Hírek

Új HP nyomtatók

A *HP* jó pár új nyomtatóval bővítette palettáját, melyek közül kettőt emelnék ki. Mindkét nyomtató multifunkciós lézer.

Az egyik a *HP Laserjet M3035/3027 MFP* mely 35 oldal percenkénti sebesség mellett 1200 dpi-s felbontást ad akár Linux alatt is (forrás: ➲ www.linuxprinting.org), az áraik 1150 eurótól indulnak.

A másik nyomtató a 8 oldal percenkénti sebességgel bíró színes HP Laserjet CM1015/1017 MFP, mely akár nagyobb mértékű otthoni felhasználásra is megfizethető (Linuxos támogatottságáról még nincs információ), ára 549 eurótól indul.
➲ http://www.hp.hu

KVM átkapcsolók távoli eléréssel

ATEN – a *KVM* (*Keyboard-Video-Mouse*) átkapcsolók piacvezető cége - bejelentette legújabb a 8 portos *KN2108*-as és a 16 portos *KN2116* átkapcsolóját, mely három adminisztrátor számára ad konzolt, egynek lokálisan, a másik kettőnek pedig *IP* kapcsolaton. A kapcsolók webböngészőn keresztül érhetőek el távolról és 1024 bites *RSA*, 56 bites *DES*, 256 bites *AES* vagy 128 bites *SSL* titkosítást választhatunk. Az eszközök szinte minden bemeneti eszközzel megbirkóznak, legyen az *PS/2*, *Sun*, *USB* szabványú, vagy akár soros port, továbbá lehetősége van az adminisztrátornak egy ablakban figyelni 4, 9 vagy akár 16 szerver képernyőjét. Ennek azonban megvan az ára, a kisebbik 2100 euróba, míg a nagyobbik 2500 euróba kerül.
➲ http://www.alphasonic.hu

Három LCD egyben?

A *Sharp* egy érdekes *LCD*-vel állt elő. A koncepció lényege, hogy attól függően, honnan nézik a monitort, más és más – összesen három – különböző képet ad. Noha a *Sharp* elsősorban autóba vagy reklámcélra szánja, nem kizárt, hogy hamarosan rendszergazdák asztalán láthatjuk viszont például egy három-az-egyben monitorként (három kijelző egy helyen).
➲ http://www.newlaunches.com/archives/sharp_triple_view_display_the_best_accessory_for_your_car.php

Sharp, de nem Zaurus

CMP2000R néven *Koreában* piacra dobta a *Sharp* legújabb, színes kijelzős szótárgépét, mely ezen kívül rendelkezik 1,3 megapixeles kamerával, MMC/SD kártyákkal bővíthető és alapvető *PDA* funkciókat is ellát (*ebook*-olvasás, *MP3*- és rádióhallgatás). Az eszköz nem meglepő módon *Linuxot* futtat.
➲ http://www.engadget.com/2006/10/09/sharp-zaurus-cmp2000r-electronic-dictionary/

Négy dudás egy csárdában

Az *Intel* bemutatta legújabb négymagos processzorait. A mellékelt linken többféle (*PovRay*, *mencoder*, játékok, stb.) összehasonlító diagramot találhat a kedves olvasó.
➲ http://www.linuxhardware.org/article.pl?sid=06/11/02/142244&mode=thread

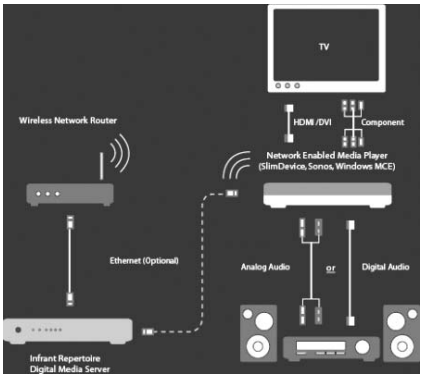
ATI grafikus processzorok kicsit másképp is használhatóak

A *Stanford Egyetem* kémiai kara érdekes és merőben új felhasználási módját találta ki az *ATI* videókártyáknak.
A videókártyák általában lebegőpontos műveletekre vannak felkészítve, éppen ezért nagyságrendekkel gyorsabbak bizonyos feladatokban, mint a számítógép főprocesszora.
A stanfordi csapat sikeresen portolta programjukat az *ATI R580*-as grafikus processzorára (ezt az *X1900*-as és *X1950*-es kártyák tartalmazzák), mely az eddigiekhez képest mintegy 20-40 szeres gyorsulást mutat.
A megoldás nagy mennyiségű lebegőpontos számokkal dolgozó feladatoknál (gyógyszerkutatás, titkosítás, szélcsatornás szimulációk, stb.) bizonyulhat úttörőnek.

➲ http://folding.stanford.edu/
➲ http://www.extremetech.com/article2/0,1697,2022643,00.asp

Linuxos médiaszerver – akár otthonra is

Az *Infrant* hamarosan szállítja *Linux*-alapú médiaszerverét. A *Repertoire* névre hallgató eszköz lejátszási képességekkel nem rendelkezik ugyan, azonban maximális kiépítésben 3 terabájtnyi tárhelyét szinte minden közismert szabvány és protokoll szerint képes kiajánlani. Az eszköz elsősorban a komolyabb felhasználókat célozza meg.



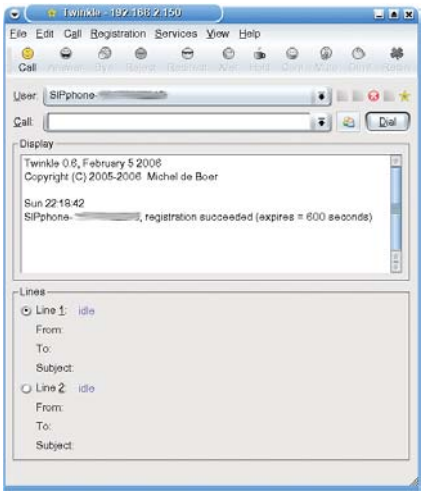
A rendszer lelke egy 280 MHz-es *RISC* processzor, ennek köszönhetően a rendszer *RAID* 0, 1 vagy akár 5 szerint is konfigurálható, az üres *PCI* slottal pedig tovább bővíthető az eszköz.

➲ http://www.linuxdevices.com/articles/AT3293796162.html

VoIP fejlesztések Linuxhoz

Két új *GPL* jogállású protokoll jelent meg a linuxos *VoIP* megoldásokhoz. Az egyik az *SRTP* (*Secure Real-Time Protocol*), a másik a *ZRTP*.

A *ZRTP Phil Zimmermann* fejlesztése, mely korábban a *Zfone* projekt részét képezte, és sokáig kérdéses volt, hogy *Phil* milyen jogállással adja majd ki. A *ZRTP* nagy előnye, hogy a kulcs sem a felhasználók, sem a tanúsítvány kezelők számára nem elérhető, kizárva így egy hátsó ajtó – és ezzel a lehallgatás – lehetőségét.



Ezzel egyidőben megjelent a *Twinkle* telefonprogram *Linux* alá (a *SUSE* disztribúcióban már benne van), mely *SIP* kompatibilis központokkal használható. A *Twinkle* a már korábban említett *SRTP* protokollt használja.

➲ http://www.linuxdevices.com/news/NS3777890434.html

A GIF formátum végre szabad

2006. október 1-től a *GIF* szabad formátum, noha addig is használták a webfejlesztők, amíg szabadalom védte, hiszen az alternatívájaként létrehozott *PNG* nem pótolta 100%-osan. A *Unisys* – a *GIF* szabadalmának tulajdonosa – azonban időközben belátta, ha perekkel próbálja érvényesíteni jogait, úgy a formátum hamar eltűnhet a süllyesztőben.

➲ http://www.freesoftwaremagazine.com/node/1772

Vékony kliens akár négy monitorral

Az *Igel Technology* bemutatta a hangtalan *PanaVeo* vékony kliensét, mely akár négy kijelzőt is képes meghajtani. Két kiépítésben vásárolható: a 7602-es két monitort képes meghajtani 2x 1920x1200 képpontos (digitális) vagy 2x 1920x1440 képpontos (analóg) felbontással, míg a nagyobb, 7604-es 4x 1600x1200 képpontos felbontást tud. Mindkét gépet 800 MHz-es *Celeron M* hajtja, az alapkiépítésben 256 megabájt memória 1 gigabájtig bővíthető, míg az alkalmazások (*Java*-s webböngésző, *mplayer*, *SAP GUI for Java*, *Igel Remote Management Suite*) 1 gigabájtos flash tárolón kapnak helyet.

➲ http://www.linuxdevices.com/news/NS5938810264.html

Linuxos D-Link telefon

Valószínűleg minden *Linux*-felhasználó szívét megdobogtatja a *D-Link* legújabb *V-Click* mobiltelefonja, mely a szokásos *GPRS* mellett *WiFi* eléréssel is bír. Magyarországon is használható lesz a telefon, hiszen 900/1800/1900 MHz-en képes hagyományos mobiltelefonként működni. 600 dolláros árához képest a belbecs elég sovány, hiszen csupán 24 megabájt memóriával gazdálkodhatunk és a kijelző is csak 176x220 képpont felbontású. Ami azonban határozottan mellette szól: beépített *SIP* kompatibilis *VoWiFi* (hangátvitel *WiFi*-n) is lesz a készülékben. 2007 első negyedévétől lesz elérhető a *D-Link* on-line áruházában.

➲ http://www.dlinkshop.com/

➲ http://www.linuxdevices.com/news/NS9525079329.html

Visual AJAX

A *Backbase* és a *BEA Systems* bejelentette, hogy megállapodást írt alá a *Visual AJAX Integrációs Projektben* való együttműködésről. Az *AJAX* egy interaktív webalkalmazások készítését támogató technológia. Segítségével a hagyományos *HTML*-hez képest kifinomultabb weboldalak hozhatóak létre. Legfontosabb előnye: képes a weboldalnak csupán a szükséges részét újratölteni, csökkentve ezáltal az adatforgalmat. A technológia fejlődése szükségessé tette, hogy a szabványosnak tekinthető *Eclipse* környezet segítségével is lehessen alkalmazásokat fejleszteni. Az integrációs projekt keretében az *Eclipse for Visual AJAX Development* terméket fejlesztik tovább a *Backbase AJAX* fejlesztői keretrendszerének segítségével.

➲ http://www.alerant.hu/bea-termekek/wl-workshop-bea-termekek.html

Nyílt forrású karakterfelismerés (OCR) a Google-al

A nyílt forrású karakterfelismerésen nagyot lendített a *Google Tesseract* projektje, mely jelenleg 1.02-es verziónál tart. Egyelőre csak angol szöveggel tud dolgozni, nem kezeli jelenleg az ékezeteket és a többhasábos dokumentumokat, de érdemes figyelemmel kísérni a projektet, hiszen ezen a téren még nincs túl sok nyílt forrású megoldás.

➲ http://sourceforge.net/projects/tesseract-ocr/

➲ http://www.linux.com/article.pl?sid=06/09/18/191251

Vezeték nélküli USB

Az *Intel* bemutatta a *Wireless USB*-t, mely a jelenlegi *WiFi* eszközök zavarása nélkül képes üzemelni. Az eszköz a jelenlegi vezetékes *USB*-t hivatott felváltani. A sebessége 3 méteren belül 480 megabit, 10 méteren belül 110 megabit.

➲ http://www.eetimes.com/showArticle.jhtml?articleID=193100143

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.


Linuxvilág
www.linuxvilag.hu

Mi újság a rendszermag fejlesztése körül

Az ***Areca RAID*** meghajtó várhatóan hamarosan bekerül a fő kernelfába. ***Andrew Morton*** és néhányan mások sikeresen dolgoztak együtt a fejlesztővel, ***Erich Chennel*** azon, hogy néhány kisebb, még megmaradt hibát kiküszöböljenek, ez a munka pedig láthatólag a végéhez közeledik. Volt ugyan egy kisebb kommunikációs zavar az első bejelentés idején, ugyanis a fő fejlesztők nem ismerték fel, hogy ***Erich*** valóban aktívan dolgozik a megtalált hibák kijavításán.
Jesse Huang, aki főállásban az ***IC Plus*** nevű cégnek dolgozik nemrég elkészítette az ***IP100A*** 10/100 megabites gyors hálózati csatoló meghajtóját, így ez bekerülhet a kernel forráskódjába. Volt azonban az üggyel kapcsolatban egy kisebb „cégháború". Ennek a kirobbanását az okozta, hogy ***Jesse*** kódjának alapját tulajdonképpen a ***Sundance*** meghajtó kódja képezte, olyannyira, hogy ahhoz képest igazából csak kisebb módosításokat tartalmaz. Amint pedig arra ***Arjan van de Ven*** bölcsen rámutatott, ilyenkor az a leghelyesebb eljárás, ha az ember kiegészíti a már meglevő meghajtót, nem pedig egy teljesen újat küld be jóváhagyásra. Erre a felvetésre válaszolva ***Jesse*** arra hivatkozott, hogy az ***IC Plus*** egyszerűn büszke rá, hogy a rendszermag forrásában van egy ***ip100a.c*** nevű fájl, sőt azt szeretnék, ha a ***sundance.c*** nevű idővel el is tűnne, és csak ez maradna. Aztán ***Jesse*** beszélt a munkáltatójával, és végül közösen arra az elhatározásra jutottak, hogy mégis csak jobb lesz a bevált módszert követni, vagyis a változtatásaikat a ***sundance.c***-be beleírni.
Al Bondi benyújtott egy olyan foltot, amely a ***ramfs*** helyett a ***tmpfs*** segítségével teszi lehetővé a ***RootFS*** és a csereterület (***swap***) összekapcsolását. Ez a módosítás lehetővé teszi, hogy a kifejezetten nagy ***initrd***-t vagy ***initramfs*** képet használó rendszereknél a lemezkép ne foglalja le ténylegesen a memóriát, csak ha ténylegesen használatban van. ***Al*** ötletére elsősorban a beágyazott rendszerek fejlesztői haraptak rá. Ugyanakkor ***H. Peter Anvin*** megjegyezte, hogy ezen a területen jelenleg a legfőbb cél az, hogy az ***initramfs*** inicializálása a bootfolyamatban korábbra kerüljön, ez ugyanis lehetővé teszi bizonyos beépített eszközök ***firmware***-jeinek betöltését. Nem volt benne ugyan biztos, milyen módon befolyásolja ***Al*** módosítása ezt a fejlesztést, vagy hogy lesz-e egyáltalán kapcsolat a kettő között, de meglátása szerint ütközés esetén a korábbi inicializáció megvalósításának elsőbbséget kell éveznie.
Nigel Cunningham benyújtotta a ***Suspend2*** kódját jóváhagyásra és az ***Andrew Morton*** által karbantartott ***– mm*** fába való beillesztésre. Annak ellenére, hogy ***Nigel*** hivatalosan évekkel ezelőtt belekezdett a ***Pavel Machek*** által írt kódrészlet átírásába, csak most és csak ide nyújtotta be azt. Számos felhasználónak máris az a véleménye, hogy a ***Suspend2*** sokkal jobb, mint ***Pavel uswsusp*** nevű kódja, így aztán sokan töltik le ***Nigel*** foltját annak ellenére, hogy az ***uswsusp*** ma már a hivatalos kernel része. Harc azért itt is lesz, mivel ***Pavel*** a régebbi, bizonyítottan működő kód hivatalos karbantartója, így az ő jóváhagyása nélkül nehezen fog bármi új bekerülni a rendszermagba. Mármost ***Pavel*** és ***Nigel*** viszonya kissé antagonisztikus, tehát a dolog várhatóan nem lesz könnyű. Ugyanakkor az is igaz, hogy a ***Nigel*** kódja ellen felhozott érvei egyre inkább elvesztik az alapjukat. ***Pavel*** szerint az ő kódjával pontosan ugyanazt meg lehet csinálni, mint ***Nigel*** változatával. A gond az, hogy a felhasználók csapatostul állítják ennek az ellenkezőjét. Szerintük a ***Suspend2*** igenis jobban működik az ő rendszereiken. Aztán azzal is próbált érvelni ***Pavel***, hogy az ***uswsusp*** egyszerűen jobb ötlet, és a felhasználóknak csupán meg kellene várniuk, amíg befejeződik a fejlesztése. A gond az, hogy a ***Suspend2*** viszont itt és most működik jól, sőt ez az „állapota" már egy ideje tart. Összességében tehát hacsak nem merülnek fel valamilyen komoly, műszaki érvek a ***Suspend2*** ellen, akkor hamarosan be fog kerülni a kernelbe.
És akkor lássuk ***Hans Reiser*** legújabb dolgait... ***Hans*** szerint a kernelfejlesztők közössége merő politikai okok miatt nem engedi be a ***Reiser4***-et a hivatalos rendszermagba. És bár egyes emberek viselkedésén valóban megfigyelhető volt a „politikai szándék", azért ***Hansnak*** nincs teljesen igaza. A helyzet az, hogy az egész vitát ő maga váltotta ki azzal, hogy újra és újra támadásokat intézett azok ellen, akik az általa fejlesztett kódot kritizálni merték, ezt pedig odáig fokozta, hogy a végén egyesek kereken megtagadták, hogy további tanácsokkal lássák el, vagy egyáltalán foglalkozzanak a munkájával. Márpedig e nélkül a ***Reiser4*** fejlesztőinek igen nehéz dolguk lesz, ha fel akarják deríteni azokat a problémákat, amelyek miatt a kód egyelőre nem integrálható a hivatalos rendszermagba. Mivel azonban ***Hans*** a mai napig nem ismerte fel, hogy a probléma ő maga, pontosabban a viselkedése, a megoldás talán az lehetne, hogy ő maga kimarad a vitákból, és azokat a ***Reiser4*** egyéb fejlesztőmérnökei folytatják le. Úgy tűnik ugyanis, hogy a kernelfejlesztők többsége tulajdonképpen szívesen foglalkozna a témával, ha nem kellene magától a névadótól tartania.

Linux Journal 2006., 151. szám

Zack Brown


Linuxvilág
www.linuxvilag.hu

Oracle Unbreakable Linux

Az Oracle bejelentette az Unbreakable Linuxot, melynek keretében Linux terméktámogatási szolgáltatást lehet vásárolni a cégtől kevesebb mint fele annyiért, mint a Red Hat esetében.

Ezzel azt szeretné elérni, hogy az *Oracle* adatbázist használó – vagy használni kívánó – cégek teljes terméktámogatási megoldást (operációs rendszer + alkalmazás szintjén) kapjanak egy gyártótól és ne okozzon gondot a problémák gyökerének behatárolása, elkerülve a gyártók egymásra mutogatását. Az *Oracle* ezzel is csökkenteni szeretné a beüzemelésbe fektetett munkát és költségeket, így vonzóbbá téve ezáltal terméküket a döntéshozók felé. Fontos az is, hogy az *Oracle* nem teszi kötelezővé az *Unbreakable Linux* szolgáltatás igénybevételét , a *Red Hat RHEL*, a *Novell SLES* és az *Asianux* továbbra is szerepel a támogatott disztribúciók listáján. Az *Unbreakable Linux* szolgáltatás a *Red Hat Enterprise Linux* disztribúcióra épül. Az *Oracle* által kiadott hibajavítások ráadásul visszaportolásra kerülnek korábbi verziókba is, melyre a cég ügyfeleinek nagy szüksége van, nem lehet minden esetben megvárni a legújabb, már javított verziót.

Az *Oracle* által támogatott *Linux* így részben eltérhet a *Red Hat Enterprise Linux* disztribúciótól, amely ezáltal a *Red Hat* szerint nem feltétlen lesz binárisan kompatibilis a termékükkel. Ugyanakkor az *Oracle* által készített foltok (patch) továbbra is szabad forrásúak maradnak, melyeket később más disztribúciók is szabadon felhasználhatnak. Az *Oracle* már eddig is adott kódszintű támogatást a már említett disztribúciók esetében, azonban eddig nem lehetett nála megvenni ezt a szolgáltatást. Emellett más nyílt forrású projektekben is részt vett a cég (például az *Oracle Cluster File System*, mely már több *Linux* disztribúció része – ➲ *http://oss.oracle.com*), így már évek óta saját Linux kernel fejlesztő csapattal rendelkezik, mely garanciát jelent a szolgáltatás minőségére.
Az *Oracle Unbreakable Linux* árait az igénybe vett szolgáltatás mértéke és a szerverben található processzorok száma határozza meg az alábbiak szerint.

Jelenleg *x86* és *x86-64* architektúrákra érhető el az *Unbreakable Linux*, valamint az *Oracle* támogat minden olyan hardvereszközt, amelyet a *Red Hat Enterprise Linux 3-as* illetve *4-es* verziója is támogat. *Oracle Unbreakable Linux*ra történő átállás nem igényel újratelepítést, amennyiben a szerveren *Red Hat RHEL 3* vagy *4* fut. Ebben az esetben elég az *Oracle* honlapjáról letölteni a frissítést elvégző programot.

KAPCSOLÓDÓ CÍMEK

A termékről további információt az alábbi oldalakon érhető el:

- ➲ http://www.oracle.com/technologies/linux/index.html
- ➲ http://www.oracle.com/corporate/press/2006_oct/Oracle-Linux-Program.html
- ➲ http://www.oracle.com/corporate/press/2006_oct/oracle-fsg.html

Oracle Linux

	Két processzorig (éves díj)	Korlátlan számú processzor esetén (éves díj)	Szolgáltatás
Enterprise Linux Network Support	99 dollár	99 dollár	Az Unbreakable Linux Networkről letölthető frissítések.
Enterprise Linux Basic Support	399 dollár	999 dollár	Nonstop (24x7) támogatás.
Enterprise Linux Premier Support	1199 dollár	1999 dollár	Nonstop (24x7) támogatás, ezenkívül támogatás a régebbi kiadásokhoz is, a megjelenéstől számított 5 évig.

Minden – jelenleg érvényes – Oracle szerződéssel rendelkező felhasználó ingyenesen kipróbálhatja az Unbreakable Linuxot 2007 január 31-ig, illetve ugyaneddig a megrendelésekből 50%-os kedvezményt kapnak a Basic és a Premier vásárlói.


Linuxvilág
www.linuxvilag.hu

Szoftverek a nyitott vállalatoknak

A Novell új, nyílt vállalati platformjáról, a nyílt forráskódról és a Linuxon futó asztali megoldásokról a Novell elnökhelyettese és műszaki igazgatója, Dr. Jeffrey Jaffe nyilatkozott.

Dr. Jeffrey Jaffe-t nemrégiben nevezték ki a *Novell* vezető alelnökévé és műszaki igazgatójává. A jövőben *Dr. Jaffe* felel a *Novell* technológiai fejlődési irányának kijelöléséért, és ő vezeti a Novell termékfejlesztéssel foglalkozó részlegeit is. *Dr. Jaffe* korábban a *Bell Labs Research* és az *Advanced Technologies* elnöke volt, emellett a *New Jersey Nanotechnology Consortium* igazgatótanácsának elnöki posztját is betöltötte. *Dr. Jaffe* tanácsát technológiai kérdésekben az *Egyesült Államok* kormánya is többször kikérte, majd *Bill Clinton* elnök 1997-ben kinevezte a létfontosságú infrastruktúrák védelméért felelős elnöki bizottság tanácsadó testületének tagjává.

– ***Dr. Jaffe, mi a véleménye a nyílt forráskódú fejlesztésekről?***
– A nyílt forráskódra épülő fejlesztések valóban átrendezik a szoftverfejlesztés szabályait. Éveken át a *Windows* asztali platform volt a domináns, mert egyetlen gyártónak sem volt erőforrása, vagy bátorsága ahhoz, hogy a „saját pályáján" támadja a *Microsoftot*. A nyílt forráskód közösség azonban egy teljesen más szemléletmódot képvisel, hiszen olyan szoftverfejlesztőkből áll, akik együttműködnek a szoftver további fejlesztése érdekében. Tagjai kihasználhatják saját fejlesztéseik eredményeit, és így licenchez juttatják a közösséget is. Így biztosítják a gyors innovációt és az új ötletek terjedését. A *Novell* által szponzorált *openSUSE.org* projekt 2005 októberi elindulása óta több mint 1,4 millió telepítést regisztráltak és már több mint 25 ezer bejegyzett tagja van a közösségnek. A *SourceForge* több mint 130 ezer nyílt forráskódú projektet követ nyomon és több mint egymilliárd dollárnyi kockázati tőke áramlott nyílt forráskódú cégekbe.

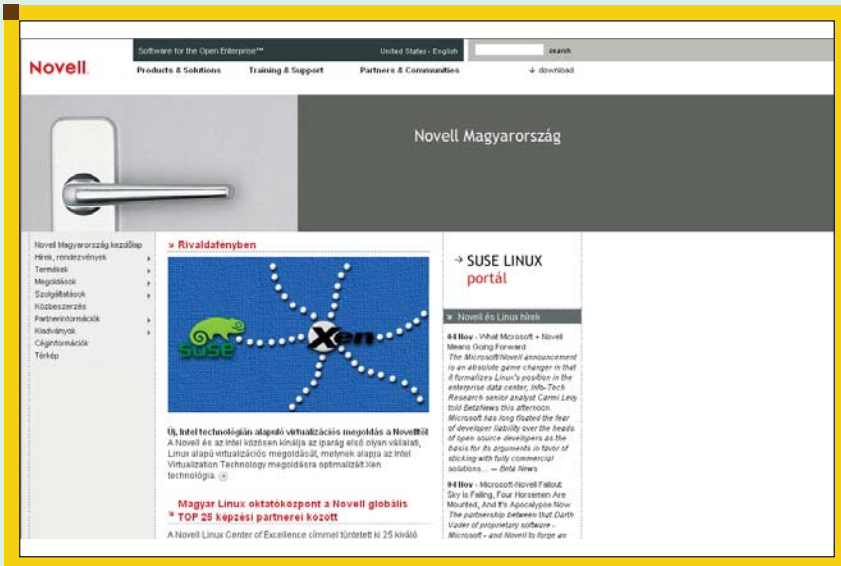
– ***A Novell új munkatársaként Önnek is át kellett állnia a Linux használatára. Mik a tapasztalatai ezzel kapcsolatban?***
– Egy új cégnél mindig van némi izgalom és bizonytalanság. Az izgalom a lehetőségekről szól, a bizonytalanságok nagy része az új informatikai környezetből származik. Sok éven át *Windows* asztali környezetet használtam, és elégedett voltam vele. A *Novellnél* eleinte aggodalommal töltött el, hogy ezentúl Linuxon kell majd dolgoznom, de akkor még nem voltam vele tisztában, hogy mennyit fejlődött a *Linux*.

– ***Használt Linuxot a Bell Labsnél?***
– Néhányan napi rendszerességgel használtak linuxos asztali környezetet, de én nem akartam gondot okozni az informatikusoknak egy nem támogatott asztali környezet használatával.

– ***A Novellnél azonban a Linux az elsődleges asztali környezet?***
– Itt a *Novellnél* a cég több mint 90 százaléka *Linuxot* futtat az asztali gépén, és az emberek több mint fele kizárólag Linux munkaállomáson dolgozik, amelynek vannak kézzelfogható előnyei is,

A nyílt forráskódú fejlesztők nem renegátok vagy műkedvelők: a nyílt forráskódú projektekben résztvevők 58 százaléka régóta informatikai szakemberként dolgozik, átlagosan 11 év tapasztalattal. Az *IDC* független piacelemző cég pedig 24 százalékos növekedést jósol a nyílt forráskódú operációs környezetek és köztesszoftverek terén – ez több mint kétszerese a *Windows* növekedési rátájának.

A linuxos asztali környezetek – sőt, valójában az egész *Linux* – üzleti sikerének biztosításához természetesen konzisztens, biztonságos és megerősített disztribúcióra van szükség, amely teljes támogatással és garanciával rendelkezik. A *Novell* piaci pozíciója, története és mérete (amely egyedülálló a *Linux*-ágazatban) garanciát jelent minderre.



1. ábra Kapcsolódó webcím: www.novell.hu

hiszen a folyamatos újraindítások és a vírusok okozta problémák már rég a múlté.

– ***Új munkatársként milyennek látja a linuxos asztali környezetet a Windowshoz képest?***
– Nagy általánosságban a két asztali környezet teljesen összemérhető. Nem mondom azt, hogy a jelenleg használatban lévő ***SUSE Linux Enterprise Desktop*** minden téren ugyanazt a funkcionalitást nyújtja, mint a ***Microsoft*** asztali munkaállomása, bár a június óta kapható ***SUSE Linux Enterprise Desktop 10*** változata a problémák nagy részét már meg is oldotta. A ***Windowsnak*** vannak plusz funkciói, melyek használata jelentős többletköltséggel jár és veszélyezteti a rendszer biztonságos működését, ezért is kell a ***Windowst*** futtató gépeket sokszor újra indítani.

– ***Úgy érti, hogy a Linux Desktop jobb?***
– Nem, én csak azt mondom, hogy éppen elég jó. A ***Linux*** megfelelő lehet egy cég számára, akkor is, ha nem illeszkedik tökéletesen minden egyéni felhasználó igényeihez, és akkor is, ha az irodai dolgozók számára szükséges legfontosabb vállalati alkalmazásokról van szó.

– ***Hogyan hasznosítható a nyílt forráskódú megközelítés az asztali rendszerek fejlesztése során?***
– A nyílt forráskód közösség az asztali rendszerek fejlesztésének ideális környezete, hiszen mindenkinek van egy egyéni elképzelése arról, hogyan lehetne asztali rendszerét jobbá tenni. A nyílt forráskód tökéletes platform ennek a milliárdnyi egyedi kívánságnak, igénynek és preferenciának az összegyűjtésére. A ***Novellnél*** mi is aktívan közreműködünk a nyílt forráskód közösség munkájában, vezető, fenntartó és egyes esetekben követő szerepet töltünk be. A ***GPL*** licencek révén munkánk eredményei mindenki számára elérhetőek.

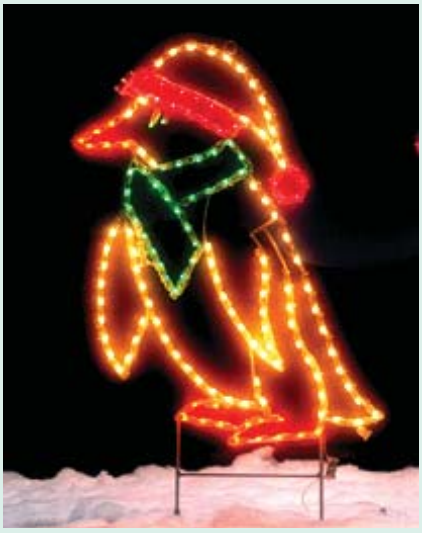
– ***Mivel járul hozzá a Novell a Linux Desktop további fejlesztéséhez?***
– A ***Novellnél*** folyamatosan azon dolgozunk, hogy használhatóvá tegyük a linuxos asztali rendszereket vállalati környezetben is – nem csak azon tranzakciós és rögzített funkciójú felhasználások esetén, ahol a ***Linux*** már most is erős, hanem az alapszintű irodai dolgozók és a magasan képzett dolgozók körében is.
A ***Novell*** több olyan projektet indított el, amely segíti a nyílt forráskód közösség munkájának koordinálását. Ezek közé tartozik az ***openSUSE*** projekt, az ***openSUSE.org***, mely az asztali rendszereken dolgozó fejlesztők közösségét fogja össze. Az ***openSUSE*** közösségben több mint 25 ezer regisztrált tagunk van, és a ***SUSE Linuxot*** naponta több mint 7500 alkalommal telepítik.
A ***„Better Desktop”*** projekt (***betterdesktop.org***) alaposabban vizsgálja néhány használhatósági módszertanunkat.
A megfigyelések során szerzett tapasztalatokat felhasználjuk a rendszer fejlesztésekor.
A ***Novell*** mindezt nyíltan teszi: bárki ellátogathat a weboldalra és szabadon felhasználhatja a tapasztalatainkat. A ***Novell*** egyfajta „szerkesztőként” is működik, miután megpróbálja kiválasztani, hogy melyik projekt a legjobb, vagy melyik illeszkedik a legjobban a többihez a kapcsolódó témakörök megoldására. Emellett támogatást biztosít a többi független szoftvergyártó számára is, a teljes körű megoldás fejlesztése érdekében.

– ***Mi a véleménye a Linux és a Novell jövőjéről?***
– A ***Novell*** elkötelezettsége a valódi vállalkozások üzleti problémáinak megoldásával kapcsolatban, páratlan az iparágban. A mi linuxos platform megközelítésünk – a ***SUSE Linux Enterprise 10*** idei nyári kiadásával – további lehetőségeket teremt arra, hogy ügyfeleink egyre szélesebb körben vezessék be a ***Linuxot*** és alkalmazzák a ***„Szoftverek a nyitott vállalatoknak”*** szemléletet a saját szervezetükben.

Boldog Karácsonyt szinte mindenkinek!

Lassan eljő újra a családi ünnep, mikor is feldíszített fenyők alatt és kibelezett pulykák felett tartós fogyasztási cikkek, továbbá nem feltétlen ízléses, ám annál drágább és haszontalanabb ajándéktárgyak láttán mímelhetünk széles örömöket.

A szerszámosláda, nyakkendő és zokni immáron a múlté – míg pár éve a karácsonyi (giccs) ajándék toplistát oly kreációk vezették, mint a bólogató *Jézus*-figura, a felfújható templom szószékkel vagy az apafiú-szentlélek legószett, mára a mobiltelefon, *iPod*, asztroelemzés, mellformáló készülék és kombináló zsírégető program dívik.
Azt nem lehet tudni, mit kombinált az a bizonyos személy, kinek cipőmérete még *IQ*-ját is jócskán fölülmúlhatja, s legsetétebb idiotizmusával (no meg valószínűleg két kezével) eltulajdonított egy pingvinfiókát az egyik brit állatkertből. A tavaly történtekről még a rendőrség is úgy vélekedik, a ***March of the Penguins***



című film ihlette ünnepi áhítatban elkövetett bűntény tettese karácsonyi ajándékul szánta a kicsiny madárkát…
25 ezer font jutalmat ajánlottak a nyomravezetőnek, ám reményt nem sokat fűztek a pingvinbébi előkerítéséhez, ugyanis legfeljebb néhány napot élhetett szegény szülei táplálása nélkül. Noha még az amerikaiak is imákat rebegtek a kis *Togáért* new yorki templomukban, a kis pingvin nem került elő. Vigaszt nyújtott azonban, hogy szülei nemsokára újabb tojáson üldögéltek, mely gyermekkezdeményt már szigorúan mozgásérzékelőkkel és biztonsági rendszerekkel védték.
Hogy ily esetek ne történhessenek meg, nem árt állatkerti örökbefogadások előtt megfelelően tájékozódni, és akkor tán elkerülhetőek az olyan (gyakori) kérdések is, mint például: „Ha befizetem az örökbefogadás összegét, mennyi idő múlva postázzák nekem az elefántot?”

Szerencsére sokan beérik pingvinből a matricával, képeslappal, fadísszel, meg cseréppel, a különcök pedig megajándékozhatják magukat e hidegben pingvin alakzatú infraégős, gázüzemű hősugárzóval, vagy éppen pingvines bólészettel.
Ez utóbbi ürítése előtt jusson eszünkbe: a munkahelyi karácsonyi bulikban a legutálatosabb az azt követő új álláskeresés..

Halusz Léna


Linuxvilág
www.linuxvilag.hu

Qtopia – Az első lépések

A Trolltech egyik mérnöke beavat bennünket a Qtopiával történő fejlesztés rejtelmeibe.

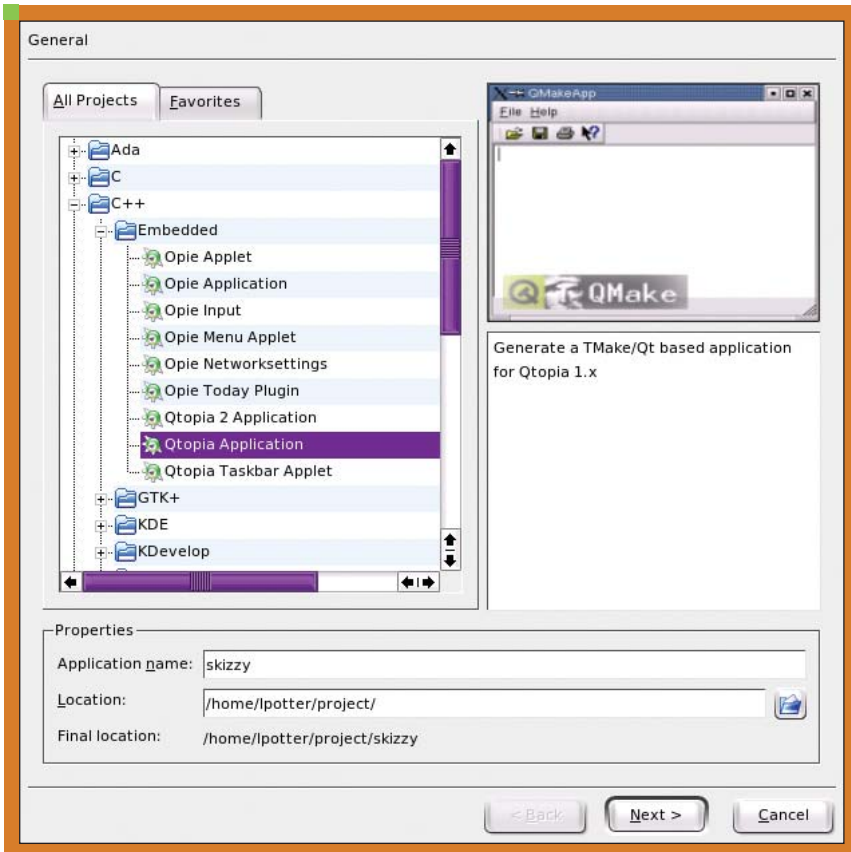
A *Trolltech* által fejlesztett *Qtopia* egy olyan beágyazott alkalmazások fejlesztésére alkalmas fejlesztői keretrendszer, amely teljes szabadságot ad a forgalmazóknak az alkalmazások testreszabásával kapcsolatban. Ezzel párhuzamosan a cég lehetővé teszi, hogy a *GPL* hatálya alá tartozó alkalmazásokhoz a *Qtopia* ingyenesen felhasználható legyen, míg a kereskedelmi szoftverek fejlesztői meg is vásárolhatják azt. Röviden tehát a *Qtopia* kétféle licensszel kerül forgalomba. A rendszerrel történő fejlesztés első közelítésben ugyanolyan, mintha a *Qt/KDE* párost használnánk, eltekintve persze néhány extra lépéstől és eszköztől. Jómagam egy *Gutenbrowser* névre keresztelt alkalmazást kezdtem fejleszteni, ami nem egyéb, mint egy olvasó- illetve letöltőprogram a *Project Gutenberg* által kínált megszámlálhatatlan szöveges dokumentumhoz.
A *Gutenbrowser* először kizárólag *Linuxon* elérhető programként kezdte a pályafutását, de aztán viszonylag gyorsan megszületett a *Windows*, *Qtopia* illetve *Opie* (*Open Palmtop Integrated Environment*) változata is. Az *Opie* amúgy egy a *Qtopia GPL*-es változatán alapuló környezet, amit egy teljes közösség fejleszt.
A *Qtopia* alapját a *Qt Embedded* fejlesztőkörnyezet képezi, így minden, a *Qt* segítségével megírt program könnyen átalakítható úgy, hogy a *Qtopia* környezetben és képes legyen futni. Nekem körülbelül két hetembe telt, amíg „részmunkaidős" nyílt forrású fejlesztőként működésre bírtam a *Gutebrowsert* egy *Sharp Zauruson*. Ebben pedig benne volt magának a *Qtopia API*-nak a megtanulása, valamit a többplatformos fordítás elsajátítása is.
Aki most akarja elkezdeni a *Qtopia* megtanulását, annak azt ajánlom, hogy várja meg a *Qtopia 4* megjelenését, mivel jelentős különbségek vannak a *Qt Embedded 2.3*-as és *4*-es változata között. Ami az átjárhatóságot illeti, *Qt3*, *KDE* sőt még *GTK+* alkalmazásokat is sikeresen portoltak már a *Qtopia 1*-es és *2*-es változataira, de ehhez szükség volt némi extra munkára. Helyettesíteni kellett bizonyos osztályokat, a *KDE* programokhoz pedig a *microkde* forrása is szükséges volt.

Milyen eszközökre lesz szükségünk

Ahhoz, hogy megkezdhessük a munkát a *Qtopia*-val először is szükségünk lesz néhány segédeszközre. A *Qtopia* jelenleg csak *Linuxon* használható tehát szükségünk lesz egy ilyen operációs rendszert futtató gépre. Nyilván kell majd egy olyan képességekkel rendelkező szövegszerkesztő, mint amilyen az *emacs* vagy a *vi*. Én a saját projektemben történetesen a *KDevelopot* használtam, amely rendelkezik egy igen egyszerű *Qtopia* sablonnal.
Ha valamilyen speciális eszközre fejlesztünk alkalmazást, akkor szükségünk lesz egy keresztfordítóra is. Esetünkben az eszköz legyen az *Archos PMA430*, ami a *Qtopia*-hoz az *arm-linux-gcc 2.95*-ös változatát használja. Bár az igaz, hogy a *GCC 3*-as jobban optimalizál, alkalmazásunknak kompatibilisnek kell maradnia az eszközön futó egyéb szoftverekkel, tehát öregecske ugyan a *2.95*-ös, most mégis megteszi. Az *ARM* rendszerekre történő fordításhoz szükséges eszközök az interneten számos helyen fellelhetők. Esetünkben az *Archos* eszközkészletet a ➲ *www.archos.com/products/overview/pma_400_sdk.html* címen találjuk, de számos más link is fellelhető a ➲ *qtopia.net* webhelyen.

Forráskód vagy kész SDK?

Természetesen a *Qtopia*-t magát sem nélkülözhetjük, de itt is van két választási lehetőségünk: letölthetjük a forráskódot, vagy használhatunk



1. ábra Új Qtopia projekt létrehozása

egy lefordított *Qtopia SDK* csomagot is. Ami a *PMA430* platformot illeti a fejlesztőkészlet *GPL* és kereskedelmi változatban egyaránt fellelhető, akárcsak a Qtopia maga. A kereskedelmi csomagot a ➲ *www.trolltech.com/products/qtopia/pricing.html* webhelyen szerezhetjük be teljesen elfogadható összegért, míg a szabad, vagyis a *GPL* hatálya alá tartozó csomagot a ➲ *ftp.trolltech.com/qtopia/sdk* helyről tölthetjük le. Mindkettő a */opt/Qtopia* könyvtárba kerül telepítés után. Ha nincs */opt/Qtopia/arm* könyvtárunk, akkor telepítés után adjuk ki a következő parancsot:

```
# ln -s /opt/Qtopia/sharp
➥/opt/Qtopia/arm
```

A KDevelop projekt

Indítsuk el a *KDevelop* fejlesztőkörnyezetet, majd a *Project* menüből válasszuk a *New Project* pontot. Nyissuk meg az *Embedded* könyvtár alatt található C++ könyvtár ikont. Kattintsunk a *Qtopia Application* nevű fájlra, amellyel egy új *Qtopia* projektet nyitunk meg. Ennek elvileg bármilyen nevet adhatunk, hívhatjuk akár víziónak is. A projekt létrehozására szolgáló dialógusablak az *1. ábrán* látható. Ha megnyitottuk az új projektet, elkezdhetjük azt saját igényeinknek megfelelően átszerkeszteni. Ha egy *Qtopia* projekthez szerkesztünk *.ui* (*user interface*) fájlokat, ehhez mindenképpen a *Qt 2*-ben található Designert használjuk, mivel a későbbi változatokkal készített fájlok nem kompatibilisek a *Qtopia*-val. Én ezt úgy oldottam meg, hogy felvettem egy egyedi külső eszközt, és az *.ui* fájlokat a *Designer 2*-ből nyitottam meg. Ezen a ponton érdemes megjegyezni, hogy nem szabad a *.ui* fájlokat dupla kattintással megnyitni, mivel ennek hatására a *Designer 3* fog elindulni, mivel a *KDevelopban* alapértelmezésként ez van hozzárendelve ehhez a fájltípushoz. Ha ezt el akarjuk kerülni, akkor a *KDevelop* rendszert parancssorból kell futtatnunk úgy, hogy előtte megadjuk a következő környezeti változókat:

```
export PATH=/opt/Qtopia/
➥bin:$PATH
export LD_LIBRARY_PATH=/opt/
➥Qtopia/lib:$LD_LIBRARY_PATH
```

Szintén célszerű beállítani *QVFb* néven a *Qt Virtual framebuffert*, amely célszerűen a */opt/Qtopia/bin/qvfb* helyre kell mutasson. Ebben a virtuális környezetben fut majd az éppen fejlesztet alkalmazásunk a felhasználói felületen. A *Qtopia* közvetlenül a frambufferben jeleníti meg a kimenetét, vagyis nem terheli az *X11* kiszolgálót.

A környezet beállítása

Először is meg kell adnunk néhány környezeti változót a *KDevelop* projekthez. Indítsuk el a *KDevelop* rendszert, majd kattintsunk sorrendben a *Project->Project Options->Run Options* pontokra. Ezután hozzuk létre a következő változókat:

```
Name: QTDIR Value: /opt/Qtopia
Name: QPEDIR Value: /opt/Qtopia
Name PATH Value: /opt/Qtopia/
➥bin:$PATH
Name LD_LIBRARY_PATH Value: /opt/
➥Qtopia/lib:$LD_LIBRARY_PATH
```

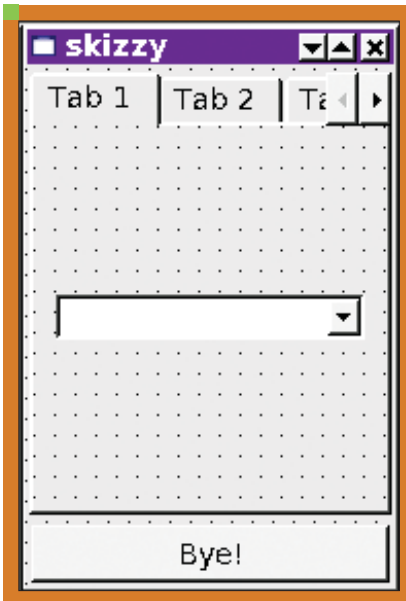
Hasonlóan a *Make Options* alatt is meg kell adnunk néhány értéket: A *skizzy.pro* projektfájlban a *LIBS* értékéhez adjuk hozzá a `-lqtopia` opciót, mivel a *Qtopia 1.7* rendelkezik egy saját könyvtárral. Ezen a ponton a *Makefile*-t kézzel kell létrehoznunk, mivel a *KDevelop* hibásan használja a `tmake` programot:

```
# export TMAKEPATH=/opt/Qtopia/
➥tmake/lib/qws/
➥linux-generic-g++
# tmake -o Makefile skizzy.pro
```

Ha mindezzel megvagyunk, az *F8* leütésével indíthatjuk a projekt fordítását. Ez az utóbbi probléma természetesen kiküszöbölhető, így a *Qtopia* későbbi változatai már a `qmake` programot fogják használni a *Makefile* felépítéséhez.

A beltartalom

Hozzáláthatunk az igazi fejlesztéshez, vagyis elkezdhetjük felruházni alkalmazásunkat néhány funkcióval.
Indítsuk el a *Designer 2* alkalmazást, és nyissuk meg vele a *skizzybase.ui* fájlt. Töröljük belőle a `Qlabel`-t. Hozzunk létre ugyanakkor egy `QTabWidget` objektumot egy



2. ábra A QLayouts használatával elérhetjük, hogy alkalmazásunk automatikusan a megjelenítő felbontásához igazodjon

`QComboBox`-szal egyetemben az első fülön, egy `QlistBox`-ot a másodikon, egy `QMultiLineEdit` elemet pedig a harmadikon (*2. ábra*).
Mentsük a módosított ***.ui*** fájlt. Nyissuk meg a ***skizzy.cpp*** fájlt a ***KDevelop***-ból. Látható, hogy az általunk fejlesztet alkalmazás a `skizzyBase`-ből van származtatva:

```
skizzy::skizzy( QWidget*
➥parent,  const char* name,
➥WFlags fl )
    : skizzyBase( parent,
➥name, fl )
```

Szeretném megváltoztatni a ***main.cpp***-t úgy, hogy az alkalmazás létrehozására egy olyan hatékonyabb metódust használjon, ami az után jelent meg, hogy létrehozták a ***KDevelop*** rendszer ***Qtopia*** sablonját.
Ehhez a szokásos main() függvényt fogjuk megváltoztatni, amely eredeti állapotában így fest:

```
int main( int argc, char **
➥argv )
{
    QPEApplication a( argc,
    ➥argv );
    skizzy mw;
    a.showMainWidget( &mw );
    return a.exec();
}
```

1. Lista Azon függvények megvalósítása, amelyekhez az alkalmazás által használt widget-ek jeleit kapcsoljuk

```
/*
Ez a függvény a Qtopis FontDatabase eszközét használja
arra, hogy egy combo dobozt feltöltsön fontok neveivel.
*/
void skizzy::fillCombo()
{
        QStringList families = fontdb.families();
        for ( QStringList::Iterator f = families.begin(); f
        ➥!= families.end();++f ) {
        QString family = *f;
                ComboBox1->insertItem( family);
        }
}

/*
Ez a függvény akkor fut le, amikor a felhasználó kiválasztja
a combo dobozt, Kitölti a második fülön található listadobozt
azoknak a fontoknak a nevével, stílusával és méretével,
amelyeket a felhasználó kiválasztott. */

void skizzy::comboSelected(const QString &selectedFont)
{
ListBox1->clear();
 QStringList styles = fdb.styles( selectedFont );
        for ( QStringList::Iterator s = styles.begin(); s !=
        ➥styles.end();++s ) {
              QString style = *s;
              QValueList<int> smoothies = fdb.smoothSizes
              ➥( selectedFont, style );
               for ( QValueList<int>::Iterator points =
               ➥smoothies.begin(); points != smoothies.end();
               ➥++points ) {
                   QString pointSize = selectedFont + " "+
                   ➥style +" "+QString::number( *points )
                   ➥+ " ";
                  ListBox1 ->insertItem( pointSize);
               }
           }
    TabWidget2->showPage(tab2);
}

/*
Ez a függvény egy mintaszöveget mutat meg a QMultiLineWidget
segítségével.
A szöveg a felhasználó által kiválasztott fonttal jelenik meg
a harmadik fülön. */

void skizzy::showFont( QListBoxItem *item)
{
        QStringList fontItemString = QStringList::split
        ➥(' ',item->text());
        QString family, style, point;
    family = fontItemString[0];
    style = fontItemString[1];
```

1. Lista folytatás

```
    point = fontItemString[2];
    bool ok;
    int i_size = point.toInt(&ok,10);

    if (!ok) {
        style += " "+fontItemString[2];
        point = fontItemString[3];
        i_size = point.toInt(&ok,10);
    }
    QFont selectedFont( family);
    selectedFont.setPointSize(i_size);

     if(style.find("Italic",0,TRUE) != -1) {
          selectedFont.setItalic(TRUE);
     }

     if(style.find("Bold",0,TRUE) != -1) {
        selectedFont.setWeight(QFont::Bold);
     }

     if(style.find("Light",0,TRUE) != -1) {
        selectedFont.setWeight(QFont::Light);
    }

    MultiLineEdit1->setFont( selectedFont);
    MultiLineEdit1->setText( tr( "The Quick Brown Fox Jumps
    ➥Over The Lazy Dog" ) );
    MultiLineEdit1->setWordWrap( QMultiLineEdit::WidgetWidth);
        TabWidget2->showPage(tab3);
}
```

A változás lényege, hogy a fenti függvényt lecseréljük a *Qtopia* alkalmazásmakrójára:

```
QTOPIA_ADD_APPLICATION("skizzy",
➥skizzy);
QTOPIA_MAIN
```

Ez a módosítás lehetővé teszi, hogy gyorsan indítható alkalmazást hozzunk létre, hiszen a lényeg itt annak a közös alkalmazás-konstruktornak a használata, amely már eleve a memóriában van.
Alkalmazásunk egyelőre nem csinál semmi értelmeset, tehát most már valóban álljunk neki az érdemi munkának a következőképpen:

```
#include <qpe/fontdatabase.h>
```

Hozzunk létre egy privát tagot:

```
FontDatabase fdb;
```

Aztán néhány függvény és privát *slot* következhet:

```
private slots:
void fillCombo();
void comboSelected(const
➥QString &);
void showFont( QListBoxItem *);
```

A *skizzy.cpp* fájlban meg kell adnunk még néhány kiegészítő sort ahhoz, ami hamarosan következik:

```
#include <qstringlist.h>
#include <qcombobox.h>
#include <qtabwidget.h>
#include <qlistbox.h>
#include <qmultilineedit.h>
#include <qfont.h>
#include <qfontinfo.h>
```

3. ábra A kész program a néhány nem túlságosan hasznos funkcióval


```
#include <qpe/fontdatabase.h>
```

Aztán meg kell valósítanunk azokat a függvényeket, amelyekhez a widget-ek jeleit kapcsoljuk (*1. Lista*).
A *Qt* és a *Qtopia* egyaránt jeleket használ a widget-ek közti kommunikáció lebonyolítására. Minden widgetnek megvannak a kibocsátható jelei, amelyeket a kapcsolati makrók segítségével köthetünk a megfelelő függvényekhez.
Kapcsoljuk például a `ComboBox1` `activated` jelét, valamit a `ListBox1` `clicked` jelét a saját slotjainkhoz a következőképpen:

```
skizzy::skizzy( QWidget*
➥parent,  const char* name,
➥WFlags fl )
    : skizzyBase( parent, name,
    ➥fl )
{
    connect(bye, SIGNAL
    ➥(clicked()), this,
    ➥SLOT(goodBye()));
    connect(ComboBox1, SIGNAL
    ➥(activated(const QString
    ➥&)), this, SLOT
    ➥(comboSelected(const
    ➥QString &)));
    connect(ListBox1, SIGNAL
    ➥( clicked ( QListBoxItem
    ➥* )), this, SLOT(showFont
```

```
    ➥( QListBoxItem*)));
    fillCombo();
}
```

Figyeljük meg, hogy a slotot kezelő függvény pontosan ugyanolyan típusú argumentumot fogad, mint amilyen az őt vezérlő jel típusa.
A *KDeveleopban* nyomjuk meg az *F8*-at, vagy válasszuk a *Build* menüből a *Build Project* pontot. A program ennek hatására a natív fordítóprogram használatával lefordul. Ha futtatni akarjuk, indítsuk el a *QVFb*-t, majd egyszerűen válasszuk a *Build* menüből az *Execute Main Program* pontot. Programunk ennek hatására megjelenik a *QVFb* keretrendszerben.

Más platformra történő fordítás

Most tehát van egy úgy ahogy működő programunk, de egyelőre csak a fejlesztéshez használt gépen fut. Le kell tehát fordítanunk az *Archos* rendszerére is. A megfelelő könyvtárak használatához először is meg kell változtatnunk a projekt tulajdonságait. Mivel immár készen állunk a valódi eszközre való fordításhoz, töröljük is az eredeti projektet a *Build* menü *Clean Project* pontjával.
A *Project Options->Make Options* dialógus segítségével végezzük el a következő módosításokat:

```
Name: QTDIR Value: /opt/
➥Qtopia/arm
Name: QPEDIR Value: /opt/
➥Qtopia/arm
Name PATH Value: /usr/local/
➥arm/bin:/opt/Qtopia/tmake/
➥bin:$PATH
Name TMAKEPATH Value:/opt/
➥Qtopia/tmake/lib/qws/
➥linux-arm-g++
```

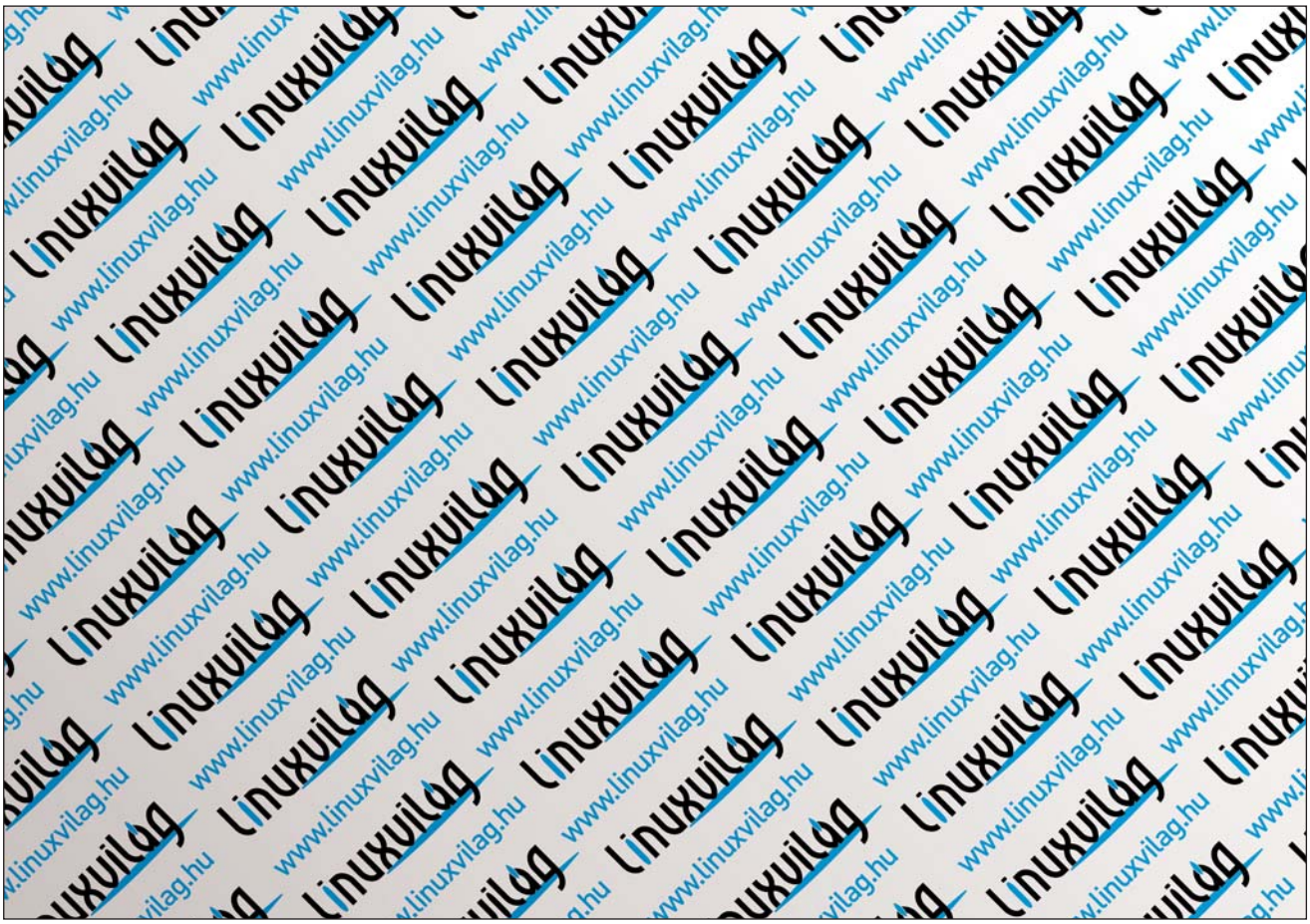
Töröljük a *Makefile*-t, majd futtassuk a parancssorból a következő parancsot annak a *Makefile*-nak a létrehozásához, ami már az *arm-linux* fordítót használja:

```
# export TMAKEPATH=/opt/Qtopia/
➥tmake/lib/qws/linux-arm-g++
# tmake -o Makefile skizzy.pro
```

A projekt lefordításához üssük le az *F8* billentyűt. A keletkező bináris állományt *USB* porton átmásolhatjuk az *Archos* eszközre, és ott futtathatjuk. Ha telepítőcsomagot akarunk létrehozni, akkor nem árt tudni, hogy a *Qtopia* a *handhelds.org* webhelyen található *Itsy Package Managemenet* (röviden *ipkg*) használja a programok telepítésére. Aki többet szeretne erről megtudni, az látogasson el a *Trolltech Qtopia.net* címen található webhelyére.

Linux Journal 2006., 142. szám

Lorn Potter
a Trolltechnél dolgozik mint a Qtopia közösség vezetője (Qtopia Community Manager). Amerikai, de jelenleg Ausztráliában, Brisbane-ben él ausztrál feleségével és fiával. A nyílt forrású programok fejlesztését saját erejéből tanulta meg. Jelenleg az Opie (Open Palmtop Integrated Environment) Projekt egyik vezető fejlesztője. Korábban zenészként, hangmérnökként és síőrültként is dolgozott.



A db4o objektumközpontú adatbázismotor beépítése saját alkalmazásainkba

A cikkből azt tudhatjuk meg, miként használhatjuk ezt a kis helyigényű, objektum-orientált adatbázis beágyazott rendszereken futó programokban.

A *db4o* egy nyílt forrású, objektum-orientált adatbázis, amit a *db4object* készít. A rendszer beágyazható abban az értelemben, hogy mivel a teljes *db4o* motor egyetlen könyvtár formájában van megvalósítva, egyszerűen összeszerkeszthetjük a saját programunkkal. A db4o motornak ráadásul van megfelelő változata *Java*-hoz, illetve a *.NET/Mono* rendszerekhez egyaránt. Ennek megfelelően ideális *OODBMS* motor *Linuxon* történő fejlesztéshez, akár *Java*-t, akár a *.NET* keretrendszert használjuk. Az ebben a cikkben bemutatásra kerülő példák *C#* nyelven készültek, és egy *Ubuntu Linux* rendszeren futó *Mono* fordítóval próbáltam ki őket. (Az alkalmazások ezen kívül biztosan futnak még *Monoppix* rendszeren is.)
Amellett, hogy nyílt forrású eszköz lévén egyszerűen letölthetjük és azonnal használni kezdhetjük, a *db4o*-nak megvan az az előnye is, hogy tömör, rendkívül jól átgondolt programozási felületet biztosít a fejlesztőnek. A legtöbb esetben a saját metódusainkat kilenc alapvető hívásból vezethetjük le. Mellesleg a könyvtár memóriaigénye nem különösebben nagy, így kifejezetten jól használható olyan esetekben, amikor a rendelkezésre álló erőforrások korlátozottak. (Ugyanakkor a db4o minden kétséget kizáróan alkalmatlan egész vállalkozások vagy szervezetek kiszolgálására.)
Annak ellenére, hogy a *db4o* egész egyszerűen használható, és az erőforrásigénye is kicsi, gyakorlatilag az összes olyan szolgáltatással rendelkezik, amit egy kereskedelmi adatbázismotortól elvárnánk. Lehetővé teszi a többfelhasználós működést, valamennyi, az adatbázishoz való hozzáférést a felhasználó számára láthatatlan módon tranzakciókba foglal, ezen kívül pedig minden működési folyamata során követi az úgynevezett *ACID* elveket (atoicity, consistency, isolation, durability).
Számos más, objektum-orientált és objektum-relációs adatbázismotortól eltérően a db4o használatakor nem kell az általunk írt kódot elő- vagy utófeldolgozásnak alávetni. Arra sincs szükség, hogy az állandó osztályokat egy speciális, az állandóságot külön kezelő szuperosztályból vezessük le. A *db4o* kiválóan működik a közönséges objektumokkal, és azt sem kell vele előre közölnünk, hogy milyen egy osztály belső szerkezete, mielőtt az ebbe tartozó objektumot letárolnánk azt egy *db4o* adatbázisban.

Amint ez a cikkből is hamarosan kiderül, mindezek a tulajdonságok néhány különleges képességgel is felruházzák ezt a fejlesztőeszközt.

Egy szótár-adatbázis kialakítása

Tegyük fel, hogy az alkalmazás, amit fejlesztünk egy szótár, mégpedig a szó klasszikus értelmében. A programnak tehát egy olyan adatbázist kell tudnia kezelni, amelyek szavakat, és azok meghatározását tartalmazza. Egy ilyen alkalmazásnál a szótári bejegyzések modellezésére a következő osztályt adhatjuk meg:

```
/*
 * DictEntry
 */
using System;
using System.Collections;
namespace PersistentTrees
{
        /// <summary>
        /// DictEntry class
        /// A dictionary entry
        /// </summary>
        public class DictEntry
        {
                private string theWord;
                private string pronunciation;
                private ArrayList definitions;

                public DictEntry()
                {
                }

                // Create a new Dictionary Entry
                public DictEntry(string _theWord,
                                string
➥_pronunciation)
                {       theWord = _theWord;
                        pronunciation =
➥_pronunciation;
                        definitions = new
➥ArrayList();
```

```
                }

                // Add a definition to this entry
                // Note that we do not check for
                // duplicates
                public void add(Defn _definition)
                {
                        definitions.Add
➥(_definition);
                }

                // Retrieve the number of
                // definitions
                public int numberOfDefs()
                {
                        return definitions.Count;
                }

                // Clear the definitions array
                public void clearDefs()
                {
                        definitions.Clear();
                        definitions.TrimToSize();
                }

                // Properties
                public string TheWord
                {
                        get { return theWord; }
                        set { theWord = value; }
                }
                public string Pronunciation
                {
                        get { return
➥pronunciation; }
                        set { pronunciation =
➥value; }
                }

                // Get reference to the
                // definitions
                public ArrayList getDefinitions()
                {
                        return definitions;
                }
        }
 }
```

Amint látható, a `DictEntry` objektumoknak összesen három eleme van: maga a szó, annak a kiejtése, valamint egy lista, ami a meghatározásokat tartalmazza. Mindeközben egy osztály, amely a meghatározásokat hordozó objektumot írja le a a következőképpen nézhet ki:

```
/*
 * Defn
 *
 */
using System;

namespace PersistentTrees
{
        /// <summary>
        /// Description of Class1.
        /// </summary>
        public class Defn
        {
                public static int NOUN = 1;
                public static int PRONOUN = 2;
                public static int VERB = 3;
                public static int ADJECTIVE = 4;
                public static int ADVERB = 5;
                public static int CONJUNCTION = 6;
                public static int PARTICIPLE = 7;
                public static int GERUND = 8;

                private int pos;
                private string definition;

                public Defn(int _pos,
                           string _definition)
                {
                        pos = _pos;
                        definition = _definition;
                }

                // Properties
                public int POS
                {
                        get { return pos; }
                        set { pos = value; }
                }
                public string Definition
                {
                        get { return definition; }
                        set { definition = value; }
                }
        }
}
```

Látható, hogy a `Defn` objektum tartalmaz egy egész típusú tagot, amely a kiejtést jelzi, valamint egy karakterláncot, amely a meghatározás szövegét hordozza. Ez a szerkezet lehetővé teszi, hogy egy a szótárban szereplő szóhoz akár több meghatározást is hozzárendeljünk.
Az ilyen típusú elemeknek egy ***db4o*** adatbázisban való letárolása döbbenetesen egyszerű. Tegyük fel, hogy a „float” szót akarjuk felvenni a szótárba, és egyszerre három meghatározást is szeretnénk hozzárendelni. Ezt a következő kóddal érhetjük el.

```
Defn _float1 = new Defn(VERB, "To stay on top of
➥a liquid.");
Defn _float2 = new Defn(VERB, "To cause to
➥float.");
Defn _float3 = new Defn(NOUN, "Anything that
➥stays on top of water.");
DictEntry _float = new DictEntry("float",
➥"flote");
```

```
_float.add(_float1);
_float.add(_float2);
_float.add(_float3);
```

Ezen a ponton tehát már rendelkezünk egy *_float* nevű, *DictEntry* típusú objektummal, amelynek a meghatározás része három bejegyzést tartalmaz.
A bevitelhez először is megnyitjuk magát az adatbázist. Egy *db4o* adatbázist mindig egy `ObjectContainer` objektum modellez, amit viszont a következő módon nyithatunk meg, vagy hozhatunk létre, ha történetesen még nem létezik:

```
ObjectCointainer db =
➥Db4o.openFile("<filename>");
```

Itt `<filename>` annak a fájlnak a neve és elérési útvonala, amely az `ObjectContainer` objektum állandó tartalmát hordozza. Az `ObjectContainer`-be objektumot behelyezni a `set()` metódussal tudunk. Tároljuk le tehát az imént létrehozott új bejegyzést:

```
db.set(_float);
```

Lehet ugyan, hogy így elsőre hihetetlennek tűnik, de ezzel a dolog el is van intézve. Összesen ennyit kell tudnunk a `set()` metódusról. A fenti egyetlen hívás nem csak letárolja a `_float` nevű, `DictEntry` típusú objektumot az adatbázisban, de beleírja a hozzá tartozó összes `Defn` objektumot is. Amikor meghívjuk a *db4o* `set()` metódusát, a rendszer automatikusan, a felhasználó számára láthatatlanul végigpásztázza az adott objektum összes függőségét, felderíti a gyermekobjektumokat, és azokat is megfelelően kezeli.
Ha tehát van egy akármilyen összetett objektumszerkezetünk, amit tárolni szeretnénk, semmi egyebet nem kell tennünk, csak átadni a `set()` metódusnak a szerkezet gyökérobjektumát. A többit a rendszer magától elintézi, és az egész szerkezetet egyetlen hívással letárolja. A *db4o*-nak nem kell külön „elárulnunk” az általunk kezelt objektumok szerkezetét, mivel képes magától felderíteni azt.
Ha nem beírni, hanem lekérdezni szeretnénk egy objektumot az `ObjectContainer`-ből, akkor azt a *db4o* úgynevezett *QBE (query by example)* mechanizmusával határozhatjuk meg. A *QBE* stílusú lekérdezéseket egy minta- vagy sablonobjektum vezérli. Kicsit konkrétabban egy lekérdezést ebben az esetben úgy kell végrehajtanunk, hogy létrehozunk egy mintaobjektumot, majd feltöltjük a mezőit azokkal az értékekkel, amelyekre szeretnénk rákeresni. Ezt aztán átadjuk a rendszernek, és ezzel képletesen azt mondjuk neki: *„Nézd öregem, itt van ez az objektum. Keresem az összes olyan tárolt objektumot, ami pont így néz ki, mint ez itt.”*
Ha tehát az imént létrehozott bejegyzést (float) akarjuk a szótárból visszakerestetni, akkor a következőképpen kell eljárnunk:

```
// Create template
DictEntry DTemplate = new DictEntry("float", "");
// Execute QBE
ObjectSet results = db.get(DTemplate);
// Iterate through results set
while(results.hasNext())
{
  DictEntry _entry = (DictEntry)results.next();
  ... process the DictEntry object ...
}
```

Látható, hogy először is létrehoztunk egy sablonobjektumot, aztán kitöltöttük a mezőit azokkal a dolgokkal, amelyekre kíváncsiak vagyunk. Azokba a mezőkbe, amelyek tartalmára nem vagyunk kíváncsiak nulla került, vagy üres karakterlánc, attól függően, hogy milyen típusú adatról van szó. (A fenti példában mi egyszerűen csak a „float” szót keressük a szótárban. Ennek megfelelően a sablonobjektum konstruktorában üresen hagytuk a kiejtésre vonatkozó mezőt, hiszen ennek most a lekérdezés szempontjából nincs jelentősége.)
Ha készen van a sablon, a lekérdezést az `ObjectContainer` objektum `get()` metódusának meghívásával hajthatjuk végre. Ennek egyetlen argumentumként magát a sablont kell átadni. A lekérdezés eredménye egy *ObjectSet* objektum lesz, amin végighaladva kiolvashatjuk a konkrét találatokat.

Indexek létrehozása

Ezen a ponton tehát már képesek vagyunk könnyedén létrehozni egy adatbázist, fel tudjuk tölteni azt tartalommal, és le is tudjuk kérdezni az elemeit a *db4o QBE* mechanizmusával. De mi van akkor, ha más, index alapú lekérdezésekkel szeretnénk kísérletezni? Mivel az adatbázis megőrzi az állandó objektumok közti relációkat, felépíthetünk egy olyan egyedi indexet és navigációs struktúrát, ami a tájékozódást segíti. Ezt az indexet aztán szintén elhelyezhetjük magában az adatbázisban úgy, hogy a megfelelő adatobjektumok is hozzá vannak „drótozva”.
Hogy mindez milyen egyszerűen megvalósítható, azt két, erősen eltérő indexelési szerkezet létrehozásával fogom demonstrálni.
Az első példában egy bináris fát fogunk létrehozni. A fa valamennyi csomópontja egy kulcs-érték párt fog tartalmazni. A kulcs egy a szótárban található szó lesz, míg az adattartalom egy hivatkozás lesz arra a *DictEntry* objektumra, ami a kérdéses szót tárolja az adatbázisban. Ezzel a szerkezettel tehát lehetőségünk lesz arra, hogy az adatbázisból először lekérjük magát a bináris fát, azon végrehajtsuk a keresést, majd a megtalált kulcs-érték pár(ok) alapján lekérjük magát a *DictEntry* objektumot is (már ha van találat).
Mivel a bináris fák felépítését és működését az Olvasó bizonyára jól ismeri, itt és most nem is mennék bele ennek a taglalásába. (Ami azt illeti manapság számos keretrendszer szabványos adatszerkezetként tartalmazza az ilyen struktúrákat. Én ennek ellenére explicit módon hoztam létre egy ilyen fát, mivel azt szerettem volna szemléltetni, hogy mennyire könnyű az adatbázisban való letárolása.)
A bináris fa megvalósítását az *1. Lista* tartalmazza. Meglehetősen kezdetleges szerkezetről van szó, ami kizárólag a beillesztést, a lekérdezést és a keresést támogatja. Nem garantálja a fa kiegyensúlyozottságát sem, hiszen csak demonstrációs célokat szolgál, nem akarunk vele semmi komolyat művelni. A *TreeNode* osztály megvalósítása, amely a bináris fa csomópontjainak szerkezetét írja le, a *2. Listában* látható. (Az *1. Listában* felbukkanó `db.activate()` hívások értelmére hamarosan fény derül.)

1. Lista A bináris fa megvalósítása

```
/*
 * Binary Tree
 */
using System;
using com.db4o;
namespace PersistentTrees
{
        /// <summary>
        /// Description of BinaryTree.
        /// </summary>
        public class BinaryTree
        {
                // The tree's root
                private TreeNode root;
                public BinaryTree()
                {
                        root = null;
                }
                public static BinaryTree
                ➥nullfactory()
                {
                        return(new
                        ➥BinaryTree());
                }
                // insert
                // Add key to tree with
                // associated object reference
                public void insert(string _key,
                ➥Object _data)
                {
                        // Use recursion for
                        // this
                        root = insert(root,
                        ➥_key, _data);
                }
                // insert
                // This is worker method for
                // inserting key and data
                // Insert _key into subtree t
                // with _data associated
                private TreeNode insert(TreeNode
                ➥t, string _key, Object
_data)
                {
                        // If this subtree is
                        // empty, build a new node
                        if(t == null)
                                t = new TreeNode
                                ➥(_key, _data);
                        else

if(_key.CompareTo(t.Key)<=0)
                                        t.Left =
➥insert(t.Left,_key,
_data);
                        else
                                t.Right =
➥insert(t.Right,_key,
_data);
                        return(t);
                }
                // search
                // Search for a key in the tree.
                // Return the array from the
                // TreeNode if found, null if
                // not
                // db is the ObjectContainer
                // holding the tree.
                public Object[] search(string
                ➥_key,
                                Object
➥Container db)
                {
                        TreeNode t =
➥search(root, _key, db);
                        if(t==null)
➥return(null);       // Not found
                        db.activate(t,4);     //
Activate to get data
                        return(t.getData());
                }
                // search
                // This is the worker method for
                // searching.
                private TreeNode search
                ➥(TreeNode t,
                                string
                                ➥_key,
                               Object
➥Container db)
                {
                        // Empty tree?
                        if(t==null)
➥return(null);

if(_key.CompareTo(t.Key)==0) return(t);

if(_key.CompareTo(t.Key)<0)
                        {
                                db.activate
➥(t.Left,2);
                                return(t =
➥search(t.Left,_key,db));
                        }
                        db.activate(t.Right,2);
                        return(t = search
➥(t.Right,_key,db));
                }
        }
}
```

2. Lista A TreeNode osztály megvalósítása

```
/*
 * TreeNode
 */
using System;
namespace PersistentTrees
{
        /// <summary>
        /// Description of TreeNode.
        /// </summary>
        public class TreeNode
        {
                public TreeNode()
                {
                }
                private TreeNode left;  // Left
➥child
                private TreeNode right; // Right
➥child
                private string key;     // Key
➥for this node
                private Object[] data;  // Data
➥associated with key
                // Create a new TreeNode, loaded
                // with
                //  key and data.
                public TreeNode(string _key,
➥Object _data)
                {
                        left = null;
                        right = null;
                        key = _key;
                        data = new Object[1];
                        data[0] = _data;
                }
                // addData
                // Adds new data item to an
                // existing node.
                // The array is extended.
                public void addData(Object
➥_data)
                {
                        Object[] newdata = new
                        ➥Object[data.Length+1];
                        Array.Copy(data,0,
                        ➥newdata,0,data.Length);
                        newdata[data.Length]=
                        ➥_data;
                        data = newdata;
                }
                // Property access
                public TreeNode Left
                {
                        get { return left; }
                        set { left = value; }
                }
                public TreeNode Right
                {
                        get { return right; }
                        set { right = value; }
                }
                public string Key
                {
                        get { return key; }
                        set { key = value; }
                }
                public Object[] getData()
                {
                        return data;
                }
        }
}
```

A következő példában egy olyan speciális indexelő adatszerkezetet fogunk létrehozni, ami kifejezetten szöveges tartalmaz keresésére van kihegyezve. Működésének lényege, hogy szintekre tagolt csomópontok sorozatából épül fel. Minden szinthez egy-egy karakternyi információ tartozik, a szinteződés maga pedig a betűk egymásutániságának felel meg. A legfelső, vagyis a gyökérelem tehát egy tetszőleges szó első betűje lehet, a második szinten levők bármely szó második helyen levő karakterének felelnek meg, és így tovább. A szerkezet elemei tehát úgy követik egymást, mint gyöngyök a gyöngysoron. Egy keresés során nincs más dolgunk, mint szintenként végigmenni felülről lefelé ezen a szerkezeten, amíg teljesen végig nem betűztük a keresett kifejezést.

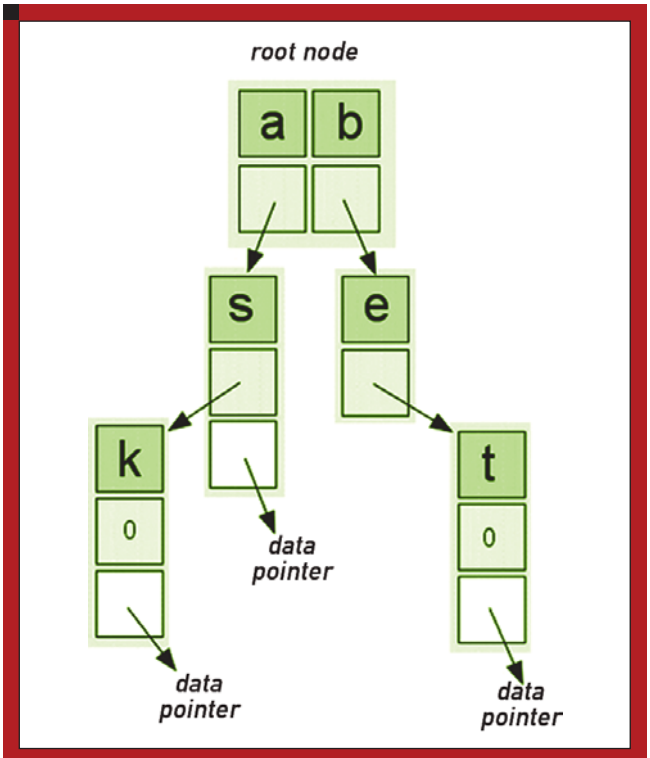
Ha elsőre kicsit nehéz lenne elképzelni a dolgot, az *1. ábra* talán segíthet benne.

Ha új szót akarunk beilleszteni egy ilyen szerkezetbe, nincs különösebben nehéz dolgunk. Vesszük a szó első betűjét, és megvizsgáljuk a gyökércsomópontot, hogy van-e egyezés. Ha nincs, akkor létrehozzuk az új karakternek megfelelő elemet, majd az algoritmus ettől kezdve új csomópontokat fog beilleszteni a szerkezetbe. (Minden új csomópontot a beilleszteni kívánt szó soron következő betűjével inicializálunk.) Ha a keresett karakter már létezik a szerkezetben, akkor az algoritmus követi a letárolt mutatót, és az alapján elmegy a következő szintre. Ez után a vizsgálat módszere teljesen azonos a korábbival, tehát egyszerűen csak rekurzív módon folytatni kell a vázolt műveletsort, amíg a szó végére nem érünk. Ha nincs több azonosítandó karakter, akkor az a csomópont, ahol éppen tartózkodunk fogja tárolni a megfelelő adatmutatót.

Az ilyen szerkezetű index keresése ugyanilyen egyszerű. Természetesen most is gyökércsomópontnál kezdünk, és megvizsgáljuk, hogy van-e illeszkedés az első karakterre. Ha van, akkor a megfelelő hivatkozás mentén a következő szintre lépünk, és ott folytatjuk a keresést a második betűvel. Ha már az első szinten sincs egyezés, akkor a felhasználó egy „nem található” üzenete kap vissza. Ha a keresés

során végigértünk a szó összes karakterén, akkor az utolsó csomópontból kiolvassuk a megfelelő `DictEntry` objektum mutatóját, és azt lekérjük az adatbázisból.
A „nyaklánc” megvalósítása a *3. Listában* olvasható.
Amint a fenti két példa is kiválóan mutatja a ***db4o*** segítségével ugyanolyan egyszerűen kezelhetjük az adatbázis-objektumokat, mintha közönséges, a memóriában tárolt objektumok lennének. Ez az állítás pedig független a megvalósítani kívánt adatszerkezettől. Ha pedig indexet szeretnénk létrehozni, nincs más dolgunk, mint a megfelelő adatelemre történő hivatkozást letárolni annak a megfelelő pontján. Ráadásul mivel az adatbázis nem tesz különbséget indexobjektumok és adatobjektumok között, nem kell létrehoznunk külön index- és adatfájlokat. Mindent, amire szükségünk van, egyetlen helyen tárolhatunk, ami sokkal nagyobb előny annál, mint ahogy az ember elsőre képzelné.
A következő kód elolvas egy szavakat és meghatározásokat tartalmazó szöveges fájlt, létrehozza a megfelelő `DictEntry` objektumokat, letárolja azokat az adatbázisban, felépíti a korábban bemutatott bináris fát és „nyaklánc” indexet, és elhelyezi azokban a `DictEntry` objektumok mutatóit:

```
string theword;
string pronunciation;
int numdefs;
int partofspeech;
string definition;
DictEntry _dictEntry;

// Open a streamreader for the text file
FileInfo sourceFile = new FileInfo(textFilePath);
reader = sourceFile.OpenText();
```



1. ábra A „nyaklánc” modell. Ez egy olyan index, amelyben a szavak egymást követő karakterei a szerkezet egymást követő szintjeinek felelnek meg. A képen látható index összesen három szót tárol: as, ask, és bet. A csomópontokban szereplő adatmutatók esetünkben azokra a szótárban tárol DictEntry objektumokra mutatnak, amelyek az adott szónak felelnek meg.

3. Lista Egy különleges, kifejezetten szöveg alapú keresésre kitalált indexelési szerkezet megvalósítása

```
/*
 * Trie
 */
using System;
using com.db4o;
namespace PersistentTrees
{
        /// <summary>
        /// Description of Trie.
        /// </summary>
        /// trie class
        public class Trie
        {
                private TriePnode root;  // Root
➥of Trie
                // Constructor
                public Trie()
                {
                        root = null;
                }
                // insert
                // Insert a key/data pair into
                // the tree.
                //  Allows duplicates
                public void insert(string
➥key,  // Key to insert
                        Object data)   // Data
➥assoc.
with key
                {
                        TriePnode t = root;
                        TriePnode parent = null;
                        int index=0;
                        int slen = key.Length;
                        for(int i=0; i< slen;
                        ➥i++)
                        {
                                char c = key[i];
                                // If a node
➥doesn't exist – create it
                                if(t == null) t =
➥new TriePnode();
                                // If this is
➥the first node of the tree,
                                // it is the
```

3. Lista folytatás

```
                                //  root.
➥Otherwise, it is stored in the
                                //  pnodes array
                                //  of the
➥parent
                                if(i==0)
                                        root =
➥t;
                                else
                                        parent.
➥setPnodePointer(index, t);
                                // If the
➥character is not on the node,
                                // add it
                                if((index=
➥t.isCharOnNode(c))==-1)
                                        index =
➥t.addKeyChar(c);
                                if(i == slen-1)
➥break;
                                parent = t;
                                t =
t.getPnodePointer(index);
                        }
                        // Finally, add the data
                        // item
                        t.addData(index, data);
                }
                // search
                // Searches for a string in the
                // trie.
                // If found, returns the
                // Object[] data array
                // associated.
                // Else, returns null
                // db is the ObjectContainer
                // holding the trie
                public Object[] search(string
                ➥_key,
                                        Object
➥Container db)
                {
                        TriePnode t;
                        char c;
                        int index=0;
                        // Empty trie?
                        if((t=root)==null)
                        ➥return(null);
                        int slen = _key.Length;
                        for(int i=0; i<slen;
                        ➥i++)
                        {
                                c = _key[i];
                                if((index=
➥t.isCharOnNode(c))==-1)
return(null);
                                if(i==slen-1)
➥break;

db.activate(t,2);
                                t = t.getPnode
➥Pointer(index);
                        }
                        // Get the data
                        db.activate(t,3);
                        return(t.get
➥DnodePointers(index).getData());
                }
        }
}
```

```
// Open/create the database file
ObjectContainer db = Db4o.openFile
➥(databaseFilePath);

// Create an empty Binary tree, and an empty trie
BinaryTree mybintree = new BinaryTree();
Trie mytrie = new Trie();

// Sit in an endless loop, reading text,
//  building objects, and putting those objects
//  in the database
while(true)
{
  // Read a word.
  // If we read a "#", then we're done.
  theword = ReadWord();
  if(theword.Equals("#")) break;

  // Read the pronunciation and put
  //  it in the object
  pronunciation = ReadPronunciation();
  _dictEntry = new DictEntry(theword,
  ➥pronunciation);

  // Read the number of definitions
  numdefs = ReadNumOfDefs();

  // Loop through definitions. For each,
  //  read the part of speech and the
  //  definition, add it to the definition
  //  array.
  for(int i=0; i<numdefs; i++)
  {
    partofspeech = ReadPartOfSpeech();
    definition = ReadDef();
    Defn def = new Defn(partofspeech,
    ➥definition);
    _dictEntry.add(def);
  }
  // We've read all of the definitions.
```

```
  // Put the DictEntry object into the
  //  database
  db.set(_dictEntry);

  // Now insert _dictEntry into the binary tree
  //  and the trie
  mybintree.insert(_dictEntry.TheWord,
  ➥_dictEntry);
  mytrie.insert(_dictEntry.TheWord, _dictEntry);
}

// All done.
// Store the binary tree and the trie
db.set(mybintree);
db.set(mytrie);

// Commit everything
db.commit();
```

Mindehhez persze szükség van néhány kisegít metódusra, amelyek például felolvassák a forrásfájlt, de a dolog logikája a fenti kódból azt hiszem világos. Figyeljük meg újra, hogy valamennyi indexelemet a teljes valójában képesek voltunk letárolni úgy, hogy egyszerűen csak átadtuk a megfelelő gyökérelemet a `db.set()` metódusnak. Az egész tehát egyetlen hívással megoldható!
Ha le akarunk kérdezni valamit az adatbázisból, az már kicsit trükkösebb. Itt már nem kezelhetjük a korábbihoz hasonló mértékben azonos módon az adatbázis objektumokat és az átmeneti adatszerkezeteket. A lemezen tárolt objektumokat be kell olvasni a memóriába, ez pedig egy explicit utasítást igényel. Az első lépés persze most is a `db.get()` metódus meghívása, amivel kikeressük az index gyökérelemét. Akár bináris fát, akár a másik módszert használjuk is az indexelésre, egy szó kikeresése a következőképpen fest kódszinten:

```
public static void Main(string[] args)
{
  Object[] found;
  DictEntry _entry;

  // Verify proper number of arguments
  if(args.Length !=3)
  {
    Console.WriteLine("usage: SearchDictDatabase
    ➥<database> B|T <word>");
    Console.WriteLine("<database> = path to db4o
    ➥database");
    Console.WriteLine("B = use binary tree; T =
    ➥use trie");
    Console.WriteLine("<word> = word to search
    ➥for");
    return;
  }

  // Verify 2nd argument
  if("BT".IndexOf(args[1])==-1)
  {
    Console.WriteLine("2nd argument must be B or
    ➥T");
    return;
  }

  // Open the database file
  ObjectContainer db = Db4o.openFile(args[0]);
  if(db!=null) Console.WriteLine("Open OK");

  // Switch on the 2nd argument (B or T)
  if("BT".IndexOf(args[1])==0)
  { // Search binary tree
    // Create an empty binary tree object for the
    //  search template
    BinaryTree btt = new BinaryTree();
    ObjectSet result = db.get(btt);
    BinaryTree bt = (BinaryTree) result.next();

    // Now search for the results
    found = bt.search(args[2],db);
  }
  else
  { // Search trie
    // Create an empty trie object fore the
    // search
    //  template
    Trie triet = new Trie();
    ObjectSet result = db.get(triet);
    Trie mytrie = (Trie) result.next();

    // Now search for the results
    found = mytrie.search(args[2],db);
  }

  // Close the database
  db.close();

  // Was it in the database?
  if(found == null)
  {
    Console.WriteLine("Not found");
    return;
  }

  // Fetch the DictEntry
  _entry = (DictEntry)found[0];
... <Do stuff with _entry here> ...
```

És akkor most elérkeztünk arra a pontra, amikor megindokolhatjuk, miért is használtuk azokat a bizonyos `db.activate()` hívásokat a keresési metódusokban az 1. és 3. Listában egyaránt.
Amikor meghívjuk a `db.set()` metódust, akkor – amint azt korábban is említettem – a db4o motor automatikusan végigpásztázza a megadott objektum valamennyi függőségét, végighalad az objektumfán és állandóvá tesz valamennyi abban elérhető elemet. Ezt a módszert egyébként ***elérhetőség alapján történő véglegesítésnek*** (***persistence by reachability***) hívják. A fordított irányban ugyanakkor, vagyis amikor a `db.get()` metódust hívjuk meg, hogy lekérjünk vele egy tárolt objektumot,

a *db4o* nem emeli be a teljes objektumfát. Ha ezt tenné, abból az következne, hogy amikor lekérjük mondjuk egy index gyökérelemét, akkor a rendszer a teljes indexet végigolvasná, plusz az összes szótárbejegyzést, plusz az összes meghatározást, és mindezeket beemelné a memóriába. Talán nem nehéz belátni, hogy egyetlen szó kikeresésének nem éppen ez a legcélravezetőbb módja.

Ehelyett a db4o az úgynevezett aktiválási mélység koncepciót használja. Tegyük fel, hogy a `db.get()` metódushívással lekértük az A objektumot, ami így bekerült a memóriába. Ha ezután meghívjuk a `db.ativate(A,6)` metódust, akkor ezzel azt mondjuk a *db4o* motornak, hogy az `A`-t tartalmazó fából szükségünk lesz minden gyermekobjektumra is, mégpedig hatszoros mélységig. Az indexek kezelése során felbukkanó `db.activate()` hívások tehát csupán arról gondoskodnak, hogy mindig bekerüljön a memóriába a szükséges adatmennyiség, amivel a keresés folytatható. (A keresés végén pedig hasonló módon jutunk hozzá a megtalált szótárelemekhez.)

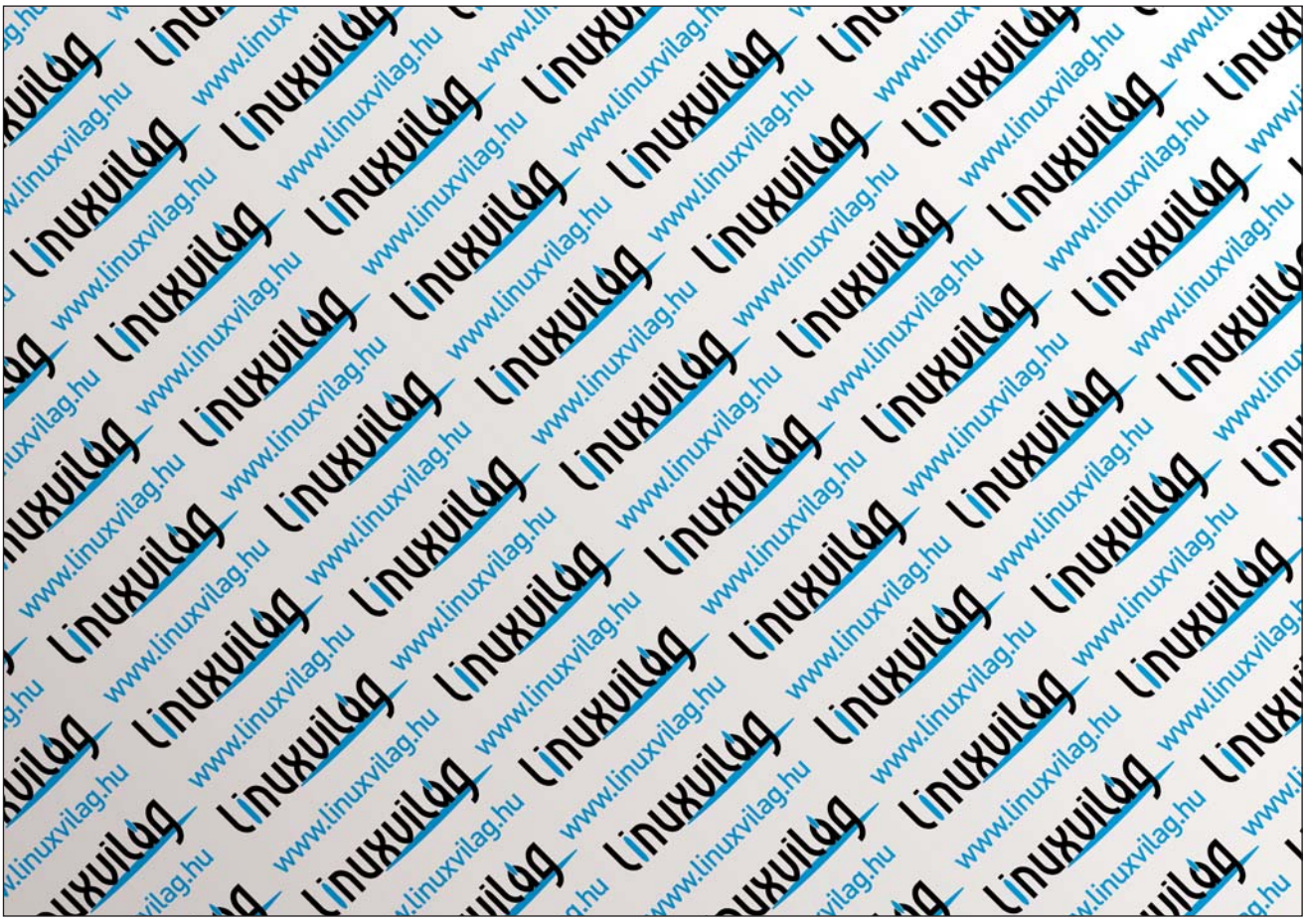
Egyedi indexek

Az objektum-orientált adatbázisok olyan flexibilitást adnak a fejlesztő kezébe, amit az *RDBMS* rendszerekből nem könnyen lehet kicsikarni. Különösen akkor mutatkozik meg ez, ha mély, összetett adatszerkezeteket tervezünk. Ilyenkor csak annyi a dolgunk, hogy a megfelelő hívással letároljuk, állandóvá tesszük a létrehozott objektumot, azzal pedig nem kell törődnünk, miként lehet összeegyeztetni az objektummodellt a relációs modellel.

Ami a cikkben bemutatott db4o motort illeti, ez egy olyan objektum-központú adatbázis-kezelő, ami ráadásul abban sem gátol meg bennünket, hogy egyszerre alakítsuk ki magát az adattartalmat, és a hozzá tartozó indexet. Bár a most bemutatott bináris fa és „gyöngysor” (trie) index nem volt különösebben bonyolult, annak bemutatására kiválóan megfeleltek, hogy a fejlesztő adott esetben ennél sokkal összetettebb indexelési és navigációs szerkezeteket is könnyűszerrel megvalósíthat. A tervezés során tehát létrehozzuk az alkalmazásnak legmegfelelőbb adatszervezési struktúrát, a megvalósítás során pedig nyugodtan dolgozhatunk a jó öreg objektumokkal, akár *Java*-t, akár a *Mono*/*.NET* párost használjuk. A legjobb az egészben azonban az, hogy a *db4o* nyílt forrású, így semmi meg nem akadályozhat bennünket abban, hogy fölhasználjuk a legközelebbi fejlesztésünkhöz. Aki többre kíváncsi, az látogasson a ➲ *www.db4objects.com* webhelyre.

Linux Journal 2006., 142. szám

Rick Grehannak több cikke jelent már meg a Byte, Embedded Systems Journal, JavaPro, InfoWorld, és a Microprocessor Report című lapokban. Társszerzője három könyvnek. Az egyik a távoli eljáráshívásról szól, egy másik a beágyazott rendszerekről, a harmadik pedig az objektum-orientált Java adatbázisokról. Jelenleg minőségbiztosítási vezető a Compuware NuMega Labs-nél.



PHP nyomkövetők (5. rész)

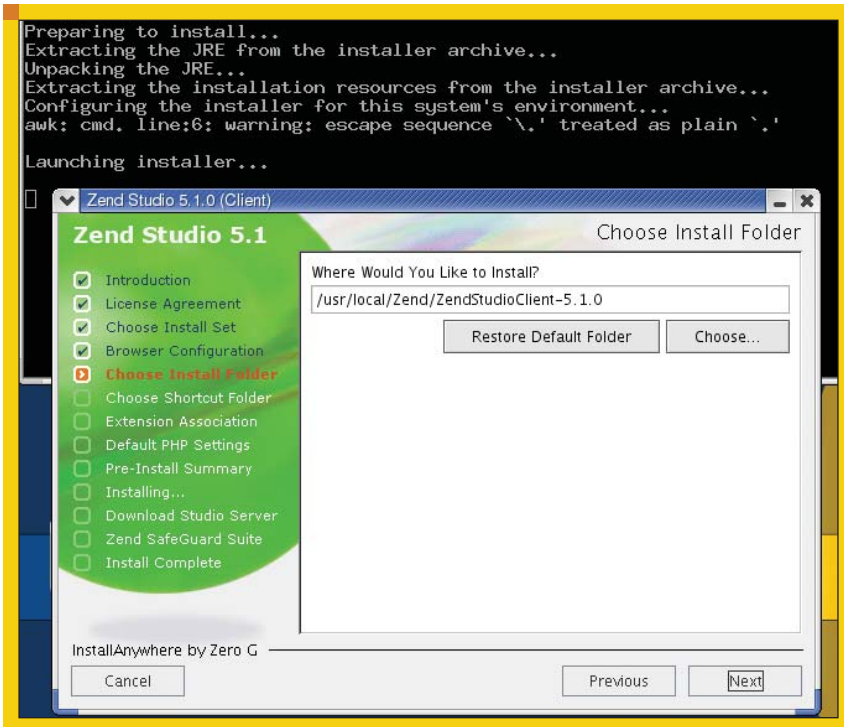
Sorozatunkban áttekintést adtunk néhány PHP nyomkövetőről, a Gubed/Quanta és a Nusphere::PhpED programokról, és néhány Xdebug-ra épülő fejlesztőkörnyezetről (Komodo, TruStudio). A befejező részben a java alapú Zend Studio-t és a hozzá tartozó szervert vesszük szemügyre.

Mi lehet a Zend diadalmenetének oka?

2006 májusában a *Zend Studio* 5.1-s verziója volt a legfrissebb. Ezt vallja magáról: *„A Zend Studio 5 az egyedüli professzionális fejlesztőkörnyezet (IDE), amely magában foglalja a PHP alkalmazások fejlesztésének minden fázisához szükséges komponenseket. A szerkesztés kiterjedt eszközkészletén túl a nyomkövetésen, elemzésen, optimalizáláson át az adatbázis-elérési eszközökig és tesztelési lehetőségekig a Zend Studio 5 felgyorsítja a fejlesztési lépéseket és leegyszerűsíti az összetett projektek szervezését.”* Önmagában ettől nem kell hanyatt esni egy programrendszer előtt, mert ódákat zeng magáról és a legjobbnak vallja magát, hiszen szinte mindegyik hasonló kategóriájú cég ezt teszi.
De mivel a *PHP* értelmező motorja maga is a *Zend* nevéhez kötődik, nem lehet nagyon csodálkozni azon, hogy az őáltaluk kiadott *PHP* nyomkövető kliens és szerver valóban a legjobbak közt van.

Meggondolások, teendők a telepítés előtt

Először úgy próbáltam meg telepíteni a szerver oldali részt, hogy a *php.ini*-ben nem takarítottam el webszerverhez kapcsolódó *Xdebug* modult. Ez aztán problémát okozott, amin utólag nem is nagyon lehet csodálkozni, hiszen valamiképp mindkét modulgyűjtemény módosítani szeretné a *PHP* belső függvényhívásait, így összeakadtak: *PHP Fatal error: [Zend Optimizer] Zend Optimizer 2.6.1 is incompatible with Xdebug 2.0.0beta5.*
A ➲ *www.zend.com/products/zend_studio* oldalról a *„Try it now”* hivatkozásról lehet letölteni a kliensoldali részt, regisztráció után.
(A szervert nem is kell letölteni egyből – ötletes módon a kliens telepítésének egyik lépéseként dönthetünk úgy, hogy azt is letöltjük.)



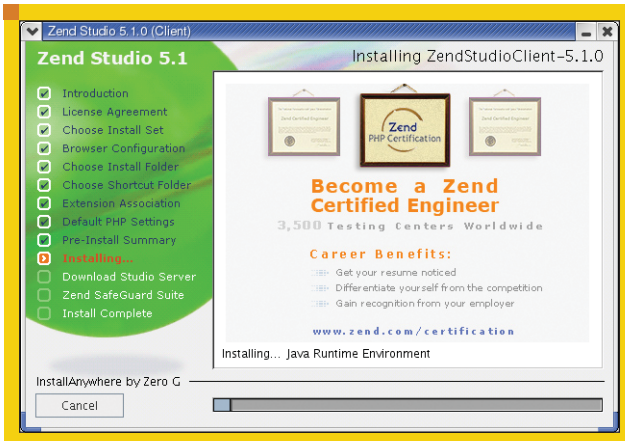
1. ábra Kezdődik a Zend Studio 5 telepítése

A telepítés

A letöltött tarlabdát csomagoljuk ki. Az eredmény egyetlen fájl lesz, melynek eleje héjprogram, hozzá csatoltan pedig bináris adathalmaz kezdődik.
Indítsuk el (*1. ábra*) egy terminálablakból:

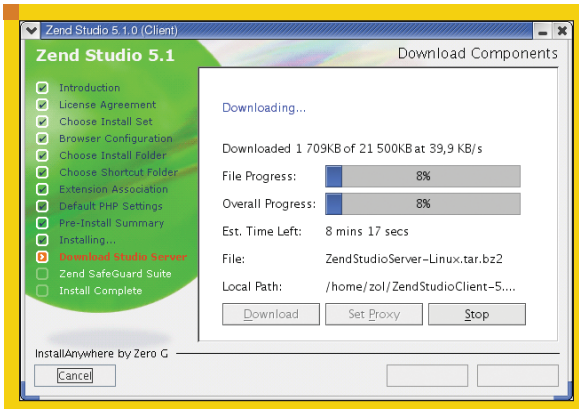
```
./ZendStudio-5_1_0.bin
```

Lépésről lépésre, kulturált módon vezet végig minket a telepítőfelület az egyes fázisokon – láttatni engedi, hogy még mi minden van hátra. Közben informatív ábrák, „hirdetések” kerülnek sorra, amiből többek közt kiderül, hogy oktatási tevékenységet is végez a *Zend* (*2. ábra*), s hogy újra felhasználható programok gyűjteménye is található honlapjukon (*3. ábra*).

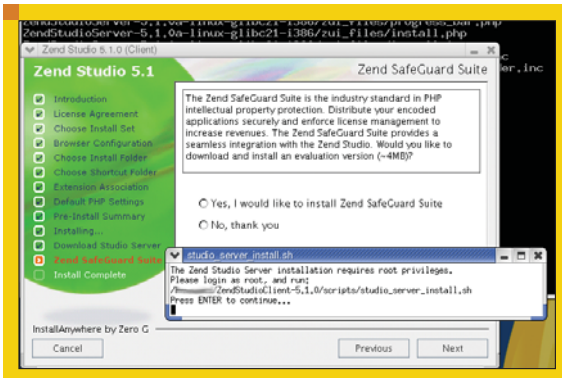


2. ábra A Java futtatókörnyezet is települ – közben buzdítást kapunk oklevél megszerzésére

3. ábra A PHP kézikönyv oldalak telepítése közben PHP alkalmazások gyűjtőhelye villan fel



4. ábra A Zend Studio Szerver részét is le tudjuk tölteni a kliens telepítése közben



5. ábra A szerver letöltése után értesítést kapunk annak telepítési módjáról, jogosultsági kérdéseiről. A főmenüben pedig dönthetünk, szükségünk van-e a Zend SafeGuard-ra.

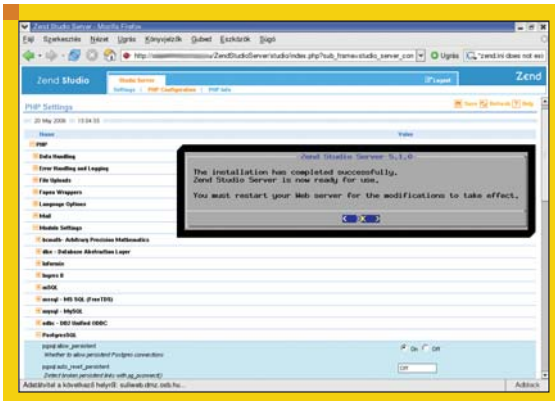
A szerver telepítését általában nem ugyanazon a gépen tesszük, mint ahol a kliensét. Mivel a távol gépre általában *SSH*-n keresztül szoktunk bejelentkezni, kifejezetten előnyös, hogy nem grafikus program próbálja meg felvarázsolni a szükséges *.so* fájlokat és egyebeket a megfelelő helyekre (merthogy lényegében ebből áll a szerver oldali telepítés). Természetesen némi információ *php.ini*-be illesztésére is szükség van, hogy tudassuk a webszerverrel a változásokat; ezt a telepítőprogram automatikusan elvégzi. Amit csak tud, felismer – amit pedig nem tud, megkérdezi, nagyon intelligens módon. Nálam ilyen sorok kerültek beszúrásra a *php.ini*-be:

```
[Zend]
studio.install_dir=/
➥valamigyoker/Zend
zend_debugger.expose_remotely=
➥allowed_hosts
zend_debugger.httpd_uid=48
zend_gui_password=6bc942ea7279f
➥27b08b43fd280a9c15e
zend_gui.ini_modifier=/valami
➥gyoker/Zend/sbin/ini_modifier
zend_debugger.allow_hosts=127.0
➥.0.1/32
zend_debugger.allow_tunnel=127.
➥0.0.1/32
zend_debugger.deny_hosts=
zend_extension_manager.optimizer=
➥/valamigyoker/Zend/lib/
➥Optimizer-2.6.1
zend_optimizer.version=2.6.1
zend_extension_manager.debug_ser
➥ver=/valamigyoker/Zend/lib/
➥Debugger-5.1.0
zend_extension_manager.optimizer
➥_ts=/valamigyoker/Zend/lib/
➥Optimizer_TS-2.6.1
zend_extension_manager.debug_ser
➥ver_ts=/valamigyoker/Zend/lib/
➥Debugger_TS-5.1.0
zend_ini_file=/valamigyoker/Zend/
➥etc/zend.ini
zend_extension=/valamigyoker/Zend/
➥lib/ZendExtensionManager.so
zend_extension_ts=/valamigyoker/
➥Zend/lib/ZendExtensionManager
➥_TS.so
```

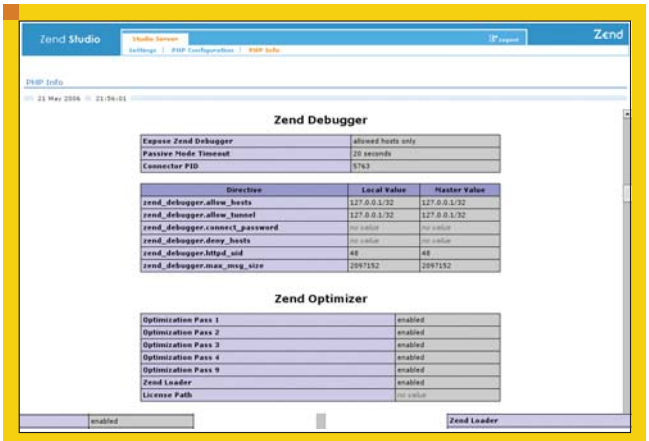
A webszerver újraindítását követően létre kell még hozni az *SSH*-alagutat, hogy működjenek a fenti beállítások:

```
ssh -R 10000:localhost:10000
➥login@szerverneve.hu
```

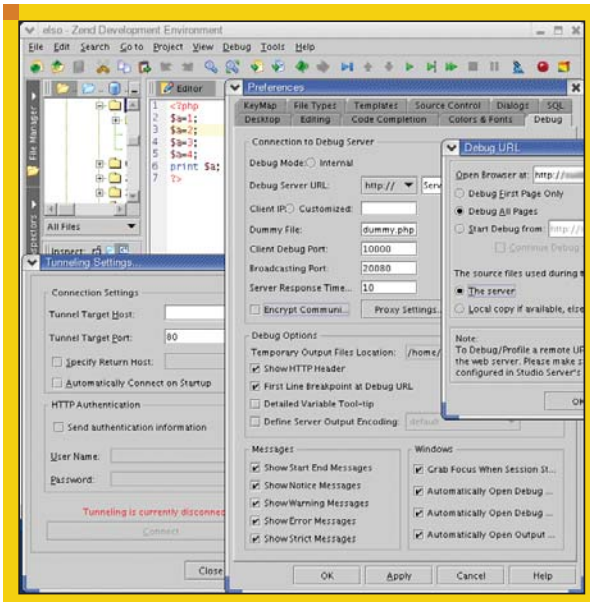
Ezek után már csak a fejlesztőkörnyezet elindítása van hátra. Öröm nézni, mennyi mindenre gondoltak a fejlesztők, hogy megkönnyítsék a *PHP*-programozók dolgát. Lehet egy gombnyomással ellenőrizni, működik-e a kapcsolat a webszerver nyomkövető kiterjesztésével. A piros golyócskára kattintva a böngészőben kapjuk meg az az oldalt, amin a szerver oldali beállítások megnézhetők és változtathatók (*6-7. ábra*) Mint sorozatunk előző részeiben kiderült, a nyomkövetési kérést



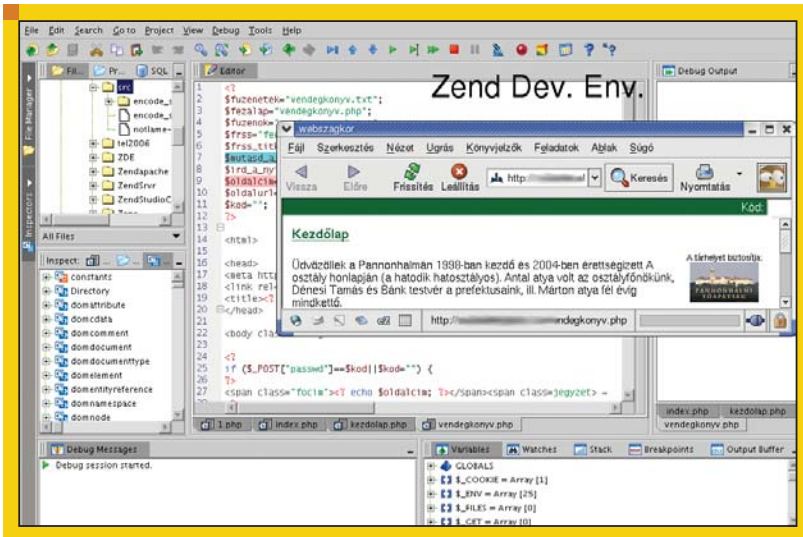
6. ábra A szerver telepítése héjprogramból történik – prózai külalakja ellenére mindent jól ismert fel a rendszer összetevőit illetően. Telepítés után a beállításokat már webfelületen intézhetjük



7. ábra A phpinfo() rámutat a Zend Debugger és más kiterjesztések jelenlétére



8. ábra Fut a ZDE (Zend Development Environment, Zend Fejlesztőkörnyezet) – benne pedig minden, amire csak gondolni tudunk, beállítható



9. ábra Teljes díszben a ZDE nyomkövetés – közben egy böngészőablak is megnyílik

a webböngészőben átadott URL-ből tudja meg a webszerver. A ***Zend Studio*** esetében ez így néz ki:

```
https://szerverneve.hu/egyeb_hely
➥?start_debug=1&debug_port=1000
➥0&use_ssl=1&debug_fastfile=1&
➥debug_host=valami_ip_cim%2C127
➥.0.0.1&send_sess_end=1&debug_n
➥o_cache=1144131487361&start_
➥profile=1&no_remote=1&original
➥_url=https://szerverneve.hu/
➥egyeb_hely
```

Meg kell mondanom, hogy – talán a ***java*** környezet labilitása miatt – fordult elő olyan helyzet, hogy nyomkövetés közben lefagyott a program, mikor egy bonyolult tömbről kértem volna áttekintést. De az is lehet, hogy csak nem volt elég türelmem kivárni az eredményt.
A program amúgy felkészült még a ***SOAP*** webszolgáltatás támogatására, ***SQL*** támogatásra és ***CVS***, ***Subversion***, ***FTP*** beépülésre is, így valóban „minden egyben” webfejlesztő eszköznek tekinthető. Ő jelenti a kivételt az alól a szabály alól, hogy ami mindenre jó, az többnyire semmire se jó.

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.

A repülőmodellezés iskolája

A CRRCSim kitűnő példája, mintegy állatorvosi lova a szabad szoftvereknek: élvezetes, egyszerűségében is célratörő, sokak tudását koncentrálóan professzionális, de a kezdők is könnyedén módosíthatják. Ráadásul komoly anyagi hasznot is hozhat, mert ha időben találkozik vele a modellezőújonc, kevesebb drága pénzen vett valódi repülőmodell bánja a tanulóéveket...

Miért is a repülőmodellezés?

Talán sok kortársamhoz hasonlóan gyerekkorom óra élt bennem a vágy, hogy – ha már én nem tudok repülni – legalább érdemben irányíthassak egy repülő tárgyat (ne csak a tányért a falhoz). Manapság már kiböjtölhető áron elérhetőek rádióirányítású (*rc,* azaz *radio controlled*) repülőgép- és helikoptermodellek, a 10-12 ezer forintos kínai kétpropellerestől kezdve a hatjegyű árat is elérő német és francia gépekig. Ha aztán valaki beleszeret a repülőmodellezésbe, ez a kedvtelés nem kíméli a pénztárcáját, főleg az első időkben, amíg egyik modelltörés a másikat követi. Ha nem valamiféle baráti társasággal vagy egyesülettel kezdjük el a modellezést, hanem egyszerűen csak bemegyünk egy modellboltba, akkor ott a megvásárolt gép mellé ajánlanak ugyan a szerencsétlen (de feltehetőleg vastag pénztárcájú) vásárlónak drága kereskedelmi tanulóprogramokat – az az információ azonban gyakran nem kerül felszínre, hogy szabad szoftver is született e célra, mégpedig nem is akármilyen minőségű.

A repülőmodell-szimulátorok fontosabb vonásai

Nem repülőszimulátorra kell gondolni, hanem kifejezetten repülőmodell-szimulátorra, ahol a program (és a modellek) tervezői azt a célt tűzték ki maguk elé, hogy a valósághű repülőmodell-viselkedést tudjanak produkálni. Mindezt ráadásul többféle modellel, többféle módon paraméterezett „légtérben", többféle helyszínen. Ami a legboldogítóbb az egészben: azzal az irányítóeszközzel lehet mindezt tenni, amivel a tényleges repülőmodellt is lehet vezérelni. Természetesen nem a rádiójelek segítségével, hanem egy kábel révén, amit akár interneten is meg lehet rendelni (magam is így jutottam hozzá fillérekért), vagy el is lehet készíteni megfelelő kapcsolási rajz alapján. Ez a vezérlési élmény egészen más, mint amikor a billentyűkkel vagy valami lövöldözésre kitenyésztett botkormánnyal irányítunk egy játékprogramot. (Természetesen van lehetőség arra is, hogy ezen eszközöket használjuk a szemügyre vett szimulátorprogramban, sőt a *Logitech Dualaction* botkormányra külön héjprogrammal lehet előállítani az indító paramétereket.) *Microsoft Windows* operációs rendszerre is van szabad szoftver e célra, az *FMS* (*Flying Model Simulator*), de az inkább a játékprogram kategóriájába tartozik (habár azt is lehet ilyen módon irányítani). Ezen újság hasábjain inkább egy másikról essen szó, amely talán kevesebb helyszínre van felkészítve, viszont az általa használt nemlineáris mozgásszimuláció (➲ *http://www.szrfk.hu/konf2005/cikkek/bauer_peter_nem_lin.pdf*) és a modellek pontos leképezése elképesztően valósághű állapotteret produkál, és így inkább fizikai szimulációs programnak tekinthető. Ennek neve:

CRRCSim

A program leginkább vitorlázórepülőkre van kihegyezve, de a motoros

1. ábra Egy valódi „Ranger" repülőmodell

2. ábra A modellezés modellezése – eltávolított rádióadó-kristállyal, kábellel

modellek is valósághűen használhatóak. Alapja a ***LARCsim***-en, a NASA ***Langley Research Center simulator***-a. Ennek repülésdinamikai modellje lett átdolgozva, melybe az ***XFOIL*** aerodinamikai tervezőprogram (szabad szoftver) szerzője, ***Mark Drela*** (MIT; ➲ ***http://web.mit.edu/aeroastro/news/magazine/aeroastro-no3/2006drela.html***) belevitte élvonalbeli nemlineáris mozgásszimulációját. A grafikai részeket ***Jan Kansky*** készítette (jelenleg is ő a ***SourceForge***-os fejlesztőhely projektvezetője). Maga a program neve a ***Charles River Radio Controllers*** repülőmodellező-klubtól ered, mely ***Massachusetts***-ben tömöríti tagjait. E klub repülőterei vannak megmintázva a programban is.

Előkészületek

Keressünk rá a ***Google***-ban a ***crrcsim*** szóra. Érdemes az ismerkedést a ***CRRCSim Wiki*** oldalán kezdeni – az a leginkább naprakész. A programot (és az esetleges kiegészítő repülőket – ***crrcsim-addon-models***) a ***SourceForge***-ról tölthetjük le. E cikk írásakor a 0.9.6-os változat a legfrissebb. Elérhetőek a ***CVS*** fejlesztői változatok is, de eddig nem volt szükségem rájuk. Letöltés és kicsomagolás után egy puritán ***make*** paranccsal le lehet fordítani a programot, feltéve, hogy van a gépünkön ***SDL*** 1.2.x (➲ http://www.libsdl.org), ***PortAudio*** 18.1 (➲ ***http://www.portaudio.org***), ***GLUT*** 3.7.x (➲ ***http://www.opengl.org/resources/libraries/glut.html***) és ***PLIB*** 1.8.3 (➲ ***http://plib.sourceforge.net/***). Ha a disztribúciónkban megvannak e csomagok, akkor valószínűleg a -dev(el) változatra lesz szükség. (Megjegyzem, hogy a ***documentation/compile.txt*** szerint

```
make -f Makefile.linux
```

a hivatalos fordítóparancs, de nekem működött a fenti egyszerű módon is.) Fordítás közben kiderül, milyen komoly matematikai és fizikai háttérrel dolgozik a program:

```
...
--- Compiling math/vector3.cpp
--- Compiling math/matrix33.cpp
--- Compiling math/
➥quaternion.cpp
--- Compiling math/ratelim.cpp
--- Compiling math/
➥linearreg.cpp
--- Compiling physics/eom.cpp
--- Compiling math/intgr.cpp
--- Compiling crrc_sound.cpp
...
```

Mivel notebook-omon nincs soros kapu, csak ***USB***, kénytelen voltam kicsit átváltoztatni a programkódot egy

```
perl -pi -w -e 's/ttyS3/
➥ttyUSB0/g' $(grep -ril
➥"ttyS3" *|grep cpp)
```

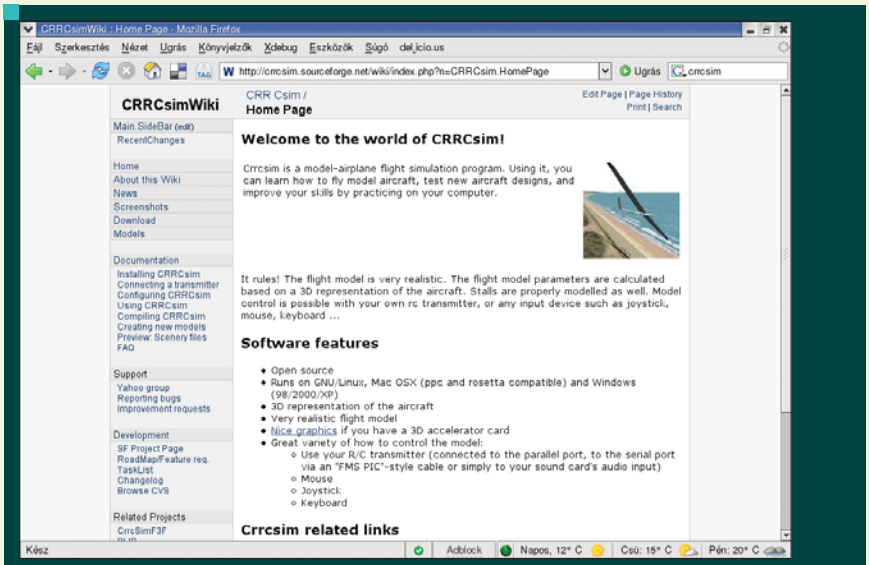
paranccsal. (Egy önéletrajzi adalék: néhány évvel ezelőtt ez volt az első olyan program, amit nem én írtam, de aminek a forráskódjához hozzányúltam, és működött a változtatás. Nyilván nem piszkálja meg az ember egy ***OpenOffice*** forrását, de egy ilyen program szinte arra termett, hogy kísérletezzünk vele.) Ha már előzőleg egyszer lefordítottuk mindenféle változtatás nélkül is a programot (ami melegen ajánlott), akkor a fenti ***PERL*** parancs kiadása után csak a módosított fájlok kerülnek feldolgozásra, azaz csak ennyit fogunk látni:

```
--- Deps: GUI/crrc_ctrlgen.cpp
--- Deps: interface_serpic/
➥tx_interface_serpic.cpp
--- Compiling interface_serpic/
➥tx_interface_serpic.cpp
--- Compiling GUI/
➥crrc_ctrlgen.cpp
--- Linking
```

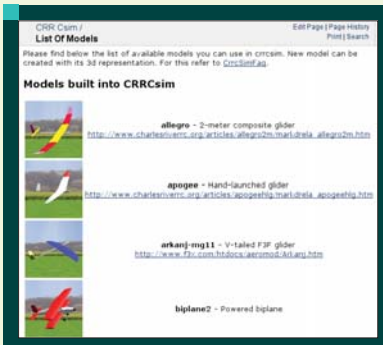
Ezek után (remélhetőleg) ott vár bennünket a `crrcsim` bináris fájl a programkód gyökérkönyvtárában. (***UHU-Linux*** 2.0-ra nem sikerült lefordítanom a hiányzó `freeglut` függőség miatt – viszont ***UHU-Linux*** 1.2-n remekül ment.) Érdemes kiadni egy

```
./crrcsim -h
```

parancsot a használható paraméterek feltérképezéséhez. Igen beszédes a kimenet, sok mindent megtudunk belőle, pl. hogy van-e már saját konfigurációs fájlunk (***~/.crrcsim***), és hogy milyen szél- és termik-viszonyok között fog futni a program. Előfordult velem, hogy – ha nem sikerült valamilyen (többnyire az irányítóeszközre vonatkozó) beállítást eszközölni – törölnöm kellett a ***~/.crrcsim*** könyvtáram tartalmát, és utána már engedett tevékenykedni.



3. ábra CRRCSimWiki

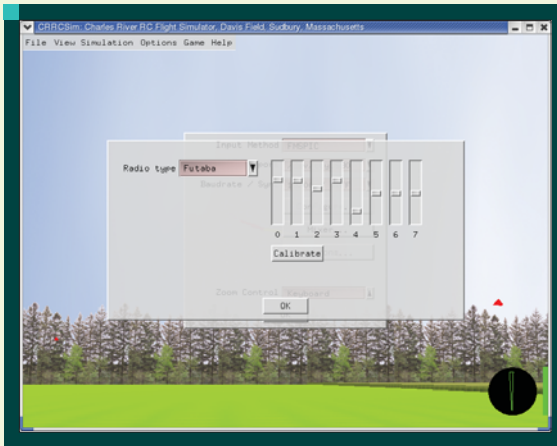


4. ábra Modellek a CRRCSim-ben

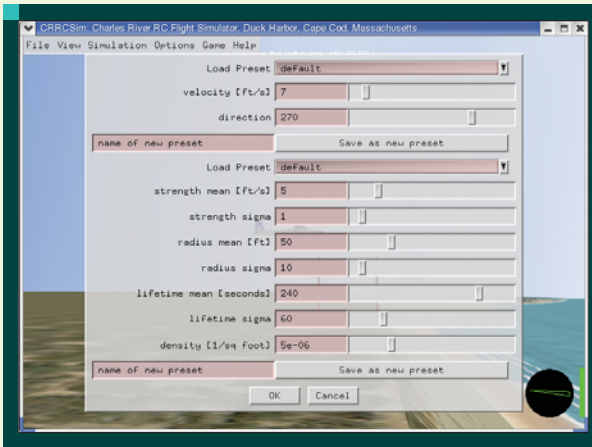
Kezdeti beállítások

Érdemes terminálból indítani a programot, mert – bár lehetőség van indítóikon használatára is – a háttérben nagyon sok hasznos információ megjelenik (angolul):

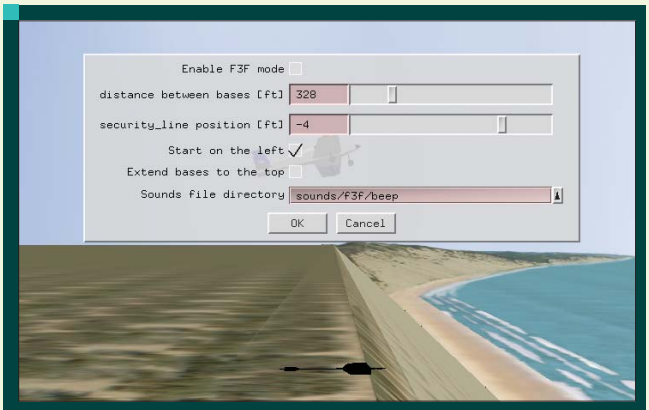
```
Loading default videomode from
➥config...
Using the following rendering
➥mode:
  Renderer:     Mesa DRI
  ➥Intel(R) 830M 20040528
x86/MMX/SSE
  Vendor:       Tungsten
  ➥Graphics, Inc
  GL version:  1.2 Mesa 6.1
  RGBA bpp:     8/8/8/8
  Depth bpp:    24
  Stencil bpp: 8
Opened audio device: 16 bit
➥signed audio (mono) at
➥48000 Hz
Initialized sine samples: 480
Initializing scenery #1
Options chosen:
...
trimmedFlightVelocity =
➥15.8775 m/s
  --- automatic power
  ➥configuration: input ------
------------------
    <automagic F="12.232605"
    ➥V="15.877516">
      <battery
      ➥throttle_min="0.2">
        <automagic T="420" />
        <shaft J="0"
        ➥brake="0">
          <propeller
          ➥D="0.24384"
          ➥H="0.170688" J="0"
          ➥n_fold="-1" />
          <engine>
            <automagic
➥omega_p="2827" eta_opt="0.78"
➥eta="0.7" />
          </engine>
        </shaft>
      </battery>
    </automagic>
  --- automatic power
➥configuration: output -------
-----------------
    <battery throttle_min="0.2"
➥C="4.7812973" U_0="18.379029"
➥U_off="1" R_I="0">
      <shaft J="0" brake="0">
        <propeller D="0.24384"
➥H="0.170688" J="0"
➥n_fold="-1" />
        <engine k_M=
        ➥"0.006176186" R_I=
        ➥"0.11138591" J_M=
        ➥"5.7451026e-05">
          <gearing eta="0.95"
➥J="0" i="1.8801685" />
        </engine>
      </shaft>
      <U_0rel>
    1.00;
    1.00;
</U_0rel>
```



5. ábra Az irányítóeszköz kalibrálása. Piros háromszög: termik

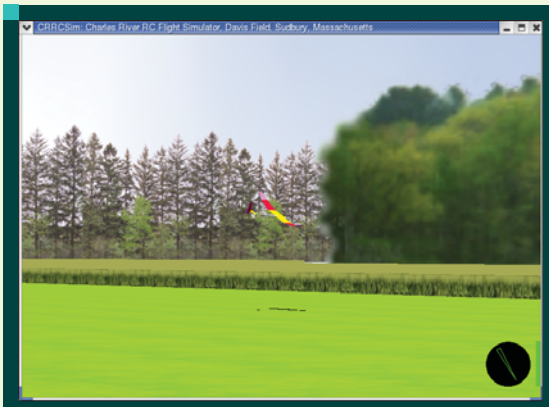


6. ábra Szél- és termik-viszonyok (wind, thermals)



7. ábra Az F3F verseny adatainak beállításai

8. ábra Póznák az F3F versenyben



9. ábra Allegro repülőgép a Davis-i repülőtéren, búzatáblával

10. ábra Zagi repülőgép a frekvenciajelző póznával

```
    </battery>
  --- automatic power
➥configuration: end ----------
-----------------
```

Esc gombbal kérhetjük a menü megjelenését. Itt az ***Options*** menüpont ***Control*** alpontját kell először kérni, hogy az irányítóeszközt megadjuk. Ha valódi repülőmodellhez tartozó eszközzel van dolgunk, akkor egyrészt a típusa beállítható (nálam: ***Futaba***), másrészt annak jelszintjét kalibrálnunk kell (*5. ábra*). Érdemes kivenni a (frekvenciát meghatározó) kristályt a szerkezetből, hogy minimalizáljuk a rádiószennyezést (hátha a környéken valaki épp modellezni szeretne). Egyébként valódi repülőtereken ki is szokott lenni táblázva, hogy ki milyen frekvencián próbálkozik, nehogy egymás gépeit kezdjék el „irányítani”. Ezen kívül a fizikai környezet paramétereit (*6. ábra*), az indítás (***launch***) erőviszonyait is megadhatjuk, illetve repülőgépek és színhelyek közül is választhatunk, valamint a kép- és hangbeállításokat is megváltoztathatjuk.

A vezérlőgombokat is megtudhatjuk a ***documentation/control.txt*** fájlból (pl. a legfontosabbak: ***r*** az újraindításhoz, ***q*** a kilépéshez kell, és ami valahogy kimaradt a dokumentációból: ***d***-vel közelről is megnézhetünk egy-egy modellt, közben ki is próbálhatjuk a vezérlőgombok hatását).

Indulhatunk!

Lehetőség van valódi ***F3F*** póznakerülgető versenyt is szimulálni (*7-8. ábra*). Megvallom, nekem még egyszer sem sikerült szintidőre teljesíteni a verseny követelményeit – van tehát még mit gyakorolnom.
Érdemes kipróbálni többféle repülőgépet, kísérletezni a termikek kihasználásával (melyeket a „***training mode***”-ban meg is lehet jeleníttetni piros háromszögekkel és koncentrikus körökkel). Ami talán az első nagy meglepetés a repülőmodellek irányításakor, az az, hogy „a repülőnek mindig ugyanarra van a jobbra”. Vagyis, ha szembe jön a gép, és készül elbillenni „nekem balra”, akkor hiába mozgatom a kormányt az ellenkező irányba (jobbra), az igazából a repülő számára balra van, azaz még inkább el fog billenni. Emiatt van befestve a repülők szárnyra pirosra/kékre (akár be is lehet ezeket jelölni az irányítóeszközön is), hogy inkább ehhez igazodjunk, mint a jobb-bal érzetünkhöz. Egy idő után ki fog alakulni ez a „relatív jobb-bal” érzék is, egészen biztos.
Sok örömet e program használatához – vezessen ez sokakat ahhoz az elhatározáshoz, hogy egy igazi repülőmodellt is elkezdjenek használni, mégpedig – a szabad szoftverek erejéből – minél biztosabb gyakorlattal!

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.

GnuPlot – Adatok ábrázolása mesteri fokon (1. rész)

Tudományos munkánál, vagy akár egyéb adatok (például hálózati forgalom, processzor terheltség, stb.) feldolgozásánál sokszor többet mond egy jól elkészített grafikon, mint több száz (ezer, vagy még több) sornyi adat. Ezen grafikonok elkészítésében nyújt segítséget a gnuplot. És még sok másban is.

Bevezetés

A *gnuplot* parancssoros eszköz, interaktívan várja a beállításokat és utasításokat, de lehetőségünk van a parancsokat fájlba írni, amit a *gnuplot* (mint egy értelmező) végrehajt. A modernebb, kényelmesebb megoldások kedvelői rengeteg grafikus felületet (GUI) találhatnak az interneten (talán a legjobb a *Qgfe* ➲ http://www.xm1math.net/qgfe/index.html).
A program forrását a ➲ http://www.gnuplot.info címről tölthetjük le, valamint a legtöbb disztribúció csomagként is tartalmazza.

Első lépések

A `gnuplot` parancs (opciók nélküli) kiadása után egy promptot (*gnuplot*>) kapunk. Kilépni az `exit` ill. a `quit` paranccsal lehet.
A program súgója nagyon jól meg van oldva, fastruktúra-szerűen elrendezve (topikok, altopikok, …). A súgót a `help` parancs kiadásával érhetjük el. Az általános információk után az altopikok szerepelnek, amik közül választhatunk. Ezután további altopikok, stb. Egy altopikba közvetlenül is beléphetünk, a `help` parancsot paraméterezve.
Például a

```
help commands plot
```

parancs a *commands* „főtéma” *plot* altémáját „nyitja meg”.

1. táblázat *A Gnuplot leggyakrabban használt parancsai*

`size`	méret
`title`	ábra felirata
`[xyz]label`	tengelyek felirata
`datafile`	bemeneti adatfájl
`[xyz]range`	a tengelyeken ábrázolt tartomány
`style`	ábrázolás módja, például vonalak, pontok, stb.
`border`	keret

A *gnuplot*-tal az ábrákat többféleképpen kezelhetjük: egy ablakban nézegethetjük, vagy fájlba is menthetjük. Nagyon sok formátumot ismer: *png, LaTeX, fig, svg, pdf, postscript, gif,* hogy csak néhányat említsünk. A kimenetet a `set terminal` paranccsal szabályozhatjuk. Például a

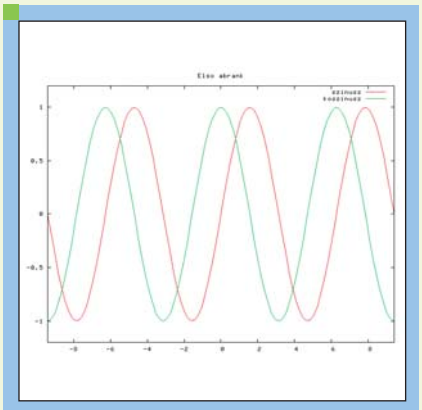
```
set terminal png
```

parancs kimenetként *png* formátumú képfájlt állít be. A fájl nevét a `set output "kepfajl neve"`-vel állíthatjuk be (a fájlnév idézőjelek között legyen!). Az ablakban történő megjelenítést a terminál *x11*-re állításával érhetjük el (egyébként ez az alapértelmezett).

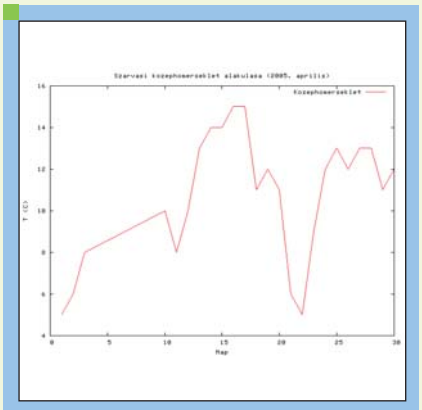
Ábrázoljunk!

Miután beállítottunk egy kimenetet, ideje, hogy megtöltsük tartalommal! Először tekintsük át az általánosságokat! Egy ábrának a konkrét tartalomtól eltekintve sok jellemzője van, például felirat, tengelyek feliratai, méret (pixelben), stb. Ezeket szintén a `set` paranccsal állíthatjuk (talán a *bash* alatti környezeti változóihoz hasonlíthatóak). A beállított értékeket a `show` paranccsal ellenőrizhetjük. Némi (ezirányú) angoltudással és a `help set` segítségével általában könnyedén megtalálhatjuk a nekünk szükséges opciót. Néhány gyakran használt változót az *1. táblázatban* foglaltam össze.
A *gnuplot* nem csak adatfájlokat képes feldolgozni, hanem konkrét függvényeket is ábrázolhatunk. Nézzünk egy konkrét példát:

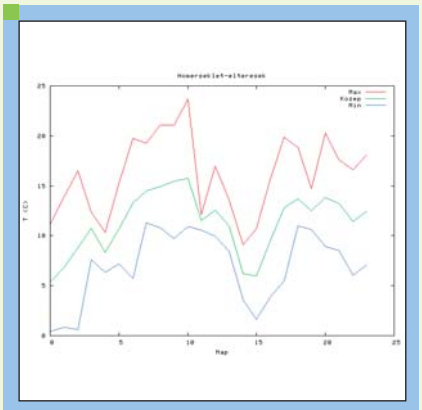
```
set title "Elso abrank"
set xrange [-3*pi:3*pi]
set yrange [-1.2:1.2]
plot sin(x) title
"szinusz",cos(x) title
➥"koszinusz"
```



1. ábra Első ábránk



2. ábra Áprilisi középhőmérséklet



3. ábra Áprilisi hőmérséklet-eltérések

A parancs-sorozat végeredménye az *1. ábrán* látható (ne felejtsük el a kimenetet még a `plot` kiadása előtt átirányítani, ha az ***x11*** nem megfelelő!). A 2. és a 3. sorban látható a ***range***-ek szintaxisa, míg a negyedik sorban a `plot` ábrázoltatja a két szögfüggvényt. Fontos észrevenni, hogy több grafikont egy ábrára úgy rakhatunk, hogy az ábrázolandókat vesszővel választjuk el, valamint egy másik fontos dolog, hogy az ábrázolandók megjelenési és egyéb tulajdonságait külön-külön is szabályozhatjuk, ezt a két `title` opción láthatjuk. Persze ezeket még csicsázhatjuk, és bonyolultabb függvényeket is ábrázolhatunk.
Ez mind nagyon szép és nagyon jó, de ezt bármilyen függvényábrázoló programmal megtehettük volna. Mégis miben nyújt többet a ***gnuplot***?

Adathalmazok ábrázolása

Nagyon sok és érdekes adat található különféle meteorológiai adatbázisokban, amelyek kiáltanak azért, hogy ábrázoljuk őket. Egy ilyen adatsort a ➲ http://www.mvk.tsf.hu/Meteorologia/met.htm címen is találhatunk, havi csoportosításban, azon belül pedig napi lebontásban (Prjevara Zoltán és Tanszékének munkája). A lementett ***HTML***-fájlokból az adatokat a következő paranccsal kinyerhetjük, amelyet már feldolgozhatunk a ***gnuplot***-tal:

```
w3m -F 2005-apr.htm | grep
➥-E "^[1-3]" | sed "s@\ \ *@\
➥@g" | sed "s@,@\.@g" >
➥2005-apr.txt
```

A `w3m` egyszerű szöveges kimenetet állít elő, a `grep` csak azokat a sorokat listázza, amelyek 1-gyel, kettővel vagy hárommal kezdődnek, az első `sed` az egymás utáni szóközöket egy szóközzel helyettesíti, a második `sed` pedig a (tizedes)vesszőket cseréli (tizedes)pontokra.
Ezután a következő parancssorozattal a *2. ábrán* látható grafikon állítható elő:

```
set title "Szarvasi
kozephomerseklet alakulasa
➥(2005. aprilis)"
set xlabel "Nap"
set ylabel "T (C)"
plot "2005-apr.txt" with lines
➥title "Kozephomerseklet"
```

Az első három sornak az eddigiek alapján érthetőnek kell lennie. A negyedik sor se szorul sok magyarázatra: ezúttal nem egy függvényből vesszük az adatokat, hanem egy fájlból, ez a `"2005-apr.txt"` rész. A `with` kulcsszó állítja be a jellemzőket: a `lines` résszel azt kérjük, hogy a diszkrét pontokat kösse össze vonallal, a `title` pedig a jelmagyarázat szövegét állítja be.
Jogosan merül fel az a kérdés, hogy a napi maximum- és minimumhőmérsékletek mennyire térnek el a napi középhőmérsékletektől. Nyilván végig lehetne böngészni és kiszámolgatni az eltéréseket, de sokkal szemléletesebb, ha szintén egy grafikonon ábrázoljuk a három mennyiséget.
Azt tudjuk már, hogy hogyan lehet egyszerre több grafikont ábrázolni egyszerre (vesszővel elválasztjuk), de hogyan közöljük azt, hogy az adatfájlból melyik oszloppal kívánunk dolgozni?

```
set title "Homerseklet-
➥elteresek"
set xlabel "Nap"
set ylabel "T (C)"
plot "2005-apr.txt" using
➥($3) with lines title "Max" ,
➥"2005-apr.txt" using ($2)
➥with lines title "Kozep" ,
➥"2005-apr.txt" using ($4)
➥with lines title "Min"
```

A fentebbi kódsorban egyedüli ismeretlen kifejezés a `plot` sorában szereplő `using`. Ez utasítja a ***gnuplot***-ot, hogy pl. a harmadik oszlopot „használja” (`using ($3)`). Nem ezen cikk feladata az adatok kiértékelése, de azért egy-két dolgot érdemes észrevenni: április 11-én nem volt nagy hőmérséklet-változás, viszont 10-én nagy volt az ingadozás.

Összefoglalás

Az eddigiek a ***gnuplot*** lehetőségeinek csak kis részét mutatták be, de már ebből is látható, mi mindenre használható. A következő részben a hőmérséklet-ingadozást még szemléletesebbé tesszük, valamint egyéb ábrázolási módok is előkerülnek.

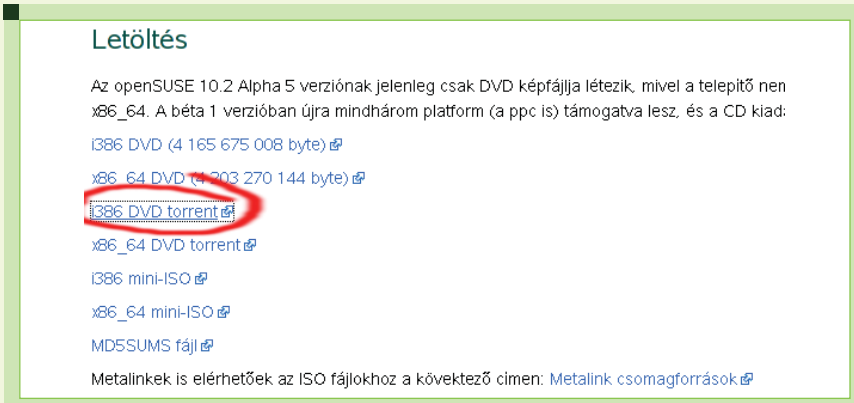
Udvari Zsolt
(udvzsolt@gmail.com)

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.

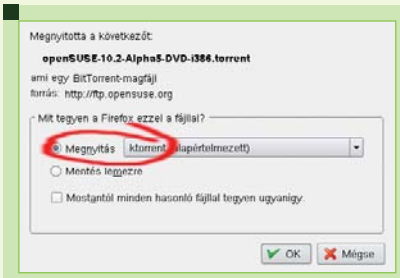
KDE alkalmazások (7. rész)
KTorrent – A KDE torrent kliense

A BitTorrent korunk egyik leghatékonyabb fájlcserélő protokollja, amelynek sok tucat programbeli megtestesítője akad a hálózaton. A KTorrent 2.0.x nemsokára a KDE hivatalos BitTorrent kliense lesz, ugyanis jelenleg még nem része a KDE terjesztésnek.

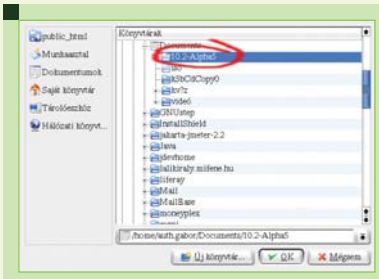
A *BitTorrent* kliensek egyik felhasználási területe (mondhatnánk, hogy a nagyobbrészt) az illegális fájlcserélgetés, ahol is zenék, videók vagy fizetős (esetleg tört) programok „cserélnek gazdát”.
A *BitTorrent* másik – a *Linux* világában fontosabb – területe a különféle disztribúciók és nagyobb programok letöltése (és feltöltése).
A *BitTorrent* működésének lényege, hogy a kliens *mindenképpen* fel is tölt a letöltött állományból, így minél többen akarnak egy állományhoz hozzájutni, annál több és több megosztó lép be, s ez egyre csak növeli a megosztott sávszélességet. Ez a jelenség nagyon jól jön akkor, amikor egy-egy *Linux* fejlesztő társulat kiad egy újabb verziót ISO formátumban, amelyet ezrek vagy akár milliók akarnak letölteni a megjelenés pillanatában: a tükörszerverek és a kiadást készítők szervere tehermentesítődik, és az izgatott telepíteni-frissíteni vágyók tömege egymástól tölti le a nagy állomány kis darabkáit.
A legegyszerűbben úgy tudjuk a *KTorrent* programot elindítani, hogy a böngészőprogramban rákattintunk egy *BitTorrent* fájlra mutató linkre.
A link által mutatott állomány információkat tartalmaz arról a nagyobb állományról, amelyet le szeretnénk tölteni (másoktól).
Előfordulhat, hogy a böngészők megkérdezik, hogy mivel nyissák meg a kérdéses állományt, ha a listából kiválasztjuk a `ktorrent` programot, akkor nagy hibát nem véthetünk. Érdemes megjegyeztetni ezt a műveletet a böngészővel, *Firefox* esetén ez a *„Mostantól minden hasonló fájllal tegyen ugyanígy”* megjelölésével történik meg.
A program indulása után ki kell választanunk a letöltés helyét, s akár új könyvtárat is létre tudunk hozni az *„Új könyvtár...”* nyomógombot használva. Érdemes olyan helyet választani, ahol van annyi szabad hely, amennyi szükséges, ugyanis kellemetlen meglepetések érhetnek, ha nagyon lecsökken a megadott fájlrendszeren a szabad kapacitás. Egyrészt a *KTorrent* beszünteti a letöltést, másrészt a különféle egyéb programok változatos hibákkal elpusztulnak (legalábbis én még nem láttam olyan hibaüzenetet *KDE* alatt egy programtól sem, hogy nincs elég hely a lemezen, valaki látott már ilyet?).



1. ábra Egy BitTorrent link

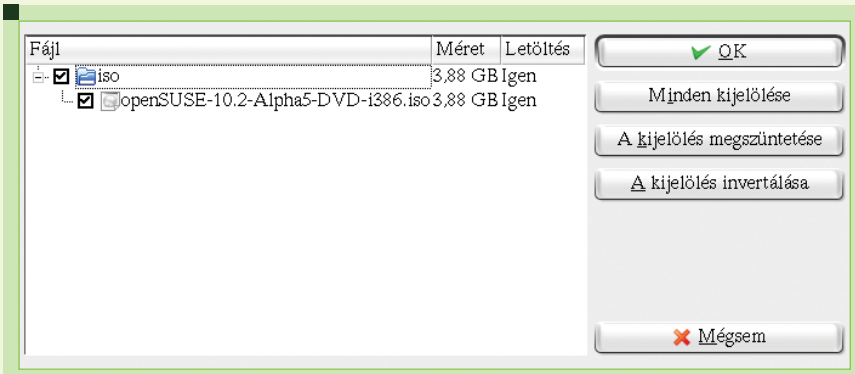


2. ábra Firefox bizonytalan

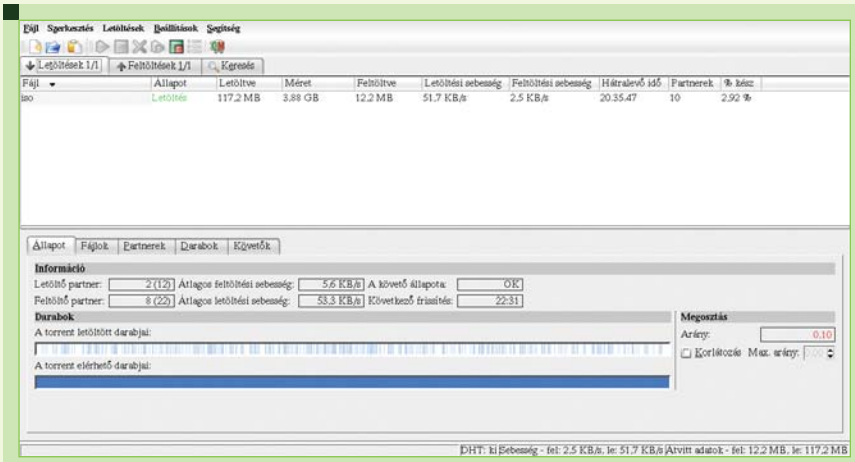


3. ábra A letöltési hely

A kiválasztott ***BitTorrent*** leíró tartalmazhatja több állomány leírását is, így előfordulhat, hogy mindegyikre nincs szükségünk, illetve valamelyiket esetleg már letöltöttük valamikor régebben, vagy nincs is rá igazán szükségünk. Mindezen lépéseken túljutva a program azonnal megkezdi a letöltést, amelyet a főablakon tekinthetünk át. Ennek az ablaknak három fő része van: a menü és az ikonsor, a letöltések, illetve különféle információk.
A menüben a szokásos dolgokat találhatjuk, kivéve, hogy nincs mentésre utaló menüpont, mert a ***KTorrent*** célja a folyamatos „mentés”. Ellenben kétféle módon tudunk megnyitni egy ***BitTorrent*** leíró állományt: fájlból közvetlenül vagy vágólapból beillesztéssel (illetve a programnak indításkor átadva paraméterként, például böngészőből indítva). A letöltéseket a videón megszokott jelölésekkel tudjuk vezérelni (elindítani, megállítani, stb).
A program által kezelt letöltéseket egy listában láthatjuk, ahol különféle információkat (letöltött mennyiség, feltöltött mennyiség, sebességek, idők, százalékok, stb) láthatunk, illetve bármelyik oszlop szerint rendezhetjük a letölteni valót.
Az információs részen mindig az éppen aktuális – vagyis kiválasztott – fájlról kapunk információkat.
A legfontosabb információ, hogy minden darabja meglegyen a ***BitTorrent*** hálózatban, különben soha nem tudjuk a fájlt egészében letölteni. Ezért aztán a ***„Torrent elérhető darabjai”*** alatti mezőnek végig sötétkéknek kell

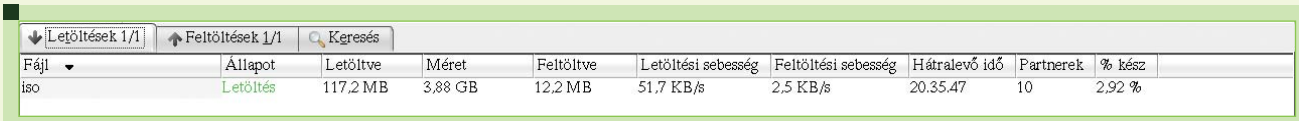


4. ábra Mit töltsünk le?

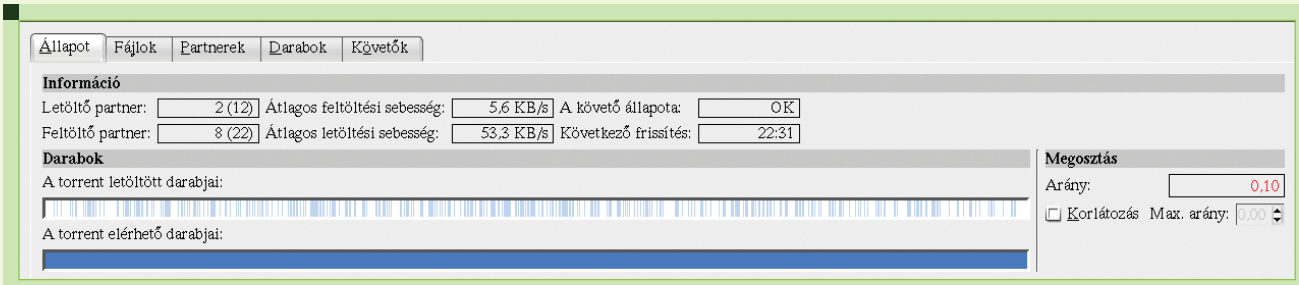


5. ábra Főablak

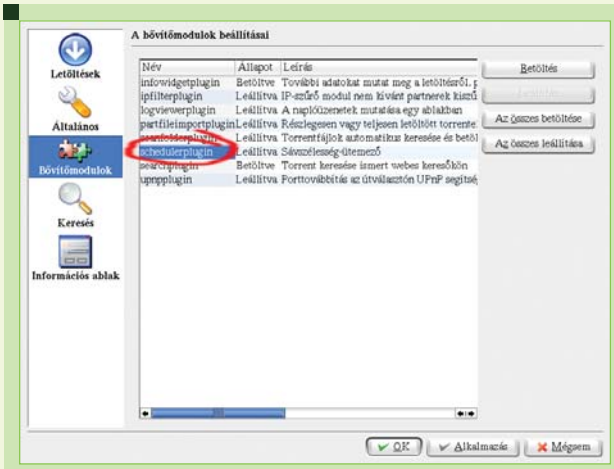
6. ábra Menü és ikonsor



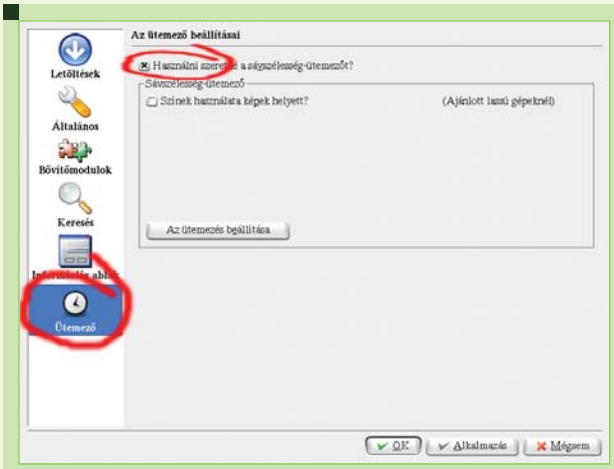
7. ábra Letöltések



8. ábra Információk

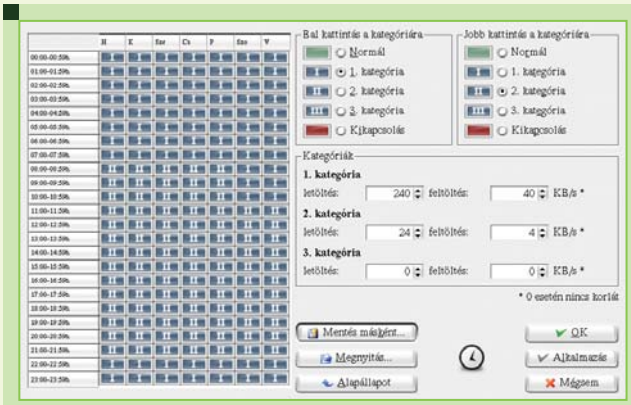


9. ábra Letöltés ütemező



10. ábra Az ütemező bekapcsolása

lennie, nem lehet benne egy apró pici fehérbe hajló vonalka sem (különben járhatunk úgy, hogy egy nagy nehezen majdnem letöltött *DVD ISO* utolsó egy darabkája hiányzik... Ilyenkor az ember megtudja, milyen az igazi bosszúság...
Ugyanezen a helyen találjuk a letöltött darabkákat, amelyek egy – kezdetben teljesen fehér – mezőben jelennek meg kék színű vonalkákként. Minél sötétebb kék egy vonal, annál több és több darabkát töltöttünk már le. Természetesen minél több darabkánk van meg, annál több és több felfelé irányú kérést szolgál ki a gépünk, így járulunk hozzá a közös sávszélességhez. Megtehetjük a beállítások között, hogy korlátozzuk (esetleg meg is szüntetjük a feltöltést), azonban gondoljuk át ezt jobban, különben úgy járunk, mint *James Fenton Kínában*, akinek lámpa volt a biciklijén, s egy rendőr erősen kérdőre vonta: „Mi lenne, ha mindenki ezt csinálná?!"
A többi információ szükségessége viszonylagos, általában a tizedik letöltésnél már senki nem nézi csodálkozó szemekkel az ugráló számokat és a változó színeket: megvárja, amíg a program a háttérben szépen befejezi a letöltést.



11. ábra Ütemezés beállítása

Beállítások

A program egyik legfontosabb menüpontja a beállítások nevet viseli. Ennek szinte legfontosabb pontja az *„Általános"* fülön az *„Ikon megjelenítése a paneltálcán"* nevet viseli, és érdemes bejelölni, ekkor a programot bezárva továbbra is futni fog, de nem foglal értékes munkaterületet.
További érdekességeket találhatunk a *„Bővítőmodulok"* fül alatt találhatunk, ahol a leginkább népszerű modul a *„Letöltés ütemező"* névre hallgat. Aktiválni kijelöléssel, majd a *„Betöltés"* gombra kattintva tudjuk.
A modul betöltése után megjelenik egy ütemező fül, amelyre kattintva elindíthatjuk az ütemezőt a *„Használni szeretné..."* kezdetű sort bejelölve.
Az ütemezés pontos paramétereit az *„Ütemezés beállítása"* gomb segítségével megnyíló ablakban tehetjük meg, amely első ránézésre eléggé komplikált kinézettel büszkélkedik. Az összetettség azonban csak álca, hamar megejthetjük a beállításokat. Először is a bal és a jobb egérgombra jelöljük meg az ablak jobb felső részén lévő I és II kategóriákat. Ezek után állítsuk be a kíván maximális le- és feltöltési sebességeket az egyes kategóriákhoz. Végül a bal illetve jobb egérgombot nyomva tartva rajzoljuk meg a letöltési sebességek térképét, aholis az oszlopok a hét napjait, a sorok pedig a nap 24 óráját takarják. Minden metszéspontban lévő cella egy kategóriát tartalmaz, vagyis abban a cellában (például Hétfő 7:00-7:59) a meghatározott kategória szerint fog a letöltés folyni.
A szabványos munkásemberek általában az éjjeli letöltés felel meg a kívánalmaknak, amikor is pihennek, napközben pedig csak egy minimális internet szeletet adunk a *KTorrent* számára, éppen csak ne haljon éhen, de ne is akadályozza a munkánkat vagy egyéb szórakozásunkat.

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

KTorrent projekt oldala:
➲ http://www.ktorrent.org/

LOAD"*",8,1 avagy a VICE emulátor

Sokan vannak közülünk, akik a számítógépekkel való ismerkedést valamilyen Commodore gyártmányú gépen kezdték. Egykor sok általános iskolában is találhattunk ilyeneket. Manapság már szinte kuriózumnak számít, ha valakinek még van egy 64-ese vagy egy 128-asa otthon. Cserébe viszont szinte minden régi géptípushoz találunk emulátorokat.

A Commodore gépei

Minden idők legsikeresebb számítógépe az 1982 augusztusában megjelent *Commodore 64*-es volt. Mire 1994-ben abbahagyták a gyártását, egyesek szerint 17, mások szerint 25 millió példány is elkelt belőle. A gép az akkori árához képest kimagasló képi és hangteljesítményt adott, megelőzve az *IBM* hasonló időben megjelent modelljeit.
A *Commodore 64*-es számítógépben használt *VIC-II* videó áramkört eredetileg játékgépekbe tervezték, ez már 1981-ben megjelent. A cég egyszerűen gazdasági okok miatt használta fel ezt az áramkört többféle modelljében is.
Ehhez még *light pent*, úgynevezett *fényceruzát* is lehetett csatlakozni, amellyel ha rámutattunk a tévé képernyő valamely pontjára, a gép meg tudta határozni a pont koordinátáit. Ezt a játéktermekben lövöldözős játékprogramokhoz használták, ahol az embernek a zseton bedobása után egy műanyag pisztolyt kellett a kezébe vennie, és céloznia, mint egy igazival.
Ugyanez elmondható a *SID* névre hallgató szintetizátorról, amelyet neves együttesek is használtak több zeneszámukban. A *Commodore 64* ennek eredményeképpen legalább olyan alkalmasnak (ha nem alkalmasabbnak!) bizonyult játékok készítésére, mint egyéb alkalmazások futtatására.
Természetesen irodai, vállalati feladatokra már nem használja senki ezeket a gépeket. Ellenben az otthoni felhasználók közül néhányan rendelkeznek még ilyennel. Vagy ha nem is, több emulátorprogram létezik, amelyek a mai *PC*-ken, virtuális gépként futtatni képesek az egykori programokat. Ezek az emulátorok ráadásul még több lehetőséget is nyújtanak, mint az eredeti gépek, például lementhető velük a memória tartalma, és így bármelyik játéknál lehetővé válik a visszatöltés. Ilyen emulátor program a *VICE* is, amely az első *Unix* típusú környezetre írt ilyen célú alkalmazás volt. Jelenleg talán ez a legnagyobb tudású, és egyben a legpontosabb imitációja az eredeti hardvernek.
Érdekes, hogy ezeknek a *PC*-s emulátoroknak a megjelenésére viszonylag sokat kellett várni, és erős gépet igényelnek. Elgondolkodtató, hogyan lehet az, hogy egy régi, 8 bites számítógépnek, amely processzora alig rendelkezett 100 utasítással, órajele nem haladta meg az egy megahertzet (a pontos érték 985 kHz!), amelynek memóriája mindössze 64 kByte-os, az emulátorban miért van mégis szüksége *Pentium* kategóriájú számítógépre. A válasz az előbb említett kép és hangkeltő áramkörök bonyolultságában rejlik.

A VICE emulátor

A *VICE* (*Versatile Commodore Emulator*) a cikk írásának időpontjában az 1.20-as verziónál tart, és nem csak *Linuxra* és egyéb *Unixokra*, hanem *Windowsra*, *BeOS-re*, *OSX-re* és egyéb operációs rendszerekre is rendelkezésre áll. Emulálni képes a *C64*-et, *C128*-at, *VIC20*-at, a *PLUS4*-et és még további típusokat is.
A legfrissebb verziója mindig letölthető a cikk végén található címről. Több disztribúció tartalmazza, de még ha a csomagkezelővel telepítjük is, általában utánajárást igényel. Ugyanis a gépek emulálásához szükség van az eredeti *ROM*-okra, és ezeket több terjesztés jogvédelmi okok miatt inkább kihagyja a csomagból. Ezért érdemes inkább forráskódból telepíteni a programot, különösen, mert az 1.20-as verzió már támogatja a *GTK+*-t és a *Gnome*-ot:

```
tar xzf vice-1.20.tar.gz
cd vice-1.20
./configure --enable-gnomeui
make
make install
```

GTK+ nélkül a program az eredeti *Xaw* elemkészlettel indul, amely elég csúnya és nehézkes. Ha ezt használjuk, ne lepődjünk meg, a menü ilyenkor nem látszik; a képre a jobb, illetve a bal oldali egérgombbal kattintva tudjuk előhozni azt. Az emulátor a használni kívánt gép típusától függően az `x64`, `x128`, `xpet`, `xvic` vagy `xplus4` parancsokkal indítható.
Az eredeti programok közül a legtöbb lemezkép formátumban már letölthető az *Internetről*, általában *.d64* kiterjesztéssel. A fájltípusok a következők:

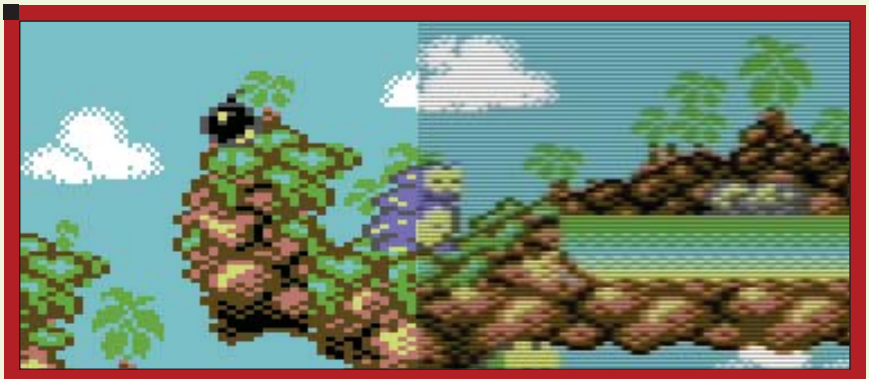
- *.d64* – ez egy hajlékonylemez egy oldala.
- *.t64* – szalagra mentett program.
- *.crt* – *cartridge* program. Ez hasonlóan működött, mint egy *Nintendo* játék, csak be kellett dugni a gép hátuljába, és bekapcsolni azt.
- *.p00* vagy *.prg* – ez is egyetlen program.

A VICE lehetőségei

Játék indításához a *File* menü *Autostart* pontját érdemes kiválasztani. A párbeszédablakban nem csak a kiválasztott lemezkép nevét láthatjuk, hanem annak tartalmát is, így ha a fájlunk több programot is tartalmaz, azok közül is választhatunk. Lemezkép fájlt megadhatunk a programnak indulási paraméterként is:

```
x64 diskimage.d64
```

Ebben az esetben az első megtalált programot fogja elindítani a *VICE*.
Ha az azon található program indítása nélkül szeretnénk csak betenni egy lemezt a virtuális meghajtónkba, akkor használhatjuk a *File* menü *Attach disk image* pontját. A régi gépeken némelyik játék nem fért el egy lemezre, és menet közben kellett kicserélni vagy megfordítani. Ilyenkor érdemes ezt használni. Egyébként itt is érződik, hogy a legtöbben játékra használták a *Commodore* gépeket, ugyanis a *VICE* programnak is van több ezzel kapcsolatos kényelmi szolgáltatása. Betölthetjük például a cserélendő lemezképfájlok listáját (*fliplist*), és egyszerűen, gombnyomással válthatunk közöttük.
A *Snapshot* menü segítségével az emulátor, és a benne futó program aktuális állapotát tudjuk lemezre menteni, illetve lehetőség van akár a játékunk rögzítésére is (*Start recording events*). Ilyenkor a *VICE* rögzíti minden billentyű lenyomásunkat, az összes botkormány mozgatást, és később visszanézhetjük a játékunkat. Több játéknak is található így a *Neten* megoldása is, mint pl. *Arno Weber* oldalán a *Boulder Dash* összes pályájának végigjátszása.
Az *Options* menüben az emuláció paraméterei állíthatóak. Lehetőség van például a virtuális gép gyorsítására is. Persze csak ameddig az igazi gépünk bírja... Ez kényelmes, ha a hosszú töltögetési időket szeretnénk átugrani. (*Autostart* használata esetén ez egyébként automatikus.) Itt kapcsolhatjuk be különböző perifériák, kiterjesztések emulációját is, például az egeret vagy a *RAM* kiterjesztéseket. Fontos beállítás a *True Drive Emulation*, amelynek használata esetén az eredeti lemezmeghajtó teljes működése megvalósul, erre egyes programok esetén

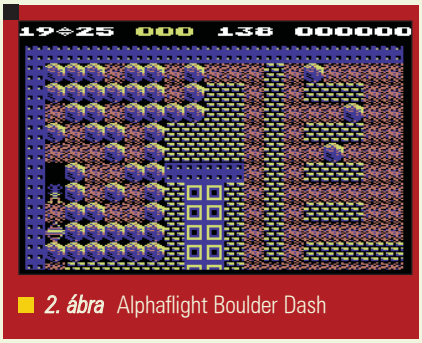


1. ábra Képrészlet simán és PAL emulációval

– általában amelyek gyorstöltőket használtak – szükség lehet. (Érdekesség, hogy ezeknél a gépeknél a lemezmeghajtó tulajdonképpen egy majdnem akkora tudású számítógép volt, mint maga az alapgép. Még akár ahhoz is lehetett *RAM* bővítést kapni!)
A *Settings* menüben további beállításokat találunk. Itt állíthatjuk be, hogyan szeretnénk az eredeti botkormányokat emulálni, lehetőség van a billentyűzet (leginkább a numerikus része) használatára, vagy akár egy igazi botkormány csatlakoztatására is. A *SID* beállítások módosításával megadhatjuk a hangminőséget. Ez az egyik olyan dolog, amivel igazán meg tudjuk izzasztani a gépünket: a legjobb minőség esetén az emuláció egy 1,5 MHz-es *Celeron M* gépen akadozik!
A *VIC-II* beállításai közül a *sprite* ütközéseket érdemes kiemelni (*Sprite-sprite* és *sprite-background collisions*). A *sprite-ok* kis 24x21 képpontból álló figurák voltak, amelyeket szabadon lehetett elhelyezni a képernyőn. Egyszerre legfeljebb nyolc lehetett belőlük (vagyis hát a leleményes programozók kitaláltak trükköket arra, hogyan használjanak még többet). Ez önmagában is hatalmas segítség volt a játékok írásához; hát még ha hozzávesszük ehhez azt a lehetőséget, hogy ha ezek a figurák ütköztek egymással, vagy a háttérgrafikával, azt jelezni tudta a *VIC-II*! A játék feladatainak nagy részét szinte nem is a processzor végezte, hanem a „videókártya”, ahogyan az a mai *3D*-s játékoknál is történik... Egyébként a mai videókártyák nem is olyan rég óta támogatják a lehetőséget, hogy egyáltalán az egérmutatót ilyen *sprite-ként* kezelhesse az operációs rendszer!
A *collisions* beállításokkal ezeknek az ütközéseknek az érzékelését kapcsolhatjuk ki, akár bizonyos játékokban könnyítési lehetőségként. Így már tényleg érthetővé válik, miért kell gyors gép az emulációhoz: tulajdonképp nem is egy, hanem három-négy processzor munkáját is el kell végezni.
Legérdekesebb képi beállítás a *PAL Emulation*, amelyet engedélyezve a *VICE* megpróbálja imitálni az eredeti, tévére kötött számítógép képét. A rajzok kicsit elmosódottak, az egymás melletti kontrasztos színek között apró csík jelenik meg... Mindenképp érdemes kipróbálni!
Egyesek szerint egy modern számítógép amúgy sem jó másra, minthogy egy emulátort futtassunk rajta. Akinek még megvan az eredeti gépe, vagy ha más nem is, egy eredeti botkormány, és nem riad vissza egy kis forrasztgatástól, még azt is megoldhatja, hogy csatlakoztatni lehessen azt a PC játék portjára. Ez egy egyszerű áramkörrel megoldható, amely elfér akár egy gyufásdobozban is. A kapcsolási rajz megtalálható az *ePanorama.net* oldalán.

Egy kis nosztalgia

Mert meg kell hagyni, nagy felbontású *3D* grafika ide vagy oda, kormányos, pedálos, sok gombos játékvezérlők, mp3-ból lejátszott zene és millió egyéb dolog... A régi gépeken a készítők inkább a játék hangulatára koncentráltak. Ki ne emlékezne *Peter Liepa* programjára, a *Boulder Dashra*, ahol a világ leghíresebb hangyájával, *Rockforddal* kellett köveket tologatni, gyémántokat gyűjteni, és a szörnyek elől menekülni! Vagy épp az *Impossible Missionre*, amelyben *Elvin Atombender*, a gonosz professzor tornyában, gyilkos robotok között ugrálva kellett jelszavakat keresni és megfejteni. Nem is beszélve

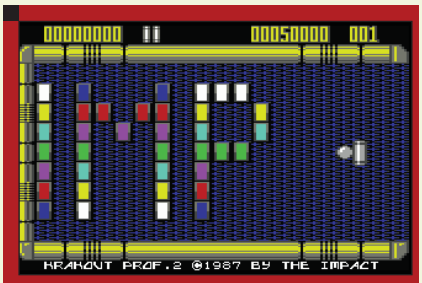


2. ábra Alphaflight Boulder Dash

3. ábra Rátkai átkai: Új vadnyugat

4. ábra Wizard of Wor

5. ábra Bubble Bobble



6. ábra Krakout

Rátkai István gépelgetős kalandjátékairól, a ***Bosszúról*** vagy az ***Új vadnyugatról***, amelyeket mindenki csak ***Rátkai átkaiként*** emlegetett. Ezekben a hősünket igen kezdetleges mondatokkal, szavak begépelésével kellett irányítanunk, például „keres”, „vár”, vagy épp „cserél bizto”, amikor a biztosítékot szerettük volna kicserélni űrállomásunkon.

A ***VIC-II*** grafikai chip felépítése és tudása különösen alkalmas volt a ***Super Mariohoz*** hasonló mászkálós játékok írására. Ilyen volt a ***Giana Sisters*** néven ismert ***Mario*** klón, vagy a ***Creatures***, amelyben egy ***Clyde Radcliffe*** nevű macit irányítva kellett megmenteni testvéreit. A faltörő játékok is ekkortájt jelentek meg; legötletesebb és legjobban kidolgozott volt talán a ***Krakout***.

Mivel a géphez két botkormányt lehetett csatlakoztatni, teljesen megszokottak voltak a két emberes játékok. Játékteremben is volt szerencsém játszani a ***Bubble Bobble*** nevű programmal, amelyikben egy (vagy két) sárkányt irányítva tudtunk buborékokat fújni, és abba zárni a pálya tetejéről érkező, minket kergető szörnyeket. A ***Wizard of Wor*** pedig még mostanában is előkerül néha. Hála a lelkes gyűjtőknek, ezek a játékok letölthetőek az ***Internetről*** is.

A két játékos üzemmódhoz tartalmaz a ***VICE*** egy hálózati kiterjesztést is.

A használatához meg kell adnunk, melyik gép legyen a szerver, és melyik a kliens, illetve opcióként megadhatjuk, hogy melyik virtuális perifériához melyik gép férjen hozzá. Alapbeállítás szerint mindkét gép irányíthatja a billentyűzetet, és egyik gépé lesz az egyes botkormány, másiké a pedig kettes számú. Így ezzel bármelyik régi, két játékost igénylő program játszható akár az ***Interneten*** keresztül is!

A ***Netplay*** mód használatát úgy érdemes kezdeni, hogy a szervernek kijelölt virtuális gépen elindítjuk a játékot, és utána a ***Snapshot/Netplay*** menüpontra kattintva létrehozunk egy szervert. A kliens géppel pedig, az emulátort ott is elindítva azonnal csatlakozhatunk a szerverhez, ugyanis ilyenkor a szerver aktuális állapotának összes információja átkerül a kliensre (mintha az előbbin mentettünk volna egy játékállást, vagyis egy ***snapshotot***). Emiatt tulajdonképpen elég, ha csak az egyik gép rendelkezik a programmal (***.d64*** vagy ***.t64*** fájl). A kapcsolódás után már indulhat is a játék!

Sajnos ez a funkció még csak kísérleti állapotban van. Én magam azt tapasztaltam, hogy a két gépen a játék nincsen egészen szinkronban, bár ez lehet, a hálózat sebességének ingadozása miatt történt. Amikor a két gép közül az egyiknek más módon is le volt terhelve a hálózati kapcsolata, mindkét emulátor elkezdett akadozni is.

Természetesen valószínűleg helyi hálózaton sokkal jobban működik ez a lehetőség!

A VICE mint fejlesztőkörnyezet

Vannak még néhányan, akik jelenleg is fejlesztenek programokat a ***Commodore 64-re***. A ***VICE*** több olyan lehetőséget is kínál, amelyek ezt segítik. Tartalmaz például egy beépített monitorprogramot, amelynek segítségével a virtuális processzor működését követhetjük, akár utasításról utasításra haladhatunk, mint egy ***debuggerben***. Vagy mód van például a fájlrendszerünkön egy adott könyvtár elérésére (***Settings/Peripheral settings/Device #8/File system directory...*** menüpont) is. A fejlesztést emiatt most már kényelmesebb egy emulátorban elvégezni.

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

KAPCSOLÓDÓ CÍMEK

A VICE honlapja:
➲ http://www.viceteam.org

Magyar Commodore 64 HQ:
➲ http://c64.hardwired.hu

Atari joystick csatlakoztatása PC-hez:
➲ http://www.epanorama.net/documents/joystick/pc_circuits.html#atari_pc_converter

Arno Weber Boulder Dash oldala:
➲ http://www.cs.vu.nl/~aeweber/bd/index.htm

Az Opera új lehetőségei: widgetek (2. rész)

Az Opera 9.0-ás változatában jelentek meg először a widget-eket, melyek kis AJAX (aszinkron kéréseket támogató, Javascripttel segített, XML kommunikációs) alkalmazások. Cikkem előző részében bemutattam, hogyan is kell egy ilyet előállítani, ebben hónapban pedig a mások által elkészítettekből mutatnék be néhányat.

Enigma

Az ***Enigma*** a második ***Világháborúban*** a németek által használt titkosító szerkezet, melynél huzalok és tárcsák előre meghatározott beállítása segítségével végezték a titkosítást. Ennek függvényében egy adott szöveg bevitele közben a szerkezet kijelzőjén megjelent a hozzátartozó kódolt szöveg. Az egyes betűkhöz nem mindig ugyanaz a kimenet tartozott, hiszen használat közben a tárcsák forogtak, így megváltozott az alapbeállítás.
A kódolt szöveg visszafejtéséhez a legelső beállítást kellett a begépelésekor alkalmazni.
A programmal a szöveg titkosítása úgy történik, hogy először az ablak közepén lévő nyilakkal tetszőleges betű hármast kell beállítani (ez felel meg az eredeti szerkezet tárcsáinak, a huzalok a webeszköz ezen verziójából kimaradtak.), amit érdemes megjegyezni, hiszen ez kell a visszafejtéshez, majd beírjuk a kódolandó szöveget, s a jobbra mutató nyilakra kattintva megkapjuk a titkosított szöveget. A visszafejtéséhez ismét be kell állítani az eredetileg használt három betűt.
Az ***Opera*** oldalán a hozzászólásokat olvasgatva jutottam arra megállapításra, hogy sokan nincsenek tisztába az ***Enigma*** működési elvével, mivel azt kérik, hogy a programozó építse be a dekódoló részt is a programba, holott az előbb elmondottaknak megfelelően az már része.
Az ***Enigmával*** előállított kódot sokáig feltörhetetlennek hittek. Ma már biztosan állíthatjuk, hogy nem az, így nem érdemes fontos információkat így titkosítani.

A widget-ek beszerzése

Az ***Operán*** belül ***webeszközök/webeszközök*** hozzáadása menüre kattintás után bejön az ➲ ***widgets.opera.com/*** oldal, és itt különböző feltételek szerint válogathatunk közöttük (szerző, típus, ország, népszerűség), majd pedig le is tölthetjük a nekünk megfelelőt.

ImageGREP

Az ***Internetre*** feltöltött rengeteg kép között egyre nehezebb a minket érdeklő megtalálása. Az ***ImageGREP*** ebben van segítségünkre. A program jelenleg három szolgáltatóval működik együtt: ***Flickr***,-rel a ***Google***-lel, és a ***Yahoo!***-val, de a programozójának ígérete szerint újabb szolgáltatók is várhatók, például ***MyOpera albums***.
A ***widget*** előnye, hogy egység, szükségtelen funkcióktól mentes felület segítségével keresgélhetünk a képek között, s így nem kell felkeresni egyik szolgáltatót oldalát sem. Csak a szűrési feltételek változnak, a két kereső motornál csak méret szerint válogathatunk, míg a ***Flickr***-nél beállíthatjuk, hogy a címkék vagy a kulcsszavak alapján illetve népszerűbbek vagy a legújabbak között keressen.

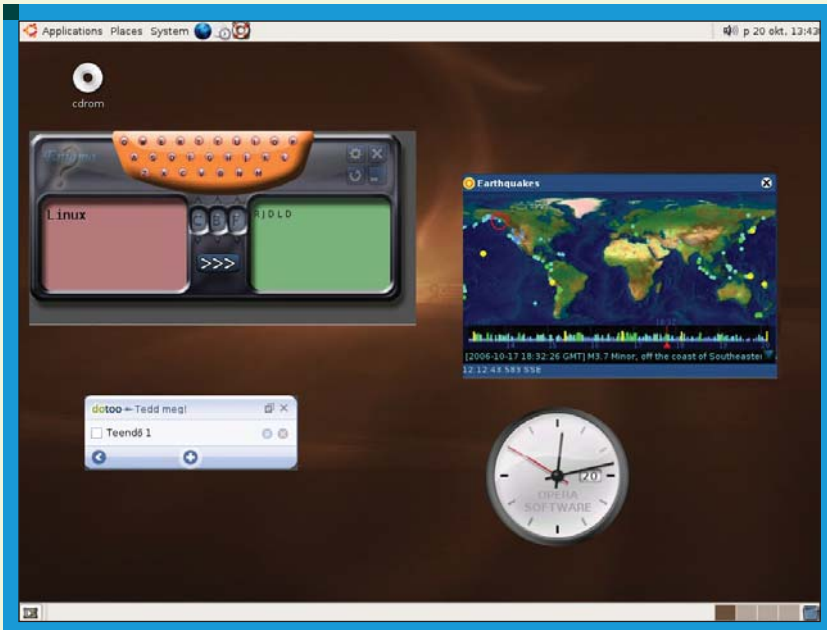
1. ábra Az ImageGREP találati képernyője a Linux szóra

Artist's Sketchbook

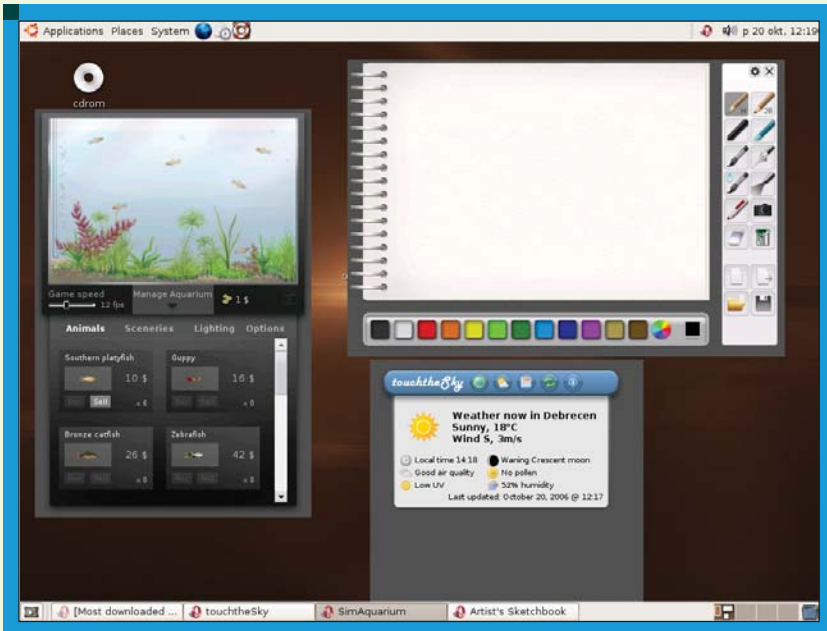
Egy egyszerű rajzoló program, mely tudásában nem ér fel ugyan a ***GIMP***-pel,de vázlatok, skiccek gyors elkészítésére kiválóan alkalmas. Ismeri a ***2B***-s és ***H***-s ceruzát, a szén és pasztell krétát., az ecsetet és a töltőtollat. Az elkészült művet ***PNG*** formátumban menthetjük az export gomb segítségével. A mentés gombbal csak az aktuális állapotot menthetjük , melyet ismételt lenyomásra felülír ekkor nincs lehetőség név vagy kiterjesztés megadására.

TouchtheSky

A ***webeszköz*** segítségével megtudhatjuk, hogy milyen az aktuális időjárás a világ különböző pontjain. Ezen kívül



2. ábra A képen az Enigma, dotoo: To-do lists, EarthQuakes látható



3. ábra A képen a SimAquarium, Artist's Sketchbook, és a TouchtheSky látható

információt kapunk pollenszintről, a helyi időről, az ***UV*** sugárzásról, a hold ciklusról, valamint a levegő nedvesség tartalmáról is.(Érdekes az, hogyha nincs pollen a levegőben egy szomorkodó ikonnal jelzi, pedig szerintem ez jó hír.)
Alapból az ***Oslóhoz*** kapcsolódó információk jelenek meg, de ha az egeret a menüsor jobb szélére visszük, előtűnik egy új ikon, erre kattintva egy űrlapra jutunk itt kell rákeresni a minket érdeklő helyre. A fentieken kívül a programban megnézhetjük a térség szatellit képét, valamit tájékoztatást kapunk a várható időjárásról is.

dotoo: To-do lists

Notórius feledékenyeknek lehet egy újabb segítsége ez a program, melyben a sürgős tennivalókról készíthetünk listákat, témánkét kategorizálva. A ***webeszköz*** első elindításakor figyelmeztet, hogy még nincs saját listánk. Ezt a program alján lévő fehér kereszt segítségével pótolhatjuk. Lényegében itt adhatjuk meg a kategória neveket. Ezekre kattintva pedig ugyanígy a teendőket is, melyek mellett lévő jelölő négyzeteket használva kihúzhatjuk az elvégzetteket a listáról, de törlésükre és átnevezésükre is van lehetőség.

EarthQuakes

A webeszköz az elmúlt hét nap földrengésének helyét jeleníti meg egy kiterített ***Föld*** térképen, különböző színekkel jelölve erősségüket. Rákattintva a minket érdeklő térképrészre, a böngészőben ➲ ***earthquake.usgs.gov*** oldal jön be, ahol részletes információt kaphatunk az adott eseményről.

Circular Tetris

A mindenki által ismert játék egy újfajta megközelítése, hiszen itt a játéktér egy tórusz oldala, erre esnek rá az építőelemek. Egy sort egy teljes kör lefedése után vesz le a program. Irányítása billentyűzetről történik, a kurzor mozgató billentyűkkel és a szóközzel.

SimAquarium

A végére hagytam a legnépszerűbb ***widget***-et, mely nem más mint egy akvárium „szimulátor". A programban lehetőség van halak vásárlásra eladására, akvárium díszítésére, megvilágításának cseréjére, ha tetszik az elkészült akvárium akkor azt le is „fényképezhetjük" és elmenthetjük ***PNG*** formátumban. Persze mindez pénzbe kerül, a program minden percben jóváír számunkra egy dollárt, valamint a halak is szaporodnak maguktól, melyeket eladva más fajtákat vehetünk. Ami a leglényegesebb: ebben a játékban nem lehet veszteni. Remélem ez a kis bemutató is megmutatja, hogy milyen széles tartalmat lefednek már a ***widget***-ek, persze a lehetőségek ezekkel még nem merültek ki. Az kérdésként felmerülhet, hogy van e szükség rájuk, ilyen az ***Opera*** böngészőhöz kapcsolódó formában. Válaszom: miért ne lenne?

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.

Egy terminál mind felett – Yakuake

Mindent elborítanak a terminálablakok? Ez a program könnyen a munkaasztalunk hősévé válhat. Egy terminál mind felett!

Sokan használunk terminált nap mint nap. Van aki többet is. Amikor pörögni kezdenek az események, akkor hirtelen kettő, sőt három sem elég. Ha olyan terminál emulátort használunk amelyik több munkamenetet, lapot is tud egyszerre kezelni, akkor már sokkal egyszerűbb az életünk.
Az ideális eszköz lehetőleg mindig kéznél van, nem zavar a munkában és bárhonnan elérhető. A ***Yakuake*** pont ilyen. Ez a program egészen addig láthatatlan, ameddig nem nyomjuk le az „F12” billentyűt. Akkor azonban a egyszerre legördül a képernyő felső részéből és máris használhatjuk. Mivel lapokat is tud kezelni, így több terminált is használhatunk párhuzamosan a segítségével, és azokat bármelyik virtuális munkaasztalunkról egy gombnyomással elérhetjük. A legszimpatikusabb az a programban, hogy nem tolakodó. Amint elveszíti a fókuszt, máris felgördül, nem akadályozva minket a további munkában.
A ***Yakuake*** név a „Yet Another Kuake” összetételből ered, vagyis az jelenti, hogy „még egy ***Kuake***”. A ***Kuake*** pedig egy olyan terminál emulátor, amit a népszerű ***Quake*** játék konzolja ihletett. Ennyi bevezetés után nézzük meg, minként lehet munkára bírni!

Telepítés

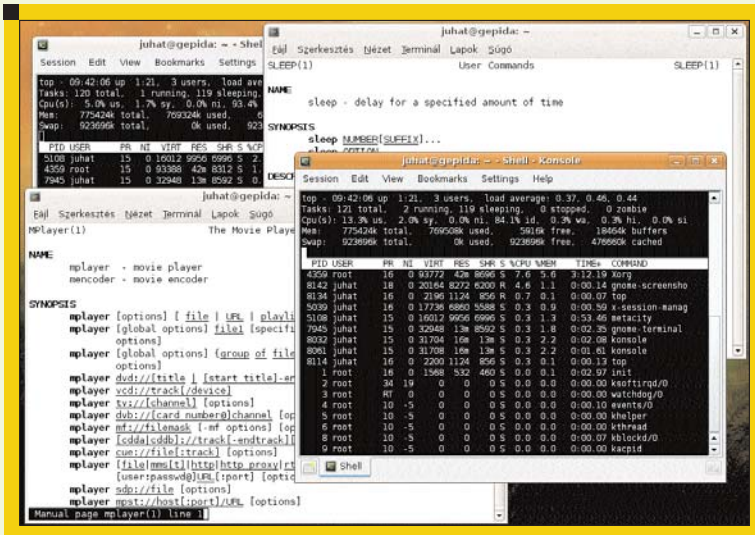
A telepítés egyszerű. Használjuk a ***Linux*** terjesztésünk csomagkezelőjét! ***Ubuntu*** alatt a legegyszerűbb, ha a

```
sudo apt-get install yakuake
```

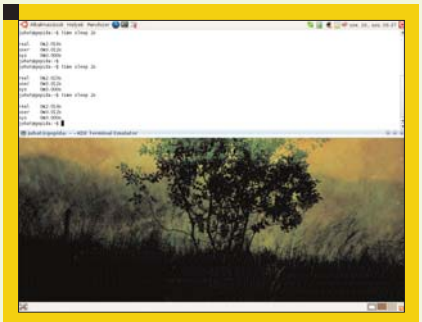
parancsot használjuk. Telepítés után indítsuk el!

Beállítások

Alig van pár beállítás amiről tudnunk érdemes. Az egyetlen érdekesség, hogy a beállításokat a program panelján jobb oldalán található kis ikonokkal lehet előcsalni. A funkcióbillentyűket mindenképpen érdemes átírni, hogy jobban a kezünkre álljanak. Ezen kívül beállíthatjuk, hogy fókusz elvesztése esetén eltűnjön-e a legördülő terminálunk.
Hasznos kis segédprogram a ***Yakuake***, remélem sokunk mindennapi munkaeszköze válik belőle! Bár eredetileg ***KDE*** alá írták, én azt tapasztaltam, hogy mindenféle probléma nélkül integrálódik a ***Gnome*** ablakkezelő alá is.



1. ábra Mindent elborítanak a terminálablakok?



2. ábra Így néz ki a Yakuake működés közben

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.

Kamerakiszolgáló működtetése héjprogramból

Egy beágyazott eszköz programozott működtetése nem föltétlen bonyolult dolog. Akár egy közönséges héjprogram is megteszi...

A svéd *Axis Communications AB* néhány éve egy teljesen új koncepciót vezetett be, amikor piacra dobta új kamerakiszolgálóit. Elképzelhető persze, hogy a cég hálózati kamerái (az egyik típusról 2000 szeptemberében cikk is jelet meg a *Linux Journalban*) nem képesek minden területen kiváltani a hagyományos, analóg készülékeket, az azonban kétségtelen előnyük, hogy a vezérlő és digitalizálást végző elektronika egy különálló, masszív felépítésű, semmiféle mozgó alkatrészt, még hűtőventilátort sem tartalmazó dobozban kapott helyet. A készülék egy és négy analóg videóbemenettel rendelkező változatban kapható, és minden típuson van egy átmenő kivezetés is. Ezekkel a kamerakiszolgálókkal a vállalatok lépésenként modernizálhatják biztonsági rendszerüket.

A kamera

A manapság kapható, *Linuxot* futtató beágyazott rendszerek természetükből adódóan meglehetősen specializáltak. A cikkben szereplő kamerakiszolgálóval azért kerültünk kapcsolatba, mert egyik ügyfelünk igényei bár nem voltak kielégíthetők kizárólag az *Axis* kameráival, a cég kameraszervereivel kapcsolatban nekünk is, és az ügyfélnek is pozitív tapasztalatai voltak.
Így aztán úgy döntöttünk, hogy egy másik, piacvezető cégtől fogunk bekérni olyan biztonsági kamerákat, amelyek képesek együttműködni ezzel a készülékkel.
Akárcsak a számítógépek egyéb hardverelemei, a kamerák és a hozzájuk tartozó optikák is amolyan pénznyelő automataként képesek működni. Tulajdonképpen kijelenthetjük, hogy nincs az a pénz, amit el ne lehetne költeni, ha az embernek különleges igényei kezdenek támadni.
Nekünk konkrétan egy olyan kültéri *PTZ* (*pan-tilt-zoom*; fordítás-billentés-nagyítás) kamerára volt szükségünk, aminek elfogadható a nagyítása is.
A *PTZ* kamera nem olcsó mulatság. A beltéri egységek ugyan csak néhány száz dollárba kerülnek, az olyan kültériek azonban, amelyek elviselik a tűző napot, a klímát és még némi vandalizmust is, több ezer dollárba kerülhetnek.
Mi egy forgótálcára szerelt átlátszó falú házzal védett típus mellett döntöttünk. Ezek azok a kamerák, amelyeket manapság a legtöbb repülőtéren is látni. Tekintettel arra, hogy ügyfelünk székhelye *Norvégiában* található, fűtött kameraházra volt szükségünk, hogy a téli jegesedét elkerülhessük.
Íme az eszköz műszaki paraméterei:

- Elforgathatóság 360 fokban
- Dönthetőség +2 és -92 fok között
- Képérzékelő: egynegyed hüvelykes CCD (3,2 x 2,4 mm)
- 22-szeres optikai zoom, 4-től 88 mm-ig állítható fókusztávolság
- Érzékenység: 0,07 lux 1/1,5 másodperces zársebesség mellett
- Zársebesség: 1/1,5-től 1/30.000 s-ig
- Legkisebb F-stop érték: f/1,6
- Környezeti terhelés: -40-től +50 fokos tartós terhelésig

A probléma

Az egész projekt mögötti alapelképzelés, amiért egy *PTZ* kamerát választottunk az volt, hogy bizonyos fix pontokat szerettünk volna fényképezni adott időközönként és a képeket feltölteni egy webkiszolgálóra. Itt aztán beleütköztünk egy problémába. A szerverhez adott szoftver természetesen kiválóan kezelte az *FTP* protokollt, és adott időszakonként is tudott képeket készíteni. Ellenben semmilyen módját nem

találtuk annak, hogy felvétel előtt a kamera pozícióját és nagyítását, vagyis egy adott *PTZ* hármas beállítsunk. Mivel ez így nem felelt meg az ügyfelünknek, a szállítással várnunk kellett, amíg áthidaljuk ezt a gondot.
A készüléket vezérlő szoftvert több úton is lehet módosítani. Először is a firmware részét képező valamennyi nyílt forrású komponens forráskódja rendelkezésre áll. Ebben sajnos a vezérlést végző *CGI* interfész nem tartozik bele, de az annyira nem is volt zavaró. A többségbe bele tudtunk tekinteni, és persze módosítani is lehet az eszköz kódját. Mindezzel együtt egy adott firmware kódját nem lehet csak úgy letölteni, azt írásban kell kérni az *Axis* megfelelő osztályától, amely aztán jelképes összegért *CD*-n elküldi a kódot.
Mivel több ügyfelünk igényeinek kielégítéséhez is szükség lett volna ennek a problémának a megoldására, viszonylag gyorsan kellett cselekednünk. Több olyan programozási felület is felmerült, amelyek segítségével átalakíthatjuk az eszköz működését a saját igényeinknek megfelelően. Vizsgálataink szerint a következő *API*-k jöhettek egyáltalán szóba:

- A készülék saját adminisztrációs felülete (webböngészőn keresztül használható)
- *HTTP API*
- Megfelelő szkriptek írása
- Héjprogramból történő vezérlés
- *PHP3 Lite*
- *GCC SDK for Linux/cris*

Akárcsak a legtöbb beágyazott eszközzel kapcsolatban, itt is számolnunk kellett néhány korlátozással és kényelmetlenséggel. Először és mindenek előtt nagyon kevés a rendelkezésre álló hely. Összesen körülbelül 100 kilobájtnyi írható fájlrendszerrel gazdálkodhattunk, ezen kellett elhelyezni az összes, harmadik féltől származó módosítást. Ennyi maradt ugyanis abból a 4 megabájtos Flash tárolóeszközből, ami a firmwaret tartalmazza. Valamennyi szoftver egy csak olvasható fájlrendszeren van, így nem helyettesíthető egy teljes, testreszabott firmware létrehozása nélkül.

Amikor hozzákezdtünk a munkához, sem *SSH* kiszolgáló, sem ilyen ügyfél nem állt még rendelkezésünkre, ami azt jelentette, hogy a héjprogramokat vagy egy külön időzítővel kellett vezérleni, vagy *PHP* szkriptből kellett őket futtatni. Akkor még csak egy kísérleti *telnet* kiszolgáló volt a készülékhez, azóta viszont portolták rá a *Dropbear SSH* kiszolgálót is.
Semmiképpen nem akartunk külön szervert üzemeltetni pusztán az automatizálás megoldásához, ezért úgy döntöttünk, hogy mindent belső héjprogramokon keresztül fogunk megoldani, amiket az *utask* nevű ütemező fog majd időzíteni. Szerencsénkre ennek az ütemezőnek valamivel kifinomultabbak a képességei, mint amelyeket a közönséges *cron* magáénak tudhat. Az *utask* például külső jelekkel is vezérelhető, vagyis meg lehet neki tanítani, hogy mit tegyen, ha egy külső digitális jel alacsonyról magasra vált, vagy ha a kettes kamera egyszer csak nem ad videójelet.

A funkciók elérhetővé tételének lehetőségei

A beágyazott rendszereknél az egyik nagy kérdés mindig az, miként tegyük elérhetővé a vezérlési funkciókat. Ettől függ ugyanis, hogy a felhasználónak mekkora erőfeszítésekkel jár majd, ha a saját igényeihez akarja alakítani az eszköz működését. Nos, esetünkben elég nagy erőlködésre volt szükség, de rögtön hozzá kell tennem, hogy ebből a szempontból nincs két egyforma biztonsági rendszer.
A böngésző alapú felhasználói felület természetesen könnyen kezelhető, és sok szolgáltatást nyújt az emberi felhasználó számára. Ugyanakkor automatizálási célokra rendszerint tökéletesen alkalmatlan.
A *HTTP API* egy egyszerű, kérés-válasz alapú *API*, amit kifejezetten automatizálási célokra készítettek. Rajta keresztül a legtöbb olyan funkciót elérhetjük, ami egy böngészőben is rendelkezésünkre állna. A visszatérési értéke általában csak egy állapotkód, vagy maga a kért objektum, például egy a videófolyamból kiemelt képkocka, vagy maga a videó. Mindez természetesen kiválóan megfelel távvezérlési célokra, így nem véletlen, hogy számos gyártó készít több tíz, vagy akár több száz kamera egyetlen vagy néhány központi helyről történő vezérlésére alkalmas szoftvereket ezzel az *API*-val.
A *PHP3 Lite* nevű nyelvjárása kiválóan megfelel akkor, ha egyedi dinamikus lapokkal akarjuk kiegészíteni a böngésző alapú felhasználói felületet, sőt általános célú belső szkriptek írására is alkalmas.
A készülékhez adott, *sh*-val kompatibilis héj programozása a hozzá tartozó segédprogramokon keresztül akkor a legmegfelelőbb választás, ha valamilyen „helyi intelligenciára” van szükség a működtetés során. Ilyen lehet például az, hogy ha valaki megnyomja az ajtócsengőt, akkor kell róla készíteni egy felvételt, aztán be kell engedni egy relé működtetésével.
Végezetül ha valamilyen kifinomult helyi viselkedést kell megvalósítani, akkor a *GCC SDK* lesz a nyerő választás.

Mi is az tehát, ami a rendelkezésünkre áll?

Amint korábban említettem, mi a héjprogramozás mellett döntöttünk, mivel ez felelt meg leginkább a konkrét szituációnak. Ezen kívül felismertük, hogy a rendelkezésünkre álló firmware változat egészen sok hasznos programot tartalmaz:

- *BusyBox* amely a */bin/sh* szerepében a *sash* héjat használja.
- *mish*: Ez egy a *Minix sh*-val kompatibilis héj.
- *utask*: Feladatütemező, amely azonban nem kompatibilis a *cron*-nal.
- *buffered*: Képkészítő/pufferelő eszköz.
- *sftpcient*: Egyszerű *FTP* ügyfél.
- *shttpclient*: Egyszerű *HTTP* ügyfél.
- *smtpclient*: Egyszerű levélküldő program.

Külön érdemes kiemelni a háttérben szerényen megbújó *shttpclient*-et, hiszen ez lehetővé teszi, hogy a belső héjprogramokból a *HTTP API* szolgáltatásait használjuk, amire szükségünk is lesz a *PTZ* vezérlés megvalósítása során. Ezen kívül természetesen számos más dologra is használhatjuk.

1. Lista A kamerakiszolgáló beépített programjának kiegészítésére készített héjprogram

```
#!/bin/mish
PATH=/bin:/sbin:/usr/bin:/usr/sbin
# stop button pressed? skip the rest.
stopp=`ls /tmp/stopp` ;
if [ "x$stopp" = "x" ]; then
 # clear working directory
 bufferd -reset -buffername BILDE ;
 # read configured positions, '~' signifies
 # a unset
 # position
 grep -v '~' /etc/sysconfig/presetpos.conf \
 >/tmp/presets ;
 # pos 1-10 reserved for non-public views
 num=11;
 while [ $num -lt 21 ]; do
  # another stop button check - break the loop
  stopp=`ls /tmp/stopp` ;
  if [ "x$stopp" = "x" ]; then
   grep Pos$num /tmp/presets >/tmp/canptz ;
   canptz=`cat /tmp/canptz` ;
   if [ "x$canptz" != "x" ]; then
    # go to position
    shttpclient "http://127.0.0.1/axis-cgi/\
com/ptz.cgi?camera=1&gotoserverpresetno=$num";
    # wait for picture to settle
    sleep 6 ;
    # save picture to /tmp/BILDE/
    bufferd -start -buffername BILDE -snapshot\
     -pre 1 -format snapshot_pos$num.jpg -uri\
     'ftp://127.0.0.1/jpg/1/704x576.jpg' ;
    bufferd -stop -buffername BILDE ;
    tmpwait=20 ;
    while [ $tmpwait -gt 0 ]; do
     sleep 2;
     expr $tmpwait - 2 >/tmp/tmpwait ;
     tmpwait=`cat /tmp/tmpwait` ;
     if [ $tmpwait -eq 0 ]; then
      # timeout creating jpeg, kill process and
      # settle for potentially incomplete
      # picture
      logger "timeout waiting for bufferd
      ➥-stop,\
killing image_buffer and continuing."
      # ps is only available as a builtin
      # command in /bin/sh
      /bin/sh -c ps >/tmp/kverk ;
      grep image_buffer /tmp/kverk >/tmp/
      ➥kverk2 ;
      imbuf_pid=`cut -b 0-5 /tmp/kverk2` ;
      rm /tmp/kverk; rm /tmp/kverk2;
      kill $imbuf_pid ;
     fi;
     if [ -f /tmp/BILDE/status ]; then
      # status file appeared - picture complete
      rm /tmp/BILDE/status ;
      tmpwait=0;
     fi;
    done;
    rm /tmp/tmpwait ;
   fi;
  fi;
  expr $num + 1 >/tmp/A ;
  num=`cat /tmp/A` ; rm /tmp/A;
 done;
 rm /tmp/presets ;
 rm /tmp/canptz ;
 # batch ftp all the pictures from this round
 sftpclient -L -m 10.0.0.1 -k /tmp/BILDE -c \
  /var/www/pictures/c2 -u web -w P2SsW1Rd -t \
  /var/www/pictures/c2/temp_c2.jpg ;
fi;
```

Jeleket küldhetünk vele egy másik webszervernek vagy kameraszervernek, feltölthetünk képeket *HTTP* protokollon keresztül, de természetesen bármely más műveletet is megvalósíthatunk vele, feltéve, hogy ahhoz az alap hitelesítésen kívül nincs másra szükség. Elvégre ez csupán egy egyszerű *HTTP* ügyfél, nem a *wget*, vagy a *cURL*.

Azoknak a problémáknak a többsége, amelyekbe a konkrét megvalósítás során beleütköztünk általában nem a héjprogramozással álltak kapcsolatban. Tapasztaltunk például átmeneti hibákat a képek feltöltésével kapcsolatban. Itt annyi volt a „gubanc”, hogy egy a kertben található futószőlő indái fölkapaszkodtak a drót nélküli átviteli rendszer egyik antennájára.

Ennyit tehát a kameraszerverek héjprogramból történő vezérléséről. Végezetül természetesen meg kell mutassam azt a szkriptet, amivel kiegészítettük a készülék gyári funkcióit (*1. Lista*). Nem túl elegáns, elismerem, de kicsit és egyszerűen megírható, köszönhetően persze annak a bölcs előrelátásnak, amivel a beágyazott eszköz készítői éltek. Ez az utóbbi amúgy önmagában is olyan lecke, amit bárkinek érdemes megtanulnia, ha ezen a területen dolgozik.

Linux Journal 2006., 142. szám

Erik Inge Bolso

UNIX szakértő és vívó. A Norvégiai Molde-ban él, 1996 óta futtat Linuxot. Másik hobbiját a Google segítségével úgy találhatjuk meg, ha rákeresünk a „balrog genealogy” kifejezésre. Aki írni szeretne neki, az a *ljcomment@tvilsom.org* címen érheti el.

KAPCSOLÓDÓ CÍMEK

A cikkhez tartozó egyéb anyagok a következő helyen találhatók:
➲ www.linuxjournal.com/article/8695

USB Pendrive-ok és Linux terjesztések, amelyek elférnek rajtuk

Három olyan Linux terjesztést tekintünk át, amelyekkel akár egy USB pendrive-ról is bebootoltathatjuk gépünket.

A *pendrive* – mint köztudomású – egy *USB* kapcsolaton keresztül működő tároló-eszköz. Az ember bedugja az *USB* csatlakozóhelyre, és ha az eszköz kompatibilis az általunk használt operációs rendszerrel, akkor ugyanúgy fog megjelenni, mint bármely más lemezegység. Manapság ráadásul már 1 GB-os, sőt még ennél is nagyobb kapacitású eszközök is kaphatók.
Volt egy időszak, amikor ugrásszerűen megnőtt az olyan *Linux* terjesztések száma, amelyek úgynevezett *Live CD*-ről futottak. Ilyen volt a *Linspire*, de létezik hasonló elven működő változata a *SUSE*, az *Ubuntu*, a *Kubuntu*, a *Knoppix* és a *Mepis* terjesztésnek is. És ezzel a felsorolás korántsem teljes.
Mármost képzeljük el, hogy a fenti két koncepciót összeházasítjuk, vagyis egy *Live CD* anyagát áttesszük egy *USB* eszközre, és onnan használjuk. Ami azt illeti, 1 GB-on elég sok dolog elfér, már ha *Linuxról* van szó. Az *USB pendrive*-nak mellesleg megvan az az óriási előnye is, hogy írható, amely tulajdonsággal egy *Live CD* értelemszerűen nem rendelkezhet. *USB* eszközzel tehát tulajdonképpen tényleg úgy dolgozhatunk, mintha egy merevlemezről indítottuk volna a gépet: menet közben magunk is létrehozhatunk adatokat, és azokat meg is őrizhetjük. Végeredményben tehát egy zsebre vágható *pendrive* segítségével létrehozhatunk magunknak egy teljes, és tökéletesen hordozható munkakörnyezetet. Amíg találunk egy gépet, ami képes elindulni erről az eszközről, addig soha nem fordulhat elő velünk, hogy egy idegen rendszeren nem találunk meg egy adott alkalmazást, vagy hogy valami máshogy van beállítva, mint ahogy nekünk megfelelne. És persze az adatainkat is magunknál tudhatjuk, bárhol is legyünk.

A lehetőségek

Ebben a cikkben a következő három, úgynevezett *LiveUSB* terjesztést vizsgáljuk meg: *SLAX*, *Damn Small Linux (DSL)* és *Flash Linux*. Mindegyiket más ablakkezelővel szerelték fel, és a bennük rendelkezésre álló alkalmazások köre is eltér.
A *SLAX* a *tmpfs* és a *Unification fs (UFS)* segítségével működik, ami néhány igen figyelemre méltó tulajdonsággal ruházza fel ezt a terjesztést. A *SLAX* amúgy a *Slackware Linuxon* alapul és 2.6-os kernelt tartalmaz.
A *DSL* egy mindössze 50 MB méretű terjesztés, amelyben egy igen ügyesen beállított *Fluxbox*-ot találunk. Ami a rendelkezésre álló alkalmazásokat illeti, van benne *Mozilla Firefox*, levelezésre pedig a *Slypheed* klienst használhatjuk. Megtalálható benne ezen kívül az *xmms*, néhány szövegszerkesztő, képnézegető és sok más. 2.4-es kernelt használ ugyan, de nagyon jók a hardverfelismerési képességei. Ugyanakkor – egyszerűen méreténél fogva – hiányoznak belőle azok a „nagy" alkalmazások, amelyek más terjesztésekben kötelező jelleggel benne foglaltatnak. Itt tehát ne is keressük például a *Gimpet*. Összefoglalva ez tehát egy kifejezetten kompakt terjesztés, amit egy ügyesen kialakított szkript telepít fel a *pendrive*-unkra.
Végül de nem utolsó sorban a *Flash Linux* egy egészen komoly, 2.6-os kernelen, és a *JFFS2* fájlrendszeren alapuló terjesztés. Kiváló a sebessége akár *Live CD*-ként, akár *LiveUSB*-ként használjuk. Számos olyan nagyobb alkalmazást is tartalmaz mint például a *Gimp*, vagy az *OpenOffice.org*. Rendszertöltőként *GRUB*-ot használ, van benne grafikus nyitóképernyő (*bootsplash*), kezeli a *framebuffert*, megtaláljuk benne a *Gnome* ablakkezelőt, az egész rendszer alapját pedig a *Gentoo Linux* jelenti.

Rendszerindítás USB-ről

Az *USB pendrive* használatában a legnagyobb kihívást tulajdonképpen az jelenti, hogyan indítsuk el gépünket erről az adathordozóról. Mivel a régebbi alaplapok *BIOS*-a nem tartalmaz ilyen funkciót, ezeknél egy külön floppylemezt kell használnunk a rendszer indításához. Az újabb alaplapok már támogatják az ilyen rendszerindítást is. A *BIOS* beállításai között keressük a következő neveket: *USBHDD*, *USBZIP*, *USB-FDD* vagy *USB-CRDOM*, de mások is előfordulhatnak.
A bootolható *pendrive* előkészítésének első lépése a rajta található eredeti partíciók törlése, már ha van rajta ilyesmi. Ezután létre kell hoznunk egy *FAT16* típusú partíciót, majd meg kell formáznunk ezt az `mkdosfs` parancs segítségével. Én a cikkben az `fdisk`-et fogom használni, de természetesen a `cfdisk` is tökéletesen megfelel a célnak.
Miután csatlakoztattuk az eszközt géphez, először is a `dmesg` segítségével ellenőrizzük le, hogy valóban működik-e:

```
dmesg | tail
```

A következőhöz hasonló rendszerüzenetet kell látnunk:

```
sda: assuming drive cache:
➥write through
sda: sda1
```

Formázzuk meg a most létrehozott partíciót a következő paranccsal:

```
mkdosfs -F 16 /dev/sda1
```

Az `sda1` helyett természetesen azt a nevet kell használnunk, ami a saját rendszerünkön a *pendrive*-on kialakított partíciónak megfelel. Ha kész a művelet, válasszuk le az eszközt, húzzuk ki fizikailag is a gépből, majd dugjuk be újra. Ha minden jól ment, akkor ezzel készen is állunk valamelyik korábban említett disztribúció telepítésére.

Damn Small Linux

Látogassunk el a *DSL* webhelyére (lásd a cikk végén) és töltsük le a legújabb változat ISO lemezképét. Írjuk fel ezt egy *CD*-re vagy *DVD*-re. Bootoljunk be erről a lemezről.
A bootfolyamat egy üdvözlő képernyővel indul, akárcsak a legtöbb *Live* terjesztésnél.
A *DSL* ezután megvizsgálja, hogy milyen hardverelemek vannak a gépünkben, telepíti a szükséges meghajtókat, és mindent beállít. Attól függően, hogy milyen kiépítettségű gép van alatta, az *X* kiszolgáló és egy *Fluxbox* ablakkezelő elindítása általában nem tart tovább neki két percnél.
Ha felállt a *DSL* rendszer a *Live CD*-ről, a *Fluxbox* asztalán nyissuk meg a *Fluxbox* menüt. A rendszer *USB pendrive*-ra való telepítéséhez válasszuk az *Apps->Tools->Install* menüpontot. Itt két lehetőségünk van a telepítés végrehajtására: telepíthetünk *USBHDD*-re vagy *USBZIP* hardverre. A *DSL* meg fogja kérdezni, hogy hol találja a *pendrive*-ot, illetve arra is rákérdez, hogy a rendszert magáról a *Live CD*-ről, fájlból, vagy a webről szeretnénk telepíteni.
Aki rendelkezik szélessávú kapcsolattal, annak én a hálózati telepítést javaslom. Ami azt illeti, ha van olyan útválasztónk, ami támogatja a *DHCP* használatát, akkor a *DSL* már elvileg fel is ismerte a hálózati kártyánkat, és kért egy *IP* címet is, vagyis a nethez való hozzáférés elvileg semmiféle problémát nem jelenthet. Jó tudni, hogy a *DSL* a *PPPoE* használatát is támogatja ha esetleg olyan internetkapcsolatunk van, amihez szükség van erre.
Nekem személy szerint hiányzott a 2.6-os kernel néhány szolgáltatása, de összességében egész jó kis terjesztés a *DSL*, a fejlesztők ígérete szerint pedig a következő kiadása már 2.6-os rendszermagra fog épülni. Egyszóval a *DSL* kiválóan megfelel, hacsak nem akarunk egy nagy irodai csomagot használni. Ebben az esetben a *SLAX* lesz a jó választás. Szeretnék ugyanakkor mindenkit figyelmeztetni, hogy mielőtt használni kezdené a *DSL* terjesztést, mindenképpen olvassa el a hozzá tartozó *Wikit*, illetve olvasgassa kicsit a fórumot is. Rengeteg ötletet, tippet találhatunk ezeken a helyeken, és persze számos olyan információt, aminek nagy hasznát vehetjük, ha beleszaladunk valamilyen problémába.

Bootolás floppy segítségével

Ha a gépünk nem támogatja az *USB pendrive*-ról való rendszerindítást, akkor a *DSL*-t kénytelenek leszünk előbb egy floppylemezről elindítani. Töltsük le tehát a *bootfloppy-usb.img* nevű lemezképet, és másoljuk fel egy hajlékonylemezre a `dd` parancs segítségével a következőképpen:

```
dd if=bootfloppy.img
➥of=/dev/fd0
```

A *BIOS*-ban állítsuk be elsődleges booteszköznek a floppymeghajtót, majd indítsuk el a lemezen található rendszert. Ez a minirendszer természetesen a *DSL USB*-s változatát fogja elindítani. Maga a módszer amúgy egyáltalán nem egyedi, hiszen gyakorlatilag minden olyan terjesztéssel működik, amelyhez be tudunk szerezni ilyen lemezképet a közvetett *USB*-s bootoláshoz.

SLAX

A *SLAX* terjesztés webhelyén bemutatkozó szövegként a következőket olvashatjuk:

„A SALX egy gyors és csodálatos Linux operációs rendszer, amely a kis méretű (3,14 colos) CD-ROM lemezeken is elfér. Közvetlenül CD-ről (vagy USB-ről) fut, telepíteni nem kell. A Live CD alapját a Slackware Linux terjesztés képezi és a Unification File System fájlrendszert használja, amely lehetővé teszi, hogy egy fizikailag csak olvasható fájlrendszer virtuálisan írhatóként viselkedjen. Működésének lényege, hogy valamennyi változás a memóriába kerül.”

Ha *pendrive*-ot használunk, ezzel az utóbbival nem is kell foglalkoznunk, hiszen a *CD*-től eltérően ez egy valóban írható adattár.
Ugyanakkor az *UFS* arra is kiválóan használható, hogy segítségével egyetlen helyi címtárban egyesítsünk fizikailag különböző helyeken, például

a hálózaton található adatforrásokat. Ez a tulajdonsága az ***UFS***-t különösen alkalmassá teszi merevlemez nélküli munkaállomások kialakítására, hiszen minden különösebb nehézség nélkül megoldhatjuk vele, hogy a ***home*** könyvtárunk egy hálózati tárolóeszközön legyen.
A ***SLAX*** moduláris rendszer, ami azt jelenti, hogy mi válogathatjuk össze az elérhető szolgáltatáscsomagot. Segítségével számos különböző célra alakíthatunk ki operációs rendszert. Nézhetünk vele például ***DVD***-t, használhatjuk rajta a ***QEMU***-t, írhatunk ***CD***-t vagy ***DVD***-t, futtathatunk tűzfalat, víruskeresőt, és még számtalan más dologra használhatjuk. A lehetőségek és rendelkezésre álló modulok felsorolását a rendszer webhelyén megtaláljuk.

A SLAX telepítése

A ***SLAX*** telepítéséhez először is töltsük le a legfrissebb változatot a rendszer webhelyéről (a címet lásd az online források között). Ügyeljünk rá, hogy a ***SLAX***-nak számos különböző változata létezik, amelyek némiképp eltérnek egymástól a teljes rendszer méretét, illetve a rendelkezésre álló alkalmazások körét tekintve.
Van tehát ***Frodo***, ***Standard***, ***Popcorm*** és ***KillBill*** kiadás. Én a ***slax-5.0.7b.iso*** standard változatát próbáltam ki, ami ***KDE***-t tartalmaz és körülbelül 200 MB a mérete.
A ***SLAX***-ot tartalmazó ***ISO*** lenyomatot a visszacsatoló eszköz (***loopback***) segítségével becsatolhatjuk. Nálam a csatolási pontként szolgáló könyvtár a ***slax*** volt, így a következő parancsot használtam:

```
mount -o loop  slax-5.0.7b.iso
➥slax/
```

Amint korábban, most is meg kell formáznunk használat előtt a ***pendrive***-ot, és egy ***FAT16***-os partíciót kell rajta kialakítanunk:

```
mkdosfs -F 16 /dev/sda1
```

Az sda1 helyett természetesen megint azt az eszköznevet kell használnunk, ami a saját rendszerünkön a kérdéses ***USB*** portnak megfelel. Ha tehát készen áll a bootolható és formázott ***FAT16***-os partíció, akkor csatoljuk be azt a következő paranccsal:

```
mount -t vfat /dev/sda1
➥/mnt/usb/
```

Másoljuk át az összes a ***slax***/ könyvtárban található fájlt onnan, ahova az ***ISO*** képet becsatoltuk oda, ahova a ***pendrive***-on kialakított partíció csatlakozik:

```
cp -rav slax/* /mnt/usb/
```

Szinkronizáljuk a fájlrendszereket:

```
sync
```

Lépjünk be abba a könyvtárba, ahol a ***pendrive*** tartalma található (esetükben ***/mnt/usb***):

```
cd /mnt/usb/
```

Másoljuk át a gyökérkönyvtárba a `vmlinuz` és az `initrd.gz` nevű fájlokat. Itt a gyökérkönyvtár természetesen a ***pendrive*** gyökerét jelenti, vagyis esetünkben a ***/mnt/usb*** könyvtárat:

```
cp boot/vmlinuz .
cp boot/initrd.gz .
```

Végezzük el a szükséges módosításokat az `isolinux.cfg` fájlban:

```
pico isolinux.cfg
```

Távolítsunk el belőle minden olyan karakterláncot, ami a `vmlinuz` vagy az `initrd.gz` előtt a `boot/` nevet tartalmazza. Ezután nevezzük át `syslinux.cfg`-re, hogy a kérdéses eszközzel kapcsolatban a ***syslinuxot*** használjuk:

```
mv isolinux.cfg syslinux.cfg
```

Végezetül telepítsük és frissítsük az ***MBR***-t a ***LILO***-val vagy a ***GRUB***-bal:

```
lilo -M /dev/sda
```

A folyamatot a `syslinux` futtatásával fejezhetjük be:

```
syslinux -s /dev/sda1
```

A ***SLAX***-ot ezzel telepítettük is, máris használható. Válasszuk le a ***pendrive***-ot, és indítsuk újra a gépet. A ***BIOS***-ban az elsődleges booteszközként adjuk meg az ***USB pendrive***-ot, és ismét indítsuk újra a gépet. A ***pendrive***-on a ***Master Boot Record*** lecseréléséhez a ***LILO***-t vagy a ***GRUB***-ot használhatjuk.
A ***SLAX*** tartalmazza a ***KDE***-t, a ***Fluxbox***-ot, a ***K3b***-t, a ***Media Playert***, webböngészőket, levelezőprogramot, irodai csomagot, a ***Kopete***-t és még számos más alkalmazást. A teljes listát a ***SLAX*** webhelyén olvashatjuk (lásd az on-line forrásokat).
A ***SLAX*** nem olyan gyors, mint a ***DSL***, viszont 2.6.15-ös kernel van benne, kiváló a hálózati támogatása, sőt – sok egyéb hasznos segédprogram mellett – megtaláljuk benne a `parted` nevű partíciószerkesztőt is. Összességében sokkal összetettebb terjesztés, mint a ***DSL***, de a méretért sajnos fizetnünk is kell, már persze ami a sebességet illeti.

Flash Linux

A *Flash Linux* nevű terjesztés alapját a *Gentoo Linux* képezi. A legfrissebb változatát tartalmazó *ISO* lenyomatot a terjesztés webhelyéről tölthetjük le (lásd az on-line forrásokat). Ezt fel kell írnunk *CD*-re, majd be kell bootolni róla. Erről a *Live* rendszerről telepíthetjük aztán a *LiveUSB* változatot, vagyis eddig minden ugyanúgy történik, mint az előző rendszernél. Látogassunk el tehát a ➲ *sourceforge.net/project/showfiles.php?group_id=124770* webhelyre, és töltsük le a három részből álló telepítőkészletet. A részek neve a következő:

```
flashlinux-0.3.4-RC2.iso-part1
flashlinux-0.3.4-RC2.iso-part2
flashlinux-0.3.4-RC2.iso-part3
```

Első lépésként egyesítenünk kell a három részt:

```
cat flashlinux-0.3.4-RC2.iso-
➥part1 flashlinux-0.3.4-
➥RC2.iso-part2 \
flashlinux-0.3.4-RC2.iso-part3 >
➥flashlinux-0.3.4-RC2.iso
```

A *Flash Linuxnak* gyönyörű nyitóképernyője illetve *framebuffer* témája van. Tartalmazza a valódi *NVIDIA* meghajtót is, ami jól fog jönni, ha valakinek mondjuk *GeForce* kártya van a gépében. A hardverdetektálás szintén kiválóan működött. A *Flash* a gépemben található valamennyi eszközt minden különösebb probléma nélkül felismerte és beállította.
Miután elindítottuk a gépet a *CD*-ről, és bejelentkeztünk, nekiláthatunk a *Flash Linux pendrive*-re telepítésének. Az eszközön most két partícióra lesz szükségünk. Kell egy legalább 4 MB méretű boot partíció, és egy másik, legalább 256 MB méretű terület.
A *Flash Linux* fejlesztői azt ajánlják, hogy a partíciókat az `fdisk` segítségével hozzuk létre. Csatlakoztassuk tehát az eszközt az *USB* portra, és adjuk ki a következő parancsot:

```
fdisk /dev/sda
```

Az `sda` nevet ismét arra kell kicserélni, ami a saját rendszerünkön a kérdéses eszköznek megfelel.

Először is töröljünk minden esetlegesen már létező partíciót, aztán hozzuk létre a 4 MB-os bootpartíciót. Az elmondottaknak megfelelően kell ezen kívül egy másik fájlrendszer is, amire a rendszer többi eleme kerül. Írjuk ki a lemezre a módosításokat, majd lépjünk ki az `fdisk` programból.
Töltsük le az *USB* eszközökhöz készített telepítőszkriptet, amit jelen esetben `flash_key.sh`-nak hívnak, majd másoljuk ezt be a *Flash Linux Live CD* gyökérkönyvtárába.
Ha a szkript nem látja a használni kívánt *USB* eszközt, akkor módosítanunk kell egy kicsit a kódon. Keressük meg tehát a `PICKDEVICE`-ban a

```
dev=`readlink /${i}|cut -d"/"
➥-f11`
```

sort, majd cseréljük le a következőre:

```
dev=`readlink /${i}|cut -d"/"
➥-f12`
```

Adjunk végrehajtási jogot a programnak, majd futtassuk le:

```
chmod 755 flash_key.sh
./flash_key.sh
```

Ez után nincs más dolgunk, mint követni a varázsló utasításait. Először ki kell választanunk azt az eszközt, amelyre a *Flash Linuxot* telepíteni szeretnénk. Ezt követi a 4 MB-os boot partíció, majd a gyökér fájlrendszer megadása. Ha minden információt begyűjtött, akkor a szkript először törli a *pendrive*-ot, majd becsatolja azt, végül pedig rámásolja a rendszert, az alkalmazásokat, és minden szükséges adatot. Ez az utolsó lépés nálam több mint nyolc percig tartott, tehát legyünk türelmesek. Ha viszont befejeződött a folyamat, a rendszer máris használatra kész.
A *Flash Linux* egyik leglényegesebb jellemzője a gyors működés. Ez elsősorban a *JFFS2* fájlrendszer használatának köszönhető. Szintén kiemelendő a hozzáférhető alkalmazások széles köre. Mindehhez ráadásul a tálalás is tökéletes, hiszen a *Flash Linux Desktop* animált és árnyékot vető egérkurzorokkal, szép ablakdekorációkkal érkezik.

Az egyetlen említésre méltó hátránya talán az, hogy kicsit körülményes a *pendrive*-on való beüzemelése. Igazából azt se nagyon értem, miért nem része a telepítőszkript magának a terjesztésnek, miért kell az külön letölteni.

Összbenyomás, záró megjegyzések

Az asztali rendszerek használói számára a *pendrive*-ról futtatható *LiveUSB* terjesztések fantasztikus segédeszközt jelentenek. Akinek tehát van egy fölösleges *USB* kulcstartója, annak érdemes egy kicsit kísérletezni. Én elsősorban a következő terjesztések kipróbálását ajánlom: *DSL*, *SLAX*, *Flash Linux*, *Feather*, *Puppy*.
Az olvasó tapasztalatai és preferenciái természetesen eltérhetnek az enyémektől, de nekem a *SLAX* volt a legszimpatikusabb. A rendszer moduláris felépítéséből adódóan jelentősen kitágul a lehetséges felhasználások köre, amit egy profi *Linux*-felhasználó nyilván értékelni fog. A *SLAX*-hoz például tartoznak külön biztonsági modulok, használhatjuk vele a *ClamAV* víruskeresőt, futtathatunk rajta *Qt*-re támaszkodó grafikus alkalmazásokat, építhetünk belőle tűzfalat, és így tovább.
Aki gyakran dolgozik biztonsági *Live CD*-kkel, annak a *pendrive* mint hordozó és a *LiveUSB* terjesztések ideális választást jelenthetnek, hiszen így két problémát oldhatnak meg egyszerre: menthetnek adatokat, ha pedig fejleszteni szeretnének, akkor a megfelelő biztonsági és fejlesztői modulokat is a *pendrive*-on tarthatják.

Linux Journal 2006., 146. szám

Juan Marcelo Rodriguez sok éve dolgozik a GNU/Linux rendszerrel. Különböző magazinoknak írt már cikkeket, aktív tagja egy helyi felhasználói csoportnak, illetve dolgozik a LugAR/USLA szervezetnek is. A billentyűzet püfölése mellett szeret olvasni, írni és zenét hallgatni.

KAPCSOLÓDÓ CÍMEK

A cikkhez tartozó on-line források a következő címen találhatók:
➲ www.linuxjournal.com/article/8949

NAGIOS – Aki mindenről elsőként értesül

Ami el tud romlani, az egyszer el is fog. A szolgáltatási szint megállapodások (SLA) világában azonban a problémákról minél hamarabb értesülnünk kell. Erre a feladatra kiváló eszköz a nagios, amely egy rendkívül sokoldalú monitorozó program. Bármit képes nyomon követni, a web kiszolgálónk elérhetőségétől kezdve, a kvóta túllépésen át, egészen egy adott kapcsolat csomagvesztésének arányáig.

Telepítés

A ***nagios*** telepítéséhez töltsük le a legfrissebb verziót a ➲ http://www.nagios.org/ oldalról, majd csomagoljuk ki, fordítsuk le, és telepítsük az *1. listában* látható módon.

A ***nagios*** valójában semmit nem monitoroz, hanem egy keretrendszert biztosít, amely különféle segédprogramokat (***plugin***) használ a szolgáltatások vizsgálatára. Ezért a ***nagios*** addig semmi hasznosat nem tehet, amíg nem adunk neki hozzá „szerszámokat". Töltsük hát le ezeket az eszközöket a honlapról, az „***Official Nagios Plugins***" címszó alatt találjuk meg, majd telepítsük ezt is a szokásos módon (*2. lista*).

Számtalan programot gyűjtöttek össze, amelyek többek között lehetőséget adnak ***Oracle*** adatbázis monitorozására, ***SNMP*** lekérdezések elvégzésére, ***RADIUS*** szerver ellenőrzésére, ***LDAP*** keresésekre, stb. Érdemes elolvasni a ***REQUIREMENTS*** fájlt, amelyben le van írva, hogy melyik ***plugin*** telepítéséhez mi szükséges. Ha nem találjuk meg itt a szükséges programot, magunk is elkészíthetjük azt, de erről majd később.

1. Lista A nagios telepítése

```
groupadd ng
useradd -g ng -d /opt/nagios
➥-s /bin/sh ng
tar zxvf nagios-2.5.tar.gz
cd nagios-2.5
./configure -prefix=/opt/
➥nagios -with-nagios-user=
➥ng -with-nagios-group=ng
➥-disable-statuswrl -with-
➥cgiurl=/cgi-bin/nagios
make
su -c 'make install; make
➥install-config'
```

2. Lista A nagios segédprogramok telepítése

```
tar zxvf nagios-plugins-
➥1.4.3.tar.gz
cd nagios-plugins-1.4.3
./configure -prefix=/opt/
➥nagios -with-nagios-user=
➥ng -with-nagios-group=ng
➥-with-cgiurl=/cgi-bin/
➥nagios
make
su -c 'make install'
```

Konfiguráljunk!

A */opt/nagios/etc* könyvtárban találhatóak a ***nagios*** példa konfigurációs fájlok egy „***-sample***" végződéssel. Nevezzük át mindet úgy, hogy eltávolítjuk ezt, azaz például a ***nagios.cfg-sample*** állományból ***nagios.cfg legyen***. Nézzük meg ezeket a fájlokat!

A ***nagios.cfg*** tartalmazza a ***nagios*** futásával kapcsolatos paramétereket, például hova naplózzon, milyen felhasználó nevében fusson, különböző naplózási-, riasztási beállításokat, stb. A legtöbb változó használható alapértelmezett értekkel rendelkezik, első körben elég a ***cfg_file*** paramétert beállítani, amely azt határozza meg, hol vannak az objektum definíciók. Ha csak néhány objektumot akarunk monitorozni, akkor jó választás a ***minimal.cfg***, amely minden információt egyetlen helyen definiál. Nagyobb hálózatok esetén azonban lehetőség van az azonos típusú leíró információk külön állományokba szervezésére.

A ***nagios*** a riasztásokat a `mail` parancs segítségével küldi ki, beállítástól függően e-mail címre vagy mobil telefonra ***SMS*** üzenetben. A konkrét módja és formátuma a ***misccommands.cfg*** állományban található, az alapértelmezett értékek általában megfelelőek.

A ***nagios*** a használt segédprogramokat, amelyek a tényleges munkát végzik, a ***checkcommands.cfg*** fájlban definiálja, amelyben a ***nagios-plugin*** csomag programjai vannak felsorolva. Ha saját programot is akarunk használni, azokat is itt kell megadni.

A cikkben egy egyszerű, néhány elemből álló konfigurációt mutatok be. Első lépésként szerkesszük a ***minimal.cfg*** állományt!

3. Lista Az értesítési adataim

```
define contact{
        contact_name                    sj
        alias                           SJ
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,r
        service_notification_commands   notify-by-email,notify-
                                        ➥by-epager
        host_notification_commands      host-notify-by-email,
                                        ➥host-notify-by-epager
        email                           jsuto@freemail.hu
        pager                           1234567@sms.pgsm.hu
        }
```

4. Lista csoport egyetlen taggal

```
define contactgroup{
        contactgroup_name       admins
        alias                   Nagios Administrators
        members                 sj
        }
```

5. Lista Egy gép paraméterei

```
define host{
        use                     generic-host
        host_name               sj.acts.hu
        alias                   sj.acts.hu
        address                 192.168.1.22
        check_command           check-host-alive
        max_check_attempts      10
        check_period            24x7
        notification_interval   120
        notification_period     24x7
        notification_options    d,r
        contact_groups          admins
        }
```

Távolítsuk el a *„COMMANDS”* rész `define command` bejegyzéseit, hiszen ezek a parancsok más állományokban már szerepelnek. Ezután készítsük el a kontakt bejegyzésünket, az enyém a *3. listában* látható módon néz ki. Megadtam a nevemet, email címemet, illetve *SMS*-elérhetőségem, továbbá hogy állandóan (7x24) jöjjenek az értesítések, illetve hogy mind az e-mail címemre, mind a mobilomra küldjön értesítést az eseményekről. Természetesen több kapcsolattartói bejegyzést is készíthetünk.
Az értesítendő személyeket csoportokba lehet összefogni. Ennek az a haszna, hogy ha például 5 személy felügyeli a hálózatot, akkor elég a csoportra hivatkozni. Igény szerint több csoportot is létre lehet hozni, és mindegyikben több személy is lehet, ez azonban egy kis hálózat egyetlen emberrel, amint az a *4. listából* is látszik. Több tag esetén vesszővel elválasztva lehet őket felsorolni.
A `HOSTS` szekcióban kell felsorolni az egyes objektumokat (azaz jellemzően számítógépeket, útválasztókat, nyomtatókat és más hálózati eszközöket), amelyeket monitorozni akarunk.
Az én egyik gépem az *5. lista* szerint néz ki.
Megadtam az objektum nevét, IP-címét, a 2 értesítés közötti időtartamot (2 óra), hogy a riasztásokat az `admin` csoport kapja meg, ill. az objektum állapotát ellenőrző program nevét (***check-host-alive***), amely nem tesz mást, mint a `ping` programmal ellenőrzi a gép elérhetőségét.
Végül az egyes gépekhez szolgáltatásokat kell rendelnünk, amelyeket a ***nagios*** ellenőrizni fog. Az előbbi gépen futó web kiszolgáló elérhetőségét a 6. listában látható bejegyzés hatására fogja megvizsgálni a ***nagios*** *(6. Lista)*. Megadtam a gép nevét, a szolgáltatás megnevezését (***HTTP***), állandóan (24x7) figyeljük, 5 percenként fut le az állapotot figyelő program, 4 óránként (240 perc) küldünk újabb riasztást, ha addig a hiba nem hárul el, itt is az `admin` csoport kapja meg az értesítéseket, továbbá a `check_http` néven szereplő program végzi az ellenőrzést. Egy géphez természetesen több szolgáltatás is rendelhető, ebben az esetben minden szolgáltatáshoz egy ***‘define service’*** blokkot kell készíteni.
Már csak egy lépés van hátra: a konfiguráció ellenőrzése, amelyet a

```
/opt/nagios/bin/nagios -v
➥/opt/nagios/etc/nagios.cfg
```

parancs segítségével tehetünk meg. Ha minden rendben van, akkor a *7. listához* hasonló üzenetet kapunk. Ezek után indítsuk el a

```
/opt/nagios/bin/nagios
➥/opt/nagios/etc/nagios.cfg &
```

paranccsal a háttérben.

Levelezzünk!

A ***nagios*** az eredményeket egy naplóban vezeti, amely a */opt/nagios/var* könyvtárban található. A minden

6. Lista Egy web kiszolgáló definiálása

```
define service{
        use                             generic-service
        host_name                       sj.acts.hu
        service_description             HTTP
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              4
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  admins
        notification_options            w,u,c,r
        notification_interval           240
        notification_period             24x7
        check_command                   check_http
        }
```

7. Lista A nagios konfigurációs állományai rendben vannak

```
Nagios 2.5
Copyright (c) 1999-2006 Ethan
➥Galstad
➥(http://www.nagios.org)
Last Modified: 07-13-2006
License: GPL

Reading configuration data...

Running pre-flight check on
➥configuration data...

Checking services...
        Checked 3 services.
Checking hosts...
        Checked 2 hosts.

.... sok szöveget kiír még

Total Warnings: 0
Total Errors:   0

Things look okay - No serious
➥problems were detected
➥during the pre-flight check
```

ellenőrzéskor frissülő ***status.dat*** állományban található az összes gép szolgáltatása, ahol a beállítások mellett látható a használt ***plugin*** tesztjének eredménye is, amely például a *8. listán* látható módon nézhet ki, ha minden rendben van.

8. Lista A status.dat fájl részlete

```
service {
        host_name=sj.acts.hu

service_description=HTTP
....
        plugin_output=HTTP OK
➥HTTP/1.1 200 OK - 7342
➥bytes in 0.002 seconds
        performance_data=
➥time=0.002078s;;;0.000000
➥size=7342B;;;0
....
        last_check=1161681000
        next_check=1161681300
}
```

A gépemen a ***ping*** programról levettem korábban a ***setuid*** bitet, ezért a ***nagios*** hibát jelzett, amit – lásd *9. lista* – elküldött e-mailben. A

```
setfacl -m u:ng:x /bin/ping
```

paranccsal korrigáltam a hibát, aminek hatására pár percen belül meg is jött a *10. listában* látható nyugta, hogy minden rendben.
Mindkét eseményről értesítést kaptam a mobilomra is. Előfordulhat, hogy egy ***SMS*** túl hosszú, hogy 1 üzenetbe beleférjen, ebben az esetben a mobil operátor feldarabolja. Ebben az esetben érdemes szerkeszteni

9. Lista Riasztás e-mailben

```
Date: Tue, 24 Oct 2006
➥11:08:30 +0200
From: ng@acts.hu
To: jsuto@freemail.hu
Subject: Host DOWN alert for
➥localhost!

***** Nagios  *****

Notification Type: PROBLEM
Host: localhost
State: DOWN
Address: 127.0.0.1
Info: /bin/ping -n -U -w 10
➥-c 1 127.0.0.1

Date/Time: Tue Oct 24
➥11:08:30 CEST 2006
```

10. Lista Levél a szolgáltatás helyreállásáról

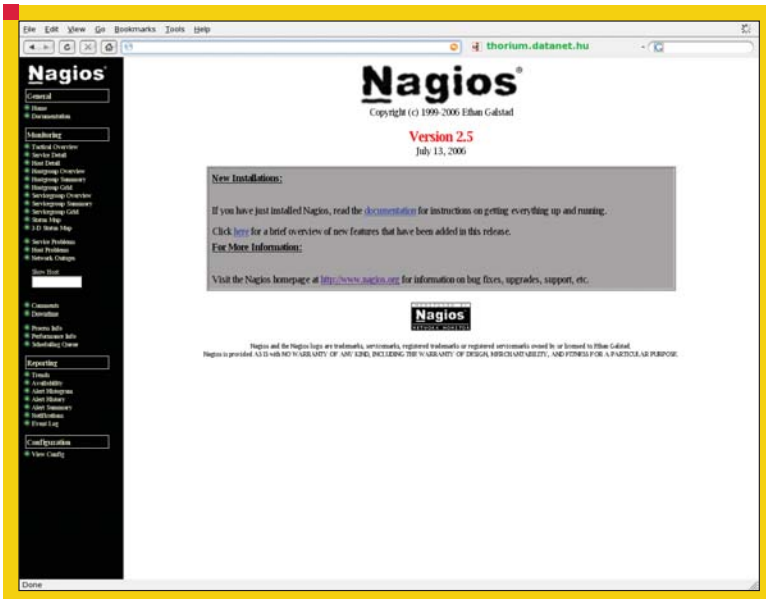
```
Date: Tue, 24 Oct 2006
➥11:13:30 +0200
From: ng@acts.hu
To: jsuto@freemail.hu
Subject: Host UP alert for
➥localhost!

***** Nagios  *****

Notification Type: RECOVERY
Host: localhost
State: UP
Address: 127.0.0.1
Info: PING OK - Packet loss
➥= 0%, RTA = 0.05 ms

Date/Time: Tue Oct 24
➥11:13:30 CEST 2006
```

a ***misccommands.cfg*** fájlt, és fazonra igazítani, hogy rövidebb legyen, néhány elem elhagyható belőle.
A ***nagios*** azonban többet is tud ennél, egy web oldalon megnézhetjük az egyes gépek ill. szolgáltatások állapotát, a múltbeli eseményeket, grafikonokat, trendeket, stb. Ehhez telepítsünk egy web kiszolgálót (ha még nincs). A példában egy dedikált ***Apache virtualhost***-ot fogunk a ***nagios***

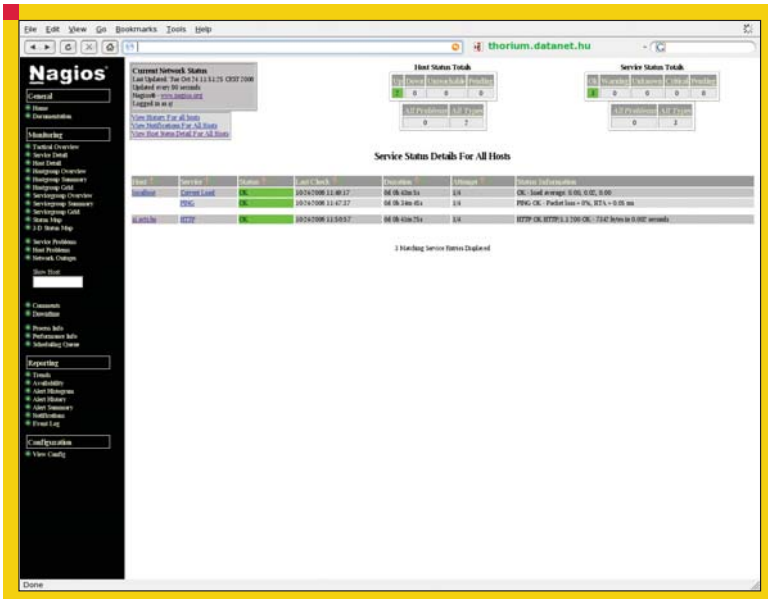
11. Lista Az Apache kiszolgáló beállítása

```
<VirtualHost 192.168.1.22:80>
   ServerName nagios.ceg.hu
   DocumentRoot
   ➥/opt/nagios/share

   <Directory
   ➥/opt/nagios/share>
      Options Indexes
      AllowOverride Limit
      ➥AuthConfig
   </Directory>

   <Directory
   ➥/opt/nagios/sbin>
      AuthUserFile
      ➥/opt/nagios/sbin/
      ➥.htpasswd
      AuthName "nagios"
      AuthType basic

      require valid-user

      order deny,allow
      deny from all
      allow from 192.168.1.22

      satisfy all
   </Directory>

   ScriptAlias /cgi-bin/
   ➥nagios/ /opt/nagios/sbin/
</VirtualHost>
```

rendelkezésére bocsátani.
A *11. listában* látható konfigurációs fájl részlet biztosítja, hogy a ➲ http://nagios.ceg.hu/ címen meg tudjuk nézni a szolgáltatások állapotát. A */opt/nagios/sbin* könyvtárba pedig tegyünk egy *.htpasswd* fájlt is tenni a megfelelő hozzáférés szabályozás érdekében.
Ahhoz, hogy egy böngészővel elérhessük a *nagios* eredményeit, szerkesszük a *cgi.cfg* fájlt, és az `authorized_for_*` paraméterekhez adjuk hozzá saját azonosító nevünket, amely a *.htpasswd* állományban is szerepel, különben hibát fogunk kapni a web felületen.
Nyissuk meg böngészővel a *http://nagios.ceg.hu/* címet.
Ekkor az *1. ábrához* hasonló képet fogunk kapni.

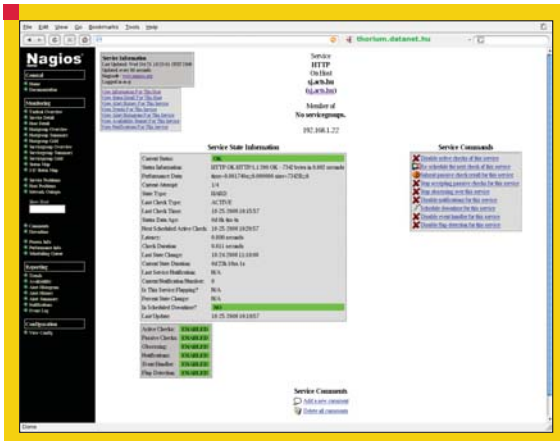


1. ábra A nagios bejelentkező felülete

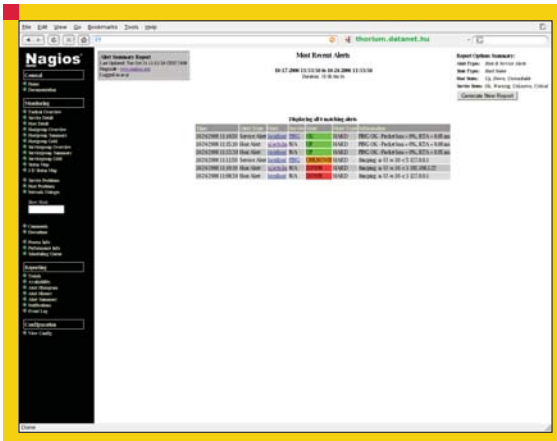


2. ábra Az egyes gépek összes szolgáltatásának állapota

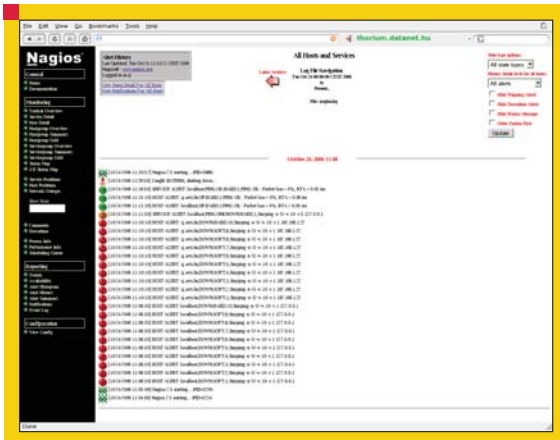
Leggyakrabban a *„Service Detail”* menüt szoktam használni, ahol – mint az a *2. ábrán* látszik – könnyen át lehet tekinteni az egyes gépek szolgáltatásainak állapotát.
Az adott gép nevére vagy valamelyik szolgáltatására kattintva annak státusza a *3. ábrán* látható módon jelenik meg, ahol több statisztikai adat is szerepel, például hányadik próbálkozás eredményét látjuk, mikor futott le az ellenőrzés, mi az eredménye, mikor lesz a következő teszt, stb.
Az *„Alert Summary”* menüre kattintva egy űrlap segítségével be tudjuk állítani (például gép, szolgáltatás, dátum), hogy a riportban pontosan mi szerepeljen. Egy lehetséges kimenet a *4. ábrán* látható.
Az *„Alert History”* menü az adott napi riasztásokat mutatja meg. Az *5. ábrán* látható az összes riasztás részletei, például dátum, gép neve, állapot, típus és a használt segédprogram a paramétereivel együtt. Az állapot lehet úgynevezett *HARD* vagy *SOFT*. Ha egy ellenőrzés nem sikerül, akkor az adott szolgáltatás *SOFT* állapotba kerül. Ha a következő alkalommal rendben válaszol, akkor nincs riasztás. Ha 4 egymást követő alkalommal (`max_check_attempts` változóval



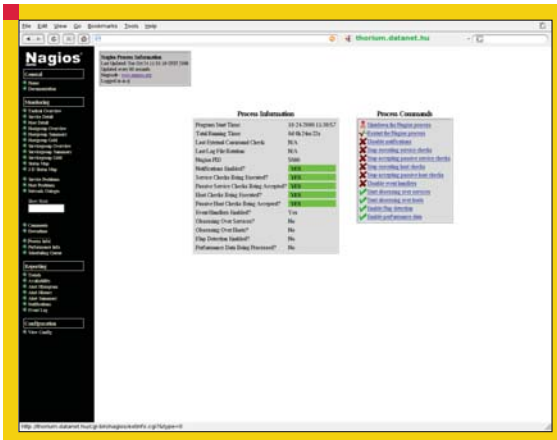
3. ábra A kijelölt gép állapota



4. ábra A legutóbbi riasztások



5. ábra A mai riasztások



6. ábra A nagios állapota

12. Lista Az első nagios segédprogramom

```
#!/bin/sh

LIMIT=100
START=`date +%s`
HOST=localhost

N=`mysqladmin -defaults
➥-file=~/.my.cnf processlist
➥| wc -l`

STOP=`date +%s`

RUNTIME=`expr $STOP - $START`

echo "$START;$HOST;$RUNTIME;
➥kapcsolatok szama: $N"

if [ $N -gt $LIMIT ]; then
➥exit 2; fi
exit 0
```

konfigurálható érték a *nagios.cfg* fájlban) is elérhetetlen vagy nem megfelelő a szolgáltatás, akkor az állapot *HARD* lesz, és riasztást küld a *nagios*. Ezzel a mechanizmussal elkerülhető, hogy egy rövid átmeneti zavar miatti *SMS* felébressze az üzemeltetőket éjjel.
A *nagios* folyamat állapotáról a *6. ábrán* látható módon is kaphatunk néhány információt, például *PID*, indítás ideje és néhány konfigurációs érték.

Szabjuk testre!

A *nagios* lehetőséget ad arra, hogy saját segédprogramot is használjunk. Ehhez írnunk kell egy tetszőleges nyelvű (például Perl, C, TCL, bash, stb.) programot, ami elvégzi az adott szolgáltatás ellenőrzését, kiír néhány adatot, végül beállítja a visszatérési értéket. Nézzünk meg a *12. listában* szereplő példát, amely leellenőrzi a *MySQL* adatbázist, és riaszt, ha túl sok nyitott kapcsolat van.
Látható, hogy a program visszatérési értéke 0, ha minden rendben van, illetve hiba esetén 2. Az összes lehetséges visszatérési érték a */opt/nagios/libexec/utils.sh* fájlban található.
A *nagios* nem csak rendszeres lekérdezések elvégzésére használható, de megfelelő program segítségével akár reagálni is lehet az esetleges hibákra. Az előbbi példában amellett, hogy riasztást küld a túl sok szimultán kapcsolatról, akár újra is indíthatja (megfelelő jogosultság birtokában) az adatbázis kiszolgálót. A *nagios* legfőbb erénye azonban az, hogy a lehetőségeknek csak az Olvasó képzelete szab határt. Ha egy igazán rugalmas és megbízható monitorozó alkalmazást keres, a *nagios* kitűnő választás.

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

Keresőoptimalizálás
Avagy hogyan használjuk ki a legjobb ingyenes reklámfelületet, a keresőket

A Google neve mára gyakorlatilag összeforrott az internetes kereséssel, ha valamilyen információra van szükségünk, akkor az esetek túlnyomó többségében az interneten keresztül, a keresők segítségével próbálunk tájékozódni. Ez érthető, mivel jelenleg nincs az internettel összemérhető nagyságú adathalmaz, melyben valószínűleg megtaláljuk kérdésünkre a választ.

Mi a keresőoptimalizálás?

Sokan sokféleképpen megfogalmazták már a keresőoptimalizálás jelentését, lényegét. Az angol ***Search Engine Optimalization (SEO)*** szóösszetétel talán jobban kifejezi azt, hogy mi tulajdonképpen a keresők motorjának tetsző oldalt készítünk, amikor arról beszélünk, hogy keresőoptimalizálást végzünk.

És vajon mi ezzel a célunk? Ha van egy szolgáltatásunk, vagy termékünk, akkor a potenciális ügyfelek, akiket termékünk, szolgáltatásunk érdekel, akkor a keresőkön keresztül minket találjanak meg elsőként a konkurenciához képest.

Túl „marketinges" megfogalmazás lenne? Lehet, de ha kicsit odafigyelünk arra, hogy hogyan keresünk, akkor észrevesszük, hogy ha az első három lapon nem találunk semmi érdemlegeset, akkor általában már nem is keresünk tovább. A kereséshez további kulcsszavakat adunk , vagy szinonimákkal próbálkozunk. A keresőoldalak alapvető érdeke tehát, hogy az adott kulcsszóra a lehető legjobban illő webhelyet ajánlja fel, ettől válik hitelessé, ezért keressük fel újra és újra, abban bízva, hogy keresési igényeinket a lehető legjobban kielégíti.

Amikor keresőoptimalizálást végzünk, akkor úgy alakítjuk ki/át a honlapot, hogy a keresőknek „tetsszen", relevánsnak találják egy adott kulcsszóra.

Ez azonban semmiképpen sem azt jelenti, hogy az adott kulcsszót rengetegszer értelmetlenül felsoroljuk az oldalon, illetve a relevancia növelése érdekében még olyan kulcsszavakat is elhelyezünk az oldalon, aminek semmi köze amúgy a tartalomhoz. A munkánk a kulcsszavak helyes megválasztásával kezdődik, és sohasem ér véget. Ha nem fordítunk kellő figyelmet a honlapra, akkor hogy mások megelőzhetnek minket. A kulcsszavak megválasztására nagyon oda kell figyelni, ezzel később foglalkozunk.

Most szeretnénk megjegyezni, hogy sokan a keresőoptimalizálás szó szinonimájaként használják a keresőmarketing kifejezést, pedig teljesen eltérő jelentést hordoznak. A keresőmarketing a keresőkkel kapcsolatos reklámozási lehetőségeket jelenti. Nagyon röviden: pénzért is megjelenhetünk a találati listákon, a „normál találatoktól" észrevehetően elkülönítve.

Tévhitek a keresőoptimalizáció kapcsán

- Sokan úgy gondolják, hogy bármilyen oldalt lehet optimalizálni, ezért gyors és látványos sikerekre számítanak. Ne essünk ebbe a hibába! Ha optimalizálásra adjuk a fejünket, akkor tudomásul kell vennünk, hogy ez nem néhány hetes munka, és valóban kemény munka, nem csodaszer.
- Ne felejtsük el, hogy hasznos, érdekes tartalom nélkül nem tudjuk oldalunkat a találati listák élére tornázni!
- Óvakodjunk az olyan cégektől, akik az oldalaink elemzése nélkül garantált helyezéseket kínálnak a keresőkben!
- Egyik cégnek sincsen „szerződése" a ***Google***-lel arra, hogy akit ők optimalizálnak, az garantáltan benne lesz ezért az első tízben! Aki tehát azt mondja, hogy garantálja, ezért előre kéri a pénzt, azzal nyugodtan legyünk bizalmatlanok.
- Ha rövid távon szeretnénk a keresőkből látogatókat csábítani az oldalunkra, akkor az átmeneti időben éljünk a ***keresőmarketing*** eszközeivel: például vásároljunk hirdetési felületet a keresők találati listájában! Érdemes kipróbálni az ***Etarget***, illetve a ***Google Adwords*** szolgáltatásokat!

A misztikus Google Page Rank

Talán sokan nem is tudják, de a ***Google*** valójában nem keresőnek indult. Készítői azon egyszerű elgondolás alapján írtak „web feltérképező" szoftvert, hogy a linkek csak egy irányba mutatnak. Vagyis tekinthetjük a webet egy irányított gráfnak, ahol a csomópontok az egyes oldalak, az irányított élek, pedig a linkek. Ezt a gráfot a hivatkozások mentén, vagyis a készítés irányában

könnyen bejárhatjuk, ellenben a másik irányba semmilyen lehetőségünk nincsen menni. A *Google* páratlan sikeréhez többek között az a tudásvágy vezetett, hogy megismerjük, milyen oldalak hivatkoznak ránk; ez ihlette a *Google Page Rank* bevezetését is.
Miután a legelső *Google* robot elkezdte bejárni az akkori webet, a fejlesztők észrevették azt, hogy ez a fajta link visszakövetéses rendszer rendkívül jó sorrendet állít fel a honlapok között aszerint, hogy melyik mennyire fontos. A nagyobb oldalakra értelemszerűen több hivatkozás mutatott, míg a kisebbekre sokkal kevesebb. Ezen hivatkozások alapján kalkulálódik a fejezet címében említett *PageRank* érték, mely 1-től 10-ig terjedően értékeli az oldalakat. Azonban a *PR* nem minden, sőt az utóbbi időben csökkent a jelentősége, ne ennek a maximalizálása legyen az egyetlen célunk! Az elgondolás alapján, ha adott két, ugyanarról a témáról szóló honlap, de az egyikre több hasonló témájú oldal hivatkozik, akkor nyilván az az oldal értékesebb. Persze ez azért nem ilyen egyszerű, mert nem mindegy, hogy milyen honlap hivatkozik ránk: egyik barátunk keveset látogatott eldugott honlapja nem egyenértékű például a *www.linuxvilág.hu* vagy a *www.fsf.hu* oldalról történő hivatkozással. A látható *Page Rank* melyet több helyen is ellenőrizhetünk körülbelül negyedévente frissül, ezzel szemben a háttérben a *Google*-nak mindig aktuális értéke van, melyet sajnos nem tudhatunk meg nap mint nap. Tehát ne csüggedjünk, ha sokáig nem változik az oldalunk *Page Rank* értéke, a *Google* azért tudja, hogy nem tétlenkedünk!

Hogyan találjunk jó kulcsszavakat?

Mielőtt belevágunk oldalunk elkészítésébe, gondolkozzunk el kicsit: mi a célunk? Milyen tartalmat szolgáltatunk? Melyek a célcsoportjaink? Mik a honlapunk előnyei? Miért érdemes a látogatóinknak újra és újra felkeresni minket? Ha ezzel megvagyunk (akár egy jegyzetfüzetbe is leírhatjuk, könnyebb eligazodni), akkor jöhetnek a kulcsszavak!
A keresőoptimalizálás egyik legfontosabb lépése a megfelelő kulcsszó megtalálása. Ez a gyakorlatban úgy néz ki, hogy összeírunk körülbelül tíz szót, amit kulcsfontosságúnak ítélünk a honlapon, majd az adott cég vagy mi magunk elvégezzük az optimalizálást.
A probléma csak az, hogy nem tudunk mások fejével gondolkodni, és bár logikusnak tűnhet, hogy az általunk kiválasztott keresőszavakra optimalizáljuk a lapunkat, de ezzel potenciális érdeklődőink közül sokat el is veszíthetünk. Saját tapasztalatunk volt, hogy az általunk kitalált 10 szóból 5 hozta be a keresőkből az emberek közel 95%-át, hiába voltunk a fennmaradó másik öt szóra is a top 10-ben.
A magyarázat nagyon ésszerű: arra az öt szóra, melyek alig hoztak látogatót, szinte senki sem keresett rá, így hát hiába volt jó helyezésünk, kevés látogató tért be hozzánk. Egy másik dologra is fény derült: hogy az emberek egy része a mi elképzelésünktől teljesen eltérő elgondolás alapján keresi szolgáltatásunkat, termékünket. Például sok ember keres úgy, hogy az adott termék többes számát használja keresőszóként, például „fogkefe”, „fogkefék”. Ha kihagyjuk kulcsszavaink közül a többes számot, akkor rögtön elesünk látogatóink egy részétől, sőt érdemes odafigyelni arra is, hogy bizonyos szavak toldalékolása a köztudatban nem egységes, például „olaj” szó többes számát „olajok” vagy „olajak” formában is használták már keresésnél.
Akkor hogyan találjuk meg a megfelelő kulcsszavakat? Kérdezzünk! Kérdezzük meg ismerőseinket, ha lehetőségünk van rá, célcsoportunknak megfelelő embereket, hogy ők hogyan keresnének rá az adott témára, utána pedig ellenőrizzük le a *Google* segítségével, hogy melyik szóra, kifejezésre hányan is keresnek rá. Célszerű azokat a szavakat kiválasztani, melyekre a legtöbb keresés érkezik; 10, de maximum 15 kulcsszó elég kell, hogy legyen, ha nem, akkor gondoljuk újra át, hogy miről is szól a honlapunk. Vagy ha két ennyire különböző profilt képviselünk, például fagylaltgépek szervizelése és *Túró Rudi* csomagolása akkor inkább csináljunk két teljesen eltérő profilú honlapot.
Most már megvannak a kulcsszavaink, a következő lépés, hogy felállítsuk fontossági sorrendjüket. Azt, hogy melyik kulcsszó mennyire fontos, nemcsak az dönti el, hogy mennyien keresnek rá, fontos szempont az adott kulcsszóra vonatkozó verseny erőssége is. Általában itt is elmondható, hogy a piaci rést kell megtalálni, vagyis érdemes azokra a kulcsszavakra összpontosítanunk, amelyekre sokan keresnek rá, de kevés honlapot optimalizáltak ezekre, vagy egyáltalán nem optimalizáltak rá webhelyet. Például ismert külföldi márkanevekre nagyon nehéz bekerülni akár csak idehaza is az első 10 közé, viszont

gyűjtőnevekkel nagyobb sikerrel próbálkozhatunk, például ásványvíz, felni, egérpad, modellvasút, fenyőbútor. Koncentrálhatunk a termékekkel kapcsolatos tevékenységekre is, például vasútmodellezés, bútorkészítés, bútorgyártás, kerékcsere.

Domain nevünk megválasztása

Amennyiben még nincsen saját tartománynevünk (domain nevünk), mert például egy ingyenes tárhelyen van a lapunk, akkor érdemes figyelembe venni az alábbi tanácsokat.
Ha már van domain nevünk, akkor mindenképpen a régi megtartása ajánlott, már csak azért is mert a ***Google*** figyelembe veszi a honlap korát. Tehát egy új weboldal eleve hátrányból indul a már 2-3 éve üzemelőkkel szemben.
Amiért viszont mégis megérheti domain-nevet váltani, vagy inkább új domain-nevet regisztrálni, az a számunkra legfontosabb kulcsszó elrejtése a domain névben. A ***Google*** nagy súllyal veszi figyelembe a domain nevet, tehát ha egyik legfontosabb kulcsszavunk benne van domain nevünkben, akkor már félig nyert ügyünk van. Ha a cégünket ***Asdfgh Kft***-nek hívják, és makszemek egyenkénti csomagolásával foglalkozunk, akkor kifizetődőbb a ***„www.makszem-csomagolas.hu”*** tartománynevet (is) regisztrálni, mint a ***www.asdfgh.hu***-t. Ugyanakkor álljunk ellen az egybeírt tartománynevek kísértésének (mint a ***www.makszemcsomagolas.hu***), mert az egybeírt szavakat a ***Google*** nem tudja különválasztani. Míg előző esetben mind a „makszem” mind a „csomagolas” szavakat megtalálja a tartománynévben, addig második esetben a „csomagolas”-t nem találja meg, mert a ***Google*** a szavak elejét próbálja meg összeilleszteni.
Merjünk olyan tartományneveket regisztrálni, mint például ***www.repa-retek-mogyoro.hu***, igaz, hogy kevésbé megjegyezhető, mintha csak sima ***www.repa.hu*** lenne, de a keresőoptimalizálás szempontjából előnyösebb az előbbi névválasztás.
Napjaink egyik újdonsága, hogy már ékezetes karaktereket is használhatunk a tartománynevünkben.
Ha fontos számunkra a jó helyezés a keresőkben, akkor ezt ne tegyük! Nem szabad elfelejtenünk, hogy a legtöbbet használt kereső itthon is a ***Google***, nekik pedig nem az a legfontosabb, hogy az ékezetes tartománynevekkel bajlódjanak. Persze valószínűleg egy idő után ezzel már nem lesz probléma, de egyelőre maradjunk az ékezetek nélküli tartományneveknél.
Ha pedig több tartománynevünk van, akkor különösen legyünk körültekintőek, és megfelelő módon irányítsuk át az egyiket a másikra! (301 permanent redirect). Erre pedig miért van szükség? A keresők nem szeretik ha ugyanaz a tartalom több oldalon is felbukkan, és ezt a duplikációt büntetik. Ezért kell tudatni a robotokkal, hogy a két oldal egy kézben van, nagyon figyeljünk erre!
Vágjunk tehát bele! Ha megtaláltuk a megfelelő tartománynevet és a kulcsszavakat, akkor elkezdődhet a tényleges munka. A keresőoptimalizálás két nagyobb részre osztható, egyrészt az oldalon belüli munkák, másrészt a külső hivatkozások optimalizálására.

Oldalon belüli optimalizálás

A keresőoptimalizálás ezen részéhez feltétlen szükség van felhasználó szintű ***HTML*** ismeretekre. Ha a következő pár sorban túl sok ismeretlen kifejezés lenne, akkor érdemes például a ***html.lap.hu*** oldalt meglátogatni.
Az oldal talán legfontosabb eleme (persze a tartalmon kívül) a ***title***. Nagyon fontoljuk meg, hogy mit írunk bele, a kulcsszavaink feltétlenül szerepeljenek, de az nem jó, ha csak szavakat sorolunk fel egymás után.
Ne feledjük, nem a keresőrobotoknak írjuk az oldalt, hanem a látogatóinknak! Célszerű egy frappáns szlogent kitalálni, amiben a lehető legtöbb kulcsszó szerepel, lehetőleg az elején, hiszen annak van nagyobb súlya.
A következő fontos elem az úgynevezett ***meta címke*** (***meta tag***), mely sokak szerint már vesztett jelentőségéből, de még mindig érdemes foglalkozni vele.
A ***meta tagek*** közül a ***description*** talán a legfontosabb, lehetőleg informatív szöveggel töltsük fel, természetesen a fontos kulcsszavakat innen se hagyjuk ki. Ennek azért van jelentősége, mert jó eséllyel a találati listánkon ez lesz az oldal leírása, azaz fel kell keltenie az érdeklődést! Megemlítendő még a ***keywords meta tag***, ahova kulcsszavakat írhatunk be. Csak olyanokat használjunk, mely az oldal tartalmában is szerepel, ha nem így teszünk, akkor ronthatunk az oldalunk minősítésén! Ne essünk abba a hibába, hogy 4-5 kifejezésnél többet próbálunk belezsúfolni, mert ezt sem értékelik a keresők.
Nézzük a lényeget, az oldal tartalmát. Először is fontos, hogy legyen valódi, egyedi tartalom az oldalon, különben hosszú távon nincs esélyünk (keressük

meg a jegyzetfüzetünket: célok, célcsoportok)… Igyekezzünk a kiválasztott kulcsszavakat különböző ragozott formában szerepeltetni a szövegben, lehetőleg minél többször. Persze ne essünk túlzásba, a keresőrobotok okosak, ha túl sűrűn szerepel egy szó az oldalon, akkor jön a büntetés! Használjunk címsorokat (*h1*,*h2*,*h3*), az itt szereplő szavaknak nagyobb jelentőséget tulajdonítanak a keresők, lehetőleg ilyen sorral kezdjük el az oldal tartalmát. A sok szöveg nagyon fontos, de nyilvánvaló, hogy nem csak ebből áll egy oldal, hanem képekből is. Ezeket is barátságosabbá tehetjük a keresők számára, használjuk az ***alt*** kiterjesztést, írjuk le, hogy miről szól az adott kép, hiszen a keresőrobotnak nincsen szeme, de olvasni annál inkább tud. Fontos szerepe van a kimenő linkeknek is, jól gondoljuk meg, hogy kire linkelünk, hiszen ez minket is minősít. Olyan oldalra ne mutasson kimenő hivatkozás, ami nem szerepel a keresők találati rangsorában, hiszen ez a mi oldalunkat is leminősíti. A külső hivatkozásokon kívül még nagyon fontos szerepe van a belső linkhálózatnak, azaz a főoldalról lehetőleg minden aloldalunkra mutasson link. Persze ha nagyon sok van, akkor ez nehezen megvalósítható, de a fontosabbak ekkor is legyenek kint a főoldalon, és mindenképp készítsünk oldaltérképet (egy dinamikusan változó fájl, melyen a honlapunk összes aloldala szerepel) a honlapunkról. A tartalom szerkesztésekor vegyük figyelembe, hogy a kód elején és a végén szereplő szövegeknek nagyobb súlyuk van, ezért célszerű ***CSS*** technikát alkalmazni, melynek segítségével nem feltétlen abban a sorrendben jelenik meg a tartalom a böngészőkben ahogy a kódban szerepel. Végül, de nem utolsósorban lehetőleg legyen az oldalunk kódja érvényes. Ezzel elkerülhetjük azt a kellemetlen helyzetet, hogy az általunk létrehozott tartalom csak bizonyos böngészők alatt olvasható.

Külső hivatkozások optimalizálása

Igazából amikor keresőoptimalizálást végzünk, akkor egy fekete dobozzal van dolgunk, senki sem tudja igazán, hogy hogyan működik a ***Google***, és természetesen ezt a titkot a fejlesztők a lehető legjobban őrzik, ha egyszer a teljes keresőmotor elérhetővé válna, akkor túl egyszerű lenne mindenkinek olyan honlapot készítenie, mellyel első lesz. Persze mindig csak egy honlap lehet első, de azzal, hogy mindenki ismerné a belső működést, jóval többen tudnának csalni, amely azt eredményezné, hogy a kereső elveszítené hitelét.
Tehát amikor optimalizálunk, akkor eddigi, és mások tapasztalata alapján próbáljuk meg úgy beállítani ennek a fekete doboznak a bemenetét (a honlapunk tartalma), hogy a kimenet a (kulcsszavakra történő legmagasabb helyezés) nekünk kedvező legyen. Ezeknek a technikáknak a többsége megfigyelésen alapul, vagyis rengeteget kell kísérletezni, és sok időt kell rászánnunk.
Ez különösen igaz a külső hivatkozások nyomon követésére, növelésére. Itt a cél az, hogy a lehető legtöbb honlapunkra mutató hivatkozást elhelyezzünk a weben. A linkeken pedig nagyon fontos, hogy a ráhelyezett szöveg (***anchor***) ki legyen töltve, és, mondani sem kell, lehetőleg a kulcsszavainkkal. Na de hogy érjük el, hogy egy másik weblapon elhelyezzenek egy ránk mutató hivatkozást? Konkrét válasz persze erre sincs, de arra igen, hogy hogyan ne tegyük. Ne spammeljünk, ne szemeteljük tele a netet. Ezzel csak sok embert felbosszantunk, az oldalunk rossz hírét keltjük, és nem is igazán érünk el vele semmit, hiszen ha nem a témánkba vágó oldalról kapunk hivatkozást, azt kevésbé veszi figyelembe a keresőrobot.
Na de akkor mégis mit tehetünk? Fektessünk hangsúlyt a hasznos tartalomra, ha ez megvan, akkor bátran ajánlhatjuk a lapunkat különböző katalógusokba (például ***www.startlap.com***, ***www.linkcenter.hu***, ***dmoz.org***). Érdemes még felkeresni azt, hogy a témakörünkkel kapcsolatban hol van „élet" a weben, azaz megkeresni a fórumokat, blogokat, és bekapcsolódni a beszélgetésekbe, de ne úgy, hogy belinkeljük a lapunkat kezdésként mindenféle hozzáadott tartalom nélkül, ez spamnek minősül. Legyünk kreatívak, találjuk meg a lehetőségeket, legyünk hitelesek, és előbb-utóbb azt fogjuk észrevenni, hogy „maguktól" szaporodnak a ránk mutató linkek.

Ellenőrizzük le munkánkat

Mivel a keresőoptimalizálás, egy képlékeny, mindig változó terület, ezért kiemelten fontos, hogy nyomon kövessük a munkánkat. Helyezzünk el statisztikai kódokat a honlapunkon (például ***www.google.com/analytics***, ***www.statgep.hu***, ***www.mystat.hu***), vagy használjunk webkiszolgáló log elemző programokat (pl. awstat), és folyamatosan kövessük a forgalmunkat. Figyeljük meg, hogy milyen oldalról jönnek hozzánk, keresők esetén pedig fontos, hogy milyen kulcsszóra keresve! Elemezzük a tendenciákat, ne legyünk makacsak, lehet, hogy nem az általunk kitalált kulcsszó a legsikeresebb, ebben az esetben cselekedjünk gyorsan!

Zsákutcák, ahonnan esetleg nincsen visszaút

Aki mer, az nyer! A kérdés az, hogy megéri-e olyan technikákat használni, melyek azt eredményezhetik, hogy honlapunkat a keresők kitiltják, vagyis keresőkön keresztül egy időre, vagy akár örökre nem lesz elérhető. Persze jó érzés ott lenni az összes fontos kulcsszóra a kereső első helyén, de jobb mindig inkább csak ötödiknek lenni, mint hosszú időre eltűnni onnan. Ne felejtsük el, a ***Google*** maga is észreveheti azt, hogy csalunk, de akár haragosaink, üzleti ellenfeleink is „feljelenthetnek" minket a ***Google***-nál.

Mik tehát azok a technikák, melyeket nem ajánlott használni?

- Az egyik legdurvább félrevezetési taktika az, amikor a honlapot kiszolgáló webszerver figyeli, hogy ki szeretne hozzá csatlakozni, és a keresők robotjainak más honlapot mutat, mint amit a ténylegesen böngésző felhasználóknak. Ez a technika azért „előnyös" mert bármi lehet a honlapon, a keresők ebből úgyis csak azt látják, amit mi akarunk, hogy lássanak. Tehát ha az oldalunk a zabhegyezésről szól, és a ***Google*** robotnak olyan oldalt mutatunk, ahol egy híres topmodell nevére optimalizált oldal szerepel, akkor a modell nevére keresők jó eséllyel nálunk kötnek ki. Ezt nevezhetjük „kereső spam"-nek hiszen ugyanannyira haszontalan minősül, mint a kéretlen levelek. Ezt a technikát a ***Google*** már figyeli (ezért tiltották ki a ***BMW*** oldalát egy ideig!), főleg külföldön,

ahol már komoly múlttal rendelkezik a ***SEO***, és bár talán még a hazai honlapok közül még nem történt kitiltás, ez bármikor bekövetkezhet.

- Mivel a keresők csak a szöveges részeket látják a honlapokból, ezért célszerű volt minél több, kulcsszavakkal teletűzdelt szöveget elhelyezni a lapon, és tekintve, hogy a túl gyakori ismétlés nem mutatna jól a weblapon, ezért ezeket valahogyan el kellett rejteni. Az egyik lehetőség az volt, hogy a szöveg és a háttér színét azonosra választották, így a szöveg láthatatlan maradt a kíváncsi tekintetek előtt, ugyanakkor a keresőrobotok rátaláltak, és beindexelték.
- Ehhez hasonló megoldás az, amikor a szöveget úgy helyezzük el a honlapon, hogy az a látható képernyő tartományon kívül essen. Ezeket a technikákat már szintén figyeli és bünteti a *Google*. Persze a webet böngészve találunk folyamatosan újabb és újabb megoldásokat, trükköket, de ezeket nem is részletezem, hiszen ha egyszer kiesünk a keresők találati listájáról, akkor valóban nagyon nehéz oda visszakerülni.

Ha nem szeretnénk önállóan optimalizálni

Az informatika mára megkerülhetetlen lett, egyre többen foglalkoznak vele, és egyre többen gondolják úgy, hogy minimális tudással maximálisan értenek a területhez. Ha nem adjuk oda a kocsinkat a vízvezeték-szerelőnek javításra; ha nem kérjük meg bőrgyógyászunkat arra, hogy végezzen el rajtunk egy rutin mandula műtétet, akkor ugyan miért kérnénk fel honlapunk optimalizálására olyan személyt, aki tőlünk hallja először a mágikusan hangzó keresőoptimalizáció kifejezést? Lehet, hogy nagyon jó barátunk a weboldal üzemeltetője, de ha ő alapból nem optimalizálta a honlapot, akkor inkább keressünk fel olyasvalakit, aki ezzel foglalkozik, vagy akár egy erre szakosodott céget.
Persze, mint az élet minden terén, itt is van különbség szakértő és szakértő között. Vannak olyan cégek, amelyek büszkék arra, hogy nem feltétlen elfogadott módszereket alkalmaznak (lásd a zsákutcáknál) mégsem tiltják ki őket a keresők, ezzel együtt inkább ne kockáztassunk.
Hogyan találjunk megfelelő céget? Ha nincsen megbízható ismerősünk, aki ajánlani tud valakit, akkor a legkézenfekvőbb, hogy a kapcsolódó kifejezésekre (pl. keresőoptimalizálás) rákeresünk a nagyobb keresőkben, és az elsők közül szemezgetünk. Hiszen ha egy cég a saját honlapját sem képes előkelő helyre vezényelni, akkor nyilván a mi honlapunknál sem brillíroznak. A kiválasztott cég honlapján nézzünk körül, ellenőrizzük a *Google Page Rank* értékét, jó esetben ez 4-es vagy magasabb. Ha nincsen *Page Rankje*, akkor jobb, ha elfelejtjük, hiszen lehet, hogy büntetésben van az oldal Figyeljünk a referenciákra is: fontos, hogy a kiválasztott cég milyen munkákat tud felmutatni. Ellenőrizzük a referencia oldalak *Page Rankjeit* is, és néhány logikus kulcsszóra keressünk rá, nézzük meg a helyezésüket. Érdemes a *site:www.optimalizalocegneve.com* parancsot megadni a keresőkben, erre megkapjuk, hogy hány oldalukat indexelték. Minél nagyobb ez a szám, annál jobb, hiszen egy jó minőségű weboldal egyik ismérve, hogy sűrűn frissül.
Azt is vegyük figyelembe, hogy a referenciaként megemlített weboldalakhoz tartozó kulcsszavakból hányat talál az Interneten a *Google*. Ha csak körülbelül 10.000-et, akkor abból nem nehéz az első tízben lenni, de ha már 1.000.000 találat felett járunk, és a lap az első tízben van, akkor bizonyára komoly céggel állunk szemben.

Zárszó

A keresőoptimalizálás témakörében már több elektronikus és nyomtatott könyv is megjelent, ezért a cikkben alkalmazott technikák nem garantálják rögtön azt, hogy a keresők első helyére kerülünk, és pozíciónkat meg is tartjuk hosszú időn keresztül, viszont a cikkben alkalmazott technikákkal jelenleg a magyar weben elég szép eredményeket lehet elérni. Természetesen teljes körű leírást, módszertant nem nyújt, nem is nyújthat ez a cikk, amolyan kedvcsinálónak szántuk, akinek felkeltette az érdeklődését a terület, azoknak az előző fejezetben említett honlapokat ajánljuk! Na és sok kávét, mert ha ebbe egyszer valaki beleássa magát, keveset fog aludni.

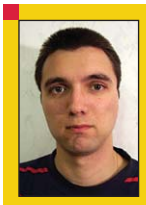
Hol szerezhetek még több információt?

A rengeteg angol nyelvű oldal mellett már szerencsére egyre több magyar nyelven elérhető oldalt is találunk, melyeket érdemes olvasgatni, sok hasznos információt nyerhetünk belőlük.

- ***www.google.com/support/webmaster*** – A *Google* útmutatója webmestereknek
- ***www.seochat.com*** – Hírek, elemzések, fórum angol nyelven
- ***www.mattcutts.com*** – A *Google* egyik vezető mérnökének blogja
- ***keresomarketing.fpn.hu*** – Olvasmányos, ugyanakkor nagyon hasznos útmutató
- ***keresomarketing.blogter.hu*** – Keresőmarketing, keresőoptimalizálás blog
- ***webni.innen.hu*** – Sűrűn frissülő oldal sok hozzászólással

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.



Szilágyi Péter
petszil@gmail.com

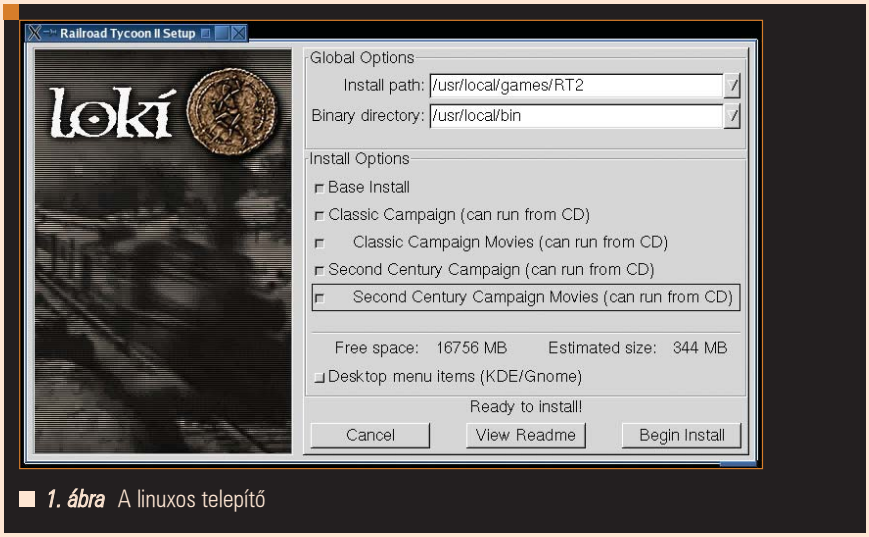
Pénzügyi területen dolgozik főállásban, de immár több, mint másfél éve a keresők világába vetette bele magát a szabadidejében, és a keresomarketing.blogter.hu oldalt szerkeszti nap, mint nap.
De ez már lassan több, mint hobbi, hiszen a felhalmozott ismerete egyre értékesebb.

Railroad Tycoon 2 Gold Edition

„A kettes vágányra tehervonat érkezik, kérjük vigyázzanak”! Valahogy így érdemes kezdenem a cikket, mely a Tycoon játékok egyik népszerű, kiforrott darabját próbálja röviden bemutatni: a program korabeli vasúthálózatok építésére és menedzselésére invitál bárkit, aki fogékony a stílusra.

Második felvonás

Azt hiszem „öreg motorosként”, számtalan végigjátszott játékprogrammal a hátam mögött bizton állíthatom: nem túl gyakori az olyan sikersorozat, melynek első tagján minden téren túlmutat a második rész. A *Railroad Tycoon* szerencsére a ritka kivételek közé tartozik: második felvonása sok tekintetben túltesz az egyébként szintén népszerű elődjén. A programot a *Pop Top Software* fejlesztői készítették, majd az akkor élete derekán lévő *Loki Entertainment* programozói ültették át *Linux* platformra (egyik első munkájukként). A linuxos verzió a *Gold Edition* kiadásból készült, ami annyit tesz, hogy a lemezen megtalálható a játék később kiadott *Second Century* című kiegészítője is.



1. ábra A linuxos telepítő

Miről van szó?

A vasút, mint jelenség biztosan sokakat vonz – így rengeteg modellező, illetve gyűjtő szeme felcsillan, amikor a vasutas „hajlamait” számítógépe előtt élheti ki. A terepasztal-mániákusoknak a *Win32* alapú *Trainz* nevű csoda lehet a megoldás, akik pedig a technika modellezése mellett gondolkodni és taktikázni is szeretnének, nos azok valószínűleg a *Railroad Tycoon* játékok epizódjaiban fogják megtalálni a szabadidejük végét: szállítmányozási, menedzselési, stratégiai játékról van szó, melynek minden mozzanata a vonatok világából táplálkozik.

Az említett második epizód lassan hét éves termék, így grafikai megvalósítása sajnos idejétmúlt. Cserébe viszont olyan hardveren is vígan fut, mely még egy alsó kategóriás irodai gép szintjét sem üti meg: *300MHz órajelű (x86* vagy *PPC)* központi egység, valamint *64MByte* központi memória társaságában már kompromisszumoktól mentesen használható – anélkül, hogy

■ *2. ábra* Menüben a Second Century!

■ *3. ábra* Megy a gőzös, megy a gőzös Vancouver-be...

bármilyen *3D* grafikai egységet kívánna (szerencsére a műfaj jellege nem is tesz szükségessé csillogó-villogó csúcsgrafikát). A teljes telepítés helyigénye *400MByte*, ami szintén átlag alatti. Röviden, tömören: érdekes és ötletes játékprogram, mely bármilyen gyenge gépen kiválóan fut.

Üzemebe állítás

Mivel *Scott Draeker* csapata már nem létezik, a *Lokitól* természetesen nem rendelhetünk telepítő lemezt. Beszerzés ügyén a két megszokott út áll a játékosok előtt. Egyikük a ➲ *http://www.tuxgames.com on-line* rendelési lehetősége (29 dollárért), a másik az illegalitáshoz közelít: a linuxos *Railroad Tycoon 2* a *p2p* fájlcserélő hálózatok gyakori vendége. Az első megoldást alapul véve, a megrendelt média megérkezése után fűzzük be a lemezt rendszerünkbe! A *CD* a *Loki* munkáira jellemző struktúrát rejt, így a gyökerében található *setup.sh* telepítő szkriptet *root* jogkörrel le kell futtatnunk. A grafikus interfésszel ellátott alkalmazás nem szorul különösebb magyarázatra: csupán annyit kell leírnom, hogy érdemes minden választható komponenst telepítésre kérni (a lemez így már nem fog kelleni játék idejére), a célmappát pedig az alapértelmezés szerinti */usr/local/games/RT2* úton hagyni. Miután az összes állomány a helyére került, a játékot fel kell peccselnünk a legutolsó *update* szkripttel, mivel az átirat alapvetően hemzseg a hibáktól. A *Loki* utolsó foltja *v1.54c* jelzéssel vonult a történelembe: a hivatalos oldalon ez már sajnos nem érhető el, így a néhai legenda *FTP* tükrét kell használnunk: az ➲ *ftp://sunsite.dk/mirrors/lokigames/patches/rt2/ URL* mögött keressünk *rt2_1.54c_(architektúra).run* nevű fájlt! Ezt letöltve, majd *root* jogokkal lefuttatva, a projekt linuxos motorja a kívánt verzióra vált (ami gyakorlatilag már hibamentes). A *Railroad Tycoon 2* ezek után felhasználóként, terminálra gépelt rt2 paranccsal indítható. A program a beállításainkat, mentett játékállasainkat kulturált módon, személyes mappánkban tárolja (*/home/$/.loki/RT2*).

■ *4. ábra* A vagonok, és a bőség zavara

Tapasztalatok

Íme, a feladat: adott államban, adott körzetben ki kell építeni a teljes vasúti hálózatot, mely személy- és teherszállításra, valamint postai feladatok ellátására is képes. Mindeközben a gazdaságos működtetésre, megtérülő fejlesztésekre is figyelmet kell fordítani, csakúgy, mint az állomások építésére és a betöltött szerepük meghatározására. Ezekből adódóan a *Railroad Tycoon 2* igen komoly kihívást jelent. A játékmenet századeleji gőzmozdonyokon, összetákolt állomásokon, kezdetleges szállító szerelvényeken keresztül vezet a *Second Century* kiegészítőhöz. Itt kezdésként a második világháború szakad a játékos nyakába, amikor is új terepen, megváltozott elképzelések szerint kell alkalmazni a kor technikáját, hogy az ügyes és türelmes stratéga búsás hasznot

húzhasson a hadi szállítmányozásból. E „küldetéslemez” kronológiája egészen a harmadik évezredbe repíti az érdeklődőt, ahol természetesen már nem a gőzmozdonyok húzzák az igát... A küldetések előtti hangulatot bevezető videók próbálják megalapozni: néhány másodperces, kis felbontású dokumentumfilmekről van szó, melyek minden egyszerűségük ellenére is megszerethetőek.
A kezelésről annyit érdemes tudni, miszerint a játéktér bal oldalán lévő gombsor segítségével lehet sínt fektetni és dózerrolni (a mutató leszúrásával, majd vonszolásával), vasúti létesítményeket emelni és elbontani, valamint új vonatot vásárolni - mindent „kellően” borsos áron (felhasználható tőkét a szállítmányozás hozamából, illetve állami segítségből remélhetünk).
A szerelvények kiválasztásához érdemes a telepítő lemezen található kézikönyvet kinyomtatni, mert az acélparipák sebessége, fenntartási költsége ezáltal már játék közben is böngészhető és összehasonlítható lesz. Erősebb és gyorsabb mozdonyt vásárolva kézenfekvő dolog meghizlalni a szerelvényeket: a vonatokon, majd a megfelelő állomásaikon dupla bal egérklikket nyomva lehet újabb kocsikat csatolni a sorba – komolyabb bevételek reményében. Magától értetődik, hogy a „pusztába” nem érdemes szerelvényt indítani, sokkal inkább javallott a fejlett városok felé kacsintgatni: az ipari létesítmények szállítmányozó ellátását komolyan megfizetik... A bal alsó térben a térkép található, jobb alul pedig az éppen kiválasztott egység: egy várost megkeresve, majd rákattintva megjelenik annak „profilja”, ami a helyből szállítható árukat nevezi meg (például alkohol, fa, élelmiszer, postai csomagok). A program negatívumaként talán csak a sokak által szóvá tett alagutak kiépítésének a hiányát tudom felróni, mivel az akadályokat kikerülni, vagy éppen domborzatra építeni nem mindig a legésszerűbb elgondolás.

5. ábra Seattle állomása

6. ábra Hírek a nagyvilágból

Szinte kötelező!

A *Railroad Tycoon* második része igazi, hamisítatlan megszállottságot feltételez: ha valaki rajong a stílusért, úgy akár hosszú hetekre a számítógépe elé lesz szegezve. Ha egy idő múlva esetleg unalmas lenne a játék, akkor a többjátékos szál és a pályaszerkesztő által érdemes újult erővel , újra és újra nekiesni a kihívásoknak. Még én is, akinek messze nem fekszik ez a stílus, cirka 25 órát töltöttem a virtuális terepen, hogy egy (középszerűnek is csak halkan mondható) társaságot fenn tudjak tartani... Amennyiben valaki nem tudna hozzájutni a játék teljes verziójához, úgy a linuxos demó példány elérhető a ➲ *http://download.softpedia.com/linux/games/rt2-demo.run* címen, minek telepítési módja megegyezik az előzőekben leírtakkal. A program nekünk átírt verziója semmilyen különbséget nem mutat a *Win32* platformra készült társától, így a stratégák számára teljes terjedelmében kötelező darab!

Kovács Zsolt (kovi@linuxforum.hu)

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

Videóvágás – minimalista módra

Ha az ember videót rögzít – legyen az TV felvétel vagy a születésnapi zsúr – óhatatlan, hogy felesleges részek is belekerülnek, amiket nem kívánunk megőrizni az utókor számára. Az ilyen részeket illik kivágni. Persze az is elképzelhető, hogy az anyag színvilága nem tetszik, de Linux alatt ezen is könnyen segíthetünk.

Egy igazi *Linux* guru képes akár parancssorból is vágni filmet *mencoder*rel. Persze ilyenkor nem kell meglepődni, ha a *Windows*on vagy *MacOS X*-en nevelkedett családtagok vagy kollégák a mentőket akarják hívni. Ennek egy fokkal kulturáltabb megoldás, amit jómagam is alkalmaztam egész sokáig: futtassuk az *ingyenesen használható Virtualdub*-ot vagy *Nandub*-ot *Wine* segítségével. Ez is csak ideig-óráig megoldás, hiszen sok *kodek szükséges* ezekhez a programokhoz, hogy akármilyen videófájlt be tudjanak tölteni, vagy el tudjanak menteni. Már-már reménytelennek tűnt a dolog, de feltűnt a színen az *avidemux2*...

Gépigény

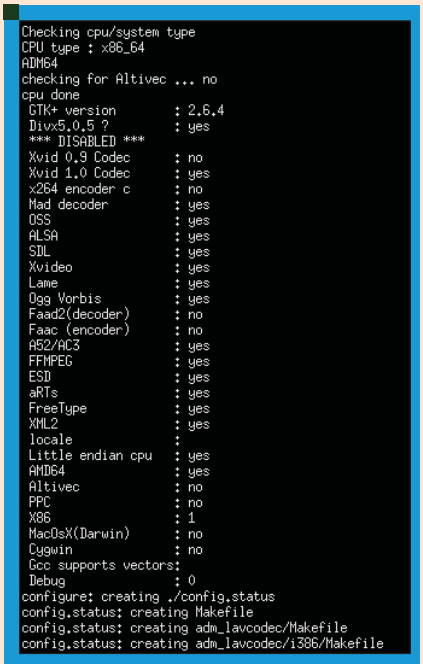
Az *avidemux2* gépigénye nem mondható túlzottnak, hiszen egy mai alsó-középkategóriás gépen is már elfut. Alsó-középkategóriás gép alatt jelenesetben egy minimum 1 GHz-es 256 megabájt memóriával és *AGP*-s videókártyával szereltet értek. Nyilván processzorból érdemes minél komolyabbat használni, mert különben gyakori lesz a „homokóra”. No nem a szerkesztés közben, hanem amikor végrehajtja a vágásokat a program és elmenti a végleges videót.
Nem mondok processzor márkát, hogy melyik jobb a videóvágáshoz. Mindenki nyilván olyat fog venni, amiben hisz és ami még belefér a költségvetésbe, de egy jó tanács: komolyabb processzorral érdemes belevágni, hiszen egy normál *Intel* vagy *AMD* processzor akár kétszer-háromszor is gyorsabb lehet ezek butított változatához képest (például *Celeron*, *Sempron*) A különbséget a processzor cache mérete adja, amely kb. 4-8-szoros (128/256 kbyte vs. 512/1024 kbyte) szokott lenni.
Illetve a másik dolog, amit jó ha tud a kedves Olvasó: hiába vett két- (vagy több) magos processzort, az *avidemux2* csak egy magot használ (jelenleg). Viszont nagy előny, hogy ilyenkor a többi magot másra használhatjuk.

Telepítés

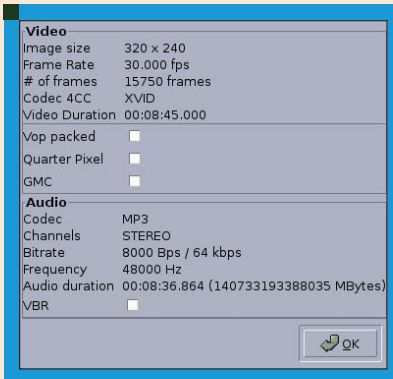
Az *avidemux2*-t nem szokták mellékelni a *Linux* telepítő *CD*-khez, hiszen folyamatosan változik, és itt az is igaz, hogy nem mindegy, mennyire van gépünkre optimalizálva a program, hiszen a régebbi architektúrák nem támogattak bizonyos parancsokat, amelyek már az újabb processzorokban jelen vannak. Éppen ezért ha modern processzorral dolgozunk, nem biztos, hogy optimális futást kapnánk egy *Pentium 2*-re szánt bináris programmal. (A mai disztribúciók többsége *Pentium 2*-re van optimalizálva.)
Jelen cikkben a 2.1.2-es (2006. március 1.) verziót fogom bemutatni, hiszen ez tekinthető az utolsó hivatalos kiadásnak, a többi egyelőre *preview* állapotban van.
Noha a forráskód kis méretű – a projekt honlapjáról csupán 4-5 megabájtot kell letöltenünk (kicsomagolva se sokkal több 20 megabájtnál), a fordításhoz már legalább 250 megabájt üres területre lesz szükségünk.
A telepítés a szokásos `configure, make, make install` trióval elvégezhető, azonban pár csomagot telepítenünk kell előtte (Debian alatt: pkg-config, libgtk2.0-dev, libsmjs-dev, libxml2-dev), amelytől függ a program. Persze ezenkívül fel lehet telepíteni további csomagokat is, de azok nem feltétlenül szükségesek a program fordításához. Az elérhető funkciókat a `configure` parancs a végén listázza ki.
Amennyiben a `jsapi.h` fájlt a libsmjs-dev csomag telepítése után sem találja a `configure` parancs, úgy használjuk a `–with-jsapi-include=<DIR>`, ahol a `<DIR>` az a könyvtár, ahol a `jsapi.h` található.
Amit érdemes feltelepíteni pluszként az adott projektek honlapjairól: *XVID*, *OSS* és/vagy *ALSA* támogatás, ha menet közben szeretnénk hangot hallani, *SDL* támogatás, *xvideo* támogatás a gyors kezelőfelületért, *lame* és/vagy *OGG* a hangtömörítésért. Nyilván ezekből nem elég a *lib* csomag, a fordításához szükséges az adott csomag fejlesztői változatai is (például *Debian*, *Ubuntu*, *UHU*, stb. esetén). Ha forrásból telepítettük ezeket, akkor legtöbbször az *avidemux2* `configure` parancsának meg kell adnunk, hogy az adott függvénykönyvtár hol található.

Alapok

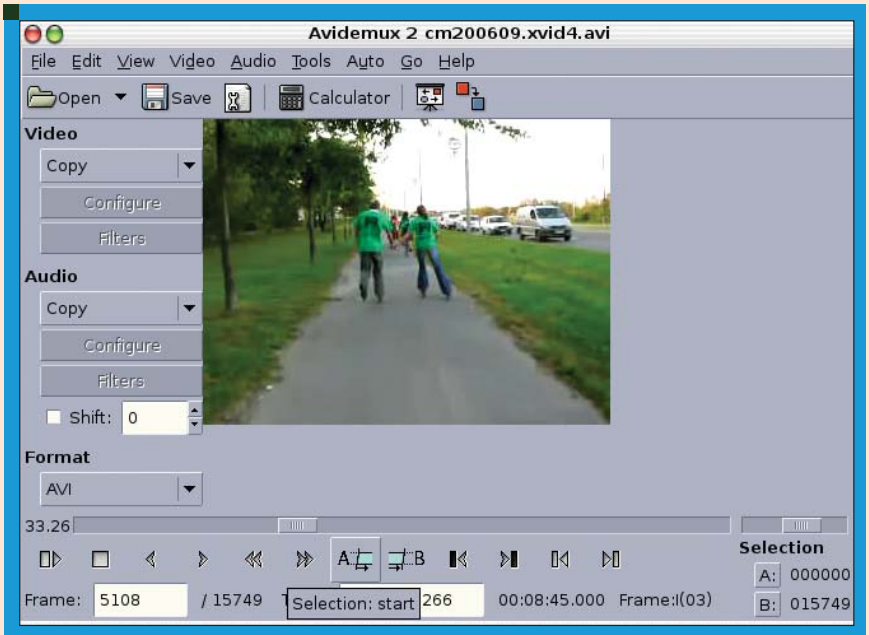
A program telepítés után az `avidemux2` paranccsal indítható.
A program nyelve a `LANGUAGE` környezeti változó segítségével állítható be.



1. ábra Így néz ki egy avidemux2 konfigurálás, ha az ajánlott kiegészítők is felkerültek



2. ábra Információk a megnyitott videóról



3. ábra Szegmens kijelölése

A 2.1.2 verzió esetén sajnos még nem tud teljesen magyarul a program. A program – mint már említettem – nem professzionális vágóprogram. Inkább kisebb vágásokra, filmek újratömörítésére ajánlanám. A filmekkel kapcsolatos beállítások és műveletek három csoportra oszthatóak: vágás, szűrők (***filters***) beállítása, tömörítés.

Vágás

Jelenesetben a vágás alatt például azt értem, hogy ha videóról digitalizálunk, akkor képkockára pontosan nem tudjuk elkapni a film elejét és a végét. Az ***avidemux2*** ezt nagyon egyszerűen meg tudja oldani: kijelöljük a szegmens elejét (A) és a végét (B), majd a szokásos vágólap műveleteket alkalmazhatjuk, jelen esetben a kivágást/törlést. Egérrel is könnyen kezelhető a program, de tapasztalatom szerint a billentyűzettel sokkal pontosabb és gyorsabb a pozicionálás. A billentyűzet jobb oldalán található számbillentyűzet használható (ne felejtsük el benyomni a ***NumLock***-ot). A 2-es és a 8-as gombokkal a kulcsképkockák között tekerhetünk, míg a 4-es és a 6-os gombokkal egy-egy képkockát. A szögletes zárójelekkel – „[„ és „]” – rögzíthetjük az aktuális szegmens elejét és végét (*3. ábra*).

A legtöbb filmtömörítési megoldás kulcskockákkal (***keyframe***) dolgozik. Röviden az elv: teljes képkockát csak bizonyos képkockák után mentek le – például megadott időközönként, vagy ha túl sokat változott a kép – a közbenső képkockákat mindig az előtte lévő kulcskockából számolják ki a programok. Éppen ezért ne lepődjünk meg, ha visszatekerésekor kicsit lassabb a program, mint előretekeréskor.

Szűrők beállítása

A szűrők beállítása előtt a videókodeket állítsuk át valamire, teljesen mindegy. Ha nem így teszünk, akkor nem férünk hozzá a szűrők dialógus ablakához. Az szűrők között többféle található. A legtöbbel csupán minimális módosításokat tudunk megejteni, például alakíthatunk a videó színvilágán, élesíthetjük a képet, szélesvásznúra alakíthatjuk, stb.

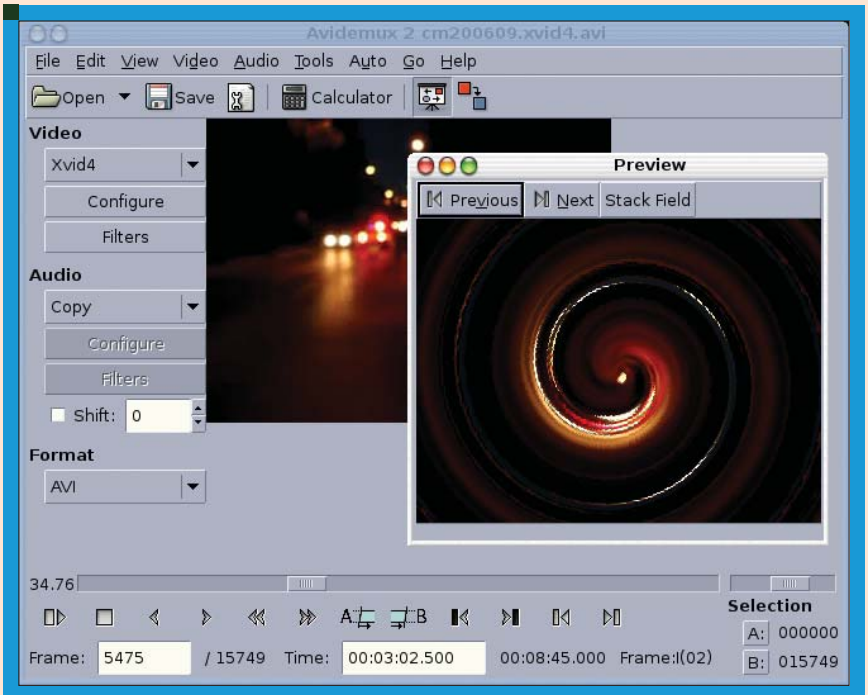
Van azonban olyan szűrő is, amellyel létfontosságú korrekciókat hajthatunk végre. Ilyen például a ***deinterlacer***. Ha például teljes felbontásban (704x576 képpont PAL esetén) veszünk fel mondjuk ***TV***-ből, akkor nagyobb mértékű vízszintes mozgásoknál a kép páros és páratlan sorai szétcsúsznak, fésűs lesz. A ***deinterlacer*** megfelelő matematikai műveletekkel ezt tudja korrigálni. Amennyiben azonban kisebb felbontásban veszünk fel (352x288 képpont ***PAL*** esetén), úgy felesleges ***deinterlacert*** bekapcsolni.

A szűrők között találunk például olyat is, amellyel ***srt*** formátumú feliratot „égethetünk” a videóra.

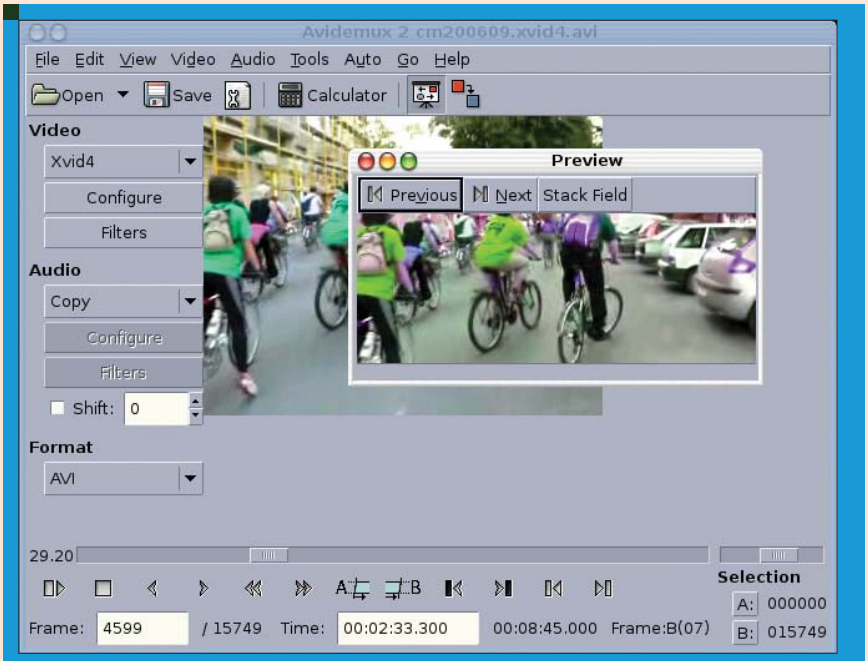
Érdekes, ámde többé-kevésbé haszontalan szűrő, amely elméletileg a tv csatornák logóit hivatott eltüntetni. A gyakorlatban ez csak halványítást jelent, hiszen a logó alatti részt nem találja, nem találhatja ki az algoritmus.

A filterek sorrendje, lévén, hogy az ember ritkán választ több hasonlót egy munkához, lényegtelen, azonban vannak esetek, amikor nem mindegy, hogy a listában melyik hol van (*7. ábra*).

Ha elkészültünk a szűrők összeválogatásával és beállításával, nyomjuk meg az ***F5***-öt (előnézeti ablak) és a csúszka alatt nyomjuk meg a lejátszást. Ekkor az előnézeti ablakban látható, hogy néznek ki az szűrők együtt.

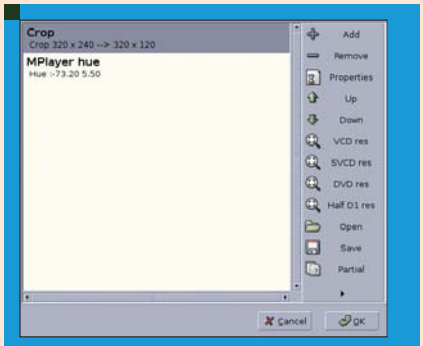


4. ábra Spirál szűrő és előnézete



5. ábra Színkorrekció

6. ábra Elérhető szűrők listája



7. ábra Használatban lévő szűrők listája

8. ábra Hang átalakítás

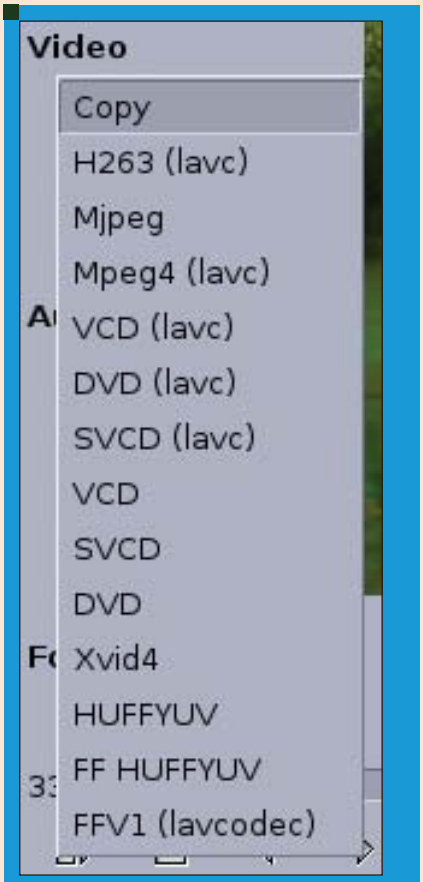
Tömörítés

Ha a vágás és a szűrők kiválasztásával megvagyunk, akkor itt az ideje elmenteni a végeredményt. Azonban egy jó tanács: mentsük el a projekt fájlt az elején. Ez az állomány írja le a vágásainkat és a szűrőket, illetve azok beállításait. A mentést azért javaslom, mert egy esetleges áramszünet esetén nem kell elölről kezdeni, már lesz egy projektfájl, amit csak be kell tölteni.

A tömörítés kodekenként különböző opciókat ad, de csak pár fontosabbat emelnék ki, hiszen a többi többnyire optimális, de akinek nem felel meg, az próbálgathatja.
Először jelöljük ki a használni kívánt audió és videó kodeket. Ha nincs a kedvenc kodekünk a listában, akkor töltsük le az adott projekt honlapjáról és fordítsuk újra az ***avidemux2***-t is.

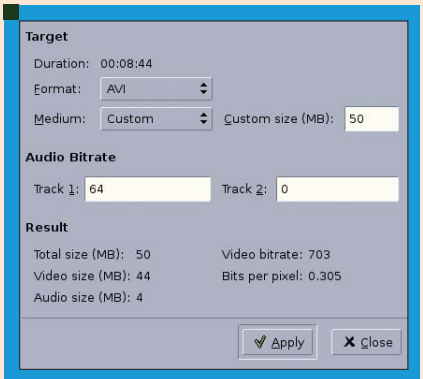
A videót (*12., 13. ábra*) jellemzően kétféleképp tömöríthetjük: fix vagy változó sávszélességgel. A fix sávszélesség előnye: hamarabb végez vele a kedves Olvasó, de nem ad optimális képminőséget. A változó sávszélesség optimális eredményt ad ugyan, de a program kétszer megy végig a videón. Az első alkalommal meghatározza, hova kell több illetve kevesebb bit. (Például egy autós

9. ábra Egy nem minimalista fordítás után elérhető hang kodekek

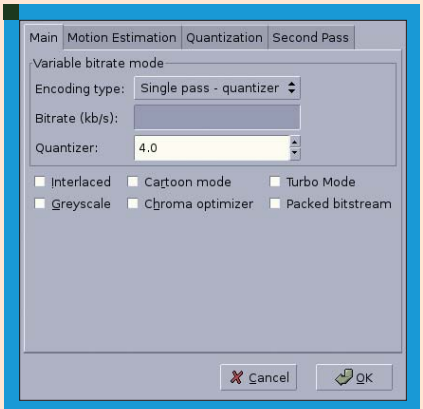


10. ábra Egy nem minimalista fordítás után elérhető videó kodekek

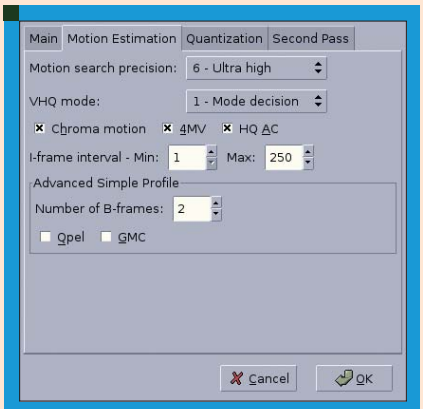
üldözésnél nagyobb sávszélesség szükséges, különben kockás lesz a kép, míg egy csendélethez alig kell valami, ott szinte nincs változás a képkockák között.) Második alkalommal pedig végrehajtja a tömörítést az első alkalommal letárolt statisztikai fájl segítségével. Az egy menetes tömörítésnél jellemzően sávszélességet vár a program, míg a két menetesnél jellemzően várt adat a fájl méretet.
A hangtömörítés (*14. ábra*) kicsit könnyebb dolog, hiszen ott 128 kbites *MP3* már többé-kevésbé kiváló hangminőséget eredményez.



11. ábra Beépített számológép, amellyel egyszerűen kiszámolható méret alapján a videó sávszélessége



12. ábra Xvid tömörítés beállításai



13. ábra Xvid tömörítés haladó beállításai

14. ábra Hangtömörítés beállításai

Végül a *Format*-nál kiválaszthatjuk, hogy milyen formátumban szeretnénk elmenteni a végeredményt. Ha *AVI*-n hagyjuk, akkor az esetek többségében megfelelő, kivéve, ha *MPEG*-et szeretnénk készíteni, mondjuk *DVD*-re. Hasznos választás még az *AVI dual channel*. Ez lehetővé teszi, hogy egy videóhoz két hangsávot tároljunk el. Ezeket az *Audió* menüben állíthatjuk be.
Ha mindennel megvagyunk, indítsuk el a mentést. Ekkor a program végrehajtja a vágásokat, szűrőket, tömörítéseket. A művelet a videó hosszától, az alkalmazott szűrőktől, a gép sebességétől és a tömörítési beállításoktól függően akár több órát is igénybe vehet.

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

Az Avidemux2 honlapja:
➲ http://www.avidemux.org/

Az Ogg honlapja:
➲ http://www.xiph.org/

Az Xvid honlapja:
➲ http://www.xvid.org/

A Lame honlapja:
➲ http://lame.sourceforge.net/

Videóvágás – ahogy a profik is csinálnák

Általában – ha filmvágásra adja a fejét a kedves Olvasó – nem árt manapság ilyen-olyan trükkökkel feldobni a végeredményt, mert a hollywoodi produkciók után a házi videóink – például egy nyaralás vagy egy születésnapi zsúr – unalmasak lesznek. Microsoft Windows alá egy komolyabb videóvágó program akár súlyos százezrekbe is kerülhet. De mi a helyzet Linux alatt?

A Cinelerra

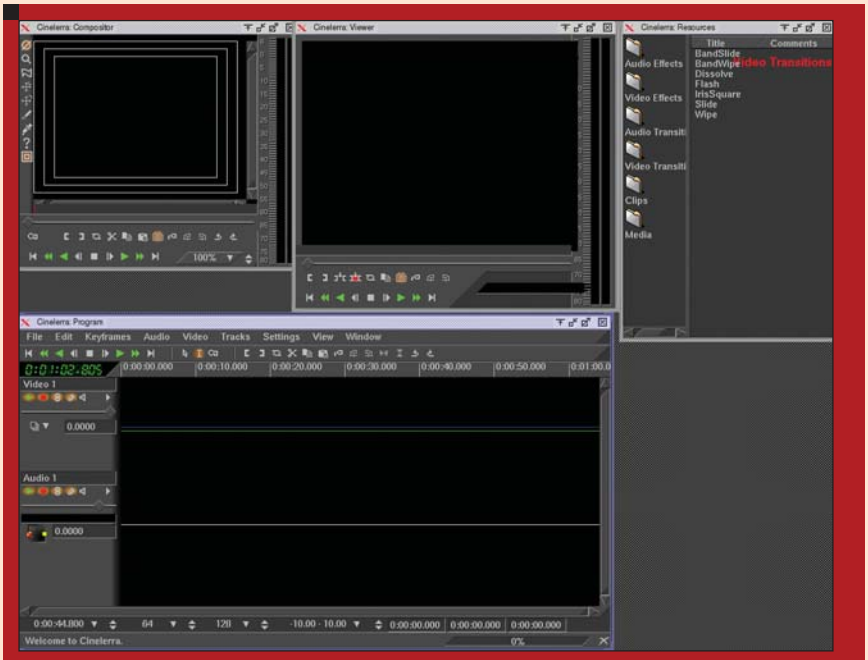
Linux alatt akár ingyenesen és emellett legálisan is hozzájuthatunk egész komoly videóvágó programhoz. Persze itt a kedves Olvasó a tudásával és az idejével fizet, hiszen míg a kereskedelmi vágóprogramok relatíve könnyen és intuitívan használhatóak, addig a későbbiekben boncasztalra kerülő *Cinelerra* használata nem mindig ilyen egyszerű. Persze a helyzet nem reménytelen, de azért azt tanácsolnám, hogy éles feladat előtt 1-2 napot szokjuk a programot.

Kell-e az a plusz 32 bit?

Nos az mindig is megosztotta az embereket, hogy kell-e 64 bites processzor egy átlagos számítógépbe. Jelenesetben nem szükséges – elindul 32 biten is –, de a program fejlesztői szerint jobban fut és valamivel kevesebb memóriával is beéri a *Cinelerra*, ha 64 bites processzoron futtatjuk.

Azok a függőségek...

A *Cinelerra* elég sok csomagtól függ, de ezek nagy része megtalálható a népszerűbb disztribúciókban. Ilyen például a *subversion*, valamint a *fortran*, *nasm* és *yasm* fordítók, a *libsndfile*, a *libjpeg62*, *libpng2*, *libpopt* csomagok fejlesztői változatai, illetve az *Ogg*, a *Theora*, *doxygen* és az *SDL*. Végül, de nem utolsósorban amennyiben *DV* (*Digital Video*) anyagot is szeretnénk használni, úgy megfelelő csomagokat is tanácsos telepíteni, ilyenek: *libraw1394*, a *libavc1394* és a *libiec61883*.

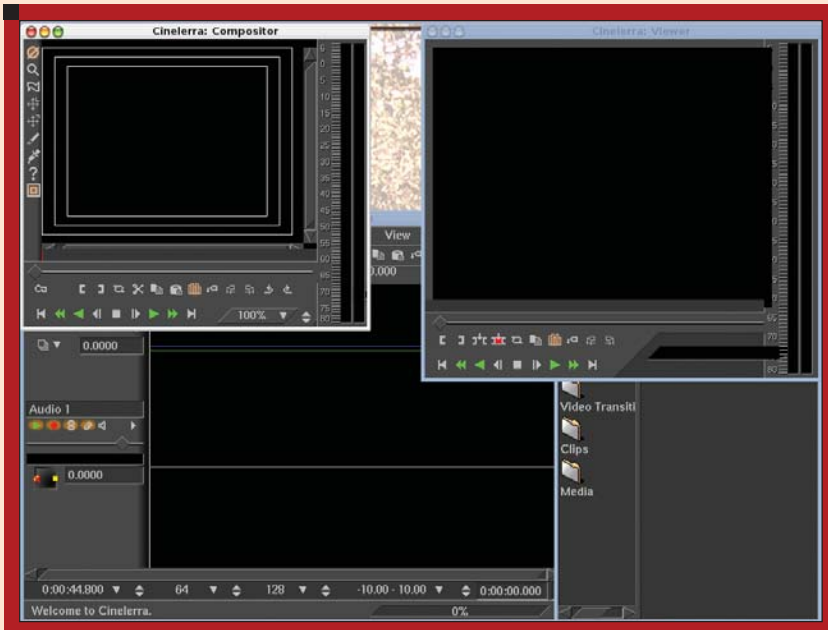


1. ábra Cinelerra az alap ablakokkal 1280x1024-es felbontás mellett

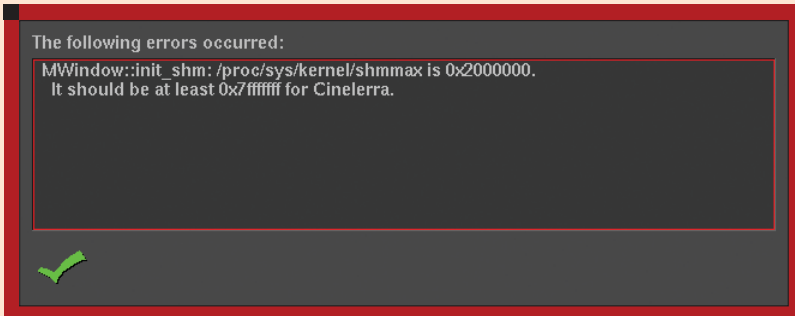
A program sikeres fordításához minimum fél gigabájt szükséges a merevlemezen (körülbelül 30 megabájt a letöltendő állomány, kicsomagolva kb. 150 megabájt, lefordítás után már 460 megabájtot foglal a könyvtár), viszont telepítés után törölhetjük a forráskód könyvtárát.
A telepítés a függőségek feloldása után a megszokott `configure, make, make install` trióval történik, de amennyiben *AMD 64*-re fordítjuk, úgy a `configure` futtatása után pár *Makefile* a használt disztribúciótól függően korrigálásra szorulhat. Nálam például a mjpegtools könyvtárban található összes *Makefile*-ban hibás architektúra bejegyzés került. *K8* került a szükséges x86-64 helyére. Ezt az alábbi parancssorral korrigálhatjuk gyorsan, melyet a *Cinelerra* forrás könyvtárából adunk ki:

```
for i in `find . -name Makefile
➥-path './mjpegtools*'`; do
➥sed -i -e 's/-march=k8/-
➥march=x86-64/' -e 's/-mcpu=
➥k8/-mcpu=x86-64/' $i; done
```

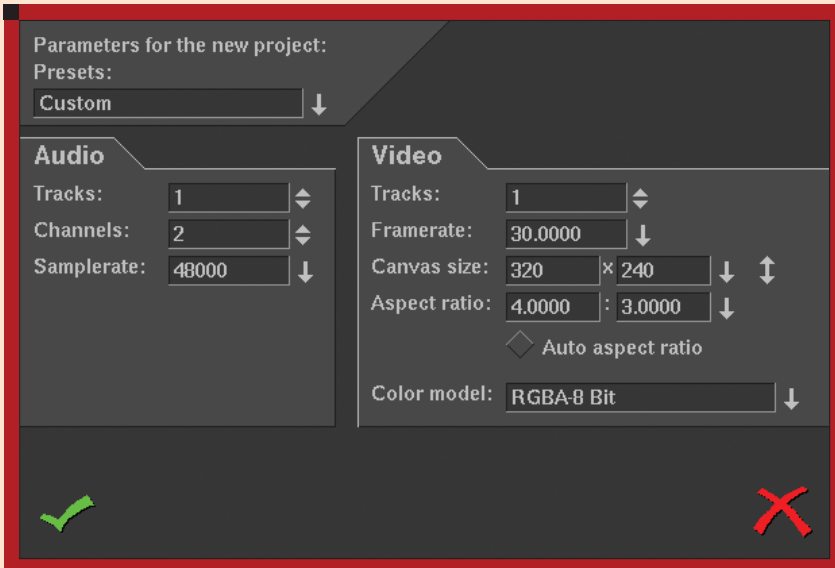
A program futtatásához minimálisan 1280x1024-es felbontást (*1. ábra*) javasolnék, mert 1024x768-as felbontás



2. ábra Cinelerra az alap ablakokkal 1024x768-as felbontás mellett



3. ábra Kicsi az shmmax értéke



4. ábra Új projekt létrehozása

(*2. ábra*) mellett túl sok idő megy el az ablakok váltogatásával, viszont a nagyobbik felbontás már könnyen átlátható munkaterületet ad. Indításkor belefuthatunk egy üzenetbe (*3. ábra*), mely az `shmmax` nem megfelelő értékére panaszkodik. Amennyiben ez a helyzet, úgy rendszergazdaként adjuk ki a program indítása előtt a

```
echo 2147483647 >
➥/proc/sys/kernel/shmmax
```

parancsot. Ez a maximális osztott memória értékét állítja be. (Esetleg érdemes a gép indítási szkriptjeibe is elhelyezni.)
A *Cinelerra* négy ablakot nyit meg induláskor, ezek:

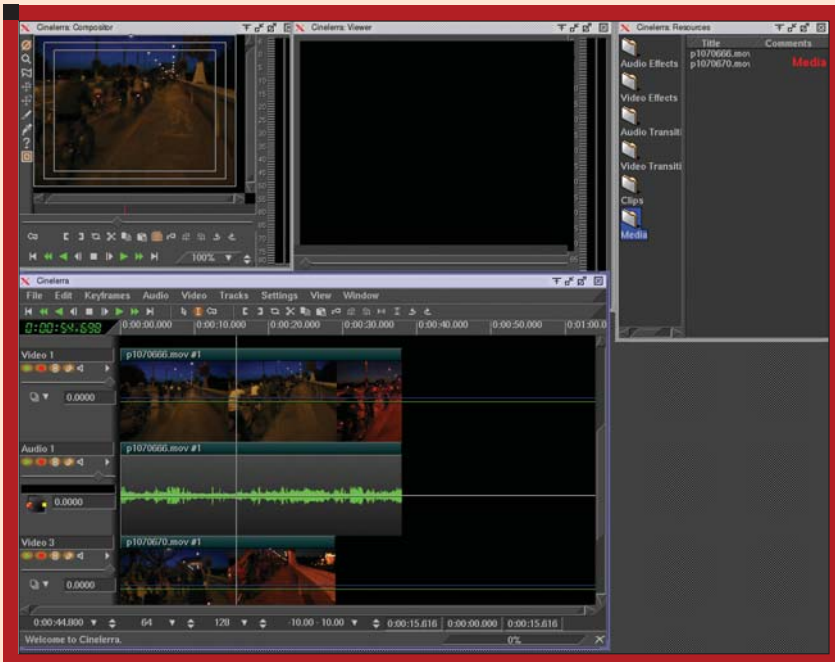
- ***Compositor*** – ebben látszik a videónk előnézeti képe.
- ***Viewer*** – ebben menet közben nézhetünk videót úgy, hogy a projektbe nem nyúlunk bele.
- ***Resources*** – itt találjuk meg a hangi és képi effekteket, úsztatásokat, illetve az eddig betöltött videókat (ez utóbbiak a ***Media ikon*** mögött lesznek)
- ***Program*** – ez úgymond a főablak, itt láthatóak a hang és videósávok, illetve a menüsor is.
- Ha ***véletlenül*** bármelyiket is ***bezártuk***, akkor a ***Program*** ablak ***Window menüjéből*** újra előhozhatjuk őket. Értelemszerűen a ***Program*** ablak ***bezárásával kilépünk*** a programból.

Első indítás után a főablak ***Settings -> Preferences*** menüpontját gondosan tanulmányozzuk át és állítsuk be a rendszerünkhöz (hangrendszer, input, output, stb.), azonban az esetek nagy részében a grafikus felület (***X11-XV***, ***X11-OpenGL***, stb.) és a hangrendszer beállítása (***OSS***, ***ALSA***, stb.) után a többi már gyári beállításokkal is megy.

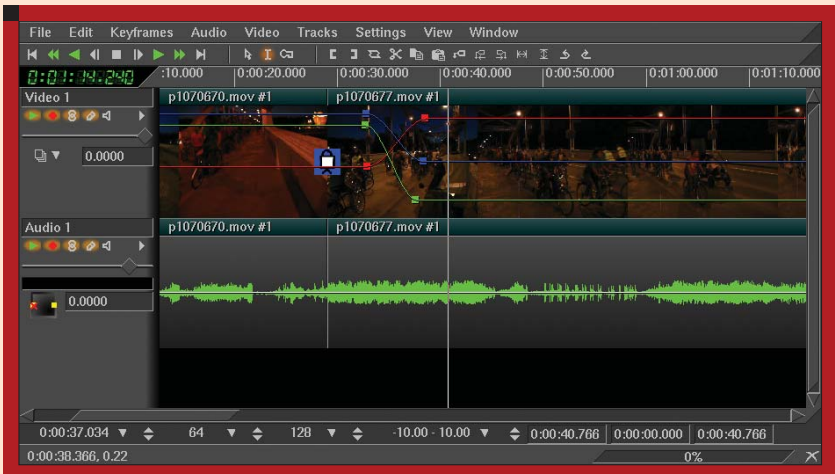
Médiaállományok, sorakozó!

Hozzunk létre egy új projektet (*4. ábra*). A létrehozáskor sok adatot meg kell adnunk: felbontás, hangcsatornák száma, színvilág, stb. Természetesen a munkát könnyítendő, található pár előre definiált beállítás is.
Noha nem kell pilótavizsga a program kezeléséhez, mégis azt javasolnám, hogy ha új párbeszéd ablakot látunk, nézegessük végig a legördülő menük tartalmát, hiszen nem tudni, később mikor lehet rá szükség.

A legelső ilyen érdekesség a fájl betöltése, megadhatjuk ugyanis, hogy hogyan szeretnénk betölteni a videót. A betöltés történhet az aktuális sáv kurzorpozíciójára, új sávként, az aktuális sávhoz hozzáfűzésre, vagy például a korábban említett Média tárba (ekkor nem jelenik meg semmi a főablakban). Sikeres betöltés után a főablakban megjelenik a Videósávban a videóból pár állókép, illetve a Hangsávban a hang görbéje (*5. ábra*).
A lapozás történhet vízszintesen (idő) vagy függőlegesen (sávok), míg a idősáv beosztási sűrűségét (zoomolását) a lenti státusz sorban vagy az egér görgője és a ***SHIFT*** lenyomása segítségével állíthatjuk.
Amennyiben a Médiatárba töltöttük be a filmrészleteket, úgy fogd-és-vidd módszerrel illeszthetjük be a készülő filmünkbe. Ez alapértelmezetten a sáv végére illeszti az anyagot. Amennyiben kevés a sáv, használjuk a hozzáadást az ***Audio*** és a ***Video*** menükben. Fontos megjegyezni, hogy a sávok nevei alatt van egy piros kör. Ha be van kapcsolva, akkor az adott sáv írható is, ha ki van kapcsolva, akkor csak olvasható, semmilyen módosítás nem lesz rajta. Ezzel például rögzíthető egy már kész sáv.



5. ábra Videó, rögtön betöltés után



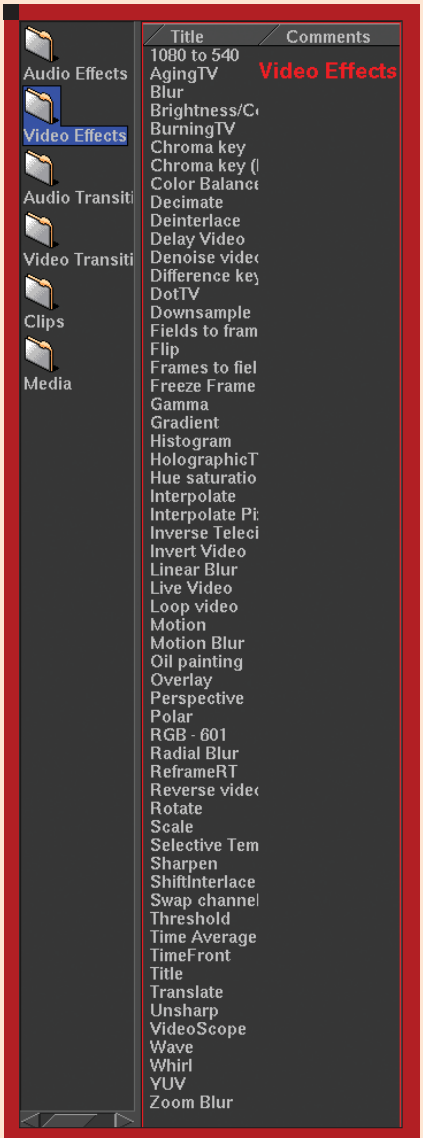
6. ábra Piros, zöld és kék görbék kuszasága

Varázsolunk... (mozgókép)

Először is állítsuk be a csatorna fényerejét. Erre a vízszintes csúszka szolgál a baloldalon. Alatta egy rétegeket ábrázoló legördülő menüben állítható be, hogy az aktuális videósáv hogy viszonyuljon a többihez. Hasonló már ismerős lehet a ***GIMP***-ből vagy más rétegeket is használó grafikai programból.
Alapesetben a videósáv közepén húzódik egy vízszintes vonal. Ha ügyesek vagyunk, ezt széthúzhatjuk három különböző színűre – pirosra, kékre és zöldre. Le kell hűtsek mindenkit, ezek nem a színvilág beállítására szolgálnak, hanem különböző geometriai átalakításokra. A kék a ***zoom***-ért felel (fontos: ez digitális ***zoom***, nagyítás esetén pixelesedik a kép), a piros a vízszintes eltolást kezeli, a zöld pedig a függőlegest (*6. ábra*). Na de hogy lesz ebből látványos film? A vízszintes vonalak megtörhetőek úgy, hogy a bal egérgombbal a kívánt helyeken tudunk új vezérlőpontokat létrehozni és azokat mozgathatjuk a kívánt hatás elérése érdekében, így például a kék görbét (egyenes, amelyre tettünk két vezérlőpontot) állítgatva úgy fog kinézni a videó, mintha a kamerás zoomolt volna. Persze ettől még nem lesz ***hollywoodi*** produkciónk, hiszen két – egymás mögé fűzött – ***snitt*** között hirtelen átmenetet kapunk, semmi úsztatás, semmi szép váltás. Ehhez a ***Resources*** ablakból ***vonszoljunk*** át egy elemet a ***Video Transitions*** gyűjteményből a ***két videó*** részlet ***illesztési*** pontjához. Ha még mindig nem vagyunk elégedettek, mert például szeretnénk régi, celluloid filmre készült mozi hatását kelteni, akkor ehhez látogassuk meg a ***Video Effects*** gyűjteményt (*7. ábra*) és onnan húzzuk rá ***AgingTV***-t a kívánt sávra. A sávra beállított effekt(ek) mindig az adott sáv alján látható(ak) és amennyiben paraméterezhető(ek), úgy azt a nagyító ikonra kattintva tudjuk megtenni. Ideiglenes lekapcsoláshoz a mellette lévő kikapcsoló gombot nyomjuk meg. Ez utóbbi azért jó, mert valósidejű előnézet (***Compositor*** ablak) lejátszás esetén még egy 3000-es ***AMD64***-en is szaggathat. Ha meggondoltuk magunkat és mégse kell a kiválasztott effekt, jobb klikk az adott csíkon, majd ***Detach*** menüponttal végleg eltávolítjuk a videóról.

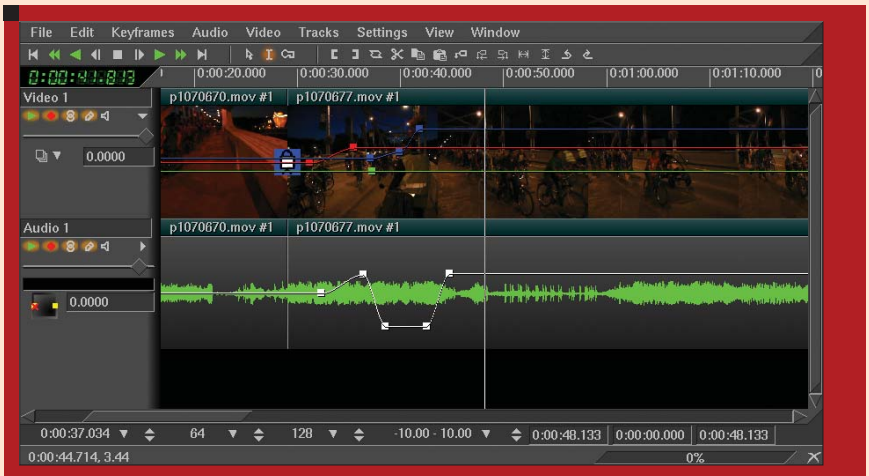
Varázsolunk... (hang)

A videó után a hang már kicsit egyszerűbb lesz. A sávok kezelése szinte ugyanaz, azzal a különbséggel, hogy itt



7. ábra A rendelkezésre álló képi effektek listája

8. ábra Hangszórók száma és helyzete



9. ábra A hang és a kép „vonalai" végleges állapotban

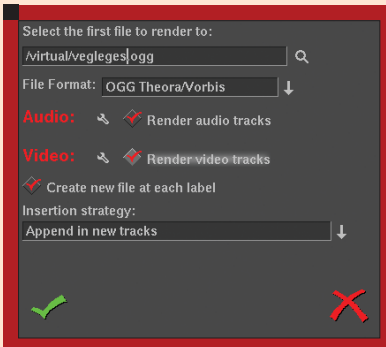
a vízszintes csúszka a hangerőt állítja. Ami a videónál nem volt: egy négyzet, benne valahány darab kis sárga négyzettel és egy piros kereszttel. A sárga négyzetek számából látható, hogy hány csatornás hanggal dolgozunk, míg a piros kereszt azt mutatja meg, hogy a hangszórók képest hova kerül a hang. (két hangszóró használata esetén ez egyszerű ***balance*** állításnak felel meg) A ***Cinelerra*** összesen 16 monó csatornát képes kezelni, ez azt is jelenti, a sztereó hang hozzáadása esetén két külön monó sávként jelenik meg. A hangszórók helyzete és száma a ***Settings -> Format*** menüben (*8. ábra*) állítható. (Fontos megjegyezni, hogy a hangszórók helyzetének állítása csak akkor ér csak valamit is, ha a valóságban is úgy vannak/lesznek elrendezve.)

Vízszintes vonalból itt csak egy fehér található, ezzel állítható a hangerő. Az effektek és az átmenetek (***transitions***) ugyanúgy használhatóak, mint videó esetén, de nyilván kevesebb van belőlük.

Mentés

Mielőtt végrehajtanánk a kijelölt műveleteket (*9. ábra*) és lementenénk a végleges videót, ***mentsük*** el a projekt állományt mindenképp, hiszen áramszünet vagy bármi más jöhet és a már eddig megcsinált rész elvész.

A kimentett projektállomány ***XML*** formátumú, tehát ha nem is túl könnyen, de akár ***vi***-ból is szerkeszthető kismértékben. Ez például akkor jön jól, ha másik gépen szeretnénk megcsinálni a végleges videót, mint amin szerkesztettük. Ilyen esetben elég a projekt állományban a fájlok elérési útvonalait kicserélni és a másik gépre is átvinni a forrás médiaállományokat.
A végleges videó elmentésekor (*10. ábra*) az elérhető formátumok listája nagyban függ a program fordításakor jelenlevő program-



10. ábra A vágások végrehajtása (renderelés)

könyvtáraktól. Azon sem kell meglepődnünk, ha esetleg bizonyos konténer és kodek kombináció esetén minden üzenet nélkül bezáródik a program. (Nekem eddig *Debian Sarge* alatt az *Ogg*-ba mentés működött hiba nélkül, a többivel előfordultak problémák.) Amennyiben ezt történne, használjunk *Ogg*-ot és plusz egy lépésben alakítsuk át *mencoder*-rel a kívánt formátumra. A mentés a videó hosszától, az effektek számától és bonyolultságától, illetve a gépünk sebességétől függően akár több óra is lehet.

Egy példa az átalakításra:

```
mencoder filmunk.ogg -o
➥filmunk.avi -oac lavc -ovc
➥lavc -lavcopts
acodec=mp3:abitrate=128:vcodec=
➥mpeg4:vbitrate=1000
```

Ezzel az *ogg* fájlból *avi* készül 1 megabites videó és 128 kbites hang sávszélesség mellett. (*Mencoderről* részletes leírást a *Linuxvilág 2006. májusi* számában talál az Olvasó, de a *Google* is rengeteg találatot ad a témában.)

...és ennyi!

Azt ugye belátja a kedves Olvasó, hogy a *Cinelerra* még elég messze áll a tökéletes programtól, de azért már nagyon jól használható eszközt kap az ember. A teljesítmény és az ár viszonyát pedig már végképp felesleges említenünk, hiszen szinte ingyen (minimális munkát kell csak befektetni, hogy működjön) kapunk egy elég profinak mondható videószerkesztő programot kedvenc operációs rendszerünk alá.

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

Cinelerra:
➲ http://heroinewarrior.com/cinelerra.php3

YASM:
➲ http://www.tortall.net/projects/yasm/

Libogg, Libtheora:
➲ http://www.xiph.org

Libraw1394, LibIEC61883:
➲ http://www.linux1394.org

Libavc1394:
➲ http://www.sourceforge.net/projects/libavc1394/

Heavy Metal F.A.K.K.2

Ki ne szeretne egy színvonalas ügyességi kalandjátékkal eltölteni néhány órát? Valószínűleg sokakat vonz a műfaj varázsa, viszont az éllovas Tomb Raider sorozat sajnos nem fut Linuxon. Ezért egy másik megvalósítást ajánlok az olvasók figyelmébe.

A hogy a rövid bevezetőben említettem, *Lara Croft* ki nem állhatja a *Linuxot*. Nem tudni, miért haragszik ránk, ennek ellenére a közismert virtuális hölgy nagyon népszerű: szinte nincs olyan játékos, aki ne ismerné a copfos, fegyveres „*Lara*-baba" kalandjait. Minket pedig megesz a sárga irigység, hiszen a műfaj koronázott királyának csak némely epizódja hívható életre Szabad rendszeren, az is csupán *Win32 wrapperek* segítségével. Emiatt nagy megkönnyebbülést jelenthet a kategória megszállottjainak, hogy a *Heavy Metal* című képregény főhőse már sokkal jobban szereti a pingvinnel fémjelzett környezetet.

1. ábra A játék linuxos telepítője

Julie világa

Lássuk tehát, miről is van szó! A *Ritual Entertainment* egy külső nézetű, ügyességi kalandjátékot publikált *2000*-ben, melynek történeti szála lineáris. E linearitás miatt a „főszereplő" *Julie Strain* sajnos nem képes arra, hogy tömegeket csábítson többszöri végigjátszásra, ám egyszer mindenkinek érdemes megtapasztalni a *Heavy Metal F.A.K.K.2* szürrealista környezetét! A megjelenítést és a fizikai modellezést a *Quake3* motorjára bízta a fejlesztő, ami még ma is jó ajánlólevél: az *id Software* által megírt mag itt már a végletekig lett optimalizálva, ennél szebb látványt valószínűleg értelmetlen lenne elvárni a nagy karriert befutott programkódtól. Szerencsére a karakterek és modellek mozgáskultúrája egyaránt részletgazdag, így ha a főhős egyedi irányítását és a kiváló zenei aláfestést is megemlítem, akkor talán már látszik, mire is akarok célozni: profi munkáról van szó. Viszont rögtön az elején azt is le kell szögezzem, hogy a kalandjáték linuxos verziója a néhai *Loki Entertainment* utolsó befejezett átirata (*2001-ből*), így a portolás hibáit javító foltok nem léteznek, a natív bináris hibáival „együtt kell élni". A *bugok* a grafikai és a hangleképezési módot egyaránt érintik, mivel a játék teljes képernyős módban csupán *640x480* képfelbontásban használható, valamint a hangeffektek is hibáznak néhanapján. Kétségtelenül fontos hiányosságok ezek, de azért a csodaszép virtuális világ még így is önfeledt kikapcsolódásra csábít...

Honnan, hogyan...

Mivel a *Loki* sajnos már nem létezik, így a linuxos verziót nem könnyű beszerezni. Hazai forrás nincs, de kitartóan keresgélve a világhálón még található néhány cég, akik (korlátozott számban) fogadnak rendeléseket. Leginkább a nagynevű ➲ http://www.tuxgames.com oldalon érdemes rendelni, ahol közel *30 Dollárért* hozzá lehet jutni a legális példányhoz. A javítatlan hiányosságok kissé „zsákbamacska" jelleget kölcsönöznek a tranzakcióknak, de aki szereti az igényes *3D* kalandjátékokat, bizonyosan jó vásárt csinál. Ha megérkezett címünkre a lemez, fűzzük be a rendszerünkbe, majd a gyökerében található *setup.sh* szkriptet indítsuk el *root*-ként! A telepítő felajánlja a *Loki*-féle *Installer*

2. ábra Csodaszép virtuális világ

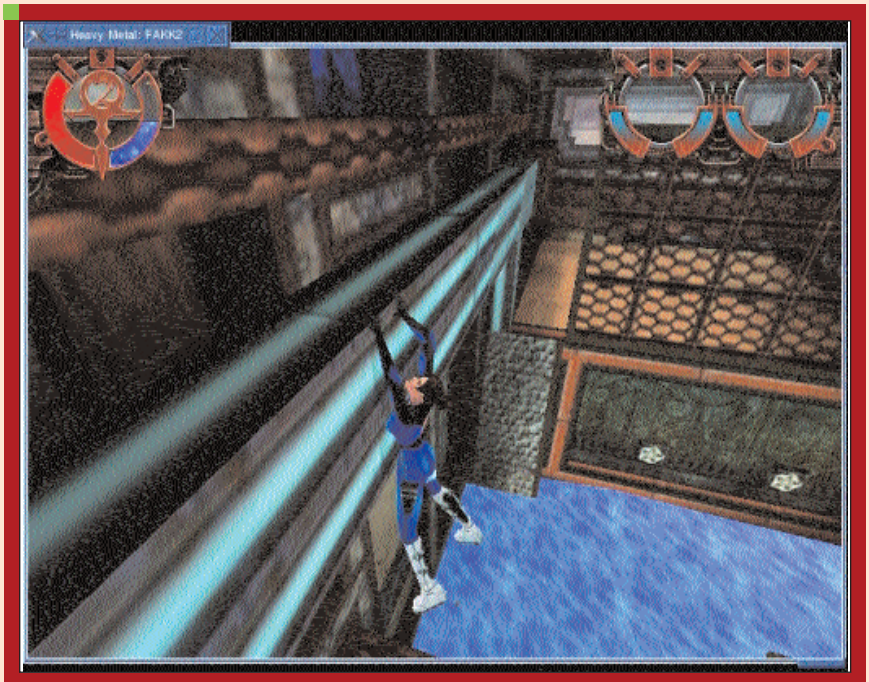
3. ábra Egy barátunk érdekes járműve...

és ***Updater*** komponensek felmásolását, majd a játék telepítéséhez asszisztál. A ***Heavy Metal F.A.K.K.2*** állományai alapértelmezés szerint az ***/usr/local/games*** útra kerülnek, indító kötéssel az ***/usr/local/bin*** mappában. A teljes telepítés nagyjából ***380MByte*** helyet foglal el a merevlemezen, viszont ezzel a módszerrel installálva később már nem szükséges a ***CD***-lemez megléte (a játék idejére sem). A program kérésünkre akár a ***KDE*** és a ***GNOME*** interfészek menüibe is beköltözik – ezen felül egy felhasználóként terminálra gépelt `fakk2` paranccsal hívható életre. A kompromisszummentes használat feltételét egy ***800MHz*** órajelű ***x86*** processzor, és egy ***32MByte*** memóriával szerelt ***3D*** grafikus hardver jelenti (betöltött ***GLX*** illetve ***DRI*** meghajtóval).

Nem kerülhetem el azt a tényt, miszerint tapasztalataim alapján a ***GLX*** kapoccsal rendelkező ***nVidia*** termékek ebben az esetben is előnyt élveznek az ***ATi*** hardvereivel szemben. A képi világ egyébként szinte fokozatmentesen hangolható: ezt az „apróságot” nem sok program mondhatja el magáról. A menükből nem beállítható paramétereket a ***/home/$/.loki/fakk2/config.cfg*** állományban lehet rögzíteni, igen átlátható és kézenfekvő módon.

Kell a rutin...

A játék teljes menetéről nem érdemes túl sok szót ejtenem, mivel ebben a műfajban fokozottan igaz az öreg mondás: „a puding próbája az evés”. Kérésünkre ***Julie*** kúszik, mászik, ugrik, dialógusokba bocsátkozik, harcol, vagy éppen védekezik. Több fegyver is szolgálatára van, kezdve a parittyától egészen az időzített bombáig, miközben a körülötte „lézengő” lényeknek és a terep tárgyainak némelyikét is képes a kezébe venni. A hölgy irányítása kissé szokatlan: legfőképpen parittyával, vagy karddal a kézben érdemes begyakorolni a mozdulatokat, de már egy „kötélről kötélre” ugrás sem egyszerű művelet...
Mivel az oktató részt ebben a kategóriában nem ajánlott megkerülni, a tanulással töltendő negyed óra megspórolhatatlan. A minimális rutin komolyan elvárt tulajdonság, enélkül a hosszú és élvezetes játékidő percekre redukálódik: nem ritkán öt-hat ellenfél támad egyszerre, vagy éppen dobozokat tologatva kell lépcsőt építeni magunk elé a továbbjutásért (a gyakorlott ***Tomb Raider*** fanatikusok azért pillanatok alatt kiismerhetik magukat a ***Heavy Metal*** világában, mint ahogyan az apró „rejtvények” sem hatnak idegenként). Azt azonban meg kell még jegyeznem, miszerint a két program abban mindenképpen eltér, hogy a ***Relic*** mesterművében a harc (és ennek technikája) a játékmenetben jobban dominál.
A ***Quake3*** motorjához visszakanyarodnék egy gondolat erejéig: nagyon érdekes látvány, ahogyan a külső nézetet a fejlesztők „megbolondították” egy perspektivikus leképezéssel (mindez szépen nyomon követhető a mellékelt képeken). A programról kiegészítő információkat, galériákat, videókat, képregényeket, ***Julie*** (akár erotikus



4. ábra „Akrobatikus" továbbjutás...

jellegű) fotóit a fejlesztő lapján, a ➲ http://www.ritual.com/FAKK2 címen érdemes keresniük a mélyebb érdeklődésű megszállottaknak... Sajnos a ***Tomb Raide***r kiesik a látókörünkből, így be kell érnünk ***Julie*** kalandjaival. Kissé más a stílus, kissé más a felépítés, de a számunkra érdekes műremeknek semmilyen szégyenkezni valója nincs a vezető sorozattal szemben. A ***Heavy Metal F.A.K.K.2*** ügyes programozói munka és ötletes kivitelezés gyümölcseként nem hibába került be ***Scott Draeker*** néhai csapatának érdeklődési körébe... Bravó ***Ritual***, köszönjük ***Loki!***

Kovács Zsolt (kovi@linuxforum.hu)

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

5. ábra A közelharc végig dominál