja az *IPv6*-ot, egyfajta a „külvilágtól” elzárt szigetet alkotva az Internet hálózatai tengerében. Ezek az *IPv6* hálózatok egymással úgynevezett *alagutakon (tunnel)* keresztül kommunikálnak. Az alagút segítségével egy hálózaton keresztül olyan csomagokat vihetünk át, amely nem kompatibilis az adott hálózattal. Egy *IPv6* csomagot például nem indíthatunk útnak az Internet kommunikációs alhálózatán keresztül, hiszen ezek az útválasztók ma még *IPv4*-et használnak (habár az újabb útválasztók már mindkét protokollt ismerik). Ezért a két *IPv6*-os hálózat között egy alagutat kell létrehozni, amelyen az *IPv6*-os csomagok vándorolnak.
Ez a gyakorlatban úgy működik, hogy az elküldendő *IPv6*-os csomagot berakjuk egy *IPv4*-es csomag belsejébe, és azt indítjuk útnak a célpont hálózat felé. Az alagutak működésére a legjobb szemléltető példa a *Franciaországot Angliával* összekötő *Csalagút*. Ebben az alagútban csak vonatok közlekedhetnek, autók csak úgy, mint a vonatok rakománya. A közúton persze az autók gond nélkül haladhatnak, de az alagútban nem, ott őket vonaton át kell szállítani.
A jövőben ezek a ma még elszigetelt *IPv6* szigetek egyre nagyobb méretűek lesznek, az egyes szigetek egymásba olvadva még nagyobb hálózatokat hoznak létre. Az áttérés utolsó pillanata valószínűleg az lesz, amikor az utolsó két nagy sziget is egymásba olvad.
Hogy ez mikor fog bekövetkezni, nehéz kérdés. Az biztos, hogy még jó pár évre van szükség. Mindenesetre az *IPv6* terjedése ma már sokkal gyorsabb mint régen, mivel az útválasztók és az operációs rendszerek egyre szélesebb körben támogatják az új internet protokollt.
Ezzel be is fejeztük a hálózati réteg bemutatását, a következő résztől a szállítási réteggel és azok protokolljaival, a *TCP*-vel és az *UDP*-vel foglalkozunk.

Garzó András (garzo@interware.hu)
Körülbelül három éve foglalkozik Linux- és más Unix-rendszerekkel. Legjobban az operációs rendszerek lelkivilága érdekli, de nyitott egyéniség. Kedvenc étele a palacsinta, és van egy Richard nevű macskája. Minden észrevételt, megjegyzést, levelet szívesen fogad.

Spamszűrés a gyakorlatban Postfix és Postgrey

Előző cikkemben (Linuxvilág 2005. október) bemutattam a leggyakrabban használt spamszűrési elveket. Ebben az írásban egy konkrét példát mutatok be arra, miként valósítható meg egy szürkelista együttműködése a Postfix levélkiszolgálóval.

Többféle megoldás létezik erre a funkcióra (➲ *http://www.postfix.org/addon.html*). Azért esett a választásom a postgrey nevű programra, mert egyszerűen és könnyen telepíthető, nincs szükség külső alkalmazásra, például adatbázisra, stb.

Az úgynevezett ***szürkelista*** szerveroldali – ***MTA*** szinten megvalósított – spamszűrő módszer, amely úgy működik, hogy az ***SMTP*** szerver a kapott levelet először átmeneti hibával elutasítja, majd amikor egy bizonyos idő múlva a küldő oldal újból próbálkozik ugyanazzal a levéllel, akkor már elfogadja azt szerverünk. A spammerek azonban nem sokat foglalkoznak ideiglenes hibakódokkal, ha egy levél nem megy el az első alkalommal, akkor veszik a következő „áldozat" email címét. A módszer előnye, hogy nincs téves pozitív azonosítás, a rendszer önműködő, nem igényel folyamatos karbantartást.

A ***postgrey*** nevű alkalmazás valójában egy ***Perl*** nyelven készített úgynevezett ***policy démon***, amely ***Berkeley DB*** adatbázisban tárolja a feladó/címzett/kliens adatokat. Képes mind ***unix domain socket***-en, mind pedig ***TCP*** porton át kommunikálni. Ha ugyanazon a gépen futtatjuk, mint a ***Postfixet***, akkor az előbbit javaslom.

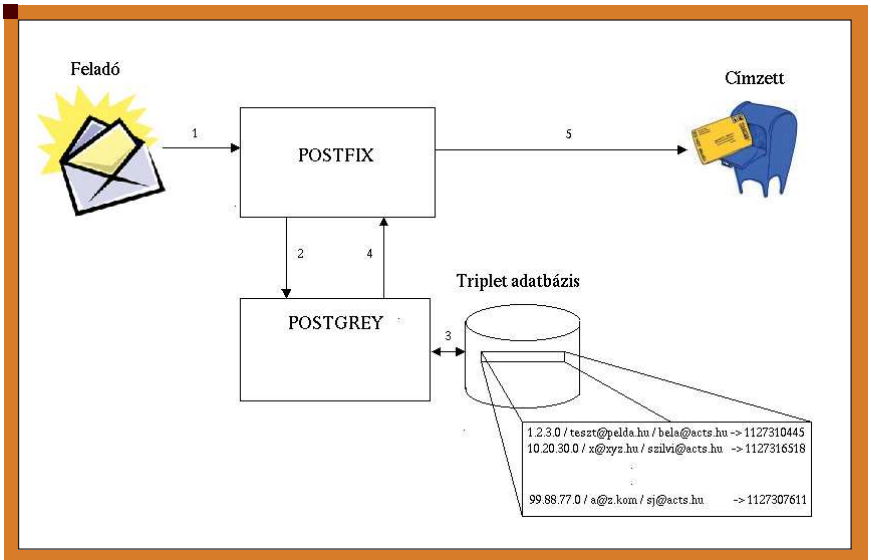
A postgrey működése

Az *1. ábrán* követhetjük nyomon, hogyan működik együtt a ***Postfix*** és a ***postgrey***. A levelet először megkapja a ***Postfix*** (1), majd konzultál a policy démonnal (2). Ez megvizsgálja, hogy a feladó ***IP***-címe (alapértelmezésben a C-osztályú hálózati maszkot nézi), a feladó-, ill. a címzett email címe szerepel-e az adatbázisban (3). Ha nem, vagy a hozzárendelt időbélyeg 5 percnél (konfigurálható érték) frissebb, akkor ***„ideiglenesen elutasítva"*** (`DEFER_IF_PERMIT`), ellenkező esetben ***„tőlem mehet"*** (`DUNNO`) választ ad a ***Postfix*** kérdésére (4). Ez alapján dönti el a ***Postfix***, hogy átmeneti hibával elutasítja az ***SMTP*** kapcsolatot (valójában a kliens `RCPT TO` parancsára ad vissza hibakódot), ellenkező esetben továbbítja a levelet a címzett postafiókjába (5).

Postgrey telepítése

0Ha gépünkön nincsenek meg az alábbi ***Perl*** modulok, akkor installáljuk fel az ***IO-Multiplex***, a ***BerkeleyDB*** és a ***Net-Server*** modulokat a ***CPAN***-ról (➲ *ftp://ftp.cpan.org/*). Töltsük le a ***postgrey*** legutolsó verzióját (a cikk írásakor 1.21) az alkalmazás honlapjáról (➲ *http://isg.ee.ethz.ch/tools/postgrey/pub/*). Csomagoljuk ki, majd másoljuk át a programot egy tetszőleges könyvtárba:

```
tar zxvf postgrey-1.21.tar.gz
cp postgrey-1.21/postgrey /usr/local/bin
```



1. ábra A postgrey működése

Hozzunk létre egy felhasználót, akinek a nevében a ***postgrey policy démon*** futni fog:

```
groupadd postgrey
useradd -g postgrey -d /var/postgrey postgrey
usermod -L postgrey
```

Hozzuk létre az adatbázis könyvtárát, és állítsuk be a megfelelő jogosultságokat:

```
mkdir /var/postgrey
chown postgrey:postgrey /var/postgrey
chmod 700 /var/postgrey
```

Itt az idő, hogy elindítsuk a policy démont, adjuk ki az alábbi parancsot:

```
/usr/local/bin/postgrey -d -u /tmp/postgrey \
--user=postgrey \
--group=postgrey \
--dbdir=/var/postgrey \
--delay=600 \
--greylist-text="Szerverunkon a postgrey
➥spamszuro mukodik. Kerjuk, kuldje ujra
➥a levelet 10 perc mulva"
```

A Postfix beállítása

Most már csak a Postfix tudomására kell hozzuk, hogy használja a policy démonunkat. Szerkesszük a */etc/postfix/main.cf* fájlt, és módosítsuk (vagy hozzuk létre) az `smtpd_recipient_restrictions` változót, amellyel az *SMTP* kapcsolat `RCPT TO` szakaszára alkalmazunk korlátozásokat. Ez a paraméter az én gépemen így néz ki:

```
smtpd_recipient_restrictions = permit_mynetworks,
➥reject_unauth_destination,
➥reject_non_fqdn_recipient, check_policy_service
➥unix:/tmp/postgrey
```

Ezzel a beállítással a *main.cf* `mynetworks` változójában szereplő gépek automatikusan fehérlistára kerülnek, azokra a *Postfix* egyáltalán nem alkalmaz korlátozást. Ha elkészültünk a *main.cf* módosításával, akkor indítsuk újra a *Postfixet* a

```
postfix reload
```

paranccsal, és készen vagyunk, a spammerek egy jó részétől megszabadultunk. Legalábbis egyelőre.

Kóstoljuk meg a pudingot!

Ha levelet kapunk, akkor a szerverünk 25-ös portján az *1. listában* látható kommunikáció folyik.
Látható, hogy a *Postfix* `450`-es kóddal (ideiglenes hiba) utasítja el a levelet. A feladó oldalon pedig hibaüzenetként megjelenik a beállított szöveg, amely megnyugtatja a küldőt, miszerint csak egy átmeneti korlátozásról van szó; és ha a levelet 10 perc múlva újraküldi, akkor az rendben el fog jutni a címzetthez.
Ha 10 perc múlva újra próbálkozik a feladó ugyanezzel a levéllel, akkor egy a *2. listában* látható kommunikációt láthatunk.

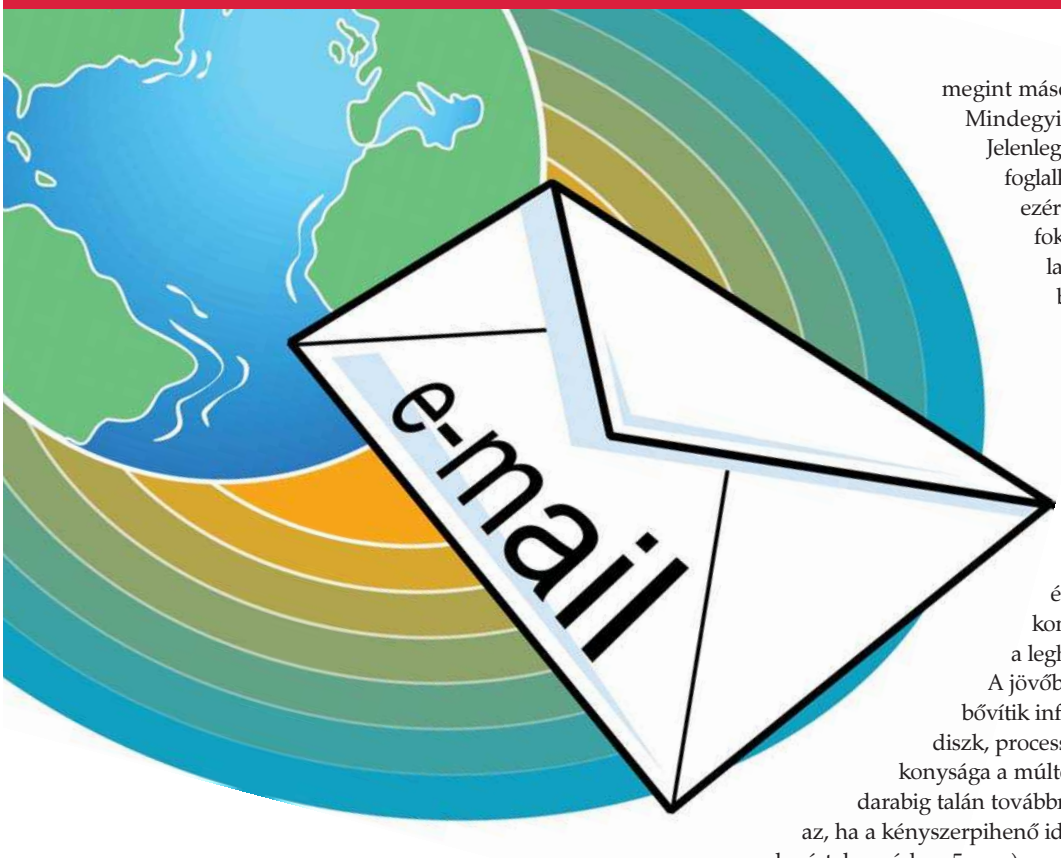
1. lista Kommunikáció a szerver 25-ös portján

```
telnet 10.2.2.2 25
220 rhodium.acts.hu ESMTP Please don't spam
HELO pelda.hu
250 rhodium.acts.hu
MAIL FROM: <teszt@pelda.hu>
250 Ok
RCPT TO: <bela@acts.hu>
450 <bela@acts.hu>: Szerverunkon a postgrey
➥spamszuro mukodik. Kerjuk, kuldje ujra
➥a levelet 10 perc mulva
QUIT
```

2. lista Tíz perc múlva ...

```
telnet 10.2.2.2 25
220 rhodium.acts.hu ESMTP Please don't spam
HELO pelda.hu
250 rhodium.acts.hu
MAIL FROM: <teszt@pelda.hu>
250 Ok
RCPT TO: <bela@acts.hu>
250 Ok
DATA
...
```

Látható, hogy most a `RCPT TO` után már nem utasította el a levelet a *Postfix*, így az mehetett tovább a `DATA` fázisba. Ha ezután ismét levelet akar küldeni az említett feladó ***Bélának***, akkor már nem utasítja el a policy démon, alapesetben 35 napig őrzi meg az ehhez a kapcsolathoz tartozó adatokat – amely érték még a havi hírlevelek szempontjából is megfelelő. De mi van akkor, ha ezúttal nem Bélának, hanem Mártának szeretnének levelet küldeni? Ebben az esetben ismét 450-es hibaüzenettel el fogja utasítani a feladót a *Postfix*. Miért?
Azért, mert a *postgrey* a kliens *IP*-cím/feladó email cím/címzett email cím formátumú úgynevezett ***tripletekben*** (hármasokban) tárolja el az információt. Ha megváltozik pl. a címzett, akkor a *postgrey* ismét egy 10 perces várakozásra kényszeríti a feladót.
Tegyük fel, hogy van néhány megbízható levelezőpartnerünk, akiket nem akarunk ezzel az átmeneti visszautasítással fárasztani. Őket ún. fehérlistára vehetjük fel. Alapértelmezésben az ő *IP*-címüket a */etc/postfix/postgrey_whitelist_clients* fájlban keresi a *postgrey*, de természetesen ez módosítható a `--whitelist-clients` parancssori kapcsoló megadásával. Ha itt megadunk egy *IP*-címet, akkor az onnan érkező összes levelet „fárasztás" nélkül átveszi a *Postfixünk*. Ebben a fájlban sorolhatjuk fel (1 cím – 1 sor) például *Fontos Üzleti Partnerünk* levelező szervereit, így az ő összes levelüket azonnal megkapjuk:




```
# cat /etc/postfix/postgrey_whitelist_clients
1.2.3.4
172.16.88.31
```

Bizonyos esetekben arra is szükség lehet, hogy némely email címre érkező leveleket is késleltetés nélkül beengedjünk. A ***postgrey*** egy másik fehérlistát is kezel, ahol ezeket a címzetteket adhatjuk meg.
A következő példában késleltetés nélkül beengedjük azokat a leveleket, amelyek bármelyik virtuális tartomány postamesterének, vagy pedig a `bela@acts.hu` címre érkeznek:

```
# cat /etc/postfix/postgrey_whitelist_recipients
postmaster@
bela@acts.hu
```

Ha menet közben módosítjuk valamelyik fehérlistát, akkor azt a ***postgrey***-nek küldött `HUP` jelzéssel adhatjuk tudtára.
A ***postgrey*** az említett fehérlistákon kívül még egyet kezel, amelyet automatikusan tart karban. Erre a listára akkor kerül fel egy kliens, ha már legalább 5 (`--auto-whitelist-clients`) alkalommal sikeresen átjutott a szürkelistán – óránként csak 1 sikeres kézbesítés számít. A ***postgrey*** feltételezi, hogy ebben az esetben legitim levelezőpartnerről van szó, és ezután már nem állítja meg ezt a klienst átmeneti hibával. Ha letelt a 35 nap (`--max-age`), akkor a klienst törli erről a fehérlistáról.

További gondolatok

A ***postgrey Berkeley*** adatbázisban tartja a tripleteket, de nem ez az egyetlen lehetőség. Az adatbázis épsége érdekében tranzakciókat használ. Más megvalósítások például ***MySQL*** adatbázisban tárolják ezeket az információkat, megint mások pedig a fájlrendszerben. Mindegyiknek megvan a maga előnye. Jelenleg a spammerek jellemzően nem foglalkoznak az elutasított levelekkel, ezért a szürkelista nagyon jó hatásfokkal működhet. A ***postgrey*** honlapja szerint, míg a szürkelista bekapcsolása előtt percenként körülbelül 15 vírus és spam került a rendszerbe, addig utána már csak 3. A szürkelista tehát hatékonyan képes csökkenteni az egyéb típusú (például ***Bayesian***, heurisztikus, stb.) spamszűrők terhelését. Amint előző írásomban is említettem, jelenleg a ***Bayesian*** és heurisztikus spamszűrővel kombinált szürkelistát tartom a leghatékonyabb védekezésnek.
A jövőben azonban – ha a spammerek bővítik infrastruktúrájukat (sávszélesség, diszk, processzor, stb.) – a szürkelista hatékonysága a múlté lehet. Ebben az esetben egy darabig talán továbbra is megfelelő védelmet adhat az, ha a kényszerpihenő idejét (ami a ***postgrey*** esetében alapértelmezésben 5 perc) megnöveljük – ***Evan Harris*** (➲ ***http://www.greylisting.org/articles/whitepaper.shtml***) eleve 1 órás beállítást javasol. Mindenesetre ha a spammerek így is tesznek, az számukra megdrágítja az egy levél tényleges elküldésére jutó egységnyi költséget, és ez nekünk jó.
Gyakori az az eset, amikor a spammerek a tartalék (backup) `MX` szervereket célozzák meg a spammel, mert az elsődleges `MX` szerver általában minden további nélkül átveszi a leveleket a tartalék `MX`-ektől. Ezt úgy lehet kivédeni, ha egy tartomány minden `MX` szerverét felruházzák szürkelista védelemmel, ill. közös adatbázisból dolgoznak, és ha az elsődleges `MX` fehérlistájában szerepel az összes tartalék `MX`. A ***postgrey*** ebben a környezetben is megállja a helyét, képes ***TCP*** socket-en is kommunikálni (a `-i` kapcsoló használatával). ***Harris*** szerint, ha a spammerek alkalmazkodnak, akkor nyilván a lehető leghamarabb el akarják küldeni a leveleket, ezért rövid időn belül – minden bizonnyal többször is – próbálkozni fognak, még mielőtt a szürkelista által megszabott idő letelne. Ebben az esetben viszont érdemes módosítani a szürkelista implementációkat úgy, hogy számolják, hány idő előtti próbálkozás történik, és egy bizonyos határ felett (hogy a legitim ***SMTP*** szerverek ne essenek bele) már nem szürke, hanem feketelistára tehetjük ezeket a próbálkozókat. Ha pedig már ez is kevés lesz, akkor új megoldások után kell néznünk, de addig is remélhetőleg lesz még pár napos és spammentes napunk.

Sütő János (jsuto@freemail.hu)
1997 óta használ Slackware Linux-ot. Szabadidejében a postfix clapf nevű vírus- és spamszűrőjét polírozza.

Linux mint Ethernet híd

Úgy adja tovább a csomagokat, mint egy híd, és úgy szűri őket, mint egy tűzfal. Ha a szükség úgy hozza, bármilyen kiszolgálót vagy készüléket külön védelemmel láthatunk el úgy, hogy a beállításain semmit nem kell módosítanunk.

Kértek valaha arra, hogy erősítsük meg egy olyan forgalomirányító védelmét, amelyhez nincs rendszergazdai hozzáférésünk? Mit lehet tenni, ha nem mi felügyelünk egy hálózatot, de erőteljesebb védelmet akarunk az általunk használt szegmensre? Egy ehhez hasonló kérés kapcsán ismerkedtem meg a ***Bridge (híd)*** világával, a linuxos ***Ethernet*** híd kifejlesztésére irányuló tervezettel.

A ***Bridge*** webhelye szerint:
Az ***Ethernet*** áthidalás egyike a több hálózat összekapcsolásával egy nagyobb hálózat kialakítására alkalmas megoldásoknak. Az áthidalásra vonatkozó szabvány az ***ANSI/IEEE 802.1d***. Híd segítségével két különböző hálózati szegmenset tudunk protokollfüggetlen módon összekötni. A csomagok továbbítása ***Ethernet***-cím alapján történik, és nem ***IP***-cím alapján, ahogy a forgalomirányítók esetében. Mivel a továbbítás a második rétegben folyik, a hídon bármely protokoll átlátszó módon haladhat át.

A kód karbantartását jelenleg ***Stephen Hemminger*** végzi, a 2.4-es és a 2.6-os rendszermaghoz egyaránt. Az áthidaló kód a legtöbb korszerű, 2.6-os rendszermagra épülő terjesztésbe be van építve. Írásom alapjaként a ***Fedora Core 3*** szolgált, mely szintén a 2.6-os rendszermagra épül. Aki leragadt a 2.4-es rendszermagnál, annak sem kell kétségbe esnie, a ***Bridge*** webhelyén (lásd az internetes forrásokat) ugyanis ehhez is találhatók foltok.
A tűzfalhíd, vagy, ha úgy tetszik, áthidaló tűzfal tűzfalösszetevőjét egy másik tervezet, az ***ebtables*** keretein belül fejlesztik. Az ***ebtables*** egy szűrőréteg az áthidalást is végző tűzfalakhoz. A szűrés az adatkapcsolati rétegbeli Ethernet keretek tartalma alapján történik. A szűrés mellett az ***Ethernet MAC***-címek módosítására is van lehetőség. Az ***ebtables*** kód révén ***iptables*** szabályokat is lehet áthidaló módban használni, így a tűzfalat ***IP***- és ***MAC***-szintű szűrőkkel egyaránt elláthatjuk.

Mi az a híd?

A híd kettő vagy több, azonos hálózati technológiát alkalmazó hálózati szegmenset összekapcsoló készülék. A szegmensek topológiája lehet eltérő, akár optikai szálas és rézkábeles szakaszokat is összekapcsolhatunk, ám a technológiának azonosnak kell lennie. A legegyszerűbb az, ha a hidat linuxos hubként szemléljük. A géphez annyi kaput adhatunk hozzá, amennyit csak akarunk, mindegyik ugyanannak a hubnak lesz a része. Ami az egyik kapun beérkezik, az az összes többin kimegy, hacsak nem mást írunk elő a szabályokban. Ha a hub működik, ***iptables*** és ***ebtables*** szabályokkal pontosan úgy szűrhetjük a forgalmat, ahogy azt bármely más linuxos forgalomtovábbító rendszeren is megtehetjük.

A kezdés

Első lépésként vegyünk egy két hálózati csatolóval ellátott számítógépet.

1. kódrészlet A hálózat beállításainak megadása előtt ellenőrizzük, hogy mindkét Ethernet csatoló működik-e

```
#> ifconfig
eth0  Link encap:Ethernet  HWaddr 00:CC:D0:99:EB:26
      inet6 addr: fe80::2b0:d0ff:fe99:eb26/64 Scope:Link
      UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
      RX packets:86208855 errors:0 dropped:0 overruns:63 frame:0
      TX packets:77098217 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:3871506445 (3692.1 Mb)  TX bytes:266311184 (253.9 Mb)
      Interrupt:5 Base address:0xec00
eth1  Link encap:Ethernet  HWaddr 00:CC:03:D8:3A:1A
      inet6 addr: fe80::201:3ff:fed8:3a1a/64 Scope:Link
      UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
      RX packets:77087614 errors:0 dropped:0 overruns:0 frame:0
      TX packets:85110321 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:264995582 (252.7 Mb)  TX bytes:3672580334 (3502.4 Mb)
      Interrupt:9 Base address:0xec80
```

2. kódrészlet Két egyszerű beállító fájl IP-címmel nem rendelkező hálózati csatolókhoz

```
/etc/sysconfig/networking-scripts/ifcfg-eth0:
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
/etc/sysconfig/networking-scripts/ifcfg-eth1:
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=static
```

Mire végzünk, linuxos gépünk normál hubként fog működni, kapui között az igényeink szerint továbbítva a forgalmat. Ha az egyik csatolót a megszokott fali aljzatba csatlakoztatjuk, a másikhoz pedig hozzákötünk egy hordozható gépet, akkor úgy tudjuk majd róla használni a hálózatot, mintha közvetlenül az aljzatba dugtuk volna.

Célunk az, hogy a híd minden hozzá csatlakozó készülék számára átlátszó legyen. Érdekességként megjegyezném, hogy a hídnak nem muszáj ***IP***-címet adnunk, hacsak nem akarunk távolról csatlakozni hozzá, például karbantartás vagy a naplók áttekintése céljából. Természetesen manapság nem sok értelme van lemondani az ***IP***-címről, és ezt mi sem fogjuk megtenni.

A hardver esetemben egy régebbi gép volt – éppen egy hasonló feladatra várt. Processzora egy ***AMD K6-450*** volt, memóriája pedig 256 MB-nyi. Egyetlen 15 GB-os ***IDE*** merevlemez volt benne, valamint egy ***3Com*** 10/100-as ***Ethernet*** kártya. Volt egy másik ***3Com*** 10/100-as kártyám is, és mivel ezek elég jól működnek ***Linux*** alatt, ez lett a második csatoló. A hídprogramon túl néhány egyszerű tűzfalszabályt akarok csak futtatni, illetve behatolásészlelésre még talán a ***Snortot***. A forgalom várhatóan kicsi lesz, és valószínűleg a ***Snort*** sem fog nagy adatmennyiségeket kezelni, vagyis a 256 MB memóriának elégnek kell lennie. Ha valaki gigabites forgalmat akar kezelni, esetleg valós idejű szimatolást akar végezni, az értelemszerűen erősebb gépet lesz kénytelen üzembe helyezni.

Ha készen áll a gép, telepítsük fel a ***Fedora Core 3***-at, illetve az általunk igényelt kiegészítő programokat. Ha a lehető legmagasabb szintű biztonságot kívánjuk elérni, törekedjünk a lehető legkevesebb szoftver telepítésére. Ha később mégis szükségünk támadna valamire, a ***YUM*** segítségével könnyedén telepíthetjük. Egyelőre elégedjünk meg egy egyszerű ***Linux***-telepítéssel, illetve ellenőrizzük, hogy felismerte-e a hálózati csatolókat. Az ***ebtables*** kód lefordításához szükség lesz a rendszermag forrására és a szokásos fordítóeszközökre is, vagyis ezeket ne hagyjuk ki. Ha a gép telepítésével végeztünk, és elhelyeztük termelési környezetébe, ne feledjük majd eltávolítani róla a felesleges szoftvereket. A telepítés befejeződése után indítsuk újra a gépet, majd jelentkezzünk be rootként.

Most létre kell hoznunk egy virtuális hálózati eszközt. Annak nevezzük el, aminek akarjuk, nálam br0 lett, mint az első ***bridge***, híd eszköz:

```
#> brctl addbr br0
```

Futtassuk le az `ifconfig`-ot. Láthatók a hálózati csatolók? Az *1. kódrészletben* látható, hogy két hálózati csatolónk van, és nincs ***IP***-cím rendelve hozzájuk. Aki esetleg már rendelt ***IP***-címet a csatolókhoz, az az egyszerűség kedvéért most törölje azt. ***Fedora*** alatt a ***/etc/sysconfig/networking-scripts/ifcfg-X*** fájlt kell átírni, ahol az ***X*** a csatoló azonosítója. Az én rendszeremen a két csatoló az `eth0` és az `eth1` volt, az ezekhez tartozó, ***IP***-címet tartalmazó sorokat kellett törölni vagy megjegyzésbe tenni. Fontos, hogy rendszerindításkor mindkét csatoló feléledjen. A *2. kódrészlet* egy alapszintű beállításegyüttest tartalmaz, ennek jól kell működnie.

Ha a fentiekkel végeztünk, ne feledjük el újraindítani a hálózatkezelést; ezt a `service network reload` paranccsal tehetjük meg.

Ez után – az alábbi módon – közöljük a rendszerrel, hogy mely eszközök tartoznak egy csoportba. Az utolsó sorral egyben a virtuális eszközt is üzembe helyezzük:

```
#> brctl addif br0 eth0
#> brctl addif br0 eth1
#> ip link set br0 up
```

Linuxos gépünk ettől a pillanattól kezdve alapszintű hubként működik. Aki nem bírja a kíváncsiságot, csatlakoztassa a csatolókat az ***Ethernet*** hálózatra, majd próbálja ki az új hubját. Természetesen a gép egyelőre vakon továbbítja a forgalmat, és nem rendelkezik ***IP***-címmel sem. Én szeretem, ha telepítés után távolról is tudok csatlakozni a gépeimhez, ezért hozzárendeltem egy ***IP***-címet a `br0` csatolóhoz, illetve forgalomirányítási adatokat is csatoltam.
A hídcsatoló ***IP***-címét a következő paranccsal lehet megadni:

```
#> ip addr add 10.1.1.18/16 brd + dev br0
```

A cím mellett meg kell határozni az alhálózati maszkot is (/16), illetve azt, hogy melyik hídeszközhöz kell hozzárendelni. Ennek elsősorban akkor van szerepe, ha egynél több virtuális eszközünk is van a gépben. Igaz, esetünkben csak egy ilyen volt, ám a parancs írásmódja szükségessé tette ennek az egynek a kiválasztását. Aki más nevet adott a hídnak, annak itt azt a nevet kell használnia.
A híd távoli elérésének lehetővé tételéhez már csak a forgalomirányítást kell elintézni:

```
#> route add default gw 10.1.1.1 dev br0
```

Itt a szokásos forgalomirányítási parancsokat és szabályokat alkalmazhatjuk, és a `br0` eszközt is pontosan úgy használhatjuk, mint az összes többi linuxos hálózati csatolót.

Tesztelés

Minden a helyén van, próbáljuk ki a rendszert. Először ellenőrizzük, hogy az összes beállítás rendben van-e:

```
#> brctl show
bridge name   bridge id          STP enabled    interfaces
br0           8000.0030843e5aa2  no             eth0
                                                eth1
```

Mint látható, egyetlen `br0` nevű hídeszközünk van, amely az `eth0` és az `eth1` csatolót használja. A jelek szerint minden rendben van.

Telepítés

Ideje nekilátnunk a fizikai telepítésnek. Az egyik hálózati csatolót csatlakoztassuk a hálózati kapcsolóhoz, pontosan úgy, mint ahogy bármilyen más számítógéppel tennénk. Az összeköttetés mindkét végén ki kell gyulladniuk a jelzőfényeknek. Egy keresztkötésű kábellel csatlakoztassunk egy asztali vagy hordozható gépet a linuxos gép másik csatolójához. Várjuk meg a jelzőfények bekapcsolását, számoljunk el tízig, majd a most csatlakoztatott gépről pingeljük meg a hálózat valamelyik gépét. A linuxos hub túloldalán található hálózatot úgy kell látnunk, mintha közvetlenül csatlakoznánk hozzá.

Túlélni az újraindításokat

Az, hogy a rendszert hogyan vesszük rá az újraindítások túlélésére, a mi választásunk. Talán a legegyszerűbb az, ha az eddig kiadott parancsokat hozzáadjuk a rendszerindítás végén lefutó */etc/rc.local* fájlhoz, ekkor a híd indítás után azonnal működőképes lesz.

Tűzfal emelése

Minden hálózati forgalmat továbbító vagy átengedő linuxos rendszer esetében lehetőségünk nyílik az áthaladó adatfolyam szűrésére. Az áthidaló tűzfalak esetében sincs ez másként. A tűzfalszabályok létrehozására és karbantartására több lehetőség is kínálkozik. A legegyszerűbb tűzfal az, amikor tiltunk mindent, kivéve azt, amit kifejezetten engedélyeztünk – az alábbiakban egy ilyen összeállítását ismertetem.
A tűzfal beállításainak megadásához le kell töltenünk és telepítenünk kell a felhasználói térben futó ***ebtables*** segédeszközöket, melyeket az ***ebtables*** webhelyéről érhetünk el (lásd a forrásokat). Írásom születésekor a legújabb változat a 2.0.6-os volt. A számos tükröoldal valamelyikéről töltsük le, majd a kezdő `configure` lépés kihagyásával végezzük el a szokásos kibontás-telepítés eljárást:

```
#> tar -xzf ebtables-v2.0.6.tar.gz
#> cd ebtables-v2.0.6
#> make
#> install
```

Ha minden rendben ment, már használhatjuk is az ebtables parancsot. Gépeljük be a parancssorba az `ebtables-t`, ennek hatására valami ilyennek kell megjelennie:

```
#> ebtables -V
ebtables v2.0.6 (November 2003)
```

Először azt kell biztosítani, hogy az ***iptables*** fogadásra legyen állítva. Mivel ***Fedora Core 3*** alatt dolgozunk, ehhez elég csak leállítanunk a szolgáltatást:

```
#> service iptables stop
#> chkconfig --level 35 iptables off
```

Hasonló hatást érhetünk el a `flush` (kiürítés) parancs alkalmazásával is. Listázzuk ki a meglévő láncokat, majd ürítsük ki őket:

```
#> iptables -L
#> iptables -F INPUT
#> iptables -F OUTPUT
#> iptables -F FORWARD
#> iptables -F RH-Firewall-1-INPUT
```

Most a tűzfalon áthaladó forgalmat teljes egészében le kell tiltanunk, érkezzen az a hálózat bármelyik részéről is. Az alábbi szabályok csak a jelenlegi példahálózatban használhatók, ha más környezetben akarjuk alkalmazni őket, akkor az alhálózatokat és az állomásokat értelemszerűen módosítanunk kell:

```
/sbin/ebtables -A FORWARD -p IPv4 \
--ip-source 10.2.0.0/16 -j DROP
/sbin/ebtables -A FORWARD -p IPv4 \
--ip-source 10.7.0.0/16 -j DROP
/sbin/ebtables -A FORWARD -p IPv4 \
--ip-source 10.4.0.0/16 -j DROP
```

```
/sbin/ebtables -A FORWARD -p IPv4 \
--ip-source 10.5.0.0/16 -j DROP
/sbin/ebtables -A FORWARD -p IPv4 \
--ip-source 10.6.0.0/16 -j DROP
/sbin/ebtables -A FORWARD -p IPv4 \
--ip-source 10.1.0.0/16 -j DROP
```

Aki szerzett már gyakorlatot az ***iptables*** használatában, annak ismerős lesz az írásmód. Közöljük az ***ebtables*** programmal, hogy amikor továbbít valamit (FORWARDING) az ***IPv4*** protokoll használatával, akkor dobjon el (DROP) minden a 10.2.0.0/16 alhálózatból származó csomagot. A többi paranccsal ugyanezt írjuk elő a többi alhálózathoz is.
A következő lépés magának a tűzfal mögött lévő eszköznek az engedélyezése. Ha ennek IP-címét nem vesszük be a továbbításra engedélyezettek közé, akkor semmi nem fog működni. Ne feledjük, ha maga a tűzfal is kap IP-címet, akkor ezt is engedélyeznünk kell:

```
/sbin/ebtables -I FORWARD 1 -p IPv4 \
--ip-source 10.1.1.5 -j ACCEPT
/sbin/ebtables -I FORWARD 1 -p IPv4 \
--ip-source 10.1.1.18 -j ACCEPT
```

Az alábbiakkal a hordozható gépem elérésére jogosult hálózati eszközöket adtam hozzá:

```
/sbin/ebtables -I FORWARD 1 -p IPv4 \
--ip-source 10.1.10.30 -j ACCEPT
/sbin/ebtables -I FORWARD 1 -p IPv4 \
--ip-source 10.1.10.19 -j ACCEPT
/sbin/ebtables -I FORWARD 1 -p IPv4 \
--ip-source 10.1.10.87 -j ACCEPT
```

Ellenőrzésképpen elsétáltam ahhoz a géphez, amelynek címét a fenti ACCEPT (elfogadás) szabályokban megadtam, majd megpróbáltam megpingelni a 10.1.1.5 címet használó hordozható gépemet. Most üljünk át egy másik, a fentiekben nem szereplő csomóponthoz – a `ping` nem fog működni.

Megvalósítás éles környezetben

Nemrég elhívtak egy felhasználó telephelyére, hogy javítsam fel egy pénzügyi kiszolgáló védelmét. A kérés egyszerű volt: kell egy tűzfal a rendszer elé, de az *IP*-címét nem lehet megváltoztatni. Két hálózati kártyával és ***Linuxszal*** felszerelkezve néhány perc alatt megoldottam a feladatot. A telepítés sem okozott gondot. A tűzfalat és a kiszolgálót egy keresztkötésű kábellel kapcsoltam össze, a tűzfal másik csatolója és a fali aljzat közé pedig normál kábel került. Ennyi volt az egész. Nem kellett újratervezni az *IP*-címzési séma egyik részét sem, egyszerűen csak össze kellett dugni a gépeket. Miután megírtam néhány szabályt, amelyek szerint az összes csomagot el kellett dobni, kivéve az engedélyezett *IP*-címekről és kapukról érkezőket, a munkát befejezettnek nyilváníthattam.
A ***Linux*** egyik szépsége, hogy egyszerre akár nagy számú szolgáltatást is képes futtatni. Vegyük a fenti példát. Adva volt egy érzékeny kiszolgáló, ami elé rövid idő alatt fel tudtam állítani egy tűzfalat. A ***Linux*** révén kapott idő- és pénzbeli megtakarításon túl azonban arra is módunk nyílt, hogy ***Snortot*** futtassunk a tűzfalon. Elég volt egy pillanatra belenyúlni a szimatolóprogram beállító fájljába (*/etc/snort.conf*), közölni vele, hogy a `br0` csatolón vizsgálódjon, és a ***Snort*** azonnal megkezdte munkáját az áthidaló csatolón.
Ezen a ponton lehetett ráérezni az áthidaló kód igazi erejére. A hálózat egyik szegmense feltűnően lassú, de nem tudni miért – ismerős a helyzet? A következő alkalommal elég fogni egy linuxos gépet, feltenni rá a ***Snortot*** és a többi felderítőprogramot, majd üzembe helyezni a hidat. Fogjunk tehát egy keresztkötésű kábelt, és már indulhatunk is a helyszínre. Mivel a híd hubként viselkedik, a linuxos gépet a hálózat tetszőleges pontjára beiktathatjuk. Elég a fizikai kapcsolatokat biztosítanunk, a gépet bárhol letehetjük, és a következő pillanatban már vizsgálhatjuk is a forgalmat. Utolsó munkánknak része volt több átlátszó ***Squid*** gyorsítótár-kiszolgáló is – ezek valóban átlátszóak, az IP-címzési sémában, az ügyfelek oldalán vagy a böngészőknél semmilyen módosítást nem igényelnek. Elég a ***Squidet*** futtató gépet a forgalomirányító elé helyezni, majd minden a 80-as kapun folyó forgalmat ráirányítani.
A ***Linux*** azon képessége, hogy átlátszó módon tud beépülni a meglévő hálózati rendszerbe, újfajta, korszerű szolgáltatások biztosítását teszi lehetővé. Mivel különböző hálózati készülékeket is egyesíthetünk egyazon virtuális elemben, ugyanazzal géppel tűzfalat is üzemeltethetünk, miközben tetszőleges szempontból figyelhetjük a hálózatot. Az egyetlen korlátot a vas sebessége és a rendelkezésünkre álló kártyahelyek száma jelenti.

Köszönetnyilvánítás

A szerző köszönettel tartozik azoknak a fiúknak és lányoknak, akik az áthidaló kódot készítették, valamint az ***ebtables*** parancskészlet szerzőinek, amiért üzembiztos, jól használható eszközt állítottak össze és tettek elérhetővé *GPL* hatálya alatt.

Linux Journal 2005. július, 135. szám

Jim Robinson a Maconben, Georgia-ban található Linux Solutions Provider, Inc. tanácsadó vállalkozás elnöke. Örömmel játssza férji és apai szerepét, gitározik és persze linuxozik.

KAPCSOLÓDÓ CÍMEK

- bridge.sourceforge.net
- ebtables.sourceforge.net
- www.snort.org
- www.squid-cache.org

Térhatás

A Blender használata (3. rész)

Világítás és renderelés

Már kiskoromban elbűvöltek a számítógépek. Mindig is vágytam egy sajátra. Minden alkalmat megragadtam hogy leülhessek elé, és nyomkodhassam. Erre a szüleim is felfigyeltek... nemsokára kaptam egy billentyűzetet.

„Remek – mondta édesapám – most már csak pár dolog kell: egy számítógép, a monitor, az egér, az egérpad, a… ”

Valahogy így van ez a 3D szerkesztéssel is. Tudunk már objektumokat létrehozni, tudjuk az ***Edit Mode*** alapjait is, már csak az a „néhány” dolog választ el minket attól, hogy megtanuljunk képeket készíteni. Sajnos nehéz helyzetben vagyok, amikor el kell döntenem miről is, írjak először. Egy-egy rész megértése sajnos nem mindig elég, hogy elérjük célunkat, de azt hiszem ez nem csak a ***Blender*** esetében, van így.
Az előző szám végén megígértem, hogy megmutatok néhány ***Edit Mode***-beli csintalanságot, illetve világítani fogunk, renderelünk és árnyékot is vetünk. Lássuk...

Hide/Reveal

Válasszunk ki egy tetszőleges objektumot, váltsunk ***Edit Mode***-ba (***TAB*** billentyű), majd jelöljünk ki néhány vertexet. Nyomjuk meg a ***W*** billentyűt. Egy menü jelenik meg ***Specials*** feliratal. Ha kiválasztjuk a ***Hide*** menüpontot, láthatjuk hogy a kijelölt vertexeink eltűntek. Nem töröltük őket, egyszerűen csak nem látszódnak. Néha jól jön, ha ideiglenesen eltüntethetjük a zavaró részeket, így csak azt látjuk a képen amivel tényleg dolgozunk. Ha az eltüntetett részeket újra láthatóvá szeretnénk tenni, csak válasszuk ki az előbbi menü ***Reveal*** pontját.

Subdivide

Szerkesztés közben az egyik leggyakrabban használt eszköz a ***Subdivide***. (***Subdivide Fractal***, illetve ***Smooth***). A menüpont hatására a kijelölt élek kettéosztódnak, minden kijelölt él közepére egy-egy új vertex kerül. A ***Subdivide Fractal*** menüpont ezeket a vertexeket nem középre helyezi, hanem véletlenszerűen elmozdítja, a ***Subdivide Smooth*** pedig megpróbálja egyenletesebbé tenni („simítani”) az alakzatot. Tartózkodjunk ezek mértéktelen használatától, ugyanis a vertexek száma exponenciálisan nő, így hamar azon kaphatjuk magunkat, hogy elfogyott a memóriánk.

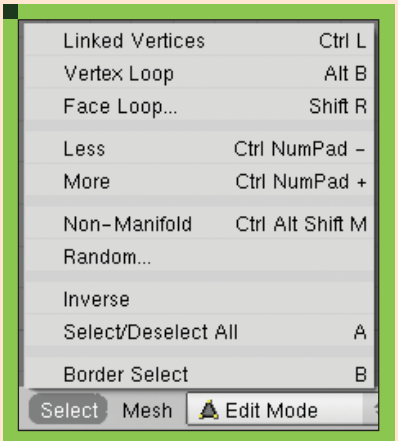
■ *1. ábra* Specials Menü

Bevel

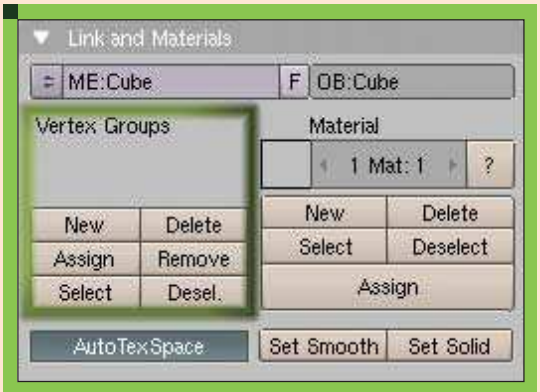
A ***Bevel*** menüpont tipikusan élek lekerekítésre való. Legkönnyebben egy kockán próbálhatjuk ki. Egyetlen hátránya, hogy nem lehet kijelölésre alkalmazni, így az egész alakzatunk áldozatául esik. Ennek ellenére nagyon jól használható eszköz.

Select

Sokszor előfordul, hogy tudjuk, mit szeretnénk kijelölni, az elhelyezkedés miatt mégis becsúszik egy-egy felesleges vertex. Majd megpróbáljuk egyesével kijelölni őket, de mielőtt végeznénk, elhibázzuk és kezdhetjük újra. Ennek elkerülésére létezik



2. ábra Select menü



3. ábra Vertex Groups

néhány trükk a ***Blenderben***, amivel időt takaríthatunk meg és a gépünkre sem kell megharagudnunk.
Ha ***Edit Mode***-ban vagyunk, a ***3D View*** fejlécén található ***Select*** menüben sok hasznos dolgot találunk:

- ***Random*** – Véletlenszerű kijelölés
- ***Inverse*** – Kijelölés megfordítása (ami nem volt kijelölve most ki lesz)
- ***More*** – A kijelölt vertexek szomszédait is kijelöli
- ***Less*** – A ***More*** fordítottja

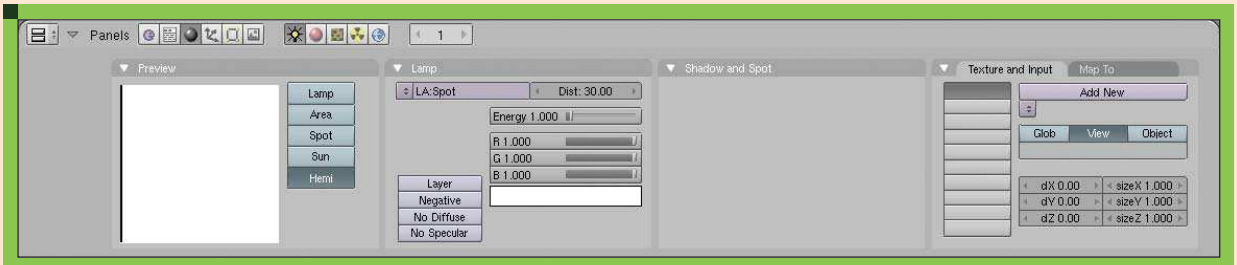
Vertex Groups

A ***Links and Materials*** panelen található ***Vertex Groups*** gombokkal vertexeinket csoportosíthatjuk, kijelöléseinket elmenthetjük, azoknak nevet adhatunk, stb. Így később egy bonyolult modell egy részét három kattintással kijelölhetünk. Új vertex csoportot a ***New*** gombbal hozhatunk létre, illetve a ***Delete*** gombbal törölhetjük. Az ***Assign*** gombbal adhatunk hozzá vertexet a csoporthoz, a ***Remove***-val pedig eltávolíthatunk. A ***Select*** és ***Deselect*** gombok pedig a csoportok kijelölése valók. Érdemes megtanulni a használatát, később az animációknál szükség lesz rá.

Renderelés

Egy későbbi számban még bemutatásra kerül néhány modellezést segítő eszköz, azonban meg kell, hogy ismerkedjünk a ***Blender*** lelkével, a ***Render*** motorral. A kép(ek) előállítása esetenként hatalmas erőforrást igényelhet, órákig – rosszabb esetben napokig – is eltarthat. Senki nem örülne neki, ha 20 perc várakozás után venné észre, hogy valami apróság javításra szorul, majd javítás után ismét 20 percet kellene várnia, hogy láthassa a végeredményt. A ***Blender*** beépített renderelője nagyon jól testreszabható, skálázható, így gyorsan készíthetünk kis méretű, rosszabb minőségű, úgynevezett előnézeti képeket, mielőtt elindítjuk a tényleges render folyamatot.
A render folyamattal kapcsolatos panelek láthatóak a *4. ábrán*. A bal oldalin (***Output***) a kimeneti mappát és a renderelés hátteréül szolgáló képet (***backbuffer***) állíthatjuk. Lejjebb különféle opciókat találunk, ezek közül az ***Edge effekt*** a legfigyelemreméltóbb.
Jobbról, a második panelen lévő ***Render*** gombbal indíthatjuk el a render folyamatot, alatta a folyamatot végző engine-t választhatjuk ki.
Választhatjuk a belső renderelőt (***Blender Internal***), illetve egy külön telepítendő programot, a ***YafRay***-t.
A ***Render*** gombtól jobbra különböző effekteket kapcsolhatunk be, illetve ki. Ezek közül a legfontosabbak a ***Shadow*** (***árnyékok ki/be kapcsolása***) a ***Ray*** (***Raytracing*** vagy ***sugárkövetés*** ki/be kapcsolása), a ***Pano*** (***panorámakép*** készítése), illetve a ***Radiosity*** (Egy olyan effekt, melynek során az objektumokról visszavert fény is befolyásolja a szomszéd objektumok színét).
Az ***OSA*** feliratú gombbal az ***élsimítást*** (***Antialiasing***) kapcsolhatjuk be, alatta a simítás mértékét állíthatjuk.
Ha az ***MBLUR*** (***Motion Blur***) gomb be van kapcsolva, a végeredmény egy utólag kicsit elmosott kép. Vigyázzunk azonban evvel az opcióval, mert – bár jól néz ki – a rendereléshez nagyságrendekkel több idő kell. Ne használjuk, ha nem feltétlenül, muszáj.
Természetesen nagyobb kép renderelése is jóval tovább tart, ezért ajánlott először mindig egy kisebbet készíteni ellenőrzés képpen – erre valók a 75%, 50% stb. gombok – majd, ha mindent rendben találunk, renderelhetjük teljes méretben is.
A kimeneti formátumot, és a kép méretét a jobb oldali (***Format***) panelen állíthatjuk, a ***PAL***, ***NTSC***, ***PC***, stb. gombok előre beállított szabványos méreteket tartalmaznak.
A kimeneti formátumok között találhatunk ***AVI***-t is, ugyanis a ***Blender*** animációk készítésére is kiválóan alkalmas, erről egy későbbi cikkben szándékozom írni.

4. ábra A render panel



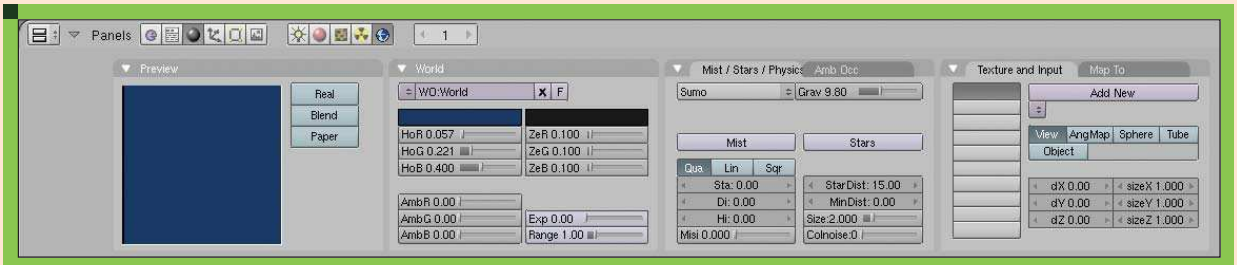
■ *5. ábra* Fényforrások beállításai

A Megvilágosodás

Ahogy a földön sem létezik élet fény nélkül, úgy a 3D-s objektumaink sem „életképesek", ha nem gondoskodunk a megfelelő megvilágításról. Akár a ***YafRay***-t akár a ***Blender*** belső renderelőjét használjuk, szükségünk van minimum egy *lamp* objektumra. Renderelni nélküle is tudunk, azonban a legszebben textúrázott ***mesh*** sem ér semmit megfelelő fények nélkül. Míg a való világban csak kevés fényforrás áll rendelkezésünkre (többnyire csak a ***Nap*** fénye, esetleg egy-két lámpa), addig a ***Blenderben*** tetszőleges mennyiségű fényforrást hozhatunk létre. Bár 'csak' öt különböző típusú fényforrás létezik, ezek színe, és erőssége szabadon variálható.

- A legegyszerűbb a ***Lamp*** típusú fényforrás. A tér minden irányába azonos erősségű konstans fényt bocsát ki, mintha egy állítható színű, és erősségű gyertya lenne.
- Az ***Area*** típusú fényforrás csak adott irányban bocsát ki fényt, így akár irányíthatjuk is.
- A ***Spot Light*** az ***Areahoz*** hasonló, de sokkal jobban irányítható fény. Leginkább egy reflektorhoz hasonlítható. Az egyetlen típus amely képes a ***Buffered Shadowingra***.
- A ***Sun (Nap)*** pontosan arra jó, amire elsőre gondolunk. Napot „szimulálhatunk" vele. Érdekessége, hogy teljesen mindegy, hogy hol van a fényforrás a térben az összes objektumot megvilágítja, a beállított irányból.
- A ***Hemi Light*** a ***Sun***-hoz hasonló, a fény azonban szórtabban érkezik. Talán egy felhős naphoz lehetne legjobban hasonlítani. Mivel szórt fény, így árnyékot egyáltalán nem tudunk vele előcsalni, cserébe viszont nagyon egyszerűen kezelhető.

Az *5. ábrán* egy ***Lamp*** panelt láthatunk. A fényforrás tulajdonságait tudjuk itt állítani. A kép egy ***Hemi Light*** panelja, más típusú fényforrásnál esetleg több beállítási lehetőség is van. Az ***Energy*** a fényforrás erősségét jelenti, az ***R***, ***G***, ***B***, tulajdonságok pedig a fény színére vonatkoznak.



■ *6. ábra* A „világ" beállításai

Baljós árnyak

A ***Blenderben*** alapvetően kétféleképpen tudunk árnyékokat elővarázsolni. Az első a sugárkövetéses módszer (raytracing) használata. Ehhez a rendereléskor engedélyeznünk kell a raytracing-et a ***Render Panel Ray*** gombjával, majd meg kell mondanunk a fényforrásunknak a ***Shadow and Spot*** panelen, hogy mi ***Ray Shadow***-ot szeretnénk. A ***Hemi Light*** az egyetlen fényforrás, amely nem képes a ***Ray Shadowingra***.
A ***Raytracing*** azonban lassú és időigényes (persze annál jobban néz ki). Lehetőségünk van úgynevezett ***Buffered Shadow*** használatára is, ami az árnyékok kiszámításának egy gyorsabb és egyszerűbb módja. Egyszerűbb objektumoknál szinte semmi különbség nincs a ***buffered*** és a ***traced shadow*** között. ***Buffered Shadow***-ot előállítani azonban csak a ***Spot Light*** képes.
A ***Shadow and Spot*** panelen találjuk a ***Ray Shadow*** gombot – illetve ***Spot Light*** esetében a ***Buffered Shadow*** gombot, és számos beállítást, például a ***Spot Light*** fényének szögét (***SpotSi***), élességét (***SpotBl***), és egy érdekes effekt – a ***Halo*** – intenzitását (***HaliInt***). Az ***OnlyShadow*** gombbal csak az árnyékot jelenítjük meg, a fényt nem. Így megtehetjük, hogy míg egy másik fényforrással világítjuk meg a testet, egy ***Spot*** lámpa árnyékát jelenítjük meg.

„A Világmindenség, meg minden"

Bizony néhány kép renderelése után könnyen megunhatjuk a kék háttérszínt, és azon gondolkodunk, bárcsak lehetne valami más háttérszínt, esetleg háttérképet választani. Lehet.
A *6. ábrán* a háttér beállításait tartalmazó gombokat láthatjuk.
A ***Preview*** panelen három gomb található:

- ***Blend*** – Ha ez az opció be van kapcsolva, a horizont, és az ég színe is külön állítható, a háttér egy átmenet lesz.
- ***Real*** – Kikapcsolt állapotban a horizont a kamerához lesz viszonyítva (döntött kameránál is vízszintes lesz a horizont), bekapcsolva a globális koordinátarendszerhez viszonyítunk.
- ***Paper*** – Ezzel az opcióval az éghorizont-ég színátmenetet kapjuk végeredményként. A horizont lesz középen.

A ***World*** panelen a tulajdonképpeni ***Horizont*** és ***Ég*** (***Horizon*** illetve ***Zenith***), illetve a környező fény (***Ambient Light***) színét állíthatjuk.
A ***Mist/Stars/Physics*** feliratú panelon található ***Mist*** opcióval egyfajta ködöt szimulálhatunk. Egész egyszerűen a ***Blender*** minden képpontot összevet a háttérszínnel, úgy hogy a távolabb első részeknél a háttérszín nagyobb hangsúlyt kap. Beállíthatjuk, hogy a kamerától milyen távolságra kezdődjön a köd, hol fejeződjön be, illetve a magasság szerint is számolhatunk.
A ***Stars***-t bekapcsolva csillagokat varázsolhatunk a világunkba.
A jobb oldali panelen textúrázhatjuk is a hátteret, így tetszőleges képet is használhatunk. Ezzel egy későbbi számban részletesebben is foglalkozunk. A legegyszerűbb azonban a renderelésnél az ***Output*** panelen egy képet beállítani backbufferként.
Azóta a billentyűzetem mellé kaptam több számítógépet is... Sőt, billentyűzetet is. De a régi még most is megvan, csak átalakítottam dvorak kiosztásúvá. Már többször megmentett. Igaz, hogy ma is néhány lépéssel közelebb kerültünk a célunkhoz, de még koránt sem értük el azt. Nem írtam még például a textúrázás mikéntjéről, és a megfelelő anyagok kiválasztásáról sem. Ezt a hiányosságot igyekszem pótolni a következő hónapban, az olvasóknak pedig sikeres renderelést kívánok.

A YafRay

A ***YafRay*** (***Yet Antoher Free Raytracer***) egy – a ***Blendertől*** különálló – ingyenes, sugárkövetéses technikát alkalmazó renderelő program. Nem a ***Blender*** része, de képes együttműködni vele. Egész pontosabban a ***Blender*** képes a ***YafRay*** bemenetéül szolgáló ***XML*** fájl előállítására. Segítségével a ***Blenderben*** megtervezett jeleneteinket egy kattintással fotorealisztikussá varázsolhatjuk. A program ingyenes, a ➲ ***http://www.yafray.org*** címen letölthető. Érdemes néhány pillantást vetni az oldalon lévő galériára is.

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.

KAPCSOLÓDÓ CÍMEK

- ➲ http://www.blender3d.org/
- ➲ http://en.wikibooks.org/wiki/Blender_Tutorial_Links_List
- ➲ http://www.yafray.org/

3D ábrázolás – PoVRay (3. rész)

Az előző részekben már használtunk néhány tömör testet, főként gömböt, amely mind közül az egyik legegyszerűbb. A gömbhöz hasonló egyszerű testeket primitíveknek nevezzük, ugyanis ezek az összetett testek kiinduló alkatrészei.

A *PoVRay* a testeket másképp készíti és használja, mint az elterjedt poligon alapú 3D programok. Ezen utóbbiak esetén a testek kisebb-nagyobb háromszögekből állnak, így természetüknél fogva üregesek és közelről nézve szögletesek. A *PoVRay* matematikai módszerekkel írja le a testeket, amelyek így tömör és sima tárgyakként viselkednek. Ennek főleg a textúrák alkalmazásakor lesz jelentősége. A tömör tárgyak nagyon sok esetben jól jönnek, de vannak esetek, amikor érdemes üreges testeket alkalmazni: a *PoVRay* azonban erre is képes.

Sík

A való világ tárgyai általában véges méretűek, vannak azonban olyan problémák, amelyeket végtelen kiterjedésű tárgyakkal leszünk képesek megoldani. Ilyen probléma például az égbolt, a sík föld vagy a tenger ábrázolása. Erre a `plane` alkalmas, amely egy végtelen síkot prezentál (*1. ábra, pov27.pov*):

```
plane
{
 y, 0
 texture{
  pigment{
   color White}}}
```

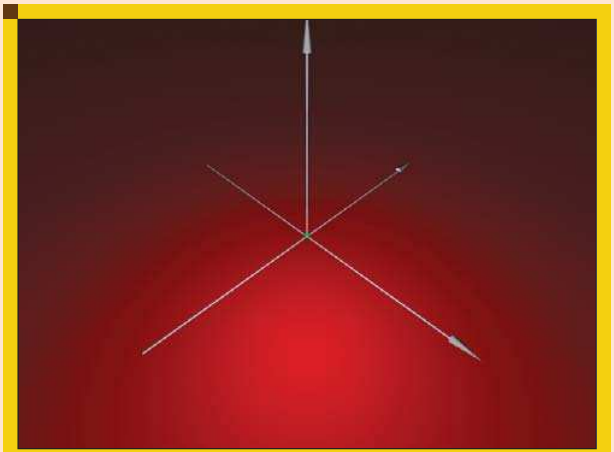
A sík egyik fontos paramétere a sík ***normálvektora***, amely a síkra merőleges és az origóból indul. Könnyedén tudjuk helyettesíteni a *PoVRay* belső neveivel, ugyanis az y név a `<0,1,0>` vektort fedi, s az ***x***, illetve a ***z*** is ezzel azonos módon képezhető. Ha ***y*** normálvektort adunk meg, akkor a létrejövő sík az ***x*** és a ***z*** tengely által meghatározott helyen jön létre.

```
plane
{
 y, -1
 texture{
  pigment{
   color White}}}
```

A síkot el tudjuk tolni a normálvektora mentén (*2. ábra, pov28.pov*), ehhez csupán a normálvektor után megadott számot kell a szükséges értékre módosítani, amely az origótól való távolságot határozza meg. A normálvektorral és az origótól való távolsággal tetszőleges síkot, illetve síkok kombinációját el tudunk készíteni. A *3. ábrán* (*pov29.pov*) három olyan síkot készítettünk el, amelyek egymással 60 fokos szöget zárnak be.

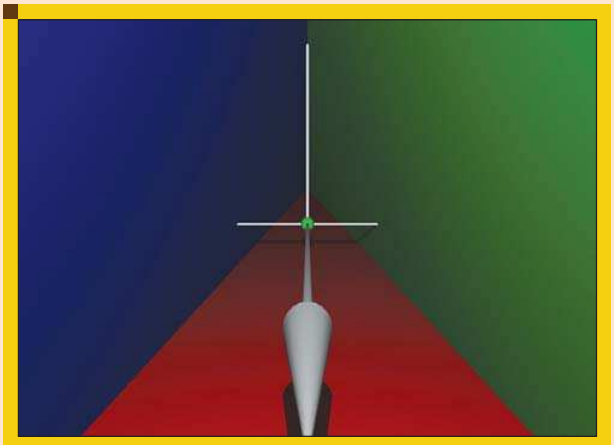
Doboz

Gyakori test a téglalap, vagy más néven a doboz, amelyet a két átellenes sarokpontjával tudunk meghatározni (*4. ábra, pov30.pov*).

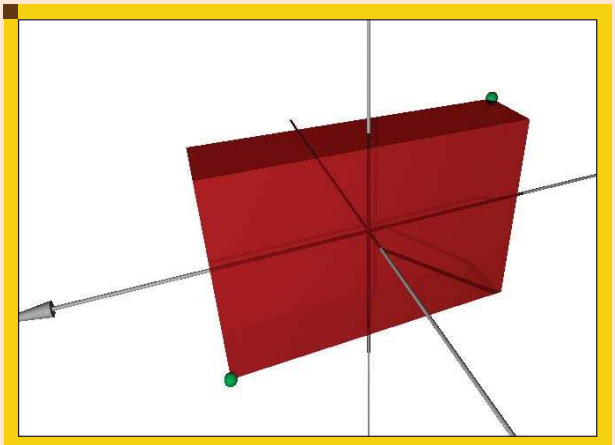


1. ábra A sík elhelyezkedése a koordináta rendszerben

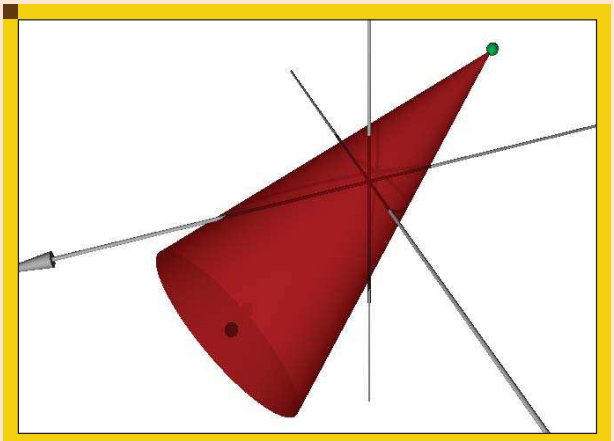
2. ábra A sík pozíciója



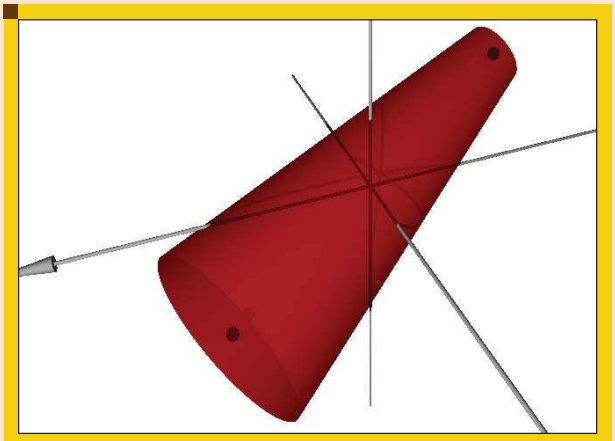
3. ábra Három sík, háromszögben



4. ábra Áttetsző téglatest



5. ábra Áttetsző kúp



6. ábra Áttetsző csonka kúp

Az ábrán egy vörös színű, félig átlátszó téglalapot láthatunk, amely a két meghatározó sarokpontjában egy-egy zöld gömböt tartalmaz.

```
box{
 <-5,3,-1>, <5,-3,1>
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Bárhova is tesszük a sarokpontokat, a téglalap élei mindig párhuzamosak valamelyik tengellyel, így külön „trükköznünk” kell a megfelelő pozícióba forgatáshoz (de erről majd egy későbbi részben lesz szó).

Kúp és csonkakúp

Egy kúpot három paraméterrel tudunk meghatározni: a bázispontjával, a tetőpontjával és az alapjának sugarával (*5. ábra, pov31.pov*).

```
cone{
 <5,-3,1> 3, <-5,3,-1> 0
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Ha csonka kúpot akarunk előállítani, akkor a tetőpontjánál nullától különböző sugarat kell írnunk (*6. ábra, pov32.pov*).

```
cone{
 <5,-3,1> 3, <-5,3,-1> 1
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

A kúp lehet üreges is, ha az `open` kulcsszót a megfelelő helyre írjuk, s ekkor a két lezáró kör alakú lemez nem kerül ábrázolásra (*7. ábra, pov33.pov*).

```
cone{
 <5,-3,1> 3, <-5,3,-1> 1
 open
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Henger

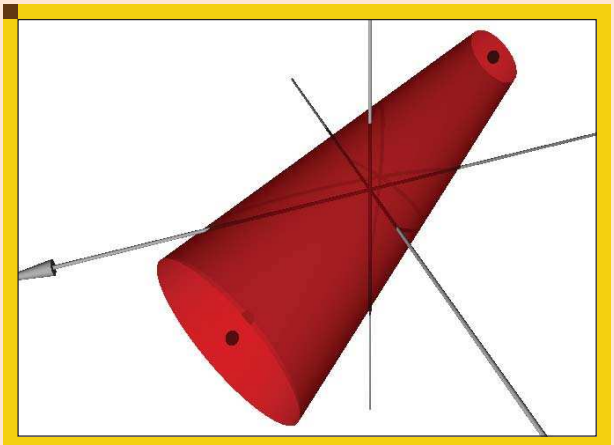
A henger teljesen azonosan kezelhető, mint a kúp, mivel a PoVRay szempontjából egy speciális csonka kúp (*8. ábra, pov34.pov*).

```
cylinder{
 <5,-3,1>, <-5,3,-1>, 2
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

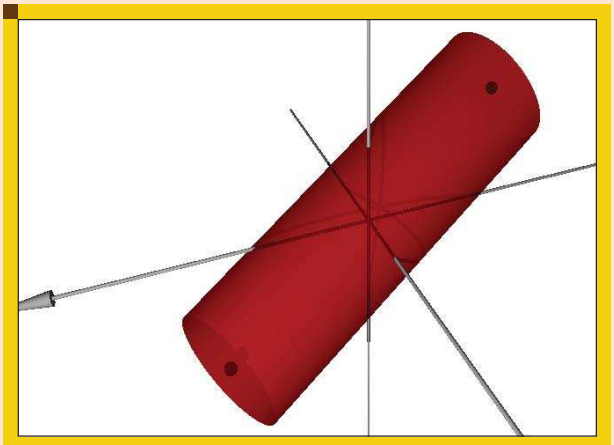
A henger lezáró korongjait az *open* kulcsszóval tudjuk eltávolítani.

Gömb

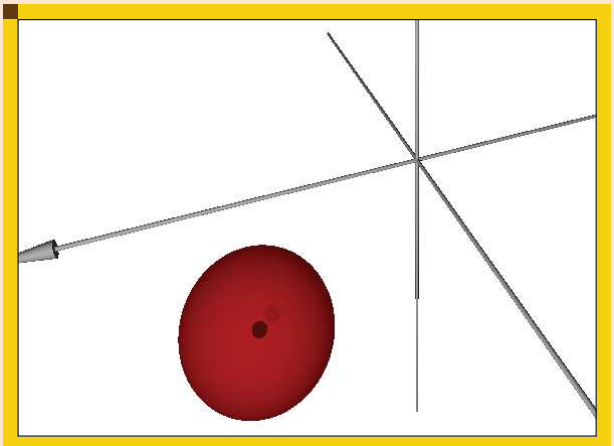
A gömb leírása a legegyszerűbb az összes test közül, a középpontját és a sugarát kell megadnunk (*9. ábra, pov35.pov*).



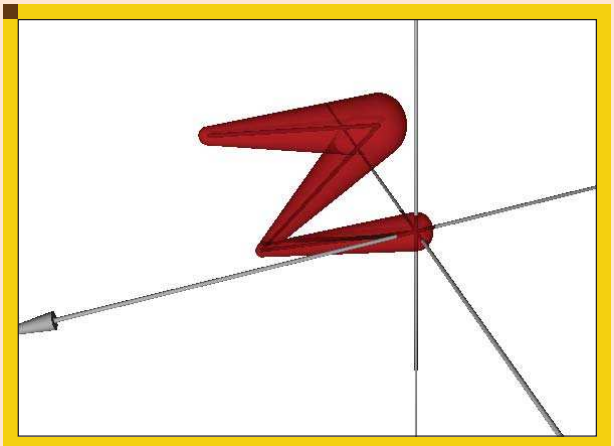
7. ábra Áttetsző üreges csonka kúp



8. ábra Áttetsző henger



9. ábra Áttetsző gömb



10. ábra „Suhanó" gömb, linear_spline útvonallal


```
sphere{
 <5,-3,1>, 2
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

„Suhanó" gömb

A gömb „egyenes ági" leszármazottja egy olyan test, amelyet a ***PovRay*** a gömb útvonalából és kontúrjából épít fel. Így tudunk olyan rudat készíteni, amelynek a két vége nem egy sima körlemez, hanem félgömb.
Ha több sarokpontot is megadunk, akkor a kontúr híven követi ezt az útvonalat (*10. ábra, pov36.pov*).

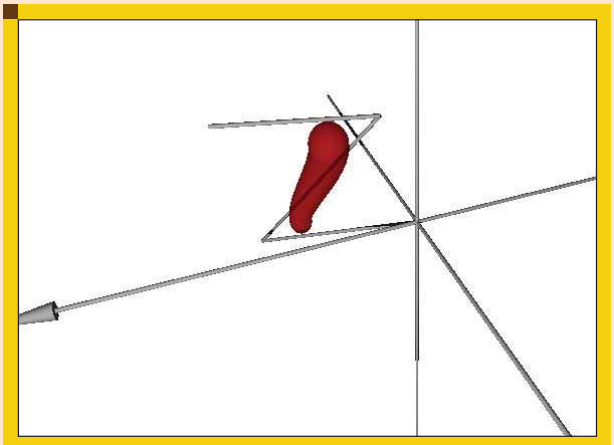
```
sphere_sweep{
 linear_spline
 4,
 <0,0,0>, 0.5,
 <4,-2,-4>, 0.2,
 <1,3,0>, 0.7,
 <5,3,-1>, 0.2
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Számomra érthetetlen okból előre meg kell adnunk a sarokpontok számát, amely alapján a program megfelelő számú térbeli pontot és sugarat vár. Ezek előtt meg kell adni a sarokpontok közötti útvonalat leíró „egyenletet" is, amely igazából három megadható kulcsszóra egyszerűsödik számunkra (a konkrét megvalósítás a programozók dolga volt):

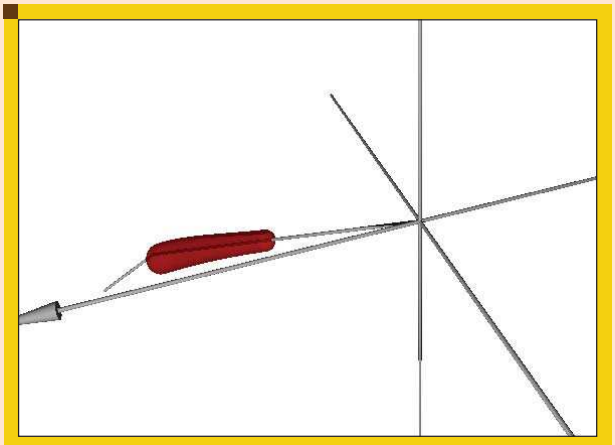
`linear_spline`, `b_spline`, illetve `cubic_spline`.

Az első útvonalra már láttunk példát, ekkor a sarokpontokat egy térbeli egyenes mentén köti össze a program és a sarokpontokba képzelt gömbök sugara egyenletes átmenettel jelenik meg.

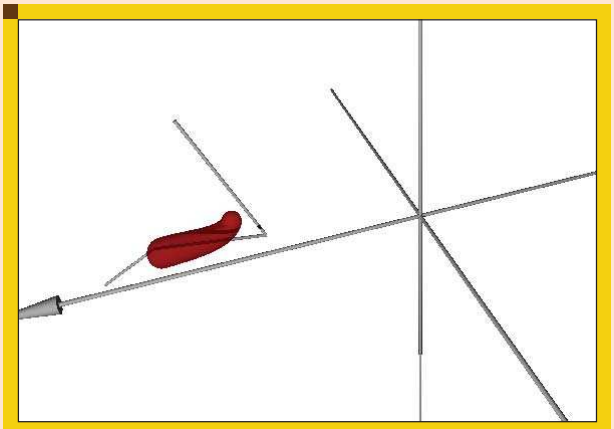
Ha a `b_spline` útvonalat választjuk, akkor a „suhanás" közel sem az egyenesek által meghatározott pontokon át történik. A létrejövő test a sarokpontokat nem érinti, hanem „tehetetlenül" mozog a sarokpontok irányába.
A *11. ábrán* (*pov37.pov*) látható testet kapjuk abban az esetben, ha a sarokpontok között a törések túl éles szögben történnek, így a „suhanás" túl nagy tehetetlensége okán nem tudja elérni a sarokpontokat.
Kicsit laposabb szögek esetén az eredmény sokkal látványosabb (*12. ábra, pov38.pov*). A szemfülesek észrevehetik, hogy az így létrejövő test sokkal rövidebb, mint az egyenes mentén mozgó gömb által létrehozott test. Ennek oka, hogy a `b_spline` (és a `cubic_spline`) az első és az utolsó sarokpontot a kezdő-lezáró „tekeredés" meghatározására használja. Ha a 12. ábrán látható útvonal első sarokpontját



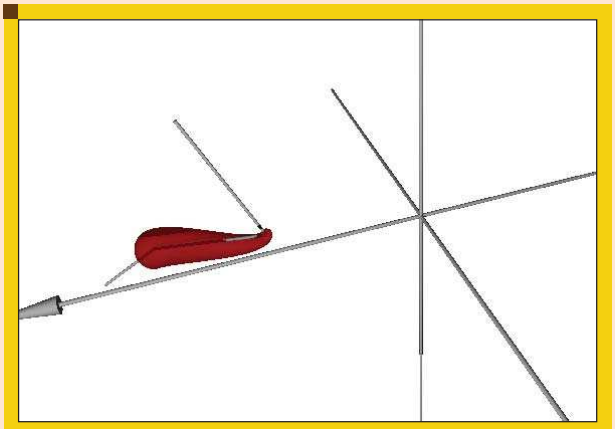
11. ábra „Suhanó” gömb, b_spline útvonallal és éles szögekkel



12. ábra „Suhanó” gömb, b_spline útvonallal és lapos szögekkel



13. ábra „Suhanó” gömb, b_spline útvonallal és éles kezdő szöggekkel



14. ábra „Suhanó” gömb, cubic_spline útvonallal és éles kezdő szöggekkel

merőlegesre állítjuk (*13. ábra, pov39.pov*), akkor látható, hogy a kezdőpont elmozdult az első sarokpont irányába.
A `cubic_spline` útvonal teljesen hasonló a `b_spline` által meghatározott útvonalhoz, csak az útvonal minden esetben érinti a közbenső sarokpontokat (*14. ábra, pov40.pov*).

„Cuppanós” testek

Kiválóan alkalmazható kémiai bemutatóhoz, illetve olyan esetekben, amikor gömbök vagy téglatestek lépnek egymással kölcsönhatásba. A „cuppanás” azt takarja, hogy ha a leírt testek közel kerülnek egymáshoz, akkor a felületük egyesül, mintha folyadékcseppek lennének. Minél közelebb kerülnek, annál jobban látszik ez az egyesülés. Példaképpen nézzünk meg egy vízmolekulát, amely egy oxigén atomból és két hidrogén atomból áll (*15. ábra, pov41.pov*).

```
blob{
threshold 0.6
sphere{<0,0,0>,4,1
texture{pigment{color Blue}}}
sphere{<1.15,-1.63,0>,1,1
➥texture{pigment{color Red}}}
sphere{<1.15,1.63,0>,1,1
➥texture{pigment{color Red}}}
scale 2}
```

Tórusz

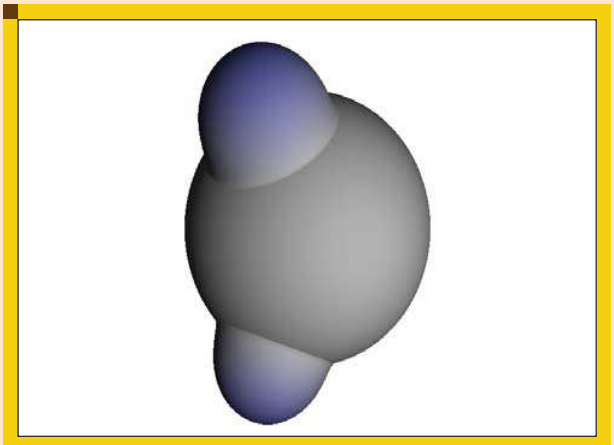
Ha egy gömböt végigvonszolunk egy kör alakú útvonalon, akkor egy tóruszt kapunk eredményül. Ezt megtehetjük a „suhanó” gömb segítségével is, de jobban járunk ha a PoVRay saját tóruszát használjuk (*16. ábra, pov42.pov*).

```
torus{
 7, 1.5
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

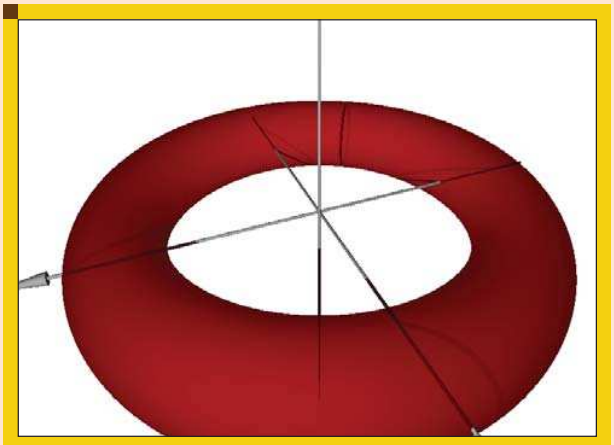
A tórusz mindig az origó által meghatározott középponttal jön létre, és két paraméter határozza meg: a fősugár (***major radius***) és a melléksugár (***minor radius***). A fősugár a kör alakú útvonal origótól való távolságát határozza meg, a melléksugár pedig a körbevonszolt gömb sugarával egyezik meg. Ha máshova és másképp szeretnénk tóruszt létrehozni, akkor koordináta transzformációval át kell helyeznünk, illetve át kell alakítanunk.

Magasságmező

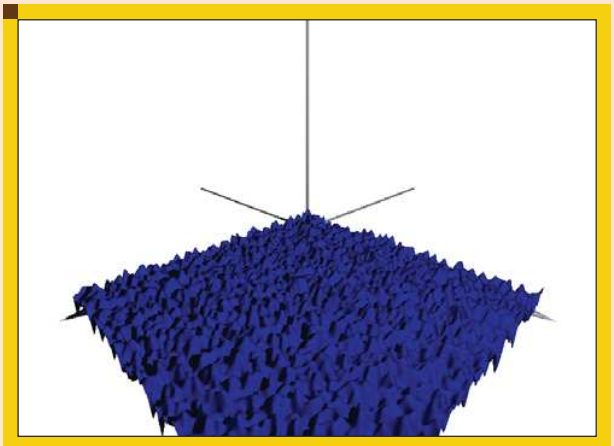
Ha hegyes-völgyes képet szeretnénk elkészíteni, azt nehezen tudjuk matematikai egyenletekkel leírni (bár nem lehetséges), így a ***PoVRay*** is csak háromszögekből képes ezt előállítani. Sok ezer háromszöget viszont nehéz megfelelő pozícióba helyezni, a magasság-mező használatával ez felesleges munka is lenne.



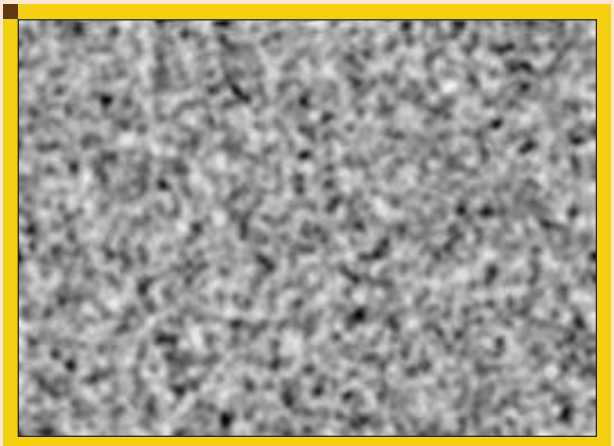
15. ábra „Cuppanó" gömbök, vízmolekula



16. ábra Tórusz



17. ábra Magasság mező



18. ábra A magasság mező forrása

A magasság-mező használatához kerítenünk kell egy olyan állományt, amely szürkeárnyalatos képet tartalmaz. A fekete területek a nulla szintet, a fehér területek az egységnyi magas szintet jelentik. A kettő közötti árnyalatok a világosságukkal arányos magasságot jelképeznek. Alapesetben a `<0,0,0>` és `<1,1,1>` pont által közbezárt téglalapon belül jön létre ez a test, ezért megfelelő transzformációkkal el kell tolnunk a kívánt pozícióba.

```
height_field{
 png
 "pov43field.png"
 smooth
 scale 10*x
 scale 10*z
 texture{
  pigment{
   color Blue}}}
```

Érdekes trükköket tudunk elérni, ha egy fényképet alkalmazunk a mezőhöz (*19-20. ábra, pov44.pov*).

Forgatott felszín (Surface of Revolution)

Gyakran találkozunk olyan testekkel, amelyeket egy meghatározott sík felület valamely tengely körül megforgatott kontúrja határoz meg. Tipikus és szép példa erre a serleg, amely ilyen forgatás eredménye (*21. ábra, pov45.pov*).

```
sor{
 11,
 <0,0>
 <7,0>
 <6,2>
 <2,3>
 <2,6>
 <3,7>
 <2,8>
 <2,14>
 <6,16>
 <7,21>
 <5,21>
 texture{
  pigment{
   color Blue}}}
```

A megforgatás mindig az ***Y*** tengely körül történik, és speciális ***U-V*** koordinátákkal kell megadni a pontokat. Gyakorlatilag az ***U*** az ***X*** irány, a ***V*** az ***Y*** irányt jelöli. Megadhatunk éles vonalakat is, mert a létrejövő felület simított és lekerekített lesz, így nem tűnnek fel durva élek a képen. Természetesen ezáltal éles körvonalakat nehezebb lesz létrehozni.

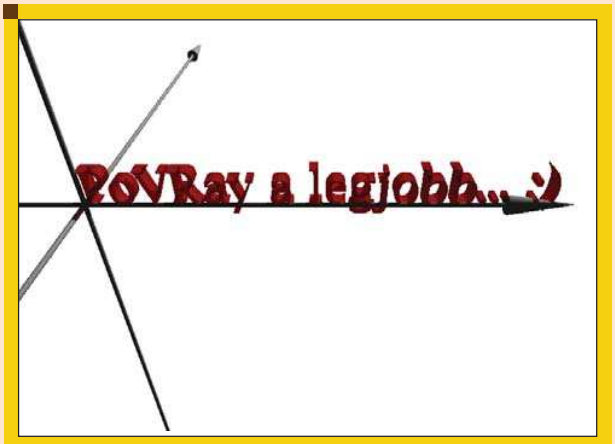
Szöveg kiírása

Néha szükséges szövegeket írni a ***3D*** világunkba, ezt is megtehetjük, készíthetünk betű kinézetű testeket, amelyek az írás szerint követik egymást (*22. ábra, pov46.pov*).

19. ábra Magasság mező

20. ábra A magasság mező forrása

21. ábra Megforgatott sík



22. ábra Szöveg kiírása

```
text{
 ttf "LucidaBrightRegular.ttf"
"PoVRay a legjobb... :)",
➥0.3, 0
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

A szövegnek négy paramétere van, amelyek rendre a betűtípusfájl elérési útja (ezt csak TTF lehet), a kiírandó szöveg, a szöveg térbeli kiterjedése, illetve a betűk közötti hely beállításához egy kétdimenziós vektor (ez utóbbi általában nulla). Ha ritkítani szeretnénk a betűket, akkor a `0.1*x` eltolás egész jó érték; ha sűríteni, akkor a `-0.1*x` a jó választás. Tudunk ferdén is írni, ha például a `<0,0.1,0.1>` eltolást használjuk, ekkor a karakterek *x* és *z* irányban `0.1` egységnyivel mozognak. (*23. ábra, pov47.pov*).

További testek

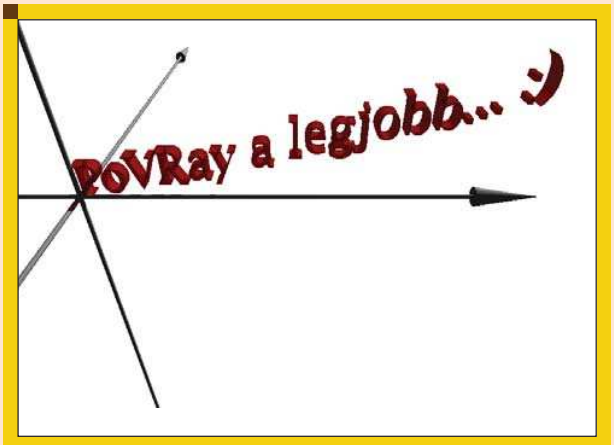
A ***PoVRay*** a felsorolt testeken kívül még egyszer ennyi lehetőséget rejt, azonban ezek használata kissé nehézkes és fárasztó lehet (fraktálok, különféle végtelen kiterjedésű testek, poligonok és alakzatok). Érdemes fellapozni a dokumentációt ezen testek megismerése végett, de csak akkor ajánlatos, ha a már ismert testekkel nem tudjuk ábrázolni az elképzelt világot.

Testek közötti halmazműveletek

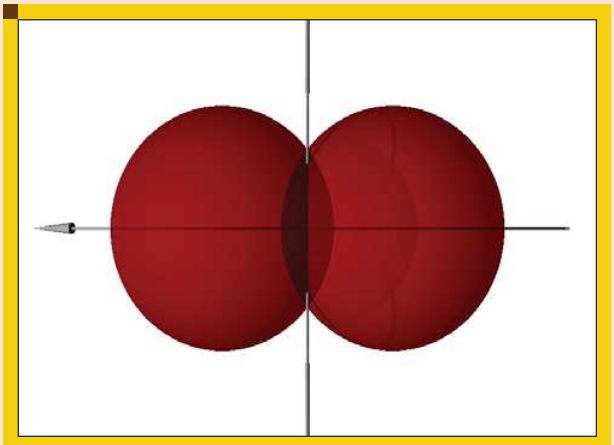
Önmagukban ezek a testek semmire sem jók, ha nem tudunk közöttük bizonyos átalakító műveleteket végezni. Ha ezekből a „primitív” testekből építkezünk, akkor szinte minden valós tárgyat el tudunk készíteni, amire szükségünk van.
Alapvetően a metszetet, a különbséget illetve a egyesítést tudjuk használni, azonban az egyesítésből két különböző módszert ajánl a ***PoVRay***. Az egyik egyesítés inkább technikai, ugyanis a testeknek minden része megmarad, az is, ami egymásba lóg (*24. ábra, pov48.pov*).

```
union{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

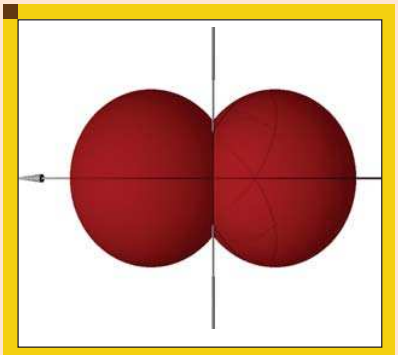
A másik típusú egyesítés a kontúrok mentén egyesíti a testeket, így nem marad egymásba lógó részük, viszont ez erőforrásigényesebb. Akkor célsze-



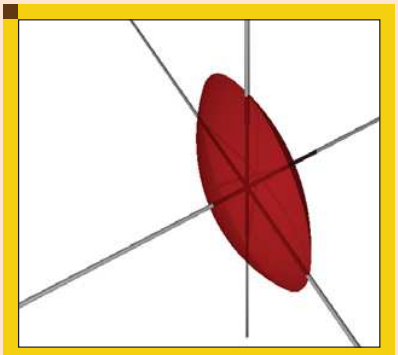
23. ábra Szöveg kiírása kis offszettel



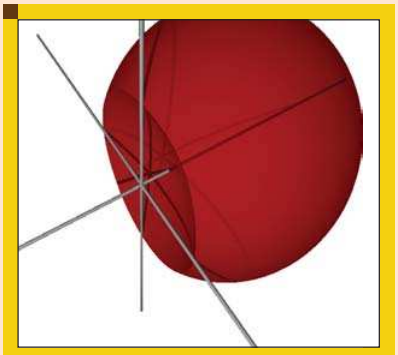
24. ábra Két gömb egyesítése, union



25. ábra Két gömb egyesítése, merge



26. ábra Két gömb metszete



27. ábra Két gömb különbsége

rű használni, ha átlátszó vagy áttetsző testeket használnunk (*25. ábra, pov49.pov*).

```
merge{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

A *PoVRay* képes kettő vagy több test közös részét kiemelni, ezt nevezzük metszetnek. Ha a kettő gömb metszetét nézzük, akkor egy lencse alakú tárgyat kapunk (amelyet később a fénytöréssel és az átlátszósággal valódi lencseként is tudunk használni) (*26. ábra, pov50.pov*).

```
intersection{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

Az utolsó művelet szerint egy testből kivonjuk azt a részt, amely egy másik testtel közös: ezt nevezzük a két test különbségének (*27. ábra, pov51.pov*).

```
difference{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

A következő részben a testek különféle transzformációival fogunk tüzetesebben foglalkozni, így kényelmesen össze tudjuk állítani az elképzelt világot primitív testekből.

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

A PovRay projekt honlapja
➲ http://www.povray.org

A cikkben említett fájlok
➲ http://user.enaplo.hu/~auth.gabor/pov/

Az SVG világa (3. rész)
Alkalmazási területek és felhasználási módok

Ha fellapozzuk a Linuxvilág októberi számát, akkor felvehetjük az előző részben elínditott fonalat. Akkor különféle SVG szerkesztőkkel ismerkedtünk meg és szépséges grafikákat alkottunk velük. Néhány kulcsszó azok számára, akik elmulasztották volna: Inkscape, Sodipodi, GLIPS Graffiti. Többek közt ezekkel a szoftverekkel élhetjük ki művészi ambícióinkat, ha SVG-ben gondolkodunk. Ebben a felvonásban kevésbé lesz középpontban az alkotótevékenység, annál inkább annak felhasználási lehetőségei.

Szerencsés esetben a művész sem magának alkot; inkább a közönség örömére. Ki is lehet a mi közönségünk, ha *SVG*-szobrásznak csapunk fel? Elég szélesvásznú a kínálat: a webet böngészők, az mms-küldözgetők, sőt a gnome vagy a kde munkakörnyezetet használók is ide tartoznak. Most egyfelől arról lesz szó, hogy technikailag hogyan tudjuk az elkészült grafikánkat a nagyközönség elé tárni a különböző csatornák segítségével, másfelől pedig az ekkor felmerülő buktatókat vesszük szemügyre. Vágjunk bele!

Web

Maga a web ugyan ósdi találmány (1989), a webre szánt vektorgrafika csak pár éve terjedt el, az *SVG* pedig még csak most van elterjedőben. Mégis a web az a felület, ahol leginkább célba találhat a munkánk, hiszen a felhasználók számát tekintve az *MMS* például sehol nincs a webhez képest, meg teljesen másra is való. Azonban kis túlzással ahány böngésző létezik, annyi lehetősegre kell felkészülnünk. A piacvezető böngésző nem rendelkezik beépített támogatással ahhoz, hogy megjelenítse *SVG*-alkotásainkat, a karakteres felületű böngészők eleve nem jönnek számításba, ha grafikáról van szó és még sokféle eshetőség keseríti az *SVG*-ben utató webmesterek életét. A helyzet ugyan rossznak tűnik, de a weben ez természetes. Mindig voltak és lesznek átmeneti időszakok, míg az újdonságokat csak egy kisebb csoport használja, amíg el nem terjednek. Láttuk, hogy az *SVG*-nél minden esély adott, hogy elterjedjen, a nagy cégek támogatása sokat nyom a latban.

A kulisszák mögött

Hogyan is varázsoljunk egy weboldalra *SVG*-t ? Nézzünk meg ehhez egy komplett *XHTML* oldalt, amin csak egy *SVG* fájl van.

```
<?xml version="1.0"
➥encoding="ISO-8859-2"?>
<!DOCTYPE html PUBLIC
➥"-//W3C//DTD XHTML 1.0
➥Strict//EN"
➥"http://www.w3.org/TR/xhtml1/
➥DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/
➥1999/xhtml" xml:lang="en"
➥lang="hu">
  <head>
    <title>SVG fájl</title>
  </head>
  <body>
<p>
Egy kis szöveg!
<object data="rajz.svg"
➥type="image/svg+xml"
➥style="width: 540px; height:
➥200px;">
    <div>Helyettesítő
    ➥szöveg</div>
</object>
Még egy kis szöveg.
</p>
 </body>
</html>
```

Akitől nem állnak távol a jelölőnyelvek, mint a *HTML* vagy az *SVG*, annak érthető a fenti szabványos *XHTML* kódrészlet. Csupán annyi történik, hogy megadjuk a fájlunk nevét es beszúrjuk arra a helyre a dokumentumon belül, ahol ezek a sorok vannak. A fenti részlet egy teljes *XHTML* oldal volt, ebből pusztán az `<object>` címke az, ami gondoskodik arról, hogy a böngésző megjelenítse a grafikánk. Itt a

```
style="width: 540px; height:
➥200px;"
```

részben a szélességet és a magasságot adhatjuk meg képpontokban. Mihez is kezdenek ezzel napjaink webböngészői? Piaci dominancia szerinti növekvő sorrendet nézzük, majd a végén kiderül, hogy miért.

1. táblázat *SVG-megfelelő telefonok*

Motorola	C980, E1000, V3X, V980, V1050
NEC	802
Nokia	6265, 6280, 7710, N70, N90, N91
Panasonic	MX6, MX7, SA6, SA7, VS3, VS7
Sagem	myX-8, myV-76
Sanyo	S750
Sharp	V601SH, V602SH, V603SH, V703SH, 802, 902, V903SH
Siemens	CX65, S65, M65, C65, SF65, SK65, SL65, CFX65
Sony Ericsson	D750, F500, K300, K500, K508, K600, K608, K700, K750, S600, S700, S710, V600, V800, W600, W800, Z500, Z520, Z800
Toshiba	V902T

Konqueror

A *KDE* alapértelmezett böngészője meglehetősen jó támogatással bír, saját *SVG* pluginja van, melyet debian alatt a *ksvg* csomagban találunk meg. Más teendőnk nincs is, csupán ezt a csomagot kell fel telepíteni. Onnantól kezdve a *Kulisszák mögött* részben látható kódrészlet életre kel. Mellesleg egyazon megvalósítással két rést tömtek be: a *KDE* asztali környezet szintén ezt használja.

Opera

Az *Opera* számos felhasználócsalogató akcióval rukkolt elő mostanában. A felhajtás úgy kezdődött, hogy a ➲ *http://www.opera.com/swim/* címen beharangozták, hogy a vezérigazgató(*CEO*) átúszik *Norvégiából* az *USA*-ba, ha egymillióan letöltik az *Opera 8.0*-t. Ez a megmosolyogni való hír inkább csak arra volt jó, hogy a böngésző nevét ne felejtsük el teljesen, viszont az már egészen más súlyú, hogy idén szeptember 20-án bejelentették, hogy az asztali számítógépekre szánt verzió ingyenessé válik. Így aztán a szabad szoftver világában ez a zárt program várhatóan nagyobb szeletet fog kihasítani a böngészőpiacból. Ezeknél az eseményeknél még érdekesebb, hogy természetesen támogatja az *SVG*-t. Semmilyen pluginre nincs szükség, a 4 Mb-os telepítőcsomag az *SVG* támogatást is tartalmazza.

Mozilla-család

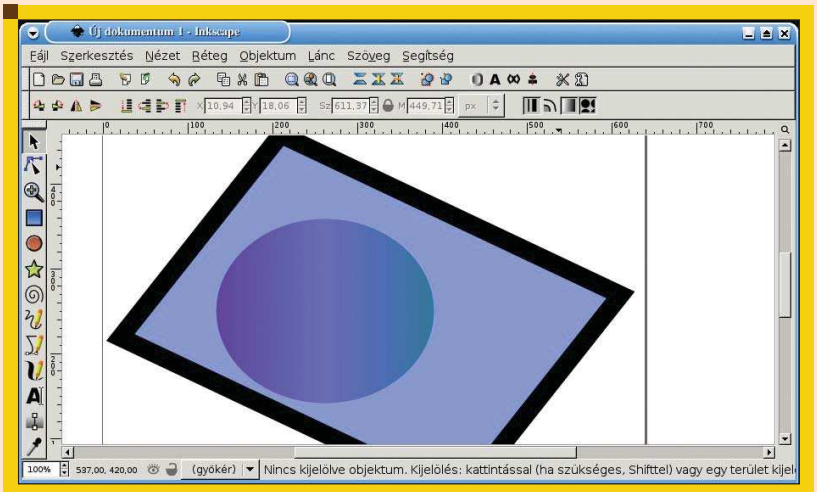
A *Mozilla* által készített böngészők gőzerővel dolgoznak azon, hogy beépített támogatással rendelkezzenek. A ➲ *http://www.mozilla.org/projects/svg/* címen nyomon követhetjük a projekt állását. Az *SVG 1.1* teljes megvalósítását tűzték ki célul, azonban a cél még messze van. A legtöbb letölthető csomagban nincs is engedélyezve az általuk készített *SVG* támogatás, de a projekt weboldaláról letölthetők azok a terjesztések(nightly builds), ahol kipróbálhatjuk a legújabb fejlesztéseket *SVG* fronton. Ezek technikailag érdekes ügyek, de nem számíthatunk arra, hogy látogatóink a legújabb verziót használják. Ezért aztán a konklúzió nem túl kecsegtető: az *Adobe SVG Viewere* remekül ellátja azt a feladatot, amit elvárunk *SVG* fronton egy böngészőtől.

Internet Explorer

A felsorolásunkból ez az egyetlen böngésző, amely nem gondoskodik házon belül valamilyen formában az *SVG*-megfelelőségről (➲ *http://svgwhiz.com/wiki/index.php?title=Internet_Explorer*). Az *Adobe SVG* plugin itt is megfelelő választás. Sajnálatos, hogy az innovatív webes technológiákat egy piaci túlsúlyban lévő böngésző súlytalannak tekinti.

A böngészők átka

Mint láthattuk, nem elég, ha beágyazzuk az *SVG* fájlt a weboldalba, a felhasználóbarát felfogás megköveteli, hogy segítsünk a látogatónknak, hogy valóban azt lássa, amit mi terveztünk. Mivel a *Mozilla*-féleségek és az *Internet Explorer* gyakorlatilag uralkodók böngészőfronton, érdemes elhelyezni az *Adobe* weboldalára egy hivatkozást(➲ *http://www.adobe.com/svg/viewer/install/main.html*), ahonnan letölthetik a plugint. Ezután semmi más teendőnk nincs, mint várni a látogatókat. Még szerencse, hogy a látogatók érkezhetnek *Windows*, *Mac OS*, *Linux* és *Solaris* alól is, az *Adobe* plugin éppúgy ajánlható nekik. A végeredmény nem túl szabad szoftveres ízű, de



1. ábra A kép még az Inkscape-ben

a weben ma még nem lehet megkerülni azt a tényt, hogy a ➲ *http://www.w3schools.com/browsers/browsers_stats.asp* aránylag kedvező arányait figyelembe véve is még a weben szörfölők 70%-a ***Internet Explorert*** használ. Ráadásul a *w3schools.com* egy szakmai oldal, valószínűleg a teljes kép ennél még egyhangúbb a piacvezető előnyére. Apropó *w3schools*: a ➲ *http://www.w3schools.com/svg/svg_examples.asp* címen remekül le tudjuk tesztelni a változatos minőségű *SVG* megvalósításokat.

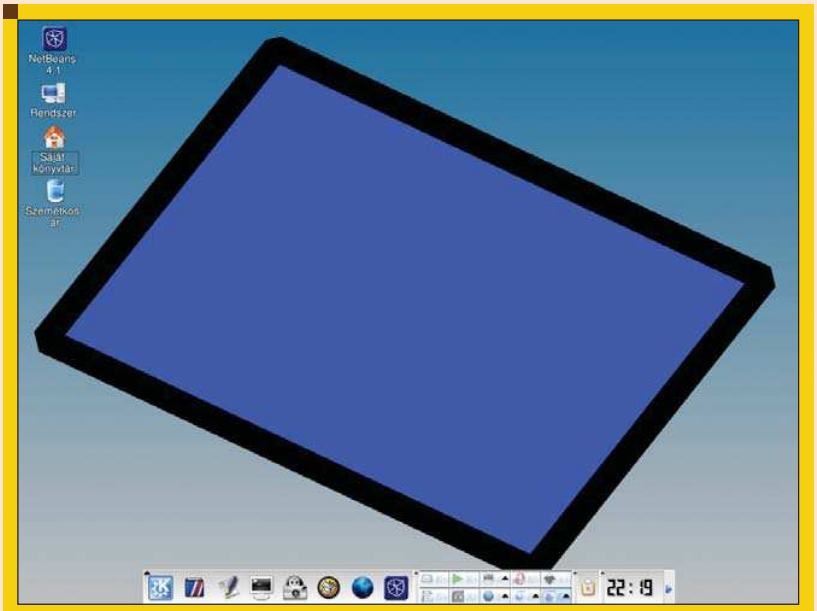
MMS

Mint azt a sorozatunk legelején említettük, az egyre fejlődő mobiltelefonok (vagy inkább mobiltelefon-bőrbe bújtatott parányi számítógépek) némelyike ismeri az *SVG* szabványt. Így aztán semmi nem áll utunkba, hogy ezt kihasználjuk. A sorozatban leírtakat követve nem meglepő az sem, hogy az*MMS* is egy *XML* fájl. Belülről:

```
<?xml version=?1.0? ?>
<smil xmlns=?http://www.w3.org/
➥2001/SMIL20/Language?>
  <head>
    <layout>
      <root-layout width="569"
      ➥height="286" />
        <region id="Image"
        ➥height="100%"
        ➥width="100%"
        ➥z-index="1"
left="0%" fit="meet" />
      </layout>
  </head>
  <body>
    <par dur="40s">
      <img id="i1"
      ➥src="kep.svg"
      ➥region="Image" />
    </par>
  </body>
</smil>
```

Kicsit részletezzük a fentieket! A második sorból kiderül, hogy ez egy *SMIL* formátum. (Igen, az *XML* az egy formátumleíró formátum.) Az érdeklődők a specifikációt a ➲ *http://www.w3.org/TR/2005/REC-SMIL2-20050107/* címen találják. Az *SMIL* a *Synchronized Multimedia Integration Language* rövidítése, ami nagyjából annyit tesz, hogy összehangolt multimédiás gyűjtőnyelv. A szerteágazó tartalomtípusokat (hang, kép és videó) tudjuk egyetlen egységbe szervezni.
A `<root-layout>` elem határozza meg, hogy mekkora ablakban (képernyőn, például a telefon képernyője) kell majd leképezni a tartalmat. A `<region>`-nal területet jelölhetünk ki(itt éppen az egészet: 100%), amibe aztán pakolhatjuk az elemeket. A `<head>` és `<body>` felosztás végképp evidencia a *HTML*-t ismerőknek: A `<head>`-be többek közt az elrendezés és a stílusok valók, a `<body>`-ba pedig a tényleges tartalom. A `<par>` itt az angol paralell-ből jön, ugyanis ezzel a címkével kell közrefognunk azokat az elemeket, melyeket egyszerre akarunk megjeleníteni. A `dur` paraméter a megjelenítés idejét adja. Bár ezeknek a segítségével komplett kis mozikat tehetünk össze, a cikk célja csupán annyi, hogy az *MMS SVG*-vel kapcsolatos lehetőségeit áttekintsük.
Gondoltuk volna, hogy amikor a tengerparti nyaralás képeit hazaküldjük *MMS*-ben, akkor nem teszünk mást, mint egy *XML* jelölőnyelvben elkészített dokumentumot készítünk a telefonunk segítségével? *JPG* képek helyett *SVG*-t is használhatunk, ma már egyre több telefon támogatja. Igaz ugyan, hogy a tengerparti nyaralást mégis *JPG*-ben fogjuk továbbra is küldeni, hiszen a kamera által készített kép pixelgrafikus, az *SVG* pedig vektorgrafikus formátum. Viszont például üdvözlőlapok készítésére az *SVG* eszményi megoldás. Magát a képet bármelyik *SVG* szerkesztővel létrehozhatjuk (az előző lapszámban szó esett néhányról), az *SMIL* fájlt azonban jobb, ha nem kézzel állítjuk össze, mivel ez esetben nem szöveges fájlok továbbítódnak, hanem bináris állományok. Azokat pedig elő kell állítani(encode) az *SMIL* fájlból. Ehhez vagy külön programot használunk vagy rábízzuk a telefonunkra a dolgot.

2. ábra Ugyanez ikonként megjelenítve



3. ábra Akár háttérképként is használhatjuk

Bevallom, hogy egészen addig, amíg ezt a sorozatot elkezdtem írni, fogalmam sem volt arról, hogy a zsebemben lapuló telefon ismeri az *SVG Tiny*, az *SVG* leszűkített szabványát. A szerencsések a táblázatban megtalálják telefonjukat, bár a lista korántsem teljes, hiszen napról-napra jelennek meg az újabb fejlesztések a piacon. Az asztali számítógépekről infraporton vagy bluetoothon keresztül lehet áttölteni a telefonra a képet, onnantól kezdve pedig úgy viselkedik, mint bármely másik kép a telefonban: megnézhető és továbbküldhető, akár épp *MMS*-ben. Ha kényelmesebbnek találjuk a számítógépen elkészíteni az *MMS*-t, semmi akadálya. A ➲ ***http://developer.openwave.com*** címről letölthető regisztráció után egy *MMS* fejlesztői eszköztár(*SDK*). A részletes ismertetése túlmutat a cikk keretein, mindössze jó tudni, hogy létezik *UNIX* platformokra kereskedelmi eszköz erre a célra is. Ehhez hasonló szoftvereket is találunk az ➲ ***http://wam.inrialpes.fr/software/limsee2/index.php?goto=Mobile*** oldalon.

Barátságosabb Linux

A legtöbben *Linux* alatt A *KDE* és a *GNOME* asztali környezetet használják. Ezek a szoftverek is lépést tartanak a grafikai formátumok folytonos evolúciójával! Az előző részben elkészített képeket vagy bármit amit *SVG* formátumban fellelünk az interneten, fel tudjuk használni linuxos asztalunk felcicomázására. Figyeljünk csak!
Így aztán akár ikonformában, akár háttérnek felhasználhatjuk minden *SVG* fájlunkat. Semmi különlegeset nem kell tenni ehhez, az eddigi *JPG* fájlok helyett *SVG*-t állítunk be – a rendszer hagyni fogja, már ha kellőképpen friss verzió. Igaz ugyan, hogy ha nem csapunk fel téma-készítőnek, akkor leginkább csak magunk fogjuk értékelni az új, házibarkács kinézetet, de garantáltan egyedi végeredményre jutunk, ha minden ikont és a hátteret is SVG-ben állítunk össze. Ha belejöttünk a nagyüzemi termelésbe, osszuk meg a szabad szoftveres közösséggel a témát. A ➲ ***http://art.gnome.org/*** és a ➲ ***http://www.kde-look.org/*** címeken valószínűleg minden értékes munka szívélyes fogadtatásra talál! Így hát ne habozzunk, témát összeállítani jó móka, ha a végeredményt publikálni pedig annál is jobb. A puszta grafikai elemek nagy kupacánál több a téma, sőt *KDE*-re és *GNOME*-ra máshogy kell elkészíteni azt a témafájlt, ami már emészthető a programoknak. A sorozat fő irányvonalába nem igazán illeszkedik egy *Témagyártó-HOWTO*, ezért most a dekorációs lépéseknél megállunk. Hol lehet még *SVG*-t használni? Bárhol. Írjunk rá programot! Ez már ugyan nem a teljesen kezdőknek szóló feladat, de a következő számban ezt fogjuk körüljárni. Ha eddig a zöld pályán suhantunk, most a piros következik. Felvonó indul!

Novák Áron
(aaron@szentimre.hu)
BME-VIK-es gólya, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal amelyeket meg lehet szólaltatni Linux alatt.