Hírek

Hajlékony OLED kijelző

A *Universal Display Corporation (UDC)* egy szakmai konferencián bemutatta legújabb, színes *OLED* kijelzőjének mintapéldányát (*AMOLED*), mely 4 hüvelyk átlóval bír, a vezérlőelektronika nélkül pedig vastagsága alig 0,1 mm, tömege 6 gramm, szemben a normál *LCD*-s megoldásokkal, ahol vastagság mintegy tízszeres lenne, a tömeg pedig négy-hatszoros. Több technológiát is egyesítettek a siker érdekében. A kutatást részben az *Egyesült Államok Védelmi Minisztériuma* támogatja, azonban a cég partnerei között találhatjuk a *Sony*-t és a *Samsungot* is.

Mini merevlemezek mobiltelefonokba

A *Seagate* bemutatta a legújabb fejlesztésű, elsősorban mobiltelefonokba szánt merevlemezét. Az egy colos, de mégis 12 gigabájtos merevlemez 30%-al kevesebb energiát igényel, mint elődje. Mindeközben a *Cornice* piacra dobta *Cornice Dragon* termékcsalád 8 és 10 gigabájtos tagját. A család 3 gigabájtos tagja már sikeresen debütált a *Samsung* új *SGH-1300*-as telefonjában.

Ingyenes a VMware Server termékcsalád

Ingyenessé vált február 6-án a *VMware* cég *Server* termékcsaládja. Ez azonban nem jelenti azt, hogy a forráskódját is megnyitották, a mostani lépésnek köszönhetően tehát csupán olcsóbbá válik a virtualizáció. Így például egy fizikai gépen belül több operációs rendszert használhat a rendszeradminisztrátor anélkül, hogy az operációs rendszerek tudnának egymásról.

➲ *http://www.vmware.com*

OpenGL-lel gyorsított X.org

Az *X.org* hamarosan *OpenGL*-es gyorsítást kap az *Xgl* révén. Ezzel lehetővé válik a jelenleg szoftveresen megvalósított hatások hardveres megoldása. Ez a jelenlegi raszteres megoldásról a közeljövőben a vektorosra helyezi a hangsúlyt.
Bár jelenleg csupán egyetlen operációs rendszer, a *Mac OS X* kínálja ezt, a közeljövőben már az *X.org*-ban is viszontláthatjuk. A fejlesztők szerint ez nem jelenti azt, hogy drága videokártyákat kellene használni. Megmarad a kompatibilitás, vagyis a régi programok is fognak futni ezzel a megoldással.

➲ *http://www.novell.com/linux/xglrelease/*

Új eszközzel bővül a Google

Az új eszköz lehetővé teszi a két számítógép között a fájl átvitelt, cserébe azonban engedélyezni kell hogy a *Google* maximum 30 napig tárolhassa azt saját rendszerében. *Marissa Mayer*, a *Google* egyik elnökhelyettese szerint azonban a magántitok feladása a legtöbb embernek nem fog gondot okozni.

Nanocsöves festék

A *NaturalNano* kifejlesztett egy nanocsöveket tartalmazó festéket. Maguk a nanocsövek rezet tartalmaznak a belőlük készült festék pedig alkalmas például arra, hogy egy helyiséget – mozi- vagy színháztermet – leárnyékoljon az elektromágneses hullámok, így a mobiltelefonok elől is. A nanocsövek egy papírlapnál húszezerszer vékonyabbak. Az árnyékolás természetesen a másik irányba is működik, tehát egy ilyen nanocsöves festékkel kifestett szobában elhelyezett WIFI hozzáférési pontot kevésbé tudnak elérni és használni illetéktelenek. Előnye a korábbi megoldásokkal szemben, hogy a réteg vékonyságának köszönhetően kevésbé terheli a falat.

3 dimenziós böngésző Firefox alapokon

uBrowser néven bemutattak egy webböngészőt, mely a *Mozilla Gecko* motorját használja az oldalak renderelésére, a térbeli megjelenítést pedig alapvető *OpenGL* parancsokkal végzik. A *Firefoxhoz* hasonló módon kezelhető és csaknem az összes oldalt megjeleníti, amit a *Firefox 1.5* is megjelenít. Egyelőre mind a forráskód, mind a bináris csak *Windows* alá érhető el.

➲ *http://www.ubrowser.com/*

Ég veled AIBO

Nyolc év után a *Sony* végleg befejezte az *AIBO* nevű robotkutya gyártását. Az *AIBO* projektet a *Sony* nagy kísérletnek szánta, ami alapvetően nem sült el túl jól. Ebben azonban közrejátszott a termék túlzottan magas ára, mely jelenleg is 2100 euró körül van az európai boltokban. Ezért az összegért azonban már egy 500 MHz-es 64 bites *RISC* processzorral felszerelt és beépített vezetéknélküli csatolóval ellátott robot négylábút tudhat magának a vásárló.

Pingvin alakú számítógép

Az olasz *Acme Systems* bemutatta pingvin alakú számítógépét. A komplett számítógép 130 euró – a házzal együtt. Az érdekessége az adja, hogy az alaplapon – ami kb. 6x6 centi – csupán egyetlen chipet találunk. Az *Etrax 100LX* -at az *Axis* cég gyártja. A chip 4 Mbyte flash memóriát, 16 Mbyte RAM-ot, egy hálózati csatolót valamint két *USB* vezérlőt tartalmaz. A számítási teljesítménye igen szerénynek mondható: 100 millió művelet perc másodperc, viszont cserébe nem igényel hűtést a *RISC* alapú processzor.

Négy új fejezet

Megjelent az *OpenGL* árnyékolónyelvéről szóló (röviden narancssárga) könyv második kiadása az *Addison-Wesley* jóvoltából. Újdonság benne a négy új fejezet, aminek köszönhetően 140 oldallal bővült a könyv.

Mágneses processzorok – a lehetséges jövő?

A *Notre Dame Egyetemen* (*Indiana, USA*) elkészítették egy – a mágnesességet kihasználó – új generációs processzor prototípusát. Jelenleg az elektromosságon alapuló processzorok már a lehetséges miniatürizálás határán járnak. Minthogy az új processzornál nincs szükség vezetékekre, az eszköz sűrűsége és számítási kapacitása jelentős mértékben növelhető, ennek ellenére közel sem lesz akkora fogyasztása, mint a jelenlegi processzoroknak, ami ideálissá teheti őket mobil felhasználásra.

Linuxos jégkrém-automata

A *MooBella* bemutatta legújabb jégkrém automatáját, mely 2.4-es kernelt és *RedHat Linuxot* futtat. A vásárlók egy 15 colos érintőképernyő segítségével 12 különböző ízből választhatnak, amit a gép 45 másodpercen belül elkészít. Ez az idő hosszúnak tűnhet, de vegyük figyelembe, hogy az eszköz csak a jégkrém elkészítésekor fagyaszt, addig szobahőmérsékleten tárolják az alapanyagokat. Végül de nem utolsósorban az eszköz az Internetre is kapcsolódhat, megkönnyítve a karbantartó személyzet munkáját.

Medve Zoltán
(e-medve@e-medve.hu)

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

Mi újság a rendszermag fejlesztése körül

A kernellel kapcsolatos projektek között feltűnt egy új, titkosított fájlrendszerrel kapcsolatos fejlesztés. Az ***eCryptFS*** alapkoncepcióját a 2005-ös ***Ottawa Linux Symposiumon*** mutatta be ***Phillip Hellewell*** és ***Michael Halcrow***, akik most a megfelelő kódot is beküldték felülvizsgálatra. Az ***eCryptFS***-t az ***Erez Zadok*** által írt ***FiST*** keretrendszer (***stackable filesystem framework***) segítségével más fájlrendszerek „tetejére" is rá lehet ültetni. A jelenlegi változat egyelőre csak befűzésenkénti visszafejtést (per-mount decription) támogat, hiszen jelenleg csak az alapfunkciók tesztelésére szánják a fejlesztők. Ugyanakkor a jövőben, ha véget ért az alap infrastruktúra tesztelése, a rendszer támogatja majd a fájlonkénti nyilvános kulcsú titkosítást, a jogosultságok finomhangolását, és számos egyéb speciális lehetőséget. Az ***eCryptFS*** legérdekesebb tulajdonsága az, hogy a titkosítással kapcsolatos metaadatokat magukban a fájlokban tárolja, így lehetővé teszi azok biztonságos átvitelét is.

Mark Gross benyújtott egy speciális karaktereszköz-meghajtót az ősszel megjelenő ***Intel NetStructure MPCBL0010*** egykártyás (single-board) számítógéphez. Az ***MPCBL0010***-át elsősorban olyan telekommunikációs eszközökbe szánják, amelyeknek gyakran kell műveleteiket valamilyen más hardvereszközhöz szinkronizálniuk. Mark meghajtójában az az új, hogy a magán a lapon található ***FPGA (Field-Programmable Gate Array)*** segítségével teszi lehetővé ezt a szinkronizációt. A rendszerfejlesztőknek emellett rendelkezésükre áll egy ***sysfs*** felület, amelyen keresztül egyszerűen szabályozhatják a szinkronizáció folyamatát. Mark elkészített egy ***ioctl*** felületet is, hogy a rendszert 2.4-es

kernel alatt is lehessen tesztelni. Ez várhatóan el fog tűnni azokból a végleges csomagokból, amelyeket a 2.6-os kernelfába beillesztenek.

A felfüggesztett mód (***software suspend***) című saga nagy kanyarokkal szélesen hömpölyögve folytatódik. ***Rafael J. Wysocki*** nemrég benyújtott egy olyan foltot, amely a teljes ***swsusp*** infrastruktúrát szétválasztja két, egymástól teljesen független rendszermagszintű alrendszerre. Közülük az egyik feladata kizárólag a futó rendszer pillanatfelvételének elkészítése, míg a másik ennek a képnek a csereterületre való kiírását illetve visszaolvasását vezérli. Ennek a megoldásnak a legfőbb előnye az, hogy lehetővé teszi a ***swsusp*** számos függvényének felhasználói térbe való áthelyezését. Emellett az új rendszer hatékonyabban használja a memóriát, elboldogul bizonyos korábban problémásnak bizonyuló méretkorlátokkal, illetve csökkenti az egyes globális változóktól való függést. ***Pavel Machek*** maga is átnézte ezeket a foltokat és nem talált bennük kivetnivalót. Igaz ugyan, hogy ennek a kódnak a fejlesztését számos szétválás, egyesülés sőt újraegyesülés illetve alrendszerekre bontás tarkította, nem szabad azonban elfelejtenünk, hogy még a jelenlegi, kissé felemás helyzet is jobb, mint amit kezdetben jósoltak. Eredetileg ugyanis sokan egyszerűen lehetetlennek tartották a szoftveres felfüggesztés megvalósítását, mondván a hardver egyes állapotait egyszerűen lehetetlen elmenteni. Nem csoda tehát, hogy ennek a szolgáltatásnak a megvalósítása jelenleg is meglehetősen problémás.

A ***git*** változatkövető rendszer fejlődési üteme töretlen. Aki még nem próbálta ki, itt az ideje, hogy megtegye.

Az ***Ubuntu*** terjesztés kernelfejlesztéssel kapcsolatos munkálatait immár ***git*** segítségével koordinálják, és nemrég az ***ethtool*** fejlesztői is átálltak erre a rendszerre. ***Randal L. Schwartz*** – nem mellesleg – nemrég arra is rámutatott, hogy a ***git***-nek a bináris tartalmak kezelése sem jelent problémát. Maga ***Schwartz*** webfejlesztéshez használja a ***git***-et, ahol bináris tartalomként elsősorban képek fordulnak elő, de ugyanígy kezelhetünk a rendszer segítségével tömörített tar állományokat, vagy bármi mást. Másik, élükön ***Jeff Garzikkal*** azon mesterkednek, hogy egy egyenrangú kiegészítést készítsenek a ***git***-hez. Ennek az értelme, hogy segítségével a felhasználók egyszerre nem csak egyetlen ***git*** tárat kérdezhetnének le, hanem egy egész ***git*** hálózatot.

Ami a fájlok átnevezését illeti, ***Linus Torvalds*** továbbra sem elégedett a rendszer ilyetén szolgáltatásaival. Szerinte a névváltozások követésének automatikusnak kell lennie, szemben a mostani megoldással, amikor a felhasználónak külön kell értesítenie a rendszert egy ilyen eseményről. Bár már nem ***Linux*** a ***git*** legfőbb fejlesztője, azért továbbra is komoly befolyása van az események alakulására,

továbbra is ő a *git* nagyszerű szolgáltatásainak legfőbb „reklámarca”, és a levelezési listákon is gyakran mutat elegáns és hatékony, ám nem kifejezetten magától értetődő megoldásokat az olvasóknak.
Adrian Bunk átvette a *Trivial Patch Monkey* karbantartását *Rusty Russeltől*. A *Rusty* által alapított projekt az olyan foltokat gyűjti össze és nyújtja be jóváhagyásra, amelyek annyira triviálisak, hogy egyszerűen nem lehetnek hibásak. Korábban sokszor megesett, hogy ezek a javítások éppen egyszerűségük és problémátlan voltuk miatt vesztek el a kavalkádban. A javítást végző fejlesztők gyakran nem igazán tudták, kihez kell az ilyen foltokat benyújtani, a jóváhagyásra feljogosított fejlesztők pedig nemegyszer megfeledkeztek róluk, mivel számos egyéb dolguk is volt. *Rusty* volt az, aki „fölszedegette a padlóról” az elhajigált foltokat, és addig nyújtotta be őket újra és újra, amíg valaki rájuk nem mondta az ámen. Ő volt az, aki félig automatizálta ezt a folyamatot, és a foltok benyújtását minden kernelkiadás előtt úgy időzítette, hogy az a leginkább megfeleljen a fejlesztőknek, beleértve magát *Linus Torvaldsot* is. Mivel *Rusty*-nak más dolga akadt, *Adrian* vette át a stafétabotot. A *Trivial Patch Monkey* jelentőségét és munkaigényét tekintve nem sokban tér el a *w.x.y.z* stabil kernelfa létrehozásától: mindkettő meglehetősen hálátlan feladat, hiszen csökkenti annak a lehetőségét, hogy a fenntartó sokkal érdekfeszítőbb, az új szolgáltatásokkal kapcsolatos fejlesztésekben vegyen részt. Ugyanakkor a közösség számára mindkettő kétségtelenül hasznos, bár azon kevesen szoktak elgondolkodni, hogy mit is köszönhetnek ezeknek az embereknek.

Zack Brown

Linux Journal 2006. 143. szám

Web 2.0 – E-volúció

Vajon mit takarhat a „Web 2.0” kifejezés? A kérdés aktuális, hiszen az utóbbi fél évben még olyan oldalakon is hosszú írásokat jelentettek meg róla, amelyek távolról sem foglalkoznak a webtervezés témakörével.

Valami fontos változásról van szó, mégis a szóösszetétel csalóka lehet. A web ugyanis nem egy program. Nincsenek tehát verziószámai. Ugyan mihez képest is lennének? A 2.0-ás verzió valami egészen mást jelöl, valami evolúciószerű változást, ami azonban részben a színfalak mögött játszódik.
A technológia zászlóshajóinak nevezett oldalakat végignézve elsőre nehezen találjuk meg bennük a közös elemeket. Az új irányvonalakat úgy érhetjük csak tetten, ha szépen sorjában példákkal alátámasztva megpróbáljuk megvilágítani a változások mozgatórugóit.

Technológia és szemlélet

A váltás nem csak technikai jellegű. A fontosabb technológiák már hosszabb ideje velünk vannak, mégis csak most kezdenek összeállni kerek egésszé. Vitathatatlanul nagy hangsúlyt kap a hibamentes, szabványos építőelemek használata, ezért aztán egyre többször találkozunk a *W3C*, vagyis a *World Wide Web Consortium* nevével, melynek – többek között –

az *XHTML* és a *CSS* szabvány kifejlesztését is köszönhetjük. A szabványkövetés különösen fontossá válik, hiszen a legnagyobb újítás és egyben a változások legfőbb mozgatórugója az egyes szolgáltatások ***együttműködésében*** rejlik.

Tartalom

A következő generációs oldalak a tartalom terén merőben más szemléletet követnek. A koncepció hasonló ahhoz, ahogy az objektum-orientáltan programozók gondolkodnak. Összecsomagoljuk az információkat és az eszközöket amivel kezelhetjük őket. Számos olyan oldal jön létre, mely csak bizonyos adatok feldolgozására fókuszál. Ezeket az oldalakat joggal nevezhetjük ***webalkalmazásoknak***, hiszen nagyon hasonló funkcionalitást nyújtanak mint bármelyik program a számítógépünkön, ráadásul még a kezelésük is hasonló.
Tekintsük példának a *Writely.com*-ot, amely egy zseniálisan összerakott online szövegszerkesztő. Rövid regisztráció után pillanatok alatt hozhatunk létre új dokumentumokat vagy szerkeszthetjük a már meglévőket. Publikálhatjuk az írásainkat blogunkba, ha az támogatja az elfogadott kommunikációs csatolófelületeket (*Bogger* vagy

Atom api). Bármily hihetetlenül hangzik, gond nélkül importálhatjuk meglévő *Word* vagy *OppenOffice* dokumentumainkat, és ha valamiért szükségünk lenne az írásainkra, exportálhatjuk is őket hasonlóképpen. A közös szerkesztés lehetősége már csak hab a tortán. Aki szerkesztett már a munkatársával közösen valamilyen dokumentumot, az tudhatja, mekkora „élvezet" nézni, ahogy a másik csigalassúsággal gépeli be a szöveget!

Együttműködés

Az egyes szolgáltatások messzemenőkig együttműködnek. Weboldalunk oldalsávjában elhelyezhetjük a fotóinkat, melyeket a *Flickr.com* segítségével kezelünk. Ez az innovatív fotókezelő

alkalmazás segít rendszerezni és megosztani a képeinket, mégpedig ingyen. Nyilvános programozói felülete segítségével diavetítéseinket könnyedén illeszthetjük be meglévő oldalainkba.

A fotók alatt pedig bőven jut hely kedvenc oldalaink listájának is, amit a D*el.icio.us* segítségével rendszerezhetünk. Ezzel a szolgáltatással kategorizálhatjuk és felcímkézhetjük azokat az oldalakat, amelyekre mindenképp emlékezni szeretnénk.

Az internet mindenkié

Az internetet sokáig csak olvastuk. Később sokat megtanultuk a háttérben lévő technológiák csínját-bínját és így tartalmakat is helyezhettünk el rajta. Mára azonban már mindenki tevékeny alakítója is egyben. A híres *Wikipedia* immár több mint másfél millió szócikket tartalmaz. Köszönhetően a tizenháromezer szerzőnek 2000 óta a legrészletesebb és legnagyobb enciklopédiává vált.
A *Wikipedia* mellett a számtalan blog is bizonyítja, hogy a szakmai tudás már nem feltétel a kommunikációhoz. Kiemelnék egy nagyon ötletes magyar kezdeményezést a *Blogter.hu*-t, amely magát „Első magyar web 2.0 portál" néven hirdeti. Az eddigiek alapján jogosan.

Láthatjuk tehát, hogy a kommunikáció gyökeresen átalakul. Mindenki lehet szerző és olvasó egy személyben. Az egész internet valami megfoghatatlan kollektív tudásközponttá válik.
A web tehát átalakul. Még korántsem érte el végső formáját, mivel azt a technológiák és a piaci igények közösen fogják lassan-lassan véglegesíteni. Hajrá, vegyünk részt ebben az izgalmas mozgalomban, mely a *„web 2.0"* kulcsszóval gyűjti zászlója köré az embereket! Ne feledjük, a két legfontosabb kulcsszó az interakció és az együttműködés.

Juhász Attila
(rabszolga@goraffe.hu)

Az Információ Technológiai Kar hallgatója a Pázmány Péter Katolikus Egyetemen. Érdeklődik a bioinformatika és a neurális hálózatok iránt. A fotózás és a tánc mellett öt éve foglalkozik webgrafikákkal. A linux terjesztések közül a Gentoo és az Ubuntu áll legközelebb a szívéhez. Fotós oldala a http://people.goraffe.com/attila címen található.

KAPCSOLÓDÓ CÍMEK

Wikipedia szócikk:
➲ en.wikipedia.org/wiki/Web2.0

A Figyelő Net cikke:
➲ www.fn.hu/index.php?id=8&cid=116086

Doransky blog bejegyzése:
➲ blog.doransky.hu/?q=node/429

Writelly:
➲ www.writelly.com

Del.icio.us:
➲ del.icio.us

Flickr:
➲ flickr.com

Wikipedia:
➲ en.wikipedia.org

Blogter:
➲ blogter.hu


Linuxvilág
www.linuxvilag.hu

Mi újság a rendszermag fejlesztése körül

A kernellel kapcsolatos projektek között feltűnt egy új, titkosított fájlrendszerrel kapcsolatos fejlesztés. Az ***eCryptFS*** alapkoncepcióját a 2005-ös ***Ottawa Linux Symposiumon*** mutatta be ***Phillip Hellewell*** és ***Michael Halcrow***, akik most a megfelelő kódot is beküldték felülvizsgálatra. Az ***eCryptFS***-t az ***Erez Zadok*** által írt ***FiST*** keretrendszer (***stackable filesystem framework***) segítségével más fájlrendszerek „tetejére" is rá lehet ültetni. A jelenlegi változat egyelőre csak befűzésenkénti visszafejtést (per-mount decription) támogat, hiszen jelenleg csak az alapfunkciók tesztelésére szánják a fejlesztők. Ugyanakkor a jövőben, ha véget ért az alap infrastruktúra tesztelése, a rendszer támogatja majd a fájlonkénti nyilvános kulcsú titkosítást, a jogosultságok finomhangolását, és számos egyéb speciális lehetőséget. Az ***eCryptFS*** legérdekesebb tulajdonsága az, hogy a titkosítással kapcsolatos metaadatokat magukban a fájlokban tárolja, így lehetővé teszi azok biztonságos átvitelét is.

Mark Gross benyújtott egy speciális karaktereszköz-meghajtót az ősszel megjelenő ***Intel NetStructure MPCBL0010*** egykártyás (single-board) számítógéphez. Az ***MPCBL0010***-át elsősorban olyan telekommunikációs eszközökbe szánják, amelyeknek gyakran kell műveleteiket valamilyen más hardvereszközhöz szinkronizálniuk. Mark meghajtójában az az új, hogy a magán a lapon található ***FPGA (Field-Programmable Gate Array)*** segítségével teszi lehetővé ezt a szinkronizációt. A rendszerfejlesztőknek emellett rendelkezésükre áll egy ***sysfs*** felület, amelyen keresztül egyszerűen szabályozhatják a szinkronizáció folyamatát. Mark elkészített egy ***ioctl*** felületet is, hogy a rendszert 2.4-es

kernel alatt is lehessen tesztelni. Ez várhatóan el fog tűnni azokból a végleges csomagokból, amelyeket a 2.6-os kernelfába beillesztenek.

A felfüggesztett mód (***software suspend***) című saga nagy kanyarokkal szélesen hömpölyögve folytatódik. ***Rafael J. Wysocki*** nemrég benyújtott egy olyan foltot, amely a teljes ***swsusp*** infrastruktúrát szétválasztja két, egymástól teljesen független rendszermagszintű alrendszerre. Közülük az egyik feladata kizárólag a futó rendszer pillanatfelvételének elkészítése, míg a másik ennek a képnek a csereterületre való kiírását illetve visszaolvasását vezérli. Ennek a megoldásnak a legfőbb előnye az, hogy lehetővé teszi a ***swsusp*** számos függvényének felhasználói térbe való áthelyezését. Emellett az új rendszer hatékonyabban használja a memóriát, elboldogul bizonyos korábban problémásnak bizonyuló méretkorlátokkal, illetve csökkenti az egyes globális változóktól való függést. ***Pavel Machek*** maga is átnézte ezeket a foltokat és nem talált bennük kivetnivalót. Igaz ugyan, hogy ennek a kódnak a fejlesztését számos szétválás, egyesülés sőt újraegyesülés illetve alrendszerekre bontás tarkította, nem szabad azonban elfelejtenünk, hogy még a jelenlegi, kissé felemás helyzet is jobb, mint amit kezdetben jósoltak. Eredetileg ugyanis sokan egyszerűen lehetetlennek tartották a szoftveres felfüggesztés megvalósítását, mondván a hardver egyes állapotait egyszerűen lehetetlen elmenteni. Nem csoda tehát, hogy ennek a szolgáltatásnak a megvalósítása jelenleg is meglehetősen problémás.

A ***git*** változatkövető rendszer fejlődési üteme töretlen. Aki még nem próbálta ki, itt az ideje, hogy megtegye.

Az ***Ubuntu*** terjesztés kernelfejlesztéssel kapcsolatos munkálatait immár ***git*** segítségével koordinálják, és nemrég az ***ethtool*** fejlesztői is átálltak erre a rendszerre. ***Randal L. Schwartz*** – nem mellesleg – nemrég arra is rámutatott, hogy a ***git***-nek a bináris tartalmak kezelése sem jelent problémát. Maga ***Schwartz*** webfejlesztéshez használja a ***git***-et, ahol bináris tartalomként elsősorban képek fordulnak elő, de ugyanígy kezelhetünk a rendszer segítségével tömörített tar állományokat, vagy bármi mást. Másik, élükön ***Jeff Garzikkal*** azon mesterkednek, hogy egy egyenrangú kiegészítést készítsenek a ***git***-hez. Ennek az értelme, hogy segítségével a felhasználók egyszerre nem csak egyetlen ***git*** tárat kérdezhetnének le, hanem egy egész ***git*** hálózatot.

Ami a fájlok átnevezését illeti, ***Linus Torvalds*** továbbra sem elégedett a rendszer ilyetén szolgáltatásaival. Szerinte a névváltozások követésének automatikusnak kell lennie, szemben a mostani megoldással, amikor a felhasználónak külön kell értesítenie a rendszert egy ilyen eseményről. Bár már nem ***Linux*** a ***git*** legfőbb fejlesztője, azért továbbra is komoly befolyása van az események alakulására,

továbbra is ő a *git* nagyszerű szolgáltatásainak legfőbb „reklámarca”, és a levelezési listákon is gyakran mutat elegáns és hatékony, ám nem kifejezetten magától értetődő megoldásokat az olvasóknak.
Adrian Bunk átvette a *Trivial Patch Monkey* karbantartását *Rusty Russeltől*. A *Rusty* által alapított projekt az olyan foltokat gyűjti össze és nyújtja be jóváhagyásra, amelyek annyira triviálisak, hogy egyszerűen nem lehetnek hibásak. Korábban sokszor megesett, hogy ezek a javítások éppen egyszerűségük és problémátlan voltuk miatt vesztek el a kavalkádban. A javítást végző fejlesztők gyakran nem igazán tudták, kihez kell az ilyen foltokat benyújtani, a jóváhagyásra feljogosított fejlesztők pedig nemegyszer megfeledkeztek róluk, mivel számos egyéb dolguk is volt. *Rusty* volt az, aki „fölszedegette a padlóról” az elhajigált foltokat, és addig nyújtotta be őket újra és újra, amíg valaki rájuk nem mondta az ámen. Ő volt az, aki félig automatizálta ezt a folyamatot, és a foltok benyújtását minden kernelkiadás előtt úgy időzítette, hogy az a leginkább megfeleljen a fejlesztőknek, beleértve magát *Linus Torvaldsot* is. Mivel *Rusty*-nak más dolga akadt, *Adrian* vette át a stafétabotot. A *Trivial Patch Monkey* jelentőségét és munkaigényét tekintve nem sokban tér el a *w.x.y.z* stabil kernelfa létrehozásától: mindkettő meglehetősen hálátlan feladat, hiszen csökkenti annak a lehetőségét, hogy a fenntartó sokkal érdekfeszítőbb, az új szolgáltatásokkal kapcsolatos fejlesztésekben vegyen részt. Ugyanakkor a közösség számára mindkettő kétségtelenül hasznos, bár azon kevesen szoktak elgondolkodni, hogy mit is köszönhetnek ezeknek az embereknek.

Zack Brown

Linux Journal 2006. 143. szám

Web 2.0 – E-volúció

Vajon mit takarhat a „Web 2.0” kifejezés? A kérdés aktuális, hiszen az utóbbi fél évben még olyan oldalakon is hosszú írásokat jelentettek meg róla, amelyek távolról sem foglalkoznak a webtervezés témakörével.

Valami fontos változásról van szó, mégis a szóösszetétel csalóka lehet. A web ugyanis nem egy program. Nincsenek tehát verziószámai. Ugyan mihez képest is lennének? A 2.0-ás verzió valami egészen mást jelöl, valami evolúciószerű változást, ami azonban részben a színfalak mögött játszódik.
A technológia zászlóshajóinak nevezett oldalakat végignézve elsőre nehezen találjuk meg bennük a közös elemeket. Az új irányvonalakat úgy érhetjük csak tetten, ha szépen sorjában példákkal alátámasztva megpróbáljuk megvilágítani a változások mozgatórugóit.

Technológia és szemlélet

A váltás nem csak technikai jellegű. A fontosabb technológiák már hosszabb ideje velünk vannak, mégis csak most kezdenek összeállni kerek egésszé. Vitathatatlanul nagy hangsúlyt kap a hibamentes, szabványos építőelemek használata, ezért aztán egyre többször találkozunk a *W3C*, vagyis a *World Wide Web Consortium* nevével, melynek – többek között –

az *XHTML* és a *CSS* szabvány kifejlesztését is köszönhetjük. A szabványkövetés különösen fontossá válik, hiszen a legnagyobb újítás és egyben a változások legfőbb mozgatórugója az egyes szolgáltatások ***együttműködésében*** rejlik.

Tartalom

A következő generációs oldalak a tartalom terén merőben más szemléletet követnek. A koncepció hasonló ahhoz, ahogy az objektum-orientáltan programozók gondolkodnak. Összecsomagoljuk az információkat és az eszközöket amivel kezelhetjük őket. Számos olyan oldal jön létre, mely csak bizonyos adatok feldolgozására fókuszál. Ezeket az oldalakat joggal nevezhetjük ***webalkalmazásoknak***, hiszen nagyon hasonló funkcionalitást nyújtanak mint bármelyik program a számítógépünkön, ráadásul még a kezelésük is hasonló.
Tekintsük példának a *Writely.com*-ot, amely egy zseniálisan összerakott online szövegszerkesztő. Rövid regisztráció után pillanatok alatt hozhatunk létre új dokumentumokat vagy szerkeszthetjük a már meglévőket. Publikálhatjuk az írásainkat blogunkba, ha az támogatja az elfogadott kommunikációs csatolófelületeket (*Bogger* vagy

Atom api). Bármily hihetetlenül hangzik, gond nélkül importálhatjuk meglévő *Word* vagy *OppenOffice* dokumentumainkat, és ha valamiért szükségünk lenne az írásainkra, exportálhatjuk is őket hasonlóképpen. A közös szerkesztés lehetősége már csak hab a tortán. Aki szerkesztett már a munkatársával közösen valamilyen dokumentumot, az tudhatja, mekkora „élvezet" nézni, ahogy a másik csigalassúsággal gépeli be a szöveget!

Együttműködés

Az egyes szolgáltatások messzemenőkig együttműködnek. Weboldalunk oldalsávjában elhelyezhetjük a fotóinkat, melyeket a *Flickr.com* segítségével kezelünk. Ez az innovatív fotókezelő

alkalmazás segít rendszerezni és megosztani a képeinket, mégpedig ingyen. Nyilvános programozói felülete segítségével diavetítéseinket könnyedén illeszthetjük be meglévő oldalainkba.

A fotók alatt pedig bőven jut hely kedvenc oldalaink listájának is, amit a D*el.icio.us* segítségével rendszerezhetünk. Ezzel a szolgáltatással kategorizálhatjuk és felcímkézhetjük azokat az oldalakat, amelyekre mindenképp emlékezni szeretnénk.

Az internet mindenkié

Az internetet sokáig csak olvastuk. Később sokat megtanultuk a háttérben lévő technológiák csínját-bínját és így tartalmakat is helyezhettünk el rajta. Mára azonban már mindenki tevékeny alakítója is egyben. A híres *Wikipedia* immár több mint másfél millió szócikket tartalmaz. Köszönhetően a tizenháromezer szerzőnek 2000 óta a legrészletesebb és legnagyobb enciklopédiává vált.
A *Wikipedia* mellett a számtalan blog is bizonyítja, hogy a szakmai tudás már nem feltétel a kommunikációhoz. Kiemelnék egy nagyon ötletes magyar kezdeményezést a *Blogter.hu*-t, amely magát „Első magyar web 2.0 portál" néven hirdeti. Az eddigiek alapján jogosan.

Láthatjuk tehát, hogy a kommunikáció gyökeresen átalakul. Mindenki lehet szerző és olvasó egy személyben. Az egész internet valami megfoghatatlan kollektív tudásközponttá válik.
A web tehát átalakul. Még korántsem érte el végső formáját, mivel azt a technológiák és a piaci igények közösen fogják lassan-lassan véglegesíteni. Hajrá, vegyünk részt ebben az izgalmas mozgalomban, mely a *„web 2.0"* kulcsszóval gyűjti zászlója köré az embereket! Ne feledjük, a két legfontosabb kulcsszó az interakció és az együttműködés.

Juhász Attila
(rabszolga@goraffe.hu)

Az Információ Technológiai Kar hallgatója a Pázmány Péter Katolikus Egyetemen. Érdeklődik a bioinformatika és a neurális hálózatok iránt. A fotózás és a tánc mellett öt éve foglalkozik webgrafikákkal. A linux terjesztések közül a Gentoo és az Ubuntu áll legközelebb a szívéhez. Fotós oldala a http://people.goraffe.com/attila címen található.

KAPCSOLÓDÓ CÍMEK

Wikipedia szócikk:
➲ en.wikipedia.org/wiki/Web2.0

A Figyelő Net cikke:
➲ www.fn.hu/index.php?id=8&cid=116086

Doransky blog bejegyzése:
➲ blog.doransky.hu/?q=node/429

Writelly:
➲ www.writelly.com

Del.icio.us:
➲ del.icio.us

Flickr:
➲ flickr.com

Wikipedia:
➲ en.wikipedia.org

Blogter:
➲ blogter.hu


Linuxvilág
www.linuxvilag.hu

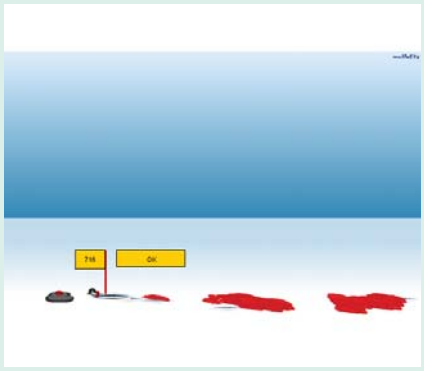
Pingvin felszállóágban

Vannak, akik szerint leginkább a gyilkolási ösztön az, amely igazán közel hozhatja az embert a természethez. Bizarr megközelítés ez olyanok szájából, kik arra okítanak: mely részén, milyen erővel és ívben kell baseballütővel eltalálni egy pingvint, hogy annak véres testrészei a lehető legnagyobb távot repüljék be…

Mielőtt állatvédők követelnék rajtam e szájakhoz tartozó testek elérhetőségeit és bilincseket csörrentenének végtelen haraggal, hamar kinyilatkozom, hogy a szájak, testek, baseballütők és véres pingvinfejek helyszíne csupán virtuális játszótér. S bár bolondokházához elég egy üres szoba és megfelelő személyek, az on-line játékokat főként gumírozott padlójú-falú szobán kívül játsszák. Sokan a munkaidő szabad perceit, illetve a szabadidő munkás óráit töltik „Szép napunk van, öljünk egyet"– egérszorítással és madárlapítással.

Ha valakit igazán elmélyült szorgoskodással találunk a monitor előtt bonyolult matematikai számításokat és egyszerű négyjegyű számokat mormolva maga elé, időnként káromkodva (rohadt madár) vagy ujjongva (rohadt madár), s alig egy-két másodpercre választva csak el ujjait az egértől – az nagy valószínűséggel nem a repülőirányítás legszorgosabb munkatársa, inkább a jetisport lelkes-törekvő űzője. A játék 4 szereplője: egy jeti, egy baseballütő, egy szikla és egy pingvin. Utóbbi szárnyas a szikla tetején várakozik, hogy egy gombnyomásra a jeti és az ütő elé vethesse magát azon okból, hogy a találkozás pillanatában egészben vagy darabokra hullva minél távolabbi ponton fúródhasson a talajba.

A pingvin nagyon is segítőkész a játékos fáradt agyának felpezsdítésében; a gyengébbek kedvéért végtelen szárnyalást mutat be és nagyobb ütéseket áll ki, a profiknak nehezítésül összevissza pattog, bátran viseli több száz kilós tüskés durung csapásait, láncfűrész trancsírozását, illetve céltáblába is fúródik (egy hógolyóval ülepében).



Kutatók még vitáznak az erőszakos játékok és az agresszió általános mértékének viszonyán, így az elkötelezett játékosok széttárhatják karjaikat a borzadó tekintetekre: Ugyan, csak levezetjük a fölös feszült attitűdöket!

Egyesek szerint mindennek terjesztői a Linux-szidalmazók, kik szerencsétlen madáron verik el a port (és a baseballütőt, láncfűrészt vagy éppen a *World Trade Centert*), ugyanakkor a játék 2004-es debütálásakor érezhetően csökkent a Microsoft termelékenysége... Tehát, hajrá!

Halusz Léna

A Novell személyazonosság alapú biztonság-filozófiája

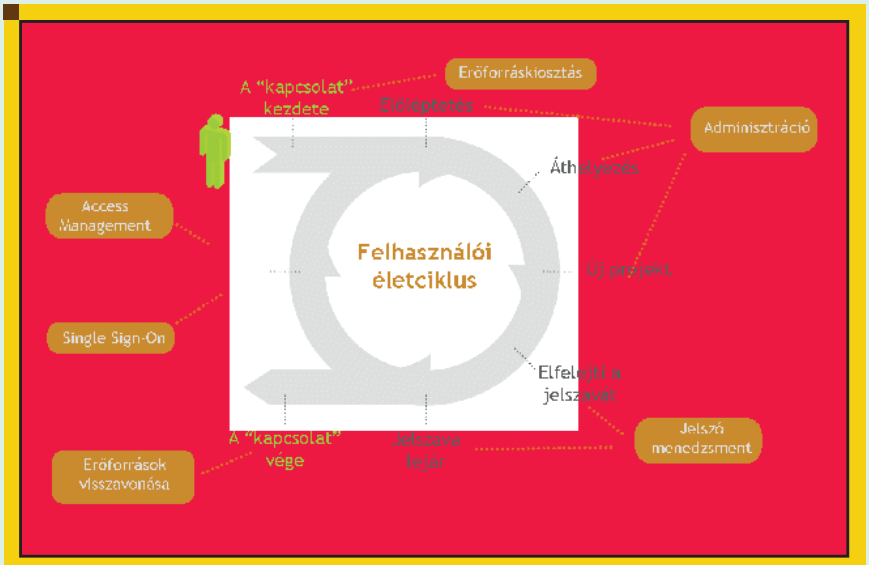
A Novell Identity Manager 3 bemutatása

Napjaink egyik legnagyobb globális kihívását az informatikai rendszerek biztonsága jelenti. A biztonsági megoldások iránti igény már nem csak a hálózat határainak biztonságára és fenyegetések észlelésére vonatkozik. Bármilyen tűzfallal, vírusvédelmi megoldással rendelkezhet egy cég, ha nem tudja kellő biztonsággal kiszűrni, hogy az adott rendszerbe érkező adatlehívások arra jogosult személytől érkeznek-e.

A *Novell*, melynek személyazonosság-kezelési megoldása jelenleg 33 %-os részesedéssel piacvezető Magyarországon, méltán büszke arra, hogy a rendszer alapjául szolgáló *Novell* címtárban (*eDirectory*) 1993-as megjelenése óta nem tapasztaltak illegális behatolást. A *Novell* tovább kívánja erősíteni vezető pozícióját ezen a területen, és ennek érdekében piacra dobta a *Novell Identity Manager* legújabb, 3-as verzióját. Az új megoldással a szervezeti hierarchiának megfelelően alakíthatjuk ki a személyazonosságok és hozzáférések biztonságos kezelését. Segítségével a munkába állás napján kioszthatóak a fontos üzleti erőforrások, igény szerint akár egy jelszóval elérhetőek a különböző rendszerek, az egyes jogosultságok kiosztása és visszavonása azonnal megtörténhet.

A biztonság megteremtése

Amennyiben informatikai rendszereinket illetően nem rendelkezünk konzisztens felhasználói adatokkal, a szolgáltatások szintjének emelése, a biztonsági gyenge pontok megszüntetése, vagy az *IT* adminisztrációs költségek lefaragása érdekében tett erőfeszítéseink eleve kudarcra vannak ítélve. Az új felhasználók

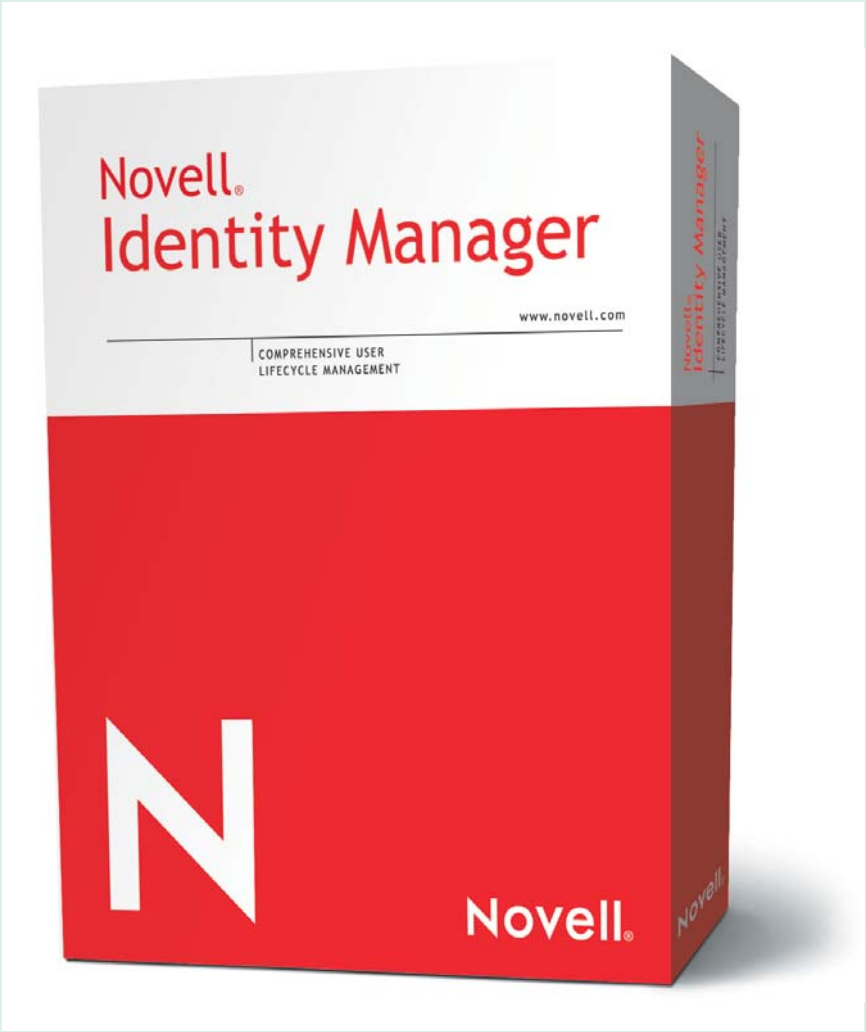


1. ábra Egy tipikus felhasználói életciklus a Novell Identity Manager rendszerében

Piaci trendek a biztonságról és személyazonosságkezelésről

Hazánkban 2003-tól 2009-ig átlagosan évi 17,1 százalékos növekedést jósol az *IDC* a biztonsági szoftverek területén. *Magyarországon* ma még a biztonsági szoftverek piacának mindössze 20 százalékát teszik ki a személyazonosság kezelési megoldások, de a piac növekedési üteme több mint kétszerese a biztonsági szoftverekének, eléri a 35 százalékot éves szinten. Ez a tendencia jól mutatja a hazai személyazonosság alapú biztonsági megoldások növekvő fontosságát a céges biztonságpolitika kialakításában. A hazai piac vezetője 33 százalékos piaci részesedéssel a *Novell Magyarország*.

tétlenül fognak ülni mindaddig, amíg nem kapnak hozzáférést az üzleti rendszerekhez, miközben kilépett alkalmazottaink még hetekig, hónapokig használhatják azokat. Sokszor okoz problémát az is, hogy a különböző rendszereket használó alkalmazottaink a közös projektek részleteit nem tudják megfelelően megosztani egymással, így nem tudnak hatékonyan együtt dolgozni. A fentiekből könnyen belátható, hogy még az üzleti szabályozások figyelembe vételével kialakított, rendszerezett folyamatok szerint működő vállalat sem tud sikeres lenni ellenőrizhető, pontos és időben történő személyazonosság-kezelés nélkül. A biztonság megteremtése tehát a vállalaton belül a személyazonosságok és hozzáférések hatékony kezelésével valósulhat meg.



A díjnyertes Novell Identity Manager

A *Novell* metacímtár technológiája a *Novell Identity Manager*, a *Novell* kiforrott, majd 10 éves tapasztalattal rendelkező, teljes szolgáltatáskörű címtár-technológiájának a kiterjesztése. A Novell technológiája számos szakmai elismeréssel gazdagodott ez idő alatt, például az *InfoWorld „a legjobb személyazonosság-kezelő"* alkalmazásnak választotta a 2006-os évre, de korábban elnyerte többek között a *Network Magazine* és a *Network Computing* közös *„Category Breaker Award"* díját, a *Media LLCs Network Computing „Well-Connected Award"* díját és a *CMP Media „Technical Innovation Award"* díját is.

A *Novell* személyazonosság-kezelő megoldásával olyan szinkronizációs mechanizmus valósítható meg, amely a különféle – akár különböző korból származó – felhasználói és alkalmazás-adattárak adatait üzleti szabályoknak és irányelveknek megfelelően integrálja és szinkronizálja. A hálózat összes címtárát, felhasználói nyilvántartását összekapcsolhatjuk egy központi címtárral, biztosítva, hogy az egyforma adatok valóban egységesek legyenek az összes nyilvántartásban és adattárban – még abban az esetben is, ha azok egyedi típusúak. Az egységesítésen túl automatizálható és biztonságossá tehető a személyazonosságok és hozzáférések kezelése a folyamatosan változó felhasználói közösség teljes életciklusán keresztül. A rendszer így garantálja, hogy csak a megfelelő jogosultsággal rendelkező munkatárs férhet hozzá az adatokhoz, ő viszont bárhonnan bármikor.

A *NIM self-service* része, a saját profil információk karbantartásához és az elfelejtett jelszavak visszaállításához nyújt hatékony eszközt, ezzel értékes erőforrásokat szabadítva fel az üzemeltetésen és a *HelpDesken*.

A Novell Identity Manager 3 újdonságai

A *Novell Identity Manager 3*-as verziószámú változata az eddigi funkciók megtartása mellett újabb területen is kínál hatékony megoldást a folyamatok egyszerűsítésére és felgyorsítására. Az *Identity Managerhez* fejlesztett munkafolyamat-támogató rendszer és felhasználói portál együttesen megszüntetik a kézi adminisztrációból származó pontatlanságokat és tévesztéseket, és egy jól kiszámolható, naplózott és biztonságos rendszert alkotnak. A *Novell Identity Manager* saját *workflow* rendszerében egyszerűen és gyorsan összeállíthatóak azok a munkafolyamatok, amelyek a vállalat meglévő jóváhagyásos engedélyezési eljárásait tükrözik le. Delegálhatók az adminisztrációs feladatok, de egyes feladatokat az alkalmazottak is beállíthatnak saját maguknak, karban tarthatják saját adataikat, visszaállíthatják elfelejtett jelszavaikat – csökkentve ezzel a *help desk* terhelését. Továbbfejlesztett webes telefonkönyv és hierarchikus fastruktúrában megjelenített összefüggések is segítik az alkalmazottak munkáját. Az *offline* üzemmódban működő *Eclipse* alapú *Designerrel* a projekteken párhuzamosan is elvégezhetőek a feladatok, majd a projekt dokumentációja automatikusan elkészíthető.


Linuxvilág
www.linuxvilag.hu

Az e-volúció zászlóshajói

Azt gondolom, 2006 az internet éve. Habár a szükséges technológiák jó része már másfél évtizede velünk van, mégis most jutottunk el oda, hogy minden kiforrottan összeálljon. Gyökeres változások tanúi vagyunk. Az internet a szemünk láttára alakul át platformmá. Az interakció megnő. A weboldalak átalakulnak.

Lassan eltűnik a felesleges oldal újratöltődés, a mehet gombok nyomogatása is a múlté. Minden oldal úgy fog fog működni, mint akármelyik asztali alkalmazásunk. Interaktívan és gyorsan. A web programozása is szemlátomást változik. Új, friss megoldásokban nincs hiány! Rövid összefoglaló következik három új szereplőről, amelyek népszerű szkriptnyelvekre, a ***Pythonra*** és ***Rubyra*** építve gyökeresen átalakítják a webet.

Ruby on Rails

Alig másfél éve indult hódító útjára az új stílusú webprogramozás zászlóshajója a ***Ruby on Rails*** (röviden ***Rails***). Egy chichagói fiatal programozó, ***David Heinemeier Hansson alkotta m***eg az alapkoncepciót ***basecamp*** nevű fejlesztése közben. A fejlesztőeszközt rövidesen nyílt forrásúvá tette. Köszönhetően az egyre növekvő közösségnek és a fejlesztésbe bekapcsolódó számos programozónak, a ***Rails*** 2005. december 13-án, alig másfél év alatt elérte az 1.0-ás verziót. ***David*** a rugalmas és rendkívül gyors kódolást lehetővé tevő ***Ruby*** nyelvet használta fel a fejlesztéshez. A főként ***Japánban*** és immár ***Amerikában*** ismert ***Ruby*** nagyon hasonlít a ***Pythonhoz***. Fejlett szintaxisával és teljesen objektum orientált megközelítésével villámgyors kódolást tesz lehetővé. Általánosságban elmondhatjuk, hogy egy ***Ruby*** vagy ***Python*** program hossza ötöde vagy tizede egy hasonló funkciójú ***Java*** programnak.

A Rails legfőbb jellemzői

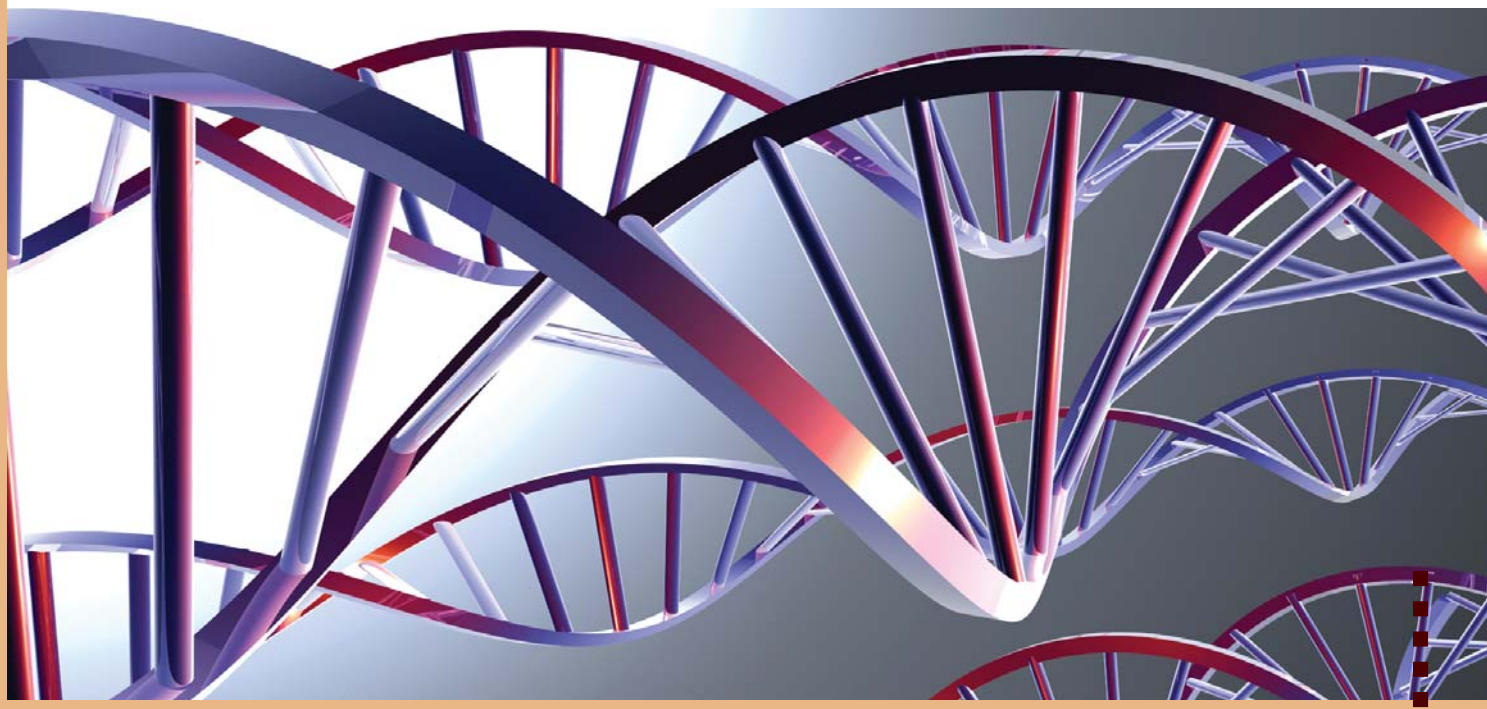
Minden igényt kielégítő fejlesztőeszköz, amely integráltan tartalmazz mindent ami adatbázis alapú weboldalak létrehozásához szükséges.
A villámgyors munkát segíti az is hogy az alkalmazásainkat az asztali alkalmazások fejlesztésénél megszokott ***Model-View-Controller*** séma felhasználásával fejleszthetjük. Három önálló részre bontjuk szét az alkalmazásainkat, ezáltal leegyszerűsítve a programozást, a hibakeresést és növelve az átláthatóságot. A ***model*** hivatott kezelni az adatainkat, a ***view*** felel a megjelenítésért és a ***controller*** pedig megmondja, hogy a felhasználó mely interakciójára mit is csináljon a program. Az egész fejlesztőeszköz ***Ruby*** alapú, nekünk csak egy adatbázis-kezelőre van szükségünk hogy nekikezdhessünk a fejlesztésnek.
A ***Rails*** kiemelt figyelmet szentel a ***DRY*** (***Don't Repeat Yourself***), vagyis a ne ismételd magad elv betartásának. Ezért nincsenek hosszú beállításfájlok, ismétlődő programozói feladatok. Például egy új programnál elég beállítani az adatbázis paramétereit és írni tíz sor kódot, hogy működő, látható eredményt kapjunk.
A Rails népszerűségéhez nagyban hozzájárul, hogy a remek leírások mellett sok videó is elérhető, melyeken végignézhetjük miként is dolgozik egy profi. Az alábbi kép egy olyan videó záró képkockája, mely azt mutatja be, miként illeszthetjük a ***Flickr.com*** által kezelt fotóinkat az oldalunkba mindössze öt perc alatt.

A *Rails* kiemelten támogatja az *AJAX*-ot (*Asynchronous JavaScript and XML*), vagyis a legfontosabb Javascript kódkönyvtárak illesztve vannak a keretrendszerhez, ami által úgy használhatjuk őket, hogy végig csak *Ruby* kódot írunk. Így könnyen készíthetünk az oldal újratöltődése nélkül frissülő menüket, animált szövegmezőket és még sok mást. Akit a téma bővebben érdekel, nézzen szét a ➲ *script.aculo.us* oldalon. Mindent amit ott láthatunk szervesen integráltak a keretrendszerbe. (➲ *wiki.script.aculo.us/scriptaculous/show/Demos*)
Azt gondolhatnánk hogy a gyors és interaktív fejlesztés nehezen eredményezhet hibamentes kódot. A *Rails* ebben is a segítségünkre van, hiszen alkalmazásaink minden porcikáját letesztelhetjük a segítségével, sőt lehetőséget ad, hogy a teszt alapú fejlesztés gyakorlatát kövessük. Ez által professzionális munkára is ideális választás lehet.

TurboGears

A mindössze fél éves *TurboGears* megpróbálja megvalósítani a *Rails* koncepcióját a *Python* programozók számára. A *Kevin Dangoor* által indított fejlesztés összeboronált számos meglévő *Python* programot, így alkotva meg egy könnyen kezelhető keretrendszert. Honlapja a ➲ *turbogears.org* címen található. A *Railshoz* hasonlóan itt is a *Model-View-Controller* séma alapján oszthatjuk részekre a programjainkat. Az ábrán jól látható az egyes rétegek egymásra épülése. Az *SQLObject* segít könnyedén kezelni az adatbázisainkat, tehát ez felel meg a modellnek, a *CherryPy* a controller, mely a különböző kérésekre eldönti milyen választ kell visszaadni a böngészőnek. A *Kid* egy könnyen tanulható sablon rendszer, mely a *Mochikit* nevű *Javascript* kódkönyvtár segítségével hihetetlenül interaktív felhasználói felületeket képes generálni.
Természetesen a *TurboGears* is a kezünk alá dolgozik, ha programjaink minden sorát tesztekkel szeretnénk ellenőrizni.

Akárcsak a *Rails* körül, a *TurboGeras* körül is nagy közösség szerveződött. Fiatalabb rendszer lévén még van néhány gyermekbetegsége. Voltak például problémák az egyes kódlapok használatával, de a hamarosan megjelenő 1.0-ás verzió már a tervek szerint teljesen használható lesz éles rendszerekben is. A megjelenő stabil kiadás a tervek szerint tartalmaz majd egy on-line *Python* héjat (shell) is, ami azért valljuk be hasznosnak tűnik távoli fejlesztéshez vagy hibakereséshez. Összességében elmondhatjuk, hogy a *Python* közösség egy olyan



A három keretrendszer összehasonlítása

Rails

Előny: A ***Ruby*** programozási nyelv ereje, nagy közösség (magyar közösség is van már), jó dokumentáció, könnyű integráció más rendszerekkel és technológiákkal, kipróbált, skálázható, webalkalmazásokhoz ideális.
Hátrány: A többnyelvű alkalmazások készítése még nem teljesen átgondolt, bár jól használható.
Értékelés: Mindenkinek nagyon ajánlom!

TurboGears

Előny: ***Pythonban*** íródott, gyorsan fejlődik, könnyen integrálható más technológiákkal
Hátrány: Még nem teljesen stabil, a dokumentáció néhol hiányos.
Értékelés: Kisebb munkákra már most is kitűnően használható. Várhatóan fél éven belül a ***Rails Pythonban*** írt társát ünnepelhetjük benne.

Django

Előny: Kipróbált, skálázható, stabil, jól dokumentált, könnyen integrálható más rendszerekkel, portálok kialakításához ideális.
Hátrány: Csak ***Postgresql*** adatbáziskezelőt támogat (ezt a stabilitás miatt akár előnyként is felfoghatjuk)
Értékelés: Portálok kialakítására ideális választás.

keretrendszerrel lett gazdagabb, amely remek eszközeivel gyors webalkalmazás fejlesztést tesz lehetővé. Minden ***Python*** rajongónak csak ajánlani tudom.

Django

Ki nem hallott még a robusztus és mindenre használható ***Zope*** alkalmazás kiszolgálóról? Nos, a kihívó megérkezett. A ***Django***-t sokan csak így nevezik: „***Zope killer***”. Hogy miért? Pillanatokon belül kiderül.
Készítői újságok weboldalának kialakításához használták. Ebben íródott többek között a ➲ ***Lawrence.com*** és az ➲ ***Ljworld.com*** is. Összetett és hatalmas oldalak kialakítására tervezték, mégpedig úgy, hogy miközben követi egy napilap gyorsan változó és szövődményes tartalmát, a lehető legkevesebb energiát igénylje a fejlesztőktől. Lássunk néhány eszközt, amelyekkel megkönnyíti a munkánkat!
Hogy ne fáradjunk többet az ***SQL*** parancsok írásával, a ***Django*** segítségével adatbázisaink tartalmát mint objektumokat érhetjük el (***Object-relational mapper***). Azt hiszem ez a funkció alapvető minden komoly keretrendszernél. Láthattuk, hogy a ***Rails*** és a ***TurboGears*** is tartalmaz hasonló megoldást. Természetesen végrehajthatunk ***SQL*** kódot ha szükséges, de a piszkos munkát leveszi a vállunkról.
Szükségünk lenne pár adminisztrátori felületre? Semmi izgalom, a ***Django*** generálja majd őket helyettünk! Segítségével nagyon egyszerűen tudnak majd a felhasználóink tartalmakat hozzáadni és módosítani. És ez a funkció a fejlesztők szerint éles környezetben is hibamentesen használható. Érdeklődőknek ajánlom figyelmébe a ***Django*** dokumentációját. Nagyon, nagyon figyelemreméltó ez az eszköz, bár ezt csak az érezheti át igazán aki már alakított ki pár felhasználói felületet.
A modern web elvárásainak megfelelően a keretrendszer segítségével könnyen definiálhatunk állandó és rövid web címet oldalaink számára. Kézbe kapunk egy könnyen tanulható sablon rendszert is, mely úgy lett tervezve, hogy a grafikusaink is szívesen használják majd.
Megbizonyosodhatunk róla, hogy a ***Django*** professzionálisan lett kialakítva. Ugyanis beépítetten megtalálható benne egy rugalmas ***Cache*** rendszer, vagyis bizonyos tartalmakat nem renderel le minden lekéréskor, hanem megpróbálja amennyire lehet csökkenteni a terhelést azok gyorsítótárazásával.
Végül megemlítendő, hogy a nemzetközi oldalak készítése is problémamentes.
Mire is lehetne még szükségünk? Aki tehát nagy és heterogén oldalak kialakítását tervezi, gondolja át a terveket és használja a ***Django***-t! A ***Django*** honlapja a következő címen található: ➲ ***www.djangoproject.com***

Összefoglalás

Három kitűnő keretrendszert láthattunk, melyek segítségével a web programozása egyszerűbb mint valaha. Ismét megtapasztalhattuk a nyílt forráskód és a fejlesztők óriási közösségének teremtő erejét, hiszen mindhárom rendszer friss és naprakész. Röviden, kulcsszavakban összefoglalom most mindhárom rendszer előnyeit és gyengeségeit.

Juhász Attila
(rabszolga@goraffe.hu)
Az Információ Technológiai Kar hallgatója a Pázmány Péter Katolikus Egyetemen. Érdeklődik a bioinformatika és a neurális hálózatok iránt. A fotózás és a tánc mellett öt éve foglalkozik webgrafikákkal. A linux terjesztések közül a Gentoo és az Ubuntu áll legközelebb a szívéhez. Fotós oldala a http://people.goraffe.com/attila címen található.

KAPCSOLÓDÓ CÍMEK

Ruby on Rails weblap
➲ rubyonrails.org

Magyar Rails oldal
➲ rubyonrails.hu

Javascript effektek könyvtára:
➲ script.aculo.us

Radrails, egy Eclipse alapú integrált fejlesztői környezet (IDE)
➲ radrails.org

Hova Art Thou, ah egy hozzáférési pont?

Bizony mondom, egy vezetéknélküli hálózati kártya Linux meghajtó nélkül rosszabb mint összeesett felfújt. Mentsük meg tehát a vendégeinknek néhány ügyes trükkel a vacsorát.

Igen, *Francois*. Az elérési pont a kandalló mellett sokkal alkalmasabb lenne számodra. Az *ESSID*? Látni fogod a listában, ha rákeresel. *Micsoda? Mon Dieu, Francois* ugye nem egy szkriptet szerkesztesz éppen? Habár örülök neki, hogy otthonosan szeretnél mozogni a *Linux* konzolon, de sokkal egyszerűbb lenne kiválasztani az elérési pont keresését, majd rákattintani és egyszerűen használni.
Oh, már értem! A kártyádhoz tartozó *Linux* meghajtó nem támogatja a keresést. Igen, nekem is volt hasonló problémám a sajátommal, de van egy megoldásom. Rövidesen meg is fogom neked mutatni, de most kevés időnk van, a vendégeink néhány percen belül megérkeznek. A borospincébe, *Francois*. Egyenesen a pince déli szárnyába, és hozd fel az 1983-as *Batard Montrachet*-t. *Vite!*
Üdvözlöm, *mes amis*, *Chez Marcelnél a* rendkívüli Linux ételek hazájában, természetesen finom borokkal és csodálatos vendégekkel együtt. Mielőtt önök megérkeztek, az én hűséges pincérem és én megvitattunk pár problémát amit a vezetéknélküli kártyáinkkal kapcsolatban tapasztaltunk. A saját notebookom vezetéknélküli kártyája megfelelően működött az otthoni hálózaton, de éppen csak működött. A szabványos *Linux Orinoco* meghajtó ugyanis, amivel sikerült felélesztenem, nem támogatott olyan alapvető dolgokat, mint teszem azt a keresés. Minden hónapban, lementem a TV stúdióba, hogy felvegyem a showműsort, és minden hónapban meg kellett kérdeznem, hogy melyik vezetéknélküli útválasztót használhatom.

Ezt az információt ugyanis kézzel kellett begépelnem az *ifcfg-eth2* fájlba. A kártyához adott windowsos meghajtó persze támogatta ezt a lehetőséget, és amint ez gyakran megesik, a gyártók nem teszik teljesen nyilvánossá az általuk gyártott eszközök specifikációját, ami nem könnyíti meg a linuxos fejlesztők életét. Őszintén tisztelem azokat a hihetetlenül tehetséges és energikus embereket, akik ellátják a *Linuxot* tökéletes meghajtókkal mialatt a gyártók fekete dobozaival kell dolgozniuk. Ez az információhiány volt az ösztönzőereje az *NdisWrapper Projectnek*, amely egy betölthető kernelmodul segítségével lehetővé teszi a windowsos *Ndis* meghajtók használatát.
Lássuk, hogyan működik a dolog. Először le kell töltenünk az *NdisWarapper* projekt webhelyéről (lásd az on-line forrásokat) a legutóbbi verziót. A honlap szerint, ha egy friss *Linux* disztribúciót használunk, akkor először érdemes megnézni a telepítőlemezeket. Talán már ott is van a szoftver, csak telepítenünk kell. Aztán szükségünk lesz azokra a windowsos meghajtókra, amelyeket a hálózati kártyával együtt kaptunk. Különösen a kártyához tartozó **.INF* fájl fontos.
Én a *Presario* notebookomat egy beépített *LanExpress* kártyával kaptam. *Linux* alatt a kapcsolódást ugyan támogatta az *Orinoco* meghajtó, de – ahogyan már említettem – a keresés nem működött.
Mivel én valójában sohasem telepítettem *Windowst* a notebookomra, ellátogattam a *HP* webhelyére, és letöltöttem a meghajtót tartalmazó fájlt egy önkicsomagaló *EXE* formájában. (Az igazság az, hogy volt rajta előtelepített *Windows*, de mivel rögtön az első indítás előtt betettem egy *Linux CD*-t, *Windows* soha nem futott a gépemen.
A CrossOver Officet segítségével kicsomagoljam a szükséges fájlokat aztán megkeressem azt a könyvtárat, ahová a program települt. A meghajtót az *INF* fájl segítségével telepítettem, amihez root jogosultság szükséges:

```
ndiswrapper -i NetWlan.INF
Installing netwlan
```

A fenti kimenetet szemlélve nem tűnik úgy, mintha sok minden történt volna. A `-l` opció használatával kideríthetjük, hogy melyik meghajtó töltődött be, és megtekinthetjük annak állapotát is:

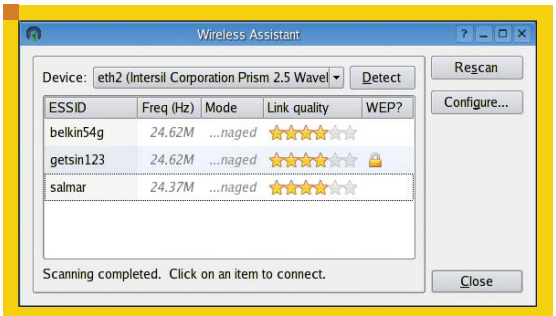
```
ndiswrapper -l
Installed ndis drivers:
netwlan driver present,
➥hardware present
```

A következő lépés az, hogy betöltsük a meghajtóprogramot a futó kernelbe, ami magának az *NdisWrappernek* a betöltését jelenti:

```
modprobe ndiswrapper
```

A következő kimenet a `dmesg` parancstól származik:

```
ndiswrapper version 1.2rc1
➥loaded (preempt=no,smp=no)
ndiswrapper: driver netwlan
➥(LAN-Express,01/18/2002,
➥1.07.29.20118) loaded
```



1. ábra A Wireless Assistant, hálózatok keresésére és a hozzájuk való kapcsolódásra alkalmas eszköz

```
ACPI: PCI interrupt 0
➥000:00:09.0[A] -> GSI 10
➥(level, low) -> IRQ 10
ndiswrapper: using irq 10
wlan0: ndiswrapper ethernet
➥device 00:02:8a:a9:e6:eb
➥using driver netwlan,
configuration file
➥1260:3873.5.conf
ndiswrapper (set_auth_mode:
➥584): setting auth mode
➥failed (C0010015)
wlan0: encryption modes
➥supported: WEP
```

Kiváló, be van töltve a w*indowsos* meghajtó a ***Linux*** rendszerünkbe, és készen áll arra, hogy használjuk. Ahhoz, hogy a fenti műveletsor minden rendszerinduláskor – misztikus módon – magától végbemenjen (a `dmesg` kivételével) felvettem minden utasítást a saját ***rc.local*** fájlomba. Most már lekérdezhetjük a közelünkben lévő elérési pontok listáját az `iwlist` parancs `scan` opcióját használva. Feltéve, hogy a vezetéknélküli hálózati kártyánk az ***eth2*** nevet viseli, a parancsnak így kell kinéznie

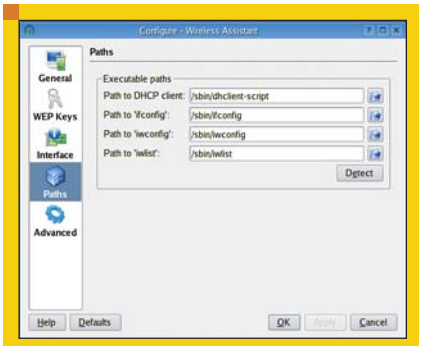
```
iwlist eth2 scan
```

Ezekután az iwconfig paranccsal csatlakozhatunk a kiválasztott hálózthoz, *IP* címet rendelhetünk az interfészhez, és így tovább. A teljes grafikus munkakörnyezet érzetéhez azonban hozzátartozna az is, hogy legyen olyan grafikus alternatíva, amivel hálózatokat tudunk keresni, ellenőrizhetjük a jel erősségét, és csatlakozhatunk az általunk kiválasztotthoz. Végül is a notebook is asztali eszköz valahol.
Az egyik legjobb általam talált eszköz erre *Pawel Nawrocki Wireless Assistant* programja. Ez egy szép felületű, kicsi program, mely egy kattintással lehetővé teszi az elérhető hálózatok keresését, és a hozzájuk történő csatlakozást. A megjelenítés minden hálózatnál tartalmazza a jelerősséget és a titkosítás módját. Az alkalmazás beállítható úgy is, hogy automatikusan kezelje a ***WEP*** kulcsokat, visszautasítson egyes hálózatokat (ad hoc vagy titkosított), minden kapcsolódásnál lefuttasson egy szkriptet, és még sok egyéb lehetőségünk is van. Az *1. ábrán* láthatjuk a programot bevetés közben.
A ***Wireless Assistant*** webhelyén (lásd források) elérhető a forráskód és számos különböző disztribúciókhoz lefordított bináris csomag is.
Talán érdemes megemlíteni, hogy a ***SourceForge-on*** csak a forráskód elérhető. Amennyiben a csomagokhoz tartozó fórumokat is követni szeretnénk, úgy a ***KDE-Apps.org*** honlap sokkal jobb hely arra, hogy információt keressünk rajta. A csomagot forrásból telepíteni a klasszikus ötlépéses műveletsorral lehet:

```
tar -xjvf wlassistant-
➥0.3.9.tar.bz2
cd wlassistant-0.3.9
./configure --prefix=/usr
make
su -c "make install"
```

A program neve valójában ***wlassistant***. Amikor a csomagot először elindítjuk, akkor automatikus megkeresi az aktív hálózati eszközöket. Amennyiben ez nem történne meg, akkor kattintsunk a ***Detect*** gombra. Amennyiben még mindig problémáink lennének, akkor valószínűleg a hálózati eszköz elérési útvonala van rosszul beállítva. Kattintsunk a ***Configure*** gombra, és a beállítás párbeszédpanel megjelenik. A panel baloldalán vannak kategóriánként felsorolva a lehetőségek, és az aktuális beállításokat a jobb oldalon található ablakban láthatjuk, módosíthatjuk. Kattintsunk a ***Path*** gombra, hogy megerősítsük a hálózati kártya elérési útját *(2. ábra)*. Ezt manuálisan is beállíthatjuk, vagy automatikusan a ***Detect*** gombra történő kattintással.

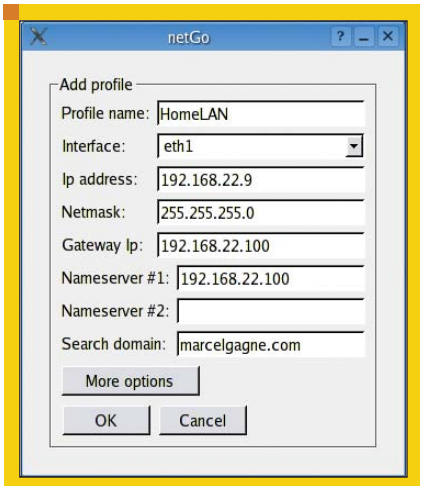
Szenteljünk némi időt a beállítás párbeszédablak tüzetesebb áttanulmányozására. Miután elvégeztük a szükséges a beállításokat, kattintsunk az ***OK*** gombra és térjünk vissza a ***Wireless Assistant*** főmenüjébe. Még mindig nem vagyunk készen! Most kattintsunk a ***Rescan*** gombra, hogy beazonosítsuk az elérhető hálózati helyeket *(1. ábra)*. Amint a képen látszik, számos elérhető hálózat közül válogathatunk. A program azt is megmutatja, hogy egy elérési pont használ-e ***WEP*** titkosítási protokollt. Ezt használni mindig ajánlatos, kivéve ha olyan nyílt elérési pontot akarunk üzemeltetni, amit bárki használhat, aki arra jár.
Kattintsuk egy általunk kiválasztott hálózatra, majd a felbukkanó ablakban gépeljük be a root jelszavát *(3. ábra)*. És ennyi! A hálózati kapcsolatunk máris működik.
Mikor régen ügyfeleket látogattam meg, és rákapcsolódtam különféle vezetékes hálózatokra, készítettem egy apró szkriptet, ami az ***ifcfg-ethX***-et és hálózat beállításait oda-vissza másolta, aszerint, hogy aznap éppen



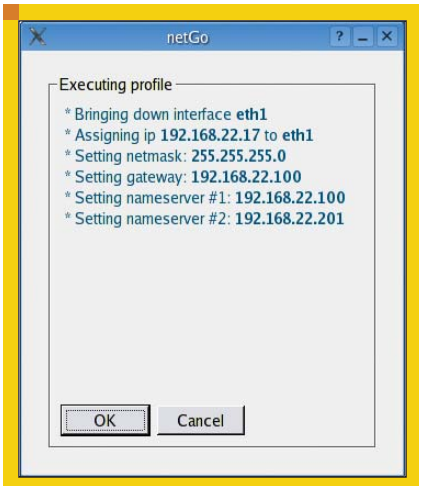
2. ábra Használjuk a Wireless Assistant beállítás párbeszédpaneljét, hogy megadjuk a vezetéknélküli kártya elérési útját

3. ábra Mielőtt megváltoztatnák a hálózati beállításainkat, meg kell adnunk a root jelszavát

4. ábra A netGo megkönnyíti sok hálózati profil felállítását, és karbantartását



5. ábra Kapcsolat részleteinek bevitele a netGo profiljába



6. ábra A netGo visszajelzést ad a profilváltást követő változásokról

melyik helyet látogattam meg. Működött, de nem volt éppen a legelegánsabb megoldás. Ugyanez a probléma a vezetéknélküli kapcsolatok világában is megvan. Ha az egyik elérési ponttól a másikhoz vándorolunk, mondjuk irodáról irodára, aztán utána haza, akkor nyilván hasznos számunkra egy olyan eszköz, amivel az összes hálózati profilt kézben tarthatjuk.

Ez az alapgondolata ***Per Johansson netGo*** programjának is *(4. ábra)*.

A ***netGo*** egy remek kis alkalmazás, amely lehetővé teszi számunkra különbözőféle hálózati profilok létrehozását, a közöttük történő váltás pedig mindössze egy kattintás. Amikor éppen nem használjuk, az alkalmazás a tálcán bújik meg. A program a ***Qt*** könyvtárat használja, így tökéletesen működik ***KDE***, ***GNOME*** és más ablakkezelők alatt is.

Letöltéséhez látogassunk el a ***netGo*** honlapjára (lásd forrásokat). Ha forrásból telepítjük a programot, akkor a már ismertetett öt lépéses kicsomagolás és fordítás műveletét kell elvégeznünk, tehát nincs miért aggódnunk. A program futtatásához adjuk ki a ***netgo*** parancsot. Ennél a pontnál meg kell adnunk a root jelszavát, hiszen csak így tudjuk a hálózati címeket változtatni.

Kezdetben a főablak egyetlen profilt sem tartalmaz. A új kapcsolatok felviteléhez kattintsunk az ***Add profile*** gombra, mire egy új ablak jelenik meg *(5. ábra)*. Legfelülre írjuk be a profil nevét, például ***LAN_Otthon*** vagy ***Kávézó***, majd válasszuk ki a hálózati kártyát. A legtöbb notebookban van beépített ***10/100 Ethernet*** kártya, és vezetéknélküli kártya is. Azoknál a kapcsolatoknál, melyek statikus ***IP*** címet igényelnek, töltsük ki az összes az ***IP*** címhez kapcsolódó mezőt, beleértve a hálózati maszkot és a többit. Ha készen vagyunk, kattintsunk az ***OK*** gombra, és mentsük a profilunkat. Amennyiben az általunk felállított kapcsolat vezetéknélküli, kattintsunk a ***More options*** gombra. Ezután kiválaszthatjuk, a hálózat tulajdonságait – ***ad hoc***, ***managed***, ***none*** – , bevihetjük az ***ESSID***-et és megadhatjuk a ***WEP*** kulcsunkat. Figyeljünk oda a ***Custom script*** mezőre is. Ennek a segítségével automatikusan elvégezhetünk egy sor beállítást és műveletet, például a tűzfal felállítását. Az itt megadott parancsok azonnal lefutnak, amint életre keltjük az interfészt. A főmenübe a ***Back*** gombra kattintva tudunk visszalépni. Hasonló módon folytathatjuk a profilok létrehozását a végtelenségig.

Egy profil aktiválásához, és a hálózati konfiguráció érvénybe léptetéséhez kattintsunk a profil nevére, majd a ***Go!*** feliratú gombra. Erre megjelenik az állapot ablak az új, már érvényben lévő beállításokkal *(6. ábra)*.

Az egyetlen komolynak nevezhető probléma, amivel találkoztam az, hogy a netGo jelenleg nem támogatja a másodlagos interfész leállítását, így ezt egyelőre kézzel kell elvégezni. Ugyanakkor érdemes megemlíteni, hogy a szerző honlapja szerint ez a funkció éppen kidolgozás alatt áll.

Nos, ***mes amis***, ha annak az órának ott a falon hinni lehet lassan itt a záróra. Mindazonáltal, biztos vagyok benne, hogy ***Francois*** nem bánja, ha egy kicsit tovább tartunk ma nyitva. Még annyi ideig, hogy utoljára teletöltsék poharaikat. Még kihozhatunk egy kicsit abból a duplán vajas ***Brie***-ből kíséretképpen a borhoz. Tekintve hogy immár mindannyian vezetéknélküliek vagyunk, kihozhatják a notebookjaikat a teraszra is, ahol még tovább élvezhetjük együtt az estét, amíg mindenki haza nem indul. Emeljük poharainkat, ***mes amis***, és igyunk egymás egészségére.

A ***votre santé! Bon appetit!***

Linux Journal 2005. 137. szám

Marcel Gagné
mggagne@salmar.com; www.marcelgagne.com

Díjnyertes író, az ontarioi Mississaugában él. Immár harmadik könyve a Moving to the Linux Business Desktop (ISBN 0-131-42192-1, Addison-Wesley). A Call for Help linuxos szereplőjeként rendszeresen látható a tévében is. Marcel hobbipilóta, Top-40-es lemezlovas volt, tudományos-fantasztikus írások szerzője, jelenleg egy kisebb origami T-Rexen dolgozik.

Források

- www.linuxjournal/article/8398

XML Túra (2. rész)

A cikk előző részében átvettük az XML nyelv alapjait. Most meglátjuk, hogyan is lehet saját XML szabályokat alkotni, valamit a cikk végén olvashatunk keveset arról, hogy hogyan tudjuk HTML segítségével az XML-ben tárolt adatokat elérni.

Az érvényes XML dokumentum és a sémák

Mielőtt hozzákezdenénk a *DTD* tárgyalásához, elevenítsük fel az előző cikkből, hogy mit is nevezhetünk érvényes *XML* dokumentumnak. ***Érvényes XML dokumentum az olyan dokumentum, mely megfelel a felhasználó által definiált tartalmi szabályoknak.*** Egy ilyen tartalmi szabályokból álló halmazt nevezünk ***sémának***. Az *XML* séma az *XML* dokumentum típusának leírása, a megkötéseken túl korlátozásokat tartalmaz az adott típus struktúrájára és tartalmára. Nagyon sok szabványos és szabadalmaztatott *XML* séma jelent meg bizonyos megkötések leírására. Nagyon sok eszköz áll rendelkezésre mely a sémákkal szemben validálni tudja a dokumentumokat. De a sémáknak nem csak a szabványosítás a haszna, bizonyos szerkesztők a definiált szabályok segítségével meg tudják könnyíteni a dokumentumok létrehozását. Tehát a séma egyfajta „tervrajz" az *XML* dokumentumokhoz. Vegyük például a *WML* nyelvet, mely *WAP* oldalak írására alkalmas, szabványos nyelv. Ez is egy megfelelő , a *WML* szabályait rögzítő sémával ellátott *XML* dokumentum. Ismertebb *XML* sémák a *XSD* (*XML Schema Definition*), *Relax NG* és a *DTD*. A következőkben a *DTD*-ről lesz szó mivel ez az *XML 1.0* szabvány része és ezt tekinthetjük a legelterjedtebbnek.

DTD

A *DTD* az *XML* dokumentumban a fejléc után bárhová elhelyezhető. Általános formája:

1. Lista – Egy nagyon alap DTD

```
<?xml version="1.0"
➥encoding="ISO-8859-2" ?>
<?xml-stylesheet
➥type="text/css"
href="dolgozok.css" ?>
<!- XML Túra 2. rész ?
➥Dolgozók listája ->

<!DOCTYPE dolgozók
[
    <!ELEMENT dolgozók
    ➥(#PCDATA)>
    <!- a dolgozók elem
    ➥deklarációja ->
]
>
    <dolgozók>Kovács
    ➥Elek</dolgozók>
```

```
<!DOCTYPE dokumentumelem_neve
➥DTD>
```

A dokumentumelem azon elem melynek jellemzőit, egyedeit, jelölési deklarációit és feldolgozási utasításait szeretnénk definiálni. A *DTD* itt most az előző mondatban felsoroltakat tartalmazza. Persze a legvilágosabb rálátásunk akkor lesz , ha végre látunk egy példát, mely szemlélteti az elhangzottakat *(1. lista)*. Tehát az *1. Lista* egy érvényes *XML* dokumentum, melynek egyetlen eleme a ***dolgozók***. A ***dolgozók*** elemnek csak szöveges tartalma lehet. A szöveges tartalomra a `#PCDATA` kifejezés utal (***Parsed Character Data***).

Az ELEMENT-ről bővebben

Az elemtípusok deklarálásának általános alakja:

```
<!ELEMENT elem_neve tartalom>
```

Az elem nevének megadásakor ügyeljünk a kicsi és nagy betűk használatára. A tartalom a következőket adhatja meg:

`EMPTY` – üres elem :

```
<!ELEMENT URES_ELEM EMPTY>
```

`ANY` – bármilyen szöveges, vegyes tartalom:

```
<!ELEMENT BARMILYEN ANY>
```

Előfordulhat, hogy egy elem más elemeket tartalmaz, ekkor fel kell sorolni a tartalmazott elemeket:

```
<!ELEMENT dolgozó (név,
➥osztály, kor, fizetés)>
```

majd utána definiálni kell azokat *(2. Lista)*.

Tehát a *2. Lista* ***DTD***-jének szöveges változata. A `doctype` után a dokumentum ***gyökérelemének*** neve következik, majd az egyes elemek deklarációja. A dolgozók elem elemtartalommal rendelkezik, mégpedig bármennyi (*) dolgozót tartalmazhat. Ha elhagynánk a csillagot, abban az esetben a dolgozók elem csak egyetlen dolgozót tartalmazhatna. Több ilyen, az elemek számára utaló jelet használhatunk. Nulla vagy egy darab az előtte álló elemből: ? , egy vagy

2. Lista – Elemek definiálása

```
...
<!DOCTYPE dolgozók
[
    <!ELEMENT dolgozók
    ➥(dolgozó)*>
     <!ELEMENT dolgozó (név,
     ➥osztály, kor,
     ➥fizetés)>
     <!ELEMENT név (#PCDATA)>
     <!ELEMENT osztály
     ➥(#PCDATA)>
     <!ELEMENT kor (#PCDATA)>
     <!ELEMENT fizetés
     ➥(#PCDATA)>
    <!-- a dolgozók elem
    ➥deklarációja –>
]
>
...
```

3. lista – Jellemzők definiálása

```
...
<!DOCTYPE dolgozók
...
<!ATTLIST dolgozó külföldi
➥CDATA "nem" neme CDATA
➥#REQUIRED>
...
>
...
```

több: **+** , nulla vagy több: * .
A dolgozó elem a név, osztály, kor, fizetés elemeket fogja tartalmazni, sorrendben.
A ***sorrend*** fontos. Mivel nem használ tunk semmilyen elemszámra utaló jelölést, így mindegyiket pontosan ***egyszer*** kell hogy tartalmazza. Ha most a dolgozó elem definíciójában a vesszőket | -ra cseréljük akkor a következő

```
<!ELEMENT dolgozó(név | osztály
➥| kor | fizetés )>
```

definíció azt fogja jelenteni, hogy a dolgozó elem a felsoroltak közül legalább az egyiket tartalmazza.

Jellemzők

Jellemzők definiálása a következőképpen lehetséges:

```
<!ATTLIST ELEM_NEVE
➥jellemző_neve jellemző_típusa
➥jellemző_alapértelmezett_tart
➥alma ...>
```

Az `elem neve` azon elem, melynek éppen a jellemzőit szeretnénk megszabni. A következő a sorban a `jellemző neve` majd az `adattípusa` és végül az `alapértelmezett értéke`, melyet akkor vesz fel ha nem adják meg az értékét. Az alapértelmezésnél adhatjuk meg, hogy az éppen soron lévő jellemző megadása kötelező vagy sem. Lássunk erre is példát: *(3. Lista)*.
Azaz a *3. Lista* szerint a dolgozó elem külföldi nevű jellemzőjének alapértelmezésben nem az értéke és a neme nevű jellemző megadása kötelező.
Az egyes jellemző típusait ***három*** nagy csoportba sorolhatjuk: ***karakterlánc, token típusú és felsorolás típusú***.
Nézzük őket szépen sorjában.
A ***karakterlánc*** a fentiekben a *3. lista* alapján ki lett tárgyalva. A ***token típusok*** egyike az ***ID***. Az ilyen típusú jellemző egyfajta kulcsként szolgál, tehát két jellemzőnek mely ***ID*** típusú, nem lehet azonos tartalma:

```
<!ATTLIST AUTÓ motorszám ID
➥#REQUIRED>
...
<AUTÓ motorszám="386">
<AUTÓ motorszám="486">
<AUTÓ motorszám="386"> <!--
HELYTELEN, MERT MÁR LÉTEZIK
➥ILYEN MOTORSZÁMÚ AUTÓ!!! -->
...
```

A második token típus az `IDREF`, mely tulajdonképpen egy mutató egy már létező `ID` típusú jellemzőre:

```
<!ATTLIST SZÜLŐ személyi_szám
➥ID #REQUIRED>
<!ATTLIST GYEREK személyi_szám
➥ID #REQUIRED szülője IDREF
➥#REQUIRED>
...
<SZÜLŐ személyi_szám=
➥"23456789">
<GYEREK személyi_szám=
➥"010100111"
szülője="23456789">
```

A harmadik az `IDREFS`, mely vehető az `IDREF` többszörösének. A fenti példát tovább fejlesztve:

```
<!ATTLIST GYEREK ... szülője
➥IDREFS  #REQUIRED>
...
<SZÜLŐ személyi_szám=
➥"23456789">
<SZÜLŐ személyi_szám=
➥"12345678">
<GYEREK személyi_szám=
➥"010100111" szülője="23456789
➥12345678">
```

Az egymás után megadott `ID` mutatókat egymástól ***szóközzel*** választjuk el. A negyedik tagja ennek a csoportnak az `NMTOKEN` típus. Értéke lehet ***pontok, számok, betűk, kötőjelek és aláhúzások sorozata***. Kettőspont is szerepelhet benne, kivétel az első karaktert. Az `NMTOKENS` ennek a sokszorosított változata. Egy jellemzőn belül több `NMTOKEN` típusú értéket adhatunk meg szóközökkel elválasztva.
Az utolsó kettő az `ENTITY` és az `ENTITIES`. Az utóbbi az első sokszorosítottja a fentebb leírt módon.
Az `ENTITY` értéke egy a *DTD*-ben már deklarált, de még nem értelmezett ***egyed*** lehet.
A token típus után van még egy típus fajta amiről nem esett szó, ez a ***felsorolás***:

```
...
<!ATTLIST dolgozó neme =(férfi
➥| nő) #REQUIRED>
...
```

A fenti példa leírja, hogy a dolgozó elem kötelező ***neme*** jellemzője vagy férfi vagy nő értéket vehet fel.
A `#REQUIRED` kulcsszó párjaként említhetjük még a `#IMPLIED`-et, mely jelzi az értelmezőnek, hogy a jellemzőt ***nem kötelező*** megadni, azaz ha elhagyják, akkor is érvényes lesz a dokumentum.

Kívül vagy belül?

Akik huzamosabb ideje foglalkoznak *SGML* nyelvekkel, azok szerintem már előre ráéreztek annak lehetőségére, hogy a *DTD*-t az azt felhasználó dokumentumon kívül is el lehessen tárolni, majd arra hivatkozni, mint például a *CSS* esetében is tettük

az előző cikkben. Egy másik állományban tárolt *DTD*-t nevezünk külső *DTD*-nek a felhasználó *XML* szempontjából. Belső *DTD*-ről beszélünk, akkor ha a definíciók abban az XML-ben vannak megírva, melyekben felhasználjuk őket, mint például az eddigi esetekben.
Tiszta megértéshez mindig jól jön egy példa *(4. lista)*.
A ***dolgozok.xml***-ben a következőképpen hivatkozhatunk a külső DTD-re:

```
...
<!DOCTYPE dolgozók SYSTEM
➥"dolgozok.dtd">
...
```

Lehetőség van a külső DTD hivatkozásával egyidejűleg még további definíciók megadására is:

```
<!DOCTYPE dolgozók SYSTEM
➥"dolgozok.dtd"
[
    <!ATTLIST név főnök (igen
    ➥| nem) #REQUIRED>
]
>
```

Ekkor a két definícióhalmaz , a külső és belső DTD unióját tekintjük a teljes dokumentum DTD-jének.

Egyed

Az *ENTITY* típus leírásánál került először szóba ez a kifejezés. Az egyedeket legkönnyebben ***konstansokként*** definiálhatjuk. Egy egyedhez társíthatunk ***DTD-t, jellemzőt, elemet***, és segítségével pedig akárhányszor ***hivatkozhatunk*** rá a dokumentumban. Ezeket szintén a *DTD*-ben kell deklarálni. Ügyes használatukkal a dokumentumunk méretét jelentősen csökkenteni tudjuk.
Itt is lehetőség adódik arra, hogy külső állományból hívjuk meg az egyed értékét. Legyen az ***egyed.xml*** tartalma:

```
<név>Szabó Ubul</név>
```

Ekkor az *5. Listában* az összetett egyed deklarációját a következőre lehet módosítani:

```
<!ENTITY összetett SYSTEM
➥"egyed.xml">
```

Fent említettük, hogy az egyed tartalmazhat *DTD* deklarációt is. Itt is ugyanúgy meghívhatunk külső *DTD*-t egy egyed értékeként. Viszont a szintaxis egy kicsit másképp fog alakulni:

```
<!ENTITY % egyed_neve
➥"<!ELEMENT ...>...">
```

A *DTD*-ben pedig a következőképpen tudunk rá hivatkozni:

```
%egyed_neve;
```

Ha külső *DTD*-t szeretnénk egy egyedre ráhúzni, akkor pedig a következő a módszer:

```
<!ENTITY % kulso_dtd SYSTEM
➥"szabalyok.dtd">
```

A DSO-ról röviden

A következőkben egy olyan ***adatkapcsolásos*** technikáról lesz szó, melyet sajnos – tudomásom szerint – a ***Linux*** alatti böngészők nagy része nem támogat, de egy másik bizonyos operációs rendszer böngészője igen. Mivel itt pedig az *XML*-ről van szó és szerintem ez a technika nagyon jól kihasználja az *XML*-t, gondoltam megér pár sort. Tehát lássuk hogyan tudunk *HTML* dokumentumba *XML* adatokat illeszteni. Nem meglepő módon kétféleképpen is megtehetjük:

```
<HTML>
...
<XML ID="kozvetlen_xml">
<!-- ide jön az XML kód -->
</XML>
...
<XML ID="kozvetett_xml"
➥SRC="forras.xml">
...
</HTML>
```

Az ilyen objektumot nevezzük *Data Source Object*-nek. Az adatainkhoz az `ID`-n keresztül fogjuk megtalálni az utat, ezért mondható, hogy ez a jellemző kötelező.
A *HTML*-ben több adatbefogadó elem is létezik: `SPAN`, `DIV`, `TABLE`. Ezek közül az első kettő csak egy elem tulajdonságait tudja megjeleníteni, például:

4. Lista – Külső DTD megvalósítása (a dolgozok.dtd tartalma)

```
<?xml version="1.0"
➥encoding="ISO-8859-2">
    <!ELEMENT dolgozók
    ➥(#PCDATA)*>
    <!ELEMENT dolgozó (név,
    ➥osztály, kor, fizetés)>
    <!ELEMENT név (#PCDATA)>
    <!ELEMENT osztály
    ➥(#PCDATA)>
    <!ELEMENT kor (#PCDATA)>
    <!ELEMENT fizetés
    ➥(#PCDATA)>
    <!ATTLIST dolgozó
    ➥külföldi CDATA "nem"
    ➥neme CDATA #REQUIRED>
```

5. lista – Egy egyszerű egyed deklarációja

```
<!DOCTYPE dolgozók
[
   ...
   <!ENTITY név "Kovács Elek">
   <!ENTITY összetett
   ➥"<név>Szabó Ubul</név>"
   ...
]
>

<dolgozók>

<dolgozó>&összetett;</dolgozó>

    <dolgozó>
     <név>&név;</név>
   </dolgozó>
</dolgozók>
```

```
...
<XML ID="dolgozokdb"
➥SRC="dolgozok.xml" >

<SPAN DATASRC="#dolgozokdb"
➥DATAFLD="név"></SPAN>
<SPAN DATASRC="#dolgozokdb"
➥DATAFLD="kor"></SPAN>
...
```

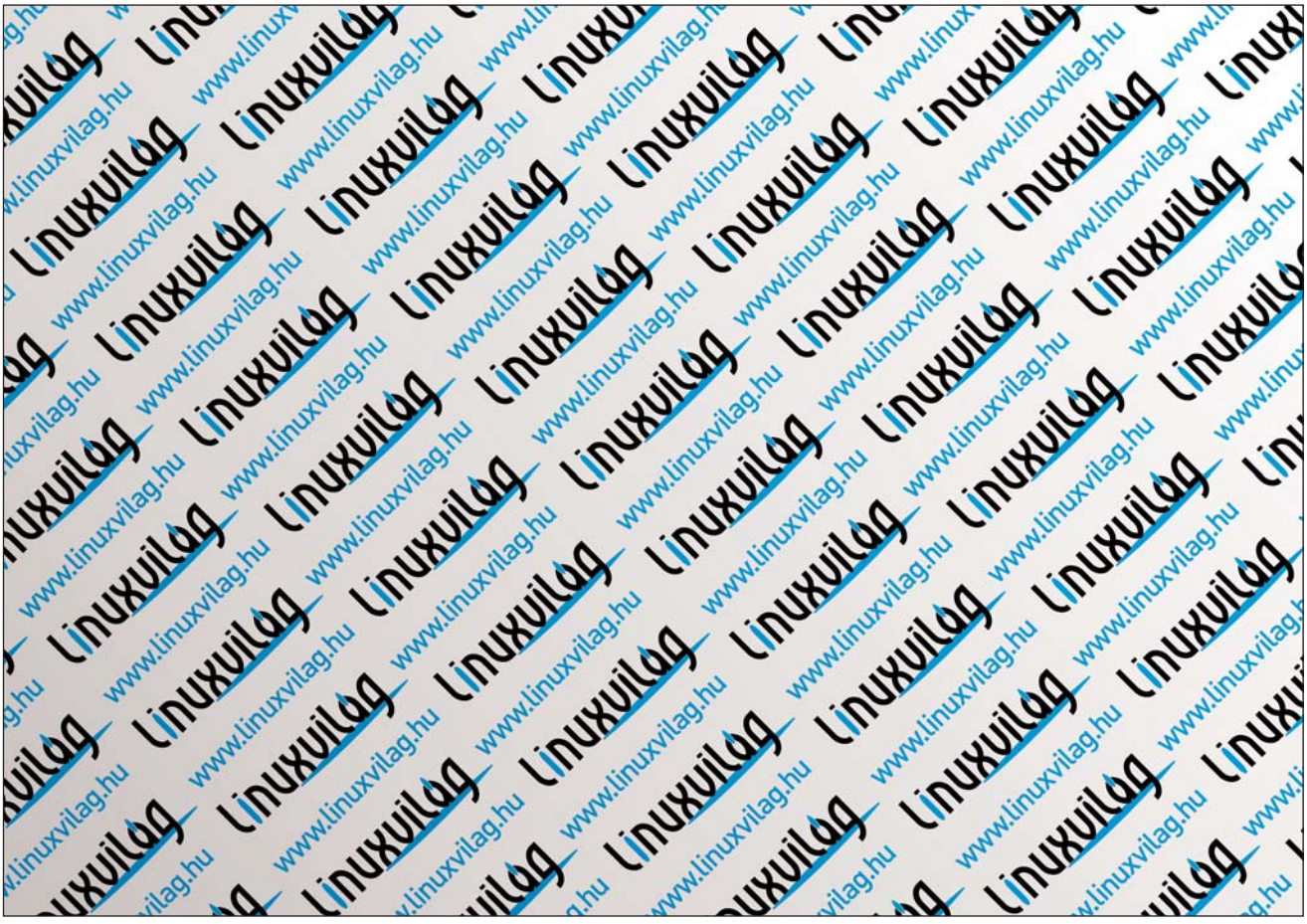
Láthatjuk, hogy a DATASRC jellemző azonosítja a ***DSO objektumot***, amíg a DATAFLD a megjelenítendő adat ***oszlopát*** adja meg. Mivel ez csak egy adat megjelenítésére szolgál, lássuk az ebből a szempontból barátságosabb megjelenítést a TABLE segítségével:

```
...
<XML ID="dolgozokdb"
➥SRC="dolgozok.xml">

<TABLE DATASRC="#dolgozokdb"
➥BORDER="1">
        <TR><TD><DIV
        ➥DATAFLD="név"></DIV>
        ➥</TD></TR>
        ...
        <TR><TD><DIV
        ➥DATAFLD="fizetés">
        ➥</DIV></TD></TR>
</TABLE>
```

Ezzel a módszerrel már minden ***dolgozók*** elembe tartozó ***dolgozó*** elem adait láthatjuk szépen táblázatba szedve.
Betekintést nyerhettünk az *XML* alapjaiba. Szerintem már sokan látják, hogy nem nehéz dolog.
A *W3C* emberei ügyesen dolgoztak, hiszen könnyen tanulható, ennek köszönhetően gyorsan elterjedt. Biztos, hogy sokan hallottak már az *RSS*-ről is (***Rich Site Summary***), mely szintén *XML* alapú, webtartalom elosztására és publikálására alkalmas formátum csoport. Főleg az újabb oldalak, hírportálok, blogok használják előszeretettel az ***Atom*** nevű *RSS* formátumot. Az *XML* nem kizárólag az interneten használt formátum. Mobiltelefonok előszeretettel alkalmazzák akár ***képekről***, akár ***videókról*** legyen szó.
Mint már említettem, az *OpenOffice* is *XML* formátumban menti a dokumentumait, majd azokat tömöríti egy állománnyá. Pár nappal ezelőtt a barátnőm fényképezőgépéről szedtem le az általa készített képeket és szemembe ötlött egy *XML* állomány, melyben a fényképezőgép tárol egy csomó információt a gép használatáról, például hány kép készült eddig, hányszor volt használva a zoom funkció, milyen a gép szoftverének verziója.
Az *MP3* lejátszók többsége is a beállításait *XML* dokumentumokban őrzi a bennük lévő memórián.
Az *XML* nagyon széleskörű eszköz. A lényeg, hogy az adatok mind ember mind számítógép által aránylag könnyen feldolgozható formában jelennek meg. Az a tulajdonsága pedig, hogy saját szabályokat, sémákat alkothatunk például *DTD*-k segítségével, csak további előnyökkel jár.
Remélem hasznos betekintést nyújthattam az olvasónak az *XML* változatos, színes világába.

Radics Péter
(peter.radics@gmail.com)

Az ELTE-n tanulok programtervező matematikus szakon. Hobbim a kosárlabda, autóvezetés, webdesign, programozás. Főleg webes alkalmazások fejlesztése érdekel. 4 éve megrögzött Linux felhasználó vagyok.



KOffice – irodai alkalmazás-család KDE alapokon (1. rész)

Szövegszerkesztés, táblázatok, bemutatók

A KOffice a KDE grafikus asztali környezetbe illeszkedő irodai alkalmazás-gyűjtemény. Néhol többet, néhol kevesebbet nyújt mint más hasonló irodai csomagok. A többi hasonló csomag véleményezését mellőzve, megpróbáljuk bemutatni a KOffice lényeges képességeit. A KOffice bemutatójának ezen első részében a csomag lelkét képező elemekre koncentrálok: szövegszerkesztés, táblázatkezelés, bemutató készítés.

Ismerkedés

A *KOffice* egy célzottan *KDE* környezet alá írt, ennek könyvtárait és felhasználói felület-elemeit használó, abba illeszkedő irodai alkalmazások gyűjtőneve. Az ***irodai*** jelző gyakorlatilag azt jelenti, hogy dokumentum-, kiadvány-, bemutató-, kép- és grafikakészítéssel kapcsolatos tevékenységek egységbe kapcsolt eszközeiről van szó. Ennek az elvnek megfelelően a *KOffice* mindezekhez kapcsolódóan tartalmaz eszközöket, illetve számos kiegészítő alkalmazást is.

Alapvető kérdés lehet az, hogy *Linux* alatt elérhető irodai alkalmazások között miért válassza valaki a *KOffice*-t az említett jellegű feladatok elvégzésére. A választás egyik oka természetesen a rendelkezésre álló eszközök és képességek – amelyeket megpróbálok bemutatni a továbbiakban. További ok lehet a tökéletes illeszkedés a *KDE* környezetbe, a felkínált eszközök sokrétűsége, a kezelhetőség, esetenként akár az egyszerűség is. A választáskor természetesen ugyanúgy tudnunk kell az esetleges hiányosságokról, mint az előnyökről, így ezekről is igyekszünk majd szót ejteni.

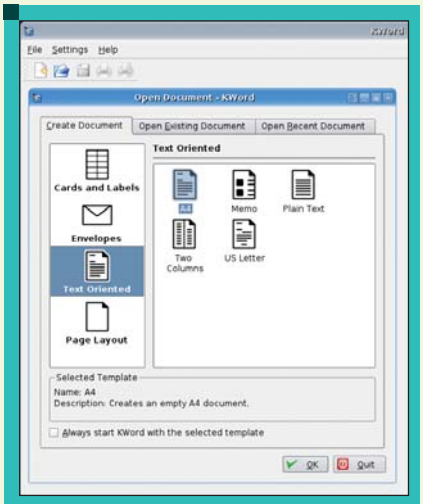
Ebben az írásban a *KOffice 1.4.2* stabil verzióját mutatjuk be, *KDE 3.5.1* környezet alatt. Az alkalmazás-csomag konkrét telepítésére nem térünk ki azért, mert gyakorlatilag minden *Linux* disztribúció adott csomagkezelőjével *koffice* név alatt megtalálható és könnyedén telepíthető. A csomag egyes elemei általában egyenként is elérhetők, telepíthetők és használhatók. A *KOffice* elemei a következők: *KWord* – szövegszerkesztés, *KSpread* és *KChart* – táblázat-kezelés és grafikon vizualizáció, *KPresenter* – bemutató készítés, *Kexi* – adatbázis-kezelés, ill. kiegészítő alkalmazások: *KFormula* – egyenlet-szerkesztés, *Kivio* – folyamatábra készítés, *Karbon14* – vektorgrafikus rajz/grafika-készítés, *Krita* – kép-manipulálás és egyszerű rajzolás, *Kugar* – report/beszámoló készítés.

Szövegszerkesztés – KWord

A *KWord* egy olyan szövegszerkesztő, amely egyrészt minden hétköznapi szövegszerkesztési elvárásnak képes megfelelni, másrészt számos olyan eszközt is tartalmaz, amellyel irodai alkalmazás esetén lehet szükséges ill. elvárható.

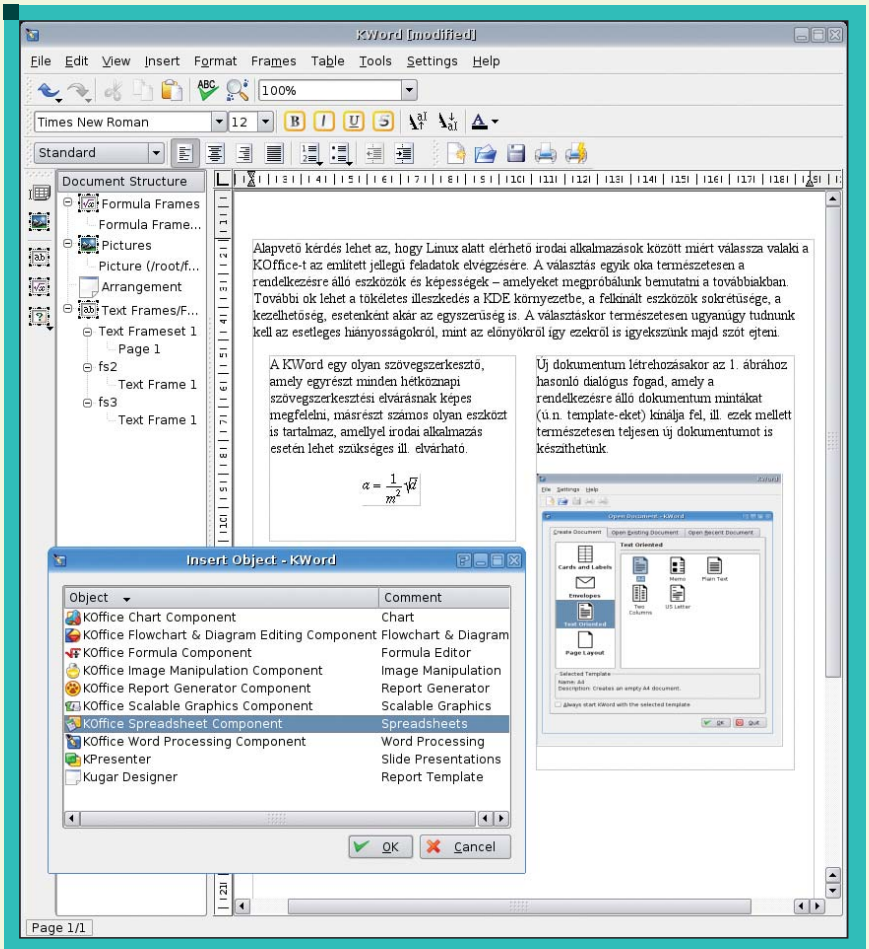
Új dokumentum létrehozásakor az *1. ábrához* hasonló dialógus fogad, amely a rendelkezésre álló dokumentumsablonokat (***template***) kínálja fel, ill. ezek mellett természetesen teljesen új dokumentumot is készíthetünk.

A *KWord* alapvető funkciói – a betű-, szöveg-, bekezdés-, oldal-, lábjegyzet-, fejléc-, tartalomjegyzék-, táblázatszerkesztési lehetőségek – azok, amelyeket joggal elvárhatunk egy szövegszerkesztő alkalmazástól, és ezek használata ill. funkciói nem térnek el más szövegszerkesztő alkalmazások hasonló eszközeitől, így ezekre részletesen nem térünk ki. Az említett eszközök a *Format* és a *Table* menükben érhetők el. A továbbiakban olyan funkciókra és eszközökre koncentrálunk, amelyek egyedivé teszik a *KWord*-öt mint szövegszerkesztőt: *frame*-ek használata,

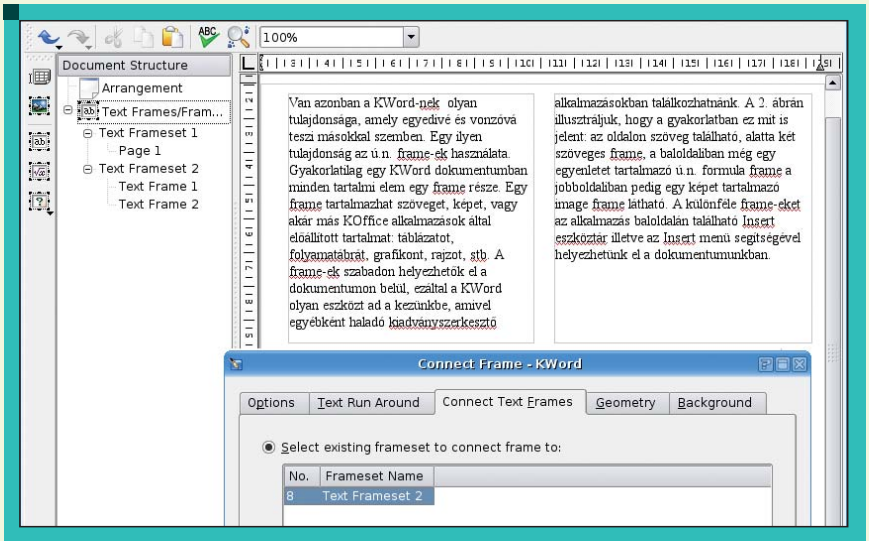


1. ábra Új dokumentum létrehozásakor választhatunk a dokumentum minták közül

magas szintű integráltság és átjárhatóság a *KOffice* más elemeivel, könnyen használható irodai funkciók, támogatott dokumentum-formátumok.
Van azonban a *KWord*-nek olyan tulajdonsága, amely egyedivé és vonzóvá teszi másokkal szemben. Egy ilyen tulajdonság az úgynevezett *frame*-ek használata. Gyakorlatilag egy *KWord* dokumentumban minden tartalmi elem egy *frame* része. Egy *frame* tartalmazhat szöveget, képet, vagy akár más *KOffice* alkalmazások által előállított tartalmat: táblázatot, folyamatábrát, grafikont, rajzot, stb. A *frame*-ek szabadon helyezhetők el a dokumentumon belül, ezáltal a *KWord* olyan eszközt ad a kezünkbe, amivel egyébként haladó kiadványszerkesztő alkalmazásokban találkozhatnánk. A 2. *ábrán* illusztráljuk, hogy a gyakorlatban ez mit is jelent: az oldalon szöveg található, alatta két szöveges *frame*, a baloldaliban még egy egyenletet tartalmazó úgynevezett *formula frame a* jobboldaliban pedig egy képet tartalmazó *image frame* látható. A különféle *frame*-eket az alkalmazás baloldalán található *Insert* eszköztár illetve az *Insert* menü segítségével helyezhetünk el a dokumentumunkban.
Frame-eket használhatunk pl. arra is, hogy könnyen kezelhető, változó méretű, pozíciójú és alakú hasábokat hozzunk létre. Ehhez csak létrehozunk szöveges *frame*-eket (*Insert->Text frame*) és összekapcsoljuk őket, amire a létrehozáskor van lehetőség (*3. ábra*). Ezzel szabad kezet kapunk a hasábok elrendezéséhez és méretezéséhez.

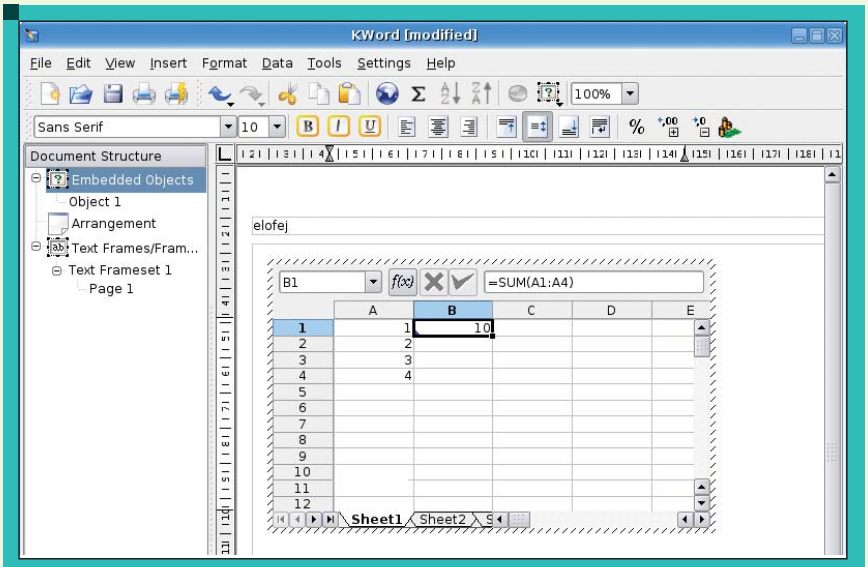


2. ábra Néhány egyszerűbb frame elhelyezése egy dokumentumban, ill. egy másik KOffice alkalmazás által létrehozandó új objektum-frame beszúrásának dialógusa

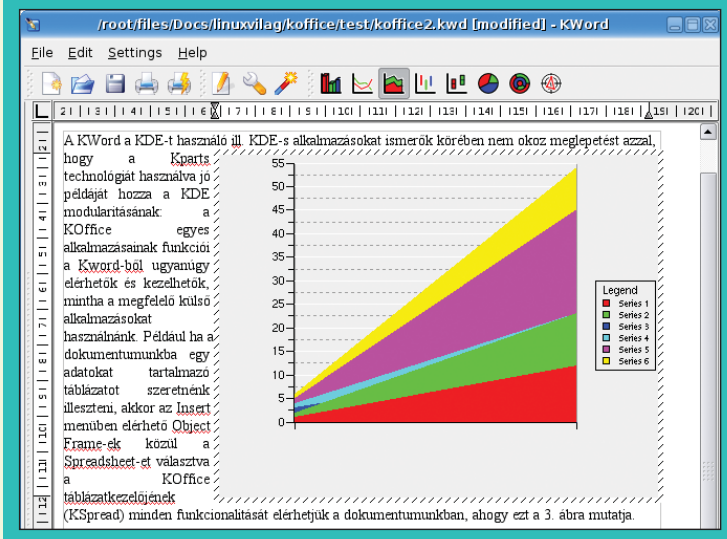


3. ábra Összekapcsolt szöveges frame-ek létrehozása, ami szabadformájú hasábos szövegelrendezést tesz lehetővé

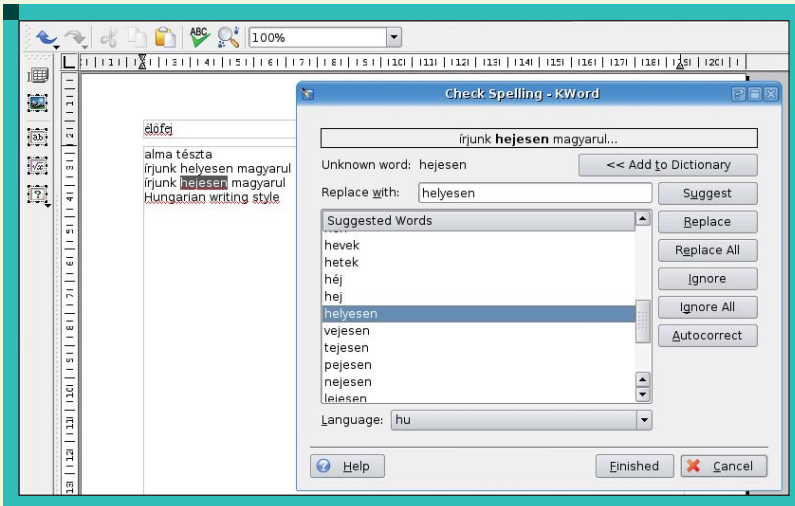
A *KWord* a *KDE*-t használó ill. *KDE*-s alkalmazásokat ismerők körében nem okoz meglepetést azzal, hogy a *Kparts* technológiát használva jó példáját hozza a *KDE* modularitásának: a *KOffice* egyes alkalmazásainak funkciói a *KWord*-ből ugyanúgy elérhetők és kezelhetők, mintha a megfelelő külső alkalmazásokat használnánk. Például ha a dokumentumunkba egy



4/a. ábra A KSpread táblázatkezelő funkcióinak használata, amint Spreadsheet objektumot szúrunk a dokumentumunkba



4/b. ábra Illesszünk grafikont a dokumentumba a KChart eszközeivel a KWord-ön belül



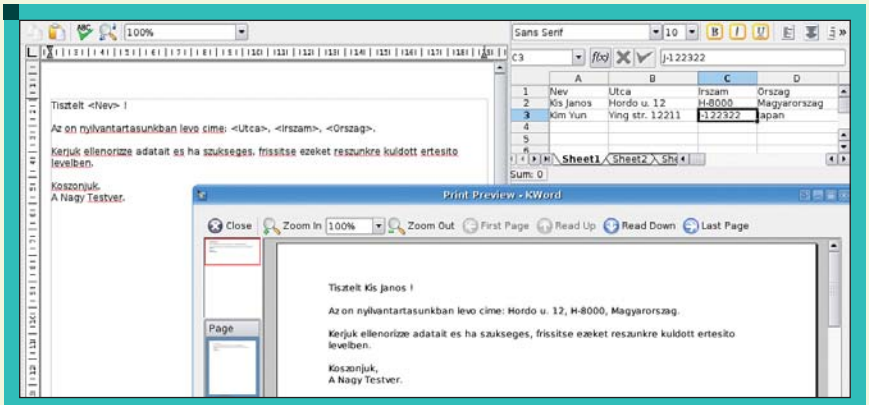
5. ábra Helyesírásellenőrzés KWord-ben

adatokat tartalmazó táblázatot szeretnénk illeszteni, akkor az ***Insert*** menüben elérhető ***Object Frame***-ek közül a ***Spreadsheet***-et választva a ***KOffice*** táblázatkezelőjének (***KSpread***) minden funkcionalitását elérhetjük a dokumentumunkban, ahogy ezt a *4/a. ábra* mutatja. De hasonlóképp, például a ***KChart***-tal grafikont illeszthetünk be anélkül hogy elhagynánk a ***KWord***-öt (*4/b. ábra*).

Fontos hogy egy szövegszerkesztő tudjon helyesírást ellenőrizni és javítani, és erre a ***KWord***-ben is van lehetőség. Helyesírás-ellenőrzéshez az ***ispell*** vagy ***aspell*** külső helyesírás-ellenőrzőket használhatjuk, mindkettőhöz létezik magyar szó-adatbázis (***ispell-hu*** ill. ***aspell-hu*** néven). ***KWord***-ön belül az ellenőrzési tulajdonságokat a ***Settings->Configure Kword->Spelling*** helyen állíthatjuk be majd a ***Tools->Spellcheck*** menüpontban vagy az eszköztáron érhetjük el (*5. ábra*).

Szövegszerkesztő alkalmazások irodai funkciói közül egy fontos elem a körlevelek és beszámolók készítése. Míg az utóbbiról később lesz szó (mivel azt egy külön ***KOffice*** alkalmazás, a ***Kugar*** végzi), a körlevelek készítéséről mindenképpen szót kell ejtenünk.

A ***KWord***-ben körlevelek készítése nagyon egyszerű feladat. Először el kell döntenünk, hogy milyen adatforrást használunk a körlevél adatainak tárolására/elérésére. A ***KWord*** lehetőséget ad külső és belső adatforrás használatára is. Külső adatforrás lehet egy helyi vagy távoli ***SQL*** szerver, egy ***KSpread*** táblázatkezelővel előállított adatfájl, vagy a ***KDE*** címjegyzéke (***KAddressBook*** alkalmazás adatai). Belső adatok esetén a bevitt adatok magában a készülő dokumentumban kerülnek tárolásra. Az adatforrás kiválasztása a ***Tools->Configure Mail Merge*** menüpontban lehetséges. Esetünkben a példa kedvéért egy ***KSpread***-ben készített táblázat tartalmazza a változó adatokat (kliensek neve és címe). A táblázat létrehozása után ***Configure Mail Merge->Open Existing->KSpread Table Source***-ot választva megadjuk a létrehozott táblázatfájl elérhetőségét, majd készíthetjük is a levelet. A táblák változó adatainak beszúrását szerkesztés közben az ***Insert->Variable->Mail Merge*** menüpontból végezhetjük. Az *6. ábra* egy példát mutat egy egyszerű



6. ábra Körlevél készítésének elemei: a készülő levél a változókkal, a táblázat a konkrét adatokkal és az első oldal nyomtatási előnézete

levélre, a táblázat adatival és az első oldal nyomtatási előnézetével. Egyes funkciók elrendezésének megszokása egyes felhasználóknak okozhat némi nehézséget, de gyorsan beletanulhatunk, mivel rövid idő alatt felfedezhető az alkalmazott csoportosítási rendszer. Például az oldalszám – és más hasonló, változó értékű elem – beillesztését az ***Insert->Variable->Page->Page Number*** helyen tehetjük. Ezúton hívom fel a kedves olvasó figyelmét az ***Insert*** menü ***Variable*** (változók) és ***Expression*** (kifejezések) pontjaira, ahol számos hasonló jellegű elemet találhatunk amelyek nagyban megkönnyíthetik dokumentumszerkesztői életünket.

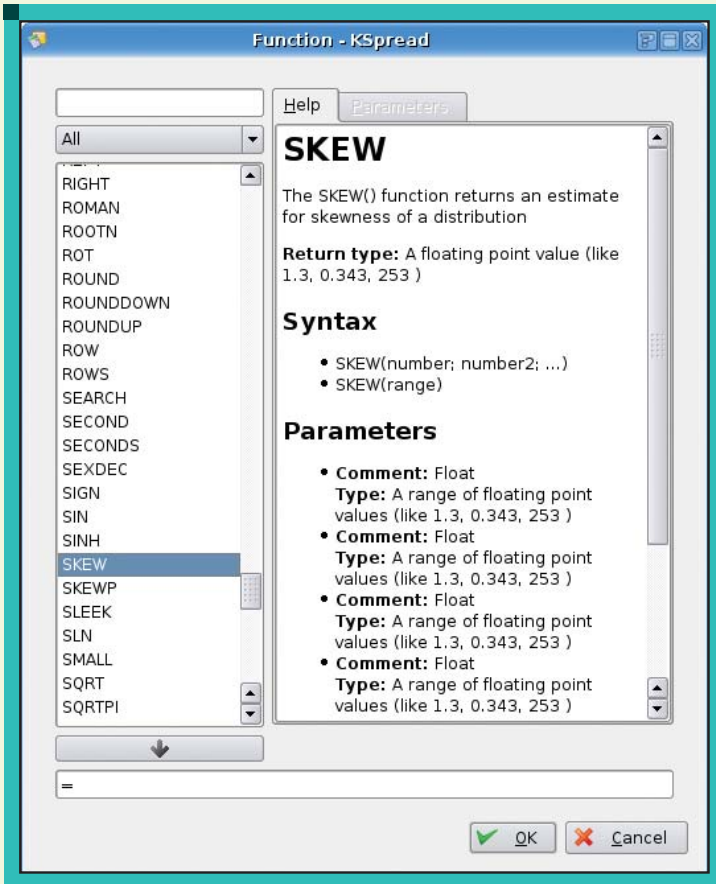
A *KWord* számos fájl-formátumot ismer és képes kezelni, a saját *XML*-alapú formátuma mellett. Így használhatunk *OASIS OpenDocument*, *OpenOffice*, *AmiPro*, *WordPerfect*, *AbiWord*, *PDF*, *TeX*, *RTF*, *HTML*, *SGML*, *WML* formátumokat. *Microsoft Word* dokumentumokat a *KWord* meg tud nyitni és olvasni, de a formátum zártságának köszönhetően a bonyolultabb *Word* dokumentumok konvertálásával előfordulhatnak gondok, ezért erre a célra lehetőleg ne használjuk.

Ebben a rövid áttekintésben az volt a cél, hogy néhány fontosabb funkciót bemutassunk, amelyek a *KWord*-öt leginkább jellemzik. A hétköznapi szövegszerkesztő-funkciók mellett ezek olyan pluszt adhatnak, amelyek miatt érdemes lehet a *KWord* használatának a megfontolása. Szintén fontos hogy a *KWord* – mint a *KOffice* csomag legtöbb eleme – nagyon gyors: szinte nem kell várni, minden azonnal történik. Hátrányok között meg kell említeni, hogy a *KWord* nem képes verziókövetésre és változtatások figyelésére, nincs benne a *Microsoft Word*-ben létező nyelvtani szabályok alapján történő ellenőrzés. Viszont mindenképpen nagy előny, hogy aki ismeri a *Word* viselkedését képek beszúrása után végzett nemkívánatos automatikus átformázás esetenként felmerülő furcsaságait, azt meg kell nyugtatnunk: a *KWord* esetén éppen a *frame*-ek használata miatt ilyen gondok nem merülnek fel.

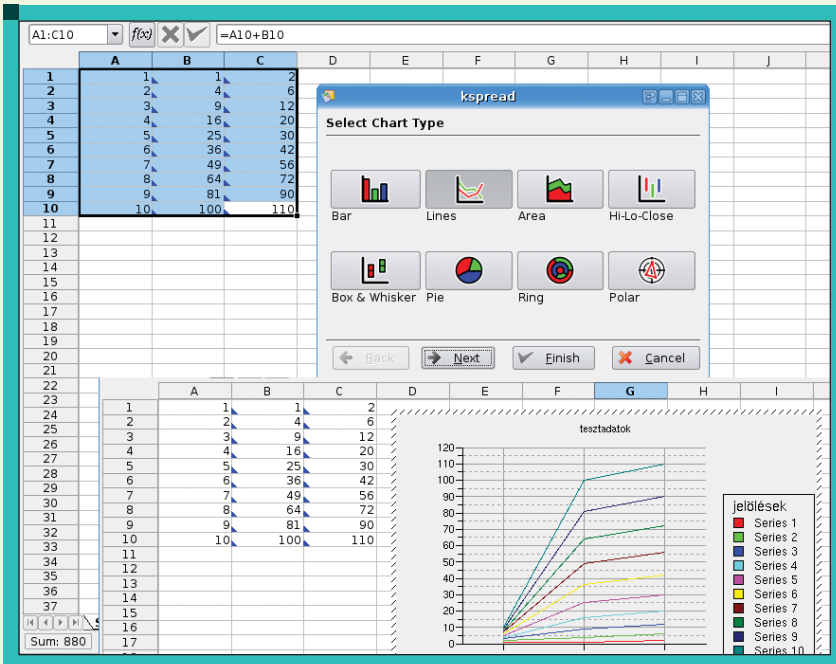
Táblázatkezelés – KSpread

A *KOffice* táblázatkezelő alkalmazása a *KSpread*. Noha a *KSpread* a legtöbb kezdő és középhaladó felhasználó számára elégséges funkcionalitást nyújt, haladó és profi felhasználók számos hiányosságot fognak felfedezni benne. Mindazonáltal a legtöbb hétköznapi táblázatkezelési alkalmazásra képes kielégítő megoldást adni. Így noha szimulációs eszközökben hiányt szenved, számtalan matematikai, statisztikai és gazdasági függvényt találunk a *KSpread*-ben, összesen több százat (*7. ábra*).

Függvények beszúrására az ***Insert->Function*** menüponttal vagy az eszköztár ***f(x)*** jelzésű gombjával van lehetőségünk. Minden függvényhez találunk leírást ami segíti azok használatát. Gondot okozhat azonban, hogy ha a cellákba rossz szintaxissal írjuk be a kifejezéseinket, akkor semmilyen segítséget vagy útmutatást nem kapunk a hiba okáról, csupán egy jelzést, hogy hibás a kifejezés.



7. ábra Számos beépített függvényt használhatunk

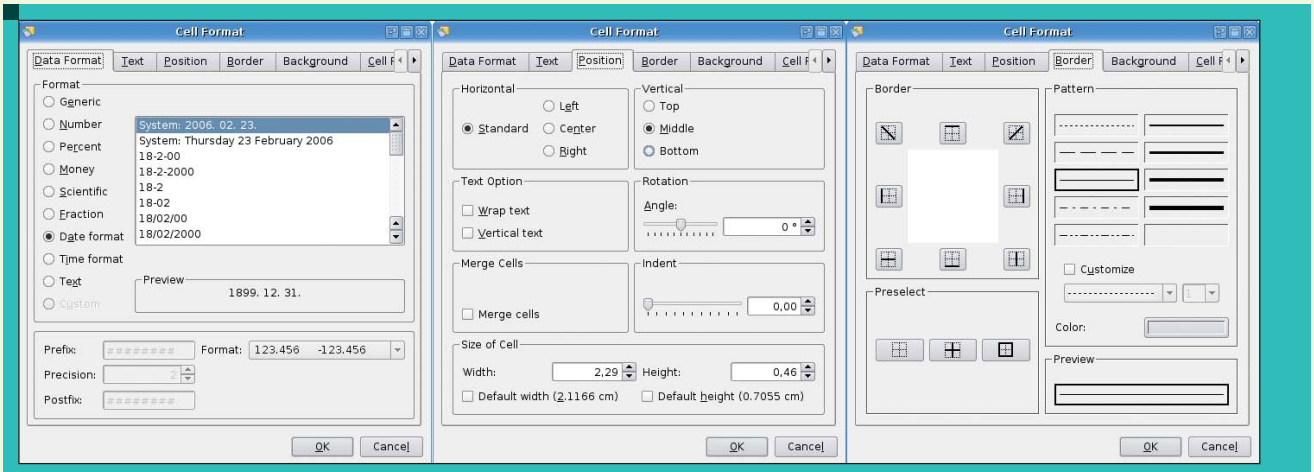


8. ábra Adatsorokból grafikon készítése: Insert->Chart menüpont. Grafikon stílusának kiválasztása fent és a létrejött grafikon lent.

Format, *9. ábra*), adatformátumot, hátteret, fontokat, színeket, kereteket állíthatunk be könnyen.
Mint azt a ***KWord*** esetén már említettük, a ***KOffice KParts*** alapokon nyugvó fejlesztésének köszönhetően mindegyik ***KOffice*** alkalmazás elérhető a többiből. Így a ***KSpread***-del készülő táblázatokba is beágyazhatók, például rajzok, képek, szövegek.
KSpread-del importálhatunk ***Microsoft Excel*** táblázatokat, de tudnunk kell, hogy a ***Visual Basic*** makrókat nem kezeli. Egyébként a ***KSpread*** számos formátumot támogat a saját *XML* alapú formátuma mellett: ***OASIS OpenDocument***, ***Excel***, ***OpenOffice***, ***GNUmeric***, ***PDF***, ***TeX***, ***HTML***.

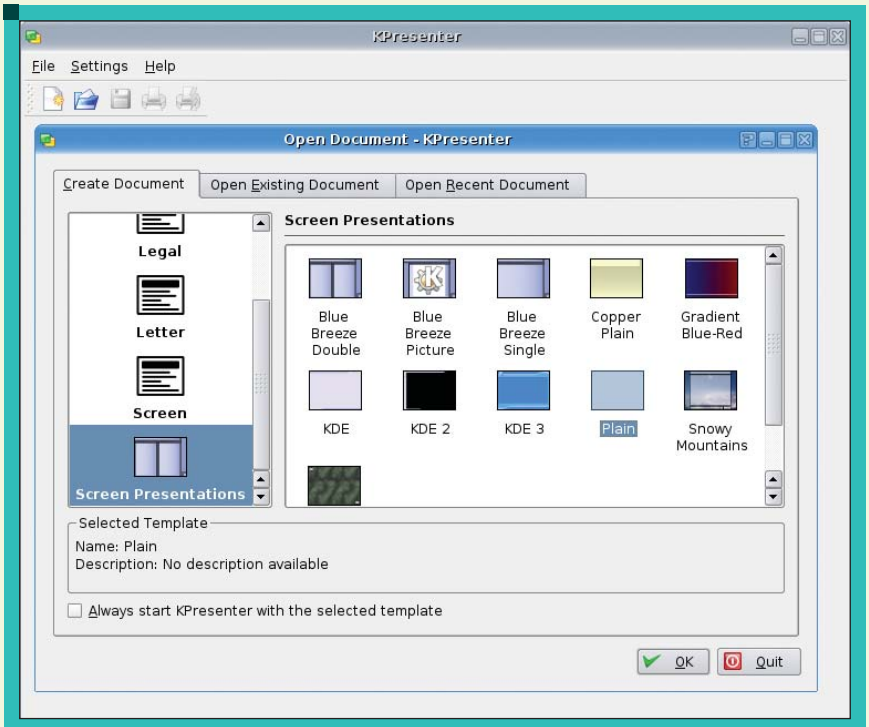
Bemutatók – KPresenter

A ***KPresenter*** egy olyan alkalmazás, amely tökéletesen illeszkedik a ***KOffice*** többi eleméhez: egyszerű, gyors, praktikus. Egyszerű bemutatók

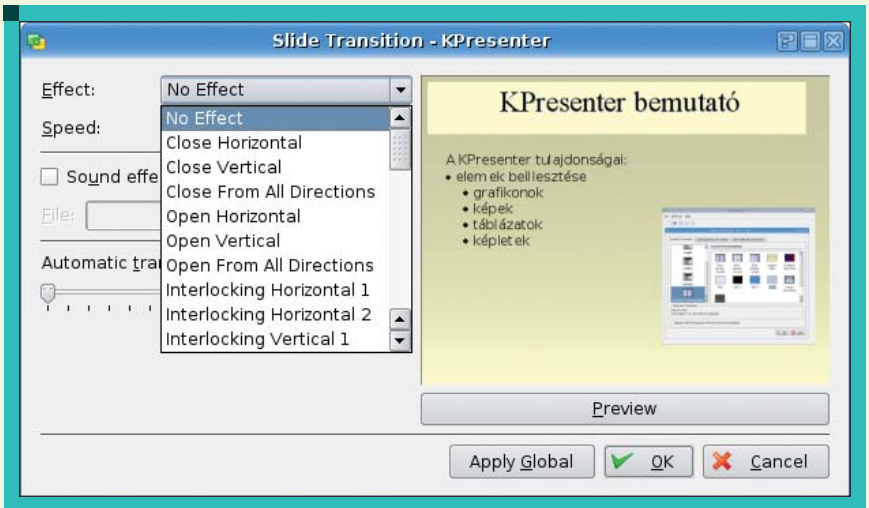


9. ábra Cellaformázási lehetőségek

Hétköznapi alkalmazások esetén – ami leginkább adatok bemutatható és kezelhető formába szervezését jelenti ill. ezekből grafikonok készítését – a ***KSpread*** egy egyszerűen használható sokoldalú eszköznek bizonyul.
Fontos szempont lehet az is, hogy a billentyűzet használata a más hasonló alkalmazásokhoz szokott felhasználók számára sem okoz nehézséget, mivel a máshol is jól megszokott billentyűparancsok többsége itt is ugyanúgy használható.
Egyik fontos eszköz grafikonok készítése a rendelkezésre álló adatokból. Erre az adat-tartomány kijelölése után az ***Insert->Chart*** menüponttal van lehetőségünk (*8. ábra*), amely a ***KOffice*** csomag ***KChart*** nevű grafikon-készítő kiegészítő-alkalmazásának elérését teszi lehetővé a táblázatkezelőn belül (ugyanúgy mint ***KWord***-ön belül). Kiválaszthatjuk a grafikon típusát, beállíthatjuk paramétereit (fontok, színek, elrendezés). Sajnos a készíthető grafikonok típusai eléggé korlátozottak, de így is elegendőek a legtöbb ábrázolási feladathoz és a készített grafikonok látványosak.
Az adatok táblázat-szintű megjelenítésének beállításaihoz számos cellaformázási lehetőségünk van, a ***Format*** menü elemeivel (elsősorban a ***Cell*** könnyedén és gyorsan összerakhatók a ***KPresenterrel***. Új bemutató létrehozásakor választhatunk a beépített vagy előzőleg készített minták közül (*10. ábra*) vagy készíthetünk új bemutatót is.
A bemutatók ***slide***-jaira az egyes tartalmi elemek könnyen elhelyezhetők, az ***Insert*** menü számos pontjának segítségével. A ***KPresenter***-ben, ugyanúgy, mint mindegyik ***KOffice*** alkalmazásban, könnyen beilleszthetők más ***KOffice*** eszközzel készített objektumok, de akár itt helyben a ***slide***-on is szerkeszthetők és készíthetők rajzok, grafikonok, táblázatok, szöveges elemek (*11. ábra*).

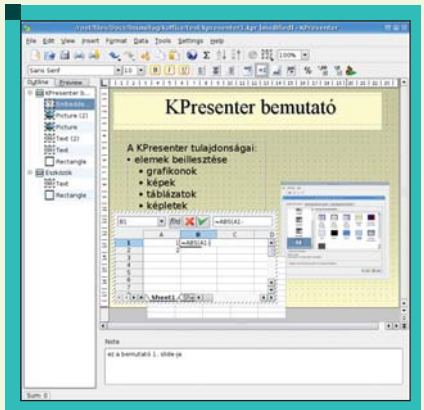


10. ábra KPresenter új bemutató: alapértelmezett minták



12. ábra Slide-ok közötti átmenetek beállítása, Slide Show->Edit Slide Transition menüpont

A bemutató *slide*-jai közötti átmenetek beállítására is számos lehetőséget biztosít a *KPresenter*, ezek a *Slide Show* menüben érhetők el (*12. ábra*).
Az *Insert->*Line és *Insert->*Shape menüpontokban különféle vonalas rajzokat és geometriai alakzatokat rajzolhatunk egyszerűen a bemutatóba. Mindezek és a fenti tulajdonságok mellett nincs viszont lehetőségünk hangok és videók beillesztésére. Többek között *KPresenter*, *OASIS OpenDocument*, *OpenOffice* prezentációs formátumokba, *HTML*-be, *PDF*-be és számos más formátumba menthetjük el bemutatóinkat. Megnyithatunk *Microsoft PowerPoint* bemutatókat, viszont ha ez bonyolult, animációkat, videókat tartalmaz, akkor lehetnek gondok a konvertálással. Mindent figyelembe véve, a *KPresenter* kiváló eszköz animációk és videók nélkül készítendő bemutatókhoz, különösen akkor, ha a felhasznált adatok is *KOffice* alkalmazásokkal kerültek előkészítésre. Ha videók vagy animációk beillesztésére van szükségünk akkor az *OpenOffice Impress* vagy más bemutató-készítő megfelelőbb választás lehet.



11. ábra Készülő bemutató egy slide-ja

Összefoglaló

A *KOffice*-t bemutató sorozat első részében rövid betekintést próbáltam adni a *KWord*, *KSpread* és *KPresenter* eszközökkel történő munka lehetőségeibe. Noha egyes esetekben haladó és profi felhasználók nagyobb tudású irodai alkalmazásokat fognak előnyben részesíteni (például *OpenOffice*, *StarOffice*, esetleg *Microsoft Office*), a legtöbb esetben a *KOffice* eszközei a hétköznapokban jól és könnyen használható alternatívákat kínálhatnak. A következő részben a kiegészítő alkalmazásoknak szentelünk figyelmet: a *Kexi* adatbázis-kezelőnek, a *Kformula* egyenletszerkesztőnek, a *Kivio* folyamatábra készítőnek, a *Karbon* és *Krita* grafikus alkalmazásoknak és a *Kugar* beszámoló-készítőnek.

Kovács Levente
(leventek@gmail.com)

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

KAPCSOLÓDÓ CÍMEK

- www.koffice.org
- www.koffice.org/kword
- www.koffice.org/kspread
- www.koffice.org/kpresenter

Régi idők UNIX-a

A SIMH rendszer segítségével megtapasztalhatjuk, hogyan működött a UNIX a digitális (h)őskorban.

Sokak szerint sokkal hajlamosabb vagyok a nosztalgiára, mint ami az életkoromból következne. Azt hiszem igazuk lehet. Bár 1976-ban születtem, rajongásig szeretem a korabeli nagygépek és miniszámítógépek történelmét tanulmányozni. Ami azt illeti, azt hiszem az emberiség egyik legnagyobb találmánya maga a történetírás. Úgy tűnik az emberi gondolkodásba bele van drótozva az a törekvés, hogy szeretjük följegyezni és elemezni elődeink cselekedeteit, és szeretünk tanulni a hibáikból és a sikereikből egyaránt. A nyílt forrás nem egyéb, mint ennek az elvnek az érvényesülése a számítástechnikában, hiszen ez a megközelítés lehetővé teszi, hogy tanuljunk mindazoknak a programozóknak a forráskódjaiból, akik előttünk éltek és alkottak egy-egy területen. Ha elemezzük egy korábbi fejlesztési projekt fejlődését, sokat tanulhatunk annak hibáiból, de ami még ennél is fontosabb, jó esélyünk van rá, hogy egyszerűen lemásolhatjuk belőle a sikeres részeket. Azzal pedig, hogy összeszedjük ezt a szabadon hozzáférhető történelmet, és profitálunk belőle, magunk részeseivé válunk a közjó továbbfejlesztésének.
Számos nagyvállalat mostanában elkezdte nyilvánosságra hozni a tulajdonában levő „intellektuális javakat". Bár egyesek nem teljesen tették magukévá a nyílt forrás filozófiáját, ezekből a gesztusértékű cselekedetekből mindenképpen sokat tanulhatunk. Ebben a cikkben azt szeretném bemutatni, miként tanulmányozhatjuk az operációs rendszerek történelmét azáltal, hogy saját ***Linux*** rendszerünkön ma már „történelminek" számító ***UNIX*** változatokat emulálunk. Jelenleg a ***SCO*** csoport (bizony ők, korábban pedig a ***Caldera Inc.***) tudja magáénak a korai UNIX változatok szerzői jogait, ám nem kereskedelmi célra szabaddá tette azokat egy úgynevezett ***„Ősi UNIX licenszmegállapodás" (Ancient UNIX License)*** keretében. Itt és most én a ***UNIX V5*** kiadására fogok koncentrálni, mivel a jelenleg hozzáférhetők közül ez a legkorábbi. A ***UNIX V6***, ***V7***, valamint a ***BSD*** különböző korai kiadásai szintén rendelkezésünkre állnak. Mindazonáltal aki azt tervezi, hogy kipróbálja ezek közül valamelyiket, az a biztonság kedvéért azért olvassa el a licenszmegállapodást, mielőtt először bebootolná a rendszert.

Idegenként egy idegen helyen: a UNIX V5 felhasználói környezet

A lemezlenyomat (***disk image***) formájában rendelkezésünkre álló ***UNIX V5*** meglehetősen ridegnek és barátságtalannak tűnik a mai kényelmes ***UNIX/Linux*** rendszerekhez képest. Éppen ezért nem árt pár dologgal tisztában lenni, mielőtt belekezdenénk az ismerkedésbe:

- A héjat itt még ***sh***-nak hívják. Összesen 858 sornyi C kód az egész, tehát ne várjunk tőle semmi olyasmit, amit a ***Bash***-től megszoktunk!
- Ha könyvtárat akarunk váltani, használjuk a ***chdir*** parancsot.
- A ***Backsapce*** és a nyilak a legritkább esetben működnek normálisan.
- A szövegszerkesztőt ***ed***-nek hívják. Aki még nem hallott róla, az látogasson el a ➲ ***en.wikipedia.org/wiki/Ed*** címre.
- Rendelkezésünkre áll egy ***Basic*** interpreter, amit ***bas***-nak hívnak.
- Hasonlóan van egy ***Fortran*** értelmező ***fc*** néven.
- A ***C*** fordító neve ***cc***.
- A ***/usr/source*** könyvtárban megtaláljuk a teljes rendszer forráskódját.
- Ami azt illeti nincs túl sok fájl az egész rendszerben, így a lista tanulmányozásához nyugodtan használhatjuk a `find / -print` parancsot.

Ahhoz, hogy tanulmányozhassuk az ilyen és ehhez hasonló operációs rendszereket, nyilván képesnek kell lennünk őket valamilyen módon futtatni a jelenleg használatos gépeken. Szerencsére vannak erre a célra megfelelő szimulátorok. Minősége és támogatottsága miatt ezek közül at egyik legnépszerűbb a ***SIMH***, amit a ➲ ***http://simh.trailing-edge.com*** webhelyről tölthetünk le.
A ***SIMH*** gyakorlatilag valamennyi manapság népszerű *nix rendszeren, sőt még ***Microsoft Windows*** alatt is futtatható. Ami képességeit illeti, számos korai géptípust tud szimulálni, beleértve a ***Digital Equipment Corp. PDP*** és ***VAX*** rendszereit, az ***MITS Altair*** nevű gépét, a korai ***IBM*** rendszereket, és még számos más öreg masinát. Ezek közül a történetileg legérdekesebbek talán a ***DEC PDP*** gépei, hiszen a ***UNIX*** ezeken kezdte meg pályafutását.
A ***SIMH*** az alapoktól kezdve építkezik, vagyis képes szimulálni a korabeli gépek processzorát, memóriáját, beégetett programját (***firmware***) és minden csatlakoztatott

1. kód

```
$ mkdir simh
$ cd simh
$ unzip /path/to/simhv34-0.zip
$ mkdir BIN  # Note all CAPS
$ gmake USE_NETWORK=1 all
# Only include USE_NETWORK=1 if your PCAP lib is up to date.
(compilation chatter omitted)
$ ls -l ./BIN/
total 11624
-rwxrwxr-x  1 matt matt  301959 Jul 16 18:45 altair
-rwxrwxr-x  1 matt matt  482274 Jul 16 18:45 altairz80
-rwxrwxr-x  1 matt matt  529317 Jul 16 18:44 eclipse
-rwxrwxr-x  1 matt matt  297590 Jul 16 18:45 gri
-rwxrwxr-x  1 matt matt  375737 Jul 16 18:44 h316
-rwxrwxr-x  1 matt matt  577678 Jul 16 18:44 hp2100
-rwxrwxr-x  1 matt matt  355225 Jul 16 18:44 i1401
-rwxrwxr-x  1 matt matt  381672 Jul 16 18:45 i1620
-rwxrwxr-x  1 matt matt  441079 Jul 16 18:46 ibm1130
-rwxrwxr-x  1 matt matt  502037 Jul 16 18:46 id16
-rwxrwxr-x  1 matt matt  508378 Jul 16 18:46 id32
-rwxrwxr-x  1 matt matt  294614 Jul 16 18:46 lgp
-rwxrwxr-x  1 matt matt  434940 Jul 16 18:44 nova
-rwxrwxr-x  1 matt matt  345034 Jul 16 18:41 pdp1
-rwxrwxr-x  1 matt matt  752055 Jul 16 18:43 pdp10
-rwxrwxr-x  1 matt matt 1055376 Jul 16 18:43 pdp11
-rwxrwxr-x  1 matt matt  474153 Jul 16 18:42 pdp15
-rwxrwxr-x  1 matt matt  459203 Jul 16 18:41 pdp4
-rwxrwxr-x  1 matt matt  460363 Jul 16 18:41 pdp7
-rwxrwxr-x  1 matt matt  499473 Jul 16 18:42 pdp8
-rwxrwxr-x  1 matt matt  467662 Jul 16 18:42 pdp9
-rwxrwxr-x  1 matt matt  352233 Jul 16 18:45 s3
-rwxrwxr-x  1 matt matt  429312 Jul 16 18:46 sds
-rwxrwxr-x  1 matt matt  982694 Jul 16 18:43 vax
```

eszközét. A ***SIMH*** tökéletesen utánozza a lemezegységeket, szalagos meghajtókat, sornyomtatókat és a hálózati eszközöket is. Ez utóbbi pedig ezt jelenti, hogy nem csak futtatni tudjuk ezeket az ősi operációs rendszereket, hanem adatokat is cserélhetünk segítségükkel a modern protokollokra támaszkodva. Mindezért a legnagyobb köszönet természetesen azokat illeti, akik hozzájárultak a ***SIMH*** fejlesztéséhez. Az ő munkájuk, no meg az az elhatározásuk, hogy annak gyümölcsét nyílttá teszik egyrészt hozzájárul ahhoz, hogy megérthessük saját digitális történelmünket, másrészt biztosítja, hogy ez a történelem mindörökre hozzáférhető maradjon valamennyiünk számára.

A SIMH telepítése

Kezdésként töltsük le a ***SIMH*** legfrissebb változatát (ez jelenleg a ***3.4-0***-ás verzió), fordítsuk le és telepítsük. Aki az ***Ethernet*** emulációt is használni szeretné, annak előbb valószínűleg frissítenie kell majd a saját operációs rendszeréhez kapott ***libpcap*** könyvtárat, mivel ez a legtöbb jelenlegi terjesztésben meglehetősen elavult. Ennek a műveletnek a leírását részletesen tartalmazza a ***SIMH*** dokumentációja, ha azonban nincs szükségünk a kérdéses szolgáltatásra, akár ki is hagyhatjuk ezt a lépés. A fordítást egyszerű felhasználóként is elvégezhetjük a következő parancsok segítségével *(1. kód)*.
Ezzel az összes jelenleg támogatott rendszer szimulátorát lefordítottuk. Mint látható, mindegyik egy-egy külön bináris állományt jelent a ***./BIN*** könyvtárban. Magát ha ***SIMH*** rendszert bármely felhasználó futtathatja, ha azonban ***Ethernet*** emulációt használunk, akkor muszáj rendszergazdaként futtatni, máskülönben a ***libpcap*** nem fér hozzá a valódi ***Ethernet*** eszközöz.

A UNIX V5 futtatása

Az 1974 júniusában kiadott ***UNIX V5*** még meglehetősen korai műve volt a ***Bell Labs***-nak. Ezt jelzi az is, hogy a rendszer java részét még assemblyben írták. Ez a lemezkép ugyanakkor már tartalmaz egy ***C*** fordítót (*cc*) a ***/usr/source*** könyvtárban pedig rengeteg érdekes forráskódot találhatunk. A felfedezés megkezdéséhez előbb le kell töltenünk a ***UNIX V5*** lemezképét (lásd a forrásokat). Ez egy zip tömörítvény, ami tartalmazza az előtelepített rendszert és egy ***README*** fájlt, amiben a szokásos licenc információt találjuk. A lemezkép tulajdonképpen egy működő rendszerről készült pillanatkép. Kicsit konkrétabban rendszerünk egy ***RK05*** lemezegységet fog szimulálni. Először s össze kell gyűjtenünk azokat az elemeket, amelyek a rendszer bebootolásához szükségesek. Hozzunk létre egy új könyvtárat, majd másoljuk bele a ***BIN/pdp11*** bináris fájlt, valamint az ***uv5swre.zip*** fájl tartalmát. Aztán egy tetszőleges szövegszerkesztővel hozzuk létre a ***pdp11.ini*** fájlt, ami a szimulátor működését fogja majd vezérelni. Az ***ini*** fájlnak a következőket kell tartalmaznia:

```
set cpu U18
attach rk0 unix_v5_rk.dsk
boot rk0
```

Ez a beállítófájl mondja meg a szimulátornak, hogy milyen típusú processzort emuláljon, illetve hogy esetünkben csatlakoztassa a ***unix_v5_rk.dsk*** fájlt egy szimulált ***RK***-rendszerű lemezegységként, az ehhez tartozó eszköz neve pedig legyen rk0. Az utolsó sor szerint magát az operációs rendszert is erről a lemezről kell bebootolni. A kérdéses könyvtárnak tehát a végén így kell kinéznie *(2. kód)*.

2. kód

```
-rw-rw-r--  1 matt matt   12299 Jan 24  2002 AncientUnix.pdf
-rwxrwxr-x  1 matt matt  913614 Jul 22 19:33 pdp11
-rw-rw-r--  1 matt matt      47 Jul 22 23:59 pdp11.ini
-rw-rw-r--  1 matt matt     263 Nov 25  1996 README.txt
-rw-rw-r--  1 matt matt 2494464 Jul 23 00:39 unix_v5_rk.dsk
```

3. kód

```
$ ./pdp11
PDP-11 simulator V3.4-0
Disabling XQ
@unix
login: root
# ls -l /
total 60
drwxr-xr-x  2 bin       944 Nov 26 18:13 bin
drwxr-xr-x  2 bin        80 Nov 26 18:13 dev
drwxr-xr-x  2 bin       240 Mar 21 12:07 etc
drwxr-xr-x  2 bin       224 Nov 26 18:13 lib
drwxr-xr-x  2 bin        32 Nov 26 18:13 mnt
drwxrwxrwx  2 bin       112 Mar 21 12:11 tmp
-rwxrwxrwx  1 bin     25802 Mar 21 12:07 unix
drwxr-xr-x 14 bin       224 Nov 26 18:13 usr
# chdir /usr/source/s1
# cat echo.c
main(argc, argv)
int argc;
char *argv[];
{
        int i;
        argc--;
        for(i=1; i<=argc; i++)
                printf("%s%c", argv[i], i==argc? '\n': ' ');
}
# cc echo.c
# mv a.out newecho
# ./newecho Hello World
Hello World
# chdir /tmp
# cat >hello.c
main()
{
   printf ("Hello World!\n");
}
# cc hello.c
# ./a.out
Hello World!
# cat >hello.b
10 print "Hello World!"
# bas hello.b
run
Hello World!
```

A ***UNIX V5*** rendszer bebootolásához most már csak a ./pdp11 parancsot kell kiadnunk. Aztán amikor megjelenik a `@` prompt, gépeljük be a `unix` parancsot. Erre gyakorlatilag azonnal megkapjuk a bejelentkezési promptot, a korabeli ***UNIX*** ugyanis bootolás közben egyáltalán nem voltak olyan beszédesek, mint a mostaniak. Szimulált rendszerünkön root jelszó nincs, így azonnal egy parancspromptot kapunk. Az egész folyamat a következőképpen néz ki a képernyőn *(3. kód)*. Ennyi kérem! A rendszerünk máris működőképes így elkezdhetünk tanulmányozni egy „igazi” történelmi ***UNIX*** operációs rendszert. Amint látható rengeteg érdekes forráskódot találunk, és van hozzá egy ***C*** fordítónk is, hogy játszhassunk vele. A ***UNIX V5*** csupán az egyik azok közül a régi rendszerek közül, amelyeket a ***SIMH*** szimulátoron keresztül megismerhetünk. A ***SIMH*** hivatalos webhelyén lemezképek egész gyűjteményét találjuk, amelyek mind egy-egy ilyen régi rendszert tartalmaznak.
Aki pedig látni szeretné, hogyan is nézett ki a valóságban egy ***PDP-11*** vagy egy ***RK05*** lemezegység, vessen egy pillantást a fotógalériára. A ***Google*** képkeresőjével szintén számos e témával kapcsolatos érdekes fotót találhatunk.

Linux Journal 2005. 140. szám

Matthew Hoskins vezető UNIX rendszergazda a New Yersey-ben található Institute of Technology-ban, ahol ő felelős a legtöbb adminisztratív rendszerért. Élvezi, amikor egészen vadul eltérő rendszereket és programokat hangolhat össze, és bírhat együttműködésre. Ehhez rendszerint egy Perl-ben megírt vékony összekötő réteget használ, amit a helyiket MattGlue (Matt ragasztója) néven emlegetnek. Amikor nem a különböző rendszereket hackeli, általában a konyhában foglalatoskodik hasonlóan érdekfeszítő dolgokkal. Matt a matt@njit.edu címen érhető el.

KAPCSOLÓDÓ CÍMEK

- www.linuxjournal.com/article/8587

Vissza a parancssorba – X-terminál emulátorok

A terminál a Linux rendszerek szerves és mondhatni kihagyhatatlan része. Nincsen olyan feladat, amit ne lehetne terminálról elvégezni. Sőt továbbmegyek: még grafikus felület sem szükséges ahhoz, hogy teljes értékű rendszerünk legyen amivel zenét hallgatunk, filmet nézünk vagy éppen CD-t írunk.

■ Egy ***Linux*** terminállal nincs lehetetlen. Manapság egy felhasználóbarát disztribúcióban minden beállítás elvégezhető grafikus felületről, különböző programok segítségével, így nem szükséges a konfigurációs fájlokat kézzel szerkesztgetni. Jelen cikknek nem célja meggyőzni egyetlen felhasználót sem, hogy használjon bizonyos feladatokra terminált. Pusztán egy egyszerű áttekintést szeretnék adni annak, aki használja, mert mondjuk észrevette, hogy egyszerűbb 20 képet ***wget*** segítségével egy két soros szkriptet írva letölteni, mint egyesével kattintgatni, illetve aki csak barátkozik a használatával.
Nos merem állítani, hogy ezen a téren is van bő program-választék. Egy alap rendszeren is elmondható, hogy van hét terminálablakunk, ezeket az ***Alt+Ctrl+F1..F7*** billentyűk lenyomásával érhetjük el. A grafikus felület általában a hetes terminálon fut (mivel maga az ***X*** is csak egy program). Ha grafikus felületen szeretnénk parancsot osztani, akkor szükség van egy terminál emulátorra, ezekről szeretnék egy kis összefoglalót adni, de ne rohanjunk előre. Egy terminál legfontosabb része a parancsértelmező (shell, héj). Ebből is sokféle létezik (***ash***, ***bash***, ***dash***, ***ksh***, ***tcsh***, ***zsh***), de mi most kizárólag a ***Bash***-sal foglalkozunk, mert a legtöbb disztribúció ezt használja alapértelmezésben valamint ez a legnagyobb tudású.

A Bash testreszabása

Talán furcsán hangzik, de a parancssoron is számos dolog testreszabható. A prompt megjelenési módjától kezdve a színeken keresztül a saját parancsokig (alias) minden. Ezeket a beállításokat több fájl is tartalmazza. Az általános beállítások a */etc/bash.bashrc*, a személyes beállítások pedig a *~/.bashrc* és *~/.bash_profile* fájlban találhatók. Az utóbbi kettő között annyi a különbség, hogy a .bash_profile a bejelentkezésre (login felületre) vonatkozó beállításokat tárolja, a .bashrc meg az általános beállításokat. Egy virtuális terminálon, amit ***X*** felületen futtatunk rögtön bejelentkezett promptot kapunk így ott nem érvényesülnek a ***.bash_profile*** beállításai. Van a saját könyvtárunkban még egy harmadik ***Bash***-sal kapcsolatos állomány, a ***.bash_history.*** Ez tárolja a kiadott parancsokat, mely segítségével vissza tudunk keresni a fel nyíllal, vagy a ***Ctrl+R*** lenyomásával (rekurzív keresés) és parancstöredék begépelésével. Visszatérve a két konfigurációs fájlra, ezek alapesetben néhány megjegyzésként inaktivált beállításra mutat példát. Két példát szeretnék konkrétan bemutatni: a prompt testreszabása, illetve az egyéni parancsok azaz álnevek (aliasok) beállítása.
A promptot egyszerűen egyénivé varázsolhatjuk néhány beállítóparancs segítségével. Színezhetjük, kiírathatjuk az elérési útvonalat, a felhasználó és gépnevet... A ***.bashrc*** fájlba helyezzük el a

```
PS1=""
```

sort és az idéző jelek közé illesszük be a nekünk megfelelő dolgot. Bővebben erről a ***Bash*** súgóoldalain olvashatunk (`man bash`), lássunk azonban néhány példát!
Ha ***DOS***-ra emlékeztető promptot szeretnénk (elérési út és >), akkor a

```
PS1="\w\[\e[37;1m\]> \[\e[0m\]"
```

karaktersorozatot kell használni.
Az útvonalat a `\w` kapcsoló kéri be.
Lehet kísérletezni a színekkel is. Elegáns megoldáspéldául zölddel kiírni a felhasználónevet és a gépnevet (hostname), majd jöhet egy kék színű `$`. Íme ennek a leírása:

```
PS1="\[\e[32;1m\]\u@\h:\w\[\e
➥[34;1m\]$ \[\e[0m\]"
```

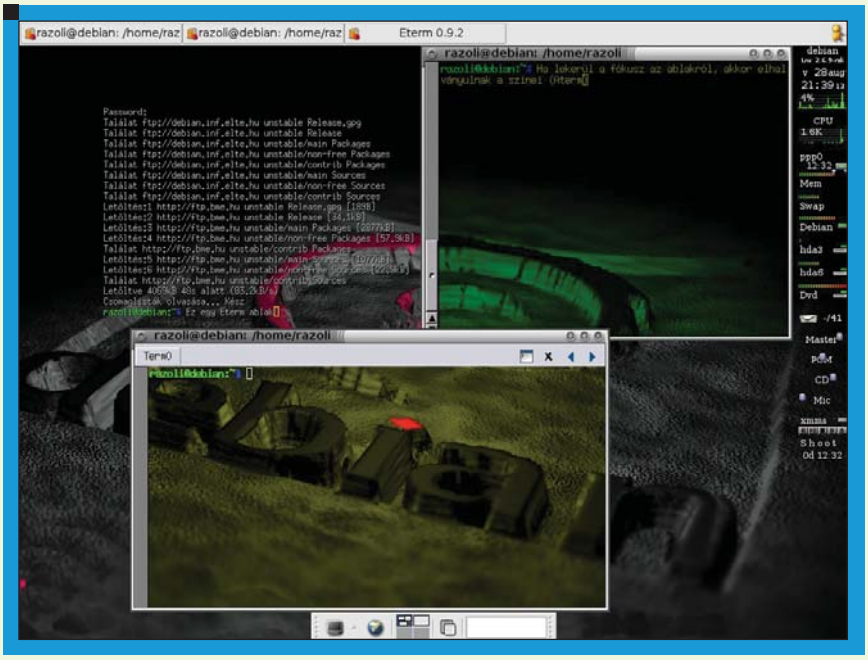
Ha a root promptjáról van szó (amelynek a leírását természetesen a */root/.bashrc*-be kell felvenni) akkor könyvtár, útvonal és piros # álljon a sor végén:

```
PS1="\[\e[32;1m\]\w\[\e[31;1m\]
➥# \[\e[0m\]"
```

Aliasok megadása is könnyebbé teszi a munkát. Itt tulajdonképpen tetszőleges parancsokat lehet meghatározni tetszőleges feladatokra. Például, ha minden esetben amikor kiadjuk az `ls` parancsot szeretnénk a rejtett fájlokat is látni valamint részletes listát kapni, akkor vegyük fel az

```
alias ls='ls -l -a'
```

sort. Így amikor az `ls`-t futtatjuk az alias lesz meghívva vagyis valójában



az `ls -l -a`. Ezenkívül ajánlott felvenni az

```
alias rm='rm -i'
```

sort, melynek hatására az `rm` (törlés) parancs kiadása után mindig rákérdez még egyszer a biztonság kedvéért. Látható lesz, hogy azoknál a terminál emulátoroknál, ahol a beállításokat parancssorból kell megadni ott érdemes aliast vagy indító ikont használni. Érdemes megemlíteni még egy meglehetősen hasznos eszközt, nevezetesen a parancskiegészítést (bash completion). Amennyiben nem aktív, akkor a */etc/bash.bashrc* fájlban távolítsuk el a kommentezést (#) a `bash completion` sorok elől. Nagyon hasznos eszköz főként hosszú fájlnevek esetén. A tab billentyűvel vezérelhető, a lehetőségekhez képest kiegészíti az adott parancsot vagy fájlnevet, ameddig tudja. Két lehetőség van: vagy kiegészíti teljesen, ekkor készen is van, vagy kiegészíti addig ameddig nincs eltérés a lehetőségek közül és kiírja a további lehetséges szavakat. Például van egy könyvtárunk, melyben van két állomány, *munkák.txt* és *munkahelyi_adat.sxw*, akkor a *TAB* lenyomására a `munk` szó jelenik meg és kiírja a további lehetőségeket, jelen esetben a fenti két fájlt; 'á'+tab hatására már a *munkák.txt* jelenik meg.
És most lássuk azokat a terminálprogramokat, amelyeken keresztül mindezt a grafikus felületen keresztül is használhatjuk.

Konsole

A *Konsole* a *KDE (K Desktop Enviorment)* beépített terminálja, ebből eredően meglehetősen funkciókban gazdag, főleg kezdő felhasználók számára, mivel majdnem minden beállítást menüből el lehet végezni. *KDE*-s voltából következően *kdelib** függvénytár függőségei vannak, így ha esetleg nem használunk *KDE*-t akkor érdemes átgondolni, hogy fel akarunk-e áldozni 40 Mb területet a függőségek miatt. Információt a ➲ *http://konsole.kde.org* oldalon találunk. Jelenlegi legfrissebb kiadott változata az 1.5.2 verziószámot viseli.

Gnome-terminal-emulator

A nevéből is látható, hogy a *Gnome* asztali környezet beépített terminál emulátora. *Konsole*-hoz hasonlóan rengeteg beépített funkcióval rendelkezik, érdemes átnézni a menüket. Szintén az érvényes rá mint a *Konsole*-ra, ha nincs *Gnome* a gépen és akkor hiányolni fogja, maga a program a *Gtk+* függvénytárra épül (*2.6.x*). Legfrissebb kiadott verziója a *2.9.2*-es.

Xterminal

Az *Xterminal* az *XFce* asztali környezet programja, ami nem azonos az *Xterm* nevű alkalmazással. Szintén menüből kényelmesen beállíthatóak a megjelenési stílusok (színek, átlátszóság) és támogatja a több ablakos terminál futtatást.
Jelenleg *0.2.4*-es verziónál tart a fejlesztés. A ➲ *http://terminal.oscillation.com/* oldalról tölthető le csomag és forrás formájában.

Xterm

Az *xterm* az *X* alapértelmezett terminál emulátora. A konfigurációs fájl a */etc/X11/app-defaults/XTerm* útvonalon érhető el. Meglehetősen sok funkció érhető el benne, például támogatja az *UTF-8* módban való indítást (`-u8`). Érdemes átolvasni a súgóját, részletes felvilágosítást kapunk minden egyes funkcióról.

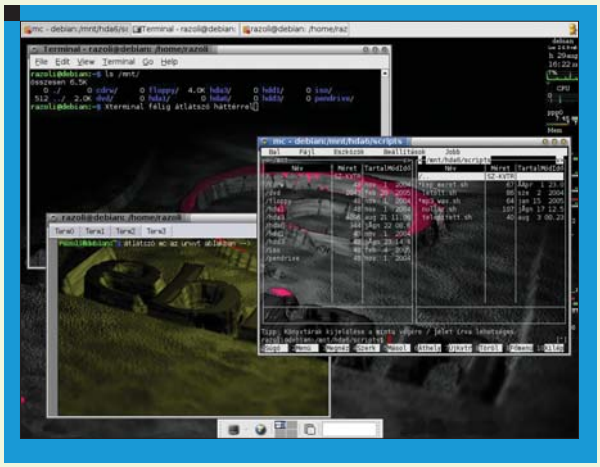
Atrem/Multi-Aterm

Az *Aterm* az *AfterStep* ablakkezelő terminál emulátora. Jelenleg két változata is létezik: az *aterm* és a *multi-aterm*. A különbség annyi, hogy a *multi-aterm* képes több terminálablakot futtatni egyszerre egy ablakban, mint a nagyobb testvérei. Legfrissebb kiadása az 1.0.0, honlap a ➲ *http://www.afterstep.org/aterm.php* címen érhető el.
Néhány példa az indításra:

```
aterm -tr -tint blue -fg yellow
➥-bg black -fade 50
```

Ez átlátszó, hátteret kékre festő terminálablakot eredményez sárga betűkkel, melyről ha levesszük az ablak fókuszt, akkor 50%-kal sötétebb lesz.

```
multi-aterm -tr -tint yellow
➥-fg white -bg black
```



Hasonló az előbbihez, de több ablakos sárgás hátterű ablakot nyit. Amennyiben mindig ugyanott szeretnénk ha megnyílna akkor toldjuk meg egy -g kapcsolóval és megfelelő paraméterekkel. Jelen esetben a `-g 100x30+20+40` eredményeképpen 100*30 karakteres ablak nyílik a bal felső saroktól számított 20;40 képpontoktól kezdve.

Eterm

Az egyik leglátványosabb funkciója az, hogy keret nélkül indítható. Ezt egy átlátszó háttérrel megtoldva olyan hatást kelt, mintha közvetlenül a munkaasztalra írnánk. Íme egy példa:

```
Eterm -x -g 80x20+100+100
➥--scrollbar false -O
➥--buttonbar false
```

Ennek hatására egy 80*20 karakteres terminálablak jelenik meg a 100:100 képponttól kezdve, menüsor és görgetősáv nélkül és a `-O` kapcsoló hatására átlátszóan. Honlapja a ➲ *http://eterm.org* címen található, ahonnan *rpm*, és *deb* csomag formájában is letölthetjük. A program a *libast* függvénytárra épül, így ez szükséges a fordításhoz és futtatáshoz. Rendelkezik saját menüvel, a beállítások onnan is elvégezhetők, de érdemes átbújni a súgóját, mert a fenti példában sem használunk menüt és kényelmesebben testreszabható. Jelenleg 0.9.3 az elérhető legmagasabb stabil verzió.

rxvt/urxvt

Kicsi, gyors, egyszerű és mégis látványos. Támogatja a *Unicode* módban való indítást (innen az *urxvt* név). Alapos man oldal tartozik a programhoz, szinte minden beállítható. Honlapja a ➲ *http://software.schmorp.de/#rxvt-unicode* címen érhető el.

```
urxvt -tr -geometry 80x25 +sb
➥-cr darkred -bc -fn
➥"xft:Bitstream Vera Sans
➥Mono:pixelsize=13" -fg
➥lightgray
```

Sorrendben a kapcsolók: átlátszó ablak, sorok és oszlopok száma, görgetőmező (scrollbar) nem jelenik meg, kurzor sötétvörös és villogó, majd a karaktertulajdonságok beállítása (betűkészlet és méret) és végül karakterszín. Érdemes kísérletezni a beállításokkal.
A legtöbb terminál tud átlátszó vagy részben átlátszó hátteret, itt látványos hatást érhetünk el, ha a *Midnight Commandert* a `-b` kapcsolóval indítjuk, amely hatására fekete-fehér, jelen esetben átlátszó fájlkezelőt kapunk. Ha egy terminált kapcsolókkal indítunk, akkor érdemes indító ikont készíteni hozzá, vagy a korábban említett alias módszert használni.

Rácz Zoltán
(razoli@linuxforum.hu)

Jelenleg egyetemista az ELTE informatika-matematika tanári szakán. A Linuxszal két éve került kapcsolatba az UHU 1.0 kapcsán. Fél éve egyetlen operációs rendszer van a gépén: egy Debian Sid.

Az Ubuntu Linux telepítése és beállítása (4. rész)

Az előző havi száraz témák után most némiképp könnyebb vizekre evezünk a sorozatban: a menürendszer, a grafikus felület csinosítása és a multimédia fontosabb beállítási lehetőségeit vesszük át. Némelyeknek a grafika mellékes, másoknak fontos, ugyanígy, van aki akár hang nélkül is használja mondjuk egy munkahelyi gépen Linuxát, míg másoknak komoly problémát jelent, ha nem tudnak megfelelően videót lejátszani. Tehát a következőkben bár könnyedebb, de sokak számára mégsem elhanyagolandó témákat nézünk át.

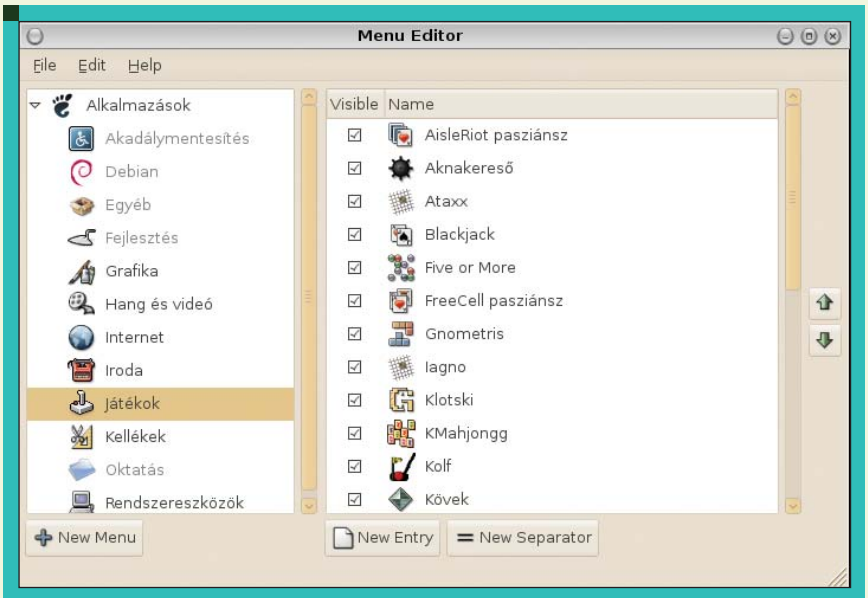
Menük

Az *Ubuntu* és a *Kubuntu* menürendszere némiképp eltér egymástól. A grafikus felületből is következik ez, hiszen az *Ubuntu Gnome*-ot használ, a *Kubuntu* pedig *KDE*-t. Mindegyiknek megvannak a maga hívei és ellenzői, de az tagadhatatlan, hogy ez a két desktop menedzser a legnépszerűbb és a legtöbb szolgáltatást nyújtó. Hogy melyik szimpatikus nekünk, azt talán az is eldöntheti, mennyire áll kézre számunkra a *Windows* grafikus rendszere, mert akkor a *KDE* az ideális választás, a *Gnome* inkább a régi *MacOS*-re hasonlít.

Gnome-ot használva három fő menüt találunk: *Alkalmazások*, *Helyek* és *Rendszer*. Az elsőben nevéhez híven tematikusan csoportosítva az alkalmazásindító ikonokat találjuk, a másodikban a lemezekkel, fájlokkal kapcsolatos dolgokat, a *Rendszer* menüben pedig a beállítások, a rendszer adminisztrációja és egyéb dolgok, mint például az *Ubuntu* névjegy kaptak helyet.

Szerkeszteni az *Alkalmazások* menüket tudjuk. Az *Alkalmazások*-on belül a *Rendszereszközök* almenüben találjuk az *Applications Menu Editor*-t. Ez valójában a *smeg* nevű alkalmazást *(Simple compliant Menu Editor for Gnome)* nyitja meg *(1. ábra)*.

Itt tudjuk aktiválni vagy deaktiválni az egyes menüelemeket, tudunk új menüket és új elemeket létrehozni, létezőket törölni, de a menüelemek sorrendjét is meg lehet változtatni.

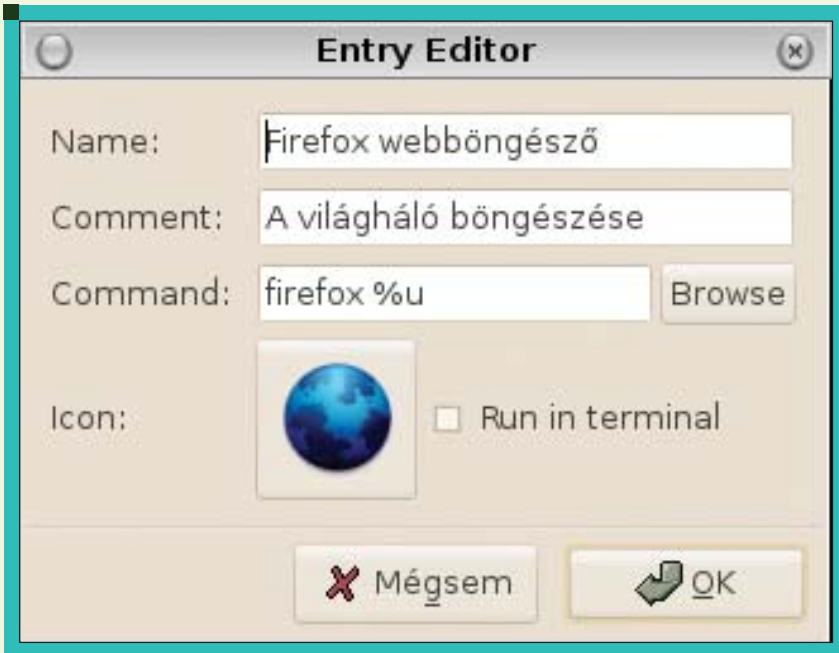


1. ábra A Menu Editor

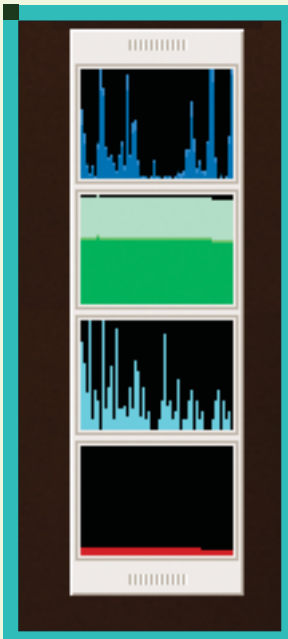
A fájlrendszerben ezeket az */usr/share/applications* könyvtárban találhatjuk meg. Némely elemek lehetnek még a saját könyvtárunkon belül a *.local/share/applications* könyvtárban is. Minden egyes menüelemet egy *.desktop* kiterjesztésű fájl reprezentál, amelyben szöveges információkat találunk a menüelemekről. Például a *firefox.desktop* tartalma:

```
[Desktop Entry]
Encoding=UTF-8
Name=Firefox Web Browser
...
GenericName[hu]=Webböngésző
...
Exec=firefox %u
Terminal=false
MultipleArgs=false
Type=Application
Icon=mozilla-firefox.png
Categories=Application;Network
MimeType=...
StartupWMClass=Firefox-bin
```

Azt hiszem, nem nehéz rájönnünk, melyik elem mit jelent. A *Menu*



2. ábra Egy elem tulajdonságai a Menu Editorban



3. ábra Lebegő panel a Rendszerfigyelő kisalkalmazással

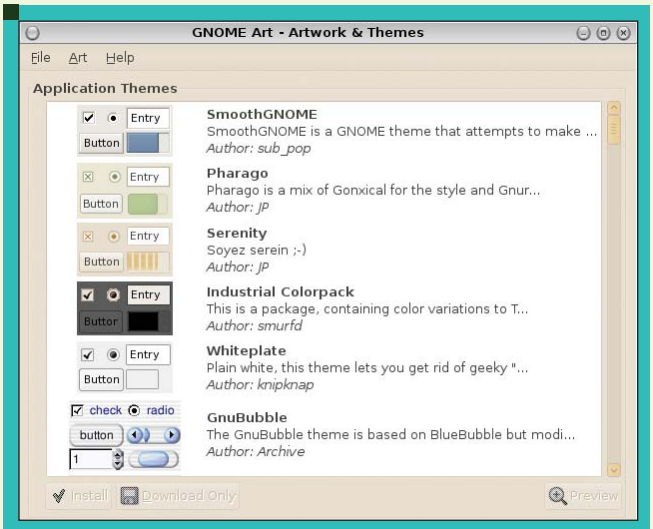
Editorban ezek közül a legfontosabb adatokat látjuk, ha egy menüelemnek a tulajdonságait kérdezzük le *(2. ábra)*.
Az említett könyvtárban az *Alkalmazások* menü elemein kívül megtalálhatjuk a rendszermenük elemeit is, amiket a *Menu Editorral* nem is szerkeszthetünk.
Bár a *smeg Kubuntuban* is használható lenne, van saját menüszerkesztője is a *KDE*-nek. Egy kicsit elrejtve, a *System Settings*-et (vagy *Vezérlőpultot*) elindítva, a *Panel* beállításokon belül az *Edit K Menu* gombbal hívhatjuk elő. Egy fokkal barátságosabb is *Gnome*-beli társánál, de véleményem szerint a *smeg* jobban használható.
A *Kubuntu* menüelemeket a fentebb említett */usr/share/applications* könyvtár *kde* alkönyvtárában találjuk meg.

Panelek

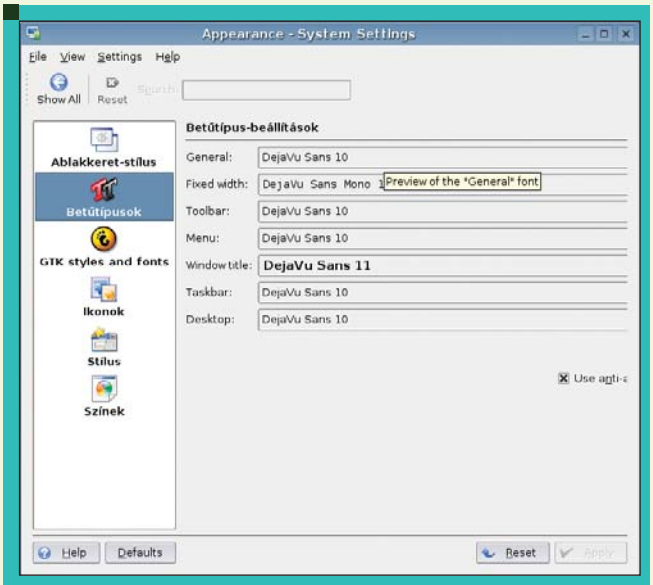
A *Gnome* úgynevezett paneleket használ a menük és egyéb elemek elhelyezésére. Két panel már települ az installálás után is: egy felül helyezkedik el, egy pedig alul. A felső panelre hasznos segítségként érdemes kirakni néhány gyakran használt alkalmazás indító ikonját. Ehhez kattintsunk a jobb egérgombbal a panel egy üres részére és válasszuk a *Hozzáadás a panelhez* menüpontot. Az *Alkalmazásindító ikonnal* egy meglévő menüelemet választhatunk ki, míg az *Egyedi alkalmazásindító ikonnál* minden paramétert magunknak kell megadnunk. Ezen kívül úgynevezett appleteket vagy kisalkalmazásokat is felrakhatunk, ezeket kategóriákba rendezve találjuk itt.
A paneleken lévő elemeket el is mozdíthatjuk vagy éppen rögzíthetjük. Egy panel elhelyezkedését (fent, lent, balra, jobbra), méretét, hátterét, megjelenítési módját szabályozhatjuk, ha a jobb egérgombos menüben a *Tulajdonságok* menüpontot választjuk. Helyzetét akár vontatással is megváltoztathatjuk. Ha a *széthúzás* be van jelölve, a panel – helyzetétől függően – a bal széltől a jobb szélig vagy a képernyő tetejétől az aljáig elfoglalja a helyet, ekkor csak a szélekhez igazodhat, míg ha nincs széthúzva, akkor „lebegő" módban a képernyőn bárhol elhelyezkedhet. Egy már létező panelen a jobb egérgombos menüből választhatjuk az *Új panel* funkciót is, azaz tetszőleges számú panelünk lehet. A *3. ábrán* látható egy ilyen lebegő panel, amelyre a *Rendszerfigyelő* kisalkalmazást helyeztem rá.

A művészi gnóm és a játékos sárkány

Gnome alatt *Rendszer* menüben a *Beállítások* alatt találjuk az *Art managert*, ami igen kézreállóan segít az asztal hátterének, az alkalmazások, ablakok kinézetének, az ikonkészletek beállításában, sőt a *gdm* kinézetét és a *gnome „splash screent"* is átszabhatjuk vele. Az *Art* menüben válasszuk ki a változtatni kívánt dolgot. Az a kellemes meglepetés ér minket (már ha szélessávú kapcsolatunk van), hogy ekkor az internetről automatikusan letölti az *Ubuntu* válogatott témáinak előképeit. Ezekből válogathatunk, kérhetünk egy jobban látható előképet is és telepíthetjük a kiválasztott témát *(4. ábra)*.
A *Kubuntu KDE* felületén a *K* menüben a *System Settings*-et kell használnunk a legkülönfélébb beállításokhoz, így a kinézet csinosítását is itt végezhetjük el az *Appearance* ikont választva. (A *Vezérlőpultban* egy kicsit más a csoportosítás). Nekem első dolgom szokott lenni a betűméretek csökkentése, ezek számomra esetlenül nagyok, a számomra kellemes beállításokat láthatjuk az *5. ábrán*.
Bár több beállítási lehetőségünk van a *KDE*-ben, a csinosításhoz bizony nem kapunk olyan rugalmas eszközt, mint amit az *Ubuntu Art Managere* kínál, az egyes stílusok, témák előre definiáltak és sehol sem látunk letöltési lehetőséget. A háttérkép és *splash screen* (itt nyitókép témaként van fordítva) beállítási lehetőségeket a *System Settings*-en belül



4. ábra Art Manager



5. ábra Betűtípusok beállítása Kubuntu alatt

a ***Munkaasztal*** ikonra kattintva érhetjük el több más funkció mellett. ***Gnome*** alatt a rendszer betűtípusok változtatása szintén igen egyszerű: a ***Rendszer*** menü ***Beállítások*** menüpont alatt a ***Betűkészlet***-et kiválasztva, egyszerűen megadjuk a kívánt betűtípusokat és -méreteket.

Multimédia

Sokszor fontos kérdés, hogy kedvenc disztribúciónkkal gond nélkül lejátszhassunk audio- és videófájlokat. Először is, vizsgáljuk meg, van-e hang az audio kimeneten, tehát hangkártyánk megfelelő dugójába illesszünk egy hangszórót vagy egy fejhallgatót. Jó esetben (***Gnome*** alatt) a felső panel jobb széle felé látunk egy kis hangszóró ikont is. Erre kattintva, állítsuk be a hangerőt is, mert könnyen lehet hogy telepítés után az alapbeállítás a nulla hangerő lesz. Nem árt megnézni a ***Rendszer*** menüben a ***Beállítások*** menüpontban a ***Hang*** alatt lévő lehetőségeket. Itt legyen bejelölve a ***Hangkiszolgáló elindításának engedélyezése*** és a ***Default sound card*** alatt a megfelelő eszköz legyen. Ha itt nem is lehet eszközt választani, akkor az ***Ubuntu*** nem ismerte fel hangkártyánkat és elképzelhető, hogy le is kell cserélnünk, ha hangokat is szeretnénk.
Amennyiben van felismert hangkártyánk, de nincs hang vagy zavarok vannak benne, a ***Rendszer*** menü ***Beállítások*** menüpontján belül válasszuk ki a ***Multimédia rendszer választó*** pontot, ahol kiválaszthatjuk, milyen rendszer szerint működjön a be- és a kimenet külön-külön. Az ***Ubuntu*** az ***ALSA***-t ***(Advanced Linux Sound Architecture)***, ***ESD***-t ***(Enlightenment Sound Daemon)*** és ***OSS***-t ***(Open Sound System)*** ismeri. Bármelyiket választhatjuk, az alapértelmezés ***ALSA*** a kimenetre és ***OSS*** a bemenetre. Érdemes különbözőket választani a be- és kimenetre, így valószínűleg nem „akadnak össze”. Hogy mikor melyik hangrendszer az ideális, szinte lehetetlen megmondani – sajnos egyik sem tökéletes, de a választás ilyen szabadsága szinte egyedülálló a linuxos disztribúciók körében.
Az ***Ubuntu*** alapértelmezett hanglejátszója a ***Rhythmbox***, videólejátszója pedig a ***Totem***. Mindkettő kiváló szoftver, de mellettük elég nagy választék van multimédiás lejátszókból. Mielőtt azonban boldogan zenét kezdenénk el hallgatni vagy videókat játszanánk le, nem árt tudni, hogy licenszelési okokból az ***Ubuntu*** nem fog mindenféle fájlformátumot lejátszani. Természetesen a ***wmv*** formátum is ide tartozik, de akár egyes ***mp3*** fájlok is ismeretlenek lehetnek számára. Ne keseredjünk el: van megoldás! Vegyük fel tárolóink közé a ***„universe”***-t és a ***„multiverse”***-t is, majd frissítsünk a ***Synaptickal*** vagy a `sudo apt-get update` paranccsal és installáljuk az alábbi csomagokat:

```
gstreamer0.8-plugins
gstreamer0.8-ffmpeg
gstreamer0.8-mad
gstreamer0.8-lame
lame
gstreamer0.8-plugins-multiverse
mjpegtools
ffmpeg
totem-xine
vorbis-tools
sox
faad
```

A ***Microsoft wmv*** formátumához való kodek sajnos nincs meg az ***Ubuntu*** tárolókban, de máshonnan letölthetjük. Ehhez vegyük fel a következő tárolót:

```
deb
ftp://ftp.free.fr/pub/Distribut
ions_Linux/plf/ubuntu/plf/
breezy non-free
```

Frissítsünk, majd töltsük le a `w32codecs` csomagot, ahol a ***Windows*** formátumokhoz találhatóak kodekek. Persze ne felejtsük el, hogy ez sajnos licenszproblémákat fog felvetni, de bizonyára ez keveseknek okoz majd álmatlan éjszakákat.
A ***Rhythmbox*** és a ***Totem*** mellett egy sor más lejátszót is használhatunk: ki-ki az ízlésének megfelelőt. Ideális választás nincs, csak a választás szabadsága. Bizonyára feltűnik majd, hogy egyesek ***KDE***-s alkalmazások. Igen, lehet ***Gnome*** alatt ***KDE***-s alkal-

mazásokat futtatni, az ***Ubuntu*** le is fogja tölteni hozzájuk a szükséges környezetet, ezzel sem kell törődnünk. Ennek egyetlen hátránya, hogy ha először telepítünk ***Gnome*** alá ***KDE*** szoftvert, letöltési mennyiségünk meglehetősen megnő.
Csak felsorolás szinten néhány ismertebb és az ***Ubuntu*** tárolókban megtalálható lejátszó és egyéb multimédia alkalmazás:

- ***AmaroK (KDE)***: Jól használható zenelejátszó
- ***Kaffeine (KDE)***: Nagytudású audio-video lejátszó
- ***Mplayer***: Magyar fejlesztésű multimédia lejátszó, nagyon sokféle formátumot ismer, képes terminálon és grafikus felülettel is futni, sőt TV-kártya kezelésére is
- ***Noatun (KDE)***: egyszerű audio-video lejátszó ***KDE*** alá
- ***Realplayer***: internetes multimédia tartalom lejátszásához használható, kiváló minőségű szoftver
- ***Vlc*** (***Gnome*** alá ***Gnome-vlc*** néven keressük): Nagyon sokféle audio- és videoformátumot ismerő, csaknem minden platformon használható multimédia lejátszó
- ***Xine***: Népszerű és rugalmas videolejátszó
- ***XMMS***: ***Winamp*** klón ***Linux*** alá, azaz zenelejátszó, sokféle formátumot támogat

CD-rippeléshez:
- ***Grip (Gnome)***
- ***Goobox (Gnome)***
- ***Kaudiocreator (KDE)***
- ***Sound-juicer (Gnome)***

Audio CD készítés hangfájlokból, listákból:
- ***Serpentine***

TV nézés:
- ***Zapping (gnome)***
- ***Tvtime***
- ***Motv***
- ***Xawtv***

TV felvétel:
- ***Mythtv***
- ***Xawtv***

Láthatjuk, hogy az ***Ubuntu*** alatt nem kell elhanyagolnunk multimédiás tevékenységünket sem, egy kis segítséggel nagy örömünkre válhat a ***Linux*** használata. Ezek után bátran indulhatunk neki a harmadik dimenzió és a játékok felfedezésének is.
A következő számban a modern videokártyák 3d-s lehetőségeit fogjuk kihasználni, valamint az eddig kicsit mellőzött eszközöket nézzük át: ***USB***-tárolók, fényképezőgépek, nyomtatók, szkennerek, videokamerák, palmtopok, ***Bluetooth*** eszközök, vezeték nélküli hálózatok beállításáról lesz szó.

Addig is: jó szórakozást az ***Ubuntuval***!

Molnár Norbert
(molnar.norbert@gmail.com)
35 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.

Irodai internetkapcsolat kialakítása mobiltelefon segítségével

A mobilszolgáltatók már előálltak olyan csomagokkal, melyek cégen belüli ingyen hívásokat tesznek lehetővé. Ezt kihasználva lassú, de használható, bárhonnan elérhető kapcsolatot tudunk létrehozni úgy, hogy a kliens közvetlenül a céges intranetre kapcsolódik, azon keresztül elérheti a céges belső információkat, és a cég internetkapcsolatán keresztül a világhálót is böngészheti.

Cégemnél, az egri ***Városi Ellátó Szolgálatnál*** építettem ki egy ilyen rendszert. A cél elsősorban az volt, hogy a GSM hálózat segítségével csökkentsük a levelek és dokumentumok kívülről történő elérésével kapcsolatos költségeket.
A mobilhívások az alközponton keresztül az adapteren, a vezetékesek ***ISDN*** kapcsolaton át mennek ki.
Az adapterre céges mobilról bejövő hívások egyrészt ingyenesek, másrészt az alközponton keresztül mellékre kapcsolhatóak, így a mellékekről kezdeményezett mobilhívásokat az alközpont automatikusan az adapteren keresztül küldi ki.
Ezeken felül a bemutatott infrastruktúra segítségével lehetőség van mobilos faxolásra, illetve *SMS*-ek küldésére-fogadására, sőt azok parancsként való futtatására is.
Talán lényeges azt is megjegyezni, hogy a cikkben szereplő technológiáknak a *GPRS*-hez, vagy a *3G*-hez nincs köze, minden a hagyományos „betárcsázós elven” működik.

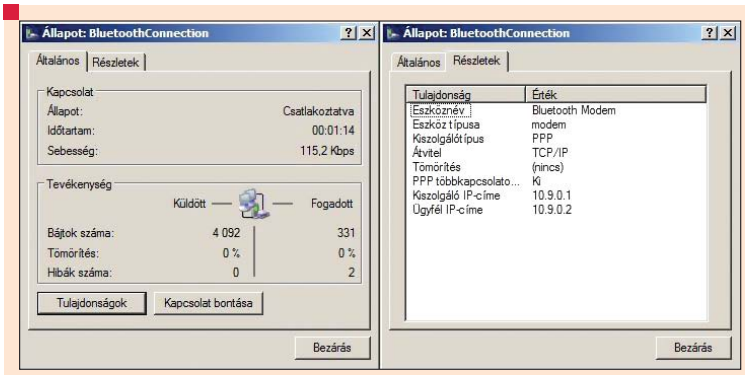
1. ábra Képek a GSM Adapterről

A GSM adapter telepítése

A *GSM* adapter esetünkben egy ***Nokia 32 GSM Terminal***, aminek az adatátviteli sebessége ugyan csak 9600 bit/sec, de képes arra, hogy szétválasztva az adat és a hanghívásokat, a hangot ***RJ-11***-es telefonkábelen az alközpontra, az adathívást pedig ***RS-232***-es csatolón keresztül a számítógépre továbbítsa.
Amennyiben nem lényeges az alközponttal való összekapcsolhatóság, adapter helyett lehet közönséges mobiltelefont is használni. Így az adatátviteli sebesség is növelhető. Érdemes régebbi, soros portra csatlakoztatható típust használni, ***USB***-vel, infrával vagy ***Bluetooth***-al lehetnek gondok. Esetünkben tehát az adapter ***RS-232***-n keresztül, soros kábellel csatlakozik a szerver */dev/ttyS1* (*COM1*) portjára. Az adaptert a dokumentációjában megadott módon be kell programozni, hogy az adat/faxhívásokat a szerverre küldje, ne a telefonközpontba.

A szerver beállítása

A szerver konfigurálása nem túl bonyolult. Két részből áll, a modemet kezelő ***mgetty+sendfax***, és a bejelentkezést és a kapcsolatot kezelő ***pppd*** beállításából. Ezek a programok szinte mindegyik disztribúció esetén alapértelmezésként települnek. Ha mégsem találjuk őket, akkor fordítani kell, de az sem bonyolult. Amennyiben az ***mgetty+sendfax*** nem a ***-DAUTO_PPP*** opcióval lett fordítva, szintén szükség lesz az újrafordítására.



2. ábra A kapcsolat tulajdonságai

Mint említettem, nálam a *GSM* adapter soros kábelen keresztül a */dev/ttyS1* portra csatlakozik, a helyes inicializáló parancsokat (*init-chat* sor az */etc/mgetty+sendfax/mgetty.config*-ban) pedig az adapter dokumentációjából és a *minicom*-al való próbálgatással sikerült kideríteni.

Az mgetty+sendfax beállítása

Szúrjuk be az */etc/inittab* fájlba a következő két sort:

```
# Dialup lines:
d2:345:respawn:/sbin/mgetty -x
➥4 ttyS1
```

Az /etc/mgetty+sendfax/mgetty.config megfelelő részét a következőképp kell módosítanunk:

```
# --- port specific section ---
#
# Here you can put things that
are valid only for one line,
not the others
#
port ttyS1
  speed 115200
  debug
  modem-type data
  init-chat "" AT OK
  toggle-dtr yes
  rings 0
  data-only y
```

A sebességre (`speed`) és az inicializáló parancsra (`init-chat`) vonatkozó beállítások a *GSM* adaptertől függően változhatnak. A `rings` megadja, hány csengetés után válaszoljon az adapter, a `data-only` és a `modem-type` sorok pedig arra utalnak, hogy az eszköz kizárólag adatmodemként működik, nem fogad hang vagy faxhívásokat, és a hívás típusát így nem is kell vizsgálni. Kommentezzük ki az */etc/mgetty+sendfax/login.config* fájl következő sorát:

```
/AutoPPP/ -       a_ppp
➥/usr/sbin/pppd
```

A bejövő hívásokat hívószám alapján – amennyiben szükséges – itt szűrhetjük meg:

```
/etc/mgetty+sendfax/dialin.config
```

A pppd beállítása

Hozzuk létre vagy módosítsuk a */etc/ppp/options* fájlt a következő tartalommal:

```
asyncmap 0
netmask 255.255.254.0
proxyarp
lock
crtscts
modem
```

Ez a fájl *pppd* globális beállításait tartalmazza, amelyek minden *PPP* kapcsolatra érvényesek lesznek, modemtől függetlenül. Az `asyncmap` és a `modem` alapvető opciók, a `netmask` az alhálózati maszkot állítja be, ami *PPP* kapcsolat esetén szintén adott.
A `proxyarp`-pal a kapcsolatot átjárhatóvá tesszük a többi interfész felé, tehát a betárcsázós kapcsolaton keresztül elérhetjük a helyi hálózat többi gépét és beállításoktól függően az internetet is. A `lock` kizárólagossá teszi a modemeszköz használatát, így más program nem „piszkálhatja” a modemet. A `crtscts` a hardveres kapcsolatvezérlést kapcsolja be.
Ezeken kívül hasznos lehet még a `debug` opció, amelynek hatására a démon részletes információkat ír a rendszernaplóba a kapcsolatunkról. Szintén jól jöhet a `bsdcomp 15,15` beállítás, amellyel a szoftveres tömörítést kapcsolhatjuk be a példában megadott maximális tömörítési értékekkel.
A *pppd* többi opciójáról a `man pppd` parancs segítségével tájékozódhatunk.
A modemnek megfelelő *options* fájlnak (*/etc/ppp/options.ttyS1*) a következőt kell tartalmaznia:

```
10.9.0.1:10.9.0.2
ms-dns 192.168.1.1
ms-dns 62.112.192.3
```

Ez a fájl az adott eszközön keresztül létrehozott kapcsolatok működését befolyásolja. Az első sor a szerver és a kliens *IP* címét adja meg, a második kettő a *DNS* szerverek címeit. Biztonsági szempontból jó döntés lehet külön felhasználót létrehozni a betárcsázós kapcsolathoz, de a meglévő felhasználók beléptetése is megoldható. Ha a felhasználónév és a jelszó megvan, már ki is próbálhatjuk a kapcsolatot.

Az ügyfél beállítása

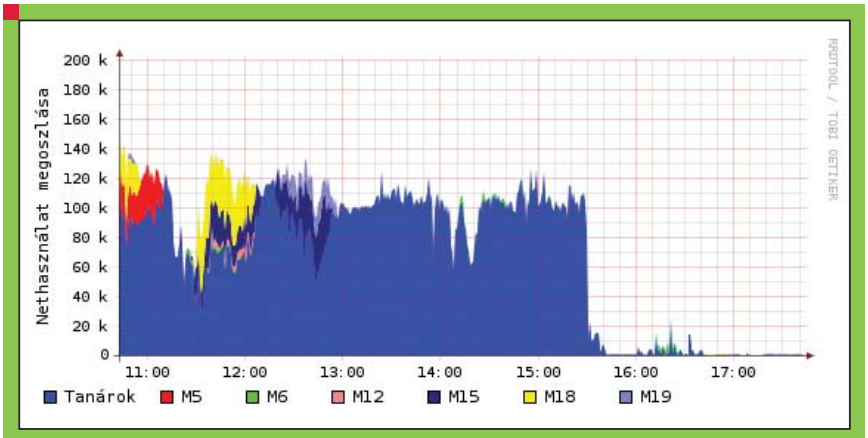
A betárcsázó számítógép konfigurálása az analóg vezetékes internetkapcsolatok beállításához hasonlóan történik, azzal az eltéréssel, hogy jelen esetben a modem valószínűleg egy mobiltelefon, mivel vezetékesről mobilt hívni drága. Legjobb, ha az adapterben levő *SIM* kártya és a betárcsázó telefon azonos céges előfizetői csomagba tartozik, és így a hívás ingyenes. A telefon és a számítógép közti kapcsolat megoldható *RS-232*-es vagy *USB*-s soros kábellel, infrával vagy *Bluetooth*-al. *Windowsos* kliens esetén szükséges lehet a gyártótól kapott modem meghajtó feltelepítése.
Az adapter telefonszámával és a szerveren létrehozott felhasználónév-jelszó párossal immár létrehozhatjuk mobilos internetkapcsolatunkat, ami ugyan csigalassú, viszont ingyenes és szinte bárhol elérhető.

Csuhai Imre
(csuhi@csuhi.homelinux.net)
Közalkalmazottként és egyéni vállalkozóként közbeszerzéssel és informatikával foglalkozom, főként szerverek és hálózatok érdekelnek. Legszívesebben Slackware-t és UHU-Linuxot használok, szórakozásképpen pedig Quakekel és egyéb FPS-ekkel játszom.

Grafikonok minden mennyiségben – RRDTool (2. rész)

Az RRDTool egyik nagy előnye, hogy a puszta adatokkal képes különféle egyszerűbb műveleteket végezni, illetve ezeket a számolt értékeket a grafikonon kiírni.

Az *RRDTool* által használt matematikai rendszer kicsit érthetetlen a köznapi matematikán nevelkedett szemnek, hiszen inverz lengyel logikán alapul, amelyet *RPN (Reverse Polish Notation)* néven illetnek angol nyelvterületen. Hamar meg lehet szokni, de sok bosszúságot tud okozni, ha nem figyelünk oda. A legjobb tanító a példa, így egy konkrét megvalósításon át mutatom be egy összetett a grafikon készítését.



1. ábra Forgalmi statisztika

Hálózati forgalom megoszlása

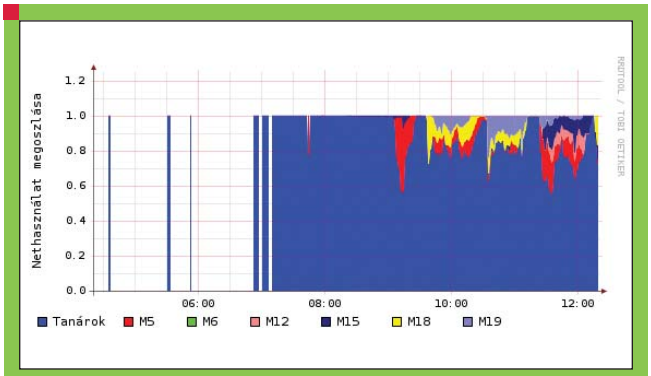
Érdekes statisztikát tudunk készíteni, ha egy iskolában a számítógéptermek által okozott forgalmat arányaiban összevetjük. Ehhez egy megfelelő tűzfalszabályt kell létrehoznunk, amely *IP* tartomány szerint számlálja az átmenő forgalmat, majd a mért adatokat rendre beletesszük egy *RRD* fájlba.
Első nekifutásra elég, ha egymásra pakoljuk a forgalmi adatokat:

```
/usr/local/bin/rrdtool graph subnet.png \
--imgformat PNG -u 150000 -l 0 --width 464
➥--height 200 \
--end now-3h --start end-7h --vertical-label
➥"Nethasználat megoszlása" \
--slope-mode --lazy --color "SHADEA#ffffff" \
--color "SHADEB#ffffff" --color "BACK#ffffff" \
"DEF:Tanar=/var/db/rrd/subnet/subnet.rrd:Tanar:
➥AVERAGE" \
"DEF:M5=/var/db/rrd/subnet/subnet.rrd:M5:
➥AVERAGE" \
"DEF:M6=/var/db/rrd/subnet/subnet.rrd:M6:
➥AVERAGE" \
"DEF:M12=/var/db/rrd/subnet/subnet.rrd:M12:
➥AVERAGE" \
"DEF:M15=/var/db/rrd/subnet/subnet.rrd:M15:
➥AVERAGE" \
"DEF:M18=/var/db/rrd/subnet/subnet.rrd:M18:
➥AVERAGE" \
"DEF:M19=/var/db/rrd/subnet/subnet.rrd:M19:
➥AVERAGE" \
"AREA:Tanar#0000FF:Tanárok":STACK \
"AREA:M5#FF0000:M5       ":STACK \
"AREA:M6#00FF00:M6       ":STACK \
"AREA:M12#FF8888:M12      ":STACK \
"AREA:M15#000088:M15     ":STACK \
"AREA:M18#FFFF00:M18     ":STACK \
"AREA:M19#8888FF:M19     ":STACK
```

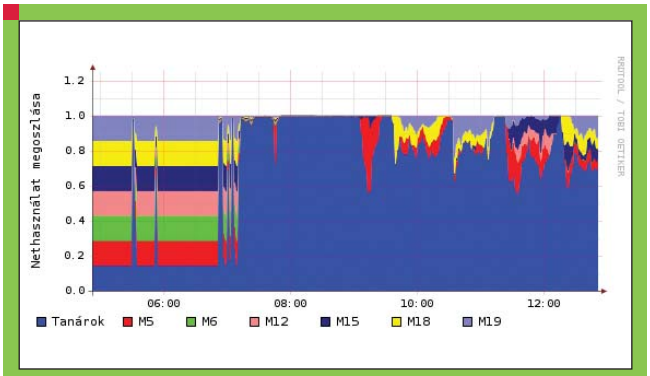
Az *1. ábrán* látható grafikon elkészítéséhez csak olyan paramétereket és tulajdonságokat használtunk fel, amelyeket már ismertettem az előző részben, s a grafikon sem azt ábrázolja, amire szükségünk van, ugyanis nem látszik rajta tisztán a hálózat terhelésének százalékos megoszlása.
A feladat elvégzéséhez egyszerűen össze kell adnunk az aktuális forgalmakat, majd a mért adatokat egyenként el kell osztanunk a teljes terheléssel, majd ezeket ábrázolni.
Az *RRDTool* ehhez egy `CDEF` direktívát ad a kezünkbe, amellyel létre tudunk hozni új változókat:

```
"CDEF:Osszeg=M19,M18,M15,M12,M6,M5,Tanar,+,+,+,+,
➥+,+"
```

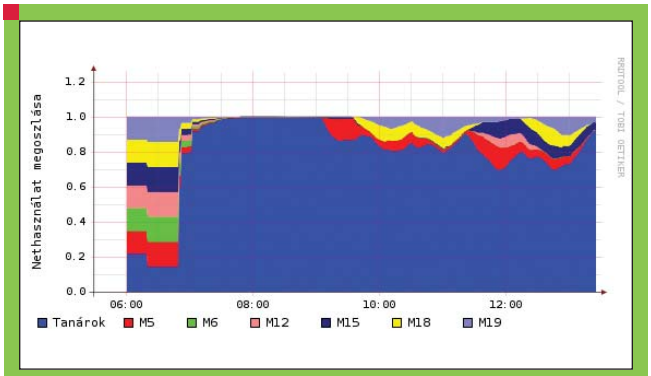
Mint említettem, az összeadás módja messze áll az általános iskolában tanult módszertől. Az inverz lengyel logika olyan ága a matematikának, amelyet a zárójelezés és egyéb problémák kicselezése okán talált ki *Jan Lukasiewicz* lengyel származású matematikus.



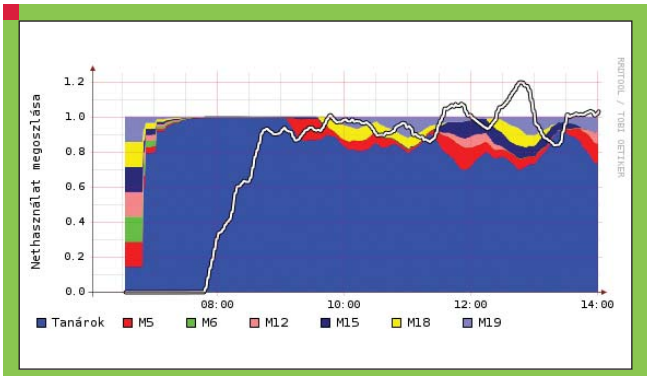
2. ábra Letöltési arányok, némi hibával



3. ábra Letöltési arányok



4. ábra Letöltési arányok simítva



5. ábra Összes letöltés

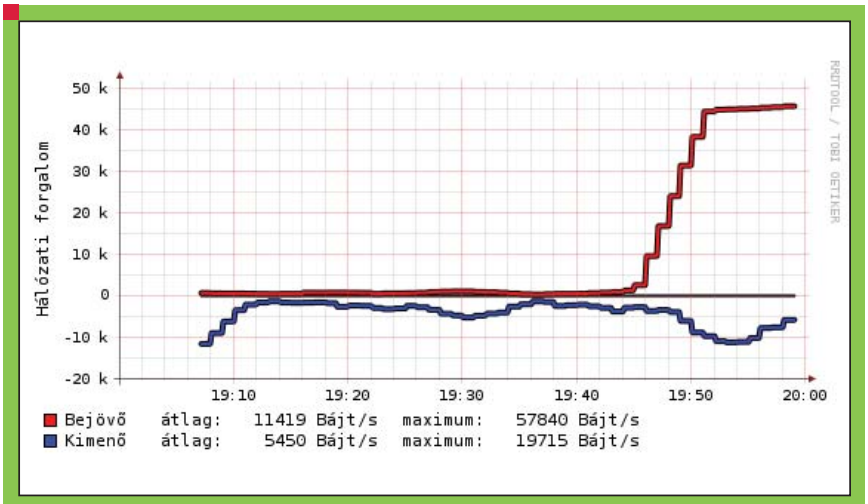
A módszer abból áll, hogy a nem a változók közé tesszük a műveleti jeleket, hanem eléjük vagy utánuk. A műveletvégzés pedig úgy zajlik, hogy a feldolgozó szál levesz egy műveleti jelet, majd annyi változót, amennyit az adott művelet igényel. A fenti példában a legelső összeadás jel fog először végrehajtódni, és az ***RRDTool*** műveletvégző eljárása leemeli a tőle balra lévő adatokat – az M5 és a Tanar változókat – a listáról, majd elvégzi rajtuk a műveletet és visszateszi az eredményt a helyükre. Az eljárás addig tart, amíg el nem fogynak a műveleti jelek, ekkor az utolsó változó értékét a megadott nevű változó felveszi. Tulajdonképpen nem értem, hogy miért nem építettek bele az ***RRDTool*** programba valami „emberibb" matematikai modult, valószínűleg a készítőknek egy ***HP*** kézi kalkulátor volt az etalon...

```
"CDEF:Osszeg=M19,M18,M15,M12,M6,M5,Tanar,+,+,+,+,
➥+,+" \
"CDEF:TanarSzazalek=Tanar,Osszeg,/" \
"CDEF:M5Szazalek=M5,Osszeg,/" \
"CDEF:M6Szazalek=M6,Osszeg,/" \
"CDEF:M12Szazalek=M12,Osszeg,/" \
"CDEF:M15Szazalek=M15,Osszeg,/" \
"CDEF:M18Szazalek=M18,Osszeg,/" \
"CDEF:M19Szazalek=M19,Osszeg,/" \
```

Az összeg kiszámítása után már csak arra kell figyelnünk, hogy ne osszunk nullával, hiszen az nem ábrázolható a grafikonon, ilyenkor kihagyásokat fogunk kapni a grafikonon. Minden egyes tartomány terheléséhez hozzáadunk egy kicsi számot, az összeghez pedig még annyit kell hozzáadnunk, amennyi tartományt ábrázolni szeretnénk:

```
"CDEF:Osszeg=0.7,M19,M18,M15,M12,M6,M5,Tanar,+,+,
➥+,+,+,+,+" \
"CDEF:TanarSzazalek=Tanar,0.1,+,Osszeg,/" \
"CDEF:M5Szazalek=M5,0.1,+,Osszeg,/" \
"CDEF:M6Szazalek=M6,0.1,+,Osszeg,/" \
"CDEF:M12Szazalek=M12,0.1,+,Osszeg,/" \
"CDEF:M15Szazalek=M15,0.1,+,Osszeg,/" \
"CDEF:M18Szazalek=M18,0.1,+,Osszeg,/" \
"CDEF:M19Szazalek=M19,0.1,+,Osszeg,/" \
```

Itt már fokozódik az ***RPN*** műveletek átláthatatlansága, ugyanis a műveletek sorrendje és a változók sorrendje különbözik! Az összeg kiszámításánál ez nem okoz gondot, hiszen összeadásnál a sorrend felcserélhető. Ha osztani vagy szorozni is akarunk, akkor úgy kell a sorrendet kialakítani, ahogy azt a képlet megkívánja. Gyakorlatilag egy `(Tanar+0.1)/Osszeg` műveletet kell végrehajtanunk, amelyet az ***RPN*** alapján a példa szerint tudunk megoldani. Az ***RRDTool*** megtalálja az első műveleti jelet, és a tőle balra lévő adatokon végrehajtja, majd a helyükre teszi az eredményt, így a `Tanar` és a `0.1` összege fog itt szerepelni. Ezek után ezt elosztjuk az Osszeg változó értékével, és elő is áll a kívánt grafikon (*3. ábra*).
Egy kicsit tudjuk javítani a grafikon kinézetét, ha a durva csúcsokat kicsit lesimítjuk a csúszóablak használatával, így egy kissé hamis, de sokkal informatívabb ábrát tudunk kapni:



6. ábra Írás a jelmagyarázaton

```
"VDEF:outAvg=out,AVERAGE" \
"VDEF:outMax=out,MAXIMUM" \
```

Ezeket akár ki is jelezhetjük a kész grafikonon, viszont sokkal jobb, ha kiírjuk a színek magyarázatához a konkrét értékeket:

```
"LINE2:inT#FF0000:Bejövő " \
"GPRINT:inAvg:átlag\: %7.0lf
➥Bájt/s" \
"GPRINT:inMax:maximum\: %7.0lf
➥Bájt/s\n" \
"LINE2:outNegT#0000FF:Kimenő " \
"GPRINT:outAvg:átlag\: %7.0lf
➥Bájt/s" \
"GPRINT:outMax:maximum\: %7.0lf
➥Bájt/s\n" \
```

```
"CDEF:TanarT=Tanar,1800,TREND" \
"CDEF:M5T=M5,1800,TREND" \
"CDEF:M6T=M6,1800,TREND" \
"CDEF:M12T=M12,1800,TREND" \
"CDEF:M15T=M15,1800,TREND" \
"CDEF:M18T=M18,1800,TREND" \
"CDEF:M19T=M19,1800,TREND" \
```

A csúszóablak lényege, hogy a grafikonon a megadott időtartam átlaga fog megjelenni, így elsimítjuk a kiugró csúcsokat, az arányok változását szépen követhető vonalakká alakíthatjuk. Az egyetlen probléma, hogy más-más ablakméretet kell használnunk az utolsó egy óra és az utolsó egy év ábrázolásához, de ezt hamar meg tudjuk oldani. Ha meg tudjuk határozni a vonalunk maximális áteresztő képességét, akkor érdemes erre a grafikonra egy vonallal rárajzolni az aktuális százalékos terhelést is. Ebből egy szám rendelkezésünkre áll, mégpedig az összes áteresztett forgalom, ezt el kell osztani azzal a számmal, amennyi a bejövő vonalunk teljes kapacitása. Ezt esetleg korrigálnunk kell a *HTTP proxy* által kiszolgált tartalommal, amely a 100% fölé tudja emelni a letöltött adatmennyiséget.

```
"CDEF:TeljesSzazalek=Osszeg,131072,/" \
"LINE4:TeljesSzazalek#000000:" \
"LINE2:TeljesSzazalek#FFFFFF:" \
```

Statisztika, függvények

A grafikonok alapján néhány mért adatot ki tudunk írni, így pontos képet kaphatunk a mért és tárolt adatokról. A `CDEF` direktíva mellett van egy `VDEF` is, amely nem az aktuális időpont értékét kezeli, hanem a megjelenített tartományon képes műveleteket végezni. Általában három műveletet szoktunk használni a rendelkezésre álló tíz közül: az átlag, a minimum és a maximum meghatározása. A `VDEF` használata nagyon egyszerű, nézzük meg egy hálózati csatlakozón átfolyó adatmennyiséget, és határozzuk meg az átlagot és a maximumot:

```
"VDEF:inAvg=in,AVERAGE" \
"VDEF:inMax=in,MAXIMUM" \
```

A kiírást úgy kell elvégeznünk, hogy az *RRDTool* a grafikon alján egy grafikus kurzort mozgat a kiírt karakterek szerint. Így a jelmagyarázat után közvetlenül ki tudjuk írni, hogy mennyi az átlagos és mennyi a maximális forgalom, a *C* nyelvből örökölt formátumszövegeket használva. Munkánk eredménye egy grafikon, amelynek a jelmagyarázata tartalmazza a kívánt értékeket.

Összegzés

Az *RRDTool* ideális eszköz, hogy különféle tendenciákat és jellegzetességeket figyelhessünk meg a rendszeresen keletkező adatainkon, legyenek ezek számítógépekhez közel álló adatok (terhelés, forgalom, hőmérséklet, stb), vagy egyéb mérések eredményei (időjárás, játék, tőzsdei információk, stb).

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

Az RRDTool honlapja
➲ http://www.rrdtool.org

A cikkben említett példák forrása
➲ http://user.enaplo.hu/~auth.gabor/rrd/

FreeCIV játék RRDTool követése
➲ http://user.enaplo.hu/~auth.gabor/freeciv/

Egy való életből vett példa
➲ http://www.enaplo.hu/index.jsp?page=visitor.loadStat

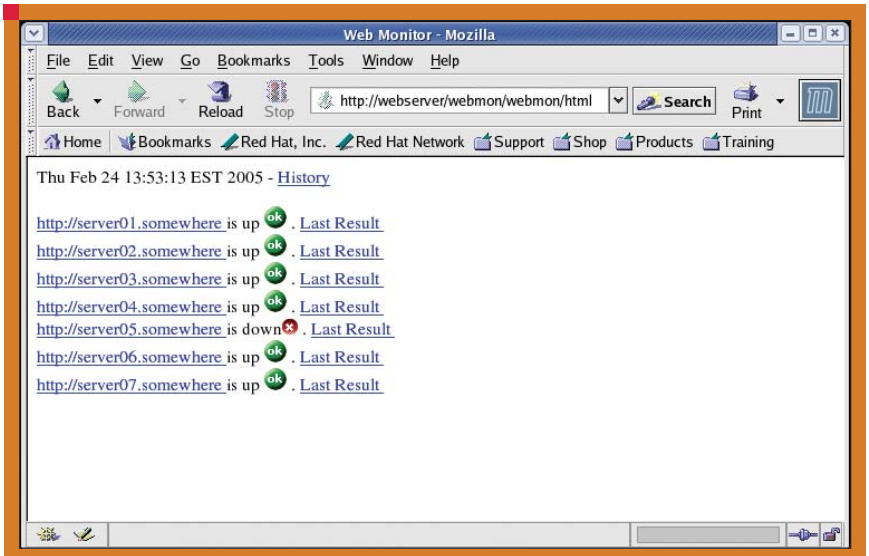
Rendszerműszerfal

Ebben a cikkben néhány olyan egyszerű héjprogramot mutatunk be, amelyekkel folyamatosan megfigyelés alatt tarthatjuk a betelésre hajlamos lemezeket, a processzort jelentősen terhelő folyamatokat, vagy a web illetve a levelezés problémáit.

A cégem körülbelül egy éven keresztül szenvedett egy megfelelő rendszermegfigyelő szoftver hiányától. A téves riasztások, a tévesen kiállított túlóralapok komolyan veszélyeztették a munkahelyi morált és pocsékolták a rendszergazdák idejét. Miután megbeszéltem a kollégákkal, hogy pontosan milyen monitorozó eszközre lenne szükségünk, az derült ki, hogy a megfigyelendő dolgok listája nem is olyan hosszú:

- Tudnunk kell a teljes *HTTP* forgalomról, vagyis nem csak a fizikai kiszolgálókról.
- Figyelnünk kell a lemezterület felhasználását.
- Tudnunk kell, hogy *SMPT* protokollon keresztül rendelkezésre állnak-e az *SMTP* kiszolgálók, vagyis megint nem elég a kiszolgáló egyszerű fizikai megfigyelése.
- Rögzítenünk kell az ezekkel kapcsolatos eseményeket, hogy kiszűrhessük az esetleges problémákat.

Ebben a cikkben bemutatom az általam kifejlesztett megoldást, illetve azt, hogy miként valósítottam meg könnyen és gyorsan a lemezek, web- és levélkiszolgálók megfigyelését. Ahhoz, hogy a monitoring rendszer egyszerű maradhasson, minden felhasznált eszköznek jelen kellett lennie valamelyik aktuális *Linux* terjesztésben, illetve nem használhattam sem összetett adatbázisokat, sem olyan kifinomult protokollokat mint teszem azt az *SNMP*. Ami azt illeti, a teljes rendszerem kizárólag a *Bash* szolgáltatásait, alapvető *HTML* megoldásokat, néhány *Perl* szkriptet és a *wget* segédprogramot használja.



1. ábra A monitor_web.sh működés közben. Futtassuk a szkriptet a cron segítségével a nekünk megfelelő időközönként. Minden futás újragenerálja a weblapot.

Valamennyi szkript felépítése ugyanazt az alapvető vázat követi, ugyanúgy telepíthetők, és valamennyi letölthető a *Linux Journal FTP*-helyéről (lásd a kapcsolódó címeket). A programok telepítését a következőképpen végezhetjük el. Először is másoljuk fel őket egy webkiszolgálóra, majd a `chmod` segítségével tegyük valamennyi mindenki számára futtathatóvá. Hozzunk létre a webkiszolgáló gyökerében egy olyan könyvtárat, amelyben a szkriptek a naplóikat tárolhatják. A `monitor_web.sh` megvalósítása során a `webmon` képességeit használtam fel. Ehhez teljesen hasonlóan a `monitor_smtp.sh` az `smtpmon`-ra, míg a `monitor_stats.pl` szkript a `stats` parancsra támaszkodik. A `monitor_disk.sh` program ezektől annyiban különbözik, hogy ezt helyileg kell telepíteni minden olyan kiszolgálóra, amit meg akarunk figyelni.

A következő lépésben a szkriptek futását időzítenünk kell a *cron* segítségével. A programok bármely olyan felhasználó nevében futhatnak, akinek megvannak a jogosultságai a `wget`, a `df -k` és a `top` futtatásához. Az illető felhasználónak ezen kívül írási joggal kell rendelkeznie a szkriptek saját könyvtárára.

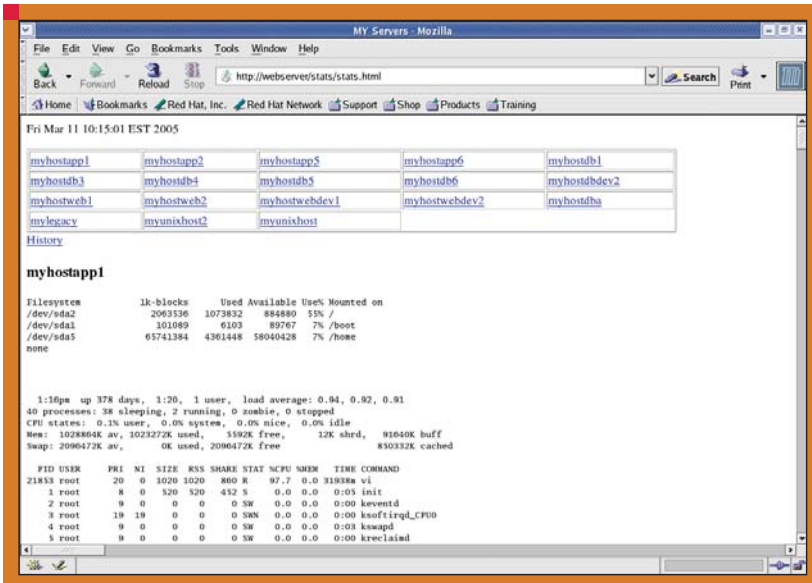
A legegyszerűbb talán az, ha létrehozunk egy monitor nevű virtuális felhasználót, és minden programot ennek a nevében futtatunk. Végül ha telepített rendszerünknek nem része a wget, ezt is telepítenünk kell.
Az első próbálkozásom a webkiszolgálók megfigyelése volt. „Motornak” a már említett `wget`-et használtam és tulajdonképpen e köré írtam egy megfelelően kialakított szkriptet. Ez lett a `monitor_web.sh`. Annak, aki esetleg nem ismerné a `wget`-et, leírom, hogy mit állít róla a program szerzője: ***„a wget olyan szabad szoftver, amely fájlokat tud letölteni HTTP, HTTPS és FTP segítségével, vagyis az interneten legelterjedtebben használt protokollokkal”*** (lásd a kapcsolódó címeket).
Telepítés után a `monitor_web.sh`-nak mindössze két információra van szüksége: hova küldhet e-mailt és mely webkiszolgálókat tartsa megfigyelés alatt. A megadott ***URL***-eknek igazodniuk kell a ***HTTP*** szabványokhoz, a kiszolgálóknak pedig érvényes `http 200 OK` karakterláncot kell visszaküldeniük ahhoz, hogy a szkript helyesen működjön. A címekben ***HTTP*** és ***HTTPS*** protokoll egyaránt szerepelhet, mivel a szkript is és a ***wget*** is támogatja mindkettőt. Ha telepítettük és futtatni kezdtük a megfigyelőprogramot, az erre feljogosított felhasználó a ***http://localhost/webmon/webmon.html URL***-en jelenítheti meg a statisztikákat és eseményeket egy webböngésző segítségével.
Vizsgáljuk most meg közelebbről a `monitor_web.sh` szkript belső szerkezetét. Először is beállítjuk az összes olyan változó értékét, amit a rendszer segédprogramjai, illetve a `wget` használ. Ezek természetesen helyről helyre változhatnak, tehát beállításuk a telepítés része. A következő lépésben ellenőrizzük, hogy él-e a hálózati kapcsolat. Ez megakadályozza, hogy túlterheljük a ***Sendmail*** rengeteg téves riasztás kiküldésével abban az esetben ha a monitorozó gép valamiért leválik a hálózatról.
Aztán egy ciklussal végigmegyünk az összes megadott ***URL***-en úgy, hogy a `wget` öt másodperces késleltetéssel kétszer kísérel meg kapcsolódni valamennyi kiszolgálóhoz. Ha a kapcsolatfelvétel sikertelen, a szkript elküld egy levelet a beállítása során megadott címre, és frissíti a statisztikai weblapot. A program ezután folyamatosan figyeli, hogy mikor éled fel újra a kérdéses kiszolgáló, és erről az eseményről is küld egy levelet. Mivel minden váltáskor csak egy ilyen levél megy ki, nem terheljük túl a címzett gépét. Mindezt a következő kód valósítja meg:

```
wget $URL  2>>   $WLOG
if (( $? != 0 ));then
  echo \<A HREF\="$URL"\>$URL\
  ➥<\/A\> is down\
          $RTAG $EF.\
          $LINK Last Result
          ➥$LTAG  >> $WPAGE
  if [[ ! -a down.$ENV ]];then
        touch /tmp/down.$ENV
        mail_alert down
  else
        echo Alert already
        ➥sent for $ENV \
        - waiting | tee -a
        ➥$WLOG
  fi
fi
```

Miután elintéztük a webkiszolgálókat, ideje, hogy a lemezhasználatot is górcső alá vegyük. Az imént meghirdetett egyszerűség szent nevében itt sem történik semmi egyéb, mint hogy a `df -k` parancs kimenete alapján értesítjük az érdekelteket a kialakult helyzetről, az időzítést pedig most is a `cron` segítségével végezzük. A kérdéses program a `monitor_disk.sh`, amelynek első, igazán lényeges részlete feltérképezi, hogy milyen fájlrendszerek vannak rendszerünkben:

```
FILESYSTEMS=$(mount | grep -iv
➥proc |\
grep -iv cdrom | awk'{print
➥$3}')
```

Természetesen kihagyjuk a dologból a */proc* fájlrendszert, és arra is ügyelünk, hogy a ***CD-ROM*** se kerüljön bele a statisztikába, ha esetleg valamelyik kolléga berakott volna egy lemezt. A százalékban kifejezett lemeztelítettségeket (`Use%`) a szkript két értékkel, `THRESHOLD_MAX`-szal és `THRESHOLD_WARN`-nal hasonlítja össze.



2. ábra A monitor_stats.pl akció közben

Ha bármelyiket túllépte valamelyik fájlrendszer, a program összeállít egy levelet, és elküldi a megadott címre. Figyeljük meg, hogy az itt látható kódsorral mindenütt gondoskodtam róla, hogy a százalékos foglaltságot a program egész számként kezelje:

```
typeset -i  UTILIZED=$(df -k
➥$FS | tail -1 | \
awk '{print $5}' | cut -d"%"
➥-f1)
```

A generált üzeneteknek felállítottunk egy levelezési listát, amelynek tagja minden kolléga, illetve az a külső személy is, akit probléma esetén riasztani kell. A dolognak ez a részét természetesen máshogy is meg lehet oldani.

Telepítve van a rendszerem ez a Perl modul?

Ha ellenőrizni akarjuk, hogy egy adott Perl modul telepítve van-e saját rendszerünkön, adjuk ki a következő parancsot:

```
$ perl -e "use Net::SMTP";
```

Ha semmi nem jelenik meg, akkor minden a legngyobb rendben, a modul telepítve van. Ha viszont hiányzik, akkor ilyesféle hibaüzenetet kapunk:

```
$perl -e "use Net::OTHER";
Can't locate Net/OTHER.pm in @INC (@INC contains:
/usr/lib/perl5/5.8.3/i386-linux-thread-multi /usr/lib/perl5/5.8.3
/usr/lib/perl5/vendor_perl/5.8.0 /usr/lib/perl5/vendor_perl .) at -e line
1.
BEGIN failed-compilation aborted at -e line 1.
```

Küldhetünk például értesítést egy adott csoport valamennyi tagjának, illetve használhatnak a kollégák személyhívót is.
Mivel az általunk használt fájlrendszerek viszonylag nagyok (72-140 GB körüliek), a kritikus értesítési szintet 95 százalékos telítettségre állítottam, hogy adott esetben legyen még időnk beavatkozni. A `THRESHOLD_MAX` és `THRESHOLD_WARN` változóknak természetesen ettől eltérő értékeket is adhatunk, saját igényeinknek megfelelően. Nálunk tovább árnyalja a helyzetet, hogy adatbázis kiszolgálóink is rendszeresen futtatnak olyan folyamatokat, amelyek viszonylag sok lemezterületet igényelnek, illetve tetemes mennyiségű naplóbejegyzést generálnak. Ennek megfelelően a megfigyelési időközt 15 percre állítottam. Ez a dolog természetesen megint hely- és alkalmazásfüggő, hiszen egy kevésbé terhelt rendszeren az egy órás időköz is tökéletesen megfelelő lehet.
A harmadik szkript (`monitor_smtp.sh`) a levélkiszolgálók levélküldési képességét vizsgálja. Szerkezetét tekintve rendkívül hasonló a két mér ismertetett programhoz, hiszen semmi egyebet nem tesz, mint ellenőrzi, hogy képes-e kapcsolódni egy megadott *SMTP* kiszolgálóhoz, és el tud-e küldeni a segítségével egy levelet. A kód tehát a szokásos módon végigmegy a felhasználó által beállított listán, és minden levélkiszolgálónak elküld egy tesztlevelet.
Ez az a pont, ahol belép a képbe az `smtp.pl` szkript. Ez egy *Perl* szkript *(1. Lista)*, amely a *NET::SMTP* modul segítségével küld levelet egy *SMTP* címre. A legtöbb mai *Linux* terjesztés amúgy eleve tartalmazza ezt a modult. Attól függően, hogy az `smtp.pl` sikeresen lefutott-e, a monitorozó szkript frissíti a naplót tartalmazó weblapot. Itt most persze nem küldünk levelet akkor sem, ha gond van, hiszen – érthető módon – az nem biztos hogy célba érne. Ez a program tehát nem is annyira megfigyelésre, inkább hibakeresésre való. Tervezem egy olyan, fejlettebb változat megírását is, amely hiba esetén egy másik, biztosan működő *SMTP* kapcsolaton keresztül mégis kiküld egy értesítést.
Végül elérkeztünk a `monitor_stats.pl` szkripthez. Ez a program minden gazdagépre bejelentkezik, és végrehajtja a következő parancsokat:

1. Lista Az smtp.pl a kimenő levelek kézbeeíthetőségét ellenőrzi az SMTP kiszolgáló lekérdezésével

```
#!/usr/bin/perl -w
#
# Title  :   smtp.pl
# Author :   John_Ouellette@yahoo.com
# Files  :   smtp.pl
# Pupose :   Send email through SMTP server
#            Called from monitor_smtp.sh
#
# Submit as
use Net::SMTP;
my $rcpt   = $ARGV[2] || 'mygroup@somewhere';
my $sender = $ARGV[1] || 'root@host01';
my $host   = $ARGV[0];
#Start Script
my $smtp =Net::SMTP->new($host, Debug => 1);
my $input="test msg for server $host";
$smtp->mail("$sender");
$smtp->to("$rcpt");
$smtp->data();
$smtp->datasend("To: $rcpt\n") ;
$smtp->datasend("From: $sender\n") ;
$smtp->datasend("Subject: $host test\n") ;
$smtp->datasend("$input");
$smtp->dataend();
$smtp->quit;
```

```
df -k
swapon -s
top -n 1 | head -n 20
hostname
uptime
```

Az eredményeket egyrészt megjeleníti egy böngészőben *(2. ábra)*, másrészt dátum szerint sorba rakva beírja egy naplóba. Ez a program tehát amolyan központi műszerfalként funkcionál, amely lehetővé teszi, hogy villámgyorsan információt szerezzünk a számunkra érdekes gépekről.
A rendszermegfigyelés ilyen kialakításának rögtön három haszna is van:

1. Teljes áttekintésünk van a processzorok, a lemezek és a csereterületek használatáról, így könnyen kiszúrhatjuk, ha valahol hiba van.
2. Könnyebben férünk hozzá egy adott kiszolgáló adataihoz. Így egyrészt nem kell hosszas lekérdezéseket fogalmazgatni, ha éppen kutatunk valami után, másrészt eleve könnyebben buknak ki a hibák, ami csökkenti annak az esélyét, hogy az éjszaka kellős közepén kapjunk egy kellemetlen telefont vagy riasztást.
3. A naplókat a vállalat vezetősége is figyelemmel kísérheti, így pontosan láthatják, mennyi jók vagyunk.

Összefoglalás

A cikkeben vázolt egyszerű rendszermegfigyelő szoftvernek köszönhetően az általam vezetett csoport termelékenysége jelentősen megnőtt, másrészt az emberek végre hinni kezdtek a megfigyelt eredményekben, mert nem kell az idejüket a téves riasztásokra vesztegetniük. A telepítménnyel kapcsolatos problémák kiszűrésében is nagy segítségnek bizonyult az, hogy szükség esetén a rendszer teljes története rendelkezésre áll. Végezetül a bemutatott programok egyszerű telepíthetősége a kevésbé képzett kollégáknak is lehetővé teszi, hogy akár egyetlen munkanap alatt beletanuljanak a rendszerfelügyeletbe.

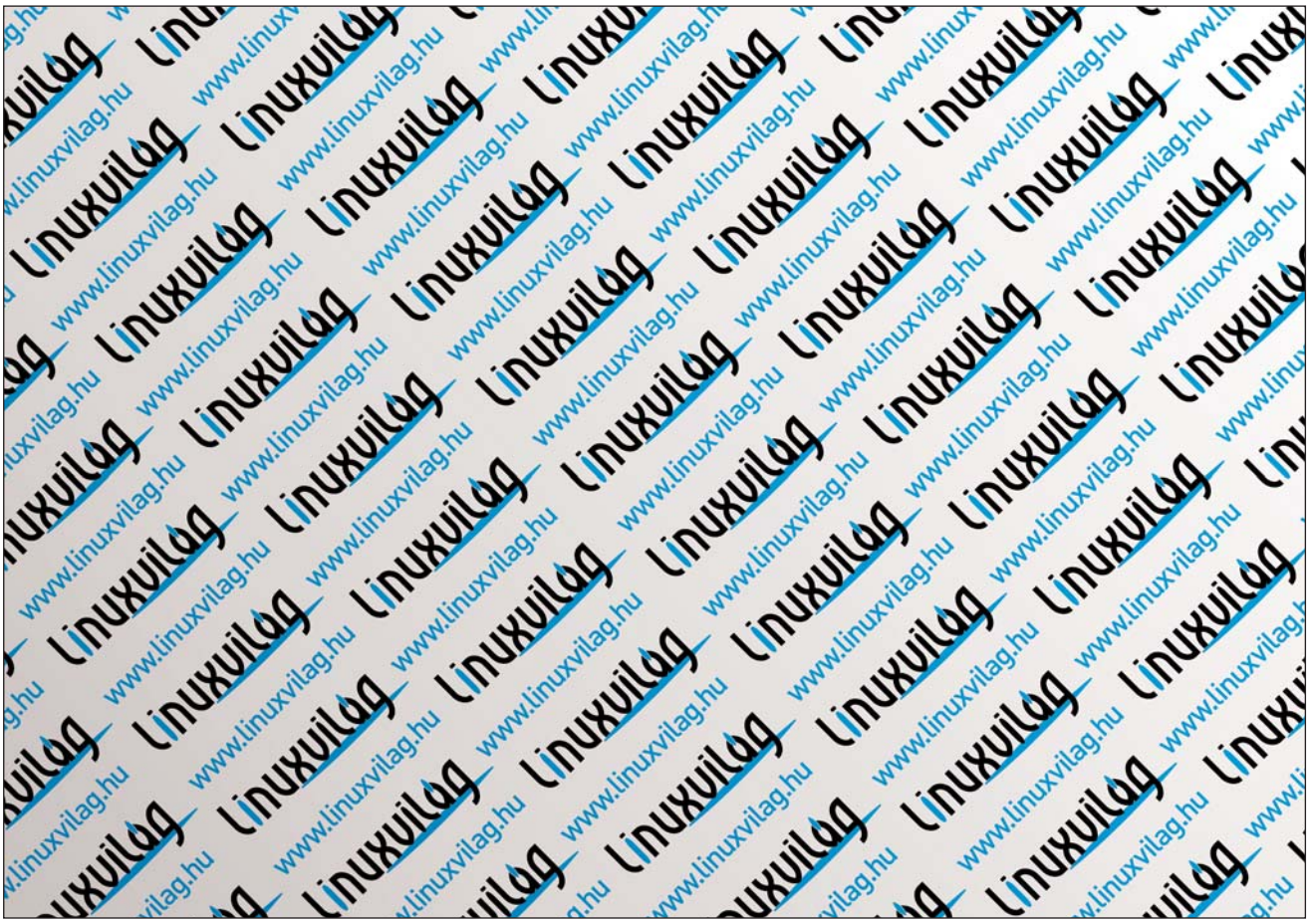
Linux Journal 2005. 137. szám

John Ouellette

Rendszergazdaként dolgozik. Kilenc év tapasztalata van Microsoft Windows NT és UNIX rendszerekkel kapcsolatban. Hiszi, hogy a parancssor nagyszerű dolog.
Johnt a john_ouellette@yahoo.com címen érhetjük el.

KAPCSOLÓDÓ CÍMEK

- ftp.ssc.com/pub/lj/listings/issue137/8216.tgz
- www.gnu.org/software/wget/wget.html
- www.gnu.org/software/bash/manual/bashref.html
- search.cpan.org/~gbarr/libnet-1.19/Net/SMTP.pm
- perl.about.com/b/a/007039.htm



A virtuális memória figyelése a vmstat segítségével

Ha a rendszerünk viszonylag sok csereterületet használ, az nem föltétlen jelenti azt, hogy több memóriára van szükségünk. Ez a cikk arról szól, miként dönthetjük el, hogy Linux rendszerünk jól érzi-e magát a rendelkezésre álló memóriával, vagy indulnunk kell a boltba...

Különösen a kezdő Linux felhasználók a virtuális memóriát gyakran gondolják valami misztikus dolognak, holott ha szánunk némi időt az alapkoncepciók megértésére, az egész nem is tűnik annyira bonyolultnak. Ezzel a tudással felvértezve aztán a vmstat segítségével megfigyelhetjük, hogyan kezeli rendszerünk a virtuális memóriát, és fényt deríthetünk azokra a problémákra, amelyek a rendszer teljesítményét visszafogják.

Hogyan működik a virtuális memória

A fizikai memória, vagyis az a memóriamennyiség, ami ténylegesen a gépünkben található nyilván véges erőforrás. A *Linux* memóriakezelője azonban egy ügyes kezelési technikával képes rá, hogy kissé kitolja a határt. Ez a rendszerszolgáltatás ugyanis időről időre felszabadítja a fizikai memória azon részeit, amelyekre a rendszer és a programok működéséhez aktuálisan nincs szükség.
Természetesen valamennyi futtatott folyamat használ valamennyi memóriát, de szinte egyetlen folyamat sem használja egyszerre az össze, általa lefoglalt területet. Ahhoz, hogy ebből a felismerésből tőkét kovácsoljunk, lennie kell egy olyan mechanizmusnak, amely kiírja lemezre a nem használt területeket, majd szükség esetén vissza is olvassa azokat. Ezt a memóriakezelést a rendszermag végzi, mégpedig a ***lapozás (paging)*** és ***csere (swapping)*** segítségével. Lapozásról akkor beszélünk, ha a rendszer egy folyamathoz tartozó memóriaterületnek csak egy részét írja ki lemezre. Ezt a műveletet szigorú értelemben csak akkor nevezzük cserének (swapping), ha az a folyamat teljes memóriaterületét érinti. ***Linux*** alatt az ilyen igazi swappelés tulajdonképpen nagyon ritka esemény. Ugyanakkor a két kifejezést a gyakorlatban általában egymás szinonimájaként használják.
Ha egy memórialap kikerül lemezre ***kilapozásról (page-out)***, ha a rendszer visszaolvassa azt a fizikai memóriába, akkor ***belapozásról (page-in)*** beszélünk. ***Laphiba (page fault)*** akkor történik, ha a kernel felfedezi, hogy egy folyamatnak szüksége lenne egy memórialapra, de az ki van lapozva, tehát előbb vissza kell tölteni.
A belapozás teljesen normális, gyakori esemény egy operációs rendszer életében, és nincs vele semmi gond. Amikor például egy alkalmazás elindul, a teljes végrehajtható kódja, és az összes adata belapozódik. Ez így van rendjén, és teljesen normális működésmódot jelent.
A kilapozások felbukkanása ugyanakkor már valamilyen problémát jelezhet. Amikor a kernel észreveszi, hogy kezd kifutni a fizikai memóriából, megpróbál területeket felszabadítani, vagy kilapozza azokat. Bár ez egy jól megtervezett és helyesen üzemeltetett rendszerben is meg-megesik néha, fontos, hogy nem legyen túl hosszú. Ha ugyanis a kilapozások általánossá válnak, elérhetünk egy pontot, amikor a rendszer több időt tölt a memória menedzselésével, mint a folyamatok végrehajtásával (***trashing***).
Az, ha rendszerünk elkezdi használni a csereterületet, önmagában még nem rossz. A dolog gyakorisága a lényeges, vagyis baj akkor van, ha ez túl intenzíven történik.
Ha mondjuk a rendszerünkön futó leginkább memóriaigényes alkalmazás éppen várakozik, nincs abban semmi rossz, ha a kernel átmenetileg lemezre írja az általa lefoglalt lapokat, vagy azok egy részét. Mi több, a várakozó folyamattok fizikai memóriából való kiebrudalása kifejezetten hasznos, hiszen a kernel így nagyobb területet tud lemezpufferként használni.

A vmstat használata

A ***vmstat*** – amint azt neve is sugallja – a virtuális memória használatáról tud különféle statisztikákat, kimutatásokat készíteni. Megmutatja, mennyi virtuális memóriánk van összesen, ebből mennyi szabad, illetve összefoglalja a lapkezeléssel kapcsolatos eseményeket. A legfontosabb azonban az, hogy segítségével a be- és kilapozásokat már megtörténésük pillanatában érzékelni tudjuk. Ez pedig elképesztően hasznos képesség.

1. kód

```
 procs                  memory    swap        io     system cpu
r  b  w   swpd   free  buff  cache  si so   bi  bo   in    cs us sy  id
0  0  0  29232 116972  4524 244900   0  0    0   0    0     0  0  0   0
0  0  0  29232 116972  4524 244900   0  0    0   0 2560     6  0  1  99
0  0  0  29232 116972  4524 244900   0  0    0   0 2574    10  0  2  98
```

Rendszerünk virtuális memóriával kapcsolatos tevékenységét azonban legjobban egy kis késleltetés beiktatásával tudjuk a ***vmstat*** segítségével megfigyelni. A késleltetés azt adja meg, hogy a ***vmstat*** hány másodpercenként frissítse a megjelenített adatokat. Ha nem adjuk meg ezt a paramétert, akkor a program az utolsó rendszerindítás és leállás közti átlagértékeket adja meg. A legmegfelelőbb általában az öt másodperces késleltetés. Ehhez a következő parancsot kell kiadnunk:

```
vmstat 5
```

Megadhatunk ismétlésszámos is, vagyis hogy hány frissítést szeretnénk látni, mielőtt a ***vmstat*** kilép. Ha nem adjuk meg ezt az értéket, a program a végtelenségig működik, de a ***Ctrl-C***-t leütve természetesen kilép.

Ha tehát öt másodperces késleltetéssel tíz frissítést szeretnénk látni, a következő parancsot használjuk:

```
vmstat 5 10
```

Ennek hatására valami ilyesmit fogunk látni *(1. kód)*.
A ***vmstat*** súgóoldala természetesen tartalmazza valamennyi mező leírását. Számunkra most a legfontosabb három a `free`, a `si` és a `so`. A `free` oszlop a jelenleg szabad virtuális memória nagyságát mutatja, a `si` a belapozások, míg a `so` a kilapozások száma. Példánkban a so oszlop végig nulla, vagyis a rendszerben nem történik kilapozás.
Ami a rövidítéseket illeti a `po` és `pi` pontosabb lenne ugyan, de történeti okok miatt a `so` és `si` honosodott meg. Most lássunk egy példát arra, amikor egy rendszerben valódi lapozás folyik *(2. kód)*.

2. kód

```
 procs                  memory    swap        io     system cpu
r  b  w   swpd   free  buff  cache  si  so   bi   bo   in    cs us  sy  id
. . .
1  0  0  13344   1444  1308 19692   0 168  129   42 1505   713 20  11  69
1  0  0  13856   1640  1308 18524  64 516  379  129 4341   646 24  34  42
3  0  0  13856   1084  1308 18316  56  64   14    0  320  1022 84   9   8
```

3. kód

```
14:23:19 up 348 days,  3:02,  1 user,  load average: 0.00, 0.00, 0.00
55 processes: 54 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:   0.0% user,   2.4% system,   0.0% nice,  97.6% idle
Mem:    481076K total,   367508K used,   113568K free,     4712K buffers
Swap:  1004052K total,    29852K used,   974200K free,   244396K cached
```

A nullától különböző értékek a `so` oszlopban egyértelműen jelzik, hogy kevés a fizikai memória, és a kernel „dolgozik az ügyön”. Ilyenkor a `top` vagy a `ps` paranccsal deríthetjük ki, hogy melyek a legnagyobb memóriaigényű folyamatok.
A memória és a csereterület használatával kapcsolatos statisztikát a top parancs is megjelenít. Íme egy példa *(3. kód)*.A ***top*** súgóoldalán természetesen megint bőven találunk információt ezekről a mezőkről.

Összefoglalás

Nem feltétlenül rossz, a ha rendszerünk időnként használja a csereterületet. Ugyanakkor ha a vizsgálat során az derül ki, hogy a kernel rendszeresen kifut a fizikai memóriából, a lapozás pedig jelentősen csökkenti a rendszer teljesítményét, akkor kénytelenek leszünk bővíteni. Ha ezt valamiért nem engedhetjük meg magunknak, akkor próbáljuk meg eltérő időben futtatni az erősen memóriaigényes alkalmazásokat, illetve próbáljuk elkerülni, hogy velük egyidőben olyan folyamatok is fussanak, amelyekre nincs föltétlen szükség. Ha pedig több gépünk is van, próbáljuk meg elosztani köztük a feladatokat.

Linux Journal 2005. 140. szám

Brian K. Tanaka

1994 óta dolgozik rendszergazdaként olyan cégeknek, mint az SGI, az Intuit és a RealNetworks. Társalapítója a Martinage-Oak LLC-nek. A btanaka@martingale-oak.com címen érhető el.

KAPCSOLÓDÓ CÍMEK

- www.csn.ul.ie/~mel/projects/vm/guide/pdf/understand.pdf
- www.phptr.com/bookstore/product.asp?isbn=0131453483&rl=1#
- sourcefrog.net/weblog/software/linux-kernel/swap.html


Linuxvilág
www.linuxvilag.hu

C Scripting Language avagy a lusta C programozók öröme

A CSL egy a C nyelv után gyorsan megtanulható szkriptnyelv. Egy informatikai rendszerben sokszor adódnak olyan programozással is megoldható feladatok, melyekhez az ember nem szívesen tölt órákat egy hagyományos programozási nyelven kódolva a gép előtt. A szkriptnyelvek csekély kompromisszumok árán segíthetnek ilyenkor – már ha egyszer megbarátkoztunk velük. Akiknek ez még nem sikerült teljes mértékben, azoknak ajánlom a CSL-t, mely csak néhány lépésnyire található a C/C++ stílusú nyelvektől, de azoknál jóval engedékenyebb a programozóval szemben.

A programozási nyelvek közül egyre nagyobb népszerűségnek örvendenek az ún. szkript-nyelvek, de leginkább csak néhány divatossá vált képviselőjüket ismeri a nagyközönség. Titkuk talán – a programozók egészséges lustaságán kívül – abban rejlik, hogy gyorsan és könnyen lehet velük meghatározott kérdésekre frappáns választ adni. Rugalmasságuk ugyanakkor korlátozottabb az úgynevezett compiler típusú (lefordítás után futtatható kódot generáló), általános célú nyelveknél, inkább egy adott környezethez igazodva, rész- vagy célfeladatok megoldásában jeleskednek.

A ***CSL*** nyelvet – mely a ***GNU*** licensz szerint szabadon letölthető és használható a ➲ ***http://csl.sourceforge.net*** címről – elsősorban azok érezhetik szívükhöz közelállóknak, akik már rendelkeznek némi gyakorlattal a ***C/C++*** vagy ***Java*** szintaxisához hasonló nyelvekben, ugyanakkor nem szeretnének egyszerű feladatok megoldására túl sok időt fordítani. E nyelvet használva nem kell azon töprengeniük, vajon szükség van-e explicit típuskonverziókra, meg kell-e semmisíteniük egy használaton kívüli objektumot, hogyan foglaljanak memóriát egy dinamikus tömbnek, és nem okoz fejtörést a kiútkeresés a mutatók erdejéből sem. Utóbbi a szerző állítása szerint nincs is a nyelv kelléktárában – ezzel azért ne értsünk egyet azonnal.

Pingvinek alatt is működő öszvérmegoldás

Ugyanakkor a szkript jellegű nyelvekhez képest jobban ragaszkodik a „klasszikus” programozási szintaktikákhoz, fogásokhoz. Kevésbé szokatlan első látásra a ***C*** vagy ***Pascal*** nyelveken felcseperedett programozóknak, mint az objektumorientáltságban is jeleskedő ***Python*** vagy az elegáns héjprogramok (shellszkriptek); képes kihasználni a reguláris kifejezések erejét – noha nem olyan mértékben, mint például a ***Perl*** – , s az adatbázisokhoz kapcsolódás is megoldható segítségével, még ha ebben tudása nem is fogható ***PHP***-hoz.

Sajnos svájci szerzője 2002-ben abbahagyta a projekt továbbfejlesztését, és a honlapról elérhető fórumon sem tapasztalhatunk újabb aktivitást, viszont eszközkészlete átlátható, és rugalmassága elegendő a legtöbb feladat megoldásához.

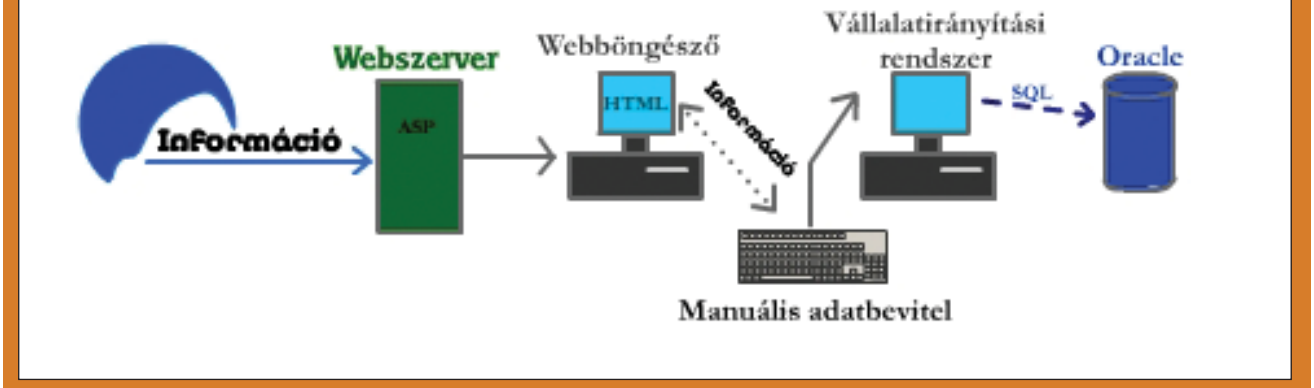
Általános célú programozási nyelv voltát hangsúlyozza a legfontosabb célterületeket lefedő eszközkészlete: sztring- és fájlkezelés, kommunikációs portok, reguláris kifejezések, matematikai műveletek, regisztrációs adatbázist kezelő valamint rendszerfüggvények, és adatbázis-támogatás egyaránt található függvényeinek sorában. Ráadásul ezek többsége egyaránt elérhető a ***Linux/BSD/Unix***, az ***OS/2*** és ***Windows*** környezetekben, hiszen a ***CSL*** ezen platformokon fut. Ráadásul képes egy lefordított típusú nyelvbe makróként beágyazódva is működni; a legfontosabb ***C/C++*** fordítókhoz ***API*** illetve osztály szintű kapcsolódási felületet nyújtva.

Szerezzük meg és használjuk

Telepítése egyszerű és jól dokumentált. A letöltött tömörített fájlt a szokásos módon használhatjuk:

```
tar xzf <fájlnév>
cd csl_verzió_platform
tar -C /usr/local -xf files.tar
ldconfig
```

Ügyeljünk arra, hogy root felhasználói jogokkal kell rendelkeznünk mindezekhez, illetve ha nem az ajánlott ***/usr*** vagy ***user/local*** könyvtárak alá telepítünk, akkor az elérési útvonalat is be kell állítanunk rendszerünk számára.



1. ábra Az információáramlás útja CSL szkript nélkül

Önálló szkripteket így futtathatunk vele:

```
csl  script_neve.csl  [paraméterek_ha_vannak]
```

Kezdő lépéseinket mi is egy *„Helló világ!”* programmal tegyük meg.

```
#loadLibrary 'ZcSysLib'
main()
{
  sysLog("Helló világ!");
}
```

Látható, hogy szintaktikáját nagymértékben a *C* nyelvtől örökölte; az `#include<függvény.h>` szerepét itt általában a fentebb látható függvényhívás tölti be, valamint az elmaradhatatlan `main(){…}` főfüggvény is ismerős. Megjegyzés: ahhoz, hogy a beépítendő fejlécfájlokat használni tudjuk, a `CSLPATH` környezeti változóban a */share/csl* könyvtár elérési útvonalát be kell állítani.

Alapelemek és használatuk

Változók terén viszont inkább az egyszerűsödő, gyorsabb kódolást szem előtt tartó nyelveket idézi: egyedüli változótípusa a `var`, ez tárolhat karakterláncot és számot egyaránt. Ebből következően értelmét vesztené, s ezért nincs is struktúra típus, valamint mutatókkal sem találkozhatunk a nyelvben.
Itt azonban egy kis pontosítást tehetünk, mert függvények hívása esetén a nyelv dokumentációja is kitér arra a tényre, hogy a *CSL* támogatja és javasolja is a cím szerinti értékátadást az & operátor segítségével (`fuggveny(var& x, var& y[])`), sőt, tömbök esetén ez még nyilvánvalóbban látszik. Nem csoda, ha ilyenkor a *C* nyelvben közismerten kezdő tárhelycím szerint, mutatók (pointerek) segítségével is elérhető tömb képe sejlik fel az olvasóban.
Még pontosabban a *C++* használatakor is sűrűn igénybe vett referencia típusú paraméter-átadás történik (a függvény meghívásakor változókat és nem mutatókat adunk meg, míg a függvénydefinícióban az & operátorral tudatjuk, hogy nem a változó értékét, hanem címét használjuk), mely által egyszerűen megváltoztatható a függvényblokkban az adott változó értéke, és nem bonyolódunk bele a mutatókba.

A *C++*-tól eltérően nem a függvénytörzsön belül kell definiálni a `static` kulcsszóval bevezetett globális változókat, és a külső deklarációknak csak akkor van igazán értelme, ha segítségével futási időben – és nem fordításkor – töltünk be változókat.
Amennyiben egy lokális tömböt inicializálunk, úgy az dinamikusan foglal magának tárhelyet (itt érezhető az interpreter jellegű nyelv előnye), míg globális esetben a szokásos statikus helyfoglalás történik. Ezért lehet érvényes a következő kódrészlet *CSL* nyelven, míg *C/C++*-ban nem:

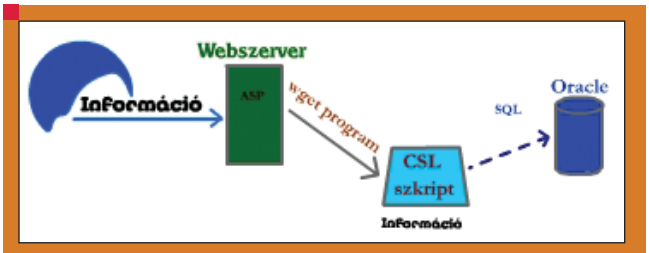
```
var x = 100;
  var tomb[++x];       // a tomb[] 101 elemet
                       // tud tárolni!
  var  masiktomb[x*2]  // a masiktomb[] 202
                       // elemet tárol
```

A dinamikus helyfoglalás előnyét szemlélteti a következő kódrészlet is, ahol a tömb méretét a `resize` utasítással növeljük meg, így már 5 sort tudunk feltöltetni kedvenc betűhármasainkkal:

```
var tomb = {
{'a', 'b', 'c'},
{'d', 'e', 'f''}
}; // eddig a tomb 2 sornyi betűhármast tudott
   // tárolni
resize tomb[5][3]; // ezzel a művelettel már 5
                   // sornyit - dinamikusan nőtt!
```

Való világ

Ennyi bevezető után nézzünk egy életből vett példát!
Cégünk vállalatirányítási rendszere *Oracle* alapokon nyugszik, s egy mezőben képes tárolni a különböző nemzetközi fizetőeszközök és a forint közti váltószámot. Ezt az információt később a kimenő nemzetközi számláknál használjuk, automatikusan átváltva az összeget például euróra, cserébe viszont a váltószámokat valakinek napi rendszerességgel frissítenie kell.
A konkrét feladat röviden tehát így hangzik: tudjuk meg a napi valutaárfolyamot, és „mondjuk meg” egy *Oracle* adatbázis-kezelőnek.



2. ábra CSL szkript segítségével rövidebb az út

Mivel az ***MNB*** honlapján megtalálhatóak a szükséges árfolyamok, a ***CSL*** és egy kis külső segítség igénybevételével a folyamat automatizálható. Nézzük, hogyan tehetjük ezt meg. A példában néha a kitekintés kedvéért lemondok az optimális megoldásról.
Betöltjük a használni kívánt függvényeket:

```
#loadLibrary 'ZcSysLib' //Kiiratásokhoz.
#loadLibrary 'ZcStrLib' //Sztringekhez.
#loadLibrary 'ZcRexLib' //Reguláris
                        //kifejezésekhez.
#loadLibrary 'ZcFileLb' //File műveletekhez.
#loadLibrary 'ZcDaxLib' //Adatbáziskapcsolathoz.
```

Tudjuk, hogy mi a pontos neve annak a ***HTML*** oldalnak, ahol a szükséges információ található. Ezért a linuxos környezetben méltán népszerű ***wget*** program segítségéhez fordulunk a letöltésben – egész egyszerűen elindítjuk kis programunkból a letöltésvezérlőt. A függvény így néz ki:

```
letoltes()
{
  //Ha már létezik a fájl, akkor töröljük, mert
  //különben a szemfüles wget más nevet adna neki.
fileDelete('engine.aspx@page=napiarfolyamok');
  //Letöltjük a html fájlt a (külső) wget program
segítségével:
sysCommand('wget www.mnb.hu/engine.aspx?page=
➥napiarfolyamok');
}
```

Íme egy részlet a letöltött fájlból elrettentésül (ebből kell kinyernünk a napi árfolyamokra vonatkozó információt):

```
<span class="MNB_Heading4">EUR</span></td><td
➥class="MNB_DgBorder" align="Right"><span
➥class="MNB_Heading5">1</span></td><td
➥class="MNB_DgBorder" align="Right"><span
➥class="MNB_Heading5">245,70 </span></td>
```

Könnyű dolgunk lenne, ha például az ***awk*** nyelvet használnánk, hiszen csak azt kellene megmondani neki, hogy mezőhatárolónak melyik ***HTML*** tag-et tekintse, s ugrálhatnánk a mezők között. Mi azonban inkább a mintaillesztést hívtuk segítségül, s a ***C*** nyelvhez hasonlóan fájlból olvasunk be változóba:

```
var mintakereses()
{
  const mit_keres = '\>EUR\<';
  const arfolyam = '[0-9]\{3},[0-9]\{2}';
  ➥//000,00 alakú tizedestörtet keres
  var minta = rexOpen(mit_keres);
  var ar = rexOpen(arfolyam, rexOpenExtended);
  ➥//okosabb regexp
  var talalat[5][2];   //>EUR< minta
                       //megtalálásakor kell.
  var artalalat[3][2]; //Eur/Ft árfolyamát ha
                       //megtaláljuk, ebbe kerül.
```

A reguláris kifejezések a `rexOpen()`, `rexMatch()` és `rexClose()` utasítások segítségével használhatók. Tudjuk, hogy a keresett pénznem `<html_tag>EUR<html_tag>` alakban található a fájlban, s azt is, hogy az árfolyam `000,00` alakú tizedestört. Utóbbira a mintaillesztés csakis olyan számokat keres, melyek egy hármas csoport, majd vessző – ezt jelzi a `[0-9]\{3}` a programkódban – , végül egy kettes csoport szám – ezt pedig a `[0-9]\{2}` – alakjában fordulnak elő. Az `EUR` szót követő első ilyen előfordulás lesz a keresett árfolyam.
Folytassuk a fájl megnyitásával:

```
//Fájl megnyitása.
var megnyit = fileOpen('engine.aspx@page=
➥napiarfolyamok', fileOpenRead
+fileOpenOld);➥//a fileOpenOld módosító azt
jelzi, hogy elvárjuk, létezzen a fájl
//Ellenőrizzük, hogy nem volt-e gond a fájl
➥megnyitásával (ill. tényleg létezik-e):
  if (!megnyit)  //hibakezelés - ezt rábízhatjuk
                 //így egy C++ programra
  {
    const hiba[3] =
    {
'Nyitasi hiba!', 'Hibas, vagy nem letezo fajl.',
fileInfo('engine.aspx@page=napiarfolyamok')
  };
  throw hiba;  //throw itt, try és catch C++
               //oldalon
  }
  //Hibakezelés vége
```

Kivételek márpedig vannak…

A hibakezelést itt most a ***CSL*** kommunikációkészségének bemutatása kedvéért nem „helyben” végezzük, hanem átadjuk azt egy képzeletbeli ***C++*** programrészletnek, mely körülbelül így nézhetne ki:

```
void Kivetelkezelo()
{
  try {
  ZCsl csl;
  csl.loadScript("programom.csl");
  ZString ret = csl.call("kivetelkezelo.sh");
  } // try vége, kivételek figyelése.
  catch (const ZException& err) {
  cerr << err.text;
  } // catch vége
} // Kivetelkezelo vege.
```

Természetesen ugyanez fordítva is lehetséges, egy *C/C++* program által „dobott” kivételt „el tud kapni” a CSL program is. Akik eddig jobbára csak C nyelvet használtak, valószínűleg `if() then() else()` szerkezetek segítségével próbálnák meg lekezelni az esetleges hibákat. A *CSL* ehhez képest a *C++/Java*-ból is ismerős, kulturáltabb és nagyobb programokban is hatékony módszert használja: a kritikus részeket „próbáló” programblokkokban (`try{}` blokkok) „dobunk” egy hibajelzést (`throw()` függvény), ha bajt észlelünk, és ezeket a programblokkot követően akár az összes kritikus rész hibaüzenet dobására figyelve „elkapjuk” (`catch()` függvény). Rövid programunkat azonban most nem tűzdeljük tele – az egyébként erősen ajánlott – kivételkezelésekkel.

Aki eurót keres, >EUR<-t talál

Folytassuk a sort a keresett `>EUR<` mintával:

```
for (var i=1; !fileEof(megnyit) ; i++)
➥//fájl beolvasása kereséshez
  {
  var sor[i] = fileReadLine(megnyit);
  var letezik = rexMatch(minta, sor, 1, talalat);
➥//megtalálható-e a minta
```

A `rexMatch` mintaillesztést végez, a mintát a sztringben keresi (egy előfordulás), s ha talált valamit, azt a `talalt[][]` tömb segítségével tudjuk lokalizálni: megkapjuk, hogy hányadik karakterpozícióban kezdődik a keresett minta, és azt is, hogy milyen hosszúságú:

```
if (letezik)
  {
  sysLog('A keresett ' + mit_keres + ' mintazat
  ➥a fajlban a(z)');
  sysLog(i + '. sorban, a ' + talalat[0][0] + '.
  ➥karakterpoziciotol kezdodik.');
```

Így könnyedén leszűkíthetjük a keresést, hiszen a fájl elejére nem vagyunk már kíváncsiak, csak a mintától a végéig érdekes számunkra:

```
var szukebb = strSubString(sor,talalat[0][0]);
➥//sztringnek az >EUR< -tól kezdődő részét
➥tároljuk
var letezik_ar = rexMatch(ar, szukebb, 1,
➥artalalat);  //ha talál 000,00 alakú  részt
```

Ha megtaláltuk a keresett mintára illeszkedő részt, akkor kiíratjuk:

```
if (letezik_ar)
{
var mettol = artalalat[0][0]; //hányadik
➥karakterpozíciótól kezdődik az ár
var meddig = artalalat[0][1]; /hányadik
➥karakterpozícióig tart az ár
sysLog('\t' + mit_keres + ' arfolyama: ' +
➥strSubString(szukebb,mettol,meddig));
return (strSubString(szukebb,mettol,meddig));
```

```
➥//fájl elejével nem foglalkozunk már
} //if(letezik_ar) vége
else return(0);
} //if(letezik) vége
} //for ciklus (fájl beolvasáshoz kellett) vége
```

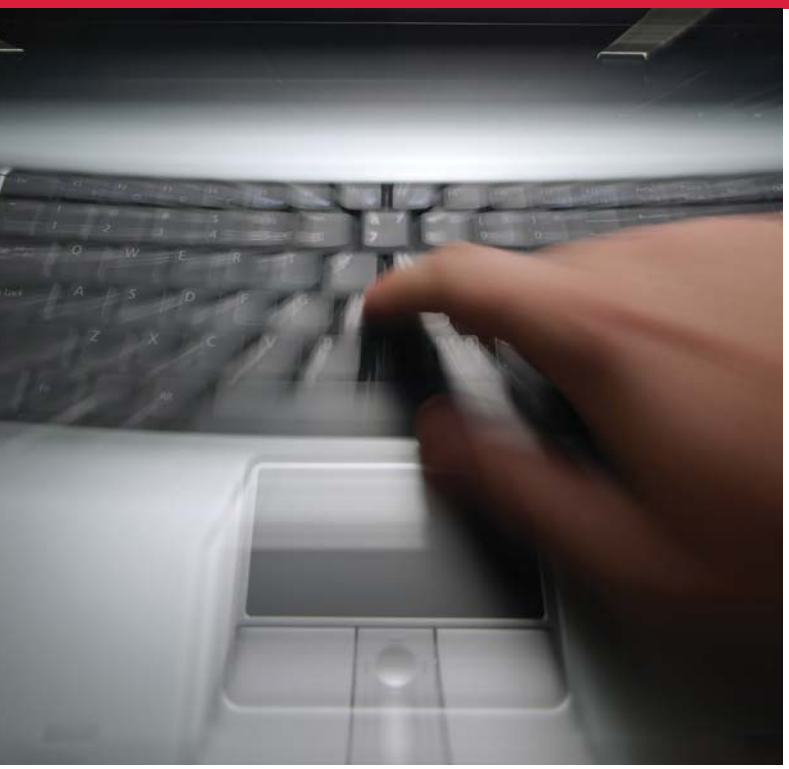
Végül felszabadítjuk a memóriát. Erre azért van szükség, mert az univerzális és gyorsabb használat (több sztringen is végezhetnénk ugyanarra a mintára illeszkedésvizsgálatot) érdekében a mintaillesztőt „lefordítottuk” a `rexOpen` függvény meghívásával.

```
rexClose(minta);       //memória felszabadítása
➥mintaillesztésből
rexClose(ar);  //memória felszabadítása
➥mintaillesztésből
fileClose(megnyit);    //olvasott fájl lezárása
}       //mintakeresés vége
```

Utolsó lépések: adatot az adatbázisba

Az adatbázishoz kapcsolódáshoz a parancssori argumentumokból szerzünk információt, ezért programunkat a következőképp kell elindítani:

```
csl program.csl
➥Felhasznalonev/Jelszo@Adatbazisnev
```



Minden más esetben hibát kapunk, ugyanis ezek feldolgozásában segít minket egy beépített függvény, mely kifejezetten az ilyen alakban elindított, adatbázis-kapcsolatokat is használó programokból képes kivágni a felhasználónevet, s egyúttal a másik két adatot is változókba teheti:

```
var adatbazis, nev, jelszo;
//parancssori argumentumokból "kitaláljuk",
➥melyik a név
nev = strSplitConnectString(mainArgVals[2],
➥jelszo,adatbazis);
```

A csatlakozás előtt át kell alakítani az árfolyamot tartalmazó `arfolyam` változót, hiszen `000,00` alakban kaptuk meg a kívánt adatot, adatbázisunk pedig `000.00` alakban várja tőlünk:

```
arfolyam = strChange(arfolyam,',','.');
➥//Kicseréli a 000,00 alakot 000.00 -ra
```

Majd meghívjuk a kapcsolódást végző függvényünket, átadva neki a szükséges bejelentkezési információkat, valamint a kinyert árfolyamot:

```
kapcsolodas (adatbazis, nev, jelszo, arfolyam);
```

Íme a függvény:

```
kapcsolodas(var& adatbazis, var& nev, var&
➥jelszo, var&arfolyam)
{
var belepes = daxConnect('oracle',adatbazis,
➥nev,jelszo);
```

Leegyszerűsítjük a megoldást azzal, hogy a `kapcsolodas` függvényen belül *SQL* utasítást hajtunk végre, a `belepes` nevű, gyakorlatilag mutató szerepű változóval hivatkozunk az éppen használt adatbázis-kapcsolatra, lekérdezéskor és jóváhagyáskor, valamint lekapcsolódáskor egyaránt:

```
daxSimple(belepes, 'UPDATE PENZUGYEK SET Arfolyam
➥= ' + arfolyam + ' WHERE Penznem LIKE
➥\'EUR%\'');
daxCommit(belepes); //jóváhagyjuk a műveletet
daxDisconnect(belepes); //adatbáziskapcsolat
➥lezárása
} //kapcsolodas függvény vége
```

Mindezt persze a beépített *CSL* függvények segítségével is megoldhattuk volna, hiszen `daxParse()`, `daxCheckCursor()`, `daxFetch()` stb. utasítások segítenek a kurzor pozicionálásában. Itt is fontos lenne a `try-throw-catch` blokk használata kivételkezelés céljából. Aki még nem sokat programozott *Linux* alatt, bizonyára furcsállja az itt is előforduló, látszólag felesleges \ jeleket. Ezek segítségével azt tudatjuk – itt most a *CSL*-lel, de általában a parancsértelmező burokkal -, hogy a \ jelet követő karaktert szó szerint értelmezze, s ne tulajdonítson neki speciális jelentést. Ha ezt elmulasztjuk, az Oracle hibát jelez, mert a programtól nem a kívánt `LIKE 'EUR%'` sztringdarabot kapja meg.
Munkánk eredményét és a *CSL* szkript ténykedését érdemes naplózni, ezt könnyedén meg is tehetjük a main függvényen belül:

```
//Minden tevékenységünket naplózzuk.
  sysDateFormat(sysDateFormatISO); //magyar
                                   //dátumformátum
  sysLogFile( sysDate() + '.log' ); //logfájl
➥neve a mai_dátum.log alakú lesz
```

Tovább bővíthetjük e kis program tudását, ha például a linuxos *cron* segítségével időzítjük a program futtatását, esetleg előtte beágyazzuk egy shellszkriptbe. Így akár komolyabb ellenőrzésekre is lehetőség nyílik, együttműködve például egy *SMTP* szerverrel küldhetünk levelet az illetékeseknek a beimportált árfolyamról naponta, vagy a hibagyanús eseteket (túl nagy vagy túl kicsi érték, kapcsolódás elutasítva stb.) és a naplófájlokat is eljuttathatjuk hozzájuk. Korántsem merítettük ki a nyelv adta lehetőségeket, s kellően elszánt és sok szabadidővel megáldott programozók néhány hatékony szkript és hagyományos program vegyítésével jól működő, komplex feladatok megoldására alkalmas, paraméterezhető szoftvereket hozhatnak létre akár a *CSL* segítségével is. Legfőbb előnyét abban látom, hogy a *C* nyelv ismerete után könnyen tanulható, kényelmesen használható, sokoldalú és hordozható eszköz birtokába kerülünk.

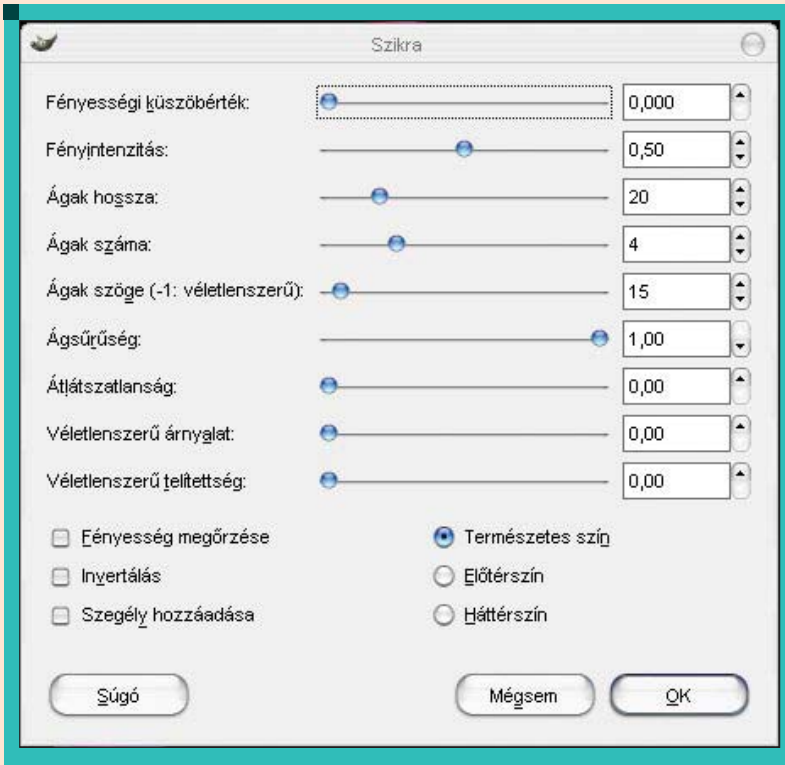
Tóth Virgil Zoltán (m_v@c2.hu)
Szoftverfejlesztő informatikus és rendszergazda, kedvence a Debian disztribúció. Szabadidejét legszívesebben felesége és szépirodalmi regények társaságában tölti. Lenyűgözőnek tartja a Linux rugalmasságát, és a vele dolgozók aktivitását.

Gimp bővítmények (3. rész)

Szikrázó fények

Minden gyakorlott fotós tudja, hogy a fényekre különös figyelmet kell fordítani. Fény olyan a fotós számára, mint a szél a vadásznak. Mindig észben tartandó merről süt és milyen erősen. Az előző részben vetettünk pár pillantást olyan eszközökre, melyekkel utólag korrigálhatjuk a fényviszonyokat a képeinken. Most szemügyre vesszük azt a pár eszközt amit még nem említettünk fényhatások közül, majd pedig pár látványos üveg-effektus következik.

Az első pillantást vessük a ***Szikra szűrőre*** (***Sparkle***), mely a ***Szűrők>Fényhatások>Szikra*** útvonalon található a menüben. Nagy meglepetés nem ér minket, hiszen nevéhez hűen arra való, hogy szikrákat helyezzünk el vele a képen. A szikrák nem akárhol fognak megjelenni, hanem a legvilágosabb részek közelében. Azonban a határértéket, mely alapján a program kiválasztja a világos részeket, mi választhatjuk. Ennek ellenére a legnagyobb nehézséget az okozza, hogy előre megjósoljuk az egyes szikrák helyét. Kisebb előrelátással mégis megoldhatjuk ezt a kérdést, hiszen ahol fehér pontokat festünk a képünkre, ott biztosan szikra fog keletkezni. A feladat egyszerűnek tűnik, mégis a szűrő ablakában rengeteg beállítás fogad minket. Az első amit rögtön érdemes beállítani, az az előbb említett ***Fényességi küszöbérték*** (***Luminosity treshold***), mely meghatározza, hogy a képünk mely részein fognak „kipattanni" a szikrák. Minél magasabb az



1. ábra A Szikra szűrő beállításai

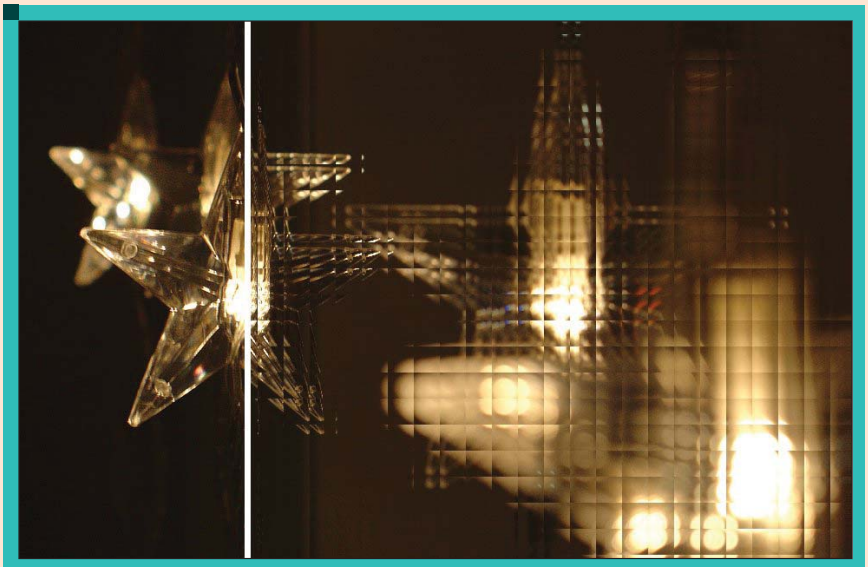
2. ábra Egyszerű trükk

3. ábra Így néz ki egy tipikus szikra

4. ábra Hasonló eredményt kapunk, ha az Ágak szöge (-1)

5. ábra Egy életszerű példa

érték, amit megadunk, annál több világos területet fognak beborítani a szikrák. Érdekes, hogy ha magas küszöbértéket adunk meg, akkor lényegesen több időre van szüksége a szűrőnek a végeredmény megjelenítéséhez.
A hordozható gépemen ez az idő akár tíz, vagy tizenötszörösére is emelkedik, amennyiben egy nyomtatni kívánt nagy felbontású képen dolgozom.
A következő beállítás, a ***Fényintenzitás*** (***Flare intensity***) paraméter változtatásával a szikrák méretét befolyásolhatjuk. Logikusan működik nagyon, hiszen nagyobb érték beállításával, nagyobb átmérőjű és sugarú szikrákhoz jutunk. Érdemes megtartani viszont az alapbeállítást, hiszen az ***Ágak hossza*** (***Spike lenght***) és az ***Ágak száma*** (***Spike points***) csúszka segítségével sokkal kényelmesebben finomhangolhatjuk a végeredményt. Mindkét érték változtatásával a szikrák nagy ágait szabályozhatjuk, köztük még sok kisebb is lesz majd. Az ***Ágak szöge*** (***Spike angle***) paraméterrel megadhatjuk az első sugár vízszinteshez képesti szögét. Ha az értéknek (-1)-et adunk meg, akkor véletlenszerű lesz a szög. Ez akkor hasznos, ha sok szikránk is lesz a képen, de nem szeretnénk hogy teljesen ugyanúgy nézzenek ki. Sokkal karakteresebb szikrát kapunk azonban, ha fix értéket adunk meg, mert akkor a kis sugarak próbálnak hasonló irányba rendeződni mint a nagyok. Ennek egyébként elejét vehetjük, ha az ***Ágsűrűség*** (***Spike density***) paraméternek kis értéket adunk meg.
Vannak opcionálisan választható lehetőségeink is. Ajánlom mindenkinek, hogy próbáljon ki pár tetszőleges variációt. Személy szerint sokat vártam a ***Szegély hozzáadása*** (***Add border***) opciótól, de valahogy nem sikerült vele szép, körvonallal kiemelt szikrát varázsolnom. Kíváncsi vagyok vajon másnak sikerült-e, vagy egy befejezetlen lehetőséggel állunk szemben?



6. ábra A fehér vonaltól jobbra eső részen jól látható a Katedrálüveg szűrő hatása

Néhány egyszerű példa következik. A példaként felhasznált képet a magyar fejlesztésű ***stock.xchng*** (➲ ***sxc.hu***) oldalról választottam. Rengeteg szabadon felhasználható fotót találhatunk az oldalon, sőt mi magunk is megoszthatjuk a jobban sikerült képeinket.
A *2. ábrán* jól látható az a trükk, ahogy a szűrőt rávehetjük, hogy ott hozzon létre szikrát ahol mi szeretnénk. Hozzunk létre egy üres réteget, és fessünk rá egy fehér pöttyöt. Mivel az tényleg a réteg legvilágosabb pontja, így oda szikra fog kerülni.
A következő képen, a *3. ábrán* látható szikrát a következő beállításokkal készítettem: ***Ágak száma*** = 4, ***Ágsűrűség*** = 1. A negyedik ábrához hasonló hatást érünk el, ha véletlenszerűre állítjuk az ***Ágak szögét*** és az ***Ágsűrűségnek*** magas értéket adunk.
Végül pedig itt egy életszerű példa. Az *5. ábra* készítésekor az előbb bemutatott trükköt használva vettem rá a szűrőt, hogy több szikrát is készítsen, mégpedig az általam választott helyeken. A fontosabb beállítások: ***Ágak száma*** = 4, ***Ágak szöge*** = (-1), tehát véletlenszerű, ***Ágsűrűség*** = 1/10.

Üveg-hatások

Nézzünk új kihívás felé! A következzenek az ***Üveg-hatások***, melyeket a ***Szűrők>Üveg-effektusok*** útvonalon találhatunk a menüben. Itt mindössze két szűrő árválkodik. Mindkettő nagyon hasznos és könnyen munkára fogható.

7. ábra Lencse alkalmazása

Legyen az első a ***Katedrálüveg*** (***Glass tile***) szűrő, mellyel olyan hatást varázsolhatunk képünkre, mintha egyenletlen üvegen néznénk keresztül. A ***Szikra szűrővel*** karöltve csodásan lehet a fotóink karácsonyi hangulatát emelni vele! Minden más olyan helyzetben is jól jön a segítsége, amikor szeretnénk a képünket elmosottá, kevésbé karakteressé tenni. Ilyen eset például, ha egy kép csak másodlagos grafikai elem, viszont túlságosan is figyelemfelkeltő.

Beállítani valónk nincs is sok. Mindössze azt szabályozhatjuk, hogy a katedrálüveg mekkora csempékből álljon. Külön állíthatjuk a vízszintes és a függőleges felbontást.

A második és egyben utolsó a ***Lencse alkalmazása*** (***Apply lens***) szűrő. Ez sajnos egy olyan eszköz, ami több is lehetne, mint ami. Látványos megoldás lenne különböző lencsén keresztül nézni a képünket, de sajnos ezzel az eszközzel csak nagyítani tudunk. Nagyobb problémának érzem, hogy az egész réteget nagyítja, tehát nincs lehetőségünk beállítani kisebb sugarú lencsét, márpedig így egy kényelmes funkciótól esünk el. Ezt a korlátot kikerülhetjük, ha kivágjuk és új rétegre helyezzük azokat a képrészleteket, amelyeket nagyítani szeretnénk, vagy kijelöljük az adott képrészletet. Sajnos a homorú és más különleges kialakítású lencsékről végképp megfeledkezett a készítő.

A *7. ábrán*, ezen a fekete-fehér képen egészen jól kivehető a ***Lencse alkalmazása*** szűrő eredménye. A pirossal kiemelt területeket kijelöltem, majd a kijelölésre alkalmaztam a szűrőt. Örülök neki, hogy ilyen sok eszközt ad a kezünkbe a ***Gimp***. Remek lehetőségek ezek, ha jól és jókor használjuk őket. A fények és üveg-effektusok használatakor nagyon is jól látszik, hogy milyen látványos eredményt érhetünk el egy jól sikerült fotót felhasználva. Azt javaslom mindenkinek, hogy vegye elő a fényképezőgépet és fotózzon! Vigyük magunkkal munkahelyre, természetbe és szórakozni is. A fotózást sajnos/szerencsére nehéz helyettesíteni. Szerkesztésre, retusálásra és minden másra pedig itt a ***Gimp***!

Juhász Attila
(rabszolga@goraffe.hu)

Az Információ Technológiai Kar hallgatója a Pázmány Péter Katolikus Egyetemen. Érdeklődik a bioinformatika és a neurális hálózatok iránt. A fotózás és a tánc mellett öt éve foglalkozik webgrafikákkal. A linux terjesztések közül a Gentoo és az Ubuntu áll legközelebb a szívéhez. Fotós oldala a http://people.goraffe.com/attila címen található.

DVD körkép

Linuxon nagyon jó DVD kezelő programok vannak, de sajnos a legtöbb disztribúció jogi okok miatt nem tartalmazza ezeket. Ráadásul megbízható bináris fájlokat sem könnyű találni. Cikkem a legismertebb DVD szoftverek (MPlayer, Xine, Ogle, VLC, DVDShrink, Vobcopy, DVD::rip) lefordításával, telepítésével, emellett a DVDShrink és a DVD::rip használatával foglalkozik.

Amikor a *Linuxszal* megismerkedtem, első benyomásom az volt, hogy nem lehet rajta filmeket nézni. Sokan beszéltek ugyan arról, hogy a *Linux* szinte minden videóformátumot probléma nélkül kezel, de nekem sehogy sem sikerült vetítésre bírni. Miután persze mélyebben átláttam a videózás rejtelmeit, személyesen is megtapasztaltam a mondat igazságtartalmát. *Linux* alá igen sok *DVD*/videólejátszó szoftver íródott. A rengeteg videóformátumnak köszönhetően érdemes az összeset felrakni a gépünkre, hogyha egyik véletlenül nem játszaná a filmet (ami ritkán fordul elő), másik szoftverrel is kipróbálhassuk. Idáig olyan filmmel nem találkoztam, amelyet egyik linuxos alkalmazás sem játszott volna le.

A videó lejátszó szoftverekről általánosságban elmondható, hogy a legjobb az, ha nem a kész binárist töltjük le, hanem magunk fordítjuk őket. Amennyiben kész binárist használunk, könnyen előfordulhat, hogy a lejátszás lassú lesz, vagy fagyni fog. Ennek az oka egyszerű: a videózás nagyon sok erőforrást igényel, ezért a lejátszó programok maximálisan kihasználják a processzorunk multimédiás képességeit. Ha A processzoron lefordítok egy szoftvert, az még nem jelenti azt, hogy egyazon *Linux* terjesztés B processzorral is menni fog. Ez kizárólag akkor igaz, ha A és B utasításkészlete megegyezik. Ha általánosan, mindenhol futó alkalmazást szeretnénk gyártani, akkor fel kell áldoznunk a processzor gyorsítási képességeit, így viszont teljesítmény veszteséggel fogunk adózni. Sajnos nagyon sok nagy *Linux* disztribúció szándékosan elbutított videó szoftvereket tartalmaz az amerikai törvények miatt. Ezek a lebutított binárisok általában használhatatlanok.

A kodekek zömét kiszedik, így gyakorlatilag semmit sem lehet velük kezdeni. Az *UHU* azon kevés *Linux* disztribúciók közé tartozik, ahol nem butították le a videólejátszó szoftvereket. Szerencsére az európai számítástechnikára vonatkozó törvények több ésszerűséget mutatnak, mint az amerikaiak, így megtehették. Az *UHU* csomagok között az itt említett programok az *Ogle* kivételével mind megtalálhatóak. Ennél a disztribúciónál lehet a binárisokat használni.

Előkészületek a fordításhoz

Az alkalmazások lefordításához a *gcc* fordítóprogramot, a *make*-et és a szükséges *devel* csomagokat telepíteni kell. Erre bővebben nem térek ki, mert teljesen disztribúció függő. Szerencsére a hibaüzenetek általában maguktól értetődőek. Amikor a fordító panaszkodik, hogy egy komponens hiányzik, a komponenst és a hozzá tartozó *devel* csomagot telepíteni kell. Én *SuSE Linux 10.0* alatt próbáltam végig a fordításokat, a hiányzó csomagokat az *OpenSuSE 10*-ből vettem. A komponensek összegyűjtése természetesen nem egy egyszerű dolog, elég sokat kell játszani vele. Előfordulhat hogy néhányat kihagytam a felsorolásból, mely *SuSE Linux* alatt alapból telepítve van.

Sajnos a legtöbb videóalkalmazás jelenleg nem fordul gcc 4.x-szel, bár sokat javítottak már. Nemrég teljes verziószám ugrás történt a *gcc*-nél (3.x-ről 4.x-re) és ez sok problémát okoz. Mindenesetre jóval simábban megy a dolog most, mint a *gcc 2.x* és *3.x* váltásnál. A fordítónk verziószámát

```
gcc --version
```

paranccsal kérdezhetjük le. Ha a parancs hatására 4-es érték íródik ki, akkor nincs szerencsénk. Ebben az esetben azt javaslom, hogy töltsük le a *gcc-3.4.5*-ös verziójának forrását a netről (➲ *http://gcc.gnu.org/*) és rakjuk fel második fordítónak.

```
mkdir /opt/gcc-3.4.5
tar xvfj gcc-3.4.5.tar.bz2
cd gcc-3.4.5
./configure --prefix=/opt/
➥gcc-3.4.5
make
make install
```

Gyors eredményt persze ne várjunk. Akár néhány óráig is eltarthat a fordítás. Amikor *gcc 3.x*-et akarunk használni, csak írjuk be, hogy

```
export CC=/opt/gcc-3.4.5/bin/gcc
```

Ettől kezdve a 3.x-es fordítóval fog menni a fordítás.

Megjegyezném, hogy telepítés után néhány program nem működik rendesen, mert nem talál meg minden függvénykönyvtárat (*lib*.so* fájlok).

Ilyenkor az

```
ldconfig
```

parancsot adjuk ki és menni fog.

MPlayer

Az első, nem kimondottan *DVD*, hanem inkább videólejátszó szoftver, amit érdemes telepíteni az az *Mplayer* (➲ http://www.mplayerhq.hu).
Az *MPlayer* alapszoftver, melyre sok más program épül, ezért mindenképpen javaslom hogy felkerüljön a gépünkre. Komoly hibája az, hogy a *DVD* menükezelés nincs megvalósítva, ezért csak sávonként nézhetjük a filmet. Videófilmek (*avi*, *mpeg*,...) vetítésére viszont kiválóan alkalmas.
A program webhelyéről három fájlt kell letöltenünk: a forrást (*MPlayer-1.0pre7try2.tar.bz2*), a kodekeket (*essential-20050412.tar.bz2*) és egy bőrt (skin), mondjuk a standardot (*standard-1.9.tar.bz2*).
Az *MPlayer* nem fordul *gcc 4* alatt, *3.x*-et használjunk hozzá. Másoljuk be a windowsos kodekeket a */usr/lib/win32* könyvtárba

```
cd /usr/lib/win32
tar xvfj   (a kodekek elérési
➥útja)/essential-
➥20050412.tar.bz2
```

Csomagoljuk ki és fordítsuk le az *MPlayert*:

```
tar xvfj MPlayer-1.
➥0pre7try2.tar.bz2
cd MPlayer-1.0pre7try2
./configure --enable-gui
make
make install
```

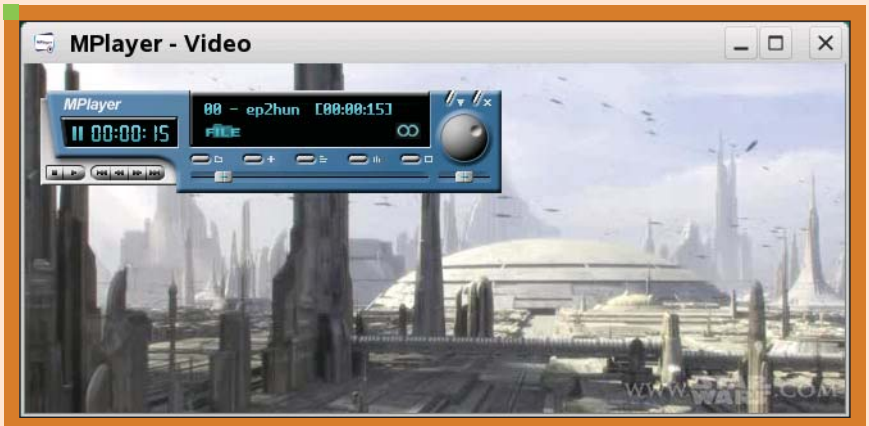
Hasonlóképpen kell a bőrt is telepíteni:

```
cd
/usr/local/share/mplayer/Skin
tar xvfj   (a skin elérési
➥útja)/standard-1.9.tar.bz2
ln -s standard default
```

Fordítás után az *MPlayer* a

```
gmplayer
```

paranccsal indítható.



1. ábra Az MPlayer standard bőrrel

A rendszerünk finomhangolása

Miután sikerült elindítani az első *DVD* lejátszó szoftverünket, elkezdhetjük a rendszer finomhangolását. Rakjunk be egy műsoros *DVD*-t a meghajtóba.

```
gmplayer
```

Aztán jobb klikk a háttéren, majd a helyi menüből válasszuk a *DVD -> Open disc...* pontot. Nyomjuk le az *f* billentyűt, ezzel átkerülünk teljes képernyős (*Full Screen*) módba. Ha szerencsénk van, azonnal elkezdődik a film vetítése, hátradőlhetünk és örülhetünk munkánk gyümölcsének. Akik nem ilyen szerencsések, azoknak felsorolok néhány tipikus problémát és a megoldásukat.

Probléma: nem látunk semmit, a vetítés video kimenetre (vo) vonatkozó hibaüzenettel leáll, vagy a kép zavaros.

Ebben az esetben egyértelműen a videóvezérlő a hibás. A megfelelő meghajtó kiválasztásával mindig megoldható a probléma. Jobb klikk a háttéren, *Preferences*, *Video* fül.
A választható meghajtók a következők:

- *xv* – Hardver gyorsítással rendelkező videó meghajtó. Először mindig ezzel érdemes próbálkozni.
- *X11* – a lejátszás hardver gyorsítás nélkül megy, lassú, más programokban az *xshm* lát el hasonló feladatot.
- *gl, gl2 OpenGL* alapú videómeghajtók.
- *xvidix* – Nagyon gyors videómeghajtó, akkor jó, ha a kártyánk támogatja a *VIDIX*-et

A két legfontosabb meghajtó az *xv* és az *X11*. Az *xv*-t először kell megpróbálni, mert általában működni szokott, az *X11*-et utoljára, mert biztos megy, lassú és ilyenkor nincs hardver gyorsítás.

Probléma: a képet látjuk, de a hangot nem hallunk, vagy a program az hangkimenetre panaszkodik (ao).

Nézzük meg, hogy a hang nincs-e lehalkítva. Ha le van, tekerjük fel. Amennyiben ez nem segít, nézzük meg, hogy nem fogja-e valamelyik multimédiás alkalmazás a hangkimenetet (például *XMMS*). A zene lejátszásánál a *Pause* gomb megnyomása általában nem engedi el az hangkimenetet. *Stoppal* állítsuk le a zene lejátszását, vagy zárjuk be az alkalmazást, ez segíteni fog.
Ha mégsem segít, akkor cseréljük le az audio-meghajtót. Jobb klikk a háttéren, *Preferences*, *audio* fül. A következő audio-meghajtók közül választhatunk: *mpegpes*, *oss*, *alsa*, *arts*, *jack*, *null*, *pcm*.
Az *oss*, vagy az *alsa* kiválasztása általában megoldja a bajt. A *null* azt jelenti, nincs hangkimenet (nem biztos, hogy ezt akarjuk), a *pcm* pedig *WAVE* fájlba ír.

Probléma: a DVD lejátszása szaggat, de amikor merevlemezről nézem a filmet tökéletes.

A probléma oka az, hogy a *CD/DVD DMA*-ja nincs bekapcsolva, ennek köszönhetően a *DVD/CD* olvasás lassú.
A *DMA*-t minden *Linux* disztribúcióban másképp kell beállítani, egyedül

2. ábra A Xine DVD játszás közben

a parancssoros megoldás az, amelyik disztribúciófüggetlen.
Adjuk ki rendszergazdaként a

```
hdparm -d1 /dev/hdc
```

parancsot (a *CD/DVD* általában *hdc*,vagy *hdd* szokott lenni). A *DMA* bekapcsolásával engem egy apró meglepetés is ért, a *DVD* írás maximális sebességen nem működött többet. Rendre elrontotta írás közben a *DVD*-ket. Végül egy szkript lett a megoldás, hogy *DVD* írása alatt kikapcsolom a *DMA*-t.

Probléma: Teljes képernyős mód nem működik, fekete hátteret kapunk, a közepén a film ugyanolyan kicsi, mint volt, nem nagyítja ki.

A probléma oka az, hogy *x11* meghajtót használunk, ahol a nagyítás nem hardverből működik. Ha megfelelően erős processzorunk van, akkor bekapcsolhatjuk a szoftveres nagyítást. A *$HOME/.mplayer/config* fájlhoz adjunk hozzá egy `zoom = yes` opciót. Ha ez nem lehetséges, az *X11*-et kell úgy beállítani, hogy több alacsonyabb felbontást is engedélyezzen (*SaX2*). A képernyőfelbontások között *Ctrl-Alt-+*, *Ctrl-Alt—* segítségével válthatunk, így képesek leszünk kinagyítani a filmet.

Probléma: A lejátszás szaggat, ilyenkor a winchester lámpája mindig kigyullad.

Ebbe a problémába egyszer futottam bele, a megoldása is érdekes lett. A reiserfs fájlrendszer okozta a bajt, időnként sokáig lefoglalta a Winchestert, ezért a video szaggatott. Miután a reiserfs-t ext3-ra cseréltem, a probléma megoldódott. Nem tudom, hogy a legújabb reiserfs-ekkel előfordul-e, de ha igen, akkor ext3-ra át kell állni.

Xine

DVD-nézésre a *Xine* videólejátszót ajánlom, mert tökéletes rajta a *DVD* menük kezelése. A *Xine* a ➲ *http://xinehq.de/* címről tölthető le. A forrást a *xine-lib-1.1.1.tar.gz* fájl tartalmazza.
A fordítása egyszerű:

```
tar xvfz xine-lib-1.1.1.tar.gz
cd xine-lib-1.1.1
./configure
make
make install
```

Ha nem csak parancssorból szeretnénk videót nézni, grafikus felületet is lehet telepíteni. Több bőr is van a programhoz, *KDE* alatt a *Kaffeine*-t javaslom, egyébként pedig használhatjuk az alapértelmezett xine-ui-0.99.4.tar.gz-t. Ennek a fordítása teljesen úgy történik, mint a *xine-lib*-é ezért nem írom le újra.
Indítása `xine` paranccsal történik. A *Xine* megpróbálja automatikusan felismerni az audio/video meghajtót, ha ez mégsem sikerül, az *MPlayer*-nél megszokott módon beállíthatjuk. Az *X11* meghajtónak az *xshm* felel meg.
Jobb klikk a háttéren, *Settings*, *Setup*, *Gui* fül: A *Configuration Experience Level* mezőt állítsuk át *Advanced*-re, és a *Video, Audio* fül alatt pedig válasszuk ki a megfelelő vezérlőket.

Ogle

Az *Ogle* a legősibb olyan *DVD*-lejátszó program, amely képes *DVD* menük kezelésére. A lejátszót a ➲ *http://www.dtek.chalmers.se/groups/dvd/* címről tölthetjük le. Fordításához szükségünk lesz még a libdvdcss, libdvdread, libmad, liba52 függvénykönyvtárak telepítésére is. A honlapjukról linkek mutatnak ezekre a projektekre. Telepítésük a szokásos

```
./configure
make
make install
```

segítségével történik. Szükség van még a *libjpeg*, *libxml2* komponensekre is, de ezeket a legtöbb disztribúció tartalmazza, ezért nem kell fordítani. Töltsük le az *ogle-0.9.2.tar.gz* fájlt, fordítsuk le és telepítsük *gcc 3.x* segítségével (*gcc 4* alatt nem fordul!). A grafikus felület telepítéséhez töltsük le és fordítsuk le az ogle_gui-0.9.2.tar.gz fájlt. A fordításhoz szükség lesz egy csomó *GNOME 1.x*-es cuccra (a *libglade* függés miatt), ezeket a disztribúciónk segítségével installálhatjuk.
Az *Ogle* egyébként régi alkalmazás és nem vagyok benne biztos, hogy még fejlesztik. Az utolsó kiadott stabil változata 2003 novemberében jelent meg. Ennek köszönhetően előfordulnak *DVD*-k, amiket nem képes lejátszani.

3. ábra Az Ogle DVD lejátszó program

VLC Media Player

A *VLC* média lejátszó szintén jó választás *DVD* nézésre. A *DVD* menüket támogatja ugyan, de ebben az esetben nem *Open Disc*-kel kell megnyitni a *DVD*-t, hanem *Open Directory*-val. Sajnos nem minden esetben működik rendesen, valószínűleg ezért is nem ez az alapértelmezett viselkedés *Open Disc* esetén. Összességében véve a *VLC*-t még egy kiforratlan fiatal, de ígéretes projektnek érzem. A fordítása sem egyszerű dolog, mert rengeteg kisebb nagyobb projektet kell hozzá letölteni (mint az *Ogle*-nál). Jó lenne, ha kiadnának olyan *VLC* változatot is, ami eleve tartalmazná ezeket a komponenseket (a *Xine* és az *MPlayer* is ezt csinálja). Az eszközmeghajtók szintén beállíthatóak a *Settings/Preferences/Audio* és *Video* fülek alatt. Az *xv* itt *XVideo* névre hallgat.

A *VLC Media Player* telepítése *gcc 3.x*-et igényel, menete a következő. Töltsük le az *FFMPEG* könyvtárat a ➲ *http://www.ffmpeg.org* címről, majd adjuk ki a következő parancsokat:

```
tar xvfz ffmpeg-0.4.9-
➥pre1.tar.gz
cd ffmpeg-0.4.9-pre1
./configure --enable-pp
➥--enable-gpl
make
make install
make installlib
```

Ha *gcc 4* az alapértelmezett fordítónk, akkor a

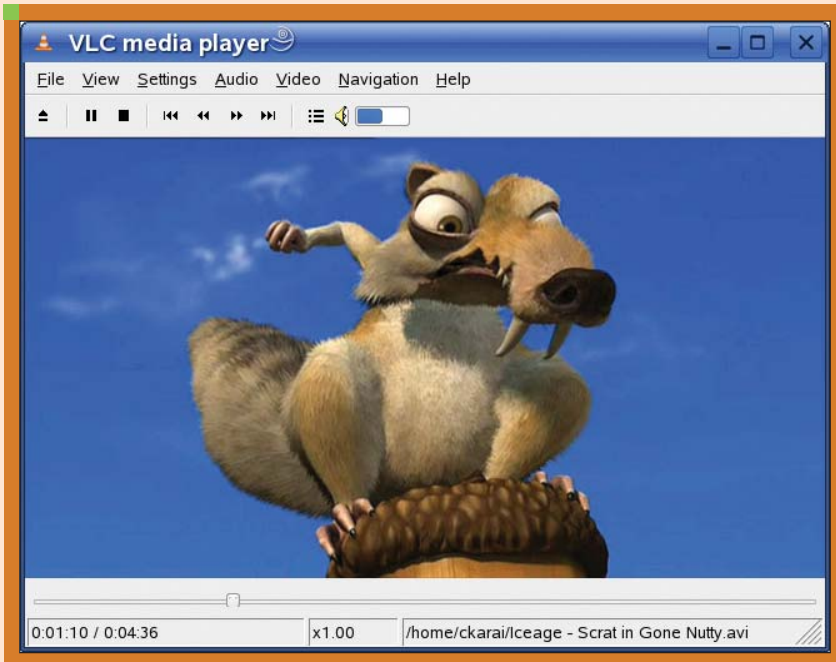
```
make CC=/opt/gcc-3.4.5/bin/gcc
```

kiadásával fordítsunk.

Az `export CC`-s dolog itt nem megy.

Telepítsük a *LIBMPEG2* könyvtárat (➲ *http://libmpeg2.sourceforge.net/*, *mpeg2dec-0.4.0b.tar.gz*). Ha az *Ogle*-nál még nem tettük meg, rakjuk fel a *libdvdcss*, *libdvdread*, *a52dec*, *libmad*, *libdvdnav* komponenseket is (➲ *http://www.videolan.org/vlc/download-sources.html*). Fordításuk a *libdvdnav* kivételével a szokásos `./configure ; make ; make install` kiadásával történik.

Ami a *libdvdnav* fordítását illeti, a következőképpen kell eljárnunk:



4. ábra A VLC Media Player

```
./autogen.sh
make
make install
```

A *VLC* projekt lefordításhoz szükségünk lesz a *wxWidget* csomagra és a hozzá tartozó *devel* csomagra, ami *wxGTK-compat* névre hallgatott *SuSE* alatt.

Töltsük le a *VLC* forrását a ➲ *http://www.videolan.org/vlc/download-sources.html* oldalról, majd fordítsuk le a következőképpen:

```
tar xvfz vlc-0.8.4a.tar.gz
cd vlc-0.8.4a
./configure --with-ffmpeg
➥-tree=../ffmpeg-0.4.9-pre1
make
make install
```

A program a `vlc` parancs kiadásával indul.

DVDShrink

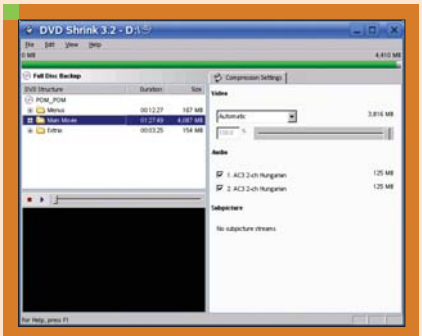
A *DVDShrink* program azért íródott, hogy többrétegű gyári *DVD* lemezeket képesek legyünk egy rétegű 4,7 GB-os lemezre másolni. A tömörítés minőségbeli romlást eredményez ugyan, de tartalék *DVD*-nek kiválóan alkalmas. A *DVDShrink* webhelye a ➲ *http://www.dvdshrink.org/*. Innen azonban nem lehet a programot letölteni, ugyanis használata amerikai jogszabályokba ütközik.

A ➲ *http://www.softpedia.com/get/CD-DVD-Tools/CD-DVD-Rip-Other-Tools/DVD-Shrink.shtml* címről ajánlom letölteni. A program *Windows* alatt íródott, de a *wine Windows* emulátor majdnem tökéletesen futtatja. A *wine*-t a legtöbb disztribúció tartalmazza, de nem biztos, hogy alapból telepítve van.

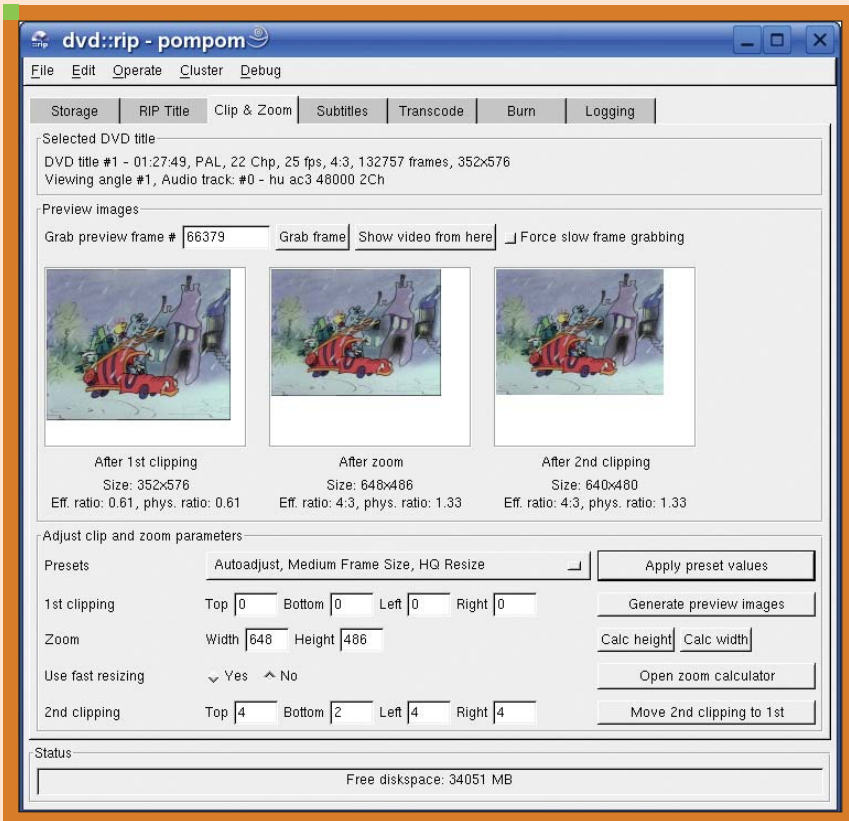
Töltsük le a *dvdshrink32setup.zip* fájlt és telepítsük:

```
unzip dvdshrink32setup.zip
wine dvdshrink32setup.exe
```

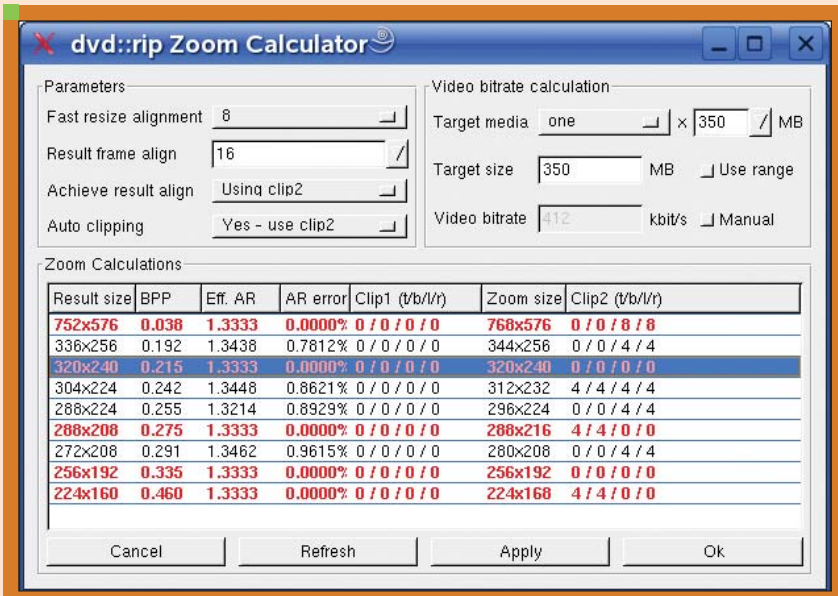
A telepítés menete a *Windowsnál* megszokott módon történik. A *wine* egy ikont is létrehoz az asztalunkon, innen lehet később indítani a programot. A *wine*-os változat használatánál egyetlen baj szokott lenni: a kódolt DVD-kkel nem képes megbirkózni



5. ábra A DVD Shrink 3.2 wine alatt



6. ábra DVD::rip vágás és átméretezés



7. ábra DVD::rip az optimális felbontás kiválasztása

(csak igazi *Windows* alatt). Az ilyen *DVD*-k másolásával a következő szakaszban foglalkozom.
Indítsuk el a programot. Helyezzük be a *DVD*-t és nyomjuk meg az *OpenDisc* gombot, vagy ha merevlemezen van a film, az *OpenFiles*-t. *Wine* alatt alapértelmezésben a `$HOME` könyvtárunkat *H*-nak, a / könyvtárat *Z:*-nak és a *DVD*-t *D:*-nek látja. Ha valamelyik eszközt nem látná, a *winecfg*-vel tudjuk módosítani a beállításokat.
Ha a *DVD* megnyitásánál *Failed to initialize ASPI device* hibaüzenetet kapunk, *winecfg* segítségével állítsuk át az operációs rendszert *NT 4.0*-ra. Így működni fog.

Ekkor megjelenik az *Analysing* ablak és a *DVDShrink* elkezdi analizálni a *DVD*-t. Az *Enable video preview* kikapcsolásával kicsit növelni tudjuk az analizálás sebességét.
Miután mindezt befejezte, a főablakban megjelenik a *DVD* tartalma. *DVD structure* alatt a *DVD* szerkezetét látjuk. Általában három részből áll: *Menu*, *Main Movie*, *Extras*.
A *Compression Settings* alatt látjuk, hogy a *DVD* kiválasztott része milyen mértékben lesz tömörítve és mely hangcsatornákat, feliratokat fog tartalmazni. Minden hangcsatorna (*Audio*) és felirat (*Subpicture*) mellett egy jelölőnégyzet van, mellyel engedélyezhetjük, vagy tilthatjuk, hogy felkerüljön a másolt *DVD*-re.
A videótömörítésénél a következő opciók közül választhatunk:

- *No Compression:*Tömörítés nélkül másol
- *Automatic:* Automatikusan dönt a tömörítés mértékéről
- *Custom Ratio:* Mi állítjuk be a tömörítés mértékét
- *Still Image:* Álló képpel helyettesít mindent
- *Still Pictures:* A videót fél másodpercenként mintavételezett álló képekkel helyettesíti

Én általában az *Extras* és a *Menu* részt szoktam maximálisan összetömöríteni (*Custom Ratio*-t minimumra húzom), és a *Main Movie*-ból pedig a nem kívánt nyelveket eltávolítom. A feliratokat általában megtartom, mert viszonylag kevés helyet foglalnak.
Amikor kiválasztottuk a megfelelő tömörítést, nyomjuk meg a *Backup!* gombot. Itt előírhatjuk, hogy hova történjen a tömörítés:

- *Hard Disk Folder:* Egy kiválasztott alkönyvtárba
- *ISO Image fájl:* vagy ISO fájlba

A *DVD* elkészítése az *OK* gomb megnyomására történik. A végeredményt egy *DVD*-író programmal (például *k3b*) süthetjük meg.

A vobcopy

Ahogy fentebb említettem a *DVDShrink wine* alatt sajnos nem képes kódolt *DVD*-t másolni. Ez csak *Windows* alól megy.

Nem kell elkeseredni, mert a *vobcopy* megoldja. A *vobcopy* a winchesterünkre kikódolja a *DVD*-t, onnan pedig a *DVDShrink* már képes elboldogulni vele. *Vobcopy* használata közben engem egy kellemes meglepetés is ért. A 2-es régiókódú *DVD* meghajtóm képes volt 1-es régiójú *DVD* lemezt dekódolni. Nem tudom, hogyan csinálta, valószínűleg nem volt a *DVD* meghajtómban hardveres védelem. A *vobcopy* a ➲ *http://vobcopy.org/* címről letölthető. A fordításhoz szükségünk van a *libdvdread* és *libdvdcss* projektekre. Ha eddig nem installáltuk őket, tegyük meg. A *vobcopy* fordítása szintén a szokásos módon megy (`./configure ; make ; make install`).
Miután elkészültünk, helyezzük be a dekódolni kívánt *DVD*-t, majd adjuk ki a következő parancsot:

```
vobcopy -m /media/dvd
```

A */media/dvd* helyre a saját *DVD*-nk mountolási pontját írjuk be. És most szépen dőljünk hátra, mert a *DVD* dekódolás igen csak sok időnket fogja igénybe venni.

A DVD::rip telepítése

A *DVD::rip* program segítségével tetszőleges *DVD*-ből *AVI*-t, *MPEG*-et, vagy *SVCD*-t készíthetünk. Telepítéséhez *gcc 3.x* szükséges. A *DVD::rip* tulajdonképpen a *transcode* program grafikus felülete.
A telepítés menete a következő.
A kódoláshoz szükségünk lesz az *XviD* kodekre. Töltsük le tehát a ➲ *http://www.xvid.org/downloads.html* címről, majd fordítsuk és telepítsük:

```
tar xvfz xvidcore-1.1.0.tar.gz
cd xvidcore-1.1.0/build/generic
./configure
make
make install
ldconfig
```

Telepítsük az *MPlayert* (ahogy fentebb leírtam).
Ha eddig nem telepítettük a *liba52*, *libdvdread*, *libdvdcss*, *ffmpeg*,*mpeg2dec* komponenseket, most tegyük meg. Az *ffmpeg* telepítése nem triviális, erről a *VLC*-s részben írtam.
Telepítsük a *lame MP3* kódolót (➲ http://www.mp3dev.org/, lame3.96.1.tar.gz) és a *transcode*-ot (➲ *http://www.transcoding.org/cgi-bin/transcode*, *transcode-1.0.2.tar.gz*).
A telepítés mindkét esetben a szokásos módon történik.
Saját Linuxunk telepítőlemezéről tegyük fel a *perl-Gtk-Perl* modult, a *libintl perl* modult pedig töltsük le a ➲ *http://www.cpan.org/modules/by-module/Locale/libintl-perl-1.16.tar.gz* címről, majd adjuk ki a következő parancsokat:

```
tar xvfz libintl-perl
➥-1.16.tar.gz
cd libintl-perl-1.16
perl Makefile.PL
make
make install
```

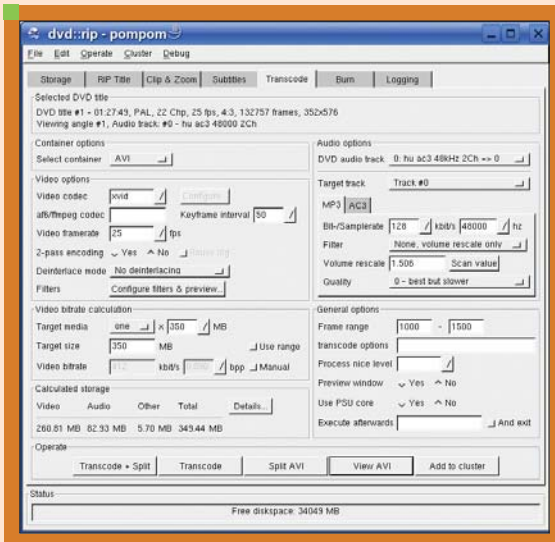
Töltsük le a ➲ *http://www.exit1.org/dvdrip/* oldalról a *Video-DVDRip-0.52.6.tar.gz* fájlt, és ez a kódot is fordítsuk le:

```
tar xvfz Video-DVDRip
➥-0.52.6.tar.gz
cd Video-DVDRip-0.52.6
perl Makefile.pl
make
make install
```

Végül indítsuk el a programot a `dvdrip` paranccsal.

DVD konvertálása AVI-ba a DVD::rip segítségével

A program indítása után megjelenik egy *Preferences* feliratú ablak. A *DVD device* helyre írjuk be a DVD eszközünk nevét (például */dev/dvd*), a *DVD mount point*-hoz a csatolási pontját (például */media/dvd*), a *Default data base directory*-hoz azt a helyet írjuk be, ahová az adatainkat szeretnénk tenni, a *Default directory for .rip project files*-ba pedig azt a helyet, ahova a projekt fájlok kerülnek. Miután mindent beállítottunk, nyomjuk meg a *Check settings on this page* gombot. Ha minden rendben van (zöld színű *OK* feliratok jelennek meg a legalsó mezőben), nyomjunk *OK* gombot.
Megjelenik a *DVD::rip* főablaka. A *File* menüből válasszuk ki a *New project* pontot. Töltsük ki a projekt nevét (például a film neve, amit *AVI*-ba akarunk konvertálni), majd kattintsunk a *RIP Title* fülre.
Helyezzük be a *DVD*-t, majd nyomjuk meg a *Read DVD table of contents* feliratú gombot. Ez betölti a *DVD* tartalomjegyzékét. Válasszuk ki a konvertálandó filmet (ez a leghosszabb sáv) úgy, hogy egyszer rákattintunk. Válasszuk ki a kívánt hangcsatornát is (*select audio track*), majd nyomjuk meg a *RIP selected title(s)/chapter(s)* feliratú gombot. Ezek után elmehetünk egy hosszabb sziesztára, mert a *DVD* rippeléséhez kell egy kis idő.
A rippelés befejezésekor kattintsunk a *Clip & Zoom* fülre. Itt tudjuk megadni a felbontást, amibe a filmet konvertálni fogjuk. Az ablak tetején három képet látunk: az első kép az első vágás, a második kép a nagyítás, a harmadik a második vágást ábrázolja, azaz a végleges képet. Használhatunk előre beállított



8. ábra DVD::rip a konvertálás beállításai

1. táblázat

Result size	Legvégső méret
BPP	Bit per pixel. Nagyon fontos adat. Ökölszabály az, hogy 1 pixelhez legalább 0,2 bitre szükség van. Ha ez alá megyünk, az látható hibákat eredményezhet. Minél magasabb ez az adat, annál jobb minőséget kapunk.
Eff. AR	Effektív képarány (X,Y aránya, aspect ratio)
AR error	Képarány hiba (százalékos eltérés az eredetitől)
Clip 1	Első vágás
Zoom size	Nagyítás/kicsinyítés utáni méret
Clip 2	Második vágás

2. táblázat

Select Container	Ezzel az opcióval választhatjuk ki, hogy MPG, AVI vagy OGG lesz. Válasszuk ki az AVI-t.
Video Options	A legfontosabb számunkra a codec kiválasztása: legyen xvid, vagy ffmpeg. A 2 pass encoding megmondja, hogy a konvertálás egy, vagy két menetben történjen. A két menet szebb képet eredményez, de fele olyan gyors. Egyenlőre kapcsoljuk ki a 2 menetet.
Video bitrate calculation	Itt állíthatjuk be, hogy hány CD-s filmet akarunk és mekkora a CD mérete. Emellett ellenőrizhetjük, hogy a BPP nem lett-e túl alacsony (< 0,2)
Audio options	Itt tudjuk kiválasztani, hogy egy adott nyelv melyik sávra kerüljön. A kívánt nyelvet a Track#0-ra tegyük, a többi nyelvet kapcsoljuk ki. Lehetőségünk van MP3 kódolásra, vagy az eredeti AC3 hang megőrzésére. Az alapértelmezett 128 kbps MP3 általában jó választás szokott lenni. A Volume rescale-lel tudjuk hangosítani, vagy halkítani a filmet. Alapértelmezésben a leghangosabbra normalizál..
General Options	A Frame range paraméter a legfontosabb számunkra. Megmondja, hogy a konvertálás mely Framek között történjen. Mielőtt elindítanánk a teljes konvertálást érdemes a film egy rövid darabját egy menetben megcsinálni. Írjunk be 1000 és 1500-at, ez 20 másodpercnek felel meg 25-ös FPS (Frame Per Second) mellett.

felbontásokat, úgy hogy ***Presets*** kombóból kiválasztjuk az értéket és megnyomjuk az ***Apply Preset Value*** feliratú gombot. A három képet a ***Generate Preview Images*** segítségével generálhatjuk újra.
A legfontosabb gomb ezen az ablakon az ***Open zoom calculator***. Ezzel tudjuk a készülő film minőségét ellenőrizni. Nyomjuk meg.
A ***zoom calculator*** előre kiszámított felbontásokat jelenít meg.

A ***Target media*** segítségével megmondhatjuk, hogy hány ***CD***-re szeretnénk tömöríteni és mekkora a ***CD***-k mérete. A beállítás után nyomjuk meg a ***Refresh*** gombot a felbontások újraszámolásához.
A ***Fast resize aligment*** mező szintén fontos. A képet gyorsabban lehet átméretezni, ha a keletkező felbontás 8-cal osztható és mindkét irányban (X,Y) nagyítunk, vagy mindkét irányban kicsinyítünk.

A ***Zoom Calculations*** felirat alatt egy táblázatot látunk. A táblázat elemei a következőek *(1. táblázat)*.
A ***Zoom Calculator*** azokat a felbontásokat, ahol nincs képarány hiba pirossal jelzi, azokat ahol van feketével. Képarány hiba esetén a film szereplőit vagy összenyomva, vagy megnyújtva láthatjuk. Ha kiválasztottunk egy felbontást, nyomjuk meg az ***Apply*** gombot, ez érvényesíti a választásunkat és az ***OK*** gomb megnyomásával kilépünk a ***Zoom Calculatorból***.
Ha feliratos filmet konvertálunk, a ***Subtitles*** menü alatt a feliratot rá lehet égetni a filmre. Nincs arra lehetőség a programban, hogy a feliratokból ***srt***-t, vagy ***sub***-ot csináljunk. A feliratok ***DVD***-n kép formájában vannak tárolva, ezért ahhoz hogy ***srt***-vé, vagy ***sub***-bá konvertáljuk őket, betűfelismerésre lenne szükség. A feliratokra ennél bővebben nem térek ki. Váltsunk át a ***Transcode*** fülre *(2. táblázat)*.
Miután mindent beállítottunk, nyomjuk meg lent a ***Transcode*** gombot. A 20 másodperces filmdarabot elég gyorsan megcsinálja. A ***View AVI*** gombbal tekinthetjük meg a végeredményt.
Ha valami nem sikerült, visszatérhetünk a ***Clip & Zoom*** fülre javítani.
A ***Split AVI*** gombbal tudjuk szétszedni a több ***CD***-s filmeket,
a ***Transcode+Split*** egy lépésben kódolja és darabolja fel a keletkező ***AVI*** fájlt.
Ha a konvertálás tökéletes volt, töröljük ki a ***Frame range***-hez beírt értékeket, váltsunk át kétmenetes módra, majd nyomjuk meg a ***Transcode*** gombot. A teljes film konvertálását általában este érdemes elindítani, hogy reggelre befejeződjön. A ***3 GHz***-es gépemen egy másfél órás film átalakításához körülbelül 7-8 óra kellett.
Remélem cikkemmel sikerült némi segítséget nyújtani a fontosabb linuxos videószoftverek telepítéséhez és kezeléséhez. Használatukhoz sok sikert kívánok!

Karai Csaba
(cskarai@freemail.hu)

Informatikus vagyok egy telekommunikációs vállalatnál, szabadidőmet legszívesebben menyasszonyommal töltöm, de szeretek focizni, táncolni és görkorizni is.


Linuxvilág
www.linuxvilag.hu

3D ábrázolás – PoVRay (6. rész)

Egy saját 3D világ felépítése és működtetése sok vesződséggel jár. Ez alatt tulajdonképpen azt kell érteni, hogy a környezetünk tele van olyan tárgyakkal és textúrákkal, amelyek rendkívül változatosak vagy összetett struktúrát alkotnak. Leginkább a növények és a távoli táj tartozik ebbe a kategóriába, mivel változatosak, kaotikusak és a legnagyobb igyekezettel sem tudjuk szabályos testekből elkészíteni ezeket. Ezen túl könnyű észrevenni, hogy a – főleg az emberek által épített – tárgyak kisebb részek ismétlődéseiből állnak.

Csalás és ámítás

A *3D* programok egyik nagy területe a filmekből is ismert digitális trükkök kivitelezése. Tulajdonképpen arról van szó, hogy a valódi világból származó képre ráteszünk egy *3D* programmal készített másik képet. Ha azonos nézőpontot, látószöget és mélységélességet állítunk be a *3D* test elkészítésénél, akkor nagyon jó minőségű végeredményt kaphatunk, amely az egyszerű szemlélőt könnyedén megtéveszti. Egy lehetetlen helyre tett tárgy még nem okoz túl nagy problémát, a tűéles kontúrok sem nagyon feltűnőek, ha nem avatott szemmel nézünk egy ilyen képet. Ellenben a nézőpont és a perspektíva (amelyet a látószög határoz meg) durva eltérése esetén azonnal felkiált minden ember, hogy valami nem stimmel ezzel a képpel.

Nézzünk példának egy tíz-tizenöt éve még javában dúló repülő csészealj őrületet, ugyanis az akkori filmtrükköket jelenleg már képesek vagyunk egy egyszerű számítógéppel elkészíteni. Ehhez keressünk egy néptelen és jellegtelen tájat, példaképpen az *1. ábrán* látható képből fogunk kiindulni, amely ideális helyszínnek ígérkezik. Ez a kép 35 milliméternek megfelelő objektívvel készült, s éppen Pécs egyik külszíni fejtése látszik a rajta a háttérben, az előtérben pedig egy lejtősebb tisztás.

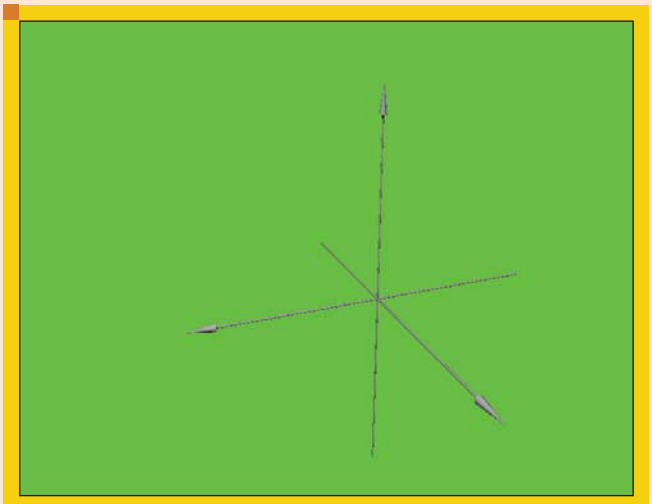
A végleges képnek illene átmennie egy egyszerűbb hihetőségi próbán, így

1. ábra Tisztás az erdőben

az csészealjunkat olyan méretűre kell készíteni, amely még nem feltűnő egy egész város vagy megye számára, viszont eléggé nagy ahhoz, hogy a „funkciójának” eleget tegyen. A legjobb pozíció ezért a *2. ábrán* látható képnek a vörösen bekarikázott része, ahova egy stilizált *UFO* is került (jelen esetben a *GIMP* jóvoltából).

A megfelelő csalás elkészítéséhez pontosan meg kell tervezni a méreteket és a távolságokat, vagyis fel kell mérnünk a fotózás pontjától az elhelyezendő tárgy irányát és távolságát.

2. ábra A repülő csészealj helye



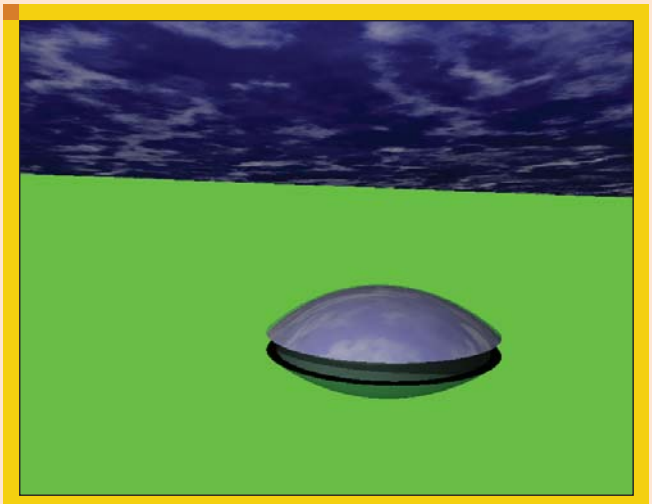
3. ábra Zöld háttéren koordináta rendszer

4. ábra A repülő csészealj helye a mezőn

Ezek után jön a munka nehezebb része, ugyanis el kell készítenünk egy repülő csészealjat a ***PoVRay*** segítségével.
Első lépésként a filmesek által használt ***bluebox*** technológiához kell folyamodnunk, ugyanis a ***PoVRay*** nem képes olyan képet készíteni, amelynél a végtelent átlátszó háttérrel helyettesítené. Sajnos olyan képet sem tud készíteni, amelynek a megadott egyéni kép van a hátterében.
A ***bluebox*** lényege, hogy az elkészített tárgyat egy matt fényű, ugyanakkor egyszínű háttérre tesszük. Ez a szín régebben a kék volt, azonban mostanában már a zöld színt szokták használni a trükkmesterek.
A mi esetünkben ez nagyon egyszerűen kivitelezhető, hiszen a háttér színét nagyon könnyen befolyásolni tudjuk, s ezen a háttérszínen semmi nem fog se megcsillanni, se tükröződni, se árnyékot vetni:

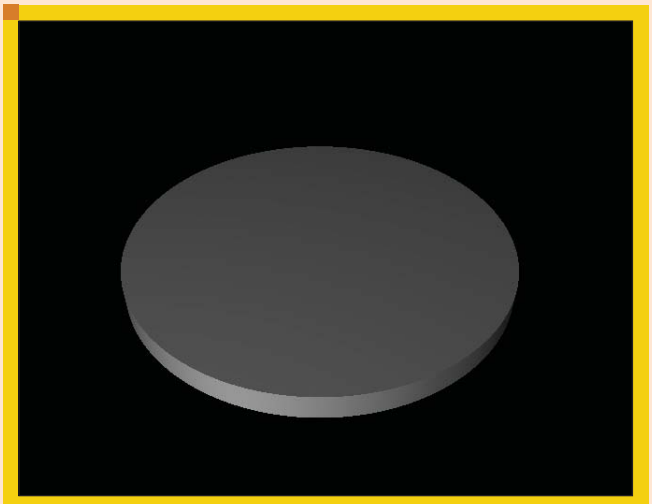
```
background{
 color Green}
```

Kezdésképpen érdemes egy szimpla koordináta rendszert készíteni, amelyet majd ráillesztünk a háttérképre (*3. ábra, pov98.pov*).
Ezen a ponton a ***GIMP*** (vagy egyéb képszerkesztő program) segítségét kell igénybe vennünk, hogy a két képből egyet tudjunk csinálni. A *3. ábrán* látható képet nyissuk meg, majd a Szerkesztés menüben válasszuk a másolást, ugyanis ennek a képnek a háttere valószínűleg az előtér fehér színe lesz, amely nekünk nem megfelelő. Ha a kép kikerült a vágólapra, akkor a ***Fájl*** menüben az ***Új*** menüpontot választva az új kép mérete pont azonos lesz a kimásolt kép méretével. Ezen dialógus ablakon belül válasszuk ki az Átlátszó kitöltést, így kapunk egy üres képet, amelynek a háttere átlátszó. Ebbe bele tudjuk illeszteni a kimásolt képünket, amelyről el kell távolítanunk a matt zöld hátteret. Ezt a ***GIMP*** szín szerinti kijelölésével tudjuk megtenni, amellyel a háttérre kattintva az összes ilyen színű képpont kijelölt lesz: ezeket egy egyszerű kivágással el tudjuk távolítani. Arra kell figyelnünk, hogy a kivágás éles átmeneteket fog eredményezni, amely leginkább a ferde vagy görbe felületek lépcsőzetességében nyilvánul meg. Ha a ***PoVRay*** oldalon próbáljuk megoldani a problémát egy kis élsimítással, akkor a ***GIMP*** nem tudja szépen kivágni az adott színt, és az átlátszó háttér nem a tárgyunk kontúrjával fog találkozni, hanem egy zöld-kontúr átmenettel. Ebből adódik, hogy a ***GIMP*** egyik elmosó

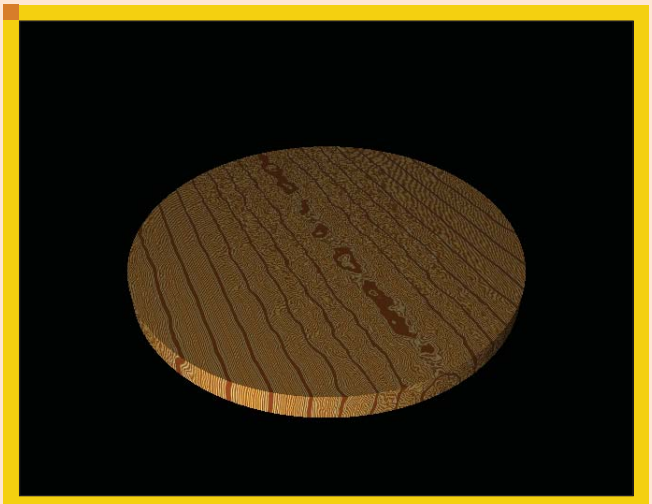


5. ábra A repülő csészealj bluebox

6. ábra A repülő csészealj a mezőn



7. ábra Az óra kezdeményei



8. ábra Az óra számlapja textúrával

eszközét kell igénybe vennünk a feladat elvégzéséhez, s megkapjuk a *4. ábrán* látható eredményt.
Ezek után már csak a csészealjat kell elkészítenünk, amelyet egyszerűen a koordináta rendszer közepébe kell tennünk. Még a mérete sem annyira fontos, mivel a ***GIMP*** képes átméretezni akkorára, amely nekünk megfelel. Az én csészealjam két gömbből indult, amelyeknek csak egy részét hagytam meg, majd ezeket ráillesztettem egy hengerre. A felső gömbcikk fényes, fémszerű textúrát kapott, az alsó pedig zöldes színt, mintha a fű tükröződne vissza rajta. A magasba pedig tettem egy felhős textúrával borított téglalapot, amelyet majd ki kell vágni a beillesztéskor, de szépen tükröződik az ***UFO*** felső részén. A kész kép az *5. ábrán* (*pov100.pov*) látható, ahol a zöld háttér felett ott egy felhős égboltrészlet is.
Néhány egyszerűbb ***GIMP*** művelet elvégzése után már csak ki kell nyomtatni a kész képet, amelyen egy kicsit többet dolgozva egyre inkább megtévesztő lehet a művünk. Bár az én kis ***UFO*** ábrám eléggé kezdetleges, ezért a 10 percnyi munkáért 15 éve egy vagyont fizettek volna a „szaklapok” (*6. ábra*).
A fenti lépéseket szinte bármilyen fényképpel megtehetjük, így csak a fantáziánk és a tudásunk szab korlátokat, ha egy nem létező dolgot a létező világban szeretnénk viszontlátni.

A virtuális világ programja

A ***PoVRay*** leíró nyelve sokban hasonlít a ***C*** vagy ***C++*** nyelvhez, s ezt már említettem a sorozat első részében. Ezen hasonlóság még arra is kiterjed, hogy a ***C*** vezérlési szerkezeteit teljesen azonos módon tudjuk használni a ***PoVRay*** állományaiban. Képesek vagyunk elágazást és ciklust készíteni, amelyek jól jönnek bizonyos testek elkészítésekor.
A leíró nyelv megismeréséhez induljunk ki egy egyszerű tárgyból: egy szimpla faliórát fogunk elkészíteni, amelyet majd bárhol fel tudunk használni, mint egy speciális objektumot. A ***PoVRay*** használatához – ha nem is feltétlen szükséges, de – jól jön egy kis programozói véna, így az óra elkészítését teljesen más alapokon kell véghezvinnünk, mint azt a grafikusok szokták művelni: programot kell írnunk.
Az egyik ***C***-ből átvett szerkezetet már ismerjük, ez az `#include` névre hallgat és arra szolgál, hogy külön állományokban tároljunk különböző dolgokat, így a fő ***PoVRay*** állományunk sokkal egyszerűbb és áttekinthetőbb lesz. Erről a direktíváról nem is lehet többet elmondani, egyszerűen csak használnunk kell, így hozzunk létre egy ***ora.inc*** nevű állományt, amely az óra alkatrészeit fogja tartalmazni:

```
#declare Ora=cylinder{
 <0.0,1.0,0.0>,<0.0,0.0,0.0>,
 ➥10.0
 pigment{
  color White}}
```

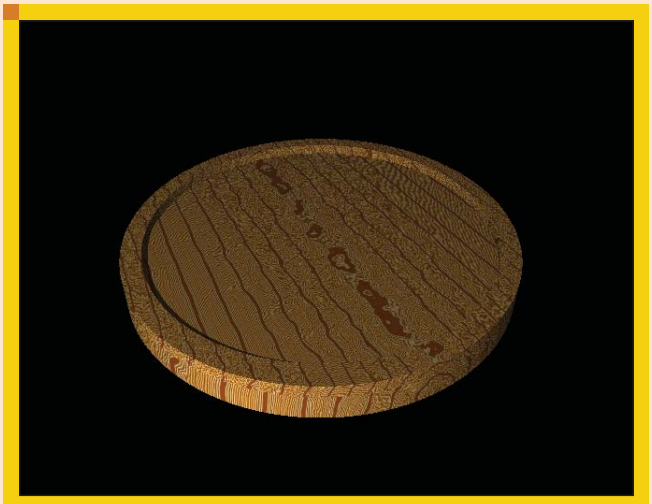
A „főprogramunk” hivatkozni fog a fenti fájlra:

```
#include "colors.inc"
#include "ora.inc"

object{Ora}
```

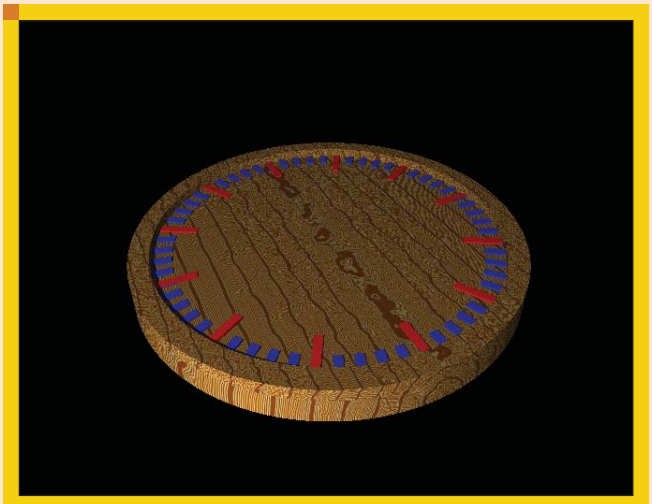
Ennek megfelelően kapunk egy szürkés korongot (*7. ábra*), mint az óra számlapja.
A ***PoVRay*** korai verziói nem tudtak változókat kezelni, bár ezt a hiányosságot nagyon hamar pótolták. A változók nagyon hasznosak tudnak lenni, ha egy-egy összetett tárgy egyes paramétereit szeretnénk módosítani, ezért célszerű az óra számlapjának textúráját egy ilyen változóba tenni, amelynek kezdőértéket is adunk:

```
#include "woods.inc"

#declare Szamlap=
 texture{
  T_Wood31
  scale 10.0}

#declare Ora=
 cylinder{
  <0,1,0>,<0,0,0>,10
  texture{
   Szamlap}}
```

Az óra számlapja egy fa erezetéhez lesz hasonló (*8. ábra*), kivéve, ha a `Szamlap` változónak a főprogramban más értéket adunk és `#declare` helyett `#macro` direktívát használunk, de erről egy kicsit később még lesz szó. Vegyünk egy karimát az órához, amelyre majd a fedő üveget fogjuk tenni, s nevezzük ezt `Karima`-nak. Fontos, hogy a változókat úgy nevezzük el, hogy ezzel se önel-



9. ábra Az óra számlapja és karimája



10. ábra Az számlap jelölésekkel

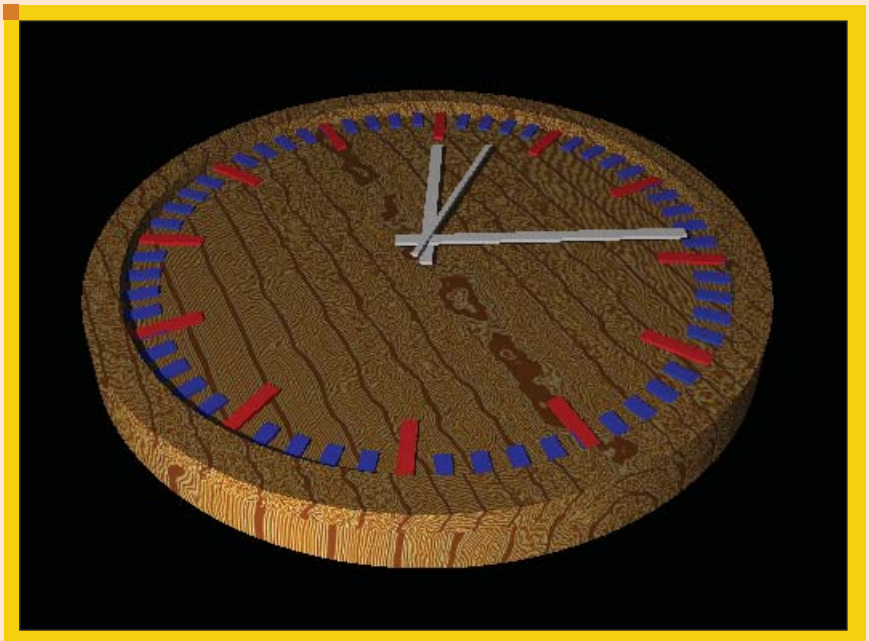
lentmondásba, se más állomány változóival ne keveredjünk. S ha a számlapon kívül egy kis karimát is szeretnénk az óránknak, akkor már lesz egy kis problémánk az elnevezésekkel, így jobb ennek elébe menni:

```
#declare OraSzamlapTexture=
 texture{
  T_Wood31
  scale 10.0}

#declare OraKarimaTexture=
 texture{
  T_Wood31
  scale 5.0}

#macro OraSzamlap()
 cylinder{
  <0,1,0>,<0,0,0>,10
  texture{
   OraSzamlapTexture}}
#end

#macro OraKarima()
 difference{
  cylinder{
  <0,1.5,0>,<0,1,0>,10
  texture{
   OraKarimaTexture}}
  cylinder{
  <0,1.6,0>,<0,0.9,0>,9
  texture{
   OraKarimaTexture}}}
#end

#macro Ora()=
 union{
  OraSzamlap()
  OraKarima()}
#end
```

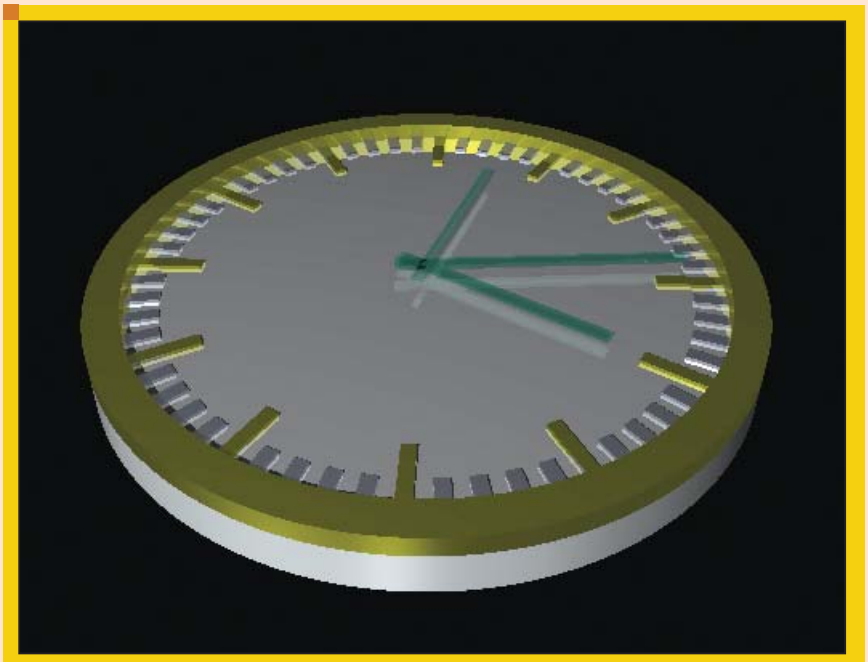
A fájl elérési útját mindenféleképpen vegyük bele az elnevezésekben (jelen esetben ez az `Ora`), amelyhez fűzzük hozzá az alkatrész nevét (például `Szamlap`), majd a *PoVRay* szerkezet nevét (például `Texture`). Ha ehhez tartjuk magunkat, akkor nehezen kerülünk névütközésbe, hiszen az `OraSzamlapTexture` egyértelműen meghatároz egy elemet, amelyből csak egy van. Egy új elem is megjelent, mégpedig a `#macro` direktíva, amely egy függvényhíváshoz hasonlít a leginkább. Egyelőre nincs paramétere, de majd azzal is találkozunk kicsit később.

A `#declare` és a `#macro` között abban van különbség, hogy az előbbi statikus viszonyt jelent, a PoVRay egyszerűen csak beilleszti az elkészített tárgyat a megadott paraméterekkel, míg a `#macro` esetén újra feldolgozza annak törzsét. Ennek akkor van jelentősége, ha meg szeretnénk változtatni a tárgy textúráját: `#declare` esetén nem dolgozza fel a PoVRay újra a megadott tárgy „programját”.

Az óra számlapjára rá kell tennünk pontosan 12 jelet, amely az egész óráknak felel meg. Ezt megcsinálhatjuk nyers erőből is, egyenként létrehozva őket, de akár okosan is: ciklust használva. A ciklus arra lesz jó, hogy az órát jelző testet lemásoljuk 12 példányban és 30 fokonként elforgassuk:



11. ábra Az elkészült óra, alapértelmezett textúrákkal



12. ábra Az elkészült óra, fémes textúrával

```
#macro OraOraJel(ora)
 box{
  <9,1,-0.2>,<7,1.1,0.2>
  texture{
   OraOraJelTexture}
  rotate <0,30*ora,0>}
#end

#macro Ora()
 union{
  OraSzamlap()
   OraKarima()
  #local szamol=0;
  #while (szamol<12)
   OraOraJel(szamol)
   #local szamol=szamol+1;
  #end}
```

A `#declare` és a `#local` között annyi a különbség, hogy a `#local` hatóköre a *PoVRay* blokkok ({ és }) között marad, míg a `#declare` globális hatókörrel rendelkezik. A ciklusváltozót – amellyel elszámolunk tizenkettőig – célszerű helyi változóként kezelni, így nem kerülünk ütközésbe egy másik ciklussal, ha például több órát szeretnénk kitenni a falra.
Az `OraOraJel` makrónak már van egy paramétere, mégpedig egy `ora` nevű változó, amelyet megszorozva harmiccal elforgathatjuk 30-30 fokkal az egész órákat jelző téglalapot. A sikeren felbuzdulva tegyük ki a perceket jelölő kis téglalapokat is (*10. ábra*).

Ezeken túl már csak az óra mutatói vannak vissza, ezeket paraméterekkel meg kell tudnunk adni, s figyelnünk kell arra, hogy hibás értéket ne lehessen megadni az óránknak. Erre szolgál az feltételtől függő elágazás, amellyel korrigálni tudjuk a hibás értékeket. Először is tegyük ki az óramutatót és vizsgáljuk le, hogy 1 és 12 között van-e a megadott érték:

```
  #if (OraOra<1)
   #declare OraOra=1;
  #end
  #if (OraOra>12)
   #declare OraOra=12;
  #end
  OraOraMutato(ora)
```

A perc- és a másodpercmutató azonos módszerrel kitehető, így tulajdonképpen kész is van a falióránk (*11. ábra*), esetleg egy védőüveget tehetünk rá, de ez a virtuális világban nem fontos.
Csak annyi van vissza, hogy az órát saját textúrával lássuk el, ehhez a főprogramban át kell írnunk a megfelelő változók értékeit:

```
#declare OraSzamlapTexture=
 texture{
  pigment{
   color White}
  finish{
   reflection 0.9
   roughness 0.01
   specular 0.4}}

#declare OraKarimaTexture=
 texture{
  Gold_Texture}

#declare OraOraJelTexture=
 texture{
  Gold_Texture}

#declare OraPercJelTexture=
 texture{
  Silver_Texture}

#declare OraOraMutatoTexture=
 texture{
  Green_Glass}

#declare OraPercMutatoTexture=
 texture{
  Green_Glass}

#declare
OraMasodpercMutatoTexture=
 texture{
  Green_Glass}

#declare OraOra=8;

Ora()
```

A következő részben egy virtuális világban a mozgás bírjuk a tárgyakat és a kamerát, illetve MPEG videót készítünk a *PoVRay* kimenetéből.

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/

Térhatás – A Blender használata (7. rész)
Modellezés görbékkel

Mindezidáig csak az úgynevezett „mesh” típusú objektumokkal foglalkoztunk, amik A vertexeken, és az azok között húzott éleken alapszanak. Most egy kis kitérőt teszünk a nem-mesh alapú modellezés világába, megismerkedünk a görbékkel, és a belőlük létrehozható felületekkel.

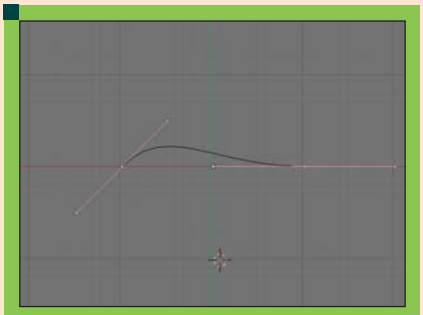
Talán azt gondolhatja most az olvasó „Mire jó ez? Mit tudhat egy görbe? Hiszen eddig is megvoltam nélküle...”. Persze valószínűleg csak azért, mert még nem fedezte fel, mennyivel egyszerűbben és gyorsabban lehet egyes alakzatokat megrajzolni görbék segítségével, mint azok nélkül. Ígérem, e cikk végigolvasása után nem úgy tekintünk többé a görbékre, mint konc, hanem mint lehetőség.

Minek nevezzelek?

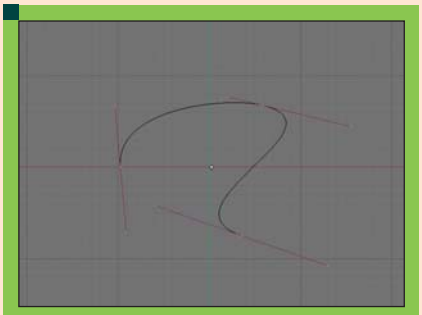
A ***Blenderben*** többféle, alapvetően különböző típusú görbe létezik. Az első egy ***Pierre Etienne Bézier*** nevű úriemberről kapta a nevét. Ő dolgozta ki ugyanis a görbék leírásának ezen módját, amit azóta számos helyen alkalmaznak tervezési, grafikai és egyéb célokra. A ***Blenderben Bezier Curve*** néven találkozhatunk vele. Testvére a ***NURBS Curve***, ami egész pontosan a ***Nem Uniform Racionális B-Spline*** görbe rövidítése, ami – be kell vallanom – nem tudom mit is jelent valójában. (aki esetleg tud valamiféle információval szolgálni, az e-mail címemet megtalálja a cikk végén). A továbbiakban mindegyik görbéről részletesen szó lesz.

A Bézier görbék

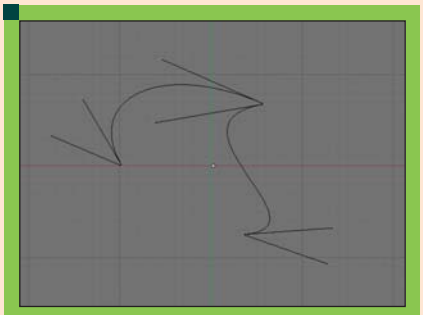
Mint már említettem, ez a görbe leírási módszer a feltalálójáról kapta a nevét. A módszer egy autógyárban született meg, ahol is karosszéria elemek formájának pontos leírására használták.



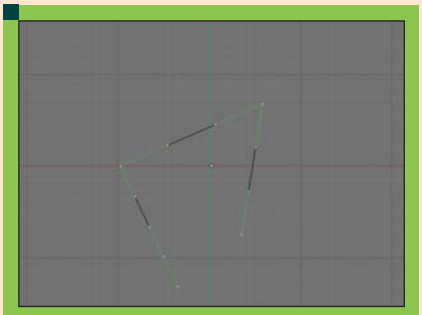
1. ábra Bézier görbe



2. ábra Bézier görbe (aligned handle)



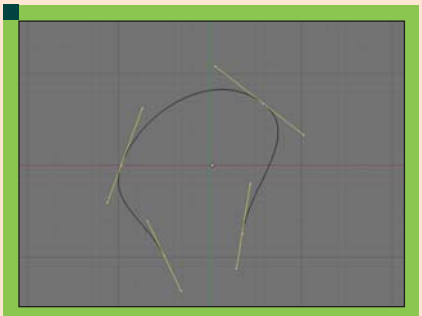
3. ábra Bézier görbe (free handle)



4. ábra Bézier görbe (vector handle)

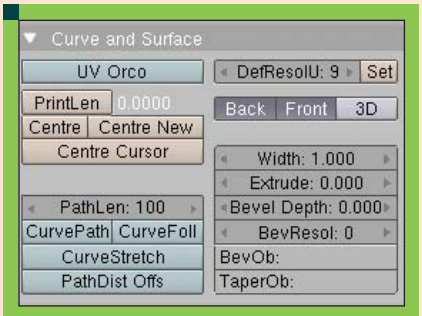
A görbét afféle koordinátákkal – úgynevezett ***controll point***-okkal – és belőlük húzott szakaszokkal – angol nevén ***handle*** – adhatjuk meg. Az *1. ábra* egy ilyen görbét mutat. Fontos, hogy a görbe pontosan áthalad a megadott pontokon (nem úgy, mint a ***NURBS*** görbék esetében). A pontok közötti rész a pontokból húzott kis fogantyúk alapján jön létre interpolációval. Ha elmozdítjuk ezeket a kis fogantyúkat, vagy mozgatjuk a ***controll point***-okat, a görbe íve is megváltozik (*2. ábra*).

A ***Blender*** többféle ***handle***-t ismer, ezek közül a legfontosabbak a ***free*** és az ***aligned handle***. Az ***aligned handle*** legfőbb ismertetőjele, hogy a ***controll point***-ból kiinduló két ***handle*** egy egyesesen helyezkedik el, és az egyik oldal egyik mozgatásával a másik is változik. Az első két képen ilyen ***handle*** látható. A *3. ábrán* már az említett ***free handle***-t látjuk. Rögtön feltűnik, hogy a kis szakaszoknak koránt sem kell egy egyenesen elhelyezkedniük, teljesen szabadon

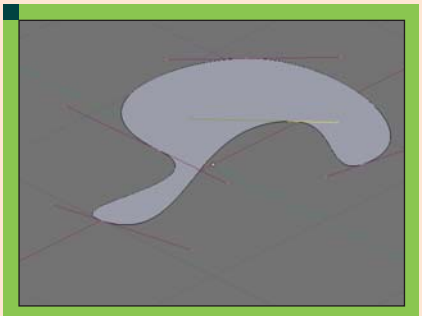


5. ábra Bézier görbe (auto handle)

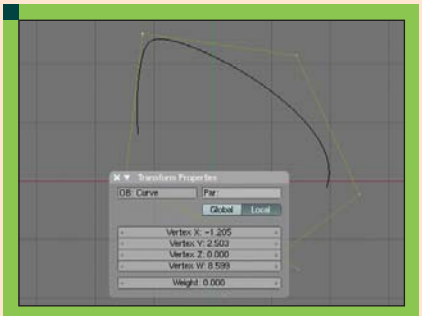
8. ábra NURBS görbe



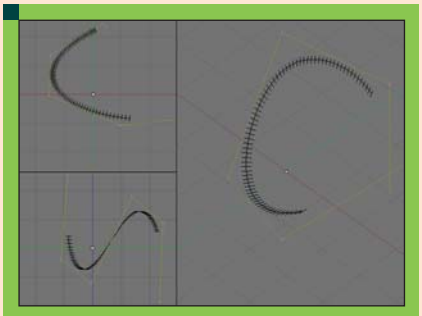
6. ábra Curve and Surface panel



9. ábra Zárt 2D-s NURBS görbe



7. ábra A W koordináta beállítása



10. ábra 3D-s Bézier görbe

mozoghatnak, és ez a görbén is meglátszik, törések formájában. A *free* és az ***aligned handle*** között a ***H*** billentyűvel válthatunk. Természetesen semmi akadálya, hogy egyik pontból *free*, míg a másikból ***aligned hande*** induljon ki.

A ***V*** billentyű lenyomásával a *4. ábrán* látható ***vector handle*** hozható létre, a ***Shift+H*** kombináció eredménye pedig az ***auto handle*** (*5. ábra*).

Ez utóbbi kettő, csak ideiglenes, ugyanis elmozdítva őket, a ***vector handle*** visszaalakul *free*-vé, az ***auto***-ból pedig ***aligned*** lesz.

Bár ezek a görbék matematikailag tökéletes alakzatok, megjelenítéskor apró szakaszokra kell tördelni őket, különben a szerkesztés és a renderelés lassú lenne. Éppen ezért minden görbének van egy saját felbontása, ami megmutatja a két kontroll pont között kiszámítandó pontok számát. Ezt az értéket a ***Curve and Surface*** (*6. ábra*) nevű panelen állíthatjuk (minden görbéhez külön). Általában egy 30-50-es érték már teljesen sima görbét ad.

A csúnya nevű görbe

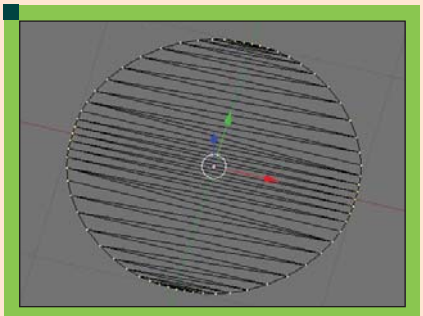
NURBS. Nem egy fülbemászó név, ugye? Nem is próbálom meg kiejteni. A ***Nem Uniform Racionális B-Spline*** görbe a ***Bézier*** hibáit hivatott semlegesíteni. A ***Bézier*** görbékkel való modellezés ugyanis nehézkes, mivel a kontrollpontok a görbe felületén helyezkednek el, és az interpoláció miatt akármilyen kicsi mozdítás is aránylag nagy változást idéz elő a görbén, így egyes felületek lemodellezése szinte lehetetlen. De itt van nekünk a ***NURBS*** görbe, ami interpoláció helyett közelítést használ. Ez a mi szemszögünkből annyit jelent, hogy a görbe nem feltétlenül megy át a kontroll pontokon, csak megközelíti azokat, így ugyanolyan pontok esetén a ***NURBS*** görbe általában simább, mint egy Bézier görbe, arról nem is beszélve, hogy nem kell törődnünk a ***handle***-nek nevezett kis szakaszokkal sem. Van azonban más nyűg. Ugyanis a ***NURBS*** görbék minden kontroll pontja rendelkezik egy negyedik koordinátával (általában ***W***), ami megadja, hogy az adott pont mennyire vonzza a görbét, vagyis a görbe mennyire közelíti meg azt. (*7. ábra*) Ha ezt a számot nagyon nagyra állítjuk, a görbe érinti az adott pontot. Természetesen minden ponthoz külön állíthatjuk ezt a számot, így tulajdonképpen tetszőleges görbéket létrehozhatunk. (*8. ábra*)

Görbék szerkesztése

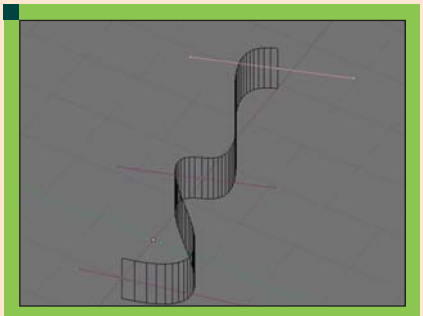
A ***Blender*** megengedi, hogy görbék kontroll pontjait a vertexekhez hasonlóan szerkesszük, így szinte az összes transzformáció (***Grab***, ***Rotate***, ***Scale***) alkalmazható rájuk, amit a vertexeknél megszoktunk. A görbe pontjait az ***E*** billentyű megnyomásával bővíteni tudjuk, illetve egy kijelölt pontot törölni tudunk az ***X*** billentyű segítségével. Két pont közé újabb pontot szúrhatunk be a ***Subdivide*** használatával, az egyetlen megkötés, hogy alapértelmezésben mindezt csak 2D-ben tehetjük. Azért írtam, hogy alapértelmezésben, mert van lehetőségünk bekapcsolni a 3D-s kiterjesztést egy görbén – ***Curve and Surface*** panel (*6. ábra*) – van azonban néhány dolog, ami csak ***2D***-s görbék esetén lehetséges.

Zárt görbék

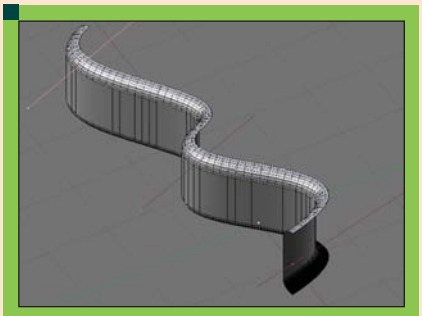
Például a ***Blenderben*** a két dimenziós zárt görbék automatikusan egy sík felületté alakulnak, míg a 3D-s görbék nem. Természetesen az, hogy a görbénk síkbeli, nem jelenti azt, hogy az X és Y tengelyekre vagyunk korlátozva, csupán azt, hogy görbénket kizárólag egy adott síkon módosíthatjuk. Ez a sík azonban változtatható egész egyszerűen úgy, hogy a görbénket ***object mode***-ban elmozgatjuk, elfordítjuk, stb. De hogyan is lehet zárt görbét létrehozni?



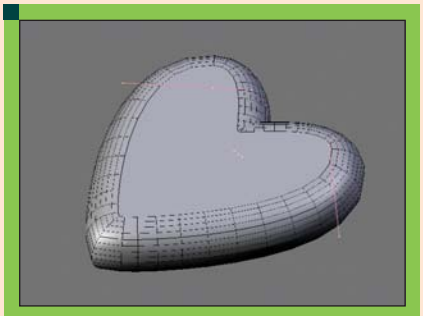
11. ábra Görbéből hálózat



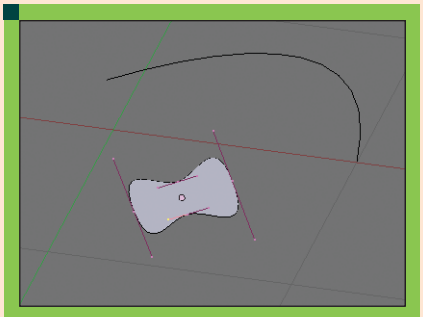
12. ábra Görbe megvastagítva



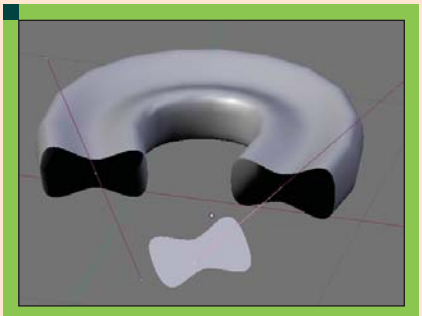
13. ábra Görbe lekerekítéssel



14. ábra Zárt görbe

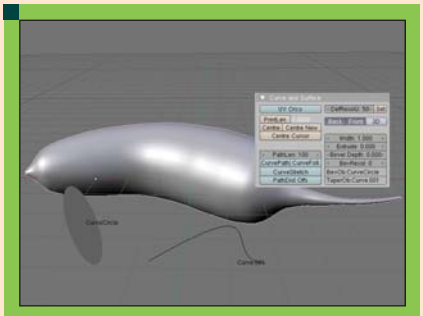


15. ábra BevOb előtt

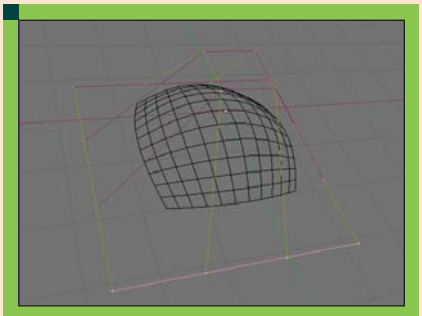


16. ábra BevOb után

Mi sem egyszerűbb, ***edit mode***-ban nyomjuk meg a ***C*** billentyűt. Ha a görbénk kétdimenziós volt, egy gyönyörű síkidomot kaptunk (*9. ábra*). A ***C*** betűt újra megnyomva, visszakapjuk eredeti görbénket. Ez igaz a ***Bézier*** és a ***NURBS*** görbékre is. Az így létrehozott felületeken lyukakat is létrehozhatunk, ha egy zárt görbét egy másik zárt görbe belsejébe helyezünk. Ilyenkor a két görbe valójában egyazon objektum része.



17. ábra TaperOb + BevOb

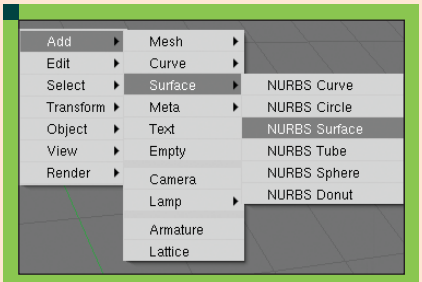


18. ábra NURBS Surface

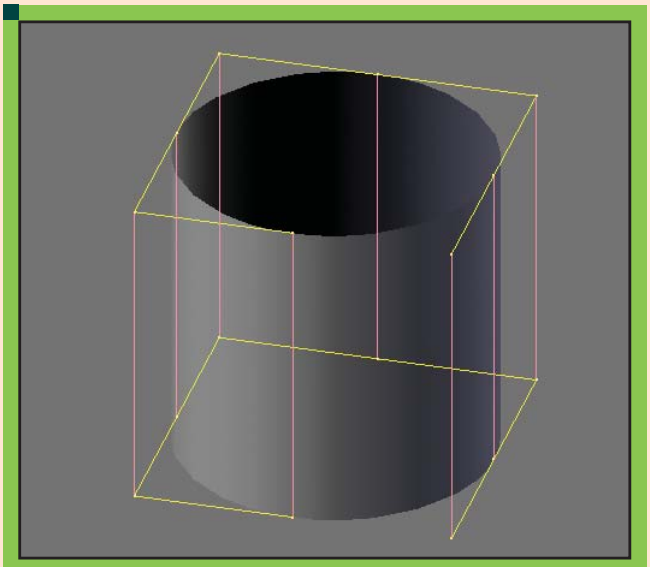
Mire jó ez?

A görbéknek önmagukban nem sok hasznuk lenne, azonban a ***Blender*** által biztosított eszközökkel együtt – ha csodákra nem is – sok minden másra képesek. Az egyik legfontosabb, hogy bármikor átalakíthatóak hálózattá (mesh) így bármilyen alakzat létrehozásában nagy segítség lehet, amit a hagyományos úton nehéz lenne modellezni. Az hálózattá alakításhoz használjuk az ***ALT+C*** gombokat. Figyelem! Ez a módosítás egyirányú, tehát jól gondoljuk meg, mit is alakítunk át, mert a visszaalakításra nincs mód. A ***mesh*** részletessége természetesen függ a görbe felbontásától, amit a már leírt módon állíthatunk be. A *11. ábrán* egy hálózattá alakított zárt görbe látható.

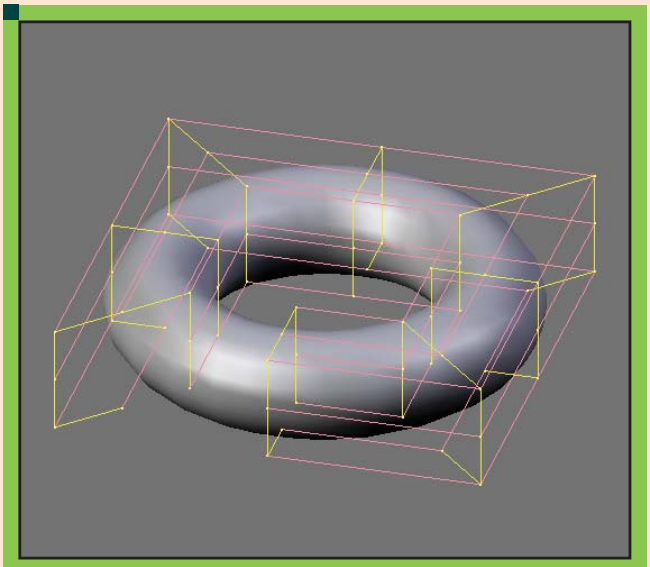
A ***Blendernek*** hála, a görbék önmagukban is használhatóak néhány dologra, így nem kell feltétlenül hálózattá alakítani őket, ha valamit kezdeni akarunk vele. Csupán a ***Curve and Surface*** panelen található beállítások számtalan lehetőséget rejtenek. Az ***Extrude*** érték megváltoztatásával például „kihúzhatjuk" a görbénket, így az máris láthatóvá válik rendereléskor. (*12. ábra*) A ***Bevel Depth*** opcióval afféle letörést vagy lekerekítést varázsolhatunk a görbénknek (*13. ábra*), a ***BevResolv***-val pedig ez utóbbi opció részletességét állíthatjuk be. Zárt görbék esetén az eredmény egy megvastagított, lekerekített alakzat (*14. ábra*) Talán már észrevettük a ***BevOb*** és a ***TaperOb*** beviteli mezőket is. Ide létező objektumok (egész pontosan létező görbék) nevét kell beírni, hogy varázsolhassunk vele. A ***BevOb*** az egyszerűbb varázslat, így azzal kezdem. Hozzunk létre egy aránylag hosszú görbét, és mellé egy rövidebbet. Helyezzük el őket a *15. ábrán* látható módon, majd a hosszabb görbe ***BevOb*** mezőjébe írjuk be a rövidebb görbe nevét. Ügyeljünk rá, hogy pontosan adjuk meg a nevet, a kis és nagybetű



19. ábra NURBS Surface létrehozása



20. ábra NURBS henger



21. ábra NURBS tórusz

különböző. Amint megnyomjuk az ***Entert***, a képernyőnkön megjelenik a várva várt csoda. Váltsunk ***edit mode***-ba, majd próbáljuk meg módosítani a hosszabbik görbénket. Az eredmény magáért beszél (*16. ábra*).
A ***TaperOb*** már valamivel összetettebb. Ha van ***TaperOb*** beállítva, akkor az adja meg, az ***Extrude***-nál beállított kihúzás mértékét, tehát ahol a ***taper*** görbe mélyebben van, ott az ***extrude*** (a másik görbén) kisebb. A ***BevOb***-al kombinálva még érdekesebb eredményt kapunk (*17. ábra*).

NURBS felületek

Nem sok értelme lenne a görbéknek, ha csak két dimenzióban dolgozhatnánk velük. A ***NURBS*** görbék egy speciális fajtájával tetszőleges 3D-s felületeket hozhatunk létre. A létrehozás tulajdonképpen úgy működik, hogy egy ***NURBS*** görbe minden kontroll pontját végighúzzuk egy-egy másik görbén, és az érintett felületet kirajzoljuk. A dolgot úgy is felfoghatjuk, mint egy tetszőlegesen eltorzított U * V nagyságú téglalap. A *18. ábrán* egy 3*3-as ***NURBS*** felületet láthatunk.
Az ilyen objektumokat ***NURBS Surface***-nek nevezi a ***Blender***, és az ***Add -> Surface -> NURBS Surface*** gombbal hozhatunk egyet létre (*19. ábra*).
Egy ilyen felületet nem olyan egyszerű bővíteni, mint egy hálót (mesh). Ne felejtsük el, hogy valójában egy eldeformált négyzetrácsunk van, tehát csak egész sorral, vagy egész oszloppal bővíthetünk. Hogy az alakzatunkon épp melyik irány az oszlop, és melyik a sor, ránk van bízva. Általában a sárga színnel jelölt él jelenti az U irányt, a lilával, vagy rózsaszínnel feltüntetett élek pedig a V-t.
Vajon eszünkbe jutott már, hogy ha egyszerű görbék lehetnek zártak, akkor talán egy ***NURBS*** felület is? Furcsának hangzik, de lehetséges, hogy felületünk önmagában végződjön. Ugyanúgy, mint a hagyományos görbéknél, itt is a ***C*** billentyűvel tehetjük meg, most azonban ki kell választanuk, hogy U vagy V irányban szeretnénk végteleníteni a görbénket. Ezt talán egy torzított hengerhez lehetne hasonlítani. Lehet mindkét irányban egyszerre végteleníteni, ilyenkor az eredmény egy szabadon módosított „úszógumi”. Azért ezt a két példát hoztam fel, mert mindkettő (a henger és a tórusz) szerepel a ***Blenderben*** az ***Add -> Surfaces*** menüben (*20-21. ábra*).

Hiányosságok

Ha ilyen jók a görbék és a ***NURBS*** felületek, mégis miért nem használjuk mindig őket? Az ok egyszerű. Mint mindennek, a ***NURBS*** felületeknek is vannak hiányosságai. Bonyolult testeket csak több felület egymás mellé illesztésével modellezhetünk. Ráadásul nincs lehetőség lyukak létrehozására sem a felületen. Be kell vallanom, hogy ezen a területen a ***Maya*** vitathatatlanul felülmúlja a ***Blendert***, de azért a ***Blender*** is tartogat még meglepetéseket.

Ötletek haladóknak

Sajnos nincs hely leírni mindent, de megpróbálok néhány tippet adni, mire lehet még használni a görbéket. A következő számban már szó lesz az animációról, de aki nem bír várni, használhatja a görbéket. Egy objektumot néhány kattintással hozzáragaszthatunk egy görbéhez, amitől az animáció közben a görbét fogja útvonalként használni.
A ***DupliFrames*** nevű eszközzel az animációkor érintett felületet is könnyen megrajzolhatjuk, csak némi leleményesség kell hozzá.
A múlt számban szándékosan nem mutattam be a ***Curve Modifier***-t. Az eszköz ugyanis egy görbét használ a mesh-ünk eldeformálására (akkor még nem ismertük a görbéket), amit ügyesen használva szintén hasznos eszközzel gazdagodunk.
Ennyi tehát márciusra, egy hónap múlva animáció, és ne felejtsük el letölteni az időközben megjelent újabb ***Blender*** változatot a ➲ ***http://www.blender.org***-ról, ami tartalmaz néhány hibajavítást.

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.

Az Unreal Tournament 2004, és gyermekei

A PC-s játékpiac egyik meghatározó szereplőjének, az Unreal sorozat első részének megjelenésével nagyot változott a játékipar: a program kezdetektől fogva folyamatosan konkurenciát állított az addigi piacvezető termékek ellenében. Ebben a cikkben a linuxos verziót, valamint a fontosabb modifikációit fogom bemutatni.

Az ***Unreal*** játékok jelentős részéről már beszámoltunk a ***Linuxvilág*** magazin hasábjain, így aki figyelmesen olvasott a sorok között észrevehette, hogy a ***Tournament*** ág sok tekintetben túlmutat a piac egyéb szereplőinek tudásán és játszhatóságán. Az első rész megjelenésekor azonnal két részre osztotta a rajongói tábort: volt aki egyértelműen a ***Quake*** vonal mögé állt, de akadt szép számmal olyan is, aki temette az ***id Software*** megoldásait. A 2003-ban megjelent epizód valójában konkurencia nélkül aratta le a babérokat, emiatt a 2004-es rész a ***Sport-Shooter*** kategóriában gyakorlatilag behozhatatlan előny birtokában tölti be kategóriájának egyik vezető szerepét. Az ***Unreal Tournament 2004*** látványvilága nem sokban különbözik egy renderelt mozitól. Hihetetlenül nagy pályák (melyek legalább ilyen hihetetlenül kidolgozottak), speciális effektek egész tárháza, megváltozott menüszerkezet és új játékmódok sorakoznak az újítások listáján. Egyúttal visszakerült a repertoárba az előző részből hiányolt ***Assault*** módozat is, ahol csapatunkkal el kell foglalnunk, illetve meg kell védenünk bizonyos objektumokat.

Taktikázásra, csapatjátékra és egyéni „hentelésre” a program egyaránt lehetőséget nyújt, aki pedig nélkülözni kénytelen a hálózati játékmódot, a beépített ***BOT*** rendszerre (tehát a számítógép irányította játékosokra) támaszkodva egy játékos módban is elszórakozhat egy ideig. Apropó, ***BOT***-ok: az elődökhöz híven, természetesen itt is oszthatunk parancsokat csapatunk tagjainak. A virtuális világban járművek (tankok, homokfutók, siklók) is használhatóak, hálózati módban kifejezetten szórakoztató, ahogyan egy haverunk vezette járgányról próbáljuk meg „levadászni” az ellenséges csapat tagjait, komoly utcai harcok közepette. Két apró szépséghibát azonban itt is vélek felfedezni: először is a játék kompromisszumoktól mentesen csak igen erős ***CPU*** jelenlétében játszható, leginkább modern ***nVidia*** gyorsítókártyával, minimum 512 Mbyte központi tár mellett. Másodszor pedig úgy gondolom, hogy a 2004-es epizódnak (tisztességesebb üzlet keretében) talán az előző rész kiegészítéseként kellett volna megjelennie, nem pedig új programként és effektíve magas árral. A játszhatóságot figyelembe véve nem meglepetés a nagy számú rajongói közösség, melybe a szabad rendszerek felhasználói is beletartoznak, hiszen a sorozathoz hűen ez az epizód is rendelkezik linuxos binárissal.

A játék ***„In Box Linux”*** képességű, tehát az említett port dícséretes módon rajta van a boltban vásárolt, gyári ***CD*** illetve ***DVD*** médiákon. ***CD*** verzió használatánál első lépésként a telepítő alkalmazást tartalmazó lemezről a ***linux-installer.sh*** szkriptet át kell másolnunk merevlemezünkre. Az átmásolt szkript indítása után feladatunk szerint csupán követnünk kell a képernyőnkön olvasható instrukciókat, melyek javarészt a lemezek cseréjére kérnek fel (a cserék ügyén fontos: a telepítő ki nem állhatja a ***supermountot***, így akinek a rendszere automatikusan fűzi be a korongokat, annak rövid időre ki kell kapcsolnia ezt a mechanizmust). ***DVD*** verziónál mentesülünk ettől, így a média gyökerében fellelhető telepítő fájl

1. ábra A linuxos telepítő

2. ábra Komoly harcok színtere

3. ábra Meggyőző látványvilág

közvetlen indításával gépünkre „varázsolhatjuk” az 5 Gbyte helyet igénylő játékot. Fontos, hogy a programot már a telepítéskor megadandó szériaszám védi az eszetlen lopkodástól – bár a védelem könnyen megkerülhető, ennek ellenére alapszintű feladatát jól látja el. A telepített ***Unreal Tournament 2004*** konzolon kiadott ut2004 paranccsal indítható (amennyiben nem kértünk kötést az elérési utakra, úgy ezt a telepítési útra állva kell kiadnunk). A grafikai és fizikai modellezésért felelős játékmotor frissítéseit legkönnyebben a ➲ ***http://liflg.org*** oldalról lehet beszerezni, ahol a hivatalos ***x86*** és ***x86/64*** peccseket ***Loki*** – alapú telepítőbe ágyazva tudjuk elérni (emiatt a frissítések ügyén mindössze annyi teendőnk akad, hogy a letöltött ****.run*** állományokat elindítsuk). A cikk írásakor fellelhető legfrissebb verzió a ***v3369*** azonosítót viseli. A hivatalos ➲ ***http://www.unrealtournament.com*** oldalon fellelhető a linuxos demó is, így aki nem szeretne pénzt adni a játékért, néhány pálya erejéig rövid betekintést nyerhet az ***FPS*** lelkivilágába.

A modifikációk

Az ***Unreal Engine*** moddolható. A grafikai elemek, modellek, a fizika és az alkalmazott játékszabályok (akár együttesen is) cserélhetőek a játékmotor felett. Mivel (mint azt említettem) a program mögött igen nagy rajongói közösség áll, így szakértő kezek által, az alapjátékra támaszkodva rengeteg életképes játékvariáns lelhető fel a világhálón. Számosságuk miatt ezeket szinte lehetetlen egytől egyig bemutatnom (a programkód lassan másfél éves, és a hozzá való ***MOD*** verziókat a kezdetektől fogva készítik), így most azt a néhány projektet fogom előtérbe helyezni, melyek számomra valami „megfoghatatlan módon” különlegesek. A programok több módon indíthatóak: az „anyajáték” menüit használva, esetleg külön „start” kötést alkalmazva.

Unreal Annihilation

Az ***Unreal Annihilation*** az „öreg” ***Total Annihilation*** egyik ***remake*** projektje. Felülnézeti, valós idejű stratégiai programról beszélünk, mely rövidddel megjelenése után (minden hasonló programot maga mögött hagyva) hosszú időre elfoglalta a műfaj képzeletbeli trónját. A ***remake*** változat a ➲ ***http://ua.tauniverse.com/*** címen érhető el ***ua_2004_verzió.zip*** állományként, melyet letöltés után struktúrahelyesen ki kell bontanunk a fő program telepítési útjára. A projekt béta állapotú, ennek ellenére a műfaj rajongóinak egy próbát ebben az állapotában is megér. Hardver igénye megegyezik a játékmotor előzőekben leírt paramétereivel, helyigénye pedig megközelí-

4. ábra TA új köntösben

5. ábra Az UnWheel világa

tőleg 300Mbyte. Az *Unreal Tournament 2004 „Community”* illetve *„Instant Action”* menüjét használva indítható el.

UnWheel

UnWheel néven érhető el minden idők egyik legérdekesebb modifikációja, ahol a komoly képességekkel bíró *Unreal Engine* egy autóversenyt kelt életre. A képi világ itt is lehengerlő, az autók viselkedése kielégítő, a pályák pedig nehezek: élvezetes kikapcsolódásra vágyó játékosoknak kötelező darab. Az alapvető verseny funkcióján túl a területbirtoklás, zászlólopás szintén jelen vannak. A ➲ *http://unwheel.beyondunreal.com* hivatalos honlapon elérhető projekt a *Loki Installers for Linux Gamers* csapatának hála elérhető *Loki*-alapú archívként is, így a ➲ *http://liflg.org url* meglátogatása után telepítése nagyon egyszerűen megoldható: mindössze a letöltött *unwheel.verzió.run* állományt kell lefuttatnunk. A hardverigényt itt kicsit erősebbnek érzem, mint az eredeti *Unreal Tournament 2004* esetén. Helyigénye 1200Mbyte körüli. Az anyaprogram *„Instant Action”* illetve *„Host Multiplayer”* funkcióját használva lehet elindítani.

Red Orchestra

Azért, hogy ne mellőzzük a történelmi vonalat, a *Red Orchestra* a második világháborúba kalauzol bennünket. Mérete az összes *MOD* között ennek a legnagyobb (1300 Mbyte helyet kér a merevlemezen), ami a játék tartalmi mondanivalóját tekintve nem meglepő. Ízlés szerint a *Szovjetunió* vagy *Németország* katonáinak bőrébe bújhatunk, részletes és élethű környezetben átélve a régen elmúlt, nehezebb időket.
A háborúnak azon legsúlyosabb szakaszát dolgozza fel a játék, melyben tízmilliós nagyságrendben estek el katonák és civilek egyaránt az északi fronton. A projekt hivatalos honlapja ➲ *http://redorchestra.clanservers.com* url mögött érhető el. Az *Orchestra* az *UnWheel*-hez hasonlóan fellelhető a néhai Loki Entertainment telepítőjébe ágyazva is, így a ➲ http://liflg.org oldalról letöltött red.orchesta.verzió.run állományt lefuttatva könnyen telepíthető. A játékot egy terminálon kiadott redorchestra paranccsal indíthatjuk (amennyiben nem jött létre a kötés az elérési utakon, úgy az */elérési_út/ut2004* mappában létrejött *redorchestra* linket kell használnunk erre a célra).

6. ábra A Red Orchestra

7. ábra A „szelíd MOD”

Piddly's Chance

Egy cimborám hívta fel a figyelmem erre a projektre. Leginkább gyerekeknek való, külső nézetű, mászkálós 3D platformjátékká szelídíti az anyakódot, aranyosan megrajzolt figurákkal és egy „hangyaszerű” főszereplővel. Hátrányként említendő, hogy a *Sony PlayStation2* vezérlőjéhez van igazítva a játékmenü, így minden kívánt kontroll paramétert kézzel kell beállítanunk a *$home/.ut2004/System/PiddleUser.ini* állományban (melynek szerkezete szerencsére áttekinthető és könnyen érthető). A projekt hivatalosan a ➲ *http://chunkyheads.com* címen érhető el. Szerencsére az előzőekben említett *LIFLG* csapat ezt a kódot is „beledrótozta” egy telepítőbe, így installálás ügyén a *piddlys.chance.verzió.run* állományt használhatjuk, az előbbiekben említett módon. Telepítéséhez nagyjából 300Mbyte szabad helyre van szükségünk. Figyelem: ez a játék sem indítható az *Unreal* menüiből: külön start kötést hoz létre önmaga számára, mely *piddly* néven lelhető fel az elérési utakon. További *MOD* verziók ügyén mindenképpen érdemes ellátogatni a ➲ *http://liflg.org* címre, és a natív verziók között keresgélni az *UT* almenükben. Az összes változatot már megemlítenem is nehéz dolog lenne, hiszen csak a legnépszerűbb kódok száma is meghaladja a húszat. A próbálkozások között a labdajátékok egyedi megvalósításától kezdve a taktikai/szimulációs *Shooter*-ig sok minden fellelhető.
Nem maradt más hátra, mint hogy minden érdeklődőnek tartalmas kikapcsolódást kívánjak!

Kovács Zsolt (kovi@linuxforum.hu)
Quake fanatikus. Négy éve a debreceni linuxosok egyike. Töretlenül hisz a Slackware terjesztésben.