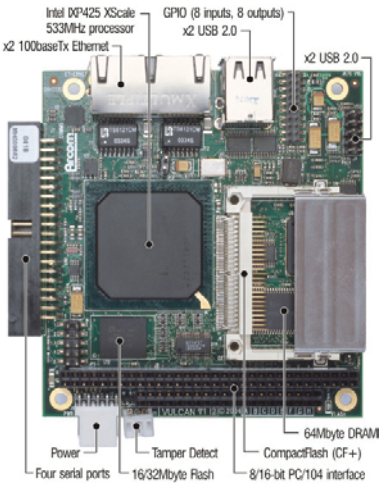
Hírek

4 millió olcsó laptop gyerekeknek

Hivatalosan is megerősítették, hogy a korábban bejelentett 100 dolláros laptopból négy ország (***Nigéria***, ***Brazília***, ***Argentína*** és ***Thaiföld***) 1-1 milliót vásárol. A laptopokon ***Linux*** lesz.

➲ http://www.linuxdevices.com/news/NS2934790376.html

Linuxos mikrogép extrém környezetben



Az ***Arcom*** legújabb ***PC/104*** szabványú ***Vulcan*** típusú célszámítógépét olyan helyekre ajánlják, ahol extrém hőmérsékletek uralkodnak, hiszen működési tartománya -40 és +85 fok közötti. 533 MHz-es ***Intel IXP425***-ös processzor a lelke, mégis képes akár 80 Megabites sebességű titkosításra is. Az eszköz a két Ethernet és négy ***USB*** csatoló mellett ***CompactFlash*** kártya hellyel is rendelkezik, így akár ***WiFi***-vel is bővíthető.

➲ http://www.linuxdevices.com/news/NS4942415046.html

Cell processzoros kiegészítő kártya

A ***Mercury Computer*** bemutatta a ***PCI-Express*** slotba illeszthető kiegészítő kártyáját, mely ***Cell*** processzort tartalmaz és ***Yellow Dog Linuxot*** futtat. A mintegy 180 gigaflops teljesítménnyel bíró eszközt elsősorban nagy számításigényű feladatokra ajánlják, mint például: videó és jelfeldolgozás, illetve raytracing. Az ára előreláthatóan 8000 dollár körül várható. A viszonylag friss ***Cell*** processzort az ***IBM***, ***Sony*** és ***Toshiba*** fejlesztette ki, és a ***Sony Playstation 3***-ban is viszontláthatjuk.

➲ http://www.linuxdevices.com/news/NS6832279023.html

Városi wifi Google-módra

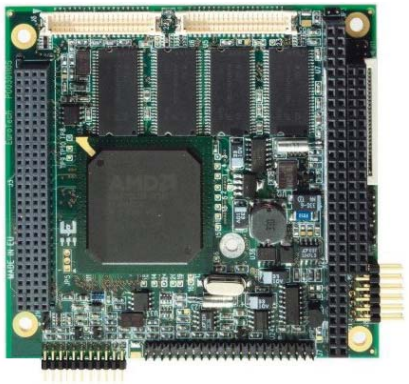
A kaliforniai ***Mountain Viewben*** nem álom többé az ingyenes és (relatíve) gyors vezeték mentes internet elérés. A szolgáltatást bárki igénybe veheti (feltéve, hogy van ***GMail***-es azonosítója) bármilyen készülékkel (legyen az notebook, ***PDA*** vagy ***WIFI*** telefon). A felhasználók 1-4 megabites letöltési és 200-500 kbites feltöltési sebességről számoltak be.

➲ http://www.pcmag.com/article2/0,1895,2004581,00.asp

25 éves az IBM PC

Idén 25 éves az ***IBM PC***, amely igen hamar ipari szabvánnyá vált mind a hardver, mind a szoftvergyártók körében. Látható, hogy hihetetlen mértékű változás ment végbe ez idő alatt, éppen ezért a szakértők is megosztottak a témában, hogy a következő 25 évben milyen fejlődési irányvonal várható.

Alacsony fogyasztású Linuxos modul



A ***Eurotech*** már forgalmazza alacsony fogyasztású modulját, melyet megbízhatósága miatt szinte bármilyen környezetbe ajánlanak. A rendszer lelke egy 133 MHz-es ***AMD Elan*** processzor 64 Mbyte memóriával támogatva. A standard portok (2db Ethernet, 4db soros, párhuzamos, stb.) mellett merevlemezt és floppy meghajtót is csatlakoztathatunk. A teljesítmény felvétele meggyőző, mindössze 4.6 Watt. A modul ***Linux*** mellett ***Windows CE***-t, ***QNX***-et, vagy a ***Linksys*** routerekből ismert ***VxWorks***-öt is futtatja.

➲ http://www.linuxdevices.com/news/NS7960886969.html

Lenovo: Mégis lesz Linuxos támogatás?

Korábban a ***Lenovo*** hivatalosan is bejelentette, nem kívánja támogatni laptopjain a ***Linuxot***, pedig a ***Linux*** felhasználók egyik legkedveltebb laptopcsaládja a ***Thinkpad*** széria. Nos úgy néz ki, megtört a jég, és várhatóan ***SUSE Linux Enterprise Desktop 10***-zel is vásárolhatunk ***Thinkpad*** laptopokat.

➲ http://www.pcmag.com/article2/0,1895,1999092,00.asp

Holografikus tároló

Noha még a ***HD-DVD – Blueray*** csata se zajlott le, a ***Hitachi Maxell*** 2006 karácsonyára ígéri a jóval nagyobb kapacitású (300 gigabájtos) holografikus tárolójukat. A lemez kicsit vastagabb (1,5 mm) és nagyobb (13 centiméter átmérőjű) lesz, mint a ***CD***-k illetve a ***DVD***-k. A hozzávaló lemezek ára 120-180 dollár között lesz, míg a meghajtó ára előreláthatóan 15 ezer dollár lesz. A ***Maxell*** szerint a 800 gigabájtos lemez 2008-ra, míg az 1.6 terabájtos lemez 2010-re várható.

➲ http://www.extremetech.com/article2/0,1697,1997498,00.asp

Linux a Sony Mylo-ban is

Szeptemberben kerül várhatóan a polcokra a ***Sony*** legújabb linuxos kézi gépe 350 dolláros áron. A viszonylag magas ár ellenére az eszközben nem lesz ***GSM*** modul, viszont több ***VoIP*** szolgáltatót (***Google Talk***, ***Skype***, stb.) is támogatni fog ***WIFI*** segítségével. Emellett médialejátszóként is használhatjuk, hiszen ***mp3***-at, ***wma***-t és ***mp4***-t is képes lejátszani. A beépített 1 gigabájtos tárhelyet ***MemoryStickkel*** tovább bővíthetjük. Végül, de nem utolsósorban a beépített ***WIFI*** internetezésre is alkalmassá teszi, ennek létjogosultságát azonban erősen megkérdőjelezi a 320x240 képpont felbontású kijelző. Az adatbevitelről készülékbe rejthető billentyűzet gondoskodik.

➲ http://www.linuxdevices.com/news/NS8202297251.html

Ipari számítógép 8 soros porttal

Az ***Advantech eAutomation*** csoportja jóvoltából már hozzáférhető 1325 dollárért az 1 U magas ipari számítógépük, mely 8 darab túlfeszültség ellen is védett soros porttal és 3 darab ***LAN*** porttal rendelkezik. Az ***UNO-2668*** 400, 600 vagy 1000 MHz-es processzorral és 256 vagy 512 Mbyte memóriával érkezik. ***Linuxot*** vagy beágyazott ***Windowst*** futtathat ***Compact Flash*** memóriáról vagy opcionálisan rendelhető merevlemezről.

➲ http://www.linuxdevices.com/news/NS6399902653.html

Router disztribúció

Az olasz ***Fulvio Ricciardi*** letölthetővé tette az általa fejlesztett router disztribúciót, melyet ***CompactFlash*** kártyáról vagy ***CD*** lemezről futtathatunk. A ***ZeroShell Net***-et weben keresztül konfigurálhatjuk, és az alábbiakat tartalmazza a teljesség igénye nélkül: ***Kerberos***, ***LDAP*** és ***Radius*** hitelestés, útválasztás, híd üzemmód, ***VLAN***, ***VPN***, ***WPA*** és ***WPA2*** a ***WIFI***-hez, ***DNS***, ***DHCP*** stb. A disztribúció futtatásához csupán egy 233 MHz-es számítógép szükséges 128 megabájt memóriával.

➲ http://www.zeroshell.net/eng/download/

Firefox bővítménynek álcázott trójai

Megjelent egy ***trójai***, mely ***Firefox bővítménynek*** álcázza magát, és ezáltal telepíttetheti magát a gyanútlan felhasználóval. A trójai rögzíti az ***egérmozgásokat*** és a ***billentyűleütéseket*** azonosítók és jelszavak után kutatva, majd időnként ***elküldi*** azokat egy előre beleégetett ***IP*** címre.
A szakértők tanácsa: bármilyen bővítményt csak megbízható oldalról telepítsünk.

➲ http://www.techweb.com/wire/security/191101268

A Google-iroda bővül

Bár a *Google Writely* programja még béta státuszú, már megnyitották a felhasználók előtt. Már márciustól fel lehetett iratkozni a várólistára, de a programot csak augusztus végétől próbálhatják ki az érdeklődők.
A *Writely* nem más, mint egy *Microsoft Word* kompatibilis szövegszerkesztő, mely azonban hálózatos alkalmazás, szemben a *Worddel*, amely a felhasználó számítógépén fut.
Most, hogy már van szövegszerkesztő is a táblázatkezelő mellett a *Google* repertoárjában, a világ kíváncsian figyeli, vajon mi lesz a következő alkalmazás?

➲ http://hu.wikipedia.org/wiki/Writely
➲ http://www.pcmag.com/article2/0,1895,2006807,00.asp

iLiad

Linux és elektronikus papír egyesítésével létrehozta az *iRex* cég a hordozható elektronikus könyvet.

A 650 eurós ár kicsit soknak tűnik érte elsőre, de szinte minden benne van, amire szükségünk lehet. A megjelenítésről egy 16 szürkeárnyalatos 1024x768 képpont felbontású érintőképernyő gondoskodik. Amennyiben nemcsak adatbevitel a célunk, hanem kézírásfelismerés is, azért sajnos további 60 eurót kérnek.
A rendszert egy 400 MHz-es *Intel Xscale* processzor hajtja 64 megabájt *RAM*-mal és 128 megabájt flash memóriával megtámogatva, mely 1 hónapnyi újsághoz vagy 30 könyvhöz is elég tárhelyet ad. Adatfeltöltés számtalan módon történhet: *USB* host (például pendrive), *Compact Flash* kártya, *MultiMedia* kártya, 100 megabites vezetékes hálózat vagy akár *WIFI*. Az akkumulátor napi 3 óra olvasás mellett akár 1 hétig is kitarthat. Az eszköz jól bírja a környezeti szélsőségeket, -20 és +70 fok között képes üzemelni.

➲ http://www.linuxdevices.com/articles/AT9929449485.html

Linuxos telefon a Nokia Communicator vetélytársa

A Road nevű a német telefongyártó bemutatott egy olyan készüléket, mely *PDA* és telefon egyben, *Linuxot* futtat *Qtopia* grafikus felülettel.
A készülék szinte minden ma használatos adatátvitelt támogat, legyen az *GPRS*, *EDGE*, *WIFI* vagy *Bluetooth*.
A készülék külső kijelzője 102x65 képpontos egyszínű kijelző, a gyomrából viszont kinyitás után előkerül a 640x240 képpont felbontású színes kijelző és a *QWERTY* kiosztású billentyűzet is. Két változatban kapható majd a készülék: S101 és S101K. A különbség csak annyi köztük, hogy a K-s készülék támogatni fogja a titkosított *VPN*-eket is.
Megfelelő teljesítményt egy 400 MHz-es *Intel PXA264*-es processzor, 128 mbyte memória és 64 mbyte flash memória biztosítja (a tárterület Secure Digital kártyával 1 gigabájtig bővíthető), míg az üzemről egy 1400 mAh kapacitású litium-polimer akkumulátor gondoskodik, amivel körülbelül 4 óra beszélgetési idő vagy 5 óra *PDA* üzem érhető el. A készülék tartalmaz 1.3 megapixeles kamerát is.
Az ára egyelőre nem ismert.

➲ http://www.road-gmbh.de/en/produkte/index_01.html

Mini PC szinte fillérekért

A bangkoki *NorhTec* mennyiségtől függően már akár 100 dollár alatt is szállítja legújabb mini *PC*-jét, a *MicroClient Jr.*-t, mely alapterületre nem nagyobb, mint egy *CD* lemez. Az eszköz működés közben 8 wattot vesz fel.
A rendszer lelke egy 166 MHz-es *Intel Pentium* kompatibilis processzor.
A dobozkában találunk még 128 megabájt nem bővíthető memóriát, *IDE* csatolót laptop merevlemezekhez, 100 megabites sebességre képes hálózati csatolót és három darab USB aljzatot. A teljesen hangtalan üzem érdekében pedig helyet kapott még egy *Compact Flash* memóriát fogadó aljzat, de képes *PXE* szabvány szerinti lemez nélküli indításra is. A mini *PC Linuxot* vagy *Windowst* futtathat.

➲ http://www.linuxdevices.com/news/NS6828123924.html

Újabb gabonakör

Furcsa ábra tűnt fel *Oregon* államban az egyik gabonatáblában. Első ránézésre azt hihetnénk, hogy a földönkívüliek is *Firefoxot* használnak, de mint kiderült, csak az *Oregoni Állami Egyetem Linux* felhasználói múlatták így az időt.

➲ http://lug.oregonstate.edu/gallery/firefox-crop-circle

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 kernelnek egyetlen egy küldetése van ezen a világon. Hogy mi az? Pufferként kell működnie a felhasználói tér és a közösen használt erőforrások között. Ennyi. SEMMI MÁS NEM SZÁMÍT."

(Linus Torvalds)

Greg Kroah-Hartman az eszközkezelők fejlesztői számára elkészített egy új, csillogó villogó csomagot, amely valószínűleg a más operációs rendszerekkel kapcsolatos hasonló termékekkel is felveheti a versenyt. Az anyagra természetesen a ***GPL*** vonatkozik és egy ***CD*** lenyomat formájában tölthető le. A lemez rengeteg forráskódot és számtalan dokumentációt tartalmaz, sőt megtalálható rajta a ***Linux Device Drivers*** című könyv harmadik kiadásának elektronikus változata is.
Az elkövetkező kiadásokban már lesz egy kereshető tárgymutató is, amivel a teljes dokumentációt átfésülhetjük. Bizonyos hírek szerint a dokumentumok közé idővel bekerül majd az ***lwn.net*** portál kernelhírekkel kapcsolatos szekciója is. A kernelfejlesztők közül nagyon sokan szeretnék, ha az anyag később valamiféle kivételesen hatékony információforrássá nőné ki magát. Ennek megfelelően számos javaslat érkezett a továbbfejlesztésével kapcsolatban. Kétségtelen, hogy a projekthez nagyon sokan fognak csatlakozni, a keletkező csomag pedig egy komoly hiányt pótol.

Chris Wedgewood a közelmúltban egy érdekes problémába ütközött, amikor megpróbálta eltávolítani egyes kernelszolgáltatások mellől a ***„kísérleti"*** (***exeprimental***) jelzőt. Ennek a jelzésnek egyrészről az a küldetése, hogy figyelmeztesse a felhasználókat az esetleges instabilitásokra, másrészről meg az, hogy jelezze, ezzel a dologgal lehet szabadon kísérletezni, ha valakinek éppen ez a szándéka. Ugyanakkor a „kísérleti" jelző használatának egészen eddig nem voltak jól meghatározott szabályai. A fejlesztők a saját belátásuk szerint dönthettek úgy, hogy használják vagy nem, így aztán érthető, hogy nem nagyon volt két egyforma döntés. A dolognak aztán az lett a következménye, hogy rengeteg, amúgy köztudottan teljesen jól működő kód mellett ott virított az ***„experimental"*** jelzés. Ez eddig önmagában még nem is lett volna gond, ha mindez nem vezetett volna a jelzés komolyságának megszűnéséhez. Számos olyan felhasználó van, aki a kernel fordításakor egyszerűen alapértelmezésként bekapcsolja a kísérleti modulok befordítását, gondolván hogy úgysem lesz abból semmi baj. A kernelfejlesztői közösség szokásait ismerve most egy hosszas vitának kell következnie arról, hogy mi is a „kísérleti" kifejezés jelentése, mikor használható, és mikor nem. Aztán ha már mindenki jól kivitatkozta magát, és jól megsértődött, akkor előjön ***Linus Torvalds*** valamivel, ami teljesen új oldaláról világítja meg a problémát. És hogy ekkor mi fog történni? Kétlem, hogy ezt ma bárki meg tudná mondani.
A ***Linux*** érdekes módon egészen eddig nem felelt meg a ***POSIX*** szabványnak a gépnevek hosszára vonatkozó előírásainak. Éppen ezért ***Randy Dunlap*** nemrég beküldött egy foltot, ami a „bedrótozott" 64 karakter hosszúságú nevek helyett lehetővé teszi a ***POSIX*** által előírt 255-es hosszat (amely a karakterláncot lezáró nulla bájton felül értendő). Bár ***Linus Torvalds*** többször hangot adott annak a véleményének, hogy a ***Linux*** fejlesztőinek elsősorban az értelmes

működésre kell koncentrálniuk, nem pedig vakon követniük a *POSIX* vagy más szabványok előírásait, azért ez a mostani egy elég alapvető hibának tűnik. *Randy* foltja tehát 2.6.17-es kernelváltozattól érvénybe lép. A *Linux* kernel jelenleg legfeljebb öt egymást követő szimbolikus link használatát teszi lehetővé, de ez a szám a 2.6.17-es változattól kezdve valószínűleg kilencre fog emelkedni. *Alexander Viro* már 2006 februárjában bejelentette, hogy célszerű lenne ezt a változtatást elvégezni, és a közelmúltban, amikor a téma újra felmerült, a megfelelő foltot is elkészítette. *Andrew Morton* kelletlenül ugyan, de hozzájárulását adta azzal a megjegyzéssel, hogy a módosítás következményeként egyes alkalmazások visszirányú kompatibilitása nyilván meg fog szűnni, vagyis ha egy alkalmazás ötnél több tagból álló hivatkozássorozatokra építi a működését, akkor egyszerűen képtelen lesz futni a 2.6.17-esnél régebbi kernelekkel. Ugyanakkor az is igaz, hogy az ötös limit felemelése már igazán időszerű volt, így *Andrew* is elismerte, hogy a 2.6.17-es változatban való felbukkanása tulajdonképpen amolyan történelmi szükségszerűség. Ráadásul amint arra a vita során egyesek rámutattak, egyes terjesztések, mint például a *Red Hat* már hónapokkal ezelőtt elvégezték ezt a módosítást a saját egyedi fejlesztésű rendszermagjukon.

Jared Hubert egy beágyazott rendszereken, különösen telefonokon használható speciális fájlrendszeren dolgozik. Ez az *AXFS*, amely az *Advanced XIP Filesystem* rövidítése. És hogy mi az a *XIP*? Nos, az pedig az *eXecute In Place* rövidítése. Összefoglalva tehát az *AXFS* egy olyan fájlrendszer, amely az alkalmazásokat elindítás előtt nem tölti be a *RAM*-ba, hanem „helyből”, magáról a statikus tárolóról futtatja őket. Ennek a megközelítésnek bizonyos esetekben számos előnye lehet, így nem csoda, hogy egyes fejlesztők, például *Mark Lord* máris élénk érdeklődést tanúsítanak az új szolgáltatás iránt. Az *AXFS* egyik haszna, hogy a programok indítása gyorsabb lesz általa, hiszen a rendszer megspórolhatja a programkód memóriába való átmásolását. A másik nagy előny egyenesen következik abból a tényből, hogy a *RAM*-alapú rendszerek a programokat eleve *Flash* memóriában tárolják tömörített állapotban. Amikor elindítunk egy alkalmazást, annak kódját a rendszer előbb kitömöríti a memóriába, csak aztán indul a végrehajtás. Ha tehát egy ilyen eszközön az *AXFS*-t akarjuk használni, akkor nem használhatunk tömörítést, vagyis nagyobb *Flash* memóriára lesz szükség. Ugyanakkor a készülék akár egyetlen bájtnyi *RAM* nélkül is működőképes lesz, ami összességében csökkenti az előállítás költségét. Mindezek mellett azért *Jared* rámutatott, hogy az *AXFS* még messze nincs kész, az eddigi eredményeket pedig alapvetően azért tett közzé, mert véleményeket szeretett volna hallani, illetve további ötleteket kapni a kernelfejlesztőktől.

Linux Journal 149. szám

Zack Brown

Megerősített hátország a Novell SUSE Linux Enterprise Serverrel

Ha az informatikai életben a nagyobb teljesítményről, jobb méretezhetőségről, felügyelhetőségről, megbízhatóságról, nagyobb biztonságról, jobb támogatásról beszélünk és a vállalatokat erről az oldalról közelítjük meg, a Novellt az elsők között kell említenünk.

A vállalat legújabb termékcsaládja, a ***SUSE Linux Enterprise 10*** a cégek számára sokkal többet kínál a világszínvonalú vállalati szerveroperációs rendszerektől elvárt funkciókból. Mint minden új kiadás esetén, a platform kiszolgálókomponense, a ***SUSE Linux Enterprise Server*** természetesen több funkcióval rendelkezik mint elődje, azonban a kiszolgáló három lényeges bővítése az, amitől a szerver nagyot lép előre az egyre nagyobb igényt támasztó vásárlók szemében.

- *AppArmor* alkalmazásbiztonság
- Tárolási alapok
- A kiszolgálók virtualizációja

Az AppArmor megerősített alkalmazásbiztonságot nyújt

A biztonság mindig is a *Linux* operációs rendszer egyik erőssége volt, de az egyes alkalmazások sérülékenysége néha kaput nyit a betörők előtt. A ***SUSE Linux Enterprise*** termékcsaládba teljes mértékben integrált *AppArmor* alkalmazásbiztonsági keretrendszer segít e probléma megoldásában, mivel egy speciális biztonsági csomagolást készít minden egyes alkalmazás köré.

Az *AppArmor* nem teljesen új a ***SUSE Linux Enterprise Server*** felhasználói számára. A korábbi, 9-es verzió ugyanis már tartalmazott egy nyílt forráskódú kernelmodult és egy biztonsági egyeztetést kezelő komponenst, azonban az irányelvek létrehozásának és értelmezésének lehetősége külön, még egyedi megoldásként volt kapható. Az *AppArmor* mostanra teljesen nyílt forráskódúvá vált (a ***GNU General Public License*** szerint) és szorosan integrálódik a ***SUSE Linux Enterprise*** keretrendszerbe.

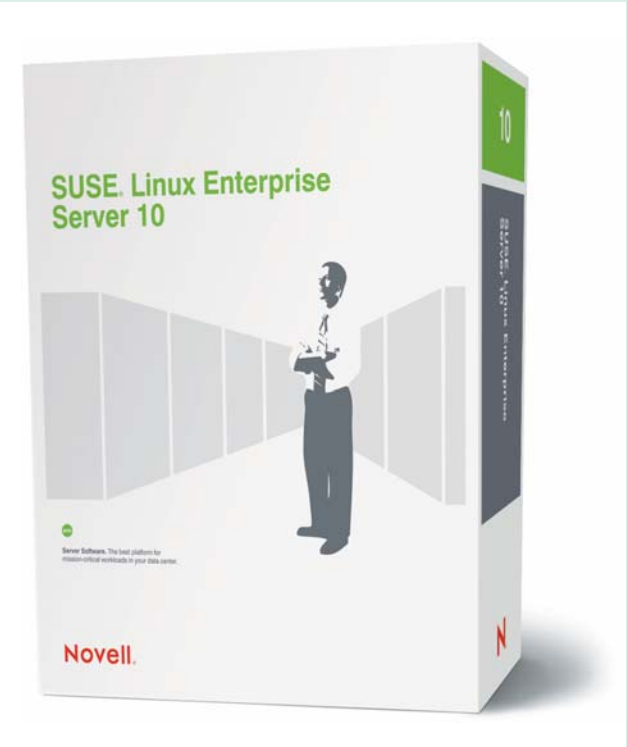
Az *AppArmor* a vállalati alkalmazások védelmére egy „fehérlista” elvet alkalmaz, ahol fel van tüntetve, hogy az alkalmazások milyen műveletek végrehajtására jogosultak. Ez a módszer sokkal hatékonyabb a „feketelistás” módszereknél, ahol az van felsorolva, hogy az alkalmazásnak mit *nem* szabad csinálnia. A feketelistás módszerek ugyanis csak addig működnek biztonságosan, amíg új támadási pontok nem válnak ismertté. Ilyenkor a felhasználók védtelenek, ameddig a gyártók el nem készítik az új javításokat.

Az irányelvek – a gyakorlatban egyszerű szövegfájlok – létrehozásuk után egyszerűen szétoszthatók a környezet más olyan kiszolgálóira, amelyek ugyanazokat az alkalmazásokat futtatják és ugyanazokat az irányelveket igénylik. Az *AppArmor* frissítésiprofilkészítő varázslójával egyszerűen frissíthetők a meglévő alkalmazás-irányelvek, vagyis könnyen implementálhatók a változások és vehetők fel az új szabályok. Miután az alkalmazás-irányelv életbe lépett, az *AppArmor* naplózza az irányelv által visszautasított alkalmazáseseményeket. A profilkészítő varázslóhoz hasonlóan a frissítésiprofil-készítő varázsló is átnézi a naplót és kérdéseket tesz fel oly módon, hogy könnyen ki tudja egészíteni a meglévő irányelvet.

A felügyelhetőség terén újdonság a ***Novell Customer Center*** (ügyfélközpont), ahol egyszerűen kezelhetők a rendszer-előfizetések. A ***Customer Center*** egy központi online portál, ahol a műszaki támogatás mellett a szoftverfrissítések, hibajavítások is egy helyen elérhetőek, és amely teljes mértékben együttműködik a ***Novell ZENworks® Linux Managementtel***. A ***Novell ZENworks Linux Management*** segít a ***Linux*** kiszolgálók és asztali gépek felügyeletében, így egészen nagy felhasználói bázis is kezelhető, és szabályozható a hozzáférés a hálózatokhoz és alkalmazásokhoz az ***OpenLDAP*** segítségével. A ***SUSE Linux Enterprise Server*** maximálisan integrálható Novell eDirectory környezetekbe, de ***Active Directory*** infrastruktúrákba is.

Robusztusabb, jobban méretezhető és nagyobb rendelkezésre állású tárolóeszközök

A ***SUSE Linux Enterprise Serverben*** található új tárolóeszköz-kezelő alaprendszer most többet nyújt, hiszen robusztusabb és jobban kezelhető



alapot biztosít, amely képes kiszolgálni kis fájlrendszereket, vagy akár fájlok millióit több terabájtnyi tárterületen. Emellett alkalmazások széles körét is kiszolgálja a webes alkalmazásoktól kezdve egészen az adatbázisokig, és a korábbiaknál nagyobb rendelkezésre állást garantál a továbbfejlesztett clusterkezelési funkciókkal és a speciális cluster-fájlrendszerrel.

Virtuálisan még több kiszolgáló

A ***SUSE Linux Enterprise Server*** disztribúció talán legizgalmasabb újdonsága a kiszolgálóvirtualizáció lehetősége. A ***Cambridge Egyetem*** által működtetett ***Xen*** nyílt forráskódú projektre épülő kiszolgálóvirtualizáció segít abban, hogy az alkalmazásokat, szolgáltatásokat és fájlrendszereket ne kelljen egy-egy adott géphez rendelni.

A virtualizáció előnye igazán akkor látszik, ha több önálló virtuális gép is fut egyetlen fizikai kiszolgálón. Ez lehetővé teszi a terhelések elszigetelését, vagyis ahelyett, hogy több alkalmazás futna ugyanazon a „túlhizlalt" operációs rendszeren, az egyes alkalmazások elszigetelhetők és a saját virtuális gépen futtathatók. Ha egy alkalmazás lefagy, – mivel el van szigetelve –, egyáltalán nem befolyásolja a többi szolgáltatást és alkalmazást, amelyek a saját virtuális gépeiken belül futnak ugyanazon a kiszolgálón. Ha egy virtuális gépen csak egyetlen alkalmazás vagy szolgáltatás fut, elegendő csupán az operációs rendszernek azokat a szolgáltatásait és komponenseit betölteni, amelyekre az alkalmazásnak ténylegesen szüksége van.

Így a fejlesztők és integrátorok számára lehetőség nyílik arra, hogy speciális virtuális gépeket készítsenek megoldásaikhoz.

A kiszolgáló virtualizációja újabb szintet ad hozzá a magas rendelkezésre álláshoz: a meghibásodott szolgáltatás automatikus újraindítását. Ez azt jelenti, hogy az alkalmazás vagy szolgáltatás csak egy rövid időre áll le, általában nem elég hosszú ideig ahhoz, hogy komoly problémát jelentsen.

A ***Virtual Machine Migration*** (virtuális gép áthelyezése) funkció lehetővé teszi egy adott virtuális gépen futó alkalmazás vagy szolgáltatás áthelyezését az egyik fizikai gépről a cluster egy másik gépére, újraindítás nélkül. Ez azt jelenti, hogy nincs leállás és az alkalmazás futási állapota teljesen megőrzésre kerül áthelyezés közben. Ez nagy előny, mert így éles működés közben is elvégezhető a rendszer normál karbantartása.

Ha a kiszolgálók konszolidációjáról van szó, a több virtuális gép futtatása egyetlen kiszolgálón, valamint az a tény, hogy a virtuális gépek akár más-más vendég operációs rendszereket is futtathatnak, nagymértékben megkönnyítheti a felhasználók dolgát. Legyen bár a ***SUSE Linux Enterprise Server*** a gazda operációs rendszer a kiszolgálóvirtualizációhoz, a virtuális gépek maguk más paravirtualizált vendég operációs rendszereket is futtathatnak. (A paravirtualizáció egy olyan virtualizációs megoldás, ahol a hardverhez nagyon hasonló, bár azzal nem teljesen megegyező szoftverfelületet „látnak" a virtuális gépek.) Vagyis az olyan meglévő alkalmazások, amelyeknek muszáj egy adott régebbi operációs rendszeren futni, elszigetelhetők a saját egyéni virtuális gépükbe, ugyanakkor mégis futhatnak ugyanazon a fizikai kiszolgálón. A ***SUSE Linux Enterprise Server*** az ***AMD*** és az ***Intel*** hamarosan elkészülő hardvertechnológiáival együtt képes lesz támogatni a teljes virtualizációt.

A ***SUSE Linux Enterprise Server*** teljesen virtualizált és paravirtualizált vendég operációs rendszer támogatást is biztosít. A ***Novell*** még ebben az évben tervezi a paravirtualizációs támogatást a ***SUSE Linux Enterprise Server 9 SP3***-hoz, valamint az ***Open Enterprise Server*** környezetben futó ***NetWare*** támogatását.

A hivatalosan bejelentett támogatásokon túl, a nyílt forráskódú közösségnek a ***Xenen*** paravirtualizált vendég operációs rendszerként futó kernelek között megtalálható a ***Linux 2.4***, a ***Linux 2.6***, a ***NetWare 6.5***, a ***NetBSD***, a ***FreeBSD***, a ***Plan9*** és az ***OpenSolaris***. Bár a vendég operációs rendszerek bejelentett támogatása egyelőre korlátozott, a ***Novell*** célja, hogy a ***SUSE Linux Enterprise Serveren*** futó ***Xen*** legyen a kapható legjobb virtualizációs platform. A jövőben tehát további újdonságokra számíthatunk még.

AppArmor-közösség

Annak érdekében, hogy még egyszerűbb legyen minden szervezet számára az ***AppArmor*** kínálta alkalmazásbiztonsági előnyök kiaknázása, a ***Novell*** meghirdetett egy nyílt forráskódú projektet, ahová mindenkit szívesen vár, hogy járuljon hozzá az AppArmor jövőbeni fejlesztéséhez, valamint küldjön el a saját alkalmazásaihoz készített ***AppArmor*** profilokat. A cél előredefiniált alkalmazásbiztonsági irányelvek nagy tárházának kialakítása, amely mindenkinek segíthet az ***AppArmor*** gyors és egyszerű beüzemelésében az informatikai környezetek védelméhez. További információ a projekttel kapcsolatban: ➲ opensuse.org/apparmor.

Mesterséges mesterséges intelligencia

Érdekesen valósítja meg a mesterséges intelligenciát az Amazon.com. Ebben a cikkben a közelmúltban bemutatott Mechanical Turk nevű szolgáltatást tekintjük át röviden.

Az *Amazon.com* igen aktívan keresi az új utakat. Nem csak világméretű virtuális vásártér szeretne lenni, hanem annál sokkalta több. Létrejött egy külön részleg az *Amazonon* belül, mely különböző webszolgáltatások kidolgozásán dolgozik. Egy webszolgáltatást úgy képzelhetünk el, mint egy távoli eljáráshívást. Mi programozók írhatunk olyan programot, amely az *Amazon.com* távoli szervereivel cserél adatot, végeztet el különböző speciális műveleteket, melyeket mi nem tudnánk magunktól elvégezni.

Kempelen Farkas visszatér

A szolgáltatás weboldala (➲ *http://www.mturk.com*) megdobogtatja a magyar programozók szívét, hiszen a következő sorokat olvashatjuk bemutatkozás gyanánt: ***„1769-ben egy magyar főnemes, Kempelen Farkas bámulatba ejtette Európát egy mechanikus sakk automatával, mely szinte minden alkalommal megszerezte a győzelmet ellenfeleivel szemben.”*** A szöveg további részéből kiderül, hogy *Kempelen Farkas* bejárta a korabeli főúri udvarokat, és elmés szerkezete még az oly messze földön híres játékosokat is térdre kényszerítette, mint *Benjamin Franklin*, vagy *Napoleon Bonaparte*.

Mi volt a titka a híres szerkezetnek? Állítólag az, hogy a fogaskerekek és a mechanikus alkatrészek mögött egy hús vér sakk nagymester rejtőzött. Érdekes, hogy *Kempelen* olyan ügyesen építette össze a szerkezetet, hogy erre a trükkre a kortársak ténylegesen soha nem jöttek rá így – pusztán elméleti síkon – nem is lehetünk benne teljesen biztosak, hogy így volt. Innen kapta a nevét tehát az *Amazon* mesterséges mesterséges intelligenciája. Számos olyan problémakör van ugyanis a számítástechnikában, mely egy gépnek szinte megoldhatatlan, viszont egy ember pillanatok alatt megadja a választ. Kinek okozna például gondot, ha azt kellene eldöntenie, hogy egy adott képen szerepel-e banán? Gondolom senkinek. Viszont ha ezt a feladatot számítógéppel szeretnénk elvégeztetni, akkor speciális programot kellene írnunk, mely rengeteg számítás elvégzése után sem tudná egészen bizonyosan megmondani az eredményt.

Hogyan működik?

Manapság az ember-számítógép interakció a következőképpen működik: az ember megfogalmaz kérdéseket, majd a számítógép a jól feltett kérdéseket hosszas számítások árán megválaszolja. Miért ne lehetne megfordítani ezt a dolgot? Miért ne tehetne fel a számítógép is kérdéseket hús vér embereknek?

Az *Amazon Mechanical Turk*-je pontosan ezt csinálja. Biztosít egy olyan programozói felületet melyen keresztül pillanatokon belül kérdéseket küldhetünk az *Amazon* szervereinek. Ott ezeket a kérdéseket emberek böngészhetik, akik egy kérdés megválaszolása után megkapják az általunk felajánlott díjat. A szolgáltatásnak mindössze 10% közvetítői díja van.

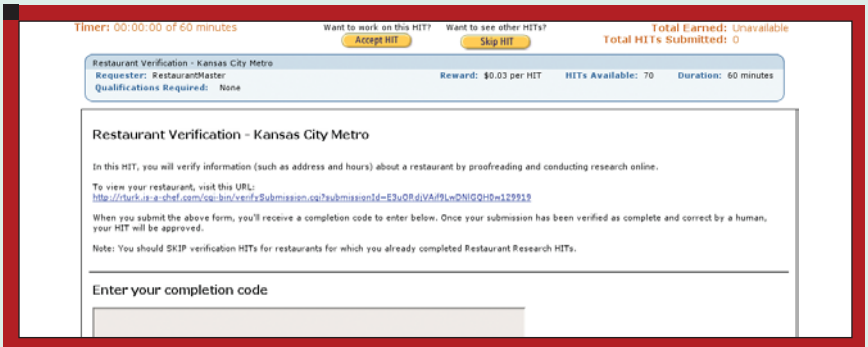
A programozó szemszögéből nézve

Nézzük meg, hogyan néz ki mindez a programozó szemszögéből! Az alábbi pszeudokód kiválóan szemlélteti mennyire egyszerű szolgáltatásról van szó:

```
read (photo);
photoContainsHuman =
➥callMechanicalTurk(photo);
if (photoContainsHuman ==
➥TRUE){
  acceptPhoto;
}
else {
  rejectPhoto;
}
```

A fenti program arra keresi a választ, hogy vajon a képen szerepel-e ember. Meghívja tehát az *Amazon Mechanical Turk*-jét, melynek segítségével pillanatok alatt megkaphatja a kérdéses eredményt.

1. ábra Egy munka rövid leírása



2. ábra Így néz ki egy konkrét munka

Amit a válaszadó lát

Vajon mit lát egy felhasználó, aki a kérdések megválaszolásával pénzt szeretne keresni? A szolgáltatás weboldalát (➲ *http://www.mturk.com*) böngészve válogathat a különböző kérdések közül. Minden kérdés mellett szerepel, a kérdést feltevő cég neve, a munka határideje, hogy mennyi idő áll rendelkezésünkre, valamint, hogy mennyi pénzt kereshetünk a megválaszolásával.

Rákattintva egy kérdéscsoportra, egy konkrét munka leírása elé jutunk, melyhez akár rögtön hozzá is kezdhetünk. Az illusztrációnak használt munkaleírás a következőket tartalmazza: meg kell látogatnunk egy étterem honlapját, majd ellenőriznünk kell, hogy a honlapon feltüntetett és a regisztrációjuk során megadott nyitvatartási idő azonos-e. Nem túl bonyolult kérdés, ugye? Valószínűleg mégsem mi fogjuk megválaszolni őket, hiszen a jutalmul felajánlott majdnem hét forintos díjazás még akkor is kevésnek tűnik, ha óránként 20-30 darab kérdést tudunk megválaszolni.

Természetesen nem ilyen rossz azért a helyzet, hiszen minden programozó eldöntheti, hogy kiknek jeleníti meg a kérdéseket. Ezért valószínű, hogy a tapasztaltabb, több munkát teljesített, vagy valamely speciális tudással(például nyelvtudás) rendelkező felhasználók jóval magasabb díjazásra számíthatnak.

Összességében egészen érdekes megoldással rukkolt elő az *Amazon*. Sok lehetőséget látok a *Mechanical Turk*-ben, és azt hiszem főleg azoknak lesz érdemes figyelemmel kísérni a sorsát, akik valamilyen összetettebb, intelligens szolgáltatás beindításán gondolkoznak.

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.

Vörös tanga és japán múzeum

Azt tudjuk, hogy ruha teszi az embert, és hogy a mezítelen emberek kevés vagy semmilyen befolyással nem bírnak társadalmunkban. De hogy miként befolyásolja az embert egy (vagy több) pingvin szmokingban, gatya nélkül – az elvarratlan szálakhoz keressünk vásznakat.

A dott ugye a félig hiányos, de másfelől ünnepi öltözet, melyben – legalábbis egy karrierista pingvin számára – éppen illő végigsuhanni a vörös szőnyegen. A filmvászonra kerülve rögvest érvényesül az emberi befolyás, tudniillik, hogy tán egy madártollal sok erotikus dolgot lehet művelni, azonban egy egész madárral e dolgok más jelzőért

kiáltanak. Így például a *Farce of the penguins* című amerikai alkotás (mely az *Oscar*-díjas *Pingvinek vándorlása* c. film paródiája) igaz szerelmet kutató főhősét csípőszorítós, kitömött vöröstangás, *push-up* melltartós, bőrszerkós, bilincscsörgéses-ostorsuhintgatós társai kísérik végig történetén. Ennél elvetemültebbek a német családi vígjátékok, melyekben hol postai munkára, hol a halászati feldolgozóiparba kényszerítik a pingvineket, némelykor konyhai robotra fogják őket.
Már *Hitchcock* is megmondta, könnyű a rajzfilmkészítők dolga szereposztás tekintetében, hiszen ha nem tetszik egy-egy figura, egyszerűen széttépik. No, ezt nem tették *Batmannel*, így megszületett a gonosz

Pingvin Danny DeVito személyében, kit narkomán tekintete dacára sem vettek be a Farce-csapatba – túlöltözöttsége miatt.
A *Warner Bros* erre föl mentőakciókat szervez, *Scooby Doo* és csapata próbálja visszajuttatni otthonába a sarkvidéki madarat, halált megvető bátorsággal küzdve a zöld megmozdulást rossz (és bandzsa) szemekkel néző *Iszonyatos Uszonyos* nevezetű figurával. Megint más pingvineket valamiért női ruhákba öltöztetnek és hercegnői rangra emelnek, mindemellett egy másik öltözet igazságtartalmát kutatják utazva és vitázva – nevezetesen a déli féltekéről az északira haladva merül felfelé a kérdés a szárnyasokban: vajon létezik-e a *Mikulás*. A *Toy Story*-ban a dráma tovább fokozódik; elnémult pingvint árusít ki jóanyja. E gyerekkereskedelmes gyermekfilmet felnőtteknek is ajánlják, bár én még mindig nem értem az áldozat, annak felmenője, továbbá a feslettkarú cowboy, a ló és a japán múzeum összefüggéseit…
Ugyanígy ámulattal követtem *Wallace* és *Gromit* kalandjaiban

a főgengszter pingvint, ki albérlő, de eközben gyémántokat rabol és fegyvernek látszó tárgyakkal fenyegetőzik. És jön oldani a feszültséget a madagaszkári állatbuli, melyben ama fontos kérdésre kaphatunk feleletet: Hús vagy hal?

Halusz Léna

Sugárkövetés (1. rész)

A sugárkövető algoritmusok szépsége, ereje a fény viselkedésének szimulációjában gyökerezik. Az olyan effekteket, mint a tükröződés vagy az árnyékok – amelyeket nehézkes scanline algoritmusokkal élethűen modellezni – a sugárkövetés természetes eredményei. Viszonylag könnyű implementálni és impresszív vizuális élményt nyújt. Most induló cikksorozatunk segítségével elkalauzoljuk olvasóinkat a ray tracing és a fotorealisztikus modellezés világába.

A klasszikus sugárkövetés összehasonlítása a hagyományos scanline algoritmusokkal

A sugárkövetés hátránya a viszonylag magas számítási idő. Míg a *scanline* algoritmusok kihasználják a fény koherenciáját, a sugárkövetés minden egyes sugárra külön értelmezi a műveletet, az előző sugár eredményétől függetlenül. Előnye, hogy nem csak poligon alapú felületekkel képes dolgozni, képes a fény és a különböző anyagok viselkedésének valósághű modellezésére.

A sugárkövetés (idegen szóval ***ray tracing)*** – mint arra a neve is utal – alapvetően a fotonok útjának követésén alapul, egy három dimenziós koordinátarendszerben matematikai függvényekkel leírt környezetből a fénytan törvényeinek megfelelően fotorealisztikus képet számol ki.

A mai grafikus ún. ***scanline*** algoritmusokkal dolgoznak. Bár ezeken a ***pixel*** és ***vertex shaderekkel*** felvértezett kártyákon is nagyon élethű effekteket lehet létrehozni (láttam már ***shader language***-ben írt sugárkövetőt is), ezek használata még mindig nem közelíti meg a sugárkövetéssel számolt képek minőségét. Jelen cikk írója szerint a poligon alapú grafikus gyorsítókat a jövőben fel fogják váltani a sugárkövetéssel dolgozó kártyák. Már léteznek ***raytracing*** kártyaprototípusok, bár kereskedelmi forgalomban még nem kaphatóak és hatalmas számításigényük miatt még gyermekcipőben járnak a fejlesztések. Egyelőre be kell érnünk a szoftveres sugárkövetéssel.

Természetesen léteznek Linux alá írt ingyenes sugárkövető programok, mint például a POV-Ray, és akár ezek használatát is ismertethetnénk e hasábokon... mi azonban eltekintünk ettől, ebben a cikksorozatban azt a célt tűzzük ki magunk elé, hogy az olvasókat az algoritmus mélységeibe kalauzoljuk, nem elégszünk meg a felszínes ismeretekkel.

Sajnos a témáról magyar nyelvű irodalom az interneten alig található, ezért igyekszem majd az egyes fogalmak angol megfelelőjét is megadni, így az olvasók könnyebben tudnak majd kapcsolódó publikációkat, írásokat keresni. Ebben a számban a matematikai alapokra is kitérünk, ezek ismerete létfontosságú lesz a továbbiak megértéséhez.

A szem működése

Kezdjük is mindjárt az alapoknál. Hogyan is működik az emberi szem? A szem a fény energiáját a retinában lévő idegelemek segítségével elektromos és kémiai ingerületté alakítja majd ezeket az ingerületeket a tényleges idegrendszer vezető elemei

1. ábra Ezen a képen jól látszanak a sugárkövetés jegyei: tükröződés, fénytörés, árnyékok

(látóidegek) az agy nyakszirti lebenyének meghatározott részéhez vezetik, ahol az elektromos információ bonyolult idegi mechanizmusok révén dekódolódik és képi információvá alakul. Szemünk optikai tulajdonságai révén fordított állású képet készít, amit az agy fordít vissza. A szem optikai része (szemlencse, üvegtest, stb.) a beérkező fényt egy fókuszpontba „tereli össze", ez a pont az ideghártya központi, kiemelt jelentőségű területe, az ún. sárgafolt (***makula***), ez az éleslátás tényleges kialakulásának a helye. A sárgafoltban megközelítőleg 127 millió receptorsejt szolgál a fényinger felvételére és kémiai ingerré alakítására.
Három fajta érzékelő sejtet lehet megkülönböztetni: a vörös, a kék és a zöld színekre érzékenyeket.
E három színből az összes többi előállítható, ezt a színkeverési metódust ***additív*** színkeverésnek nevezzük.
Az additív színkeverés tehát a szemünkben történik, ez a színes televízió, a monitorok és egyéb színes kijelző készülékek működésének alapja.
A szemünkbe egyidejűleg, a szem tehetetlenségét kihasználva gyors egymásutánban érkező, vagy egymáshoz igen közeli apró pontocskák formájában beérkező ***trikromatikus*** (három különböző hullámhosszúságból álló) színingerek additív színkeverés útján egyetlen színné alakulnak.

Egy kis matematika

Most, hogy túlestünk a látás anatómiai részén, kicsit megtornáztatjuk a bal agyféltekét, rátérünk a matematikai alapozásra. Azon olvasóink, akik e matematikai alapokkal teljesen tisztában vannak, akár át is ugorhatják ezt a részt. Az egyszerűség kedvéért legtöbbször vektoriális egyenleteket adunk meg, tehát feltételezzük az elemi vektormatematika ismeretét. Néhány műveletet speciálisan jelölök majd a formulákban, ezek: * = skalár szorzás (két vektor esetén a lentebb ismertetett skalár szorzat), sqrt = négyzetgyök, ^ = hatványozás (programozásban jártasabb olvasóinknak már ismerősek lehetnek e jelölések). A vektorok egy-egy komponensére a nagy betűvel szedett vektor neve utáni kis betűs komponensjelöléssel hivatkozok, azaz például Vx a V vektor X koordinátájára utal.
Egy három dimenziós koordinátarendszer X,Y,Z tengelyekből áll, ahol az egyes pontok helyvektorait a tengelyeknek megfelelő bázisvektorok (i,j,k) valamint adott skalár mennyiségek (X,Y,Z koordináták) szorzataként létrejövő vektorok összegeként definiálhatjuk, azaz P = X*i + Y*j + Z*k.
Három dimenziós koordinátarendszerben egy egyenes térbeli pontjait a következő egyenlettel definiálunk:
P = P0 + V*t

Ahol p0 az egyenesen fekvő adott pont helyvektora, V az egyenes irányvektora, t pedig egy tetszőleges skalár mennyiség. Bármilyen értéket adunk t-nek, a formulából számított P pont mindig az egyenesen fekvő pont lesz.
Az egyenes irányvektora az egyenes irányával párhuzamos vektor. Az irányvektor nagyon könnyen kiszámítható az egyenesen fekvő két pont ismeretében V = P1 - P0
Ha a kapott vektort normalizáljuk, akkor a fenti egyenletben t a P0 ponttól mért távolságot jelenti az egyenes mentén az irányvektor irányában. Egy adott vektor normalizálása egy olyan vektort eredményez, melynek iránya megegyezik az eredeti vektoréval, hossza egységnyi:
Vn = V / |V|
A három dimenziós vektor abszolút értékét (azaz hosszát) Pitagorasz tétele alapján egyszerűen kiszámíthatjuk a következő formula segítségével:
|V| = sqrt(X^2 + Y^2 + Z^2).
Amint azt a későbbiekben látni fogjuk, gyakran lesz szükségünk két vektor által bezárt szög kiszámítására. Ezt az ún. skaláris szorzat adja. Két vektor (A és B) skaláris szorzata egy olyan skalár érték, amely egyenlő a két vektor által bezárt szög koszinuszával.

A*B = cos(alfa) azaz cos(alfa) =
➥(Ax*Bx + Ay*By +
➥Az*Bz)/(|A|*|B|).

A gömb

A sugárkövetés metszéspontszámításokon alapul, futásidejének oroszlánrészét azzal tölti, hogy metszéspontokat keres fénysugarak (szakaszok) valamint a térbeli objektumok között.
A térbeli objektumokat, felületeket egyenletekkel kell definiálnunk ahhoz, hogy metszéspontokat tudjunk számolni. Az egyik legegyszerűbb felület a gömb (angolul ***sphere***).
Próbálkozzunk meg egy egyenes és egy origóban álló gömb metszéspontjának kiszámításával.
Tudjuk, hogy a gömb felülete azon pontok halmaza, amelyek egy adott ponttól (a gömb középpontja) azonos távolságban vannak. Ez a távolság a gömb sugara. Az origóban álló gömb pontjai ki fogják elégíteni a |P| = r egyenletet. Ha mind az egyenes mind

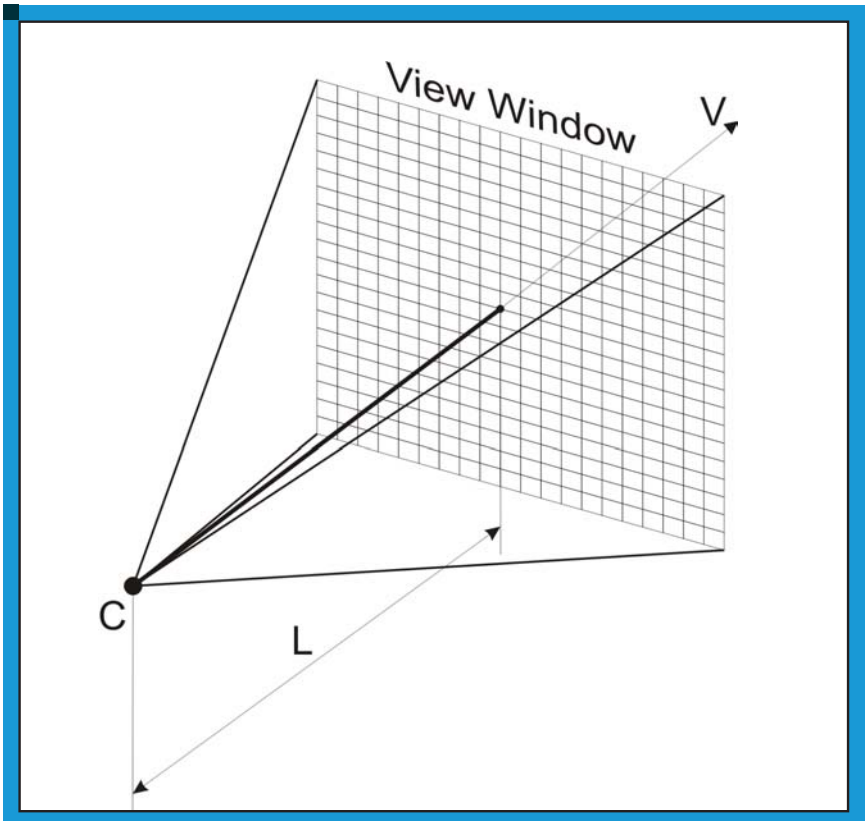
a gömb egyenletét koordinátákkal írjuk fel, a következő egyenletrendszerhez jutunk:

```
X = X0 + Vx*t
Y = Y0 + Vy*t
Z = Z0 + Vz*t
R = sqrt(X^2 + Y^2 + Z^2)
```

Az egyenes X,Y,Z koordinátáit rendre behelyettesítve a gömb egyenletébe majd a kapott egyenletet egyszerűsítve és a másodfokú egyenletek általános alakjára (a*x^2 + b*x + c = 0) hozva, a következő formulához jutunk:

```
(Vx^2 + Vy^2 + Vz^2)*t^2 +
➥(2*(X0*Vx + Y0*Vy + Z0*Vz))*t  +
➥(X0^2 + Y0^2 + Z0^2 – r^2) = 0
```

Ezek alapján a tetszőleges középpontú gömb egyenlete is levezethető, a tetszőleges középpontú gömb egyenlete |P – O| = r azaz r = sqrt((X – Ox)^2 + (Y – Oy)^2 + (Z – Oz)^2), ennek levezetését az olvasóra bízzuk.
A másodfokú egyenlet determinánsát kiszámítva (d = b^2 - 4*a*c) megtudjuk, hogy az egyenes metszi-e a gömböt (d nem negatív, azaz a másodfokú egyenletnek van megoldása) illetve hogy egy vagy 2 pontban metszi (amennyiben egy pontban metszi, ez az egyenes a gömb egyik érintője).
Az másodfokú egyenletek általános megoldóképletét (t1 = (-b + sqrt(b^2 – 4*a*c))/(2*a*b), t2 = (-b - sqrt(b^2 – 4*a*c))/(2*a*b)) felhasználva megkapjuk azon t értékeket, amelyet az egyenes egyenletébe behelyettesítve egy olyan pontot kapunk, amely a gömbfelület ponthalmazának része is egyben, azaz a gömb és egyenes metszéspontját!
Amint láttuk, a gömbfelületet egy másodfokú egyenlettel lehet definiálni, a gömb tehát egy másodfokú (kvadratikus) felület. A későbbiekben több másod és magasabb fokú felületet is ismertetni fogok, elégedjünk meg egyenlőre e felülettel.
Összegezve az eddigieket, most már ki tudjuk számítani fénysugarak és gömbök metszéspontját, azaz azt a pontot, ahol egy adott helyről indított fénysugár eltalál egy gömböt. Hamarosan látni fogjuk, hogy szükségünk lesz a felület metszéspontban számított normálvektorára is. A normál vektor egy olyan vektor, amely merőleges a felületre. Ez gömbök esetében rendkívül egyszerű, a normál vektor a gömb középpontjából a metszéspontba mutat, azaz: N = P – O. Amikor normálisokat számolunk, figyeljünk arra, hogy a normál vektor a felületből mindig kifelé mutasson!



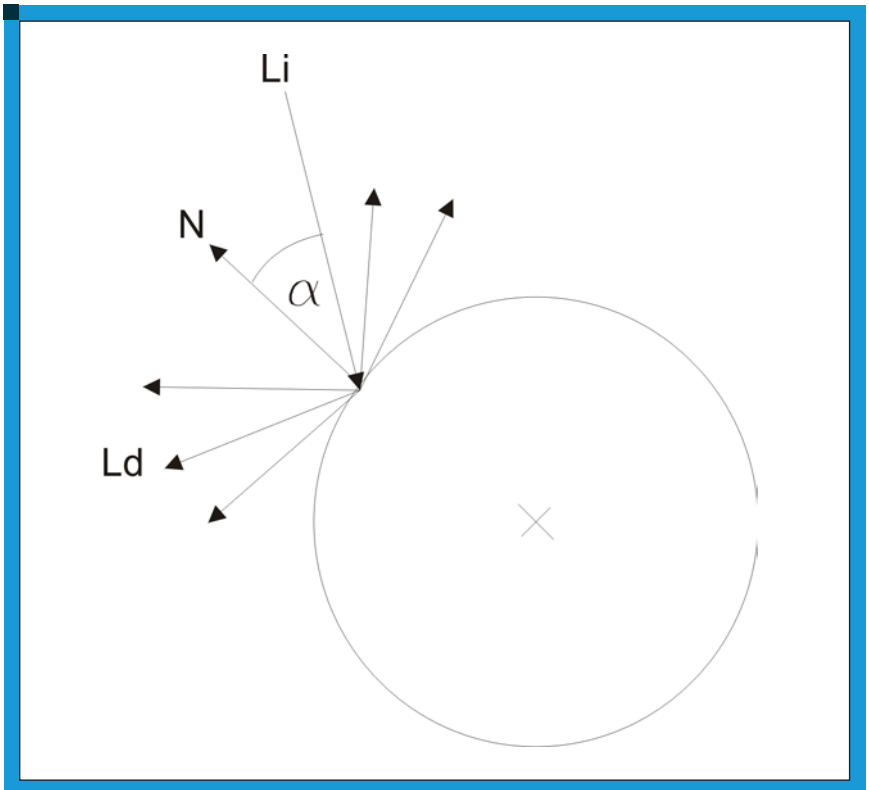
2. ábra A virtuális kamera felépítése és paraméterei

Fényforrások

A klasszikus sugárkövetés pontszerű fényforrásokat használ, a fotonok a tér egy definiált pontjából indulnak útjukra, azaz e fényforrásoknak nincs térbeli kiterjedése A valóságban ilyen fényforrások természetesen nem léteznek, hiszen minden fényt emittáló testnek kell legyen térbeli kiterjedése, ugye? Mi most ettől eltekintünk (későbbiekben még visszatérünk erre a témára), a számítások egyszerűsítése érdekében egyelőre csak pontszerű fényforrásokkal számolunk, azon beül is a legegyszerűbb esettel, irányítatlan, konstans fényforrással dolgozunk. Ez a fényforrás a tér minden irányába egyenlő mértékben bocsát ki fotonokat, a fényerősség pedig nem csökken a fényforrástól való távolság növelésével, tehát konstans a tér bármely pontjában. Az ilyen fényforrásnak mindössze két paramétere van: a kibocsátott fény színét és erősségét definiáló RGB hármas, valamint a fényforrás pozíciója.

Szemünk, a virtuális kamera

Ez mind szép és jó – mondhatja az olvasó – na de hogyan számoljunk ebből két dimenziós képet? A megoldás az emberi szem felépítését utánzó virtuális kamerában rejlik. A kamera pozícióját (a szem analógiájával élve a szemünk fókuszpontját) a tér egy adott pontjába helyezzük el, majd meghatározzuk azt az irányvektort, amerre a kamera néz (nézetvektor, idegen szóval ***view vector).*** *A* kamerapozíciótól adott távolságra pedig elhelyezzük a nézősíkot (***view plane***), amely merőleges a kamera ***view vector***-ára. A nézősík és a kamera pozíciója közti távolságot szemtávolságnak nevezzük, a továbbiakban L-lel jelölve. Ez a paraméter egy tetszőleges, 0-nál nagyobb érték, a perspektívát, kameránk látószögét szabályozhatjuk vele.



3. ábra A diffúz fény nézőpontfüggetlen, nagysága a beérkező fényenergia és a beesési szög függvénye

A nézősík az a sík, ahol a kiszámolt képünk keletkezik, ennek egy részén (***view window*** – nézőablak) fogjuk fel és tároljuk a beérkező fényenergiát. A nézőablak területét kis négyzetekre osztjuk fel oly módon, hogy minden egyes négyzet megfelel egy-egy pixelnek a számított képen, tehát a ***frame bufferünk*** minden egyes pixeléhez hozzárendeljük a nézőablak egy-egy celláját. Ha például egy 640x480 méretű képet akarunk számolni, a nézőablakot 640 oszlopra és 480 sorra osztjuk. A nézőablak méreteit ***FOV***-nak (***Field Of View*** – látómező) is nevezik a sugárkövetés terminológiájában. Az ***FOV*** – az L szemtávolsággal egyetemben – befolyásolja a perspektívát, kameránk látószögét. Az ***FOV*** áldásos hatása, hogy különböző felbontású képek számításakor minden esetben ugyanazt a térrészt fogja be kameránk (gondoljunk bele, ***FOV*** számítások használata nélkül nagyobb felbontású képek esetén a kameránk szélesebb látószöggel rendelkezne).
A sugárkövetés célja tehát nem más, mint a nézőablak minden egyes rácspontjára (azaz a ***bitmap*** minden egyes pixelére) kiszámítani a környezetből beérkező fényenergiát.

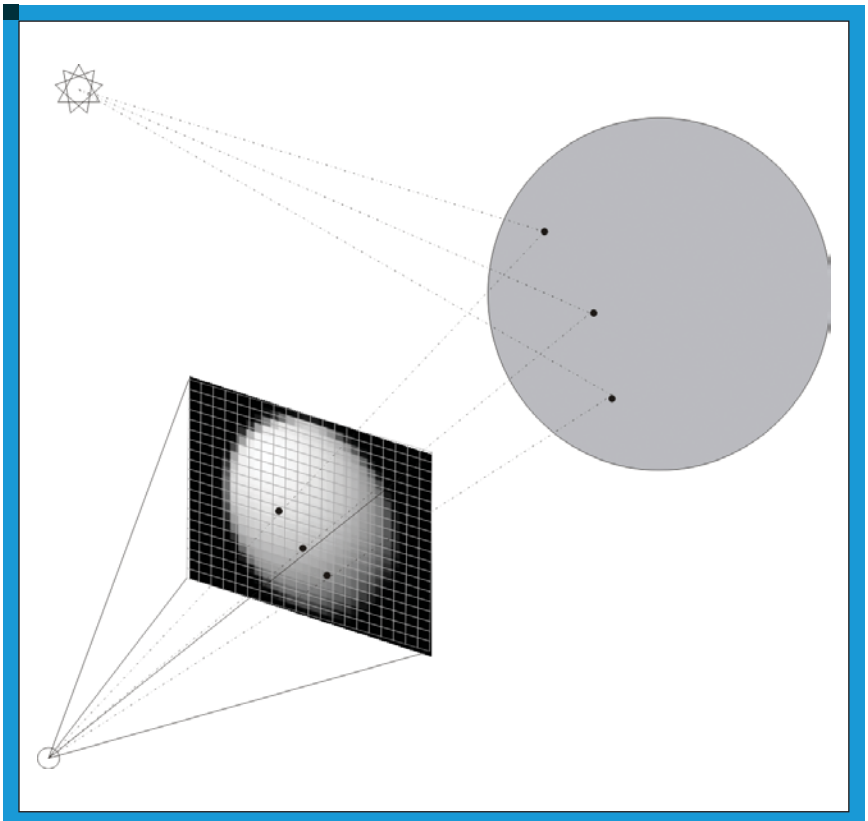
Anyag és fény

Gyorsan összefoglalnám az eddig tanultakat: fotonokat tudunk indítani a tér pontjaiból (fényforrások) és azon fotonokat, amelyek a kamerába nézősíkjába csapódnak be, fel tudjuk fogni, így kirajzolva a számított képet.
Igen ám, de hogy is van ez? Mi történik azokkal a fotonokkal, amelyek a fényforrásból egy objektumba csapódnak? A színes felületek színét az határozza meg, hogy a rájuk vetülő különböző hullámhosszúságú fényekből mennyit vernek vissza, illetve nyelnek el. A tárgyak színét tehát a megvilágítás színe is befolyásolja.
A különböző anyagok különbözőképpen reagálnak a fényre, ezért a modellezendő anyagtól (***material)*** függően kell a fény viselkedését leírni illetve modellezni. Mindjárt be is vezethetjük az árnyalási modell (***shading model)*** fogalmát, amely leírja, hogy egy felület milyen módon reagál a fényre.
Az árnyalási modell egy olyan függvény, amely az annak megfelelő anyagtulajdonságok alapján meghatározza, hogy az anyag felületére érkező fénysugár milyen reakciót idéz elő (a fényenergia mely részét tükrözi, melyiket szórja, melyiket nyeli el).

Több ilyen modell is létezik, egyesek a valóság egyre pontosabb megközelítésére születtek, míg mások csak egyetlen valós anyagtípus viselkedését próbálják meg modellezni. Az árnyalási modellek túlnyomó többsége elhanyagolja a fény hullámtulajdonságaiból adódó jelenségeket, azaz nem lehet velük például egy prizma viselkedését szimulálni. Egy már megírt és működő sugárkövető algoritmust némi hozzáértéssel módosíthatunk úgy, hogy figyelembe vegye a fény hullámtulajdonságait is (a fény spektrumát felpartícionálva több hullámhosszon is ki kell lőni ugyanazt a sugarat, valamint az árnyalási modelleket és a fényutak meghatározását is hullámhosszal paraméterezhetővé kell tenni), ennek megvalósítását azonban az olvasóra bízzuk.
Mivel ez a cikksorozat első része, az eddigiekben minden esetben a legegyszerűbb modellt ismertettük. Nem teszünk kivételt az árnyékolási modellekkel sem.
Lássuk tehát a diffúz, más néven a Lambert modellt. Lambert egyenlete egy tökéletesen diffúz, egyenetlen felületet vesz alapul, amely a tér minden irányába azonos fényerősséggel veri vissza a beérkező fényenergia egy részét. Ez azt jelenti, hogy a fényerősség nézőpontfüggetlen, azaz a tér bármely pontjából tekintünk egy felület adott pontjára, a visszavert fényerő konstans lesz. A diffúz fény függ a felület becsapódási pontban állított normálisától valamint e normális és a beérkező fénysugár által bezárt beesési szög koszinuszától:

Ld = Li*cos(alfa)

ahol Ld az eredményként kapott diffúz fény, Li a beeső fény, alfa pedig a már említett beesési szög. A diffúz árnyalási modellben szereplő együtthatókat a 2. *ábra* szemlélteti.
A diffúz árnyalási modellben egyetlen anyagtulajdonságunk van, ez pedig az anyag diffúz színe, melyet a már említett (és a számítástechnikában leggyakrabban használt) additív színkeverésnek megfelelően egy RGB hármassal definiálunk, 0 és 1.0 valamint a közéjük eső valós számokkal.
Itt jegyezném meg, hogy cikksorozatunkban mindvégig RGB hármasokkal dolgozunk, tehát amikor színre



4. ábra A diffúz árnyalással számolt kép kirajzolódik a bitmapben (a nézőablakban)

hivatkozok valahol, akkor az a szín mindenképpen egy RGB hármassal van definiálva. Ez a szín megadja, hogy az anyag felületére érkező fény mely részét veri vissza a környezetbe diffúz fényként, a következő képlettel számítható mindhárom összetevőre:

Ld = Cd*Li*cos(alfa)

Ez a modell kizárólag matt felületű anyagok modellezésére alkalmas, amelyek nem rendelkeznek tükörszerű fényvisszaverő vagy fényáteresztő tulajdonsággal.

Ray casting

Most már megvan mindenünk ahhoz, hogy egy egyszerű sugárkövetőt leprogramozzunk. Útjukra tudjuk indítani a fénysugarainkat a fényforrásból, ezen sugarak egy része az általunk definiált gömbökbe fog csapódni, s ezekről a diffúz árnyékolási modellnek megfelelő mennyiségű fény fog a kamera síkjára érkezni.
Ez eddig szuper, csakhogy figyelmen kívül hagytuk eddig azt a tényt, hogy a fénysugarak nagy része nem a kamera síkjára fog érkezni, legnagyobb részüket elnyeli a nagy semmi, ugye?

Ahhoz pedig hatalmas mennyiségű fénysugarat kellene kilőnünk és követnünk a fényforrásból, hogy a kamera síkján definiált rács pontjaira elegendő mennyiségű foton csapódjon be. Leszögezhetjük, hogy ez egy nagyon költséges algoritmus. Ennél lényegesen hatékonyabb megoldás lenne, ha a fénysugarakat visszafelé követnénk, azaz a kamerából indulnánk el a fényforrások felé. Ez utóbbi megoldást ***backward***, előbbit ***forward ray tracing*** -nek hívják szakmai berkekben. Mi főleg a ***backward ray tracing***-gel fogunk foglalkozni, de későbbi cikkekben még valószínűleg visszatérünk a ***forward*** megoldáshoz is egy-két szó erejéig.
A diffúz modellel megvalósított visszirányú (***backward***) sugárkövető megvalósítása tehát a következő fő lépésekből áll:

1. Felinicializáljuk a ***scene***-t, azaz definiáljuk a gömbjeinket (középpont, sugár, diffúz szín), a kamerát és a fényforrásokat

2. Kilövünk egy sugarat (egyenest) a kamerából a nézőablakon definiált következő rácsponton (a következő pixelen) keresztül. Ezen az egyenesen 2 pontunk van: a rácspont és a kamerapozíció Ezt a sugarat elsődleges sugárnak (***primary ray***) vagy szemsugárnak nevezzük (***eye ray***), mivel a kamerából lőjük ki, és (mint azt a rekurzív sugárkövetés ismertetésénél látni fogjuk) a fényút végigkövetésében ez az első kilőtt sugár is egyben.

3. A fenti egyenletek segítségével könnyedén meghatározható, hogy ez az egyenes metszi-e a térben elhelyezett gömbjeink valamelyikét. Fontos megemlíteni, hogy a negatív t értékeket figyelmen kívül kell hagynunk, hiszen ezek kamera mögötti metszéspontokat takarnak.

4. Amennyiben igen, kiválasztjuk a legközelebbi metszésponthoz tartozó felületet (hiszen ez takarja a mögötte levőket), s megnézzük, a fény milyen szögben érkezik a metszéspontban állított normálishoz képest.

5. A normális és a metszéspontból a fényforrásba mutató vektor (a fényvektor, azaz L = I – LS) által bezárt szög koszinusza kiszámítható a fent említett skaláris szorzat segítségével. A diffúz formula pedig cos(alfa) és a fényerősség alapján megadja, hogy mennyi fény érkezik a kamera síkjának adott pontjára. Itt figyeljünk arra, hogy egy fényforrás a gömbnek csak az egyik félgömbjét világítja be, tehát ha a skaláris szorzat negatív (a szög nagyobb, mint 90 fok), akkor a pont árnyékban van!

6. A 2-6 lépéseket ismételgetjük egészen addig, amíg a nézőablak minden rácspontját nem érintettük, azaz a kimeneti képünkön minden pixelre ki nem számítottuk az RGB hármast.

Ezeket a lépéseket a következő pszeudokód jellegű C függvény is jól szemlélteti, ennek segítségével az alkotó kedvű olvasók a következő számig már meg is írhatják az első ***Hello World*** sugárkövető programocskát.

Ebben a nagyon leegyszerűsített algoritmusban nem ismertettük a *scene* inicializációt (az olvasóra bízzuk, hisz implementációfüggő), a kamera pozicionálása és irányítása pedig nincs megoldva. Kameránk az origóban helyezkedik el és előrefelé (a Z tengely pozitív irányába) néz (azaz a nézővektora 0,0,1).
A kamera tetszőleges pozicionálásához és irányításához már fel kell használunk a sugártranszformáció módszerét, amelyet a következő számban fogok ismertetni. A számolandó kép méreteit `iWidth` és `iHeight`, az ***FOV*** méreteket `iFOVWidth`, `iFOVHeight` jelöli. Az algoritmus nem számol tükröződéssel, fénytöréssel és árnyékokkal sem.
A fény útját csak az első becsapódásig követjük, a felületek közti fénytani kölcsönhatásokat elhanyagoljuk. Ezt az algoritmust nevezzük ***ray castingnak***, ***Arthur Appel*** mutatta be még 1968-ban. A rekurzív sugárkövetést, már fénytörések, árnyékok, tükröződések modellezésére is alkalmas. A rekurzív sugárkövetés ***Appel*** úr úttörő munkáján alapszik, de ezt már ***Turner Whitted***-nek köszönhetjük, aki 1979-ben publikálta munkájának gyümölcsét. Amint azt a neve is mutatja, a rekurzív sugárkövetés tulajdonképpen rekurzív ***ray casting***, de erről bővebben majd a következő számban olvashattok.
A következő számban tehát ***Whitted úr*** művével fogunk kezdeni, ezután rátérünk a sugártranszformációkra, a textúrázásra, Phong árnyalási modelljére, majd további felületekkel ismerkedünk meg. Egyesek talán kicsit „szájbarágósnak” érezhetik az olvasmányt – nem minden alap nélkül. Az ő megnyugtatásukra azért leírom, hogy ez szándékos, de csak az első számot terveztem ilyenre, a továbbiakban nagyobb léptékkel haladunk majd és igyekszem majd tömörebben fogalmazni. Ettől függetlenül remélem sokak érdeklődését sikerült felkeltenem a téma iránt. Ha valakinek kérdései merültek fel a sugárkövetéssel kapcsolatban, vagy véleményét szeretné kifejezni, az ne hezitáljon, dobjon egy e-mailt a lenti címre, amint időm és energiám engedi, válaszolni fogok.

1. Lista A ray casting implementáció diffúz árnyalással

```
void RenderScene() {
  int iL = 300;  // Ez a fókusztávolság, azaz a nézősík távol
                 // sága a kamerapozíciótól
  Vector  vectNormal, vectEye, vectIntersection;
  RGB rgbSphereDiffuseColor = {1.0, 1.0, 1.0}; // A gömbök
                        //diffúz színe (ez esetben fehér)
  RGB rgbPixelColor;
  for (int iX = 0; iX < iWidth; iX++) {
    for (int iY = 0; iY < iWidth; iY++) {
      vectEye = {((iX - (iWidth/2))*iFOVWidth)/iWidth,
      ➥((iY - (iHeight/2))*iFOVHeight)/iHeight,  iL};
      rgbPixelColor = {0.0, 0.0, 0.0};

      // Minden gömböt megvizsgálunk, van-e metszéspontja
      // a szemsugárral (aSpheres egy
      // globális tömb, amelyben a gömbdefiníciókat tároljuk)
      for (int iSphereIdx = 0; iSphereIdx < giNumOfSpheres;
      ➥iSphereIdx++) {
        // Van metszéspontunk?  Amennyiben van,
        // IntersectSphere() true-val tér vissza, valamint
        // a metszéspontot és az abban adott felületi normá
        // list a mutatóikon keresztül átadott
        // Vektor  struktúrákba írja.
        if (IntersectSphere(vectEye, aSpheres[iSphereIdx],
        ➥&vectNormal, &vectIntersection))  {
          // Minden fényforrás hozzájárulását hozzáadjuk
          // a metszéspontból a kamerába érkező
          // fény színéhez
          for (int iLigtSrcIdx = 0; iLigtSrcIdx <
          ➥iNumOfLightSources; iLigtSrcIdx++) {
            // CalcDiffuse kiszámítja a metszéspontba érkező
            // fényt az  iLigtSrcIdx által
            // indexelt fényforrásból. aLightSrcs egy globális
            // tömb, amelyben a
            // fényforrásainkat tároljuk)
            rgbPixelColor += CalcDiffuse(vectNormal,
            ➥vectIntersection, aLightSrcs[iLigtSrcIdx],
            ➥agbSphereDiffuseColor);
          }
        }
      }
      // PutPixel  a megadott koordinátákon található pixelt
      // a megadott szinűre
      // színezi, azaz kirajzolja a pixelt a kiszámított
      // színnel.
      PutPixel(iX, iY, rgbPixelColor);
    }
  }
}
```

Szendi Ákos
(akos.szendi@gomortel.hu)

27 éves, szabadúszó programozóként tevékenykedik. A Miskolci Egyetem villamosmérnök szakos hallgatója. Kevéske szabadidejében gitározni tanul vagy épp egy jó könyvet tart a kezében.

PHP nyomkövetők (3. rész)

Sorozatunk előző részeiben áttekintést adtunk a Linux alatt elérhető PHP nyomkövetőkről. Szó esett a PHP nyomkövetés általános meggondolásairól, valamint beszéltünk a Gubed/Quanta nyílt forrású programok működéséről, beállításáról. Most a Nusphere::PhpED fejlesztőkörnyezetről és nyomkövetőről lesz szó.

Miért éppen a PhpED?

Ebben a részben kivételt tennék annyiban, hogy a *PhpED* program több platformra is létező implementációi közül nem a Linuxos, hanem a *Microsoft Windows*-ra írt változatot venném górcső alá. Ennek verziószáma jóval előrébb tart Linuxos társánál. Nehezen érthető számomra, hogy a *DBG* szerzője és a *NuSphere* vezetője, *Dmitri Dmitrienko* miért intézi ezt így. Azonban népszerűsége és használhatósága miatt nem hagyhatjuk a *PhpED*-et figyelmen kívül. E sorok nyomdába kerülésekor talán már a linuxos változat is eléri azt a 4.5-ös verziószámot, ami alapján e sorok születtek.

Ha az ember csak úgy vaktában elkezd *PHP* nyomkövetőt vagy profilkészítő alkalmazást keresni, mert eddig még nem használt semmi ilyesmit, de a baj most erre kényszeríti, akkor nagy valószínűséggel ez az eszköz akad a kezébe. Közkedveltsége abból is fakad, hogy van benne *FTPS (TLS/SSL)*, *WebDAV/HTTPS (SSL)* és *SOAP* kliens támogatás, tud kapcsolódni *SQL* szerverekhez, igen jó a dokumentációja, s hogy a támogatói levelezőlistán maga a szerző válaszol meg szinte minden levelet (ami őt érinti). Olvasható is vele egy interjú a ➲ *www.nusphere.com* oldalon. Innen tölthető le maga a program is.

PhpED fejlesztőkörnyezetet IDE, de gyorsan!

Tegyük fel, hogy úgy áll a helyzet, hogy van egy (távoli) webszerver, amin ott vannak a *PHP* anyagaink, fájljaink. Semmi előzményünk nincs semmilyen fejlesztőkörnyezetben, csak nyomkövetni akarunk, de még tegnap, ha lehet. Ekkor hogyan érdemes elkezdeni a munkát a *PhpED*-del?

A webszerver okosítása

Először is a webszerverben futó PHP-értelmező verziószámával egyező számú *dbg__.so*-fájl másolandó fel a *PhpED* kliens oldali könyvtárából a szerver *extension_dir* könyvtárába (aminek tényleges mivoltát pl. egy *phpinfo()*-ból, vagy magából a *php.ini*-ből tudhatunk meg). A *PHP*-értelmező verziószámát a `php -v` parancs is megmutatja.

A *php.ini*-be be kell írnunk néhány paramétert. Melegen ajánlott, hogy csak a *localhost*ot engedjük:

```
extension=dbg.so
[debugger]
debugger.enabled=on
debugger.profiler_enabled=on
debugger.hosts_allow=localhost
debugger.hosts_deny=ALL
debugger.ports=7869, 10000/16
```

Amiatt lehetséges a szerveren a *localhost* (helyi gép) használata, mert majd a kliensünk operációs rendszeréből indítunk egy ssh-alagutat a webszerver felé, és onnantól kezdve az „helyinek” fog látszani:

```
"c:\program files\putty\putty"
➥-ssh -R 7869:localhost:7869
➥login@szerverneve
```

A webszerver újraindítása után (sajnos rendszergazdaként lehet csak ilyet művelni, vagy – egy jól beállított rendszeren – az erre kiválasztottak ezt `sudo`-val is megtehetik) érdemes kiadni egy `php -m` parancsot, ami a szervermodulokat mutatja. A lista végén látszania kell a nyomkövető protokollért felelős modulnak:

```
[Zend Modules]
DBG
```

A PhpED beállítása

Abban az esetben, ha tényleg szinte semmit nem akarunk, csak hibát keresni, akkor is meg kell néhány információt adnunk a fejlesztőkörnyezet számára:

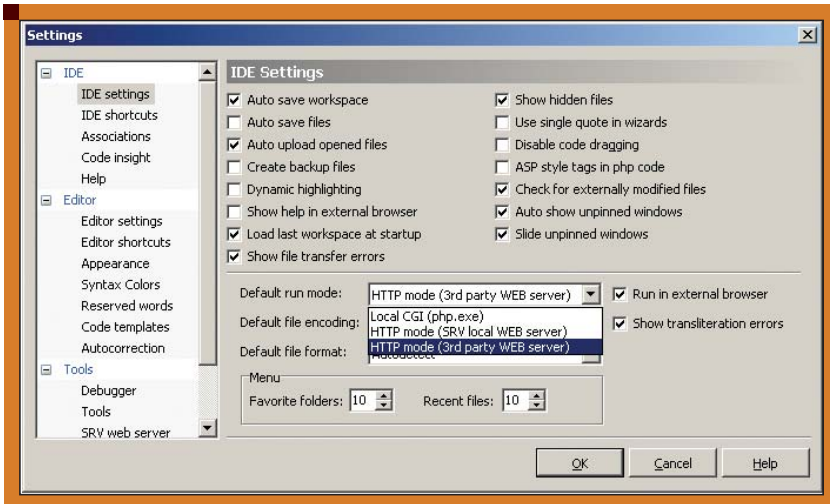
- Új munkaterületet (workspace) nevezünk ki, mondjuk „szimpla” néven
- Új projektet is gyártunk, melynek a gyökérkönyvtárát például „minimalis”-ra nevezzük. Ilyenkor még meg kell adni legalább a „Mapping root URL”-t és a webszerver document gyökérkönyvtárát, például *http://szerverneve/egyeb* és */var/www/html/ez_az_a_resz/*
- Ekkor feljön egy dialógusablak, amiben végre elmenthetjük a projektünket például *minimalis.ppj* néven. Érdemes ügyelni arra, hogy a projekt gyökérkönyvtárának a neve is emlékeztessen erre a névre.

A nyomkövetés indítása

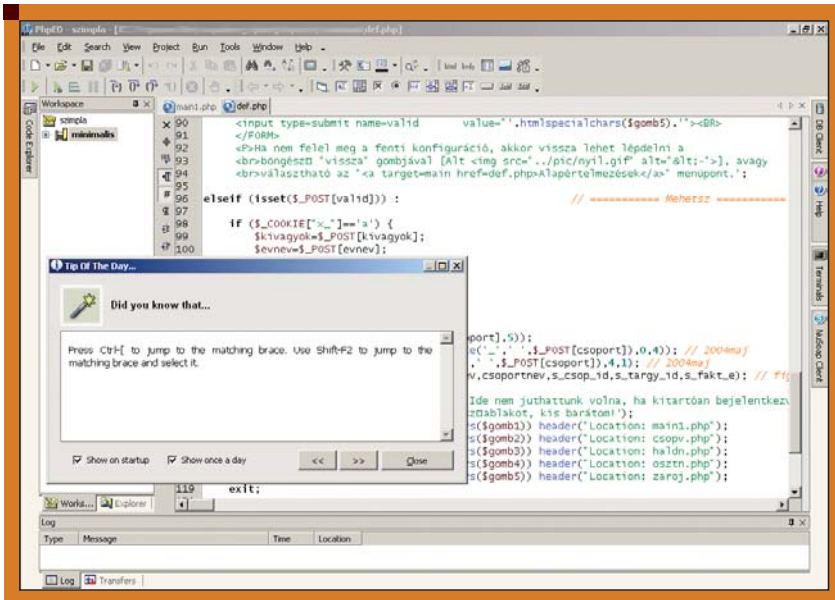
Ezek után már nyithatjuk is a vizsgálni kívánt URL-t: A *Tools* menüből az *"Open URL"*-be írva:

http://szerverneve/admin

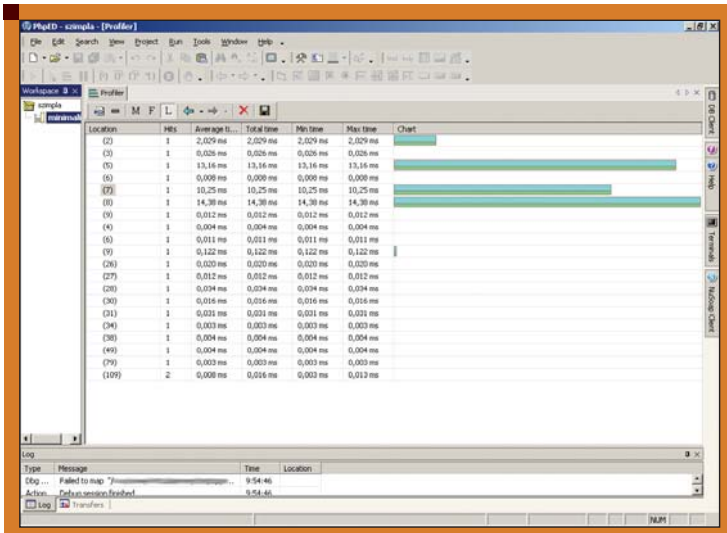
Itt van lehetőség eldönteni, mely jelölőnégyzetbe kattintunk pipát: nyomkövetni akarunk-e (***Run debug session***), teljesítményt szeretnénk elemezni (***Run profiler session***), esetleg külső böngészőt szeretnénk használni (***Run in external browser***).
És már megy is a nyomkövetés. Lehet lépegetni sorról sorra. Az első soron automatikusan megáll a nyomkövetés – ennek megváltoztatásához van is egy jelölőnégyzet a beállításoknál. Állíthatunk töréspontot, feltételhez kötve is. Lehet nézni a változók értékét (egérrel rámutatva), vagy akár változtatni rajtuk.
A beépített profilkészítő (*3. ábra*) bosszantó módon csak 20 sort mutat az ingyenesen kipróbálható változatban, úgyhogy lényegében felesleges is sokat beszélni róla, annyi más szabad eszköz van e célra. Ha valakinek mégis kell a ***PhpED*** teljesítményelemzésének eredménye, el lehet menteni egy ***.xml*** fájlba (floppi ikon). Ha a teljesítményelemzés közben szeretnénk megnézni azt a kódrészletet, aminek a mérési adatait láthatjuk, akkor szükség van még valamire, amit könytárleképezésnek (***mapping***) hívnak, és ami a forráskód beazonosítására szolgál. Ez persze nem árt a normál nyomkövetéshez sem; akkor ugyanis nem ad figyelmeztetéseket az *„**unmapped files**"*, azonosítatlan fájlok miatt.



1. ábra Beállítások



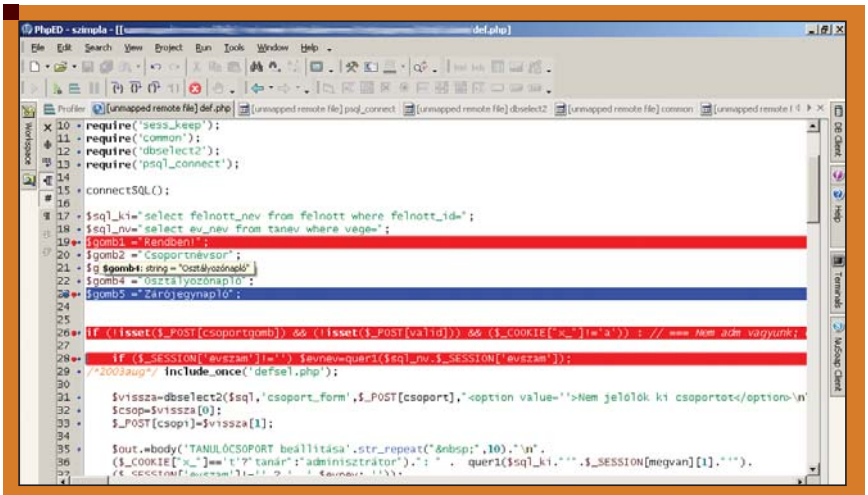
2. ábra Tipp mutatásával indul a PhpED kliensprogram



3. ábra A profilkészítő

Könytárleképezés után szebb az élet

Ez nem egy triviális feladat. Nem véletlen, hogy a ***PhpED*** levelezőlistáján minden második kérdés erre vonatkozik. Eleinte nem volt világos, hogy mit mivel akar összepárosíttatni a program. Eleinte azt a két információt adtam meg csak, hogy mi a webes URL és a webszerver fájlrendszerében való elérési út. Kiderült azonban, hogy csak akkor működik megfelelően



4. ábra A PhpED teljes díszben

a program, ha a helyi gépen is megtalálhatóak az érintett fájlok. (Vagy én nem találtam meg a módját a másképp szervezésnek; mindenesetre a nagy vetélytárs, a *Zend Studio* jobban viselkedett kezeim között e tekintetben.) Magyarán a projekt tulajdonságait beállító fő ablaknak három (nem egymás közelében levő) sora tartozik össze: A fent említett webes URL és a webszerver fájlrendszerében való elérési út mellett a *„Projekt”* rész alatti *Root directory*-t (gyökérkönyvtárat) is be kell állítani úgy, hogy onnan kiindulva azonos elérési úttal lehessen itt megtalálni a *.php* fájlokat, mint a webszerveren. S ekkor már a profiler is meg tudja nevezni a forrásokat, és rá tud állni az általunk keresett részre.

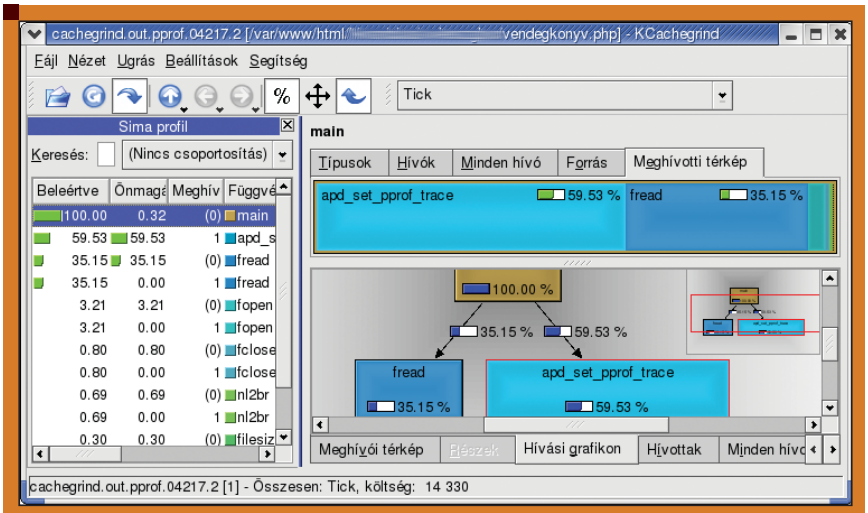
Ha 20 sornál hosszabb a kód: ne APD fel!

A *PhpED* teljesítményelemzése csak 20 sort mutat. Azonban aggodalomra nincsen ok: rendelkezésünkre áll az *Advanced PHP Debugger* (*APD*) ➲ *pecl.php.net/package/apd*. Neve megtévesztő. Szerzőjével, *George Schlossnagle*-vel váltottam egy e-mailt, amiben megkérdeztem, milyen klienst tud ajánlani programjához. Azt írta, hogy programja igazából teljesítményelemzésre való – nyomkövetéshez használjam az *Xdebug*-ot. Próbáltam őket (az *APD*-t és az *Xdebug*-ot, amiről a következő részben lesz szó) egyszerre beintegrálni a webszerverbe, de ilyenkor sajnos elszállt a webszerver *segfault* hibával, ami nem is csoda, hiszen ez a modul is módosítja a *PHP* belső függvényhívásait, így más kiterjesztéseket zavarhat. Az *Xdebug* behúzását tehát „kikommenteztem” (pontosvesszővel) a *php.ini*-ben. *„PHP Performance Profiling”* (*PHP* teljesítményelemzés) címmel olvasható angol nyelven egy részletes írás az *APD*-ről: ➲ *www.linuxjournal.com/article/7213*, valamint a ➲ *www.php-editors.com/php_manual/ref.apd.html* is jó eligazítást ad. Az *APD* inkább azt tűzte ki céljául, hogy olyan profilkészítő lesz a PHP számára, mint amilyen a *gprof* a *C* nyelv számára, vagy a *Perl Devel::DProf*-ja.

Az APD forrásból fordítása a `phpize; ./configure; make` parancssorral történhet. A fordításhoz kell néhány csomag a webszerverre, ha még nincs ott: d*pkg* alapú rendszereknél az `apt-get install php-dev php-devel automake gcc cpp` parancssor segít sokat. Ezeket a fordítás után `apt-get remove`-val érdemes eltávolítani, ne éktelenkedjenek ott feleslegesen (és a biztonságot aláásva). A keletkezett *modules/apd.so* fájl bemásolandó a fent tárgyalt *„extension_dir”* könyvtárba (így tud majd betöltődni, mint *„zend_extension”*). A *php.ini* fájlba még be kell írnunk e sorokat, majd újraindíthatjuk a webszervert:

```
zend_extension = /az/apd.so/
➥elérési/útja
apd.dumpdir = /a/kimeneti/
➥fájlok/könyvtárának/neve
apd.statement_trace = 0
```

Ez utóbbi a soronkénti nyomkövetést állítja. 1-esre állítva igencsak lelassul az alkalmazás. A középső sorból az látszik, hogy meg kell adni egy kimeneti könyvtárat (amit aztán célszerű pl. egy `chmod 777` paranccsal írhatóvá tenni) – ide fognak érkezni az *APD* által gyártott kimeneti fájlok, például a *pprof.04214.0.* (A fájlnév végén levő számok processzazonosítóra utalnak.) Az *APD* lefordítása után találunk a könyvtárban egy *pprofp (PHP profiler parser)* nevű, parancssorból futtatható szkriptet, sok kapcsolóval (mi szerint rendezze az adatokat,



5. ábra APD-kimenet kcachegrind ruhában

mit mutasson és mit ne stb.). Az APD kimenetéből tehát messze nem pusztán az egyes lépések futási ideje derül ki, hanem lehet pl. függvényhívási fát is kérni.
Dpkg alapú rendszerekben a ***kdesdk*** csomag része a ***pprof2calltree***, amivel a a ***kcachegrind*** számára feldolgozhatóvá tehetjük az APD kimenetéből kapott fájlokat.
A ***kcachegrind*** (és tartozéka, a ***callgrind***) letölthető innen:
➲ ***kcachegrind.sourceforge.net***.
A teljesítményelemzésnek alávetendő PHP szkriptünk elejére illesztendő az `apd_set_pprof_trace();` parancs. Ha megkapja a PHP-értelmező az adott szkriptet, elkészül a ***pprof-kimeneti-fájl*** az ***apd.dumpdir***-be. Ezek után egy jól célzott

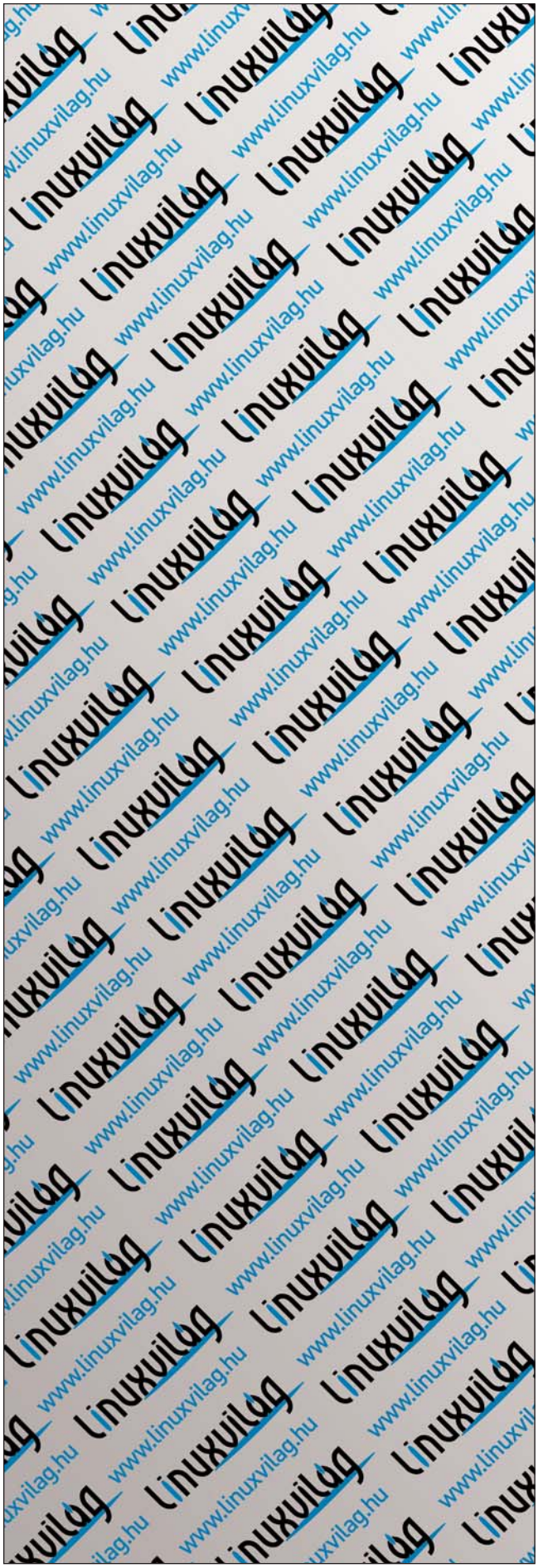
```
pprof2calltree -f  pprof- kimeneti-fájl
```

paranccsal előállíthatjuk a ***cachegrind.out.pprof-kimeneti-fájl***-t, amit már átadhatunk a ***kcachegrind***-nek. Az eredményül kapott szép (interaktív) felületen vizuálisan láthatjuk, mire mennyi erőforrás kell (s azt is, hogy a profilkészítés bizony erősen használja a gépünket).
Jó tudni, hogy általában a ***.kde/share/config/kcachegrindrc***-ben vannak eltárolva a legutóbbi nézethez tartozó beállítások.
A grafikonban alul (kissé elrejtve) található a függvényhívási grafikon megjelenítőgombja. A forrásoknak a helyi gépen is megtalálhatóaknak kell lenniük a profil grafikonjának elkészítésekor, ha a kódba is bele akarunk látni.
Ha valódi, éles weboldalakat szeretnénk mérni, akkor minél többféle módon érdemes azt megtenni, különböző terheltségek mellett, és több adatsort átlagolni a hiteles válaszhoz. Amit az APD (és más profilkészítők) tudnak mérni, az a valós idő, nem pedig a processzoridő (hiszen többfeladatos rendszerekben más programok is ténykednek a processzoron), és attól is eltér, amit a felhasználó átél, amíg megkapja a kért weboldalt. Ha csak úgy odaírjuk az `apd_set_pprof_trace();` parancsot egy PHP fájl elejére, gondot okozhat, hogy sok fájl keletkezik az éles környezet számítógépén. Ettől le is lassul a rendszer. Emiatt érdemes egy feltételhez kötni a profilkészítési kérést a mérendő PHP fájlunkban:

```
$DEBUGIPS = array('84.2.93.34',
➥'192.168.0.13');
if(array_search($_SERVER [REMOTE_IP],
➥$DEBUGIPS)) {apd_set_pprof_trace();}
```

Azaz csak a kiválasztott IP-címekről érkező kérések esetén indul el a mérés.
Következő részünkben két ***Xdebug***-alapú PHP-nyomkövetőt tekintünk meg, az ***ActiveState::Komodo*** és a ***Xored::TruStudio*** fejlesztőkörnyezeteket.

Szabó Zoltán
(szz@freemail.hu)
Négy gyermekével és feleségével Pannonhalmán él. Tíz éve kísérletezik a Linuxszal. Matematikát és informatikát tanít, diákotthonban keseríti a rábízottak életét. Szívügye a PHP, a PostgreSQL és a Moodle.

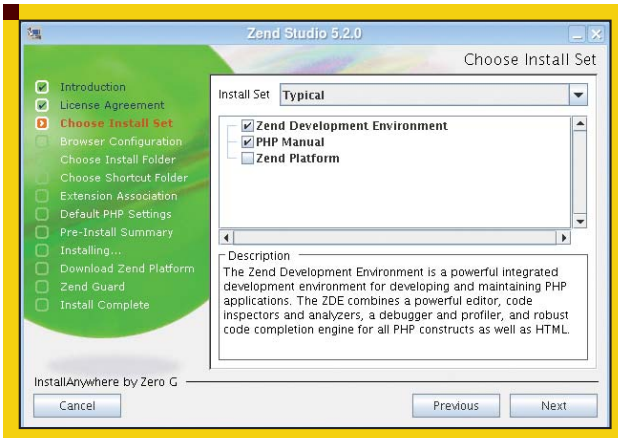


Zend Studio – PHP fejlesztés egységbe zárva

A PHP alkalmazások fejlesztése során számtalan eszközzel találkozhatunk: különféle szövegszerkesztőkkel, verziókövető rendszerekkel, dokumentációkészítővel és még valamilyen hibakereső eszközzel is. Még ennél is több szolgáltatást kapunk egybecsomagolva a Zend Studio Java alapú integrált fejlesztőkörnyezettől.

Az első lépések

A szoftvert a *Zend Technologies* készíti, a kipróbálásra szánt időkorlátos verziója letölthető a ➲ *www.zend.com* weboldalról, ugyanitt megvásárolható a *Professional* változat licence is 299$-ért. A program *Javaban* íródott, így a *Studio Windowson*, *Linuxon* és *Mac OS X*-en képes futni. A linuxos változat egy *tar.gz* archívum formájában érkezett hozzám, melyben egy végrehajtható fájl lapult. A telepítő eltér az átlagos linuxos telepítésektől, egy varázsló vezet végig a telepítés lépésein. Fontos, hogy a *Zend Platformot* is kiválasszuk a telepítendő összetevők közül, mert különben számtalan kiegészítő szolgáltatás nem fog működni. Ez a legnagyobb buktatója a telepítésnek, különben csak végig kell menni a lépéseken. Ahhoz, hogy a *Platformot* is használni tudjuk, kell legyen webszerver a rendszeren. Ha véletlenül hiányozna, ezt pótoljuk az *Apache* feltelepítésével! Miután követve a telepítő utasításait, életre keltettük a rendszert, nézzünk körbe, hogy mit is kaptunk!



1. ábra A telepítés menete

Szövegszerkesztés

A fejlesztési munka egy részében nem történik más, mint a programozó egy szövegszerkesztőbe beírja a program kódját. Egy jól megírt szerkesztő nagyban javítja a hatékonyságot. Vajon a *Zend Studioba* épített szerkesztővel mi a helyzet? Aki az *emacs* vagy a *vi* tradicionális, nagy tudású unixos szövegszerkesztőket mesteri szinten tudja kezelni, annak ez a szerkesztő nem lesz egyáltalán hatékony. Ellenben a kevésbé profikat az alábbiakkal kényezteti el. Kezeli a *PHP* és a *HTML* szintaxist. Ez alapján színezi a forráskódot, amivel sok elírás azonnal felfedezhető. Képes a kód kiegészítésére, azaz a már meglévő függvény és objektumneveket, tulajdonságokat automatikusan felajánl, a meglévő deklarációkat megmutatja, hozza a hozzá írt *DocBlock* dokumentációt is. Ez igaz a beépített és az általunk írt objektumokra és függvényekre is. Gépelés közben szintaktikai ellenőrzést végez, így azonnal jelzi a hibákat. Képes automatikusan tabulálni a beállított kódolási szabály szerint a kódot. Létre tud hozni *DocBlockokat* már meglévő függvény vagy objektum alapján, amit nekünk már csak ki kell egészíteni. Képes a megjegyzésbeli illetve a kódbeli egységeket (kapcsos zárójelekkel elhatárolt részek) kattintásra összecsukni illetve kinyitni utána, így mindig csak az a rész van a szemünk előtt, amivel épp ténylegesen dolgozunk. Képes a forráskódban navigálni, megkeresni objektumok vagy függvények deklarációját. Ezeket egyébként kigyűjti egy listába is. Mi az, amit ezért a tudásért fel kell áldoznunk? Egy már megszokott szerkesztő működését. Talán segítség azért, hogy a gyorsbillentyűk teljesen testre szabhatóak.

Hibakeresés

Nem az a kérdés, hogy fogunk-e hibákat véteni a kódolás során, hanem az, hogy milyen gyorsan fogjuk azokat kijavítani! A program teljes körű és kényelmesen használható hibakeresőt (*debugger*) tartalmaz.
A hibakeresőben pont azok a szolgáltatások érhetőek el, amit bárhol máshol kapnánk. Be lehet állítani töréspontokat, változók figyelését. A futtatást lehet szabályozni: soronként léptethetjük, elindíthatjuk, leállíthatjuk és adott helyre ugorhatunk. A futtatás során megkapjuk a *PHP* által adott figyelmeztetéseket is egy keretben. Ezen túl van *Analyze* menüpont,

ami elég sok programozói tévedést képes kiszűrni. Cserében persze néha olyankor is hiányosságot talál, ahol nincs: nem képes megérteni azért a kód célját, ez a nehéz feladat még mindig a fejlesztő dolga.

Teljesítményhangolás

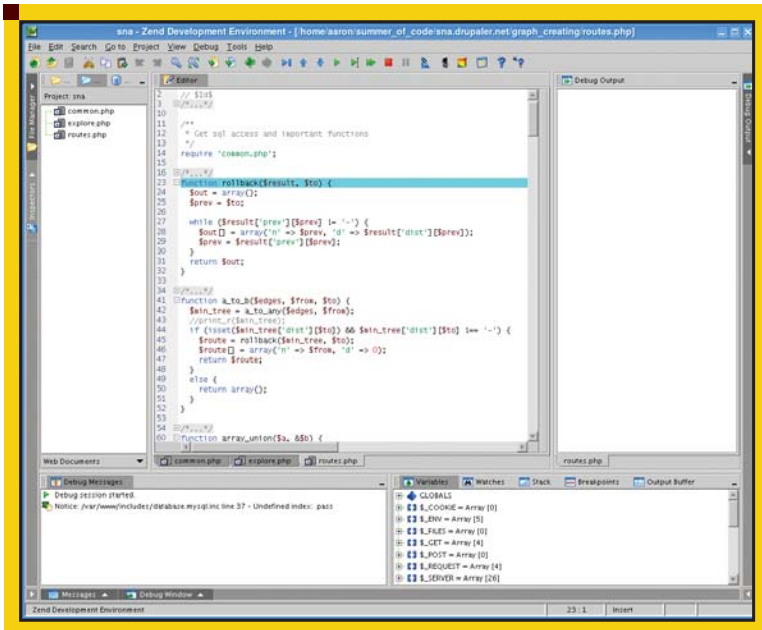
Nem kis feladat egy már meglévő alkalmazást nagyobb sebességre gyorsítani. Nincs rá titkos recept sajnos. Viszont a gyenge pontok megtalálása teljesen automatizálható, így a lényegre tudunk koncentrálni, ami pedig a rosszul teljesítő kódrészlet átírása, nem pedig a mérés maga. Ehhez a funkcióhoz telepíteni kell a *Platformot* is, mert ilyenkor a szkript a webszerveren kell fusson.

A *Tools/Profile URL* menüpontot választva csak egy teendőnk van: megadni, hogy milyen elérési úton található a kérdéses program. Ha csak a helyi gépen fejlesztünk, ehhez a lépéshez elérhetővé kell tenni a webszerveren keresztül is a kódot. Miután a szkript lefutott, kapunk egy új ablakot, amiben mindent megtudunk a program futtatásáról és könnyedén kiszűrhetjük a leglassabb részeket.

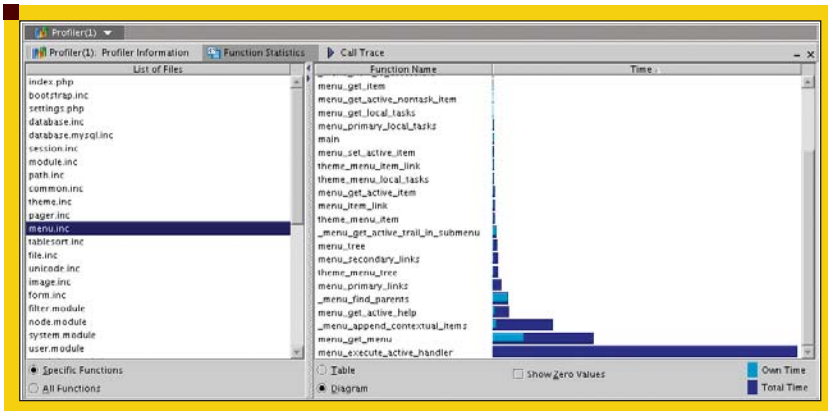
Amennyiben a programunk több fájlból áll, a futtatás után látni fogjuk egy csinos diagramon, hogy melyik fájlban mennyit időzik a végrehajtás. Ezen kívül minden fájlról részletes függvénylistát kapunk, hogy melyik függvény meddig futott és hányszor került meghívásra. Meg van különböztetve az az idő, amit ténylegesen a függvény fogyasztott el, meg az, amit csak a függvény által meghívott újabb függvények igényeltek. Kilistázhatjuk a függvényhívási hierarchiát is. Az oszlopok csökkenő illetve növekvő rendezésével gyorsan kibukik egy rosszul megírt függvény. Jól látszik az is, ha egy függvényt indokolatlanul sokszor hív meg egy hiba folytán a program.

Adatbázis-kezelés

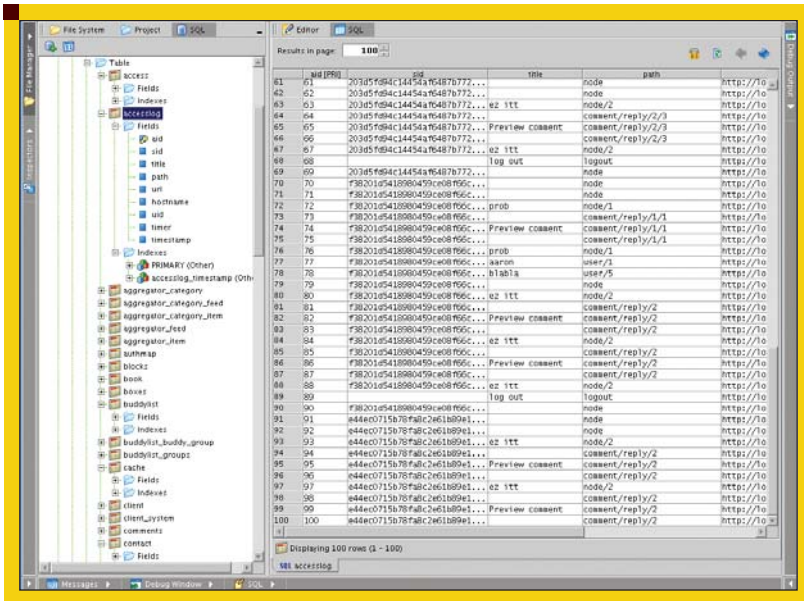
Kevés olyan *PHP* nyelven megírt alkalmazás van, mely ne használná valamelyik *SQL* adatbázismotort. Igaz ugyan, hogy általában ezekhez jár kezelőfelület, biztos ami biztos, a *Studioba* bele van építve egy *SQL* ügyfél, ami számtalan kiszolgálóval (a nevesebb példák: *MySQL*, *MSSQL*, *Oracle*, *PostgreSQL*) képes együttműködni. A kezelőfelületbe jól beépül



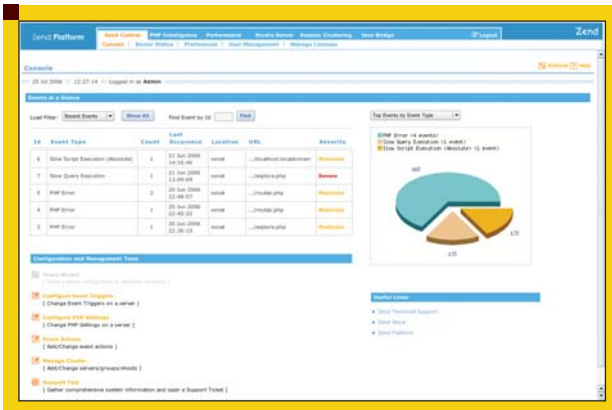
2. ábra Hibakeresés akcióban



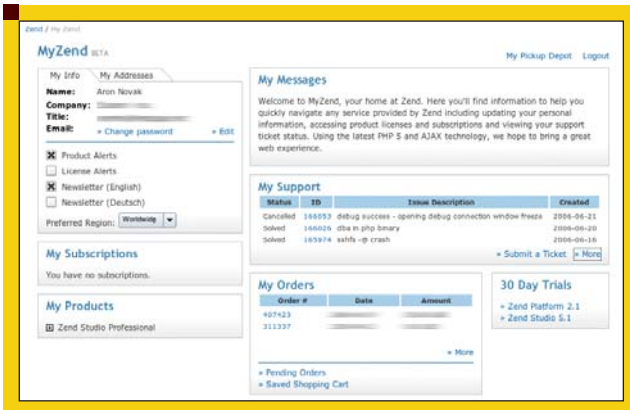
3. ábra Teljesítménymérés



4. ábra Adatbáziskezelés a Studion belül



5. ábra A Zend Platform webes kezelőfelülete



6. ábra A terméktámogatás központja

a kliens, egy mozdulattal válthatunk a programkód és az adatbázis között. A szokásos funkciókat kapjuk: a táblákat listázhatjuk, beszúrhatunk rekordokat, áttervezhetjük a struktúrát. Természetesen a saját kézzel írott *SQL* lekérdezéseinket is futtathatjuk, azok végeredményét éppúgy láthatjuk. Az sem hátrány, hogy ha több adatbázis-kiszolgálót használunk, akkor ezentúl egy egységes felületen keresztül érhetjük el azokat. Persze azt ne feledjük, hogy az egyes termékekre hangolt *SQL* ügyfelek többet nyújtanak általában, mint egy olyan, amelyik több fajtával is együttműködik.

Bűvészkedés a kiszolgálón

Ha a *Platform* telepítését is kértük, akkor a webkiszolgálónk otthont ad a *Zend Platform* kezelőfelületének is, ahol már a *Studio* fejlesztőkörnyezeten túlmutató dolgokat találunk. Magához a *Studiohoz* jár egy fejlesztői *Platform* licenc is, ami lehetővé teszi többek között, hogy a *PHP* programjaink szerveren történő hibakeresésé-hez és teljesítményhangolásához használjuk. De mint láthatjuk, a *Platform* ennél sokkal többre képes. Az *5. ábrán* a nyitóoldalát látjuk a webes kezelőfelületnek, ahol az eseménynapló összegzése található. A *Platformban* van *PHP* gyorsítótár is, aminek segítségével a dinamikus tartalmat az *Apache* gyorsabban képes kiszolgálni. Erről jelentést is kapunk a weboldalon: pontosan látható, hogy mekkora haszonra teszünk szert, hogy a gyorsítótár üzemel. Számszerűen látjuk, hogy mennyivel gyorsabban futnak le a szkriptek. Ha valódi oldalt üzemeltetünk, akkor hasznos, hogy különféle eseményekre figyelést tudunk beállítani. Az előre beállított eseményfigyelésekre példa: lassú szkriptvégrehajtás és futásidejű programhibák. A *Platform* a *Studio* felhasználóinak azért is fontos, mert ezen a felületen lehet beállítani, hogy működjön a távoli hibakeresés. A *Studio Server* menüpont alatt kezelhetjük azon számítógépek listáját, melyek ezt a szervert felhasználhatják hibakeresésre.

Terméktámogatás

Mivel egy kereskedelmi termékről van szó, mi sem természetesebb, hogy jár hozzá terméktámogatás. Ennek minősége épp olyan szempont kell legyen, mint a szoftver szolgáltatásai, hiszen melyik szoftvernél ne futnánk előbb-utóbb olyan szituációba, amit magunk nem tudunk megoldani? Egy zárt forráskódú terméknél még fontosabb a támogatás, hiszen itt a megfelelő hozzáértés birtokában sem tudjuk orvosolni a problémáinkat, ellentétben a nyílt forráskódú eszközökkel. Akadt hiba a program használata közben, nem is kicsi: az Apache webszerver folyamatosan *Segmentation Fault* hibákat dobott bizonyos *PHP* kódok végrehajtásakor. A hibajelentés menete ki van dolgozva: fel kell adni egy hibajegyet, aztán minden ezzel kapcsolatos ügymenet ahhoz tartozik. A *6. ábrán* lehet látni a *Zend* megoldását arra a problémára, amire a nyílt forrás a *BugZillát* készítette. A hibajegy feladása után egy munkanappal ténylegesen érkezik reakció a beadott problémára, kérdésre. Az előbb említett Apache hiba részleteit ugyan több héten keresztül kérdezgették a hibajegyen keresztül, de végül ezt a nagyobb hiányosságot is orvosolták egy új binárissal. A reakcióidő az ígért értéknek megfelelt, hiszen minden új problémára egy napon belül volt válasz. A program árában egy év támogatás van. Egyébként gesztusértékű, hogy az időkorlátos próbaverzióval is adhatunk fel hibajegyet.

Hátulütők

Ezeknek a megoldásoknak a jó része elérhető nyílt forráskódú verzióban is: *PHP* gyorsítótár, hibakereső, teljesítménytesztelő mind létezik szabad licenc alatt is. Amiről nincs tudomásom: hasonló tudással felvértezett grafikus integrált fejlesztőkörnyezet, ahol a kódkiegészítéstől elkezdve egészen a távoli hibakeresésig minden egy fedél alatt megtalálható. Ez a megoldás pontosan ellentétes a unixos filozófiával, miszerint minden program lásson el egy részfeladatot, de azt nagyon jól és a sok kis eszköz együttes használatával bármi megoldható. Kérdés, hogy megéri-e az árát a *Studio*? Próbáljuk ki! Ha kevésbé hatékonyan tudunk benne fejleszteni, mint az eddigi szabad szoftveres megoldásainkkal, akkor csak mentünk egy kört a próbaverzióval.

Novák Áron
(aaron@szentimre.hu)

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

Minőségi dokumentáció egyszerűen phpDocumentor

Az interneten kering egy leírás az „igazi programozóról”, amely ugyan az informatikus humor becses darabja, de mint minden viccben, ebben is van némi komoly tartalom. Mint a cikkben idézet részletből is látható, a dokumentációírás sem az igazi, sem a botcsinálta programozóknak nem kedvenc elfoglaltsága, akármilyen más híresztelést halljunk is.

Az igazi programozó...

A tipikus „igazi programozó” élettere terminálja közvetlen közelére terjed ki. Ezen terminál körül a következők találhatók:

1. Az összes program listája, amelyen az igazi programozó valaha is dolgozott, nagyjából időrendi sorrendben, elfoglalva a szoba összes sík felületét.
2. Vagy fél tucat félig üres csésze hideg kávé. Esetleg csikkek is úszhatnak a kávéban, ha nem lehetett megtalálni a listák között a hamutartót. Néha a csészékben narancslé van.
3. Hacsak nem különlegesen jó, *OS JCL* kézikönyvek és a *Principles of Operation*, néhány érdekesebb részletnél kinyitva.
4. A falon egy sornyomtatóval készült naptár (az 1969-es évre).
5. A padlón szétszórva csokipapírok.
6. A fiók alján egy folyamatábra-rajzoló sablon, amit az előző lakó hagyott itt. (Az igazi programozó programokat ír, nem pedig dokumentációt.)

Forrás: ➲ *www.cab.u-szeged.hu/local/doc/UNIX/orlando/igazi.html*

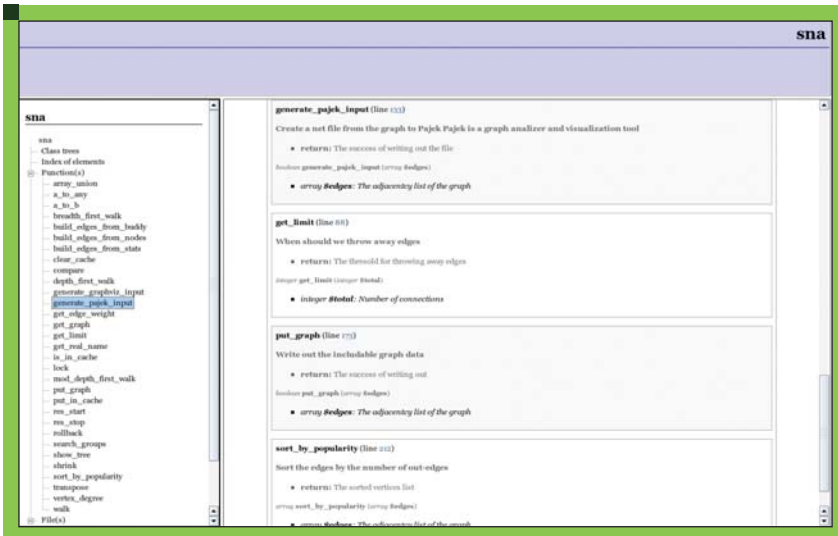
Miért készítsünk dokumentációt?

A keretes írásunk zárójeles megjegyzése adja fonák módon cikkünk mottóját. Ugyan manapság talán a hátulgombolós programozók fiókja alján sem találunk folyamatábra-rajzoló sablont, a jó minőségű és részletes dokumentáció elkészítése semmivel sem vált egyszerűbbé. Talán a dokumentációkészítő programok korunk sablonjai. Ha most arra gondolunk, hogy akkor hátulgombolós programozókká degradáljuk magunkat efféle eszközökkel, akkor fontoljuk meg a következőket: *„Az igazán dögös programozó hexa számológép nélkül találja meg a hat megabájtos dumpban a hibát.”* . Ha még nem értük el a programozói törzsfejlődés ezen evolúciós lépcsőfokát, akkor inkább legyünk korrekt hátulgombolósok, mintsem csapnivalóak.

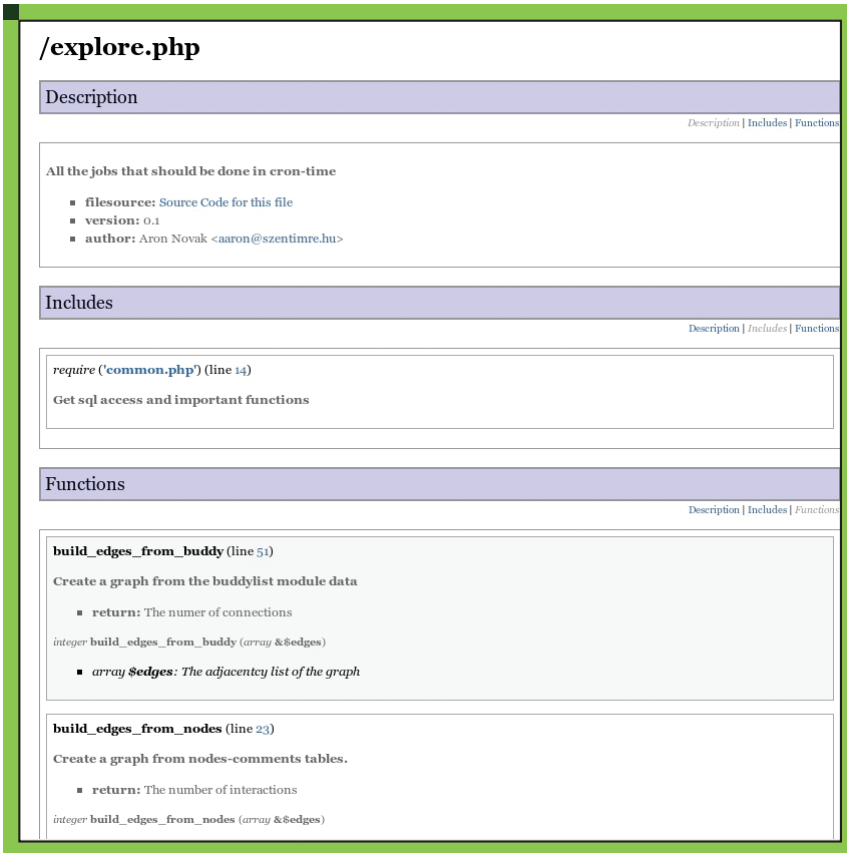
A *PHP* nyelv nagy népszerűségre tett szert a *LAMP – Linux/Apache/Mysql/PHP* összeállításnak köszönhetően, hisz *LAMP* felállásban működik számtalan webszerver a világhálón. A *PHP* alkalmas arra is, hogy minimális tudással egyszer használatos kódot írjunk szükségmegoldás gyanánt, de olyan népszerű és nagy projektek is ezt használják, mint a *Drupal*, a *PhpMyAdmin* vagy a *Gallery*. Egy nyílt forráskódú programnál ahhoz, hogy létrejöjjön egy fejlesztőgárda, a dokumentáció egy elengedhetetlen kellék. Hiszen kinek van kedve mindent az alapoktól kezdve a forráskódból kibogarászni? Egy jól megírt dokumentáció nagyban felgyorsítja a beletanulási időszakot. Így már az újonc is képes használható kóddal hozzájárulni a kódbázishoz. Először is kinek szól a dokumentáció? A fejlesztőknek és a felhasználóknak egyaránt, bár a két célcsoportnak különböző mélységű és stílusú leírásokra van szüksége. A fejlesztőknek szóló részét természetesen maguk a fejlesztők írják, a felhasználónak szólót már nem biztos, hogy célszerű a programozóknak írnia, hiszen ott nagyon fontos, hogy a felhasználó nyelvén szóljon a leírás.

Az eszköz

A *phpDocumentor* egy *PHP* nyelven írt alkalmazás, melyet *GPL* alatt terjesztenek. A projekt honlapjáról (➲ *www.phpdoc.org*) letölthető a legfrissebb változat, ez a cikk írásának pillanatában az *1.3.0RC6* verziót takarta. A *phpDocumentor* kétféle üzemben is képes működni. Egyfelől hagyományos parancssori alkalmazás, amelyet kapcsolókkal vezérelünk, másfelől egy webes alkalmazás, ahol pár kattintással összeállíthatjuk, mit szeretnék kérni. Itt az első változatot fogjuk áttekinteni, több okból is. Egyrészt látni fogjuk, hogy a munka kisebbik része magának



1. ábra Ez történik a függvényekkel feldolgozás után



2. ábra Minden forrásfájlról készül összefoglaló

a *phpDocumentornak* a használata, az előkészületek sokkal több energiát igényelnek, másrészt a webes felülettel semmivel nem járunk előrébb, viszont kell hozzá egy webszerver, ami *PHP* szkriptek írásához nem feltétel. Amennyiben a második eshetőség jobban érdekel minket, csupán annyi a teendőnk, hogy a letöltött *.tgz* fájlt nem akárhová, hanem a webszerver könyvtára alá bontjuk ki és máris használatba vehetjük az intuitív kezelőfelületet.

Előkészületek

A *phpDocumentor* ezen verziójába még nem építettek bele magas fokú mesterséges intelligenciát, mely tetszőleges forráskódot képes hasznos leírással ellátni. Mivel a közeljövőben erre nem is nagyon számíthatunk, érdemes elsajátítani egy speciális kommentezési technikát, melyet számtalan dokumentációkészítő eszköz képes értelmezni. Például a *Doxygen* vagy a *Javadoc* is, így *C++*-hoz és *Javahoz* is részben alkalmasak az itt leírtak. A *PHP* többféle megjegyzésformátumot támogat, *C*, *C++* és *Perl* stílusban is kommentezhetünk. A *phpDocumentor* azt várja, hogy a forráskódunkat úgynevezett *DocBlock* rendszerben lássuk el megjegyzésekkel, amiből már lehet automatikusan generálni a dokumentációt. Egy már meglévő programnál nagy munka ezekkel kiegészíteni a kódot, érdemes már a kezdetektől használni ezt a stílust. Így jelezzük, hogy *DocBlockot* írunk:

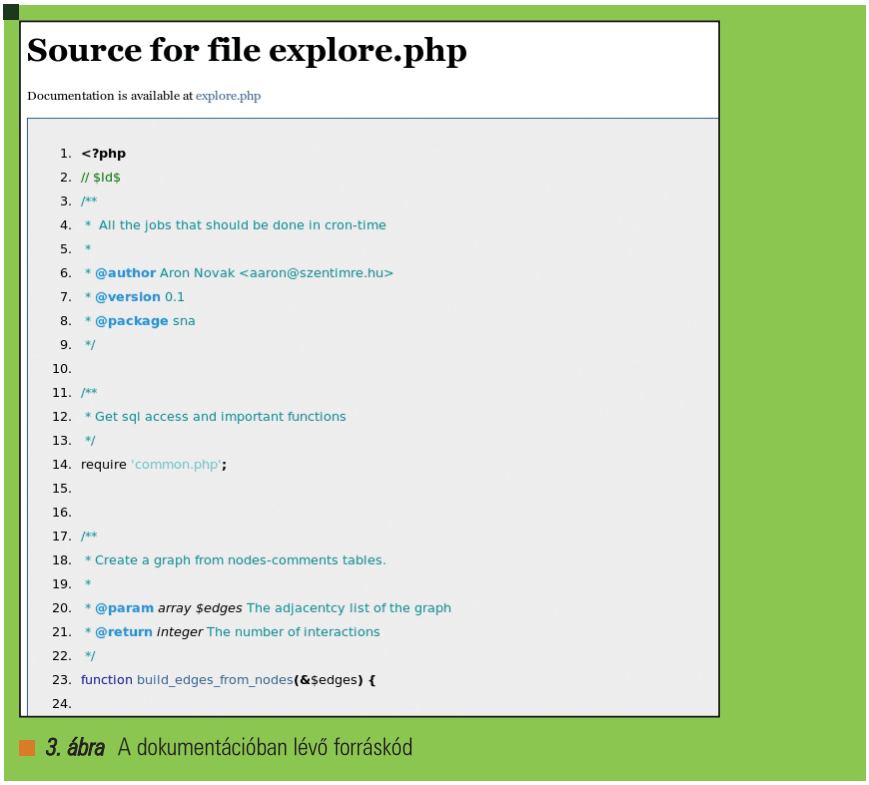
```
/**
 *
 */
```

Ez a kerete a blokknak, amit aztán meg kell tölteni értelmes tartalommal. Láthatjuk, hogy ez a forma a *C* stílusú megjegyzés kiegészítésének tekinthető. Ezeket a blokkokat bárhol elhelyezhetjük a kódban, célszerű konstansok, függvények és osztályok előtt írni egy blokkot, amiben részletezzük az adott elem szerepét. Minden ilyen *DocBlock* három részből tevődik össze ez alapján:

```
/**
 * Rövid leírás
 *
 * Hosszabb leírás
 *
 * Címkék
 */
```

A rövid és hosszabb leírást azt minden kötöttség nélkül leírhatjuk, például egy függvény előtt beleírhatjuk, hogy milyen algoritmust használ, vagy amit épp fontosnak érzünk. A címkéket már egy között formában kell megadni. Ha az aktuális blokk a fájlra magára vonatkozik, akkor e minta alapján készítsük el a megjegyzést a fájl elejére:

```
/**
 * Rövid leírásokra
 *
 * Hosszú leírás
```


Source for file explore.php

Documentation is available at explore.php

```
1.  <?php
2.  // $Id$
3.  /**
4.   * All the jobs that should be done in cron-time
5.   *
6.   * @author Aron Novak <aaron@szentimre.hu>
7.   * @version 0.1
8.   * @package sna
9.   */
10.
11. /**
12.  * Get sql access and important functions
13.  */
14. require 'common.php';
15.
16.
17. /**
18.  * Create a graph from nodes-comments tables.
19.  *
20.  * @param array $edges The adjacentcy list of the graph
21.  * @return integer The number of interactions
22.  */
23. function build_edges_from_nodes(&$edges) {
24.
```

3. ábra A dokumentációban lévő forráskód

```
 *
 * @author Szerző neve
 * @version Verziószám
 * @package Csomagnév
 */
```

Ha pedig egy függvényről van éppen szó, ott nyilván a paraméterek és a visszatérési értékek a fontosak:

```
/**
 * Rövid leírás
 *
 * Hosszú leírás
 *
 * @param integer $param1 Mi ez
 ➥a paraméter?
 * @param boolean $param2 Mi ez
 ➥a paraméter?
 * @return integer Mit adunk
 ➥vissza?
*/
```

Fontos megjegyezni, hogy a leírások és a címkék opcionálisak, készíthetünk blokkot úgy, hogy csak ez egyiket-másikat írjuk bele. Ha mindezekkel elláttuk a *PHP* forráskódot, ami nem kis munka egy nagyobb lélegzetvételű programnál, sőt még értelmes leírásokra is futotta az energiánkból, akkor a *phpDocumentoron* már nem fog múlni a jó minőségű dokumentáció!

Parancssoros használat

Talán ahhoz vagyunk szokva, hogy a *PHP* programoknak webalapúnak kell lennie, ez korántsincs így. A *PHP 5*-ben már kimondottan a parancssort támogató új lehetőségek jelennek meg. A *phpDocumentor* jelen állapotában egy héjprogramot használ előtétként. Hogyan is használjuk ezt az előtétet?

Ehhez először is tudni kell, hogy mit tud kezdeni egy blokkokkal felokosított forráskóddal a program. Tud belőle számtalan formátumban dokumentációt készíteni (*HTML*, *PDF*, *CHM* és így tovább). A kimenetet lehetőségünk van tetszés szerinti megjelenéssel felruházni, készíthetünk sablonokat, így a projekt arculatához szabhatjuk a dokumentációt. Most mi először a kiegészített forráskódból legyártott dokumentálást – az alapvető működést vesszük szemügyre. Először is gondoskodjunk róla a kényelem érdekében, hogy a kitömörített *phpDocumentor* könyvtár legyen benne az elérési útban és adjunk futtatási jogot a *phpdoc* héjprogramra, mert a kitömörítés után nincs rajta x jog. Lépjünk be a forráskód könyvtárába és ott adjuk ki az alábbi parancsot:

```
phpdoc --directory . --output
➥/dokumentacio-utvonala/
```

Ez már elég a programnak, hogy az alapbeállításokkal elkészítse a *HTML* formátumú dokumentációt a megadott könyvtárba.

A *phpdoc*, mint minden tisztességes parancssoros program, kilistázza a kapcsolói listáját, egy érdekességre felhívnám a figyelmet.

Ha a `--sourcecode` kapcsolót megadjuk, akkor minden forrásfájlból kapunk egy szépen kiszínezett, sorszámozott verziót. Ennek több előnye is van, egyfelől a weben a látogatók ezt tudják böngészni, nagyon látványos. Másrészt pedig emlékezzünk a keretes írásra, a nyomtatott programlista nem öncélú!

A forráskód kinyomtatásához nincs más teendőnk, mint generálni egy dokumentációt a megfelelő kapcsolóval és a webböngészőből elindítani a nyomtatást. Az eredményt a *3. ábrán* láthatjuk.

DocBlocks – egy univerzális eszköz

Fontos látni, hogy ugyan a cikk a *phpDocumentor* alcímet viseli, viszont ezt a dokumentálási technikát számtalan más helyen fel lehet használni, így a *DocBlocks* használatával megbarátkozni biztosan hasznos tevékenység. Indulásnak a legjobb a projekt honlapján megtalálható példafájlokat megnézni, ott szépen látszik a technika alkalmazása: ➲ *manual.phpdoc.org/HTMLSmartyConverter/HandS/phpDocumentor/tutorial_sample3.pkg.html*. A felsoroltakon kívül számtalan módon lehet használni a *phpDocumentort*, például lehetőség van felhasználói dokumentációt is létrehozni. Az nem is kérdés, hogy egy ilyen szoftver dokumentációját mivel készítik. Érdemes megnézni a ➲ *manual.phpdoc.org* címet, a legjobb referencia ennek a szoftvernek!

Novák Áron
(aaron@szentimre.hu)

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

Google Earth

Földgolyó a monitoron

A 21. században talán már nincs ember, akit meglepne az alábbi kijelentés: műholdak tömkelege kémleli felettünk a Föld felszínét. Ezek közül jó néhány szerkezet profi optikával van ellátva, így a lehetőség adott: tegyük a monitorunkra az általuk készített fotókat!

Miért jó ez?

A *Google Earth* esetén a következőkről van szó: a műholdak professzionális lencséi mára olyan részletes képekkel szolgálnak a földi világról, melyeken akár egy „apró” személygépkocsi is tisztán beazonosítható. A részletekben dús fotók (bizonyos határok között) még nagyíthatóak is, számottevő „kockásodás” nélkül. Gondoljunk csak bele, milyen nagyszerű és hasznos lenne, ha egy valósághű, szabadon tallózható *„Föld* – modellre” feszített térkép szerint láthatnánk a terep fényképeit! Nos, a címben említett program pontosan erre a célra született: a műholdak képeire támaszkodva teszi könnyen böngészhetővé környezetünket.

A nekünk készített verzió

Az alkalmazás kezdetben csak a *Windows* felhasználóit szolgálta: nekik már régebben „megadatott”, hogy az *Internet* lehetőségeit kihasználva, az egyszerűen kezelhető *Google Earth* segítségével szemléljék a távoli fotókat. Később a fejlesztő csapat (szokásához hűen) ezt a projektet is elkészítette *Linuxra*: a szabad rendszerre fejlesztett *Google Earth* a ➲ *http://earth.google.com* címen érhető el, béta állapotú bináris változatában. A program három, számunkra fontos verzióban létezik – melyek közül a „legkisebb” mindmáig ingyenesen elérhető és használható (a másik két példány *GPS* képességeinek terén, illetve térképészeti funkcióiban különbözik az elsőtől). Az alkalmazás kódja teljes egészében át lett írva „részünkre”, így a letöltendő anyag (a *Google Picasaval* ellentétben) nem tartalmaz futásidejű *wrappert*. Ügyes munkáról van szó, hiszen a natív verzió nem különböztethető meg a *Win32* felületre írt társától, a hivatalos dokumentumok pedig az alábbi átlagos környezetet ajánlják használatához:

- *Linux* rendszermag legalább *2.6* verzióban
- *glibc* könyvtár *2.3.5* azonosítóval
- *X.org* szerver, legalább *6.7* verzióban
- Szélessávú *Internet* kapcsolat
- *512 Mbyte* központi memória
- *2 Gbyte* szabad terület a merevlemezen
- *3D* videó hardver, legalább *32 Mbyte* memóriával

Ha számítógépünk (és operációs rendszerünk) paraméterei minden ponton teljesítik az elvárt szintet, akkor bátran nekikezdhetünk a program installálásának! Ehhez nem kell mást tennünk, mint letölteni a (nagyjából *15 Mbyte* méretű) *GoogleEarthLinux.bin* telepítőt, majd ezután *root* jogkörrel lefuttatni. A közel *50 Mbyte* helyigényű állományok alapértelmezés szerint az */usr/local/google-earth* útra kerülnek,

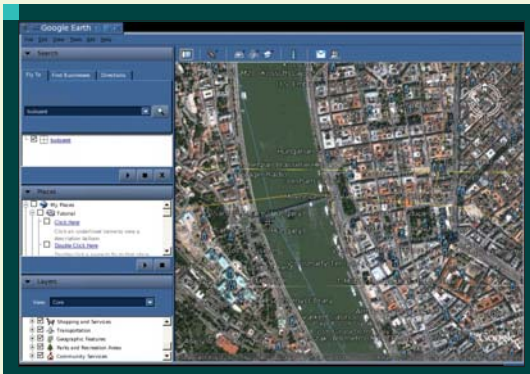
1. ábra A Google Earth linuxos telepítője

indító kötéssel az */usr/local/bin* mappában. Az alkalmazás egy konzolon, felhasználóként kiadott googleearth paranccsal hívható életre. A *Google Earth* természetesen *Live* terjesztéseken is működésre bírható, minek megfelelően akár egy *Slax* disztribúción is sikeresen használható!

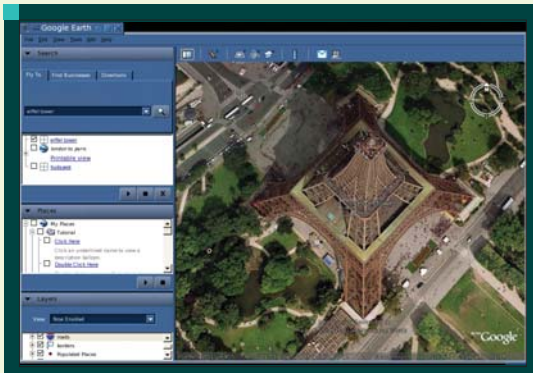
A használatba vétel

Indítás után a kezelőpanel és a *Földgolyó* távoli képe fogadja az érdeklődőket. A *3D*-s *Földmodell* a bal egérgombbal vonszolva szabadon forgatható, az egérgörgő tekerésével pedig az aktuálissá tett center hozható közelebb (illetve vihető távolabb). Bármely közeli kép könnyen megdönthető a középső egérgomb lenyomásához kötött vonszolással (két dimenziós fényképekről van szó, így a döntést természetesen nem igazi *3D* renderelés biztosítja, hanem egy perspektivikus leképezési trükk).

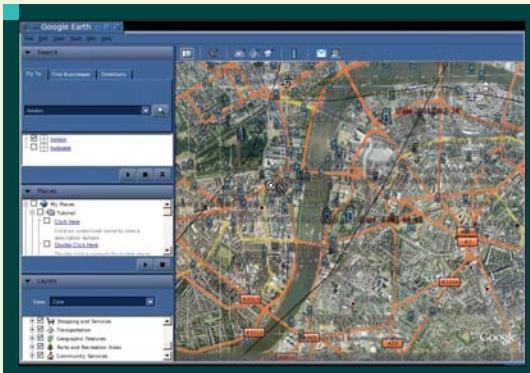
2. ábra Íme, a Golyóbis



3. ábra Budapest, madártávlatból



4. ábra Az Eiffel torony



5. ábra London fő útjai

6. ábra Londonból, Párizsba...

A munkaterület alsó részén olvashatóak le a kiszemelt helyszín koordinátái. A bal oldali blokkban szerepel a *„Search”* (*Keresés*) menüpont, almenüivel együtt. E lehetőségek között egy városnevet beírva, a *3D* modell úgy fordul el, hogy a keresett terület kerül centerbe – majd az automatika igen látványosan „odarepít” a felszín közelébe.

A keresés beviteli mezőjébe nem csak városok, hanem híresebb építmények nevei is beírhatóak, például az *Eiffel Torony* (*„Eiffel Tower”*), vagy akár a *Brooklyn Híd* (*„Brooklyn Bridge”*). A *Keresés* további lehetőségei két tetszőleges város közötti optimális útvonalat is megjelölik! Szintén bal oldalon található a *„Layers”* (*Rétegek*) menü, melyet használva a leképezett terepen plusz információk és egyéb kiegészítések kapcsolhatóak (úgy, mint a lakott területek kulturális intézményeinek kijelzése, utcanevek feliratozása, vagy éppen a fő utak és államhatárok szemléltetése).

A grafikus interfész sajnos nincsen honosítva, de akinek valamely idegen nyelv erőssége lenne, a *„Tools / Options / General”* menüben öt nyelv (Angol, Német, Spanyol, Olasz, Francia) közül választhatja ki a magának leginkább megfelelőt. Ennél egyszerűbben és lényegre törőbben nem lehet megvalósítani a Föld felszínének realisztikus böngészését: a cikkhez mellékelt képek talán önmagukért beszélnek. Személy szerint csupán egyetlen dolog zavar: jelenleg nem minden terepről érhető el részletes fotó, így például a szülővárosomról egy szerény felbontású képpel kell beérnem. Ez a hiányosság viszont törvényszerűnek is mondható, hiszen a *40.000km* kerületű *Golyóbis* elképesztően nagy felületű, emiatt „nem jut” minden négyzetkilométerre borotvaéles, nagy felbontású grafika.

A használat értéke

A felhasználó minden bizonnyal hasznos órákat fog eltölteni a program előtt ülve, akár vonzódik a földrajzhoz, akár idegenkedik tőle. A *Google Earth* tartalmas kikapcsolódást ígér, mivel ötletes megoldásokra és látványos leképezésre épít, miközben egyszerű kezelhetőséggel vértezték fel. Valamiért már az is szórakoztató, ahogyan (például) a lakóházunkat keresgéljük türelmetlenül a fotókon...

Egyetlen buktatóként szinte csak az *Internet* „minőségére” kell ügyelni, mivel egy *512Kb/s* sebességű *ADSL* kapcsolat jelenti a belépőt a kompromisszummentes használathoz.

Az idejét múlt vonalkapcsolt, illetve az „öregebb” bérelt vonalak körében ez a csodálatos világ élvezhetetlenné válik: a szerény átviteli sebesség miatt nem képes a fényképeket hatékonyan közvetíteni felénk.

Kovács Zsolt (kovi@linuxforum.hu)

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

GoboLinux – Egy alternatív Linux terjesztés

Manapság nagyon sok Linux-terjesztés létezik, ezért nem is tűnik fel szinte senkinek ez a brazil disztribúció (szerintem méltánytalanul). Az egész rendszernek nagyon érdekes filozófiája van, ezért nem ajánlható azoknak, akik szeretik, ha minden lehetséges beállításhoz van akár szöveges, akár grafikus felhasználói felület és nem kedvelik a szokatlant.

Viszont aki szereti, ha a legtöbb programja forrásból van telepítve (akár optimalizáció, akár egyéni konfiguráció vagy akármi más miatt), viszont nem akar „igazi forrás-alapú" disztribúciót (*Gentoo*, *SourceMage*, *Lunar Linux*, stb.) használni, vagy szeretnek „bütykölni" vagy lázadóbb típusúak és szeretik a szokatlan dolgokat kipróbálni, azoknak a *GoboLinux* ajánlható.

A telepítés

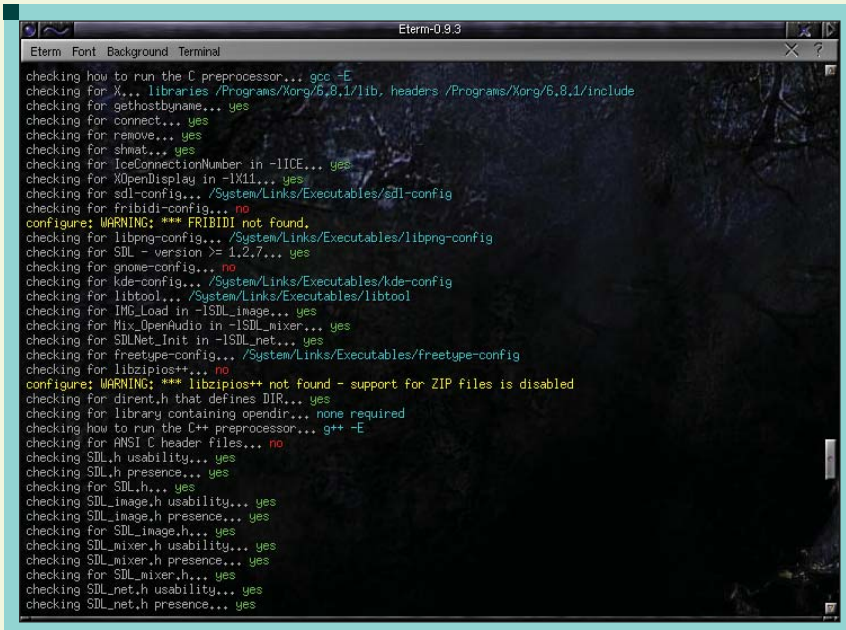
Először a ➲ *www.gobolinux.org* címről töltsük le a megfelelő ISO fájlt, ezt írjuk CD-re és bootoljunk róla. A telepítő-CD egyben *Live-CD*-ként is funkcionál, így vészhelyzetben is jól jöhet. Az indítás végén egy teljes értékű Linux alatt találjuk magunkat, és a `startx` paranccsal egy *KDE*-felületet indíthatunk. Az asztalon találunk egy *Install GoboLinux* ikont, amelyet elindítva a telepítés megkezdődik. Ha nem akarunk (vagy nem tudunk) grafikus felületet használni, akkor a konzolon adjuk ki az `Installer` parancsot (amelyről a rendszer tájékoztat is bennünket).
Maga a telepítés egyértelmű, azaz aki nem először fut neki a dolognak, az biztosan el fog vele boldogulni, így ezt nem is részletezném. Egyetlen egy megjegyzést azért kell tenni: már a bevezetőben is említettem, hogy a *GoboLinuxnak* érdekes filozófiája van. Ezért a *root* felhasználó neve alapértelmezetten nem *root*, hanem *gobo*, viszont ha nagyon ragaszkodunk a hagyományokhoz, nyugodtan lehetünk *root* is.
Amíg a telepítés zajlik, olvassuk tovább a cikket, hogy lássuk, mi az, ami ránk vár illetve miért lehet jobb választás ez a disztribúció, mint a megszokottak.

Bináris disztribúciók és a forrásból való telepítés

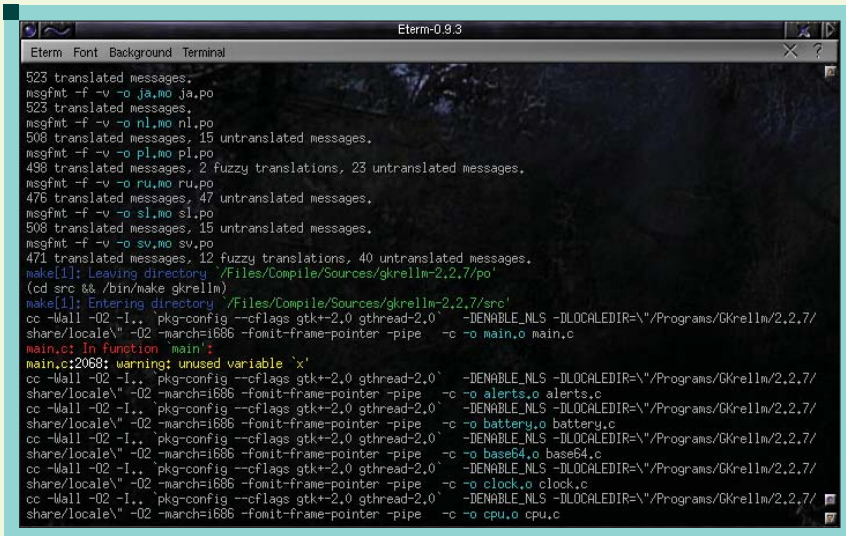
A bináris disztribúciók nagy többségének egyik hátránya, hogy nemigen támogatják, ha forrásból telepít a *root* bármit. Ez a nem-támogatás akkor (is) jelentkezhet, ha egy *X* csomag (egyik) *Y* függősége forráskódból van telepítve, így *Y* nem is jelenik meg a csomagkezelő (*dpkg*, *rpm*, *installpkg*, *pacman*, ...) adatbázisában és *X* telepítésekor hiányolja. Ilyen eset leginkább akkor fordulhat elő, ha egy olyan programot szeretnénk, ami a kedvenc disztribúciónkhoz nem elérhető csomagban, így kénytelenek vagyunk fordítani és a már említett *Y* függőségből frissebb kell neki, mint ami csomagból elérhető. Sokszor a felülírás sem szerencsés, mivel a csomagkezelők egy része a telepített fájlok méretét is tárolja.
Ennek egy lehetséges feloldása, ha a felhasználó a rendszer működéséhez szükséges alapcsomagokat és néhány (vagy sok), viszonylag kevés függőséggel rendelkező programot csomagból telepít, a többit pedig forrásból, tehát a rendszerhez tartozó csomagkezelőt ezután gyakorlatilag nem használja. Ebben az esetben a csomagkezelő előnyeitől is meg kell válnunk, azaz nem fogjuk tudni, miket telepítettünk fel, melyik fájl melyik csomaghoz tartozik, egy csomagban milyen fájlok vannak, sőt, a telepített programok verziószámát a legtöbb esetben nem is tudjuk. Ezen hátrány felismerésével egyidejűleg egy kívánságot fogalmazhatunk meg hallgatólagosan: olyan keretrendszer kell, amely támogatja a forrásból való telepítést és nyilván is tartja a telepített programokat verziószámukkal együtt (a mai modern korban nevetséges lenne mindezt papírral és ceruzával rögzíteni), esetleg a bináris csomagokat is bírja kezelni.

A GoboLinux csomagkezelése

Az alapötlet roppant egyszerű, ennélfogva zseniális: *„a csomagkezelő maga a fájlrendszer"* (idézet a honlapról). Azaz: rakjunk minden programot külön könyvtárba, így egyszerűen tudni fogjuk, ki hova tartozik és tudni fogjuk azt is, mik és milyen verzióval vannak feltelepítve. Egy példával megvilágítva rögtön világos lesz az alapelv: legyen a telepítendő program a *bash*, ennek is a 3.1-s verziója. *GoboLinux* alatt ez azt jelenti, hogy van egy */Programs* könyvtár, itt van az *összes* telepített program. Tehát a *bash* a */Programs/Bash/* könyvtárban.
A beállításai (ami normális esetben a */etc*) a */Programs/Bash/Settings* alkönyvtárban, a kifejezetten 3.1-es



1. ábra PrepareProgram színes kimenete



2. ábra A make színes kimenete

verzióhoz tartozó állományok pedig a */Programs/Bash/3.1/{bin,doc,man,lib,include,...}* könyvtárakban.
Ahhoz, hogy a megszokott módon lehessen futtatni a parancsokat („begépelem, Enter, és fut”), két lehetőség van: vagy az összes telepített programban szereplő ***bin*** (és ***sbin***) könyvtárt felvesszük a PATH környezeti változóba (ami valljuk be, sok telepített program esetén használhatatlan), vagy pedig egy könyvárban létrehozok minden egyes ***bin***-beli állományról egy szimbolikus linket, és ezt a könyvtárat veszem fel a PATH-ba. Nyilván a második út a járható, és egyszerűbb is kezelni. *GoboLinux*-ban is ez történik: minden csomag aktuális verziójában levő ***bin*** könyvtár összes fájljára mutat egy szimbolikus link a */System/Links/Executables* könyvtárban, amire pedig egy-egy szimbolikus lánc mutat */bin* és */usr/bin* néven. Hasonló módon a ***lib***-ek a */System/Links/Libraries*-ba, az ***include***-ok a */System/Links/Headers*-be, stb. linkelődnek. Persze a többi könyvtár (*man*, *share*, ...) hasonlóan és mindegyik */System/Links*-belire mutatnak a tradícionális */usr/{bin,include,lib,man,share,...}* linkek.

A verziókezelés is egy egyszerű ötlet alapján megy: az aktuális verzióra egy szimbolikus link mutat ***Current*** néven, tehát ha a ***bash***-ból a 3.1-es verzió van telepítve, akkor a */Programs/Bash/Current* a 3.1-re mutat.
Az ilyen módszerrel megvalósított csomagkezelés legfőbb előnyei a következők:

- Egy programból egyszerre több verzió is lehet fenn, és mindig az a ***Current***, amelyiket akarjuk.
- Mindenféle bűvös csomagkezelő nélkül tudunk lekérdezéseket végrehajtani az alap-parancsokkal (`ls`, `find`, ...).
- Nincs a csomagkezelőnek a szó hétköznapi értelemében vett adatbázisa különféle egzotikus formátumban, amit esetleg egyszerre csak egyvalaki nyithat meg.
- Minden egyes fájlról könnyen meg tudjuk állapítani, hogy melyik csomagnak a része.
- Nincs különbség, hogy egy program forrásból vagy bináris csomagból került a rendszerbe.

Tehát az alapötlet zseniális, már csak meg is kell valósítani. Azzal a néhány (körülbelül száz) bináris csomaggal nincs is gond, azt az `InstallPackage` parancs megoldja. Mi a helyzet a forrásokkal?

Forrásból való telepítés, csomagok készítése

A legtöbb forrás telepítése a legtöbb (és a hagyományos) esetben a

```
./configure && make && make
➥install
```

hármassal történik (esetleg még utána `ldconfig`, `makewhatis -u` vagy más egyéb). A *./configure* egyik előnye, hogy legtöbbször meg lehet adni neki különféle könyvtárakat, hogy a lefordított szükséges részeket hova szeretnénk installálni (a `--prefix`, `--libdir`, `--mandir`, stb.).

Ezt mi is megoldhatnánk manuálisan minden egyes programnál, ami elég nehézkes és nem az ilyen aprócseprő munkák miatt használunk ***Linuxot***. Ezen (és sok más) problémára a ***GoboLinux*** nyújt erre egy szkript-gyűjteményt, ennek egyik része a `PrepareProgram` és a már említett `InstallPackage` is. Az előbbi szkript megfelelően paraméterezi a `./configure`-t, tehát helyette a `PrepareProgram Bash 3.1`-et kell futtatni (a `configure` egyéb opcióit, például `--enable-gui`, `--enable-shared`, stb. ez után adhatjuk meg, amit a `PrepareProgram` átad a `configure`-nak). Ezután következik egy csoda: a ./configure fut le szintaktikailag színezve (syntax highlight)! Tehát például a ***no***, ***error*** kulcsszavak pirossal, a ***warning***-ok sárgával, stb (*1. ábra*).
Ezután a szokásos `make` következik, ami szintén színesen fut le (mivel a `make` egy alias a `ColorMake` szkriptre), tehát nagyon jól követhető (*2. ábra*), mikor mi a kimeneti állomány, milyen ***warning***-ok és ***error***-ok keletkeznek a fordítás során.
Miután a programot lefordítottuk, fel is kell telepíteni. Ezt (általában) a

```
make install
```

paranccsal megtehetjük. Ezután a szimlinkeket kell létrehozni, amit nagyobb programok esetén hosszadalmas lenne egyesével. Erre megoldás a `SymlinkProgram` nevű szkript, amelynek a használata az eddigi példával:

```
SymlinkProgram Bash 3.1
```

A `SymlinkProgram` nagyon jól paraméterezhető, tehát egyedi esetekben is jól működik. Egy nagy hiányossága van, amit könnyen lehet pótolni a szkript átírásával: a ***libexec*** könyvtárat egyszerűen kihagyja (és a `PrepareProgram` se kezeli). Ha egy programot el akarunk távolítani, akkor a `DisableProgram` használható, amely megszünteti (törli) a linkeket, majd ezután egyszerűen az `rm -r` paranccsal töröljük az egész könyvtárat (itt jegyezném meg, hogy az összes szkript ***bash***-ban íródott, tehát könnyen követhető és módosítható

azok számára is, akik nemigen ismerik a szkript-nyelveket).
Viszont ha nincs `./configure`, akkor kénytelenek vagyunk magunk megadni a telepítési útvonalat vagy egy beállított `PREFIX` (esetleg `prefix`) környezeti változóval futtatva a `make` parancsot (ez a jobbik eset) vagy pedig magát a `Makefile`-t kell kézzel szerkesztenünk (esetleg a sed-et vagy bármilyen szövegszerkesztőt igénybe véve). Ebben az esetben célszerű létrehozni a megfelelő könyvtárszerkezetet a */Program*-ban (mivel néhány program nem hozza létre magának a könyvtárakat és hibát jelez, ha nem tudja a nem létező helyre bemásolni a megfelelő dolgokat), ezt pedig a

```
PrepareProgram -t Bash 3.1
```

paranccsal tehetjük meg. Ebben az esetben a `-t` opció eredményezi azt, hogy csak a könyvtárstruktúrát hozza létre.
A ***GoboLinux*** a csomagokat (alapértelmezetten) ***.tar.bz2*** formátumban tárolja, így könnyen kezelhetőek. Csomagok készítése (az általam eddig használt disztribúciókkal összevetve, és szerintem amúgy is) nagyon egyszerű a ***CreatePackage*** szkripttel, egyszerűen megadjuk, hogy melyik könyvtárból készítsen csomagot, ő pedig mindent megcsinál. Még függőségi listát is készít, amit az `InstallPackage` értelmez, és megszakítja a telepítést (hacsak nem utasítjuk ennek ellenkezőjére), ha valami nem teljesül. A függőségek meglétének ellenőrzése lényegében annyi, hogy a szkript megnézi, hogy az adott program telepítve van-e a megfelelő verziószámmal, tehát a */**Programs**/**Deps**/**depsVerzió*** könyvtár létezik-e.
Hogy ez bináris csomagból vagy forrásból van-e fenn, teljesen mindegy, ellentétben a legtöbb bináris terjesztéssel.
A honlapomon (➲ ***www.udvzsolt.extra.hu***) néhány csomag található ***GoboLinuxra***, valamint a hivatalos honlapon linkeket találunk hivatalos és nem hivatalos csomagokat tartalmazó tárhelyre/tükrökre. Ne számítsunk sok csomagra, hiszen ez a terjesztés inkább forrás alapú, mint bináris.

A könyvtárszerkezet

A ***GoboLinux*** könyvtár szinten is „szakít a ***UNIX***-os tradíciókkal”, tehát más a könyvtárszerkezet, viszont a megszokott szerkezet is elérhető: például a */**dev*** könyvtár a */**System**/**Kernel**/**Devices***-re, a */**etc*** a */**System**/**Settings***-re, a */**proc*** a */**System**/**Kernel**/**Status***-ra egy-egy link. A fő ***home*** könyvár nem a */**home***, hanem a */**Users*** (a ***root***-nak is itt van a ***home*** könyvtára), a szokványos csatolási pontok a */**mnt*** ill. manapság a */**media***-val szemben */**Mount***. Ha a home-könyvtár külön partíción van és másik ***Linux*** alatt is ezt használjuk,

3. ábra Scandisk-téma bootoláshoz

akkor célszerű a */Users*-re egy linket létrehozni */home*-ként, hogy ne legyen kavarodás.

A boot-olás

Mint ahogy *GoboLinux* alatt eddig szinte semmi se a megszokott, a bootszkriptek sem azok. Nem a leginkább elterjedt *SysV init* szkripteket használja. A bootoláshoz szükséges állományok a */Programs/BootScripts/VerzióSzám* alatt találhatóak, míg a szkriptek a */Programs/BootScripts/Setting/BootScripts*-ben. Ami bootolás során történik:
a */Programs/BootScripts/Settings/Boot Scripts/BootUp* nevű szkript lefut, majd a *Console* és esetleg a *Graphic* (az */etc/inittab* szerkesztgetése nem árt!). A bootolás beállításai a */Programs/BootScripts/Settings/Boot Options*-ben vannak. Itt lehet megadni, hogy milyen kernelmodulokat kell betölteni, a hálózati beállítások, a konzol betűkészlete is itt szerepel. És itt is egy érdekesség: a bootoláshoz (nem framebuffer vagy bootsplash alapon, hanem egyszerű konzolon) még témát is megadhatunk, tehát az üzenetek milyen "körítéssel" jelenjenek meg: lehet *AppleII, CheckList, GoboText, GoboText-II, Hat, Progress, Progress-II, Quotes, ScanDisk, Slack* és *SplitScreen* közül választani (igen, a nevek ismerősek, tehát pl. a *Scandisk* téma úgy néz ki, mintha a *ScanDisk* lemezellenőrző futna más szöveggel, lásd *3. ábra*), sőt mivel az egész nagyon általánosan lett megírva, mi is írhatunk egy sajátot (az alaptémákat célszerű tanulmányozni), amelyet hasznos függvények segítenek (színkezelés, kurzor pozicionálása, stb.). A témákat ki is lehet próbálni (még bootolás előtt) a

```
TestBootTheme Theme_Neve
```

paranccsal. A *3. ábra* is így készült.

Összefoglalás

A disztribúció teljes bemutatása túllépne a cikk terjedelmén, és nem is ez volt a cél, hanem csak figyelemfelkeltés és kedvcsinálás.
A terjesztésnek sok hasznos és érdekes tulajdonsága, beállítási lehetőségei vannak, igazi „bütykölős" rendszer, olyannyira, hogy például még a véletlenszám-generátor inicializálását is nekünk kell megírni (a részletekkel kapcsolatban `man urandom`), tehát kezdőnek egyáltalán nem ajánlható (a *FAQ*-ban szólnak is erről).
Eddig elsősorban az előnyökről volt szó, azért essen szó a hátrányairól:

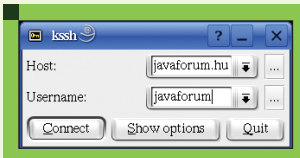
- A `SymlinkProgram` és a `DisableProgram` a *libexec* könyvtárat nem kezeli, és egy-két sorral mindkettőt bővíteni kell, ahelyett, hogy lenne egy konfigurációs fájl, amiben tárolná, hogy melyik könyvtár tartalmát hova linkelje (amit a saját *Linux From Scratch* rendszeremben meg is valósítottam).
- A csomagkezelés részleteiből kifolyólag nincsenek „programtípusok", tehát nem tudjuk eldönteni egy programról, hogy ő játék vagy felhasználói program vagy valami más (ez is meg van valósítva az *LFS*-emben).
- Sok program nem szereti, ha az állományok nem ott vannak, ahol ő feltételezi vagy pedig a szimbolikus láncokat nem kezeli (ilyen a `makewhatis` program, amit egy kis átírás után rábírhatunk a szimlinkek felvételére is).
- Valahol nagyon általánosan (absztraktul) vannak megírva a szkriptek (például bootfolyamat), valahol pedig gyakorlatilag „beégetve" minden (csomagkezelő szkriptek). Véleményem szerint az előbbi alternatíva a jobb, de a vegyes megoldás a lehető legrosszabb, mivel így a rendszer nem tűnik egységes egésznek.
- A források keresése/letöltögetése nehézkes (mindent nekünk kell megkeresni), függőség-kezelés gyakorlatilag nincs (ellentétben például a *Gentoo* ebuildjeivel), ami internet nélkül egy kicsit keserves. Viszont ez a „Csináld magad" mozgalom híveinek inkább előny.

Összességében egy nagyon jól használható, nagyon érdekes és ami sokak számára fontos, nagyon gyors (ami leginkább a forrás-alapnak köszönhető) disztribúciót ismerhetünk meg a *GoboLinux* személyében.

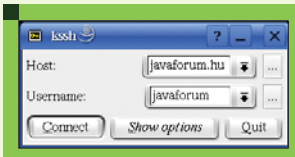
Udvari Zsolt
(udvzsolt@gmail.com)
25 éves vagyok, egy gimnáziumban tanítok matematikát és fizikát. A Linuxszal először 2004 elején találkoztam, az UHU Kamionja volt, ami elgázolt. Azóta 4-5 disztribúciót hosszabban is használtam, jelenleg egy saját építésű LFS-t nyúzok.

KDE alkalmazások (6. rész)

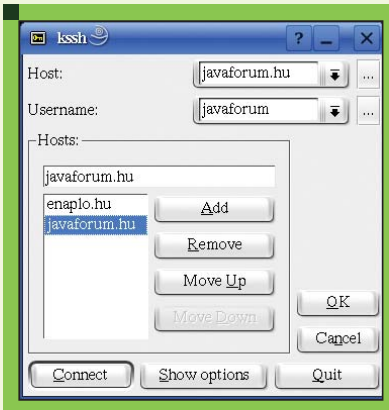
KSSH – Grafikus SSH kliens



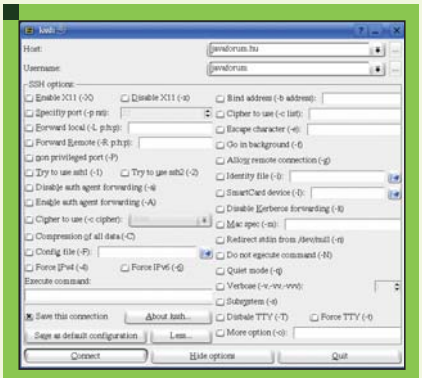
1. ábra A program indítása



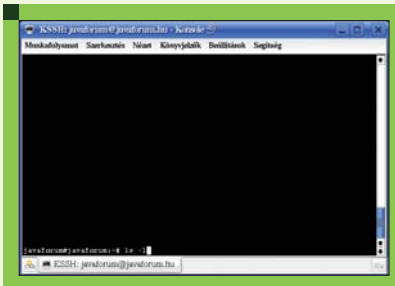
4. ábra Aktív beállítás



2. ábra Új gép hozzáadása



3. ábra Beállítási lehetőségek



5. ábra KSSH Konsole alatt

A *KSSH* program nem része a *KDE* grafikus munkakörnyezetnek, általában a külön feltelepíthető programot között találjuk meg a legtöbb terjesztés esetén. Ha mégsem találnánk meg, akkor a ➲ *http://kssh.sourceforge.net/* címről letölthetjük és lefordítás után telepíthetjük a gépünkre. Elindítása után egy puritán kis ablak fogad minket (*1. ábra*), amelyben kettő legördülő menü és néhány nyomógomb kapott helyet. A legördülő menükkel tudjuk kiválasztani a gép nevét – ahova kapcsolódnánk, illetve a felhasználó nevét – amely néven belépnénk a megadott gépre.
Mind a két legördülő menü mellett van egy-egy három pontot tartalmazó nyomógomb, amelyet megnyomva a kis ablak mérete megnövekszik, s itt hozzáadhatunk vagy törölhetünk neveket, illetve módosíthatunk a sorrenden mindkét legördülő menü tartalmát tekintve.

Fontos nyomógomb a kis ablak alján lévő *„Show options”* feliratú, amely elénk tárja az *SSH* program gyakoribb beállítási lehetőségit, a *„More...”* gombra kattintva pedig a lehetőségek megduplázódnak (*3. ábra*).
A *KSSH* előnye, hogy a listában szereplő gépnév-felhasználónév pároshoz rendeli a beállított opciókat, vagyis a következő belépéskor már nem kell külön beállításokat végeznünk, ha egy bizonyos gépre való belépéskor port-átirányítást vagy *X11* adattovábbítást szeretnénk használni.
Az alapértelmezettől eltérő beállítás esetén a program dőlt betűkkel szedi a *„Show options”* gombot. A *KSSH* program használatához ismernünk kell az *SSH* program kapcsolóit, mivel a *KSSH* egyszerűen csak meghívja a *Konsole* nevű *KDE* konzolt, és abban futtatja a megadott paraméterek szerinti *SSH* programot (*5. ábra*).
A meghívott *Konsole* alatt minden megszokott tevékenységet el tudunk végezni, amire esetleg szükségünk van (kódlapváltás, 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

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

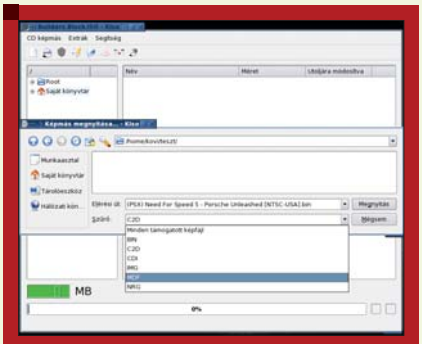
KIso – Az ISO mindenes

Bizonyára sokan dolgozunk nap mint nap lemeztükrökkel, CD és DVD lenyomatok különböző változataival. A sokféle formátum hatékonyságát illetően megoszlanak a vélemények, de hogy teljes legyen a zűrzavar, Linux alatt néhány programspecifikus lenyomat igen nehezen kezelhető...

Néhány hónappal ezelőtt a magazin hasábjain keresztül bőséges információhoz juthattak az érdeklődő Olvasók ***p2p*** fájlcsere, valamint ***CD*** írás témaköréből egyaránt. Aki vette a fáradtságot, és szánt néhány percet e hasznos tudnivalók áttanulmányozására, esetleg a gyakorlatban is megpróbálta kamatoztatni frissen szerzett tudását, bizonyára összefutott néhány adathordozó tartalmának egy állományba fogott „bitre pontos" lenyomatával. Itt említhetném akár a nagy múltú ***ISO*** képeket, vagy az idővel együtt köztudatba ivódott ***BIN/CUE*** megoldásokat is: nem biztos, hogy a két kezemen meg tudnám számolni, mennyi életképes formátummal találkozhat a felhasználó.
Sajnos azonban a helyzet korántsem rózsás... A legnépszerűbb égető programok saját eljárásokat használnak lemezlenyomatok készítésére – de akár a közismert, ***Alcohol120%*** elnevezésű w***indowsos CD*** emulátor is egyedi lemeztükröket használ saját formátumaként. Nem tisztem eldönteni, hogy melyik a leghatékonyabb formátum, ennek ellenére személy szerint naponta összefutok az ismerős kiterjesztésű ****.iso***, ****.nrg***, ****.bin***, ****.img***, ****.mdf*** , ****.ccd*** fájlokkal egyaránt, melyek használata mára már operációs rendszertől függetlenül is nélkülözhetetlen. Gondoljunk bele: mennyire megszokott és célratörő módszer egy tetszőleges ***CD*** tartalmát például ***Multimedia Descriptor*** állományként elérnem a „világ másik végéről" úgy, hogy azt mindenféle manipuláció nélkül azonnal ki tudjam írni saját adathordozómra – az eredeti lemez pontos mását visszanyerve.
A dolog szépséghibája mindössze annyi, hogy az előbb említett ***Alcohol120%***, vagy akár a ***Nero***, a ***CloneCD*** elismert és elterjedt megoldásai célszoftverek nélkül nem (vagy csak igen nehezen) kezelhetőek ***Linux*** alatt natívan. A cél tehát adott: konvertáljuk a problémás lenyomatokat könnyen használható formátumba!

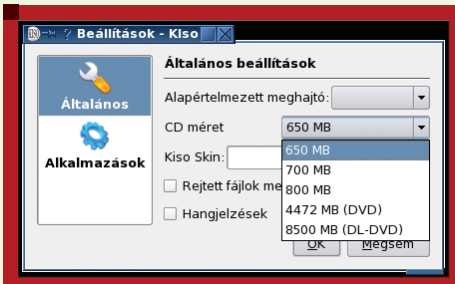
A KIso

„A ***KIso*** egy ***KDE3.x*** verzióhoz készített grafikus interfész, mely ***Linux*** alatt a legkönnyebb megoldást kínálja az előző nehézségekre"... Nagyjából ilyen ajánlással kezdődik a címben említett program hivatalos ismertetője.
A ***KIso*** szerzőjét ***Stephan Gansnek*** hívják. Egy Németországban élő informatikus mérnökről van szó, aki programozott már ***Commodore 16***-on át akár ***Amiga500***-as masinákat is, ***C*** nyelven pedig nagyjából tíz éve alkot. Mindez azért fontos számunkra, mert a projekt (***v0.2*** verziója óta) teljes egészében ***C++*** nyelven íródik (ezt megelőzően ***GAMBAS*** volt az alappillér). A ***KIso*** jelenlegi állapotát fejlesztője ***v0.8.3*** azonosítóval jelöli, ami alapján könnyen észlelhető: a kód még nem érte el stabil állapotát. Ennek ellenére tudása meggyőző, puritán felületét (is) meghazudtolóan.
A program hivatalos lapja a ➲ ***http://kiso.sourceforge.net URL*** mögött található, innen indulva érhető el a ***GPL*** licenc szerint terjesztett aktuális forrásállománya is.

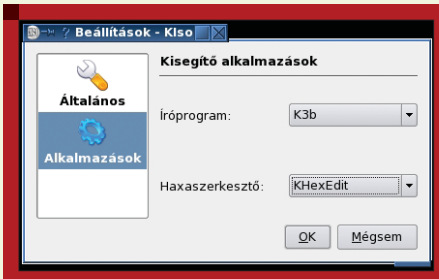


1. ábra A program felülete egyszerűbb nem is lehetne

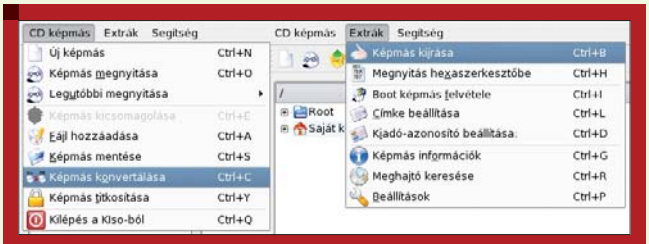
Lefordítása nem ördöngösség: a megszokott, ***root*** jogkörrel kiadott `./configure`, `make`, `make install` parancsokkal lehet életet lehelni a szabadon letöltött, kicsomagolt ***tarballba***.
A konfiguráló szkript ***libkde***, ***libcdio***, ***sudo***, ***mkisofs*** csomagokat keres függésként, így ezekkel mindenképpen rendelkeznünk kell ***Linux*** rendszerünkben.
Opcionális függésként, amennyiben jelen vannak ***libmcrypt*** és ***Mcrypt*** csomagok is, a ***KIso*** képes lesz az önmaga által létrehozott lenyomatokat titkosítani. A legfőbb terjesztésekhez (***Slackware***, ***Debian***, ***Mandrake***, ***SuSE***) és ezek származékaihoz készült, előre felépített bináris csomagok számtalan helyen elérhetőek a világhálón, így érdemes kísérletezgetni a fordítás megkerülésével. A telepített kód egy terminálon (szuperfelhasználóként) kiadott `kiso` paranccsal indítható (tulajdonképpen e speciális jogkör miatt van szükség a függés egyik tárgyát képező ***sudo*** csomagra).



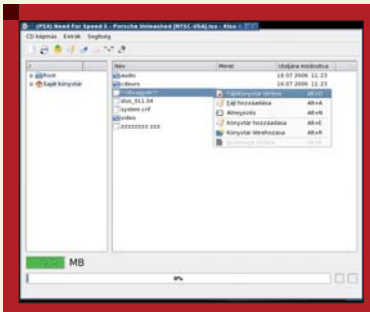
2. ábra A DL DVD lenyomat sem jelent nehézséget



3. ábra A segédprogramok beállítása



4. ábra Letisztult menürendszer



5. ábra A konvertált tartalmat új fájllal bővítettem

A használatba vétel

Azt hiszem sokakkal egyetértünk abban, hogy ritka az olyan program, melynek használati leírása rövidebb az installálás ismertetésénél. A ***KIso*** ennek a furcsa helyzetnek az egyik szerencsés iskolapéldája: ilyen egyszerű kezelhetőséggel ritkán találkozni linuxos körökben: nincs olyan művelet, amely két kattintásnál többe kerülne. Az *„Extrák”* menüpontban található beállítási lehetőségek mindössze a lemeztükrök maximális méretére, valamint a segédprogramok kiválasztására korlátozódnak.

A *„CD Képmás”* menü tartalmazza a lenyomatok létrehozási, manipulálási és konverziós lehetőségeit, melyek (szó szerint) egy kattintással hívhatóak életre. ***ISO*** állományt egyaránt lehetséges kreálni optikai adathordozóról, illetve általunk kijelölt fájlokról és könyvtárszerkezetről (természetesen munkánkhoz adhatunk indítható tartalmat is, a megfelelő ***IMG*** állományt kitallózva az extra lehetőségek között). A manipulációs lehetőségek kimerülnek a megnyitott ***ISO*** tükrök tartalmának kurtításában és bővítésében, ahol az ismert „húzd és ejtsd” műveletek segítik a munka menetét.

A konverziós menüben mindössze annyi dolgunk akad, hogy egy bemenetként használható állományt megnyitunk – a többi teendőt már automatikusan elvégzi helyettünk a program. Csupán egy dologra kell odafigyelni: az átalakítandó fájlnak írható területen kell lennie, mivel (kérdés nélkül) erre az útra fog kerülni az abból készült ***ISO*** lenyomat.

A legfőbb lehetőségek között található a kriptográfia kapcsolhatósága is. Nézzétek meg a mellékelt képeket, a ritka módon egyszerű kezelőfelület szinte vezeti a felhasználó kezét...

A táblázatban olvasható lista kifejezetten kritikus nehézségek megoldását ígéri - nem alaptalanul. Csupán egy kényes pontot ragadnék ki: a sok egyéb mellett támogatott *„DiscJuggler”* módszerrel tárolt tartalmat sokszor még ***Windows*** alatt is körülményes kezelni... A ***KIso*** alapértelmezett kimenete az említett ***ISO*** formátum, melyet ***Linux*** alatt hurok eszközként bármikor csatolhatunk a szükséges pontra, a `mount /elérési_út/lenyomat.iso /csatolási_pont -o loop` parancs kiadásával.

Tapasztalatok

Nem szerencsés összehasonlításokra hivatkoznom, de nem hallgathatom el az Olvasótól: ugyanazon az ***AMD64*** bázisú gépen ***Windows XP SP2*** felületen kipróbáltam néhány komolyabb, kereskedelmi jellegű ***ISO*** konvertáló programot, és ***Linux*** alatt a ***KIso***-t egyaránt. A projektek hatékonysága bizony összemérhető volt, az átalakítások / módosítások ideje pedig (bemeneti formától függően) ***20%*** határon belül mozgott, plusz és mínusz irányban vegyesen.

A ***KIso*** így mindazoknak a figyelmét megérdemli, akik ***Szabad Rendszer*** alatt szeretnék megoldani az ***Internetről*** letöltött (vagy a barátaiktól, ismerőseiktől kapott) idegen lenyomatok konvertálását és átformálását.

A projekt különböző verziói minimális (és azonosítónként eltérő) hiányosságokkal rendelkeznek a fejlesztő eredeti elképzeléseihez képest: ezekről az ismert „hiányosságokról” a ***weblap*** ide vonatkozó része szimpatikus módon, mellébeszélés nélkül tájékoztatja az érdeklődőket.

A konvertálások előtt érdemes lehet áttanulmányozni az egyes formátumok közötti technikai különbségeket is, hiszen bizonyos dolgokat műszakilag lehetetlen kivitelezni. Zenei sávokat például ne akarjon senki az adatai mellett *ISO* formában tárolni...

Kovács Zsolt (kovi@linuxforum.hu)

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

Firefox kiterjesztései (3. rész)

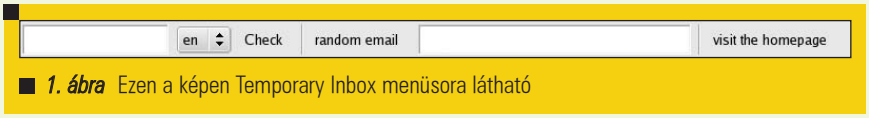
A Firefox kiterjesztéseinek száma az utóbbi időben is folyamatosan növekedett. Sajnos ezek közül egy jó pár egy adott honlap reklámozást szolgálja, de ha ezeket kihagyjuk akkor is rengeteg új funkcióval bővíthetjük a böngészőnket. Ebben a hónapban nem egy téma köré csoportosítottam a bemutatandó kiterjesztéseket, hanem az új felfedezetteim közül választottam ki az érdekesebbeket.

TabRenamizer

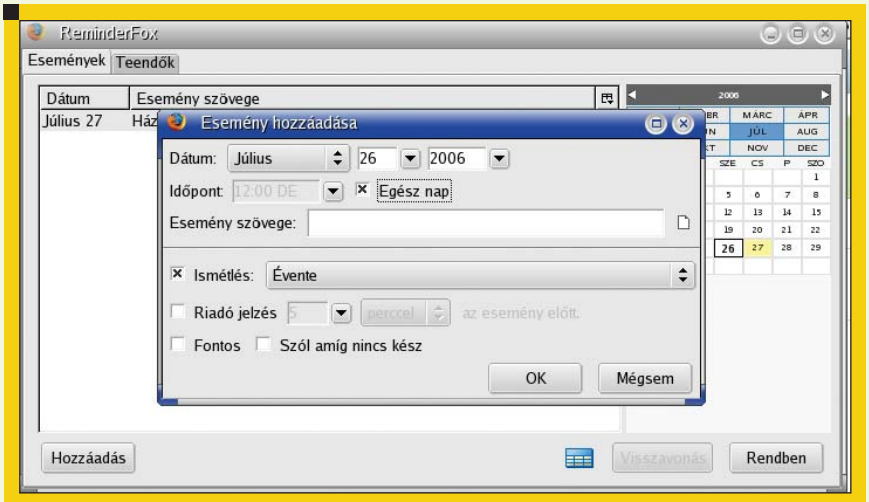
Ha valaki nem szereti azt, hogy a körülötte lévő emberek meglátják, hogy a böngésző különböző fülein milyen oldalakat nyitott meg, annak ajánlom a ***TabRenamizer-t.*** *A* program megváltoztatja az aktuális oldal kivételével az összes fül feliratát valamilyen semleges oldaléra. Ha az ***Eszközök/Rename tabs (auto)*** lehetőség be van jelölve, akkor ezt mindig megteszi és egy gyors `ctrl-t` megnyomása után már el is vannak rejtve az oldalak.

Idokep.hu

Egy magyar kiterjesztéssel folytatnám e havi bemutatómat. E program segítségével megjeleníthetőek a különböző magyarországi városokhoz tartozó időjárási adatok. Először az ***idokep.hu*** feliratra kell kattintani a státuszsorban, ekkor megjelenik a választott városunkhoz tartozó hőmérséklet adat. E névre jobb egérgombbal kattintva megkapunk még pár adatot a térségről valamint itt válthatunk a városok között, és a ***idokep.hu*** oldalhoz tartozó különböző térképeket (hő, szél felhő, légnyomás) is itt hozhatjuk be, melyek folyamatosan frissülnek. Az oldal hivatalos mérési adatokat, és a látogatók által beküldötteket is tartalmaz. Nekem ez a kiterjesztés jobban tetszett, mint a hasonló szolgáltatásokat nyújtó ***Forecastfox,*** mely segítségével nemzetközi adatok is elérhetőek.



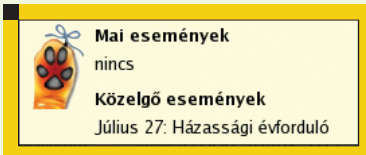
1. ábra Ezen a képen Temporary Inbox menüsora látható



2. ábra Ezen a képen a ReminderFox feladatütemezője látható

Temporary Inbox

Manapság már a legtöbb oldal a különböző szolgáltatási elérését, melyek akár egyébként ingyenes is lehetnek regisztrációhoz köti. Aki nem szeretné megadni valós e-mail címét vagy már unja a ***spam*** áradatot, annak célszerű ebben az esetben egy eldobható e-mail címet megadni. Egy ilyen e-mail cím gyors megszerzését segíti elő a ***Temporary Inbox.***
Ha egy olyan oldalra kerülünk, mely regisztrációt kér, akkor egyszer rákattintunk a ***random e-mail*** gombra, amely megad egy *valami@temporary-inbox.com* címet ezt kell megadnia sajátunk helyet. A megerősítést kérő levél a ***temporaryinbox.com*** oldalára érkezik. a ***check*** gomb lenyomása után megnézhetjük a kapott levelet, ahol a szokásos megerősítő linkre kattintva már aktivizáltuk is regisztrációnkat. A levél hat óra után automatikusan törlődik. Ha kiléptünk a ***Firefox***-ból, de később ellenőrizni akarjuk az ideiglenes postafiókunkat, akkor a ***check*** előtti üres beviteli mezőbe a kukac jel előtti szöveget be kell írni.

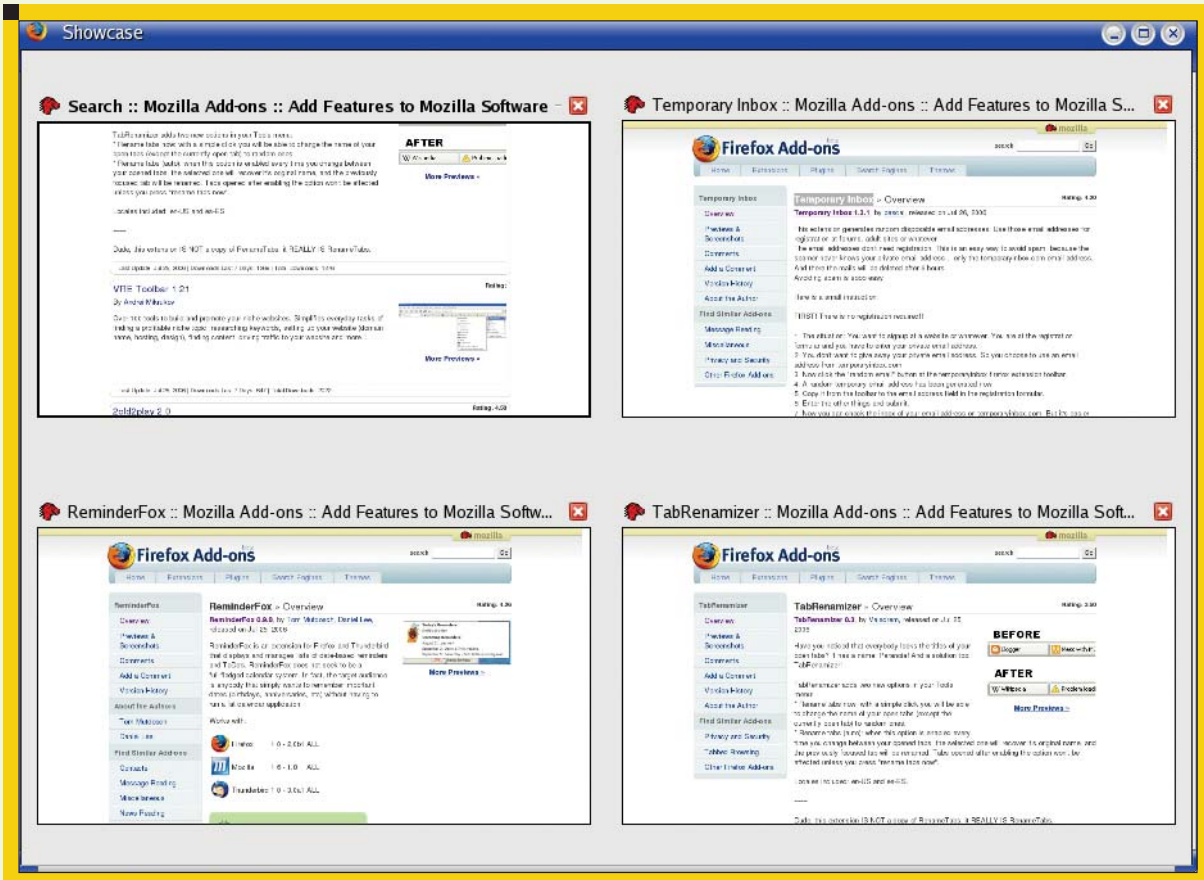


3. ábra Ezen a képen a ReminderFox figyelmeztető ablaka látható

eseményeket. Megadhatjuk, hogy az esemény ismétlődik-e évente, havonta, hetente vagy naponta. Legyen-e riadójelzés vagy sem. Az eseményeket 15 napos intervallumban mutatja, amely megváltoztatható. A böngésző adott napi első elindulásakor figyelmeztető ablak jelenik meg a teendőinkkel.

TrustWatch Search

Az e havi utolsó kiterjesztés, a *TrustWatch Search* biztonságosabbá teheti a böngészést az által, hogy a *Google, a Yahoo* és az *MSN* találati oldalain megjelenő linkeket leellenőrzi, hogy nem szerepelnek-e a *TrustWatch* által



4. ábra Ezen a képen a ReminderFox által megjelenített fülek láthatóak

Hasonló szolgáltatást nyújt a magyar *INFA* (*Internet Fejlesztési Alapítvány*), melyet a ➲ www.no-spam.hu/ címen érhetünk el, itt meg kell adni egy valós címet, ahova továbbítódni fognak a levelink, és ide kapjuk meg a *valami@no-spam.hu* címünket, amit bárhol fel is használhatunk, ez a cím egy általunk beállított idő után, mely nem lehet több 45 napnál, törlődik.

ReminderFox

A kiterjesztés feltelepítése után egy bokréta jelenik meg a státuszsor jobb oldalán. Ha felé visszük az egeret, akkor megjelenek a mai és a közelgő események valamint a mai teendők, ha vannak ilyenek. A bokrétára kattintva megjelenik egy naptár, ahol a különböző napokhoz rendelhetünk

Firefox Showcase

A program a böngésző fülek újfajta elrendezését teszi lehetővé. Az F12 lenyomása után egy új ablakban megjelenek a különböző fülek tartalmai, ezek közül egy kattintással választhatunk, majd pedig ezután visszakerülünk a böngészőhöz. Nem kell a fülek között váltogatni egyből szembetűnik, hogy melyik oldalt kerestük.

Number Madness

A kiterjesztés egy érdekes kezdeményezés, a *tili-toli* játék egy megvalósítása, melyet a *Games/Number Madness* telepítés után megjelenő új menüben indíthatunk el. Használatához nem kel felkeresni egyetlen oldalt sem. A játék lényeg, hogy számokat kell sorba rendezni.

vezetett fekete listán. A link meglátogatása előtt megnézhetjük, hogy más felhasználók esetleg nem találták-e gyanúsnak a kérdéses honlapot és, hogy fűztek a megjegyzést az oldalhoz.

Fekete Imre
(imre.fekete@gmail.com)

Programtervező-matematikusként végeztem a Debreceni Egyetemen. A Linuxtól kezdetben idegenkedtem, de ma már csak azt tudom mondani róla, hogy remek rendszer.

Picasa – Képkezelés felsőfokon

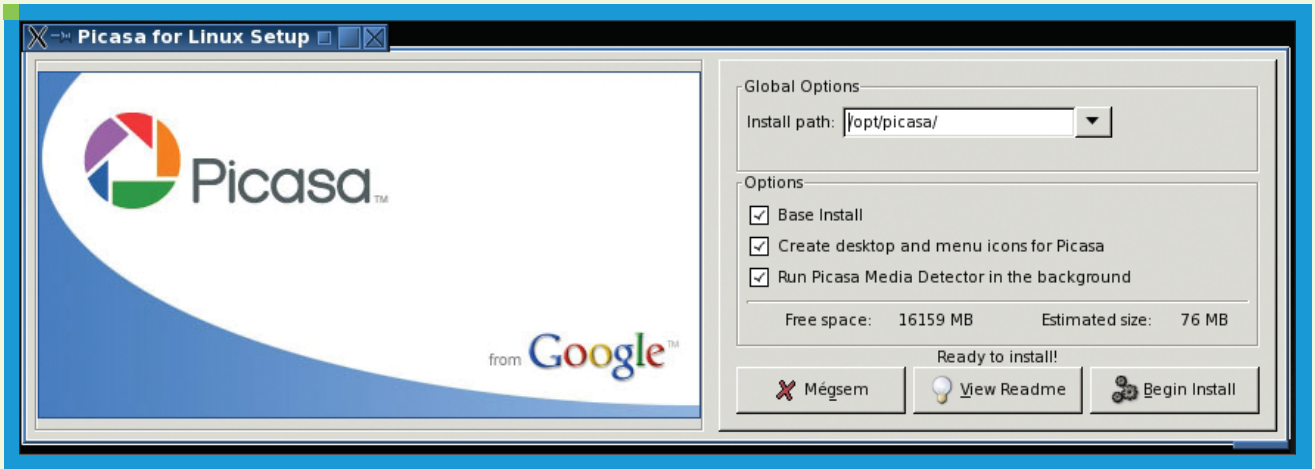
Manapság már az átlagfelhasználó számítógépén is természetes a tízezernyi fotóból álló digitális médiagyűjtemény. A kamerák és az Internet lehetőségei rendíthetetlenül hizlalják a kollekciókat, így ezek rendszerezése és indexelése mára nélkülözhetetlen feladat.

1500 nap, 11000 fotó

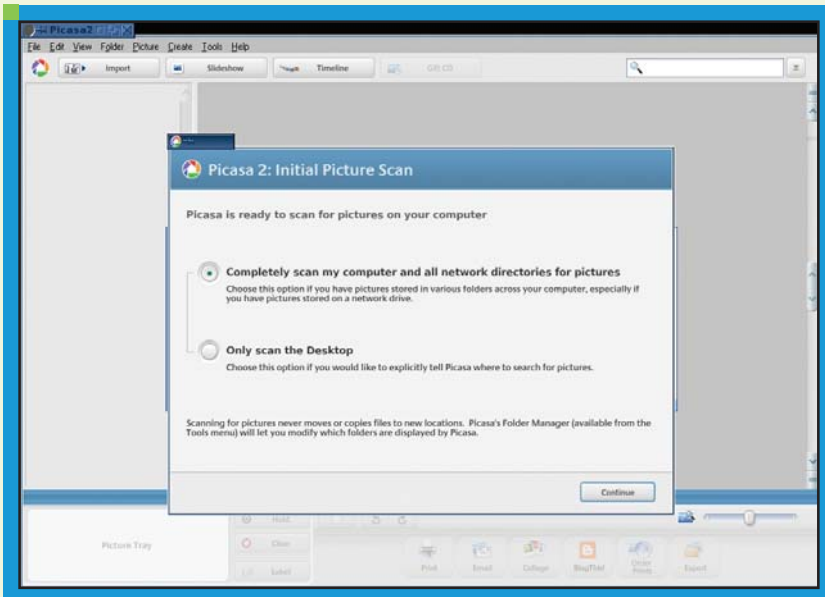
Nemrég kínkeservesen összeszámoltam, mennyi kép gyűlt össze az otthoni számítógépemen az elmúlt négy esztendőben. Íme, az eredmény: nagyjából 11000 fotó, rajz és ábra pihent a személyes mappámban. Mondhatnám magam trehánynak, szórakozottnak, említhetném a szabadidőm kóros hiányát is, de sajnos ez mit sem változtatna azon, hogy a képeim legalább 50 különböző könyvtárban voltak szétszórva, semmitmondó fájlnevekkel. A környezetemben és baráti körömben szintén tetten érhető volt a jelenség... Sőt, némely esetben még akkor is tovább hízott a kollekció, ha eközben már a tulajdonosa is tudta: soha sem lesz türelme órákat tölteni egy fotó későbbi előkeresésével. Mivel a teljes, kézi átszervezés és átnevezés ennyi állománynál több napot vett volna igénybe, én sem erőltettem e hosszadalmas procedúrát. Inkább a *Windows* platform egyik népszerű alkalmazását hívtam életre ***Linuxon***.

Ismét csak a Google!

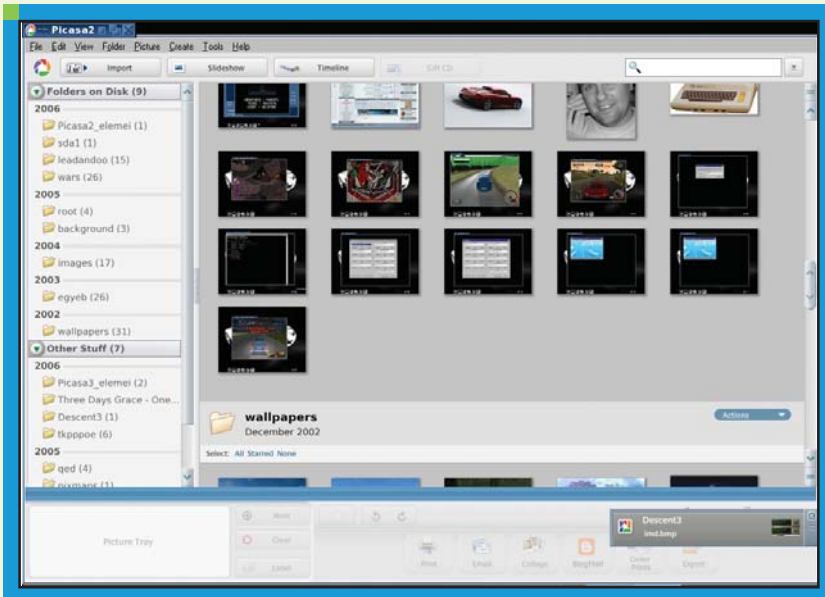
Az utóbbi időben mind biztosabban terjeszkedő *Google* nevéhez fűződik a probléma legnépszerűbb megoldása, a *Picasa*. Az eredetileg *Win32* felületen futó programnak 2006 derekán jelent meg a linuxos verziója, ami által a Szabad Rendszer felhasználói is könnyedén rendszerezhetik képeiket. A dolog apró szépséghibájaként a linuxos verzió nem „klasszikusan" natív megoldás, hanem egy olyan projekt, melyben a windowsos program össze van „drótozva" a *Wine* wrapper binárisával és a *Mozilla* renderelő motorjával. Sebaj, a *Wine* ebben az esetben is kiállja a próbát: a nekünk érdekes verzió ugyanazzal a szerény géppel is beéri, mint *Windows* környezetben, a használhatósága pedig megegyezik az eredeti verzióéval. Függőségként mindössze egy működő *X11*-re, valamint *v2.3* verziójú *glibc* könyvtárra van szüksége rendszerünkben. A projekt honlapját a ➲ *http://picasa.google.com/linux* címen érhetjük el, itt találhatóak a *Picasa* linuxos állományai **.rpm* és **.deb* csomagként, valamint terjesztéstől független **.bin* telepítőbe ágyazva. (A letöltendő fájl minden esetben *24Mbyte* méretű, melyből a *Picasa* valójában *9Mbyte* méretet foglal el, a maradék területen az említett *Wine* és a *Gecko* „osztozik".)
Az előbbi formátumok a megfelelő disztribúciók csomagkezelőjével hívhatóak életre: a *Debian* csomagmenedzserét használó rendszerekben `dpkg -i picasa.verzió.deb` parancs vezet célra, a *Red Hat* megoldását tartalmazó *Linuxon* pedig az `rpm -Uvh`



1. ábra A Picasa *.bin telepítője



2. ábra A merevlemez teljes szkennelésének bevezetése



3. ábra A káosz renddé szelídül

`picasa.verzió.rpm` utasítás használatos. A független, ***.bin*** formájú telepítő ***root*** jogkörrel kiadott `sh picasa.verzió.bin` utasítással állítható üzembe. A telepített program felhasználóként terminálra gépelt `picasa` paranccsal indítható.

A szervezésen túl

Az indított alkalmazás első alkalommal átfutja a háttértárainkat, képek után kutatva, a későbbiekben elsősorban a változásokra figyel. A keresés mindig a háttérben történik, ennek eredménye a bal oldali blokkban látható, ahol az alapértelmezett elrendezés a könyvtárak létrehozásának idején alapul (ez a *„Folder / Sort By”* menüben megváltoztatható). A jobb oldali blokkot a megtalált ábrák bélyegképei építik fel. Természetesen mind a két rész görgethető. Az alsó részen található a képtálca (melyre képeket húzva, később azok rögtön elérhetőek lesznek), valamint a jelölések és transzformációk gombjai (például forgatás), a kézenfekvő szolgáltatási menü (nyomtatás, e-mail küldése, csoportosítás, blog, export, nyomtatás on-line megrendelése), és a ***zoom*** csúszkája. Egy képen jobb egérgombot nyomva előhívható a program helyi menüje, ami lehetőséget nyújt az adott ábra teljes megtekintésére, valamint ennek alapszintű szerkesztésére, címkézésére, elrejtésére, mozgatására és törlésére.

A szerkesztés (ez dupla bal klikkel is aktiválható a bélyegeken) lehetőségei a következők: *„Basic Fixes”* (általános alakítás), *„Tuning”*, *„Effects”* (effektek). Az általános alakítások menülapján a levágás, a „vörös szem” elnyomás, automata kontraszt és színtelítettség, valamint a fénytelítettség állítható be. A tuning szekció a fényerő, a fényes felületek és az árnyékok kezelésére szolgál, lehetőséget adva a színhőmérséklet kézi elmozdítására is. Az effektek által pedig előre beállított vizuális hatásokat lehet alkalmazni a kiszemelt állományon. Fontos, hogy minden egyes változtatás több lépcsőben is visszavonható az *„Undo”* gombbal. A szerkesztés gombjai felett akár paraméterezhető diavetítés is kapcsolható a *„Slideshow”* műveletét kérve, de *„Timeline”* lehetőséggel is élhet az érdeklődő, ahol látványos módon kerül személtetésre az összes médiatár, létrehozási időpont alapján.

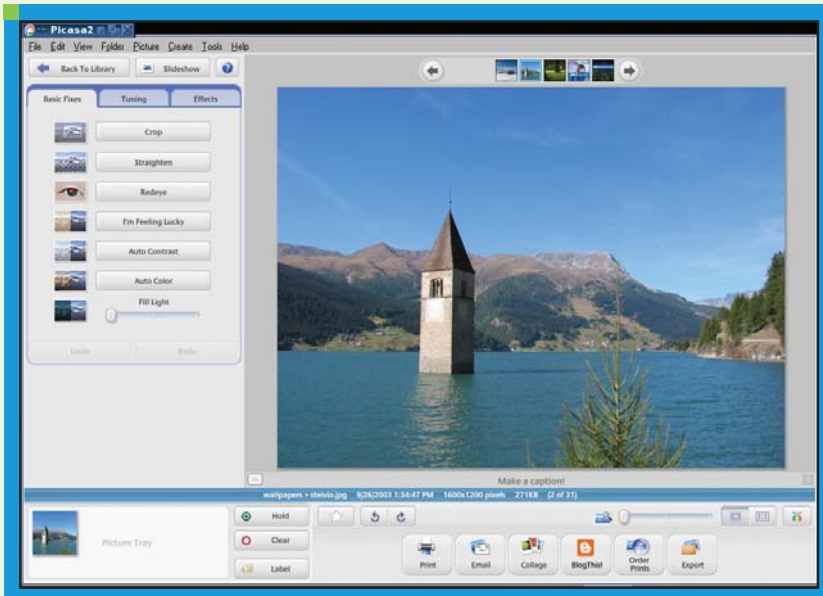
A diavetítés ideje alatt ***.mp3** formátumú hangsávok lejátszása is kérhető a *„Tools / Options / Slideshow”* menüben, ezáltal egy kisebb, de elegáns prezentáció elkészítése is a ***Picasa*** képességlistáját bővíti. Utolsóként, de nem utolsó sorban említendő: a fő panelen található *„Import”* gomb segítségével, a ***gphoto*** csomag által használható digitális kamerákról közvetlenül is felhasználhatunk képeket.

Az „élő példa”

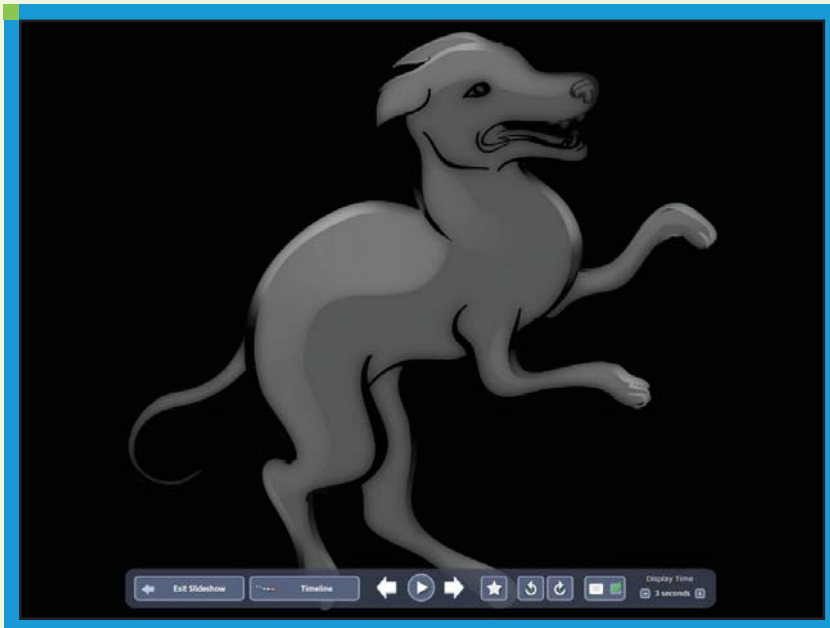
A száraz ismertetés után nézzük az életszerű példát, saját esetemről másolva! Tegyük fel, hogy a személyes mappánkban rengeteg olyan könyvtár van (almappástul), melyek sehogyan sem azonosíthatóak neveik alapján. Bennük számtalan állomány, köztük a számunkra értékes médiák. A fájlok java része digitális fényképezőgépről származik, melyeket anno egyszerűen átmozgattunk a *PC*-nk merevlemezére: a képek neve sajnos jellemzően ***dcim_sorszám.jpg***, mely elnevezés akár többször is előfordul a különböző mappákban.

Nos, több ezer fotó esetén, ilyen körülmények között képes lenne-e bárki megkeresni a „digitális

emlékekeit"? Kötve hiszem...
A *Picasat* használva viszont merőben más a helyzet! A programot indítva, indexelés után a *zoom* paramétert olyan értékre állítom, hogy a bélyegek felismerhetőek legyenek, de a lehető legtöbb miniatűr látszódjék a jobb blokkban. Így a keresés ideje az eredeti idő töredékére csökken.
A fontos találatokon jobb klikket nyomok, majd kiválasztom az *„Add Label"* funkciót, hogy a szükséges képekhez azonos nevű címkét rendeljek. Az így egy csoportba került ábrák helyi menüjén kérem az *„Edit"* lehetőséget. A *„Tuning"* fülre kattintva az automatikára bízom a színhőmérséklet és a fényerő korrigálását (a gyengébb minőségű képekre is gondolva). Ha végeztem, létrehozok egy privát albumot, ahová a feljavított médiákat átmozgatom „húzd és ejtsd" módszerrel.
A privát albumból kiválasztok néhány olyan fotót, melyeken kötegelve kérek változtatást a *„Picture / Batch Edit / Auto Contrast"* menüt választva (így a fotók hasonló kontrasztaránnyal fognak rendelkezni). Ezek után – még mindig a programból – elküldöm a képeket egy célszemélynek, a *gmail* fiókom *account*-ját felhasználva (a fő panelen lévő *„Email"* nyomógombot használva). Hát nem szuper?!
A *Picasa* nélkül órákig keresgélhettem volna „kézzel", a találatokat átnevezhettem volna csoportosan a *krename* csomaggal (a címkézés és mozgatás helyett), majd javíthattam volna az egyedi hibákat (a jóval többre hivatott és emiatt bonyolultabb) *GIMP*-pel, végül a *gmail* honlapot megcélozva, a külön emiatt indított böngészőmből kellett volna elküldenem a válogatott ábrákat. Ez többszörös időigényt, embert próbáló türelemjátékot jelentett volna a *Google* megoldásához képest.



4. ábra A szerkesztés menü



5. ábra A diavetítés egyik fotója

Miért érdemes még?

Az előbb példaként felhozott *„Picture / Batch Edit"* funkció hasznosságát mintázva, a *„Create / Make a Poster"* menüpont szintén a kényelem jegyében született: a nagy felbontású ábráinkat pillanatok alatt képes (poszternyomtatáshoz előkészítve) feldarabolni. A változások, bélyegképek, címkék és információs bejegyzések a személyes mappánk megfelelő */home/$/Picasa_Documents* alkönyvtárában jönnek létre, így mentesülünk az egyébként ismerős, indexelt mappákban sokszor megjelenő *Thumbs.db* állományoktól is.
Reményeim szerint látható, hogy a *Picasat* használva rengeteg időt spórolhatunk meg a digitális fotóink, rajzaink és ábráink között keresgélve, illetve azok között szerkesztve. Mi tagadás, emellett a felhasználó idegeinek is komoly szolgálatot tesz a digitális rendszerező: több ezer kép között igencsak megerőltető és bosszantó dolog lenne órákig keresgélni néhány médiát, nem is beszélve arról, hogy a megtalált, régi fotókon elég sokszor fel kell javítani valamely minőségi paramétert. A lehetőség adott: az ingyen elérhető *Picasa* semmilyen kívánnivalót nem hagy maga után.

Kovács Zsolt (kovi@linuxforum.hu)

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

Állandó vírusvédelem megvalósítása a ClamAV segítségével

Linux alatt vírusvédelem? Minek az nekünk? Linux alatt nincsenek is vírusok! Sajnos ez így nem igaz, mert hiába írják a vírusok 99 százalékát más operációs rendszerre, senki sem garantálhatja azt, hogy ez mindig így marad. Ha pedig Wine-t használunk, akkor az alkalmazások mellé jó eséllyel kaphatunk vírusokat is, melyek így képesek futni Linux alatt.

Amennyiben ez még mindig nem lett volna elég indok arra, hogy ***Linux*** alá vírusirtót telepítsünk, akkor nézzünk néhány olyan példát, amikor szükségünk lehet arra, hogy ***Linux*** alatt végezzünk vírusirtást.
Ha van egy ***NFS*** kiszolgálónk, amit közösen használnak linuxos és windowsos kliensek, akkor a Windows-on futó kliensek védelmének érdekében a kiszolgálón futtathatunk ***ClamAV*** vírusellenőrzést.
A másik példában linuxos kliensünkkel kapcsolódunk ***RDP***-n keresztül egy ***Windows*** terminálhoz, olyan módon, hogy egy adott könyvtárat a terminálon is és a kliensen is látunk. Ha el szeretnénk kerülni azt, hogy a megfertőzzük a kiszolgálót, akkor kliens oldalon (is) ellenőriznünk kell a fájlok tisztaságát.
A ***ClamAV*** kitűnő ingyenes eszközt kínál erre a feladatra, a probléma csak az, hogy nem ismeri a rezidens vírusellenőrzést, tehát a fájlokat nem akkor ellenőrzi le amikor éppen hozzáférünk, hanem ha beállítjuk, akkor bizonyos időközönként periodikusan. Ez egyrészt egy kis biztonsági rést hagy a vírusoknak, hiszen lehetőségük van két ellenőrzés között terjedni, másrészt pedig adott esetben többszörös pluszmunkával terheli a gépünket, ha például minden éjjel leellenőrizzük a teljes megosztott könyvtár tartalmát, mikor napközben csak tíz fájlt változtattunk meg.

Rezidens vírusvédelem Linux alatt

A telepítést az ***Xubuntu 6.06 beta 2*** verzióval végeztem el, de lényeges különbség nincsen a korábbi ***Xubuntu***, ***Ubuntu***, ***Debian*** verziók esetén sem. A ***Dazuko*** modul felelős azért, hogy értesítse a ***Clamavot*** az egyes fájlműveletekről. A ***Dazuko*** modult, így saját kernelünkhöz kell fordítani. Szerencsére ehhez nem kell az egész kernelt újrafordítani, viszont szükségünk lesz a kernel forráskódjára, a kernel fejlécekre és a kernel fordításhoz szükséges csomagokra is, mivel a ***Dazuko*** modul a változóit a kernel forrásából állapítja meg. Adjuk ki a következő parancsokat, hogy rendelkezésünkre álljanak a szükséges csomagok.

```
apt-get install gcc-4.0
➥gcc-4.0-base binutils cpp-4.0
➥libcc1
```

esetleg a ***gcc-4.0*** helyett választhatjuk a ***gcc-3.4***-et is,

```
apt-get install build-essential
➥bin86 kernel-package
➥libqt3-headers libqt3-mt-dev
apt-get install linux-source-
➥2.6.15 linux-headers-2.6.15
```

Lépjünk be a könyvtárba, ahová letöltöttük a rendszermag forrását

```
cd /usr/src
tar -xvjf linux-2.6.15.tar.bz2
```

Hozzunk létre egy szimbolikus linket, mely az aktuális kernel forráskódjára mutat.

```
ln -s /usr/src/linux-2.6.15
➥linux
```

Másoljuk be a jelenlegi kernel konfigurációs állományát a kernel forráskód könyvtárba

```
cp /boot/config-2.6.15
➥/usr/src/linux/.config
```

Amennyiben mégis új kernelt akarunk fordítani, akkor már csak a libncurses5-dev csomagra lesz szükségünk a `make menuconfig` parancs futtatásához. Telepítsük a ***clamav-freshclam***, ***clamav*** és a ***clamav-deamon*** csomagokat, de a ***dazuko*** csomagot ne az ***apt***-n keresztül telepítsük, mert az ott található már nagyon régi, ezért jobban járunk, ha a hivatalos oldalról töltjük le a legfrissebbet.

```
wget http://www.dazuko.org/
➥files/dazuko-source_2.2.1-
➥1_all.deb
```

```
dpkg -i http://www.dazuko.org/
➥files/dazuko-source_2.2.1-
➥1_all.deb
```

Valószínűleg a ***dazuko-source*** csomag függőségi hibába fog ütközni, ezért telepítenünk kell a ***module-assistant*** csomagot is, ekkor már a ***dazuko-source*** is hiba nélkül feltelepül. Szükségünk lesz még a debhelper csomagra is, különben a ***modules-assistant*** képtelen lefordítani a ***dazuko***-t és állandóan hibát fogunk kapni a fordítás során. A ***module-assistant*** kielégítő működéséhez először futtassuk le a `module-assistant prepare` parancsot, majd pedig készítsük el a ***dazuko*** betölthető kernelmodulunkat az

```
m-a a-i dazuko
```

paranccsal.
Amennyiben minden rendben van, akkor végre betölthetjük a ***dazuko*** modult, de előtte a ***capability*** modult ideiglenesen távolítsuk el a rendszerből.

```
rmmod capability
modprobe dazuko
modprobe capability
```

Ahhoz, hogy mindez teljesen automatikus legyen, hozzuk létre a */etc/modules.d/dazuko* fájlt a következő tartalommal:

```
install dazuko modprobe -r
➥capability;\
modprobe -i dazuko; \
modprobe -i capability
```

Ezennel a ***dazuko*** figyeli a fájlműveleteket, de az ***Ubuntuban*** lévő ***ClamAV*** nincsen felkészítve arra, hogy együttműködjön a ***Dazuko*** modullal, ezért kis módosításokkal újra kell fordítanunk azt. A dolog nem annyira bonyolult, mint amennyire annak tűnik.

```
apt-get install fakeroot
➥build-essential
apt-get build-dep clamav
apt-get source clamav
```

Ezután lépjünk be az így létrejött *clamav_xx.x* könyvtárba és a */debian/rules* fájlban a `./configure` kezdetű sorokból vegyük ki a `--disable-clamuko` opciót. Majd pedig fordítsuk újra a *ClamAV*-ot a következő utasítással:

```
dpkg-buildpackage -rfakeroot
➥-uc -us
```

A folyamat végeztével egy könyvtárral feljebb megtaláljuk a telepítéshez szükséges ***deb*** csomagok, melyeket a `dpkg -i *.deb` paranccsal telepíthetünk is. Most meg kell mondanunk a *ClamAV*-nak, hogy milyen esetekben ellenőrizze le a fájlokat, illetve azt is meg lehet mondani, hogy mely könyvtárakban ellenőrizze le őket. A */etc/clamav/clamd.conf* állományhoz írjuk hozzá a következő két sort, mely megmondja, hogy fájl hozzáférés és fájlmegnyitás esetén fusson le az ellenőrzés.

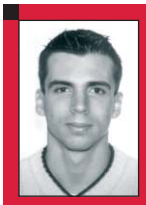
```
ClamukoScanOnAccess
ClamukoScanOnOpen
ClamukoIncludePath /home
```

A *Clamuko*, *Dazuko* együttműködéshez szükséges, hogy a *ClamAV root* jogosultságokkal fusson, így a *User* kezdetű sorban a *clamavot* írjuk át root-ra. Ha ezzel készen vagyunk, akkor már csak a teszt van hátra. Töltsük le az ***eicar*** antivírus tesztállományt (teszt szekvenciát) a ➲ http://www.eicar.org/download/eicar.com címről, majd pedig teszteljük le, hogy engedélyezi-e a rendszer a hozzáférést. Ha igen, akkor valószínűleg valamit nem jól csináltunk, ellenőrizzük le a lépéseket, hogy nem kaptunk-e valahol hibát. Amennyiben nem tudunk hozzáférni az adott fájlhoz (és amúgy a jogosultságunk meglenne hozzá), akkor működik a rezidens vírusvédelem *Linux* rendszerünkön.

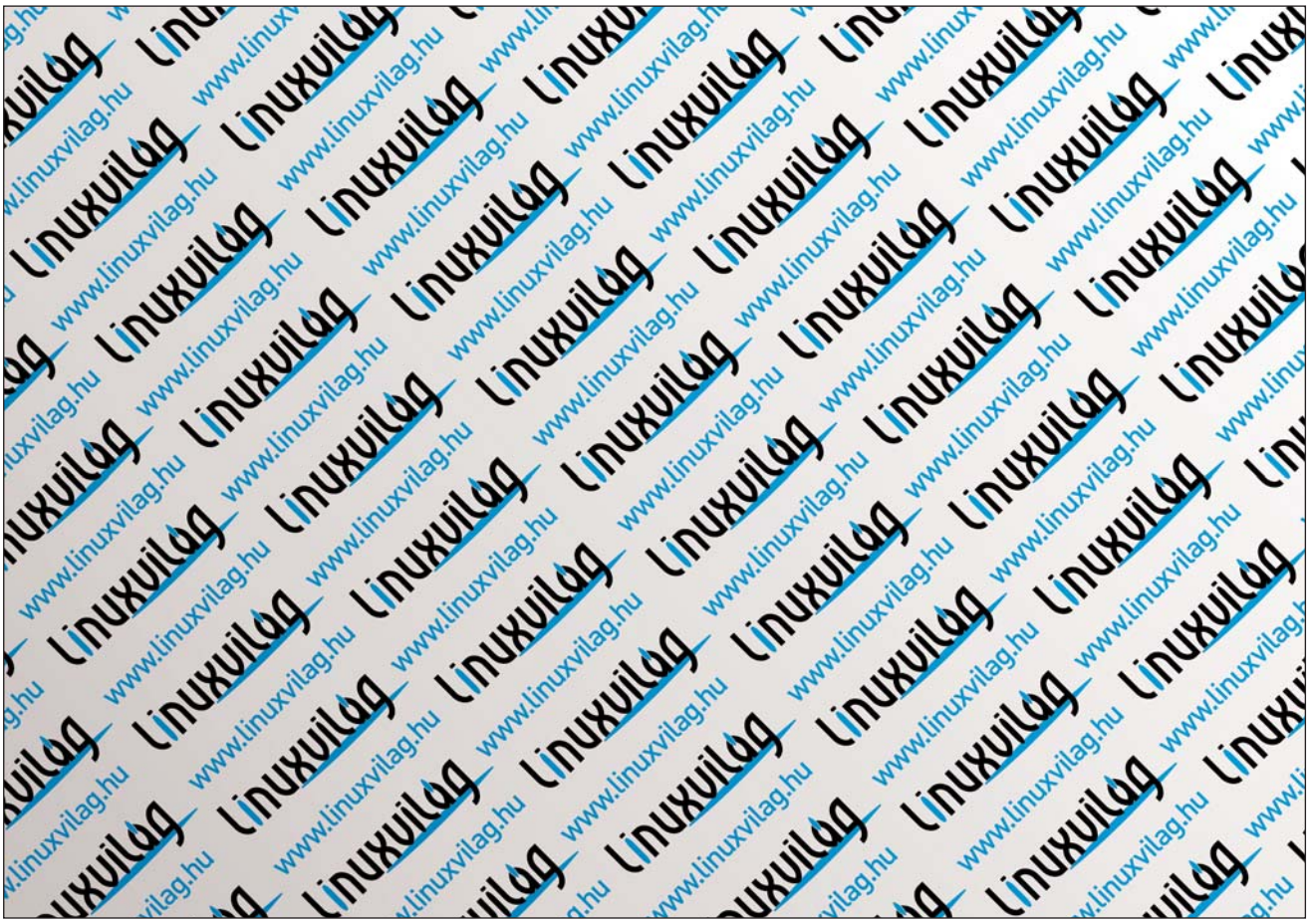
Rezidens vírusvédelem Windows alatt

Aki egyáltalán nem találkozik *Windowsos* számítógépekkel, és nem is érdekli őket annak a lehetősége, hogy hogyan védjük meg a *Windowst* a vírusoktól tisztán szabad szoftverekkel, attól most elbúcsúznék, és köszönöm, hogy eddig velem tartott és végigolvasta a cikket, remélem hasznosnak találta.
Mivel a cikk témája a rezidens vírusellenőrzés *ClamAV* segítségével, ezért most kitérnék a *ClamAV Windows* alatt használható verziójára a *ClamWin*-re. Több dolog miatt is döntöttem amellett, hogy a *Linuxvilág* magazinban kitérjek erre a témára.
Mint magánembert sokan kerestek meg tanácsot kérve, hogy milyen programokat használjanak otthon (megértettem, hogy egyelőre nem szeretnének *Linuxra* váltani) ugyanakkor boldogan használnának biztonságosabb böngészőt, levelezőt, ingyenes és jogtiszta képszerkesztőt, hangvágót stb. Ezért mindenkinek az adott feladatra megfelelő szabad szoftvert ajánlom, esetleg telepítem fel a gépére. Pontosan ezért konzekvensen mindenkinek a *ClamAV*-ot ajánlottam mint vírusirtót, csak sajnos (eddig) hiányzott belőle a memória rezidens védelem. Mint rendszergazda munkám során találkozom *Windowsos* gépekkel is, és mindent megteszek azért, hogy a lehető legnagyobb biztonságot élvezzék ők is, de ezidáig nem tudtam megoldani azt, hogy a *ClamAV* csak azután engedje futni az alkalmazásokat, miután meggyőződött azok vírusmentességéről. Az esti vírusellenőrzés pedig lehet, hogy már csak eső után köpönyeg. Mint egyszerű user valahogy szerettem volna megoldani a családi pc védelmét is, szem előtt tartva azt, hogy az egyszerűség, a gyorsaság és az ingyenesség testesüljön meg a víruskereső szoftverben is, ha már kell otthonra egy *Windows* is.
Hogy a kép teljesebb legyen a *ClamAV*-nak létezik egy portja *Mac OS X* alá is. Ennek neve *ClamXav* mely egy letisztult, a *Mac OS X* kinézetéhez illeszkedő felületet nyújt a *ClamAV*-nak. Sajnos ha jól tudom, akkor egyelőre *Mac OS X* alatt a *ClamAV*-ot nem lehet rávenni a memória rezidens védelemre.
Szerezzük be a szükséges programokat a vírusellenőrzéshez, ha még nem tettük

volna meg. Szükségünk lesz tehát a *ClamWin*-re, melyet a ➲ *http://www.clamwin.com/* oldalról szerezhetünk be, illetve szükségünk lesz a *Winpooch* nevű okos kis alkalmazásra, melyet a ➲ http://winpooch.free.fr/home/index.php oldalon találhatunk meg.
A *Winpooch* egy remek szoftver mely a rosszindulatú szoftverek és trójai programok elleni védekezést segíti oly módon, hogy valós időben monitorozza az egyes folyamatok aktivitását a rendszerünkben. Segítségével engedélyezhetjük bizonyos programoknak, hogy adott könyvtárba írjanak, ugyanakkor egyéb könyvtárba megtilthatjuk az írás jogát. Ugyanígy megtehetjük azt is, hogy bizonyos programokat eltiltunk attól, hogy a regisztrációs adatbázisba írjanak, persze ekkor számolni kell azzal, hogy a program esetleg hibásan fog működni.
A program használatáról egy külön cikket is lehetne írni, de mi most csak a vírusellenőrzésre fogunk koncentrálni, azon belül is három különböző szabálytípust mutatok be.
Miután telepítettük a *Winpooch*-ot és a *ClamWin*-t is, láthatjuk, hogy a rendszer szinte minden egyes megmozdulására a *Winpooch* megkérdi tőlünk, hogy az adott műveletet engedélyezzük-e. Ha még nem zártuk ki magunkat a rendszerből :-), akkor mentsük el az alapértelmezett konfigurációs fájlt, és egy teljesen újat hozzunk létre helyette, mely kezdetben üres.
A program helyére írjunk egyszerűen csak egy * karaktert, mert a szabályunkat minden programra akarjuk érvényesíteni. A + ikonnal tudunk új szabályokat hozzáadni a jelenlegi konfigurációhoz.
Az első esetben ellenőrizzünk le minden *.doc* kiterjesztésű fájlt amikor azokat megnyitjuk. A szabályunk tehát a következő lesz, a *Reason* legördülő menüből válasszuk ki a *File::Read* opciót, a *Type* legördülő menüben válasszuk ki a *Wildcards* menüpontot, majd pedig adjuk meg a *Value* mezőnek a **.doc* kiterjesztést. A *Virus Scan* opciót feltétlenül válasszuk ki, majd mentsük el az első szabályunkat.
A második esetben egy adott könyvtár összes fájlját fogjuk leellenőrizni megnyitás előtt, például minden letöltött fájlt a *d:\downloads* könyvtárba töltünk le. A második szabály akkor annyiban fog különbözni az elsőtől, hogy a *Type* legördülő menüből a *Path with Wildcards* menőpontot válasszuk ki, és a *Value* mezőbe pedig írjuk be a *d:\downloads** karaktersort.
A harmadik szabály lesz talán a legértékesebb számunkra, ez fogja leellenőrizni a fájlokat még mielőtt azok futnának. A *Reason* legördülő menüből most válasszuk ki a *Sys::Execute* opciót, a *Type* legyen *Any Value*, értéke pedig * legyen. Az előzőekhez hasonlóan most is engedélyezzük a vírusellenőrzést.
Mindenkinek jó vírusirtást, de még inkább vírusmentességet kívánok.



Horváth Ernő
ernohorvath@gmail.com
24 éves, műszaki informatikus. Három évvel ezelőtt ismerkedett meg komolyabban a Linux rendszerekkel és emellett érdeklődik még a robotika és a biztonságtechnika iránt is. Ha lenne szabadideje sokat kirándulna, biciklizne és filmeket nézne.



Bluetooth – Vezetékek nélkül, szabadon

Manapság sokféle vezetékmentes szabványt használnak (IrDA, Bluetooth, Wifi) és jó pár vár (WiMax) az elterjedésre. Mostanság a Bluetooth-t egyre többen kezdik felfedezni, így én is. Hogy másoknak ne kelljen az ösvényt újra és újra kitaposni, megosztom a Linux és Bluetooth összeházasításával kapcsolatos tapasztalataimat.

A múlt

Mielőtt vadul belevágnék a közepébe, szeretnék rávilágítani, miért pont ***Bluetooth*** a szabvány neve? A tizedik században élt egy dán király: ***Harald Blatand*** (angolul: ***Harold Bluetooth***). Ő volt az aki egyesítette a hadviselő feleket a mai ***Norvégia***, ***Svédország*** és ***Dánia*** területén. Különböző – kultúrájú és gazdaságú – népek békés kommunikációja. A ***Bluetooth*** szabvány is valami ilyet csinál. Összefog több különböző eszközt (mobiltelefon, számítógép, headset, GPS vevő, stb.)

A szabványról

Hogy megértsük a ***Bluetooth*** működését, nem árt, ha ismerünk pár adatot. A szabvány a 2,45 GHz körüli sávokat használja. Ez azt jelenti a gyakorlatban, hogy a falakat nem nagyon szereti, illetve ***wifi***-vel „szennyezettebb" területen gyengülhet a vétel, de mobiltelefon és headset közé ideálisabb, mint a vezeték. Létezik belőle ***Class 1***-es (1 milliwatt adóteljesítmény), ***Class 2***-es (10 milliwatt adóteljesítmény), valamint ***Class 3***-as (100 milliwatt adóteljesítmény). A ***Class 1*** és ***2*** körülbelül 10 méteres hatósugarat biztosít nyílt terepen, míg a Class 3-as akár 100 métert, de ez utóbbi elég ritka.
A ***Bluetooth***-t nem a ***LAN (Local Area Network)*** felváltására tervezték, hiszen csak 1 megabites sávszélességet tud nyújtani (kb. 700 kbit-es letöltés aszimmetrikus módban és körülbelül 500 kbit-es szimmetrikus módban), illetve egy master (mester) eszközhöz maximum 7 slave (szolga) eszköz tud csatlakozni. A ***Bluetooth*** sokkal inkább ***PAN (Personal Area Network)***, vagyis személyi hálózat.

Párosan szép az élet

A ***Bluetooth*** eszközök csak akkor tudnak egymással kommunikálni, ha úgymond párosítják őket. Ez olyan, mint amikor a való életben két üzleti partner névjegykártyát cserél.
A ***Bluetooth***-nál kölcsönösen azonosítót cserélnek, így biztosítva, hogy két kijelöl eszköz csak akkor ismerje egymást, ha mindkettő tulajdonosa jóváhagyta azt.
Jó tudni, hogy sok eszköz – például a ***Bluetooth headset***-ek nagy része – képes több mester eszközt kezelni, így nem kell folyton törölni a már meglévő párosítást, valahányszor változik az eszköz (számítógép, mobiltelefon, ***PDA***). Viszont ha egy ***headset***-et – vagy valami egy másik eszközt – már használ valami, úgy az a többi eszköz számára láthatatlan, még akkor is, ha egyébként párosítva vannak.

A kékfogú pingvin

Linux alatt már elég régóta támogatottak a ***Bluetooth*** eszközök, de én személy szerint igazán csak a 2.6.13.4-es kerneltől kezdve próbálgattam.
A linuxos ***Bluetooth*** tapasztalataim szerint az alábbiakra képes, később be fogom mutatni, hogyan:

- Internetmegosztás (laptophoz, ***PDA***-hoz ideális)
- Fájlátvitel (például mobiltelefonról vagy mobiltelefonra)
- Hangkártya (ha nincs hangkártya a gépben)
- Betárcsázós internet (végszükség esetére)

Persze többre is képes lehet, de ha átlátjuk az alapvető dolgokat, akkor a fenti négy esetből könnyen kitalálhatjuk egy kis utánajárással a dolgokat.

Forogjon a kernel

A ***Linux*** általában kezeli a mai ***USB***-s ***Bluetooth*** kulcsokat. Jelen tesztben egy Class 1-es ***X-Micro*** típusút használtam. Ahhoz, hogy működjön, nyilván az alaplapnak megfelelő ***USB*** vezérlőt le kell fordítani, legyen az ***OHCI***, ***UHCI*** vagy ***EHCI***. Ha van működő ***USB*** alrendszerünk, akkor egy `lsusb` paranccsal megtudhatjuk, milyen eszközünk van konkrétan. Nálam például egy ilyen:

```
...
Bus 002 Device 002: ID
➥0a12:0001 Cambridge Silicon
➥Radio, Ltd Bluetooth Dongle
➥(HCI mode)
...
```

Általánosságban elmondható azonban, hogy az alábbi részek szükségesek a működő ***Bluetooth***-hoz:

```
CONFIG_BT=m
CONFIG_BT_L2CAP=m
```


```
CONFIG_BT_SCO=m
CONFIG_BT_RFCOMM=m
CONFIG_BT_RFCOMM_TTY=y
CONFIG_BT_BNEP=m
CONFIG_BT_HIDP=m
```

A fentiekhez rövid magyarázat: az ***RFCOMM*** a betárcsázós internethez és a fájlok áttöltéséhez szükséges, míg a ***BNEP*** a helyi hálózatért felelős. Az eszközmeghajtóknál pedig érdemes csak azt lefordítani, amivel rendelkezünk.
A 2.6-os kernel esetén, amennyiben a `make menuconfig` segítségével konfiguráljuk a kernelt, úgy a ***Bluetooth*** a ***Networking*** menüben található. Innen érdemes mindent modulba fordítani, hátha kell később. Persze vannak dolgok, amiket bele kell még fordítani, például a betárcsázós internethez a ppp támogatás, de az nem létfontosságú.
Debian alatt a `bluez-utils` csomagot szükséges feltelepíteni annak minden függőségével együtt. Amennyiben a használt disztribúció nem rendelkezik ezzel a csomaggal, úgy azt a ***Bluez*** projekt honlapjáról tölthetjük le.

Internet továbbosztása

A ***Bluetooth*** például kiválóan alkalmas arra, hogy az asztali számítógépünk segítségével például a ***PDA***-ra (vagy akár laptopra) továbbítsuk az internetet. Ehhez nem kell mást tennünk, mint telepíteni a `bluez-pin`, `bluez-utils`, `libbluetooth1` csomagokat.
A csomagok feltelepítése után a `/etc/bluetooth` könyvtárban találunk pár fájlt. Ilyen például a `hcid.conf` és a `pin` fájl. A minimális `hcid.conf` valahogy így néz ki:

```
# HCI daemon configuration
# file.
# HCId options
options {
        autoinit yes;
        security auto;
        pairing multi;
        pin_helper
        ➥/usr/bin/bluez-pin;
}
# Default settings for HCI
# devices
device {
        name "%h-%d";
        class 0x3e0100;
        iscan enable; pscan
        ➥enable;
        lm accept;
        lp
rswitch,hold,sniff,park;
}
```

A `pin` fájl pedig az adott ***Bluetooth*** eszköz (például ***USB kulcs***) általunk választott ***PIN kódját*** tartalmazza. Ezt fogja a másik eszköz kérni párosításkor.
Nos a ***PAN*** (személyes hálózat) felépítéséhez szükségünk van a `pand` kiszolgálóra, amely `bluez-utils` csomagban található.

```
pand --listen --role NAP --
➥master -autozap
pand --connect
➥xx:xx:xx:xx:xx:xx --service
➥NAP -autozap
```

Az `xx`-ek helyére a ***Bluetooth*** adapterünk ***MAC*** címét várja a program, ami vagy szerepel az eszköz csomagolásán, vagy pedig a `hciconfig` paranccsal kérdezhetjük le. Most a másik eszközzel például a ***PDA***-val csatlakozzunk a ***Bluetooth*** adapterhez.
Ha ez sikerült, akkor utána már a `bnep0` hálózati csatoló is megjelenik a hálózati kártyák listájában (`ifconfig -a`), így címet is tudunk hozzárendelni, legyen mondjuk:

```
ifconfig bnep0 192.168.2.1
➥netmask 255.255.255.0 up
```

Fontos, hogy ne abból az alhálózatból válasszunk címet, amiben a számítógépünk hálózati csatolója van.
Ha ez megvan, a ***PDA***-n is vegyünk fel egy hasonló ***IP*** címet (ugyanabból az alhálózatból), legyen ez most a 192.168.2.2. Ezután már pingelhető a két eszköz oda-vissza. Persze még nem lehet teljes az öröm, hiszen a ***PDA*** még nem lát a nagyvilágba. Ehhez szükséges, hogy az ***iptables*** megfelelően legyen benne a kernelben – tudjon ***NAT***-olni (***Network Address Translation***) illetve támogassa a ***masquerade***-ot és a ***forward***-ot. A következő parancsokkal tudunk kapcsolatot kialakítani:

```
iptables -t nat -A POSTROUTING
➥-s 192.168.2.0/24 -o eth0
➥-j MASQUERADE
echo 1 > /proc/sys/net/
➥ipv4/ip_forward
```

A két sorhoz egy kis magyarázat: az első sorban megmondjuk, hogy a 192.168.2.0-ás alhálózatból érkező kéréseket továbbítsa az ***eth0*** hálózati csatolóra – ezen jön be nálam az internet. A második sorban pedig az alapvető csomagtovábbítást engedélyezem a két interfész között.

Állományok átvitele

Az ***OBEX (OBject EXchange)*** szabvány lehetővé teszi, hogy akár ***Bluetooth***, akár infra kapcsolat segítségével állományokat töltsünk fel a telefonunkra, vagy töltsünk le onnan. A művelethez csupán pár apró program szükséges, amely ***Debian*** alatt az `obexftp` csomagban található, de letölthető a projekt honlapjáról is.
A fájl átvitelhez az első lépésként ki kell deríteni a cél eszköz ***MAC*** címét. Ehhez a `hcitool` parancs `scan` opcióját használjuk.

```
debian:~# hcitool scan
Scanning ...
        xx:xx:xx:xx:xx:xx
        ➥Nokia 6021
debian:~#
```

Most, hogy már ismerjük a *MAC* címet, a következő lépés csatlakozni az eszközhöz. Ezt az `rfcomm` parancs `bind` opciójával tehetjük meg az alábbi módon:

```
rfcomm bind /dev/rfcomm0
➥xx:xx:xx:xx:xx:xx
```

Amennyiben nem rendelkezünk a `/dev/rfcomm0` eszközökkel, hozzuk létre az alábbi módon:

```
mknod /dev/rfcomm0 c 216 0
```

Amennyiben valamiért több eszközre lenne szükség, úgy a nullák helyére beírhatunk egytől kilencig számokat. Amennyiben sikerült a kapcsolódás, úgy kiadhatjuk az alábbi parancsot. Ha esetleg kifelejtjük a *MAC* címet, akkor mindig az első és leggyorsabban reagáló *Bluetooth* eszközre próbálja meg végrehajtani a parancsot. Az alábbi parancs például kilistázza az eszköz főkönyvtárát. (Ezt általában *XML* formátumban kapjuk meg.)

```
obexftp -b xx:xx:xx:xx:xx:xx -l
```

A *SajatKepek* könyvtár tartalmát például a `-l` opció paraméterezésével érhetjük el.

```
obexftp -b xx:xx:xx:xx:xx:xx -l
➥/SajatKepek
```

Állományt feltölteni a `-p`, letölteni a `-g` opcióval lehet. Fontos megjegyezni a *műveleti sorrend*et. Az alábbi példában a képet a *SajatKepek* mappába mentem a telefonon, de az opciók fordított – tehát először a `-p` és utána a `-c` – sorrendje nem ezt eredményezné.

```
obexftp -b xx:xx:xx:xx:xx:xx -c
➥/SajatKepek -p goodman02.gif
```

Hasznos opciók még: `-k` a törléshez, `-G` letöltés majd törlés, `-C` könyvtár váltás és létrehozás, amennyiben nem létezik.
Ha le szeretnénk zárni a *Bluetooth* kapcsolatot, adjuk ki a `rfcomm release /dev/rfcomm0` parancsot. Fontos megjegyzés, hogy nem minden telefon támogatja akármilyen állomány átvitelét, illetve az általam kipróbált telefonok egyikéről sem tudtam *JAVA (J2ME)* alkalmazást letölteni, vagy a telefonra feltölteni. Erre ezután is a *GPRS* vagy *CSD* (esetleg *HSCSD*) áll rendelkezésünkre.

Kék a zene...

Megfelelő *Bluetooth headset*-tel és *Bluetooth* adapterrel megvalósítható az is, hogy az *XMMS* a *Bluetooth headset*-ben szólaljon meg. Nagy minőséget azonban ne várjunk, mert elsősorban nem *Mozart* vagy *Beethoven* átvitelére tervezték ezeket az eszközöket, hanem normál beszédre.
Nem is akarom tovább fokozni a kedélyeket. Ahhoz, hogy tudjuk ilyen célra használni a számítógépünket, szükség lesz egy működő *ALSA* rendszerre, valamint az alábbi programokra, csomagokra: `automake` (minimum 1.7-es verzió), `libbluetooth-dev`, `libasound2-dev` (vagy másik nevén: `alsa-devel`). Kernelből sem mindegy, melyiket használjuk, ugyanis a sikerhez minimum ***2.6.11.7***-es-re lesz szükségünk.
Régebbi kernellel, vagy különálló *ALSA*-val nem működik a dolog.
A fordítás általában az alábbi parancsokkal megy végbe:

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

Amennyiben ***duplex*** (kétirányú) módon szeretnénk használni a ***headset***-et, úgy a kernelfordításakor az `emu10k1`-et (***Sound Blaster Live***) is bele kell fordítani.
Van még pár parancs, amit be kell gépelnünk. Később ezek esetleg ***bash scriptbe*** írhatóak, hogy egyszerűbb legyen az élet. Elsőként illesszük be a modult:

```
modprobe snd_bt_sco
```

Mielőtt a következő lépést végrehajtanánk, a modul fejlesztői azt javasolják, állítsuk le az `esound` szolgáltatást, amennyiben fut:

```
esdctl stop
```

Innen már szabad az út, készítsük elő hangátvitelre a ***Bluetooth adaptert*** a következő módon:

```
hciconfig hci0 voice 0x0060
```

Indítsuk el a szükséges szolgáltatást (az `xx`-ek helyére természetesen írjuk be a ***headset MAC*** címét, amit a már korábban ismertetett `hcitool` segítségével kérdezhetünk le)

```
btsco xx:xx:xx:xx:xx:xx
```

Innen kezdve például az *XMMS*-ben már látszik a ***Bluetooth headset***, mint hangkártya. (***Opciók -> Beállítások -> Kimenet: ALSA -> Beállítás***)
Amíg szeretnénk használni a ***headset***-et, addig ne lépjünk ki a `btsco`-ból. A `modprobe` és a `hciconfig` futtatásához természetesen rendszergazdai jogosultság kell, míg a `btsco` normál felhasználóként is futtatható. Ezt elkerülendő a `snd_bt_sco` modult érdemes beírni a `/etc/modules` fájlba, illetve

Egy tipikus kapcsolódás

```
debian:~#  wvdial gprs
--> WvDial: Internet dialer
➥version 1.54.0
--> Cannot get information
➥for serial port.
--> Initializing modem.
--> Sending: ATZ
ATZ
OK
--> Modem initialized.
--> Sending: ATDT*99***1#
--> Waiting for carrier.
ATDT*99***1#
CONNECT
~[7f]}#@!}!} }
➥}2}#}$@#}!}$}%\}"}&} }*} }
➥g}%~
--> Carrier detected.
➥Waiting for prompt.
~[7f]}#@!}!} }
➥}2}#}$@#}!}$}%\}"}&} }*} }
➥g}%~
--> PPP negotiation
➥detected.
--> Starting pppd at Tue Mar
➥14 18:56:09 2006
--> pid of pppd: 14656
--> Using interface
➥ppp0
--> local  IP address
➥84.224.17.98
--> remote IP address
➥10.6.6.6
--> primary   DNS address
➥193.225.153.17
Caught signal #2!  Attempting
➥to exit gracefully...
--> Terminating on signal 15
--> Connect time 3.4
➥minutes.
--> Disconnecting at Tue Mar
➥14 18:59:32 2006
debian:~#
```

a `hciconfig`-ot ***setuid*** jogosultsággal ellátni – bár ez utóbbi ***nem javallott*** biztonsági okok miatt.

Elnézést, kapcsolná az Internetet?

Manapság már nem nagy igény, hogy mobiltelefonról tudjunk internetezni, sőt az is megoldható, hogy a telefon csak a ***modem*** szerepét játssza és a laptopon is internetezhetünk.

A jó hír: a mobilszolgáltatók már jó ideje lehetővé teszik azt is, hogy ne az eltöltött idő arányában, hanem a ***le/feltöltött bájtok után fizessen*** az ügyfél (***GPRS – General Packet Radio Service***). Persze a tarifák még messze állnak a Kánaántól, de már nem elérhetetlenek.

Először is szükséges egy jól konfigurált ***kernel***, ami tartalmaz ***Bluetooth*** soros port (***tty***) és ***ppp*** (***point-to-point protocol***) támogatást. Ha ezzel megvagyunk, fel kell telepíteni a ***ppp*** démont (***Debianban*** `ppp` a csomag neve), valamint egy tárcsázót (én a `wvdial`-t használom).

A `wvdial` beállítása egyszerű.

A ***/etc/wvdial.conf*** fájlnak a következőt kell tartalmaznia:

```
[Dialer gprs]
Phone = *99#
// vagy Phone = *99***1#
// ez az opció telefonfüggő
Username = ''
Password = ''
New PPPD = yes
Modem = /dev/rfcomm0
```

A kapcsolat felépítése az ismertetett konfigurációs állomány után már egyszerűnek mondható. Adjuk ki az `rfcomm bind /dev/rfcomm0 xx:xx:xx:xx:xx:xx` parancsot. Amennyiben az `rfcomm0`-át már használjuk, úgy az `rfcomm1`-et és így tovább, ekkor azonban ne felejtsük el átírni a ***/etc/wvdial.conf*** fájlban. Ezután adjuk a `wvdial gprs` parancsot. Ha mindent jól csináltunk, akkor megkezdődik a tárcsázás. A folyamatot bizonyos telefonoknál a telefon kijelzőjén is nyomon követhetjük. Ha végeztünk, a kapcsolat befejezéséhez nyomjunk a ***wvdial*** ablakában egy ***CTRL+C*** kombinációt. Ha valami hiba lépne fel a tárcsázás során, úgy az a ***/var/log/messages*** fájlba kerül.

Amennyiben már bootoláskor szeretnénk csatlakozni a telefonunkhoz, úgy a ***/etc/bluetooth/rfcomm.conf*** fájlt hozzuk létre az alábbi tartalommal – az `xx`-ek helyére a mobiltelefonunk *MAC címét* kell írni:

```
rfcomm0 {
  bind yes;
#       # Bluetooth address of
          ➥the device
          device
➥xx:xx:xx:xx:xx:xx;
#       # RFCOMM channel for
          ➥the connection
          channel 1;
#       # Description of the
          ➥connection
          comment "mobiltelefon";
}
```

Bluetooth 2.0

Az új változat már köztünk van. ***Háromszor gyorsabb***, mint elődje, ***alacsonyabb*** az ***energiafogyasztása*** is – ez utóbbi mobil alkalmazásoknál jöhet jól –, továbbá a mostani rendszerekkel ***kompatibilis***. A nagy kérdés, hogy van-e rá igény, hiszen itt a ***wifi*** és küszöbön a ***wireless USB*** is. Mindezek sokkal nagyobb sávszélességet ígérnek, mint az új ***Bluetooth***. Szerintem azonban még sok kernelverziónak kell kijönnie, mire akár a ***wifi***, akár a ***wireless USB*** leváltja a ***Bluetooth***-t a jelenlegi alkalmazásokban, tehát 2-3 évre mindenképp ajánlott a ***Bluetooth*** beszerzése, amennyiben nem a sebességet hajszoljuk.

Medve Zoltán
(e-medve@e-medve.hu)

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

KAPCSOLÓDÓ CÍMEK

Linux kernel
➲ ftp://ftp.kernel.org/pub/linux/kernel/v2.6

BlueZ projekt honlapja
➲ http://www.bluez.org/download.html

Bluetooth ALSA honlap
➲ http://bluetooth-alsa.sourceforge.net/

ObexFtp honlap
➲ http://triq.net/obexftp.html


Linuxvilág
www.linuxvilag.hu

LDAP azonosítás a gyakorlatban – egy esettanulmány

A kliensek beállítása (2. rész)

Ha az előző részben leírtak alapján sikerült telepíteni a kiszolgálót, már csak a kliensek beállítása van hátra. Ebben a részben mellesleg valamennyi olyan kiszolgáló beállítását is bemutatom, amelyek rendszerünkben LDAP azonosítást használnak majd.

Névfeloldó szolgáltatás beállítása

Ezt minden gépen el kell végezni, amelyiken az *LDAP* adatbázisból azonosít. Először is telepítettem a ***libnss-ldap*** csomagot:

```
apt-get install libnss-ldap
```

A telepítéskor megkérdezi a következőket és a válaszoknak megfelelően létrehozza az */etc/libnss-ldap.conf* fájlt:

- Mi az *IP* címe az *LDAP* kiszolgálót futtató gépnek
- *LDAP search base*: adatbázis alap (*dc=cegnev,dc=hu*)
- *LDAP* kiszolgáló verziója
- Igényel bejelentkezést az *LDAP* adatbázis (nem)
- Csak a tulajdonos számára legyen-e írható/olvasható a beállító fájl (nem)

Módosítottam az */etc/nsswitch.conf* fájlt a következőképpen:

```
passwd:         files ldap
group:          files ldap
shadow:         files ldap
hosts:          files dns
networks:       files
protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       files
```

Csak a *Linux* újraindítása után fog működni, de éppen ezért előbb célszerű a *PAM*-ot is beállítani, mert az is újraindítást igényel. A beállítást úgy lehet ellenőriztem, hogy létrehoztam egy ideiglenes fájlt, egy olyan felhasználó tulajdonába adtam amelyik csak az *LDAP* adatbázisban létezik, majd ellenőriztem. Ha például a ***nagypeter*** felhasználó már létezik az adatbázisban:

```
touch /tmp/fajlnev
chown nagypeter /tmp/fajlnev
ls -l /tmp/fajlnev
```

A PAM rendszer beállítása

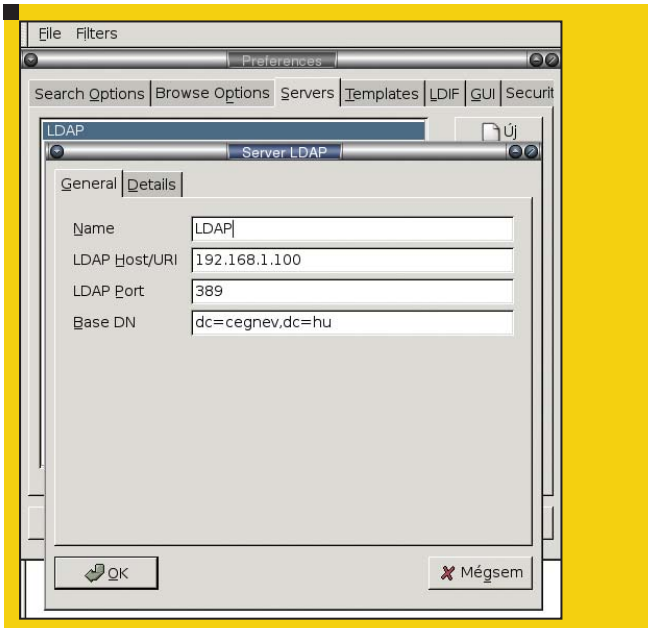
Ezt is minden gépen el kell végezni, melyen az *LDAP* adatbázisból szeretnénk azonosítani. Először telepítettem a *PAM LDAP* modulját:

```
apt-get install libpam-ldap
```

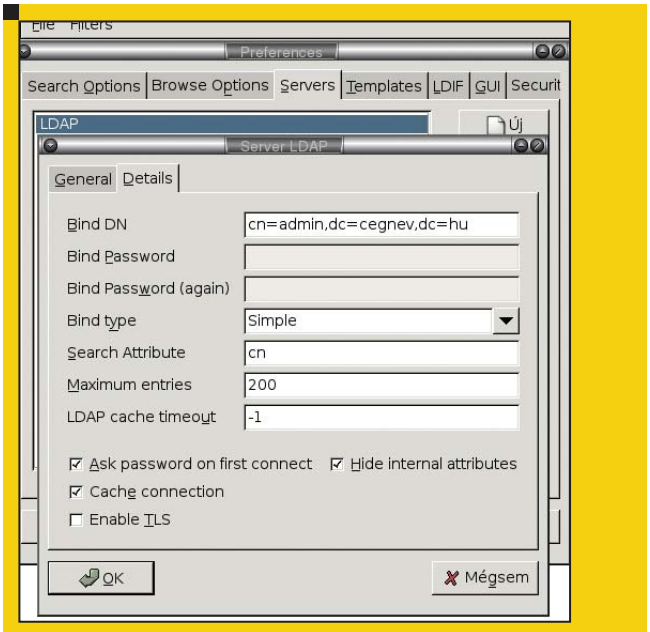
Ha a telepítés a fenti módon parancssorral történik, akkor az */etc/pam_ldap.conf* fájban kell a beállításokat elvégezni. Egyszerűbb azonban a ***libpam-ldap*** és a ***libnss-ldap*** csomagokat egyszerre a ***dselect*** segítségével telepíteni, mert ekkor a feltett kérdésekre adott válaszok alapján az */etc/pam_ldap.conf* tartalma is módosul.
Az */etc/pam_ldap.conf* fájl következő sorait kellett módosítanom:

```
host 127.0.0.1
base dc=cegnev,dc=hu
ldap_version 3
scope sub
pam_password md5
```

Módosítottam a következő fájlokat az eredeti sorok kikommentezésével és új sorok beírásával. A módosítás



1. ábra GQ beállítása: LDAP kiszolgáló adatainak megadása



2. ábra GQ beállítása: adminisztrátor megadása

után először az *LDAP* kiszolgálóról próbál azonosítani, ha ez nem lehetséges, akkor pedig a rendszerfájlokból:

/etc/pam.d/common_account fájl:

```
# account       required        pam_unix.so
account         sufficient      pam_ldap.so
account         required        pam_unix.so
```

/etc/pam.d/common_auth fájl:

```
# auth    required      pam_unix.so nullok_secure
auth      sufficient    pam_ldap.so
auth      required      pam_unix.so try_first_pass
```

/etc/pam.d/common_password fájl:

```
# password  required      pam_unix.so nullok
                          ➥obscure min=4 max=8 md5
password    sufficient    pam_ldap.so
password    required      pam_unix.so nullok
                          ➥obscure min=4 max=8 md5
```

/etc/pam.d/common_session fájl:

```
# session       required        pam_unix.so
session         sufficient      pam_ldap.so
session         required        pam_unix.so
```

A beállítás a *Linux* újraindítása után fog működni. Kipróbálni legegyszerűbben egy csak *LDAP* adatbázisban lévő felhasználó nevében történő bejelentkezéssel lehet.

Meglévő Linux csoportok és fiókok átvitele

Amennyiben a rendszerfiókokon és rendszercsoportokon kívül csak *Samba* fiókokra és csoportokra van szükségünk, úgy ez a lépés természetesen kimarad.

Az átvitelre (migrációra) a *migrationtools* csomagot kell telepíteni:

```
apt-get install migrationtools
```

Módosítani kell az */etc/migrationtools/migrate_common.ph* fájlban a következőket:

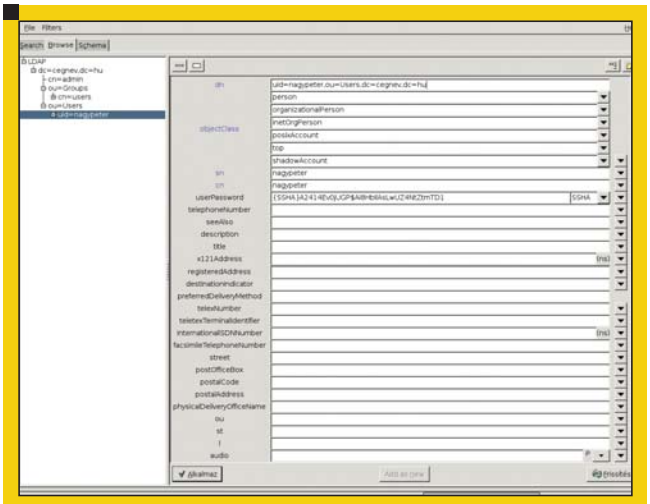
```
$DEFAULT_BASE = "dc=cegnev,dc=hu";
$EXTENDED_SCHEMA = 1;

# A következő sorokat pedig ki kell
# kommentezni:
#$DEFAULT_MAIL_DOMAIN = "padl.com";
#$DEFAULT_MAIL_HOST = "mail.padl.com";
```

Az */usr/share/migrationtools* könyvtárban lévő programokkal lehet a szükséges *.ldif* fájlokat létrehozni, a következő módon:

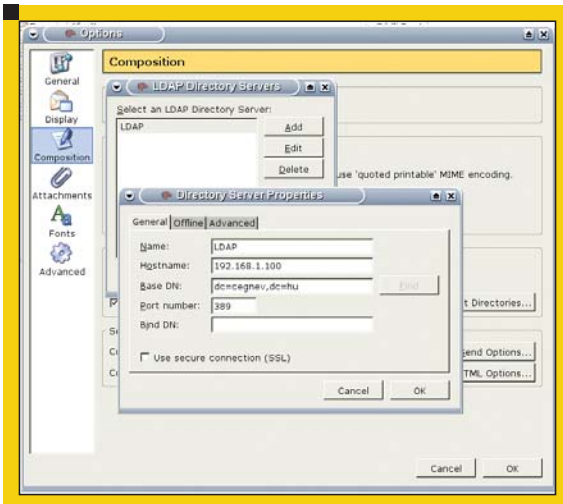
```
cd /usr/share/migrationtools
./migrate_base.pl  > ./base.ldiff
./migrate_hosts.pl  /etc/hosts > ./hosts.ldiff
./migrate_passwd.pl /etc/passwd > ./passwd.ldiff
./migrate_group.pl /etc/group > ./group.ldiff
```

A létrejött *.ldif* fájlok tartalmát a következő parancsokkal lehet az *LDAP* adatbázishoz adni:

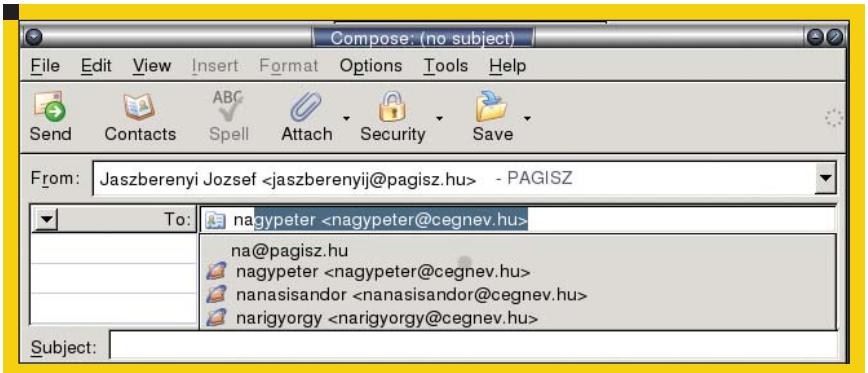
```
ldpadd -x -D "cn=admin,dc=cegnev,dc=hu" -W -f
➥./base.ldiff
ldpadd -x -D "cn=admin,dc=cegnev,dc=hu" -W -f
➥./hosts.ldiff
ldpadd -x -D "cn=admin,dc=cegnev,dc=hu" -W -f
➥./passwd.ldiff
ldpadd -x -D "cn=admin,dc=cegnev,dc=hu" -W -f
➥./group.ldiff
```



3. ábra Adatbázis böngészése GQ-val



4. ábra Mozilla Thunderbird beállítása LDAP kiszolgálóhoz



5. ábra E-mail cím keresése címzett megadásához

Bind DN:
(cn=admin,dc=cegnev,dc=hu)

Az *Ask password on first connect* kapcsolót bekapcsolva hagytam, így nem lehet megadni az adminisztrátori jelszót ebben az ablakban, hanem az adatbázis első elérésekor bekéri a program.
A *Browse* fülre kattintással lehet az adatbázist megnézni és szükség szerint módosítani. A *Search* fülre kattintás után az adatbázisban keresni lehet.

Grafikus LDAP adatbázis-kezelő program telepítése

Az eddigiekben minden esetben parancssorból mutattam be az adatbázis műveleteket. Számos grafikus felületű program áll rendelkezésre (például *GQ*, *Directory Assistant*, *Webmin LDAP* kezelő moduljai, *php4-ldap*, stb.) ezeknek a műveleteknek a kényelmesebbé tételére. Én a *GQ*-t használom a saját gépemen *KDE* alatt, de kiválóan működik más ablak kezelőkkel is.

Telepítettem *gq* csomagot:

```
apt-get install gq
```

Elindítása után először beállítottam az adatbázis elérésének adatait, a következő módon:

```
File -> Preferences -> Servers -> New
Name: beállítás tetszőleges neve (LDAP)
LDAP Host/Uri: LDAP kiszolgáló IP címe, vagy DNS
neve (192.168.1.100)
LDAP port: LDAP kiszolgáló portja (389)
Base DN: adatbázis alap (dc=cegnev,dc=hu)
```

Mivel az adatokat nem csak megnézni hanem módosítani is akartam, a *Details* fülre kattintás után beállítottam az adminisztrátort:

Levelezőkliens beállítása

A fejlettebb levelezőprogramok képesek az *LDAP* kiszolgálóban nevek, illetve e-mail címek keresésére. A *Mozilla Thunderbird* beállítását mutatom be, de természetesen más programok (pl. *MS Outlook Express*) is tudják használni, csak a beállításuk másképpen történik.
Megjelenítettem az *Edit: szerkesztés -> Preferences: beállítások -> Composition: Levél írása* ablakot. *Directory Server: címtár kiszolgáló* kapcsolót bekapcsoltam, majd az *Edit Directories...: címtárak szerkesztése* gombra kattintottam.
A megjelenő ablakban *Add: hozzáadás* gombra kattintottam, majd a mezőket kitöltöttem a következő módon:

NAME: beállítás tetszőleges neve (*LDAP*)
Hostname: kiszolgáló *DNS* neve, vagy *IP* címe (*192.168.1.100*)
Base DN: adatbázis kezdőpont (*dc=cegnev,dc=hu*)
Port (*389*)

Megfelelő beállítást követően levél írásakor a címzett mezőbe elég begépelni a címzett nevének, vagy e-mail címének néhány kezdőbetűjét és megjelenik az összes olyan név és cím, ami az adott betűkkel kezdődik.
A keresést szűkíteni több betű begépelésével lehet.
MS Outlook Express használata esetén kicsit bonyolultabb ez a művelet: új levél írásánál a címzett

1. Lista /etc/ldap/slapd.conf módosítása Samba azonosításhoz

```
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/
                ➥inetorgperson.schema
# Hozzáírtam a SAMBA schema fájlt:
include         /etc/ldap/schema/samba.schema

#  Módosítottam, hogy milyen index állományok
#  legyenek:
# index           objectClass eq
index           uid,uidNumber,gidNumber,
                ➥memberUid   eq
index           cn,mail,surname,givenname
                ➥eq,subinitial
index           sambaSID                    eq
index           sambaPrimaryGroupSID        eq
index           sambaDomainName             eq

# Módosítottam a hozzáférési jogosultságokat,
# hogy a SAMBA jelszavakat
# is tudja módosítani az LDAP adminisztrátor és
# a tulajdonos:
# access to attrs=userPassword
access to attrs=userPassword,sambaNTPassword,
➥sambaLMPassword
  by dn="cn=admin,dc=cegnev,dc=hu" write
  by dn="cn=replicator,dc=cegnev,dc=hu" read
  by anonymous auth
  by self write
  by * none

access to *
        by dn="cn=admin,dc=cegnev,dc=hu" write
        by * read
```

megadása ablakban az *LDAP* kiszolgáló kiválasztása után a keresés funkcióval végezhető el.

LDAP kiszolgáló beállítása Samba kiszolgálóhoz

Munkahelyemen a munkaállomásokon *MS Windows* működik, ezért van szükség *Samba* kiszolgálóra. Természetesen az *LDAP* kiválóan működik *Samba* nélkül is, így ha valakinek nincs szüksége rá, akkor ezek a lépések értelem szerűen elhagyhatók.
Az *LDAP* kiszolgáló részére a *schema* fájlok írják le, milyen mezőket tud tárolni. Ezek helye a */etc/ldap/schema* könyvtárban van. Ha *Samba* azonosítást kívánunk használni, akkor ki kell egészíteni a könyvtár tartalmát a *Samba* részére készített *schema* fájlal. Telepítettem ehhez a *samba-doc* csomagot, a tömörített *schema* fájlt bemásoltam kicsomagolva az */etc/ldap/schema* könyvtárba és beállítottam a tulajdonost, illetve a jogosultságot:

```
apt-get install samba-doc
zcat /usr/share/doc/samba-doc/examples/LDAP/
➥samba.schema.gz > /etc/ldap/schema/samba.schema
chown slapd.slapd /etc/ldap/schema/samba.schema
chmod 640 /etc/ldap/schema/samba.schema
```

LDAP kiszolgálót leállítottam, majd módosítottam az */etc/ldap/slapd.conf* fájlt (itt most csak a *master* kiszolgálón végzett módosításokat mutatom be; *1. Lista*).
Ellenőriztem a */etc/slapd.conf* fájlt:

```
slaptest
```

LDAP kiszolgálót elindítottam, majd működését ellenőriztem.

Samba fiókok kezelése

Az *smbldap-tools* csomag *Perl* nyelven íródott programjaival kényelmesen lehet a *Samba* fiókokat kezelni.
Ezek a parancsok megtalálhatók a *samba-doc* csomag telepítése után, az */usr/share/doc/samba-doc/examples/LDAP/smbldap-tools-x.x.x* könyvtárban is. A csomagot az elsődleges *Samba* tartományvezérlőre (*PDC*) telepítettem:

```
apt-get install smbldap-tools
```

A függőségek miatt nálam 14 csomagot telepített. Parancsai a */usr/sbin* könyvtárba kerülnek, de használatuk előtt be kell állítani. Átmásoltam először a */usr/share/doc/smbldap-tools/examples* könyvtárból az *smbldap.conf.gz* fájlt kitömörítve és az *smbldap_bind.conf* fájlt az */etc/smbldap-tools* könyvtárba:

```
zcat /usr/share/doc/smbldap-tools/examples/
➥smbldap.conf.gz > /etc/smbldap-tools/
➥smbldap.conf
cp /usr/share/doc/smbldap-tools/examples/
➥smbldap_bind.conf /etc/smbldap-tools/
➥smbldap_bind.conf
```

Munkahelyemen eddig is működött *Samba* tartományvezérlő. Mivel a tartomány azonosítónak (*SID*) nem szabad megváltozni, lekérdeztem a régi *Samba* kiszolgálónkon az értékét és egy fájlba írattam, ahonnan szövegszerkesztővel másoltam át:

```
net getlocalsid tartomanynev > SID
```

Módosítottam az */etc/smbldap-tools/smbldap.conf* fájlt a *2. Listának* megfelelően.
Az */etc/smbldap-tools/smbldap_bind.conf* fájlban beállítottam a master kiszolgáló adminisztrátort és jelszavát:

```
masterDN="cn=admin,dc=cegnev,dc=hu"
masterPw="adminjelszo"
```

A titkosítás nélkül megadott jelszó miatt veszélyes lenne ha ehhez idegen hozzáférne, ezért módosítottam a jogosultságát:

```
chmod 600  /etc/smbldap-tools/smbldap_bind.conf
```

2. Lista /etc/smbldap-tools/smbldap.conf fájl

```
# Mentett tartomány azonosító (SID) érték
SID="S-1-5-21-2139989288-483860436-2398042574"

# Ha van szolga LDAP kiszolgáló, akkor a címe
# és a portja
slaveLDAP="127.0.0.1"
slavePort="389"

# LDAP kiszolgáló címe és portja
masterLDAP="127.0.0.1"
masterPort="389"

# A TLS bejegyzéseket csak titkosított
# kapcsolat esetén kell beállítani

# Adatbázis kiindulási pontja
suffix="dc=cegnev,dc=hu"

# Felhasználófiókok, csoportok, gépek, stb.
➥tárolóegységeinek neve
usersdn="ou=Users,${suffix}"
computersdn="ou=Computers,${suffix}"
groupsdn="ou=Groups,${suffix}"
idmapdn="ou=Idmap,${suffix}"

# SAMBA Domain neve
sambaUnixIdPooldn="sambaDomainName=CEGNEV,
➥${suffix}"

# Keresési hatókör az adatbázisban
scope="sub"

# Milyen titkosítást használjon a jelszavakhoz
hash_encrypt="SSHA"

# A továbbiakban a fájlban UNIX és a SAMBA fiók
# alapbeállítások vannak
#  shell beállítás, home könyvtár, login
# script, stb. ezeket értelem szerint
# módosítottam
```

A beállítások elvégzése után először létrehoztam a ***Samba*** számára szükséges alap adatbázis-szerkezetet a következő paranccsal:

```
smbldap-populate
```

A csomag további parancsait az *1. Táblázat* tartalmazza. A parancsok alkalmasak ***UNIX*** és ***Samba*** csoportok és fiókok kezelésére egyaránt. Valamennyi parancsról segítséget lehet kiíratni a parancs után gépelt `-h` kapcsolóval. Az ***smbldap*** parancsok használatára a *2. Táblázatban* találunk néhány példát.
Samba gépfiókot létrehozni én csak 2 egymás után kiadott paranccsal tudtam.

1. táblázat *Az smbldap parancsai*

`smbldap-useradd`	fiók létrehozása
`smbldap-userdel`	fiók törlése
`smbldap-usermod`	fiók módosítása
`smbldap-usershow`	fiók megjelenítése
`smbldap-userinfo`	alapértelmezett héj, teljes név, stb. módosítása
`smbldap-passwd`	jelszó módosítása
`smbldap-groupadd`	csoport létrehozása
`smbldap-groupdel`	csoport törlése
`smbldap-groupmod`	csoport módosítása, csoporttagok hozzáadása és törlése
`smbldap-groupshow`	csoport adatainak és tagjainak megjelenítése

2. táblázat *Néhány példa az smbldap parancsainak használatára*

`Smbldap-groupadd csoport`	Létrehoz megadott nevű UNIX csoportot
`smbldap-groupadd -a csoport`	Létrehoz megadott nevű Samba csoportot
`smbldap-groupmod -m fiok csoport`	Hozzáadja a megadott fiókot a csoporthoz
`smbldap-useradd fiok`	Létrehoz megadott nevű UNIX fiókot
`smbldap-useradd -a fiok`	Létrehoz megadott nevű Samba fiókot
`smbldap-passwd fiok`	Módosítja a fiók jelszavát

Előbb az

```
smbldap-useradd -w -d /dev/null -s /bin/false
gepnev
```

paranccsal létrehozom a megadott nevű gépfiókot, majd a

```
smbldap-usermod -a gepnev$
```

művelettel hozzáadom a fiókhoz a ***Samba*** mezőket.

Elsődleges tartományvezérlő Samba kiszolgáló beállítása LDAP azonosításhoz

A ***Samba*** leállítása után módosítottam az */etc/samba/smb.conf* fájlt az *3. Listának* megfelelően. Itt most csak az ***LDAP*** kiszolgálóval való együttműködés miatti fontos részeket mutatom be:
Minden ***Samba*** kiszolgálóra telepítettem ***LDAP slave*** kiszolgálót a gyorsabb működés érdekében, ezért van a ***localhost*** megadva a kiszolgáló nevénél. Az `add machine script` kezdetű sor biztosítja, ha egy munkaállomást újratelepítés

3. Lista /etc/samba/smb.conf fájl LDAP azonosításhoz

```
[global]
  workgroup = DOMAIN_NEV
  netbios name = SERVER_NEVE
  server string = %h Linux (Samba %v)
  browseable = yes

  security = user
  local master = yes
  os level = 255
  domain master = yes
  preferred master = yes
  domain logons = yes

  log file = /var/log/samba/log.%m
  max log size = 1000
  syslog only = no
  syslog = 0

  passdb backend = ldapsam:"ldap://localhost"
  ldap timeout = 10
  ldap admin dn = cn=admin,dc=cegnev,dc=hu
  ldap suffix = dc=cegnev,dc=hu
  ldap group suffix = ou=Groups
  ldap user suffix = ou=Users
  ldap machine suffix = ou=Computers
  ldap idmap suffix = ou=Idmap

  add machine script = /usr/sbin/smbldap-useradd
  ➥-w "%u"
  # add user script = /usr/sbin/smbldap-useradd
  ➥-m "%u"
  # ldap delete dn = Yes
  # delete user script = /usr/sbin/smbldap-
  ➥userdel "%u"
  # add group script = /usr/sbin/smbldap-
  ➥groupadd -p "%g"
  # delete group script = /usr/sbin/smbldap-
  ➥groupdel "%g"
  # add user to group script = /usr/sbin/
  ➥smbldap-groupmod -m "%u" "%g"
  # delete user from group script = /usr/sbin/
  ➥smbldap-groupmod -x "%u" "%g"
  # set primary group script = /usr/sbin/
  ➥smbldap-usermod -g '%g' '%u'

  encrypt passwords = true
  ldap passwd sync = Yes
  # unix password sync = Yes
  passwd program = /usr/sbin/smbldap-passwd
  ➥-u %u
  passwd chat = "Changing password for*\nNew
  ➥password*" %n\n "*Retype new password*"
  ➥%n\n"
  obey pam restrictions = no

  admin users = administrator
  invalid users = root
  guest account = nobody

  logon script = scripts\%U.bat
  logon path =
  logon home =
```

után vissza kell tenni a tartományba, automatikusan frissüljön a fiókja. A `logon path =` és `logon home =` paraméter nélküli sorokra azért van szükség, hogy alapértelmezés szerint helyi profilja legyen a felhasználóknak. Ha ez a két sor hiányzik, akkor az alapértelmezés a központi vándor profil, még akkor is, ha a felhasználó fiókjában törölve van a profil elérési útja.
A *Samba*-nak megadtam az *LDAP* adminisztrátor jelszavát, amit a */var/lib/samba/secrets.tdb* fájlban tárol:

```
smbpasswd -w adminjelszo
```

Elindítottam a *Samba* kiszolgálót:

```
/etc/init.d/samba start
```

Ellenőriztem a *Samba* működését a felhasználófiókok kiíratásával. Mivel még más fiókot nem hoztam létre, csak az *administartor* és a *nobody* felhasználó jelenik meg, melyeket az `smbldap-populate` parancs hozott létre:

```
pdbedit -L
```

Egy fiókról részletes információt a következő paranccsal lehet megjeleníteni:

```
pdbedit -Lv fioknev
```

Meglévő Samba fiókok és csoportok átvitele (migráció)

Ez az egyik legnagyobb probléma, mert az átvitt fiókok nevének, azonosítójának, jelszavának, stb. meg kell egyezni, máskülönben használhatatlanok lesznek. Erre a problémára nem találtam igazán jól használható megoldást, így magam írtam meg a szükséges programot *Perl* nyelven. A program futtatásához 4 fájlra volt szükségem az eredeti kiszolgálókról. Ezek közül az első kettő az a */etc/passwd* és */etc/shadow* (melyek a felhasználókat és a gépfiókokat is tartalmazták) az eddigi *NIS master* kiszolgálónkról másoltam le.
A másik két fájlt az eredeti *Samba* kiszolgálón állítottam elő, a következő parancsokkal:

```
pdbedit -Lv > smbfiokok
pdbedit -Lw > smbpasswd
```

Valamennyi fájlt átmásoltam a *master LDAP* kiszolgáló egy könyvtárába. Szövegszerkesztővel átnéztem a *Samba* kiszolgálón előállított fájlokat és eltávolítottam a zavaró üzeneteket az elejükről és végükről. Az „*smbfiokok*” fájlnak a művelet után „----------” karaktersorozattal kell kezdődni (a fiókokat ez választja el egymástól), az *smbpasswd* fájlban pedig minden egyes sornak egy fiókadatot kell tartalmazni. Az *smbfiokok* fájlból eltávolítottam azokat a fiókokat, melyeket nem kívántam átvinni az *LDAP* adatbázisba. A *passwd* és *shadow* állományokat nem módosítottam, mivel a program csak azokat a fiókokat olvassa ezekből az állományokból, melyek az *smbfiokok* fájlban megtalálhatók. Elindítottam az általam készített *smbuserldif* programot, ami elkészítette a felhasználó- és gép-fiókok hozzáadására alkalmas *smbldap.ldif* fájlt. Ennek tartalmát a szokásos módon az adatbázishoz adtam:

```
smbldap -x -D "cn=admin,dc=cegnev,dc=hu" -W -f
smbldap.ldif
```

A csoportok átvitelére a fenti program nem alkalmas, ezért egy másik kis programot készítettem, amelyik ezt a feladatot elvégzi. A *NIS* kiszolgálóról lemásoltam az */etc/group* fájlt és szövegszerkesztővel eltávolítottam azokat a csoportokat, melyeket nem kívántam átvinni. Az általam készített *migrategroup* program futtatásával létrehoztam a *UNIX* csoportokat és felvettem azokba a szükséges felhasználókat. Ha valaki *Samba* csoportot kíván átvinni, akkor hasonlóan járhat el, de a programban az `smbldap-groupadd` parancs után egy `-a` kapcsolót kell elhelyezni. A program nem fogja helyesen beállítani azonban a *Samba* csoport azonosítóját, ezért azt le kell kérdezni az eredeti *Samba* kiszolgálón és az adatbázisban módosítani (például *GQ*). *Samba* kiszolgálón a *Samba* csoportokat és azonosítójukat a következő paranccsal lehet lekérdezni:

```
net groupmap list
```

Postfix levelező kiszolgáló beállítása LDAP azonosításhoz

Telepítettem a *postfix LDAP* azonosító modulját:

```
apt-get install postfix-ldap
```

Létrehoztam egy */etc/postfix/ldap-aliases.cf* fájlt a következő tartalommal:

```
version = 3
server_host = localhost
search_base = dc=cegnev,dc=hu
```

A */etc/postfix/main.cf* fájlt a következőképpen módosítottam:

```
alias_maps = hash:/etc/aliases,
ldap:/etc/postfix/ldap-aliases.cf
```

Postfix konfigurációját újra beolvastattam

```
/etc/init.d/postfix reload
```

Courier-POP3 és Courier-IMAP beállítása LDAP azonosításhoz

Munkahelyemen a *Courier POP3* és *IMAP* kiszolgálók titkosítás nélküli és titkosított változata is telepítve van, mivel a hálózaton kívülről csak titkosítottan érhető el a levelezés. Az azonosítás a *PAM*-on keresztül történik. Módosítani kell az */etc/pam.d/pop3* és */etc/pam.d/imap* fájlok tartalmát, mivel azokban nem a *Debian „sarge”*-ban szokásos common-* fájlokra történő hivatkozás van. Mind a két fájlban „kikommenteztem” az eredeti sorokat és beírtam a következőket:

```
@include common-auth
@include common-account
@include common-password
@include common-session
```

Azonosítást használó squid proxy beállítása LDAP kiszolgálóhoz

Intézményünkben az internetezés csak *proxy* kiszolgálón keresztül, azonosítás után lehetséges. Ezzel szűrni tudjuk a tanulók által megjeleníthető tartalmat és a *proxy* használat lehetővé teszi a sávszélesség korlátozást is, amire a szűkös internet sávszélességünk miatt nagy szükség van.
A */etc/squid/squid.conf* fájlba a következő sorokat írtam:

```
# "basic" az azonositas tetszoleges neve
# -h utan kell megadni az LDAP kiszolgalo IP
cimet
auth_param basic program /usr/lib/squid/ldap_auth
-b "dc=cegnev,dc=hu" -v 3 -h 192.168.1.100

# Az acl.dolgozok es acl.tanulok fajlokban vannak
a felhasznalonevek (soronkent egy nev)
# Az acl-ek ezután már szokás szerint
használhatók
acl dolgozok proxy_auth "/etc/squid/
➥acl.dolgozok"
acl tanulok  proxy_auth "/etc/squid/
➥acl.tanulok"
```

Az *LDAP* azonosítás nálunk több hete hibátlanul működik. Miután minden olyan kiszolgálóra telepítettem *slave* kiszolgálót, mely gyakran igényel azonosítást, a sebességgel sem volt semmi probléma. A kiszolgálók erőforrásainak terheltségét csak minimális mértékben növelte a *NIS*-hez képest.
A kedvező tapasztalataim miatt minden közepes és nagyobb hálózatban ajánlani tudom kipróbálását.

Jászberényi József
(jaszberenyij@pattantyus-gyor.sulinet.hu)

Szeret biciklizni, kirándulni, olvasni, sörözni és szabadban főzni. A stratégiai játékoktól a műszaki CAD programokig sok minden érdekli. Legtöbbet szerverprogramokkal foglalkozik és néha mérgelődik.


Linuxvilág
www.linuxvilag.hu

Linux fotósoknak

Nem is olyan régen még csak a kiváltságosok használhattak digitális fényképezőgépeket, sőt mint ahogy már megszokhattuk, többnyire csak Windows vagy MacOS alatt. Persze, mint sok más területen, úgy a fotózás területén is sokat fejlődött a Linux az utóbbi időben.

Nem is olyan régen – amikor még én is az első fényképezőgépemet vásároltam – még akár százezer forintot is az asztalra kellett tenni egy – már igazán – jól használható gépért. Persze azóta eltelt pár év – szám szerint körülbelül három – és ugyanilyen tudású gépet már akár harminc-negyvenezer forint körüli áron is találhat a kedves olvasó.

Hardver

Fényképezőgépet vásárolni *Linux* alá az egyik leghálásabb dolog, amennyiben memóriakártyára szeretnénk dolgozni. Én elítélem azokat a gyártókat, akik beépítenek tizenegynéhány megabájt memóriát a fényképezőgépbe, hiszen vésztartaléknak jó lehet, de *Linux* alatt szinte esélytelen az ott tárolt adatok elérése, míg a kártyát bármilyen kártyaolvasóba behelyezve olvashatjuk annak tartalmát akkor is, ha a fényképezőgépünk nem támogatott.

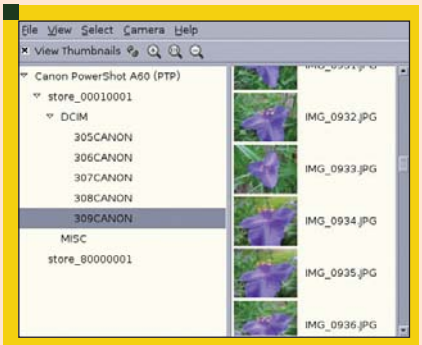
Áttöltés – kábel vagy kártyaolvasó

Ha fényképezőgéppel szeretnénk áttölteni a képeket, akkor nem árt, ha a fényképezőgép kezeli a *mass-storage* szabványt. Amennyiben ez a helyzet, úgy *pendrive*-ként látszik a *Linux* számára és egyszerű csatolás (*mount*olás) után elérhető a kártya tartalma.
A másik lehetőség, amelyet számos fényképezőgép kínál, az a *PTP*, vagyis *Picture Transfer Protocol*. Ezzel azonban óvatosan kell bánni, hiszen a linuxos programok nem feltétlen támogatják az adott fényképezőgépet. Erről később a *gphoto2* és *gtkam* programok ismertetésénél térek ki.
Láttam már olyan fényképezőgépet is, amelynek a menüjében lehetett kiválasztani, hogy a két protokoll közül melyiket használja, de ez ritka.
A kártyaolvasók mellett több érv is szól: például nem biztos, hogy egy *Nikon* fényképezőgép *CF* (*Compact Flash*) kártyát mondjuk egy *Canon* hibátlanul olvas, vagy fordítva. Érv az is, hogy a mai *kártyaolvasók* általában *többféle kártyát is olvasnak*, így ha egy barátunk megkér, hogy írjuk ki a memóriakártyáján lévő képeket *CD*-re vagy *DVD*-re, adhat bármilyen kártyát, akkor sincs gond.
Továbbá a sebesség sem elhanyagolható, egy mai átlagos – 3-5 ezer forintos – kártyaolvasó *USB 2*-es szabványú, vagyis képes akár *480 megabites* sebességre is, de ez általában 30-40 megabit (4-5 megabájt másodpercenként) vagy speciális kártya esetén akár a duplája, triplája is lehet.
Nálam azonban nem kérdés, hogy kártyaolvasóval töltöm át az adatokat. Két oka is van. Az egyik: az új fényképezőgépem (*Panasonic FZ30*) csak *PTP*-t támogat, de a *gphoto2* a cikk írásakor – még – nem ismeri az eszközt. A másik ok sokkal gyakorlatiasabb: a fényképezőgépem – annak ellenére, hogy 2005-ös modell – csak 12 megabites sebességre lenne képes, amivel már a jelenlegi fél gigabájtos kártyám áttöltése is körülbelül 10 perc lenne.

Áttöltés (gphoto2 és gtkam)

Még mielőtt valaki voksolna a *gphoto2* és a *gtkam* között, jó tudni, hogy a *gtkam* a *gphoto2* grafikus megvalósítása, ami a kezdő (vagy csak lusta) felhasználóknak jobb. Hogy teljes legyen a cikk, bemutatom, hogyan lehet áttölteni a *gphoto2*-vel a képeket vagy annak egy részét.

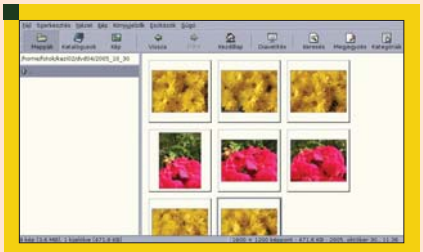
1. ábra Gtkam egy Canon Powershot A60-al



2. ábra Gtkam, a kis képek áttöltése után

Mi is az EXIF és miért jó?

Az ***EXIF (Exchangeable image file format)*** nem más, mint egy adatblokk a digitális fényképekben, amely rengeteg hasznos adatot tartalmaz(hat), mint például a ***fotózás dátumát/idejét*** (amennyiben pontosan beállítottuk a fényképezőgép óráját), a használt ***felbontás***t, a ***blende*** nyílást, a ***zársebesség***et, a ***fókusztávolság***ot, a használt ***gép típus***át, sőt akár GPS koordinátákat is, amennyiben tud ilyet a fényképezőgép.
Mindez persze arra jó, hogy mondjuk hiába változott a jpeg fájl dátuma, felbontása, stb., az *EXIF*-ben tárolt adat továbbra is az eredeti marad (amennyiben nem töröljük a fájlból), így évek múlva is tudunk keresgélni a képek között bármilyen, az *EXIF*-ben letárolt tulajdonság alapján.



3. ábra Gthumb akcióban

Először is meg kell győződnünk, hogy látja-e a fényképezőgépet a számítógép. Erre a `gphoto2 --auto-detect` parancs alkalmas. Jó esetben kiírja a fényképezőgép típusát, valamint, hogy milyen módon csatlakozik a géphez.
A fényképezőgép által kezelt mappák letöltéséhez használjuk a `-l` opciót. (Amennyiben több fényképezőgép van egyszerre a *PC*-re csatlakoztatva, úgy paraméterekkel kiválaszthatjuk, melyik gépen kívánunk dolgozni.) A `-L` opcióval megnézhetjük a fájlok listáját.
A képek áttöltése `-p` paraméterrel (tartomány letöltése) vagy `-P` paraméterrel (ekkor mindent letölt) indítható.
Mint már mondtam, a gtkam kicsit barátságosabb. Itt csak annyi a dolgunk, hogy a *Camera* menüben felvesszük a kameránkat típus és kapcsolódási mód alapján. Ha ez megvan, onnan már gyerekjáték a képek letöltése. Első körben az előnézeti képek (*thumbnails*) töltődnek le (persze ez kikapcsolható és akkor csak a fájlnevek látszanak), aztán kijelölve, hogy nekünk mik kellenek, jönnek le a teljes méretű képek. Több kép letöltésének kijelölése a már ismert *bal egérgomb* lenyomásával, valamint a *CTRL* vagy a *Shift* gombok nyomva tartásával történik.
Fontos megjegyezni, hogy mind a *gphoto2*, mind a *gtkam megfelelő privilégium* esetén megy. Ez azt jelenti, hogy a felhasználónknak szerepelni kell a *plugdev* és a *camera csoport*okban.

Nézegetés (gthumb, gqview, fbi, jhead)

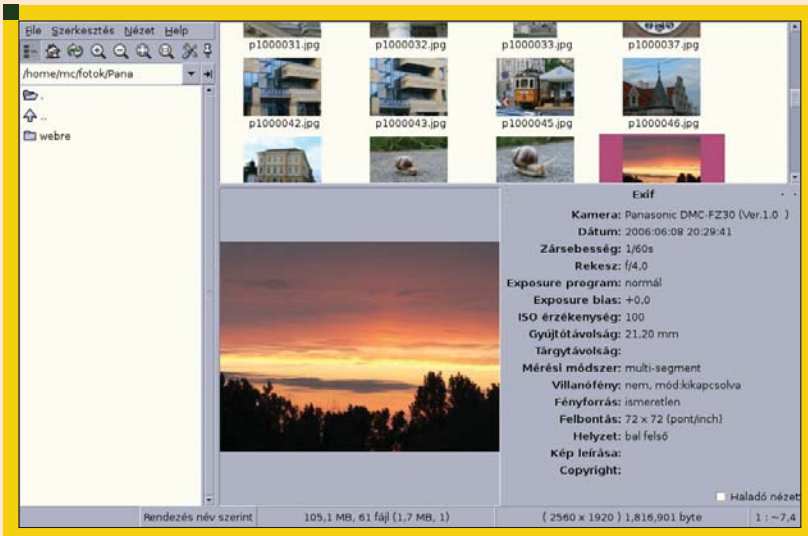
Szép és jó az áttöltés, de meg is kell nézni a képeket. Erre a feladatra a *gthumb*-ot vagy a *gqview*-t ajánlom. A *gqview* szerintem sokkal használhatóbb annak, aki *Windows* alatt már megszokta az *Irfanview*-t. Hasonlóan egyszerű a kezelése. Az *„F”* gombra *teljes képernyőt* kapunk, míg az *„X”*-re a képet *ablakméretűvé* méretezhetjük, illetve az *„1”*-esre *1:1*-ben látni a pixeleket. Természetesen az *Irfanview* gazdagabb funkciókban, de ennek ellenére a *gqview* is számos hasznos lehetőséget kínál. Például az aktuális képen *jobb gomb*ot nyomva azonnal indítható a *Gimp*, vagy *megnézhetjük* a képhez tartozó *Exif* információkat is.
A kedvencem a csoportos elforgatás. Ki jelöljük a képeket *CTRL + bal egérgomb* kombinációval, majd jobbgomb után a helyi menüben lehet kérni balra vagy jobbra forgatást.
Persze nemcsak az X kiváltsága a képnézegetés, hiszen parancssorból is rengeteg dolgot meg tudunk oldani. Itt van rögtön például az *fbi* nevű képnézegető program. Ehhez nem kell más, mint működő *FrameBuffer* a kernelben, és máris megy.
Az *EXIF* információkat konzolos felületen a *jhead* segédprogrammal tudjuk lekérdezni.

Szerkesztés (imagemagick, convert, raw-szerkesztés)

Mindenkinek szoktam javasolni, hogy ne csak a *GIMP* legyen a gépen, de az *Imagemagick* csomag is. Ennek, mint látni fogjuk, lesz pár gyakorlati haszna. *Windows* alatt például az *Irfanview* tud csoportos átméretezést és/vagy minőségbeállítást (például *Jpeg* fájloknál). Nos ezt sem a *Gqview*, sem a *GIMP* nem tudja alapból. Persze lehetne egyenként, de tapasztalatom szerint az ember alapvetően lusta. Éppen ezért jómagam az *Imagemagick* részeként létező `convert` parancsot használom egy egyszerű `for` ciklussal a képek átméretezésére. Minthogy a `convert` parancs összes opciójának tárgyalásához minimum egy *Linuxvilág* számot fel kellene áldozni, így csak pár – általam a leghasznosabbnak tartott – opciót tárgyalok.
Hogy is néz ki egy egyszerű átméretezés?

RAW képek

A *raw* képek olyan képek, amiket a fényképezőgép processzora nem dolgozott fel, tehát nem ment rá élesítés, lágyítás, veszteséges tömörítés, így könnyebb a komolyabb utómunka vele. Egy-az-egyben azt kapjuk meg, amit a fényképezőgép érzékelőjéből maga a gép is kiolvas, nem nyúl hozzá a fényképezőgép processzora. Ezekkel a fájlokkal alapértelmezetten nem tudunk mit kezdeni, tehát például a *GIMP* sem olvassa be. Létezik azonban *Dave Coffin* jóvoltából a *dcraw* nevű program. Ez lehetővé teszi, hogy a *raw* formátumot „emberi" formátumra alakítsuk. Sőt, a *dcraw* létezik *GIMP-plugin*ként is. Persze az elképzelhető, hogy a disztribúciónk által szállított *dcraw* nem támogatja a fényképezőgépünket, így érdemes a fejlesztő honlapjáról letölteni a legfrissebbet, és lefordítani a honlapon leírt fordítási opciókkal. A fordításhoz természetesen szükség van pár csomag jelenlétére, de mindezt a forráskódból kiolvashatjuk.



4. ábra Gqview

```
for i in `ls *.jpg`; do convert
➥$i -quality 80 -resize
➥640x480 temp/$i; done
```

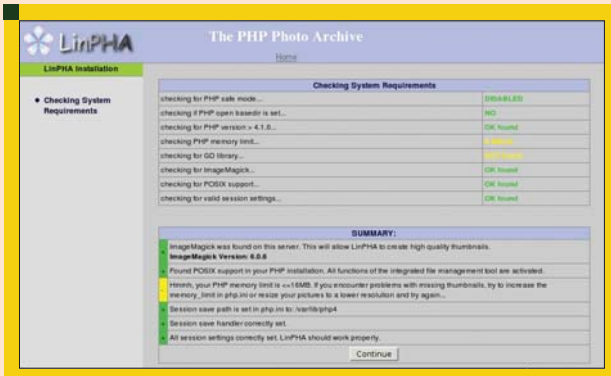
Ez nem tesz mást, mint átméretezi az összes jpg végződésű képet az aktuális könyvtárból maximum 640 képpont széles és 480 képpont magas 80%-os minőségű képekre, majd a kész állományokat a *temp*-be helyezi, amennyiben létrehoztuk a könyvtárat. A `convert` parancs nemcsak átméretezni tud, hanem feliratokkal is el tudja látni a képeket. Erre egy másik példa:

```
for i in `ls *.jpg`; do convert
➥$i -resize 640x480 -quality
➥60 -fill black -draw
➥"roundrectangle -10,456
➥200,472 6,6" -font helvetica
➥-fill yellow  -pointsize 15 -
➥draw "text 6,470 'Foto by
➥Medve 2006.07.13.'" temp/$i;
➥done
```

A parancssor rövid magyarázata: átméretezi a képeket maximum 640 pixel széles és 480 pixel magas képekre 60-as minőség mellett. Emellett rak rá egy lekerekített fekete téglalapot is, amelynek a bal oldala lelóg a képről, illetve 15 pixel magas sárga betűkkel a megadott koordinátákra írja a megadott szöveget. Fontos a sorrend, hiszen mindig a legutolsó művelet kerül legfelülre.

Webes publikálás (LinPHA)

Az ember általában nem(csak) magának fotózgat, hiszen ha valami páratlant örökít meg, azzal lehet dicsekedni, ha pedig csak egyszerű családi képről van szó, azt jó megmutatni a rokonoknak. A 21. század hajnalán hogyan máshogy lehetne ezt megtenni, mint egy *webes galéria* segítségével.
Erre eddig a *LinPHA* tűnt a legjobbnak. Maga a telepítés nem bonyolult folyamat, csupán pár klikk, illetve futtatnunk kell a *mysql* vagy a *postgresql* valamelyikét, persze amennyiben kevesebb privilégiummal rendelkezünk – nem rendszergazda – a szerveren, úgy ennek megfelelő telepítésre is van mód. Lássuk a telepítés részleteit.
Csomagoljuk ki egy olyan könyvtárba, amit lát az Apache (nálam például */var/www*).
A böngészőbe írjuk be az így keletkezett *LinPHA install könyvtárának* elérési útvonalát (nálam: *http://127.0.0.1/linpha-1.1.1/install/install.php*).
Hajtsuk végre a telepítést (fontos: az adatbázis rendszergazdai jelszónak nem fogad el üres jelszót).
Ha elkészültünk a telepítéssel, *biztonsági kockázat* miatt *töröljük az install könyvtárat*, hiszen arra már nem lesz szükség.
A belépéshez tartozó linket a jobb felső sarokban találjuk.
Itt szeretném megjegyezni, hogy a *Linpha* képes felhasználók és csoportok kezelésére is, ennek megfelelően pedig korlátozhatjuk a képek hozzáférését. Természetesen a program lehetővé teszi a képekhez történő hozzászólást is, amennyiben engedélyezzük.
A képek publikálása sokkal egyszerűbb, mint azt az ember egy ilyen kaliberű programtól várná: egyszerűen csak bemásoljuk a képeket a *linpha* könyvtárán belül található *albums* könyvtárba, majd pedig a galéria admin menüjében *legeneráltatjuk* az előnézeti képeket, illetve le tárolhatjuk az *Exif* információkat is, amennyiben van ilyen igényünk.
A program lehetővé teszi a menet közbeni *vízjelezés*t is a megfelelő *plug-in* bekapcsolása esetén. Ez annyit tesz, hogy az eredeti (szerveren lévő képhez) nem nyúl, hanem az átalakítottat adja át a webszervernek.
Ez egyrészről jó, hiszen nem kell két példányt tartani a képekből, másrészről *eléggé* elviszi a *gépidő*t az ezzel járó *művelet*. Érdemes tehát a vízjelet kikapcsolni, hacsak nem feltétlen szükséges.
A bekapcsolható *plug-in*-ek között találunk még többek között *RSS*-t és *vendégkönyv*et is.
Természetesen a program nagyszerűsége nem merül ki ezekben, hiszen lehetővé teszi, hogy megfelelő jogosultságok esetén – amit a helyi

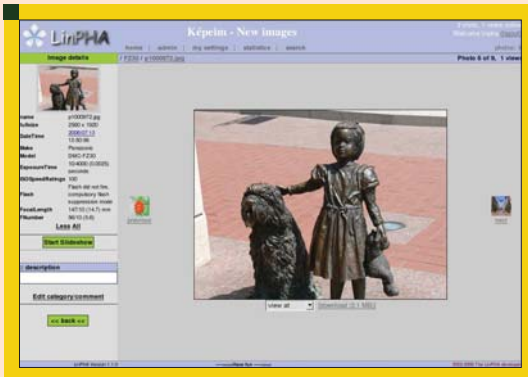


5. ábra Telepítés előtti ellenőrzés

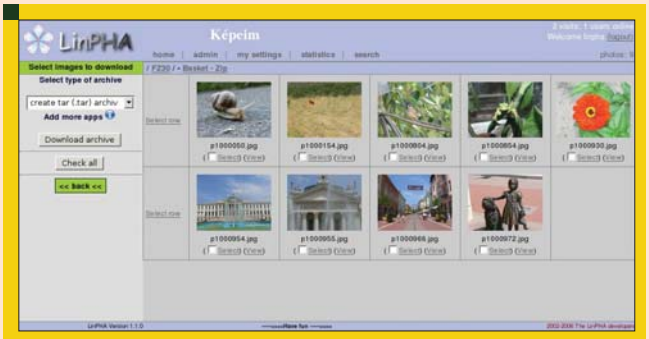
6. ábra Telepítés

7. ábra Sikeres telepítés

8. ábra Galéria



9. ábra Egy kép kinagyítva, bal szélen EXIF információk



10. ábra Sok kép letöltése egyszerre

LinPHA adminisztrátor oszt ki – a képeket egyszerre elküldjük ***email***ben, letöltsük tömörített állományként (***tar*** / ***tgz*** / ***tbz2*** / ***zip***), vagy akár kinyomtassuk. Az már csak hab a virtuális tortán, hogy egy adott időszakra vetítve megnézhetjük, melyek voltak a legnépszerűbb képek.

Jó tanács a végére

Minthogy a digitális képek sérülékenyebbek, mint a papír képek, ***érdemes gyakran lementeni*** őket akár olvasható médiára (***CD, DVD***), illetve több példányban, hiszen a mai olcsó adathordozók nem biztos, hogy olyannyira megbízhatóak, mint azt a gyártóik hangoztatják.

Nem szégyen többször learchiválni, sose lehet tudni. Inkább legyen két mentés, mint egy se.
Jó fényeket mindenkinek!

Medve Zoltán
(e-medve@e-medve.hu)

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

KAPCSOLÓDÓ CÍMEK

RAW programok
➲ http://www.cybercom.net/~dcoffin/dcraw/

LinPHA
➲ http://linpha.sourceforge.net/wiki/index.php/Main_Page

Gimp bővítmények (5. rész)
Képtelen játék a pixelekkel

Vidám mosolyt fakaszthatunk pillanatok alatt barátaink arcán, ha valamelyik ismerősünkről készült fotót kicsit megbűvöljük. Lehet nagyobb orrot, szemet és szájat alkotni a kezelésbe vett portréra, és ezzel a lehetőségeink még korántsem értek véget. Mind ehhez segítséget nyújt a Gimp, különösképpen pedig a torzító szűrők.

A klinikai egér szerepét a mai alkalommal egy olyan lófej fogja betölteni, mely egykor az athéni ***Parthenon*** tetejét díszítette. Eredetije a londoni ***British Múzeumban*** látható. Ott készült ez a fotó is. E nemes vonásokon fogom tehát illusztrálni a torzító szűrők hatását (***Szűrők->Torzítás***). Valószínűleg az ókori görögök nem is sejtették, hogy művészetüket a nyílt forrás szolgálatába fogja állítani az utókor...

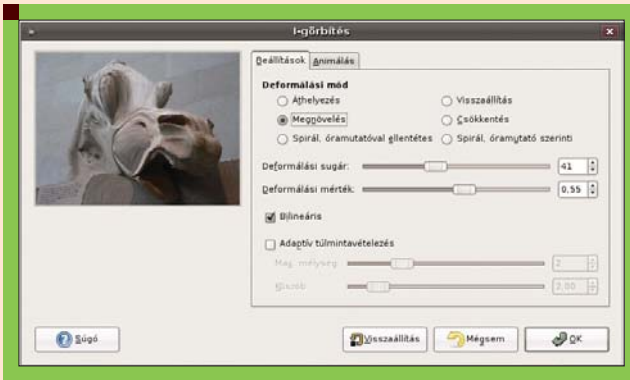
Hullámok

Hullámokat (Waves) pár kattintással varázsolhatunk a képre. A beállításokkal nem sok gondunk lesz, hiszen hasonlóan működik minden mint ahogy azt a ***Fodrozódás (Riple)*** szűrő esetben már láttuk. Még kevesebb beállításról kell gondot viselnünk most. Adjuk meg az ***Amplitúdó (Amplitudo)*** és a ***Hullámhossz (Phase)*** értékeket, és döntsük el, hogy mit tegyen a program a kép szélein. Ezt a mód felirat alatt, az ***Elkenés (Smear)*** vagy a ***Feketítés (Blacken)*** kapcsolót kiválasztva tehetjük meg.

IWarp

Az ***I-görbítés (IWarp)*** már a nevében hordozza, hogy itt valami mókás dolog fog következni. Ugyanis olvashatjuk elgörbítésnek is, ami csakugyan fedi a valóságot. Valóban, csalódni nem fogunk, hiszen pár egérkattintással a félreismerhetetlenségig torzíthatjuk jó barátaink arcvonását. Ez a szűrő végre nem sablon alapján dolgozik! Segítségével mi magunk vehetjük a kezünkbe az irányítást, így nincsen két megegyezően módosított kép. Csinálhatunk nagyobb szemet, szájat, hegyes ***Pinoccio*** szerű orrot és még sokféle mókát! Az előképre kattintva, mi magunk alakíthatjuk át a képünket. Minden kattintás, húzás változtat valamit az eredeti vonásokon. Válasszunk egy ***Deformálási módot (Deform mode)***, és fogjunk neki az átalakításnak. Hogy az egerünk mekkora környezetében változzon meg a kép azt a ***Deformálási sugár (Deform radius)***, intenzitását pedig a ***Deformálási mérték (Deform amount)*** adja meg. A bilineáris kapcsolót kiválasztva lágyabb hatást érhetünk el, az ***Adaptív túlmintavételezés (Adaptive supersample)*** ugyancsak szebb képet eredményez nagyobb számítási igény mellett. Lehetőségünk van arra is, hogy a lépésekből animációt készítsünk. A szükséges beállításokat az ***Animálás (Animate)*** fül alatt találhatjuk meg.

1. ábra Hullámok szűrő beállítási lehetőségei



2. ábra Az I-görbítés beállításai

3. ábra A Mozaikot már a régi görögök is szívesen alkalmazták...

4. ábra A Szél szűrő mozgást imitálhat

Page Curl

A *Lap felhajtása (Page curl)* régi trükk már. Mégis hasznos, hogy egy kattintással alkalmazhatjuk is a képünkre. Természetesen ezt is a torzító szűrők között találjuk.

Mozaikok

Ha már szó esett az ókori görögökről... Mind ők, mind a rómaiak igen szerették a mozaikokat. Biztosan mosolyra húzódna a szájuk, ha meglátnák a *Gimp* Mozaik szűrőjét. A következő eszközzel igazán korhű alkotásokat készíthetünk. Ismét nem egy univerzális eszközről van szó, valószínűleg nem fogjuk minden képünkre alkalmazni. Sok beállítási lehetőségével mégis számos próbálkozásra ösztönözhet. A beállítások magukért beszélnek, azokat emelném ki, melyek jelentését az első ránézésre jótékony homály borítja. A *Csempefelosztás (Alow tile splitting)* engedélyezése például azt jelenti, hogy megengedjük a programnak, hogy azokon a helyeken ahol sok szín találkozik, ott a csempéket szétdarabolja és így finomabb mintát hozzon létre. Ellenkező esetben robusztusabb eredményt kapunk.
Az *Előtér és háttérvilágítást (FG/BG lightning)* bekapcsolva módosulnak a színek, úgy mintha egy előtér színű és egy háttérszínű lámpa világítaná meg a csempéket.

Polárkoordináták

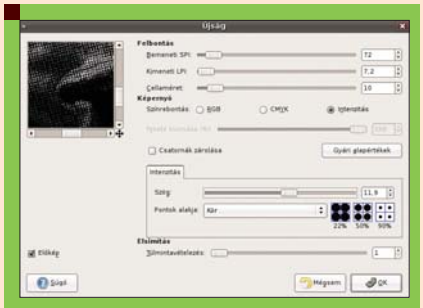
A Polárkoordináták szűrő segítségével a képünk kör vagy négyzet alapú reprezentációját kapjuk vissza, sőt a kettő közötti bármelyik átmenetet is megrajzoltathatjuk. Nem fogjuk sokat használni az biztos, de azért raktározzuk el valahol az emlékezetünk legmélyén, hogy ilyen is van.
Ha például egy üveggömbre szeretnénk rávarázsolni magunkat, talán még használt vehetjük.

Szél

A *Szél* szűrőt használhatjuk arra hogy a mozgást szemléltessük, vagy mint általános torzító eszközt. A képünkön vékony fekete-fehér vonalak fognak megjelenni. Ez hozzátartozik a hatáshoz. A szűrő ugyanis megkeresi a kép markáns éleit és a vékony vonalak azokból indulnak ki. Ezért tűnik az eredmény mozgónak, hiszen a valóságban is a mozgó tárgyak élei mosódnak el a fényképeken. Válasszunk először stílust! A *Szél (Wind)* esetén vékony vonalakat fog rajzolni, míg Széllökés (Blast) esetén vastagokat a program.

Újság

Mókás illusztrációk készíthetőek az *Újság (Newsprint)* szűrővel. Képünket átalakítja, mintha egy alacsonyabb felbontású eszközön jelenítenénk meg. Megváltoztatja a képpontok alakját telítettségét, így olyan hatást ér el, mint a kis felbontású nyomtatón készített lenyomatok. Adjuk meg a kezdeti és az eredményül kívánt felbontást (inchenkénti vonalak számát). Az kapott kép felbontása függ a cellamérettől, azzal fordítottan arányos. Döntsük el, hogy milyen elv szerint szeretnénk színre bontani a képet, választhatjuk az *RGB*, *CMYK*, vagy az intenzitás alapú bontást. Nekem az intenzitás alapú bontás tűnt a legszimpatikusabbnak. Választhatunk a különböző képpont formák közül is, létezik kör, vonal, négyzet, rombusz. A kézikönyv szerint egy jól kivitelezett színre bontó eljárás esetén nincs szükség élsimításra, ez azért kerül be mégis a szűrőbe, hogy csökkenteni tudjuk a színmélységet.



5. ábra Az újság szűrő beállításai

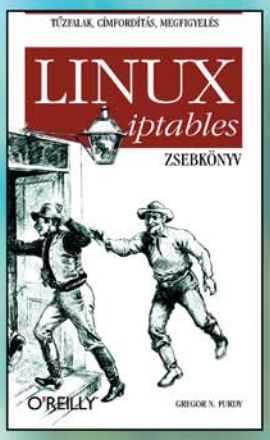
Videó

Említést érdemel még az előbbihez ötletben hasonló *Videó (Video)* szűrő. Ezzel ellenőrizhetjük, hogy miként nézne ki a képünk egy rossz minőségű tévén, csíkosan, pixelesen. Egyszóval a múlt század hatvanas éveinek technikájával.

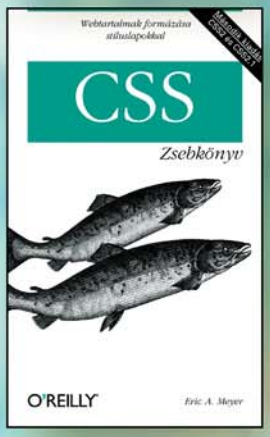
Juhász Attila
(rabszolga@goraffe.hu)

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.

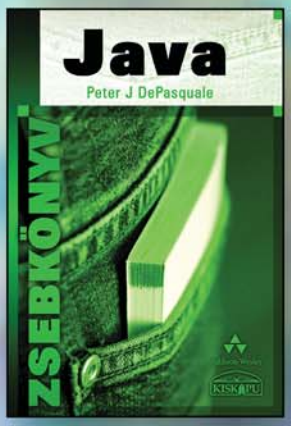
Tudás a zsebben



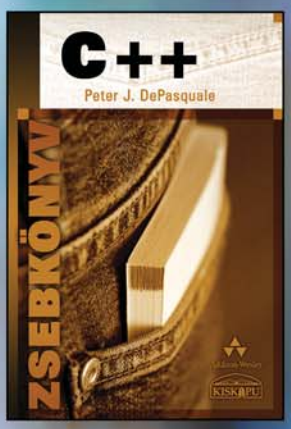
Napjainkban a hálózati biztonság egyike a rendszergazdák számára legfontosabb területeknek. A Linux Netfilter rendszere – vagy ahogy a kezelésére szolgáló parancs után gyakran hívják az iptables – számos dologra képes ezzel kapcsolatban, csak tudnunk kell használni. Ez a zsebkönyv egyrészt általános áttekintést ad a Netfilter rendszer felépítéséről és működéséről, másrészt szerepel benne az összes olyan szintaktikai elem és paraméter, amelyek beállításával a lehető legnagyobb biztonságot érhetjük el.



A HTML kezdeti zűrzavaros fejlődése után az igazi kibontakozást az az egyszerű felismerés hozta meg, hogy célszerű a tartalmat és a megjelenés leírását teljesen különválasztani. Így születtek meg a stíluslapok, illetve a CSS, amivel egészen részletesen szabhatjuk meg, hogy a weblapok egyes elemei hogyan jelenjenek meg a képernyőn. A részletességnek persze ára van: rengeteg paraméter nevét kellene fejben tartanunk, ami a legtöbbünknek természetesen nem nagyon megy. Ez az apró könyv a feledékeny webfejlesztőknek készült.



Bár az objektumközpontú programfejlesztés során a valósághoz sokkal jobban igazodó gondolkodásmódot követhetünk, ez a „logikai kényelem” azonban az ilyen nyelvek összetettségének drámai növekedésével is együtt jár. Ez alól természetesen a Java sem kivétel: számos kulcsszót és felületet kell ismernünk ahhoz, hogy hatékonyan tudjuk használni a nyelv szolgáltatásait. Ez a zsebkönyv tartalmazza a legfontosabb kulcsszavak leírását, rengeteg szintaktikai példát, illetve programozási tippet.



A C++ meglehetősen összetett programozási nyelv. Az alapelvek megértése után ugyan gyorsan tanulható, de elődjével, a C-vel szemben rengeteg kulcsszót, szintaktikai elemet és gyakran használt szerkezetet kell a kezdő fejlesztőnek fejben tartania. Így aztán gyakori, hogy bár az ember pontosan tudja, mit akar megvalósítani, keresgélnie kell az ehhez szükséges nyelvi elemeket. Ezen igyekszik segíteni ez az alig 100 oldalas könyvecske.

Részletes információ és letölthető mintaoldalak:

www.kiskapukiado.hu

KISKAPU
www.kiskapu.hu

A Pixel képszerkesztő

Hamarosan a Linux alatt dolgozó grafikusokra és fotósokra is boldog idők köszöntenek. Egy kiváló minőségű és tudású pixelgrafikus program tűnt fel, mely hamarosan sokunk eszköztárát gazdagíthatja.

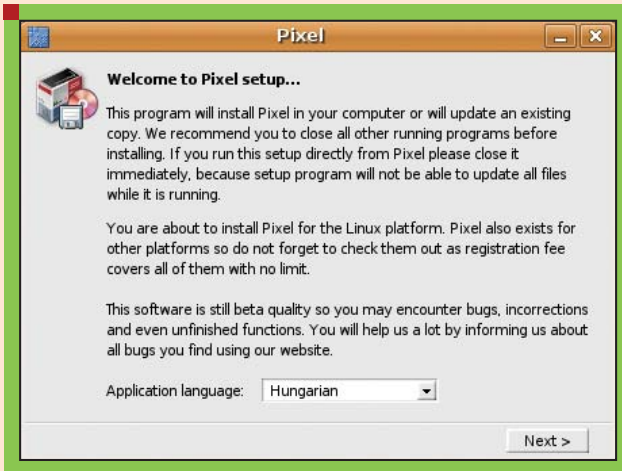
A *Pixel* nevéhez méltón, egy pixelgrafikus program, melyet több tucat operációs rendszeren tudunk használni. Futtathatjuk például *Windows*, *Linux*, *MacOSX*, *BeOS*, *Zeta*, *QNX*, *MorphOS*, *FreeBSD*, *eComStation*, *OS/2*, *SkyOS* és a jó öreg *DOS* alatt is. Meggyőző nem? Tudásában az *Adobe Photoshop* és a *Gimp* ellenfeleként tarthatjuk számon. Az utóbbit számos területen túl is lépi a mindössze 13 MByteos program.

Vajon hány programozó dolgozhat rajta? Legnagyobb meglepetésemre kiderült, hogy a *Pixelt* mindössze egy ember, *Pavel Kanzelsberger* fejleszti. Nyolc éve írta meg az első verziót, melyet ő és barátai használtak akkor még *DOS* alatt. Az idők során egyre többen kezdték el használni és fokozatosan érkeztek a megkeresések, melyben kérték, hogy portolja a programot valamelyik újabb operációs rendszer alá. Végül az idei év elején *Kanzelsberger* úgy döntött, hogy otthagyja munkahelyét és teljes munkaidőben a *Pixel* befejezésén fog dolgozni. A program meglepően sok operációs rendszert támogat. Vajon hogyan valósította meg mindezt *Kanzelsberger*? Mivel szerette volna leegyszerűsíteni a portolás nehézségeit, egy kicsi *SDL* alapú függvény könyvtárat írt, melynek neve *eLiquid*. Ez teszi lehetővé a *Pixel* gond nélküli futtatását több tucat operációs rendszeren, miközben a program kezelése és külalakja mindenhol ugyanaz marad.

Ma a felhasználók fele a *Windows* felhasználók táborából kerül ki, a stabil második helyet pedig a *linux* alatt dolgozók foglalják el.

A telepítés

Most már sokat tudunk. Nézzük tehát, mit tud a program! Kezdjük a telepítéssel! A béta verziót megtalálhatjuk és letölthetjük a ➲ http://www.kanzelsberger.com/pixel/ oldalról. A jobb oldali menüből válasszuk ki a *Download* vagyis letöltés menüpontot. Amint ez megvan, kezdődhet a csodálkozás. A *Pixel* tényleg rengeteg platformra elérhető! Válasszuk ki a nekünk megfelelőt. A dolog kicsi figyelmet kíván, hiszen van *PowerPC* és *Intel* processzorra fordított változat is. A *Debian* alapú *Linux* disztribúciót használóknak pedig van *deb* csomag is a választható lehetőségek sorában. Nekem gondjaim akadtam *Ubuntu Dapper* alatt a *deb* csomaggal, így a telepítővel ellátott verziót választottam. Maga a telepítés egyszerű, mindössze az lephet meg minket, hogy a magyar nyelv is a támogatott nyelvek között van. A web oldalon található összefoglaló azt írja, hogy mivel a program teljes *Unicode* támogatással bír, így gyakorlatilag bármelyik nyelvre könnyen lefordítható.

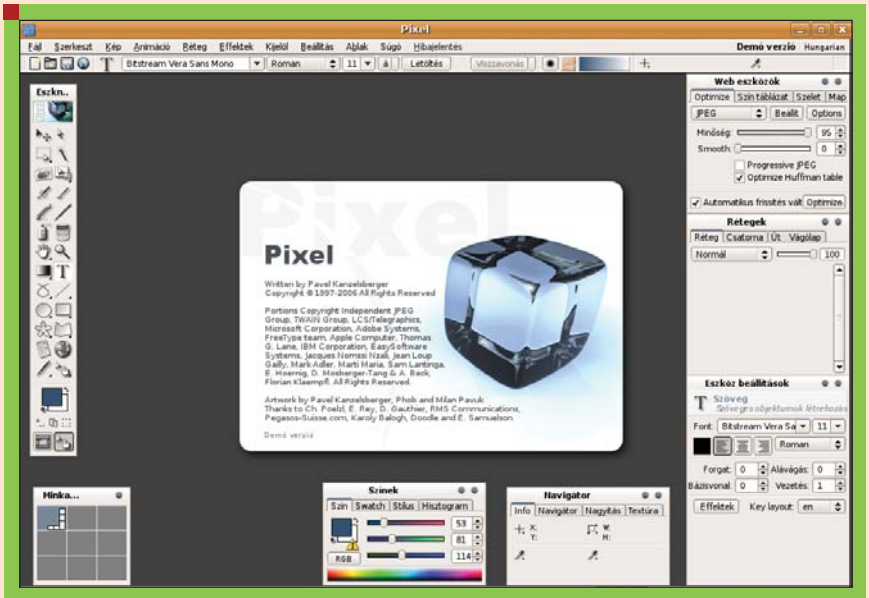


1. ábra A program a magyar nyelvet is támogatja

Első benyomások

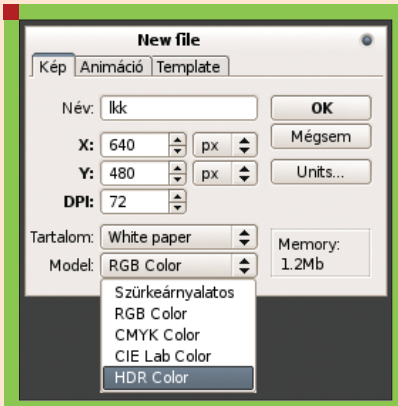
A program elindítása után egy hallatlanul ismerős felület fogadott. Mintha valahol már találkoztam volna már vele. A helyzet nem meglepő, hiszen a program az *Adobe Photoshop* által kialakított konvenciókat követi. Habár a menük nevei és elhelyezésük eltérhet, mégis már az első indítás után otthon érezhetjük magunkat.

A fenti menüsor és az alatta található eszköz beállító panel már a legújabb *Photoshop* verziók által kitaposott utat követi. Az eszközöket is könnyen megtalálhatjuk, hiszen a jól megszokott helyet, a bal oldalt foglalják el. A párbeszéd ablakok ugyancsak ismerősek már. Van pár apró meglepetés is, ilyen például a jobbra fent látható web eszköztár, mellyel a kép minőségét befolyásolhatjuk a mentés során. Itt választhatjuk ki a formátumot, a tömörítési módot is. Jobbra alul az aktuális eszköz mindenre kiterjedő beállítását végezhetjük el.



2. ábra A program felülete

3. ábra A munkaterület váltó



4. ábra Az új fájl párbeszédablakban láthatjuk a támogatott színmodelleket

Végezetül pedig vegyük szemügyre a bal alsó sarokban elbújó piciny párbeszédablakot! Ez az ami a *Photoshopból* már rég óta nagyon hiányzott. Segítségével virtuális munkaasztalokat használhatunk a *Pixelen* belül! Egyszerűen zseniális! Személy szerint sokat várok ettől a lehetőségtől, habár mint az illusztrációból is látszik, sikerült elírni a fordítás során az eszköz nevét.

Használat közben

Lássunk munkához, hozzunk létre egy új fájlt! Meglepetésemre a *Ctrl+N* billentyűkombináció hatására máris megnyílt a párbeszédablak. Kicsit utána nézve kiderült, hogy a billentyűkombinációk nagy része is *Photoshop* kompatibilis.
Egyből észrevehetjük, hogy a *Pixel* számos színmodellel tud dolgozni. Az *RGB*, szürkeárnyalatos, *CMYK*, *CIE* mellett még *HDR*, azaz nagy dinamikatartományú képeket is szerkeszthetünk vele. Ez egy fontos sarokpontja lehet a program térhódításának, hiszen a *CMYK* a grafikusok, a *HDR* támogatás pedig a fotósok számára igen fontos.
A másik dolog ami egyből feltűnhet, hogy a program gyors! Nincs reakció ideje. Mindig lehet tudni mi történik, és még a sok számítást igénylő szűrők hatását is pillanatok alatt láthatjuk.
Észrevehetjük, hogy bizonyos esetekben, például *HDR* színmodellt használva a néhány menüpont nem elérhető. Ezek a funkciók még hiányoznak. Az ígéret szerint az 1.0-ás verzió megjelenéséig mindegyiket befejezi a fejlesztő.

Lássuk milyen lehetőségeink vannak még! A *Pixelből* nyomtathatunk és szkennelhetünk is. Kezelhetünk több tucat képformátumot. Használhatunk rétegeket és utakat (path). Az 1.0-ás verzió pedig kezelni fogja a *PSD* (a *Photoshop* formátuma) kiterjesztésű fájlokat is, használni tudja majd a *Photoshop* plugineket és végül de nem utolsó sorban a vektoros rétegek teljes *SVG* támogatással lesznek kiegészítve.

Összbenyomás

Összességében elmondhatjuk, hogy egy könnyen és széleskörűen használható program a *Pixel*. Személy szerint mindig örülök, ha újabb grafikus program jelenik meg *Linuxra*. *Photoshophoz* való hasonlósága és professzionális felépítése miatt sok *Linuxra* áttérni kívánó grafikusnak, webgrafikusnak, fotósnak nyújthat használható alternatívát.
A végleges 1.0-ás verzió szeptember közepétől lesz elérhető, addig béta verziókat vehetjük mindössze kézbe. Sajnos a program sem nem nyílt forrású, sem nem ingyenes.
A teljes verzióért 35 dollárt kell fizetnünk, amiért teljes támogatást kapunk egészen a 2.0-ás sorozat megjelenéséig. Ha valakinek komolyan felkeltette az érdeklődését a program, siessen, hiszen a végleges verzió megjelenésével az ár 75 dollárra fog emelkedni.
Jó hír, hogy bár a *Pixel* forráskódja zárt, a fejlesztése közben használt függvénykönyvtár, az *eLiquid* hamarosan nyílttá válik a fejlesztő ígérete szerint, segítve ez által a portolható alkalmazásokat fejlesztőket.

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.

Egy régi hálózati szimulációs játék felélesztése

Az Ultimate Boot CD-ről többféle szabad operációs rendszert be lehet tölteni, és akár IPX hálózatot is létre tudunk vele hozni. Ezzel pedig feléleszthetők régi, de igen élvezetes játékok, többek között a Netwars...

T.I.M és Netwars

Ha még nem lenne ott a kedves olvasó polcán vagy merevlemezén ez az alapmű, akkor az ***Ultimate Boot CD*** (azaz ***UBCD***) letölthető a ➲ ***www.ultimatebootcd.com*** oldalról. Ehhez a példához most elegendő lenne az alapverzió is, de a ***CD*** igazi szépségét az ***INSERT Linux*** disztribúció adja – érdemes hát azzal együtt tartani. Alapvetően nem hiszem, hogy érdemes túl sok időt számítógépes játékokkal tölteni. Kivételt azonban mégis tehetünk: a logikai és a szimulációs játékok talán nem rombolnak túl sokat az ember személyiségén, hanem inkább fejlesztik gondolkodását, ügyességét, esetleg közösségi érzékét.

Az előbbire jó példa a klasszikus determinisztikus világképet modellező ***T.I.M***, (***The Incredible Machine***), melyben egy-két labda útjába kell mindenféle tereptárgyat helyezni (futószalag, dobbantó, lejtő, akvárium, macska stb.), míg el nem jut szándékolt céljába az egy-két leejtett bőrgolyó. Kipróbáltam az ***UBCD***-vel, remekül megy. Utóbbira pedig tekintsük példaként a ***Netwars***-t. A maga egyszerűségében fantasztikus játék – térbeli mozgással, három dimenziót leképező színes radarral, megvédendő és elpusztítandó tárgyakkal, egyéb huncutságokkal. Maga a név a ***Starwars***-ra (***Csillagok Háborúja***) és a ***Netware***-re is utal, hiszen ez a név kapcsolódik az ***IPX*** alapú hálózathoz, ami alatt igazán jó játszani ezt a játékot. Azonban az elmúlt években sehol nem botlottam már ***IPX*** alapú hálózatba, ami pedig alapfeltétel.

A ➲ ***netwars.xnet5.com*** weboldalon (ahonnan le is lehet tölteni az ***Advanced Netwars***-t) szó esik arról, hogy ***DosBox***-ban futtatva ***IPX-Wrapper***rel lehet ezt a játékot hálózatban játszani, de számomra lassúnak, szakadozottnak tűnt ez a megoldás.

1. ábra Netwars

Lássuk a játékot – kevés előkészület, több kézimunka

Az ***UBCD*** betöltőlemezei között akad egy olyan (***F6***, ***F4*** gombbal kiválasztva – ***MSRRC: Bart's N/W Disc Clone***), amelyik megoldja ezt a problémát. Ehhez a játékhoz csak az ezt képviselő képfájlra lesz szükségünk: az ***images/fdnwd288.igz***-re. Ezt úgy nyerhetjük ki, hogy a ***loop*** eszközön keresztül befűzzük az ***UBCD .iso*** fájlját (lásd lentebb), majd belépünk az ***images*** könyvtárba, és kimásoljuk munkaterületünkre ezt a fájlt (amit aztán célszerű írásra is alkalmassá tenni egy `chmod u+w` paranccsal). Ennek segítségével akár olyan CD-t is lehet készíteni, amin csak ez a betöltőlemez van (kicsit átalakítva), és mindenféle extra közreműködés nélkül be lehet róla tölteni a ***Netwars***-t. De ha valaki nem akar ezzel időt tölteni, akkor egy natúr ***UBCD*** is megteszi (no meg a szükséges ***Netwars.exe*** fájl valamilyen adathordozón).

Vigyázni kell egy ponton, hogy a „***Select profile***" kérdésre ne az alapértelmezett ***IP***-t válasszuk, hanem az ***IPX***-et. Intelligens módon úgy van megoldva a konfiguráció, hogy néhány másodpercet vár minden kérdéses esetben (látjuk, ahogy visszaszámlál, míg hozzá nem nyúlunk), de aztán továbblép a betöltés az alapértelmezett értékekkel (ami az ***IP*** alapú hálózat).

Egy ***DHCP***-vel működő hálózatban (!) is létre lehet hozni e rendszerrel az ***IPX*** alapú kapcsolatot, de úgy is, hogy csak két gépet kötünk össze egy keresztbe kötött (***crosslink***) Ethernet kábellel. Ez utóbbi esetben figyelni kell a megfelelő

kérdésnél, mert a *DHCP* az alapértelmezés. Ha nem *DHCP*-vel oldjuk meg a címek kiosztását, akkor a megadandó két *IP*-címet (például 2.2.2.2 és 2.2.2.3) beírhatjuk a megfelelő rubrikába, valamint az egyik gépet átjárónak érdemes kinevezni, és az ő címét beírni *gateway*-nek. Megadandó még például a 0.0.0.0, mint alhálózati maszk („*subnet mask*") – ezt könnyű begépelni, de a rendszer ezt a számot 255.255.254.0-ként fogja kezelni. Üresen lehetett hagyni a névkiszolgálót és az egyéb adatokat.
Ezek után még felhasználói nevet és jelszót is megadhatunk, de ennek nincs jelentősége esetünkben (akár üresen is maradhat). Ha megkaptuk a parancssort, már indíthatjuk is a játékot.

2. ábra Egy régi játék a frissen gyártott IPX-hálózatban

A játék beállításai

Az igazi élmény a több játékossal való vállvetett küzdelem (*multi player*). Az irányítást (*Control*) célszerű billentyűzetre állítani. A *Graphics* menüponttal az igazán gyenge gépek is működőképessé tehetőek, sokszögek helyett csak vonalakat rajzolva. Az *Options* menüpontban érdemes kérni a *Shout-out* módot, ami összehangolja a több játékost, és addig nem enged senki elhaltat újra belépni, míg ki nem derül, ki a győztes. Valamint eldönthető, hogy az embereken kívül legyenek-e számítógép-vezérelt űrhajók (akiknek ilyen neveik vannak, mint *Wolf, Saber*...). Kezdők esetén jobb az, ha nincsenek ellenségek, jobban meg lehet mutatni a billentyűk hatását. (*Disable computer ship*). De ha vannak is, akkor is eldönthető, hogy az emberek együtt küzdenek-e a gépi járművek ellen, vagy mindenki mindenki ellen.
A billentyűk megismerése kulcsfontosságú, főleg amiatt, hogy idejében gázt adjunk a *SHIFT*-tel, mert egyébként pillanatok alatt szétlőnek a számítógép-vezérelte hajók (ha vannak). *CTRL:* a fék (majd a hátramenet). Kurzorgombok viszik a tér négy irányába a hajót (kár, hogy ez nem állítható, mert a repülőszimulátoroknál éppen a lefelé nyíl nyomja le a repülő farkát, azaz emeli meg a repülőt, itt pedig épp fordítva). Szóközzel lövések dördülnek, de az igazán hatékony követőrakétákat a *CapsLock* indítja. A *tab* teszi lehetővé a turbó gyorsítást, „A" betűvel pedig láthatatlanná tevő köpeny (cloak) borul ránk. *F1*-gyel nézőpontot válthatunk, *F2*-vel szüneteltethetjük a játékot (ha magunkban játszunk). Annyiban logikus ez a billentyűkiosztás, hogy nem kell messzire nyúlni a funkciókért. Sőt, a másik kéz környékén, „szimmetrikusan" is definiálva vannak ugyanezek a billentyűk, ha valakit nem zavar a kurzorgombok közelsége (*Enter, Backspace,]* (azaz magyar billentyűzeten *„ú"*), és a jobboldali *SHIFT* és *CTRL* is működik). Ami nincs benne a leírásban (talán azért, mert semmi „haszna" nincs): lehetséges a tengelye körül is forgatni az űrhajónkat az x és y gombokkal.

Egy játéklemez kialakításának elve

Ha valaki egy kis időt szán arra, hogy magától betöltődhessen a játék mindenféle extra közreműködés nélkül, akkor erre is van mód (pontosabban csak *DHCP*-vezérelt hálózatban juthatunk el a célhoz „érintés nélkül" – statikus *IP*-címek esetén figyelni kell a megfelelő ablak megjelenését, és a leírt módon be kell állítani az értékeket). A fent említett *images/fdnwd288.igz* fájlt át kell neveznünk *fdnwd288.img.gz*-vé, ezt már *gunzip*-pel ki lehet tömöríteni. Nevezzük át az eredményül kapott *fdnwd288.img* fájlt valami más névre, például *fdnwd289.img*-vé, hogy ha netán vissza szeretnénk másolni testreszabott képmásfájlunkat az *UBCD*-re, akkor ne írja felül az eredetit. Visszacsatoló eszközön keresztül lehet becsatolni a képmásfájlt egy előre elkészített (mondjuk *k* nevű) könyvtárba:

```
mkdir k; su -c 'mount -o loop
➥fdnwd289.img k'
```

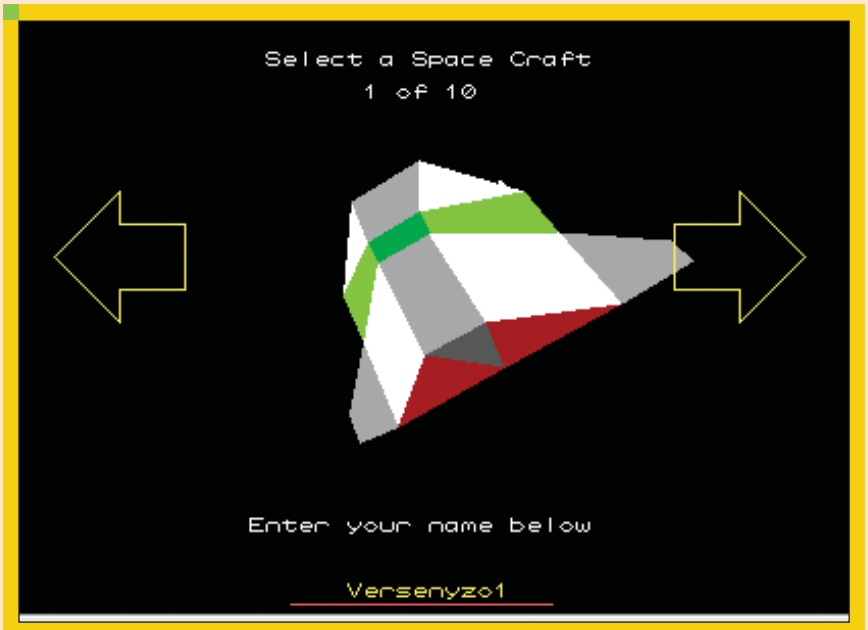
Csodák csodája, belépve a *k* könyvtárba, épkézláb fájlokat látunk, amiket ráadásul szerkeszteni is lehet (hogy miként érdemes, erről a következő egységben lesz szó). Lecsatolva a képmásfájlt, minden változás megmarad. Kiírható (`dd if=fdnwd289.img of=/dev/fd0`) egy floppyra, és onnan – minden egyéb operációs rendszer nélkül – betölthető futtatható, vagy pedig egy

```
mkdir l; cp fdnwd289.img
➥l;mkisofs -b fdnwd289.img -o
➥kimeneti.iso l
```

paranccsal előáll egy olyan *.iso* fájl, ami CD-ről tölthető (`cdrecord -blank=fast kimeneti.iso`). Esetleg *gzip*-pel tömörítve és *.igz* végűvé átnevezve „visszatehető" az *UBCD*-re, nyilván megfelelő menüponttal ellátva a *boot/menus/custom.scn* fájlt.

A netwars játéklemez kialakításának konkrét lépései

A következőket tettem a felcsatolt *fdnwd289.img* képmásfájllal, mielőtt előállítottam volna belőle a játéklemezt:
A */etc/profile*-ban csak az *ipx.pro* fájlt hagytam meg, a többit töröltem. Ennek nemcsak helytakarékosság volt az oka, hanem az is, hogy a *„Select profile"* menü ezen fájlok alapján ajánlja fel a választási lehetőségeket, és nekünk csak az *IPX* kell. Helytakarékossági okból gyomláltam ki a hálózati kártyák meghajtói közül a *lib/odi16* összes fájlját, hiszen mi 32 bites klienskent fogunk kapcsolatot létrehozni (bátrabbak a *lib/odi32*-ből is



3. ábra Űrhajó választása

4. ábra Várakozás a többi versenyzőre

irthatnak olyanokat,amire biztosan nincs szükség, én azonban ezt nem ajánlom – hosszasan kísérleteztem vele, de mindig kisült, hogy túl sokat töröltem e fájlok közül). Töröltem még a *level1* könyvtárból a *vc.cab* és az *ntfs.cab* fájlt, hogy maradjon elég hely az átalakítandó *level3/nwlan.cab* fájlnak. Sajnálatos módon ugyanis csak olyan *.cab* előállítót találtam Linuxra (*Linux CABinet file creator: lcab*; ➲ *www.geekshop.be/rien/lcab* ➲ *www.usinglinux.org/archivers*), amelyik nem tömörít, így – mivel kénytelen voltam belenyúlni ebbe az archívumba – a végeredmény sokkal helyigényesebb lett, mint az eredeti. Még arra maradt hely, hogy létrehozzak a gyökérben egy *netwars* könyvtárat, és ebbe elhelyezzem a *netwars.exé*-t és a *readme.nw*-t. Felmerül a kérdés, hogy miféle helyigényről beszélek én itt, hát van a merevlemezen bőven hely... De egy *loop* eszközön keresztül becsatolt betölthető floppylemeznél az eredeti méretbe kell bezsúfolni mindent. (El tudom képzelni, hogy „nagyítható” ez a fájl, vagy belemásolható a képmásfájl egy eleve nagyobbra készített fájlba, de nem sikerült rájönnöm, hogyan. Örömmel veszem azok visszajelzését, akik tudják a módját.)
Még néhány változtatást eszközöltem a */etc/global.set*-ben: `p_user=!`, `g_timeout=1` értékadásokkal, vagyis, hogy ne várjon felhasználónév-megadást az *NwLan*, és ne kelljen olyan sokat várni a visszaszámláló ablakokra (bár, érdekes módon, ez nem minden visszaszámlálásnál érvényesült).
Még adós vagyok azzal a kérdéssel, hogy miért kellett hozzápiszkálni az *nwlan.cab*-hoz, s hogy miként automatizálhatjuk a *netwars* elindítását. E két legyet egy csapásra intézzük el: Sajnos hiába tettem az *autoexec.bat* fájl végére a *netwars*-indítást, muszáj voltam *cabextract*-tal kitömöríteni *nwlan.cab autorun.bat* fájlját kiegészíteni egy `@call a:\netwars\netwars.exe` paranccsal. A visszatömörítést az *lcabperl*-lel végeztem, és az így kapott (meglehetősen nagy) *nwlan.cab*-ot másoltam vissza az eredeti helyére.
Ha valaki nem szeretné mindezeket végigcsinálni, de kíváncsi az eredményre, megtalálhatja a *Linuxvilág* honlapján, vagy itt:
➲ *www.osb.hu/z/nw.iso*.
Megvallom, hogy e játék tekintetében először az *UBCD*-n levő *OpenDOS*-szal kísérleteztem, hiszen ebben a gyűjteményben láttam életemben először a *Netwars*-t, és igen szépen dokumentált rendszer élt emlékeimben; de valahogy nem sikerült elindítanom. Talán egy következő *UBCD*-kiadásban kevésbé gépfüggő profillal jelenik meg.

Játék memóriából

Sikerült tehát elindítani ezt a kedves jószágot, és valóban látták egymást a légi járművek. Elegendő egyetlen *UBCD* vagy házilag gyártott cél *CD* is a hálózati játékhoz – elindulása után már át lehet tenni a másik gépbe – memóriából is tud futni. Mindenkinek tudom ajánlani – egészen gyenge gépen is jó szórakozást nyújt, és a közös játék mindig jobb, mint a magányos függővé válás. A teljes igazság persze az, hogy a *Linux* megismerése és felderítése a legjobb repülőszimulátornál is izgalmasabb számítógépes játék.

Szabó Zoltán
(szz@freemail.hu)
Négy gyermekével és feleségével Pannonhalmán él. Tíz éve kísérletezik a Linuxszal. Matematikát és informatikát tanít, diákotthonban keseríti a rábízottak életét. Szívügye a PHP, a PostgreSQL és a Moodle.


Linuxvilág
www.linuxvilag.hu

Gorky 17

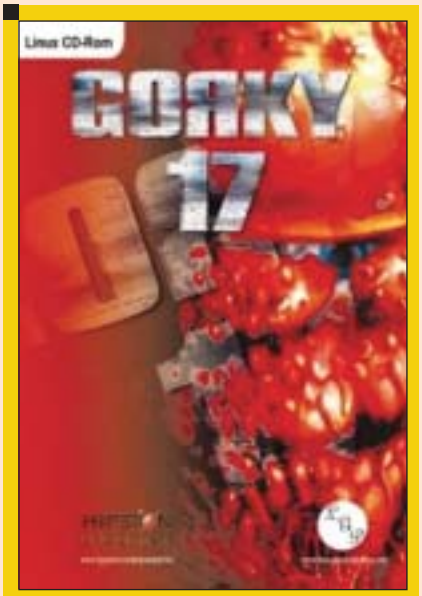
Néhány hónappal ezelőtt népszerű, Linuxon (is) futtatható valós idejű stratégiai játékokkal foglalkoztunk. Most a kategória egyik közeli rokonáról, pontosabban a körökre osztott műfaj egyik kiemelkedő képviselőjéről lesz szó.

A programról

A lengyel illetőségű *Metropolis SH* fejlesztette *Gorky 17* csaknem nyolc éve, 1999 nyarán látta meg a napvilágot. Az addig ismeretlen csapat munkája meglepően kedvező fogadtatásban részesült világszerte: a játékprogram erőteljes hangulati elemeket, változatos taktikai modelleket vonultatott fel egy groteszk és eltorzult világban. A rövid kerettörténet így szól: a *NATO* hírszerzése információkat szerzett a *Gorky 17* névre keresztelt orosz katonai város lerombolásáról, ahol korábban teleportációval kapcsolatos kísérleteket végzett a helyi vezetés. A híreket követően rátaláltak egy elhagyott bázisra, a romos épületek között pedig egy speciális kutató laboratórium maradványaira. Néhány nappal később a kiküldött felderítő csapatnak nyoma veszett, miközben ellepték az utcákat mindenféle torz és rémisztő lények. Végül a nemzeti erők lezárták a bázis teljes területét, és egy három fős, különlegesen képzett egységet küldtek be a kialakult helyzet feltárására. Nos, a bevezető itt véget ér: mint az sejthető, az alakulatot nekünk kell irányítanunk, rengeteg összecsapást és tartalmi fordulatot megélve.

A részletek...

A program egészen 2006-ig csupán *Win32* felületen futott, mivel *Linuxra* nem volt átírva, ezen felül a *Wine* és *Cedega wrapperek* sajnos ki nem állhatták a windowsos lemezt. 2004 derekán röppentek fel először azok a hivatalos hírek, melyek szerint a *Hyperion* csapata már dolgozik a (kereskedelmi jellegű) linuxos verzión, de a munkálatok igencsak lassan haladtak. Hosszú várakozás után, a 2006-os esztendő végre meghozta a várva várt natív változatot. A *Hyperion* hírnevéhez hűen, ez a projekt is kellően profi munkáról tesz tanúbizonyságot, hiszen nem különböztethető meg az eredetijétől, miközben a *DirectX* alapokon nyugvó rétege teljesen ki lett iktatva: immár *SDL (v1.2.9)/OpenGL* programozási felületekre



1. ábra A nekünk készített natív verzió doboza

2. ábra A linuxos Gorky 17, KDE asztalon

3. ábra A terep felderítése

támaszkodik a játék motorjának egésze. Az elkészült művet a ***Linux Game Publishing*** terjeszti, így a ***CD*** on-line rendelhető meg a ➲ ***http://gorky17.linuxgamepublishing.com*** oldalon. A terjesztés apró szépséghibája, hogy amíg a ***Win32*** kiadás nagyjából 1000 Forintért szerezhető be a hazai multiáruházakban, addig a számunkra értékes változat rendelési alapdíja 25 Font (a cikk írásakor ez közel 8000 Forintot jelent). Utóbbi összeg külföldön ugyan nem számít soknak, azonban az említett, közel nyolcszoros árkülönbség az itthoni viszonyok között már tetten érhető. Ennek ellenére, a stratégiai játékok hazai rajongóinak is érdemes megrendelnie a linuxos ***Gorky 17***-et, hiszen „ereklyének" sem utolsó darab...

Üzembe állítás

Miután megérkezett a telepítő diszk, fűzzük be a rendszerünkbe, majd a csatolási pontra lépve (***root*** jogkörrel) indítsuk el a gyökerében lévő ***setup.sh*** szkriptet! A letisztult, grafikus interfésszel ellátott telepítő használatáról felesleges szólnom. Miután a bő ***500 MByte*** helyigényű állományok az alapértelmezett helyükre kerültek (minek feltétele a szériaszám megadása is), a program egy ***x-terminálon,*** felhasználóként kiadott gorky17 paranccsal indítható. Figyelem! A játék menürendszere nem tér ki a teljes képernyős, illetve az ablakos működés kapcsolási lehetőségére. Emiatt a személyes mappánkban (első indításkor) létrejött, ***/home/$/.lgp/Gorky17*** nevű rejtett könyvtárban, a ***settings.ini*** állomány ***„fullscreen"*** bejegyzését kell megváltoztatnia annak, aki hozzám hasonlóan jobban bízik az ablakos futtatásban. A játék ***„Options"*** menüjének ***„Video"*** pontjában lehetséges a ***„Software"*** / ***„OpenGL"*** leképezési módok között váltani, utóbbi ***API*** természetesen működőképes ***GLX*** (illetve ***DRI***) kapoccsal ellátott ***3D*** grafikus hardvert igényel. A játék ugyan fixen, 640x480 képfelbontásban fut – ellenben biztosan állíthatom, hogy senki kedvét sem fogja elvenni a kissé pixelesnek ható (egyébként príma mozgó modellezéssel ellátott) grafika. Hangszolgáltatás ügyén kizárólag ***OSS*** leképezés támogatott, így akinek ***ALSA*** megoldás működteti hangkártyáját, annak egy ***ALSA-OSS*** átjárót mindenképpen be kell állítania a rendszerében. Apropó, hangok! A fentebb említett „honi" ***Win32*** verzió teljes mértékben honosítva van, így joggal merülhet fel igény a feliratok és hangok lecserélésére itt, az átírt játékban is. Nos, a feliratok cseréje természetesen megoldható, viszont a szinkronhangoktól óva intek mindenkit: köztudott, hogy ennyire félresikerült hazai megoldással ritkán találkozni a ***PC***-s játékok világában. Kissé szomorú vagyok, hogy ezt kell írnom, de aki játszott valaha

4. ábra Egy kellemetlen ellenfél, ivadékaival együtt

5. ábra A karabély távolról is hatásos fegyver

is a „magyaros” változattal, bizonyára tudja, miről beszélek: némelyik érzelemmentes és közönyös beszólás, egyéb félresikerült megoldás komikussá tenné az egyébként hangulatos belső világot.

A „puding próbája”

Miután elindítottuk a programot, annak menüjében találjuk magunkat. Az egyszerű és átlátható beállítási lehetőségeket értelmetlen lenne taglalnom, lépjünk be inkább a játékba. A hosszúnak és nehéznek várható küldetés két üzemmódra oszlik. Egyik mód értelme a felderítés, másiké a körökre osztott harc. A „mászkálós” felderítés ugyan kissé unalmas, de a végjátékhoz szükséges végigcsinálnunk: ezen a ponton sok hasznos és szükséges tárgyat lehet begyűjteni a város utcáiról, valamint a teljes történet is ezáltal bontakozik ki (a felderítő mód természetesen nincs körökre osztva). A lényeg a belső harcokon van: körönként megszabott lépésszámból kell a csapatunkat győzelemre vezetni, mutánsok és egyéb speciális erők ellenében.
Lássuk tehát a legfontosabb tudnivalóinkat! Az elfordulás, a tárgyak átadása ebben a játékban nem kerül akciópontba – de utóbbi csakúgy, mint a támadás (és a védekező állás felvétele) egyszer tehető meg körönként. A *Gorky 17* „sava-borsa” az ellenfeleink stílusában rejlik, akik kifejezetten komoly munkára késztetnek bárkit: szinte mindegyikük saját harci egyéniséggel bír, más jellegű támadások ellen védett. Emiatt némely csata előtt fontos elmenteni az aktuális állást, hiszen az adott ellenség immunitását csak akkor tudjuk megtapasztalni, ha már megtámadtuk – ekkor viszont már odaveszett egy lépésünk, és vele együtt talán a kezdeményezés is... Ha változó létszámú csapatunkból bárki elesik, az a küldetés sikertelenségét jelenti, így fontos elvárás az erős és hatékony karakterek „nevelése”. Ezért érdemes mindössze egy-két hatékony fegyver használatát „erőltetni”, javarészt ezek használatával küzdeni: idővel nagyon komoly tűzerőre lehet szert tenni egy fegyvernemhez kötődő jártasság által. Bizonyos elért pontszám után az aktuális karaktert meglátásunk szerint fejleszthetjük, öt tapasztalati pontot szétosztva annak fő jellemzői között.
Talán mondanom sem kell, hogy érdemes komolyan meggondolni, mely tulajdonság kapjon kitüntetett szerepet, mivel némelyik terepen akár négy „típusú” ellenséggel kell egyszerre megküzdenünk, melyek mindegyike máshogyan harcol...
A sikeres küldetésekhez (véleményem szerint) mindenképpen szükséges két olyan karakter, amely képes távolról is hatékonyan támadni az ellenfeleket, illetve gyógyítani az első vonalat, miközben a többi társunk (például lángszóróval vagy bombákkal) közelharcban keseríti meg a szörnyek életét. A legfőbb ellenség (melynek „személye” a történeti szál egyik kiszámítható meglepetése) még ilyen körülmények között is garantáltan komoly kihívást jelent majd a kitartó játékosoknak!

Csak elfogultan!

Aki úgy érezné újoncként, hogy szeretne bepillantást nyerni ebbe a nagyon hangulatos világba, annak javallott megpróbálkoznia a linuxos demóval, mely szintén elérhető a fentebb címen érhető el. Ennek telepítése, használata megegyezik a teljes változat ügyén említettekkel, jellege miatt azonban nem lehet benne játékállást menteni, és csak néhány pályát tartalmaz. Akár a demóról, akár a teljes programról legyen szó, talán azzal jellemezhetném leginkább, hogy amikor leülök elé, végig azt érzem, hogy „na még egy utolsó pálya, aztán tényleg kilépek”.
Persze, amikor az órámra nézek, gyakran előfordul, hogy lassan indulnom kell a munkahelyemre, mert az utolsó pályából túl sok „utolsó” pálya lett...
Tipikusan olyan program ez, amelyet egyszerűen lehetetlen megunni: annak ellenére viszem végig újra és újra, hogy a pályákat és a történetet évek óta ismerem (utoljára talán az *Incubation* nevű körökre osztott csoda fogott meg ennyire). Ha ehhez még hozzáteszem azt, hogy e virtuális világ megelégszik *64 MByte* központi memóriával és egy szerény teljesítményű, régebbi *x86* (vagy *PPC*) processzorral, akkor elismerően bólogatok: bravó *Metropolis*, bravó *Hyperion*!

Kovács Zsolt (kovi@linuxforum.hu)

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

Kicsi a bors...

Ebben a hónapban néhány olyan közkedvelt ügyességi játékról olvashattok, melyek méretükben, hardverigényükben mérhetetlenül szerények.

A rövid bevezető ne tévesszen meg senkit: az „apró” kategóriába jó néhány olyan projekt tartozik a világhálón, melyek képesek hosszú órákra lekötni bárkit, aki kikapcsolódásra vágyik. Hogyan lehetséges ez? Valójában egyszerű a recept: a sokszor dollármilliós fejlesztésű, több *CD*-s játékok grafikai és fizikai kivitelezésének ellenében itt elsődleges szempontként a játszhatóság dominál, jellemzően kiforrott ötletekre alapozva. Éppen emiatt, a kis méretű (általában ügyességi) játékok között sokszor találkozhatunk ***remake***, ***retro*** programokkal is. Mivel a stílus képviselői gyakran szabadon elérhető forráskóddal rendelkeznek, így az életképes próbálkozások könnyen üzembe állíthatóak ***Linux*** alatt.

1. ábra Íme, a Tower Toppler

Tower Toppler

Tower Toppler néven érhető el a „8 bites korszak” ***Nebulus*** című alkotásának újjászületett, ***PC***-s verziója. Eme oldalnézetű, ügyességi játéknak a céljaként két irányban mozgatható sárkányunkkal különböző kőtornyok tetejére kell feljutnunk. Mindez kifejezetten nehéz feladatnak mutatkozik, hiszen a mozgásunkra reagált terep elfordulása, eltűnő támaszok, csúszó lépcsők, liftek, pattogó labdák és „élő” ellenfeleink egyaránt ellenünk dolgoznak. Ha az említett cél nem tűnne kellőképpen kihívónak, megfejelve az említett nehézségeket minden egyes pálya teljesítésekor szűkösen szabott időkerettel kell gazdálkodnunk. A játék a szó valódi értelmében ügyességi jellegű, hiszen jó ütemű mozgással, ugrással és lövéssel tudunk továbbjutni a következő tornyokhoz. Két teljesített pálya között, amolyan bónuszként, víz alatti terepen (tengeralattjárót irányítva) is növelhetjük pontjaink számát. A program alapötlete matuzsálemi korú: személy szerint már ***Enterprise128*** géppel is játszottam a ***Nebulus***-szal, majd később ugyanez a kihívás ***Amiga500*** előtt is elbűvölt. Mindezt azért említem hangsúlyosan, mert ennek tükrében bátran állítom: az eltelt húsz év ellenére a játék semmit sem veszített hangulatából. Sőt! Az igen időtálló elképzelés még ma is korosztálytól függetlenül ígér önfeledt szórakozást. A ***Tower Toppler*** gyakorlatilag mindent örökölt elődjétől, többek közt (sajnos) azt is, hogy nem lehet benne játékállást menteni. Így, a nehézségre való tekintettel néhány pályánként kap a játékos egy cheat kódot, melyet indítás után az ***Options/Game*** menüpontba kell gépelnie, ha a legutóbb teljesített pályákon szeretné kezdeni a „toronymászást”.

A projektben ezen felül elérhető egy egyszerűen használható pályaszerkesztő is. A programért látogassunk el a ➲ http://toppler.sourceforge.net címre! Töltsük le a szabadon elérhető ***toppler-verzió.tar.gz*** forrás archívot, majd bontsuk ki valahová. A kicsomagolt könyvtár tartalmát felépíteni és telepíteni rootként kiadott `./configure`, `make`, `make install` parancsokkal lehetséges (a forráskód függősége gyakorlatilag kimerül az ***X*** kiszolgáló és az ***SDL*** könyvtárak meglétében, így a legtöbb asztali rendszeren egyszerűen fordítható). Amennyiben befejeződött a művelet, felhasználóként terminálra gépelt `toppler` paranccsal léphetünk be ebbe a varázslatos programba. Első indításakor mindenképpen javallott végignézni a menüszerkezetét, ahol (többek közt) a grafika alpha simítását érdemes azonnal bekapcsolni.

2. ábra A Head over Heels linuxos verziója

Head over Heels

Nahát, ismét egy régi ismerős! A most említett ***Head over Heels*** nem meglepő módon szintén ***remake*** verzió.
A ***RetroSpec/Tomaz Kac*** csapata készíti az új kódot, mely közel húsz éves ötletre épül (és amely projekt már a ***Z80*** központi egységek által fémjelzett 8 bites hőskorban is nagy népszerűségnek örvendett). Ügyességi, mászkálós, kalandjáték hibridről van szó, melyet izometrikus nézetből irányíthatunk, mindezt kellemes ***retro*** zene kíséretében, szépen megrajzolt helyszíneken át. Célunk igen egyszerű: a feladatul kapott terep minél nagyobb területét kell bejárnunk a két rendelkezésre álló karakterrel, minden ellenfelet és apró fejtörőt magunk mögött hagyva.
Mivel a két „szereplő” külön szobából indul, így érdemes őket minél hamarabb egy helyre juttatnunk: egymás fejére állva minden olyan extrát (például nagyobb ugrás) ki tudunk használni, melyek csak ebben a helyzetben hívhatóak elő, nem mellékesen, egy idő múlva már csak ebben a formában, együttes erővel tudunk továbbjutni bármely szobából.
A projekt fő ügyességi jellegén túl, a játékos néha rá lesz utalva szürkeállományára is, mivel sokszor végig kell gondolni, miként tudunk az adott szobából tovább lépni életünk elvesztése nélkül. Le a kalappal a programozók előtt: ez a játék is kortól függetlenül ígér kellemes kikapcsolódást. Sajnos a forráskód ez esetben nem publikus, ennek ellenére gondoltak ránk, és a fejlesztők készítettek belőle egy natív linuxos verziót is. A bináris igen könnyen üzembe állítható:
a ➲ ***http://retrospec.sgn.net*** oldalról indulva le kell töltenünk a játékhoz tartozó ***hohlin-verzió.tar.bz2*** állományt, majd ezt kicsomagolva, a mappa gyökerében lévő ***install.sh*** szkriptet el kell indítanunk rootként. Miután a bináris az alapértelmezett helyére került (***/usr/local/share/games/HoH***), felhasználóként terminálra gépelt `hoh` paranccsal indítható a móka. A játékmenüt itt is érdemes végignézni, legfőképpen a vezérlőbillentyűk beállításának ügyén (ugrás, lövés, karakterek közötti váltás). Fontos megjegyeznem, hogy a ***Head over Heels*** lehetőséget ad az aktuális állás elmentésére, ezzel viszont érdemes csínján bánni, mivel „egy menetben” csupán kétszer tudjuk megtenni.

X-Moto

Talán akadnak közöttetek, akik emlékeznek az ***ElastoMania*** nevű hazai „csodára”. Arra a programra, mely anno a (***Win32*** hajtotta) ***PC***-ken szerzett magának nagy létszámú rajongói tábort, igen egyszerű ötletre építkezve. A következőről volt szó: oldalnézetből irányítva, triál motorunkkal kellett mindenféle nyaktörő mutatvány segítségével mihamarabb az aktuális pálya végpontjába érni, miután az utunkba (jó indulattal sem) eső tárgyakat összeszedtük. Hamarosan házi bajnokságok épültek a program köré, pedig a projekt alapvetően kereskedelmi jellegű volt, ingyenesen csupán a demóval lehetett szórakozni.
Nos, itt a „friss-ropogós” megvalósítás, avagy az ***X-Moto***. A szabad forráskód modelljének köszönhetően néhány hónap alatt harmincnál is több pálya kivitelezését, és gyakorlatilag hibamentes felépítést sikerült elérniük a fejlesztőknek. Érdekes, hogy a teljesen ***2D*** megvalósítás ellenére szükség van a játékhoz ***OpenGL***-t támogató ***3D*** grafikus eszközre is, mivel a képi effektek (füst, a motor villáinak torzulásai, stb.) az említett ***API***-n keresztül vannak megvalósítva. Az ***X-Moto*** életre hívásához látogassunk el a hivatalos, ➲ ***http://xmoto.sourceforge.net*** oldalra! Miután letöltöttük az aktuális

3. ábra Az X-Moto az ElastoManiát idézi

forrás archívot, csomagoljuk ki egy tetszőleges területre, majd a megszokott módon, rootként kiadott `./configure`, `make`, `make install` parancsokkal építsük fel a forráskódot. A függőségek ügyén kissé nehezebb (de nem megoldhatatlan) feladattal szembesülünk, mint az előbbiekben: a konfiguráló szkript az ***SDL*** könyvtárakon felül ***lua***, ***ode***, ***OpenGL*** komponensek után is kutakodik.
Amennyiben a program sikeresen lefordult, úgy felhasználóként terminálra gépelt `xmoto` paranccsal indítható.
A cikk írásakor fellelhető legfrissebb verzió a kezdetleges állapotot sugalló 0.1.13 azonosítót viseli, ennek ellenére a teljesen üzembiztos és kiválóan használható. Menürendszere csupán a legfontosabb paraméterek változtatására ad lehetőséget, így biztosan nem fog senki eltévedni a lehetőségek között.

4. ábra Csalóka látvány: a remek CircusLinux

CircusLinux

A ***CircusLinuxban*** az ***Atari*** egykori üdvöskéjének (***Atari2600***) egyik programja köszön vissza, linuxos ***remake*** verzióként. Valójában többről van itt szó, mint az ***Atariról***, hiszen nem is tudok olyan „tisztességes” 8 bites masinát mondani, amelyen a játék ötletét ne kivitelezték volna. A feladat egyszerű: két bohóc ül egy apró libikóka két oldalán, melynek megfelelő szárára esve az egyik bohóc a másikat fellövi a levegőbe. Célunk a játéktér felső részén szkrollozó lufikat kipukkasztani a felrepülő figurákkal. Tényekedésünk pontra megy, miközben életeink száma előre meg van határozva (a felső sor lufijait kigyűjtve kaphatunk plusz életet). Rendkívül aranyos és szórakoztató játékprogramról van szó, megcáfolva az első ránézésre törvényszerűen felmerülő „ez túlzottan puritán” gondolatot... A ***CircusLinux*** életre hívásához látogassunk el a ➲ ***http://www.newbreedsoftware.com/circus-linux*** oldalra, és töltsük le a szabadon elérhető ***circuslinux-verzió.tar.gz*** forrás archívot. A forráskód felépítése a tarball kicsomagolt tartalmában, rootként kiadott ***./configure***, `make`, `make install` parancsokkal történik.
A beállítószkript elsősorban az ***X*** kiszolgáló és az ***SDL*** könyvtárak irányában kutakodik, így a projekt minden asztali disztribúción könnyen üzembe állítható. Amennyiben minden bináris állomány a helyére került, a program felhasználóként terminálra gépelt `circuslinux`, illetve `circuslinux --fullscreen` paranccsal indítható. A paraméter nélküli indító ablakban hívja be a játékot, így érdemes az utóbbit erőltetni, mivel ablakban futtatva az egér sajnos lehúzható a játéktérről. Az eredménytábla kilenc típusra van osztva: a menük beállításánál három kapcsoló által kilenc játékmódot tudunk előállítani. Had áruljam el a legmeglepőbb dolgot: a játék említett elődje, a ***Circus Atari*** a cikk írásakor már erősen közelít huszonhatodik születésnapjához...

Zárszó

Akárhogy is, de ezek az „apróságok” komoly potenciált hordoznak magukban. Természetesen nem gondolom, hogy a népszerű kereskedelmi kódoknak bármiféle konkurenciát is jelenthetnek, ettől függetlenül nem hiába hódolnak tömegek ennek a „könnyedebb” műfajnak. Van, akit a múltidéző hangulat fog meg (és még nem is említettem a ***Krakout***, ***Invaders***, ***Pong***, ***Tron*** klónokat), vagy éppen azért telepíti bármelyiküket, mert szerény képességű géppel rendelkezik. Akinek gyermek van a családjában, valószínűleg ott is van helye ezeknek az „egyszerű” programoknak, tehát a felhasználók összességében nagy létszámú tábort alkotnak. Végül hadd kanyarodjak vissza a hardverigényre. Őszintén mondom, nem tudtam lemérni az említett projektek minimális igényét, pedig a cikk anyagait szándékosan egy ***650 MHz/x86*** processzorra épített, 128 MByte memóriával ellátott „kölcsöngépen” élesztettem fel, melynek grafikai és hangeszköze egyaránt szerény képességű darab volt (az ***X-Moto***-hoz azért kénytelen voltam elővenni egy erre a célra erősen túlméretezett GeForce kártyát). 2.5 GByte méretű merevlemezére „felhúztam” egy ***Slackware Linuxot***, és csupán a maradék néhány MByte területtel ga(rá)zdálkodhattam szabadon. Megtehettem, hiszen az említett játékok letöltött anyagai összesen sem igényeltek 50 MByte területnél többet, valamint kompromisszumoktól mentesen futottak még ezen a harmatgyenge vason is. Ezek után nem lepődnék meg, ha fele ekkora hardverrel is megelégednének. Roppant jól szórakoztam ezekkel a programokkal, hosszú órákig nem is ültem vissza a saját gépem elé. Ha végig gondolom, milyen jól esett „kiügyeskednem” magam, akkor csak elismerően bólogatok a műfajt illetően: valahogy így kell ezt csinálni.
Mindenkinek tartalmas időtöltést kívánok!

Kovács Zsolt (kovi@linuxforum.hu)

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