Hírek

Linuxos okostelefon Blackberry kinézettel

A *Grundig* legújabb *Linux*os okostelefonjával – a *B700*-al – elsősorban a felső kategóriát célozza meg. Tudását tekintve méltó ellenfele lehet a *Blackberry*-nek, hiszen négysávos *EDGE* kompatibilis *GSM* modullal rendelkezik, így a Földön bárhol használható. 2,4 hüvelykes (320x240 képpontos) kijelzőjén tekinthető meg a beépített 2 megapixeles kamera képe is. Alapesetben 100 megabájtnyi tárhellyel gazdálkodhat a felhasználó, de maximum 1 gigabájtos *MicroSD* kártyákkal bővíthető. Az eszköz amellett, hogy *Linux*os, támogatja a *J2ME* alkalmazások futtatását is. A *Blackberry*-k nagy előnye, hogy a szerver jelzi, ha jött e-mail és nem nekünk lekérdezni. (Olyasmi technológia, mint az *SMS* a mobiltelefonoknál. – szerk.) A *Grundig* megemlíti ugyan a *POP3 push* lehetőséget is, de hogy ez mennyire hasonlít a *Blackberry* megoldására, arra már nem tér ki a híradás.

➲ http://www.linuxdevices.com/news/NS2918775846.html

A pingvin, ami mindent lát

A *Nuvation* bemutatta extra pici (79 x 43 x 47 mm) kameráját, melyet *Linux* vezérel. A megszokott analóg kimeneten kívül *Ethernet* hálózaton is képes továbbítani a kép anyagot. A 300 MHz-es *ARM9*-es processzor lehetővé teszi akár azt is, hogy a maximálisan 720x480-as felbontású (akár 30 képkocka másodpercenként) képeket akár magában a kamerában elemezzük. A kamera működés közben kevesebb, mint 5 wattot fogyaszt és adapter helyett táplálhatjuk *Power-over-Ethernet* eszközzel is. A támogatott kodekek között megtalálható a *H.264, H.263, MPEG-4* valamint a *motion-JPEG (MJPEG)* is. Minthogy ez még csak bemutató darab volt, így az ára ismeretlen.

➲ http://www.linuxdevices.com/news/NS7019659766.html

3G-s mobiltelefon

Hamarosan kapható az olasz *Enteos* cég két új *Linux*os okostelefonja – *iGo 730* és *LinAP* – , melyekben *MontaVista* rendszerek üzemelnek. A telefonokat előreláthatóan mobil szolgáltatók is forgalmazzák majd.

➲ http://www.linuxdevices.com/news/NS9710475530.html

Linuxos okostelefon

A taivani *Accton Technology* piacra dobta legújabb multimédiás okostelefonját, amely azontúl, hogy négysávos, támogatja a *wifi*-t is. Ennek nagy előnye, hogy ahol rendelkezésre áll, ott az olcsó *wifi*-t használja, illetve *GSM* és a *wifi* hálózatok között akár beszélgetés közben is tud váltani. *MicroSD* kártyákkal a háttértára 2 gigabájtig bővíthető. A telefon érintő kijelzővel kerül a boltokba, melyet elsősorban a kínai és japán felhasználók fognak értékelni. Adatátvitelről *USB* és *Bluetooth* gondoskodik. Az eszköz *J2ME* alkalmazásokat is tud futtatni, webböngészőként pedig az *Operát* használhatjuk.

➲ http://www.linuxdevices.com/news/NS9521201305.html

Vékonykliens megoldás a Novelltől is

A *Novell* bemutatta a *SUSE Linux Enterprise Thin Client* termékét, amely a *SUSE Linux Enterprise Desktop* rendszerből és egy lemezképkészítő eszközből áll. A kaliforniai *Windsor Unified School District* (windsori egyesült iskolakörzet) 2000 számítógépet állított át vékonykliens-architektúrára és *SUSE Linux Enterprise Desktop* rendszerre, így 75 százalékkal csökkentette hardverköltségeit, és majdnem 95 százalékkal szoftverköltségeit.

Apró Linux rendszer pehelysúlyú Java támogatással

A kanadai ***Techsol*** bemutatta az eddigi talán legkisebb fogyasztású ***Linux***ot futtató rendszerét – a ***Gateway Express***-t –, mely alig 2 wattot igényel. Az eszköz nem nagyobb, mint egy ***DVD*** meghajtó, ennek ellenére helyet kapott benne ***RS232*** és ***RS485***-ös soros port, ***USB*** host és kliens port, valamint egy 10 megabites hálózati csatoló. Az eszközön futó ***Medallion Linux*** 2 megabájt ***NOR***, 32 megabájt ***NAND*** tárhellyel valamint 32 megabájt memóriával gazdálkodhat és képest ***Java*** alkalmazásokat is futtatni. 300 dollár alatt van az ára ezer darabos rendelés felett.

➲ http://www.linuxdevices.com/news/NS6858563145.html

Univerzális set-top box

A ***Sigma Designs*** és a japán ***Celrun*** bemutatta legújabb set-top boxát a londoni ***IPTV*** kiállításon. A ***Celrun 900H*** azontúl, hogy rendelkezik a szokványosnak mondható csatlakozókkal (analóg antenna, ethernet, ***HDMI***, kompnens, kompozit, S-videó, optikai és ***USB***), rendelkezik a nemrég elfogadott ***UWB*** (***Wimedia Ultra-wideband***) csatolóval is. Az eszköz minden ismertebb formátumot (***H.264, WMV9, MPEG2***, stb.) támogat. Az adások rögzítése a beépített 80 gigabájtos merevlemezre történik. Amennyiben ***wifi***t is szeretnénk használni, úgy arra is van mód, de azt ***USB***-n kell csatlakoztatni.

➲ http://www.linuxdevices.com/news/NS5189647857.html

Nagyképű linuxos médialejátszó

Alig jelent meg az ***Archos 604***, máris megjelent nagyobbik testvére a ***704***, mely 7 hüvelykes (800x480 képpont felbontású) kijelzővel érkezik. Az 550 dolláros árban viszonylag magas, de a készülékben 80 gigabájtos merevlemez és 802.11b/g ***wifi*** is helyet kapott. A készülék alapfelszereltségben ***Mpeg4v2***-t, ***wmv***-t, ***mp3***-at, ***wav***-ot és ***wma***-t tud kezelni, azonban 20 dolláros egységáron további kodekek – ***h.264, mpeg2, ac3, aac3*** – kaphatóak hozzá. Hogy igazán használható legyen, érdemes beszerezni a 100 dolláros dokkolót is (ezzel felvenni is tud, nemcsak lejátszani), illetve a 70 dolláros utazó szettet, mellyel autóban is használhatóvá válik. Üzemidejét tekintve: egy töltéssel 25 óra zenelejátszást vagy 5 és fél óra filmnézést biztosít a beépített akkumulátor.

➲ http://www.linuxdevices.com/news/NS6542038482.html

Németország nemet mondott a FAT szabadalmi igényre

A német szabadalmi ügyekben illetékes bíróság elutasította a ***Microsoft FAT*** fájlrendszerekkel kapcsolatos szabadalmi igényét (hosszú és rövid fájlnevek névtere – ***„common namespace for long and short file names”***), mondván: nincs mögötte igazi felfedezés. A ***Microsoft*** termékenként 25 centet vagy egyszeri 250 ezer dollárt követelne a licenszért ***Európában*** – kivéve most már ***Németország***ot.

➲ http://www.linuxdevices.com/news/NS2298023810.html

Minden napra egy .deb

Érdekes projekt bukkant fel az ***Interneten***. Célja: minden napra ajánlani egy ***Debian*** alá elérhető csomagot, mely nem túl közismert. Hasznos kezdeményezés, hiszen ki akkora varázsló, hogy a több, mint 15 ezer csomagot ismeri?

➲ http://debaday.debian.net

Vékonykliensek a CeBIT-en

A holland székhelyű ***Acropolis Automation*** négy új modellel jelentkezett az ***Athena*** sorozatból. Az új ***T3000***-es modellek 1 gigahertzes ***Via Eden*** processzort tartalmaznak, melyen ***Linux*** és ***Windows (CE*** vagy ***XP embedded)*** futtatható. A ***T3100T*** és a ***T3090T*** különálló eszköz, míg a ***T3117F*** és a ***T3119F LCD***-be építve kapható. A 17 hüvelykes 1280x1020 képpontos, míg a 19 hüvelykes 1600x1200 képpontos felbontást ad. A maximális memória méret az ***LCD***-s eszközök esetén fél gigabájt, egyébként pedig 1 gigabájt. Minthogy az eszköz merevlemez nélkül érkezik, így indíthatjuk hálózatról (***PXE***) vagy két darab ***DiskOnModule (DOM)*** és egy ***CF*** kártyával bővíthetjük. Az árról és a piaci bevezetésről még nem nyilatkozott a cég.

➲ http://www.linuxdevices.com/news/NS2016451866.html

Open Transport Tycoon Deluxe

Az ***Open Transport Tycoon Deluxe*** már elérhető ***Windows***os okostelefonokra és ***PDA***-kra, valamint ***Palm OS 5*** kompatibilis eszközökre. A portolt verzió letölthető az alábbi linkről, azonban pár fájl még szükséges hozzá: ***trg*.grf, sample.cat, gm_tt*.gm***

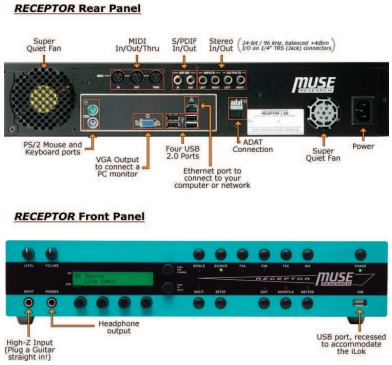
➲ http://www.esoftinteractive.com/openttd.html

Pingvin az anyósülésen

Linux alapú navigációs szoftverrel ellátott okostelefon került a polcokra ***Kínában***. A ***Tranzda PM328*** a szokványos funkciókon kívül tartalmaz egy ***GPS*** modult is, amelyet a svájci ***U-blox*** gyártott és egyedülálló érzékenységgel bír.

➲ http://www.linuxdevices.com/news/NS7266092854.html

Wine + Muse Research = ???



A ***Muse Research*** – felismerve a hiányt – megalkotta a ***Receptor***-t, mely nem más, mint egy ***WINE*** emulátorral felvértezett ***Linux***os számítógép, amely zenei alkalmazások futtatására van kihegyezve. A ***2U*** magas eszköz ***VST*** formátumú hangeffektek futtatására és virtuális hangszerek leképzésére képes. Noha a zenészek már eddig is használhatták a fenti technikát, élő előadásokon asztali számítógépek használata nem igazán bizonyult célszerűnek, a laptopok pedig gyengék voltak. Az eszköz egy ***MSI*** alaplapot tartalmaz, melyen 2 gigabájtig bővíthető a memória. A 2.2 gigahertzes processzor egy ***AMD Turion***, amely ***64 bites***. Igény szerint 160, 400 vagy 750 gigabájtos merevlemezzel kérhető. Zenészeknek hasznos információ, hogy rendelkezik szabványos ***MIDI*** ki és bemenetekkel, valamint elektromos gitár erősítő nélkül is csatlakoztatható. Operációs rendszere ***Fedora***. Ez mind szép és jó, azonban az árcímke 2000 és 2500 dollár közötti aszerint, hogy milyen kiegészítőkkel kérjük.

➲ http://www.linuxdevices.com/articles/AT7270104304.html

Távoli asztal – másképp

A ***Matrox***tól hamarosan rendelhető a távoli grafikus csatoló – ***Extio F1400*** –, mellyel a számítógép és a felhasználó akár 250 méterre is lehet egymástól. A felhasználó munkahelyén lévő passzív hűtésű eszköz optikai kábelen csatlakozik a távoli számítógéphez. Az eszközre akár négy ***DVI*** vagy analóg bemenetű monitort is csatlakoztathatunk a 6 db ***USB 2.0***-ás port mellett.

A számítógépbe szükséges kártya a ***Matrox***tól külön rendelhető. A multimédiában dolgozók számára hamarosan elérhetővé válik az a két, illetve négy fejes verzió is, amelyek támogatják a ***Firewire***-t. Az eszközt jelenleg az ***ABB Automation Technologies*** teszteli.

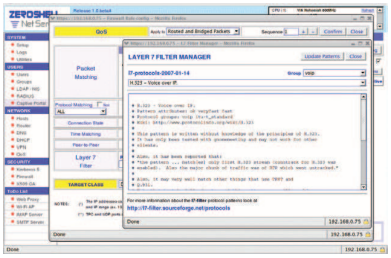
➲ http://www.linuxdevices.com/news/NS2448835068.html

Flash alapú merevlemezek

Super Talent bemutatta két legnagyobb kapacitású ***SATA*** csatolós flash alapú merevlemezét. A 3,5 hüvelykes 128 gigabájt kapacitású, míg a 2,5 hüvelykes 64 gigabájt kapacitású. A marketing igazgató a strapabírás és a kis energiaigény mellett azt is megjegyezte, hogy felére csökkent a gigabájtonkénti tárhely költség, míg az adatátviteli sebesség megduplázódott az egy éve bemutatott modellükhöz képest.

➲ http://www.i4u.com/article8215.html

Router disztribúció kifinomult QoS szabályzáshoz



Megjelent a ***Zeroshell 1.0.beta4*** verziója. A ***CD***-ről vagy ***CompactFlash*** kártyáról indítható disztribúció kifinomult ***QoS-t (Quality of Service)*** tesz lehetővé. ***Fulvio Ricciardi***, a ***Zeroshell*** vezető fejlesztője szerint az ***L7-filter*** alkalmazásával, valamint a konfigurációs felületbe történő integrálással most már szolgáltatásonként (például ***SIP, Skype***, stb.) is beállítható a prioritás és egy minimális, garantált sávszélesség.

➲ http://www.zeroshell.net/eng/
➲ http://www.linuxdevices.com/news/NS9446520379.html

Nagy váltás lesz...

A japán gazdasági, kereskedelmi és ipari minisztérium a közeljövőben nyílt forrású ***Linuxra*** szeretne váltani a jelenleg futó ***Windows 98*** és ***Me*** helyett. A döntés amiatt merült fel, hogy ezekre a szoftverekre megszűnt a támogatás, viszont frissítés esetén mintegy 400 ezer iskolai számítógépre kellene kifizetni az új verzió licenszét. Japán iskolákban már eddig is használtak nyílt forrású rendszereket.

➲ http://gyaku.jp/en/index.php?cmd=contentview&pid=000112

Az első Blu-Ray/HDDVD meghajtó PC-be

A ***Hitachi-LG*** elsőként dobott piacra optikai meghajtót (***GGW-H10N***), mely mindkét adathordozót képes olvasni, sőt a ***Blu-Ray*** lemezt írni is. A ***CeBIT***-en bemutatott eszköz 1200 dolláros áron kapható.

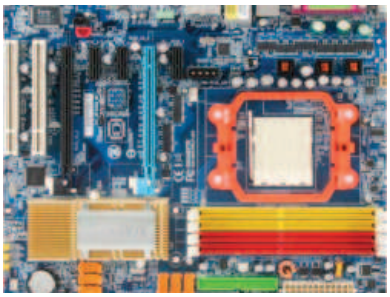
➲ http://www.eetimes.com/showArticle.jhtml?articleID=198000616

Van új a nap alatt: Ian Murdock

Ian Murdock, a ***Debian*** atyja, immár a ***Sun***-nál dolgozik. Kapva kapott a lehetőségen, hiszen mindig is kedvelte a céget, egészen 1992-től, amikor is legelőször találkozott a termékükkel a ***Purdue egyetem***en. Ennek a találkozásnak köszönhető az is, hogy megszakította üzleti tanulmányait és elkezdte helyette az informatikát.

➲ http://ianmurdock.com/2007/03/19/joining-sun/

Az első alaplap LinuxBIOS-al



Megjelent az első – asztali gépekbe szánt – alaplap, amely ***LinuxBIOS***-t tartalmaz a már jól megszokottakhoz – ***Phoenix, Award***, stb. – képest. Aki ilyen alaplapra vágyik, ***Gigabyte M57SLI-S4*** néven találja meg a boltokban. Az alaplap ***AM2***-es foglalatú 64 bites ***AMD*** processzorokat fogad. ***DDR2***-es memóriával 16 gigabájtig bővíthető. A ***BIOS***-ról: töredékére csökkent az idő, míg megjelenik a boot menedzser, azáltal, hogy minimális rendszerellenőrzést végez. További lehetőségek: hálózatos indítás, soros konzol, stb.

➲ http://news.softpedia.com/news/First-desktop-motherboard-supported-by-LinuxBIOS-47991.shtml

AJAX fejlesztés – még egyszerűbben

Az idei ***EclipseCon*** konferencián három ***Eclipse*** kiegészítőt mutattak be, mellyel még egyszerűbb az ***AJAX***-os alkalmazások fejlesztése. Ezek az alábbiak: ***Eclipse Rich Ajax Platform***, ***Eclipse Ajax Toolkit Framework***, valamint az ***Eclipse Dynamic Languages Toolkit Project***.

➲ http://news.zdnet.com/2100-3513_22-6165148.html

k3b 1.0

Megjelent népszerű ***KDE***-s ***CD/DVD*** író program, a ***k3b 1.0***-ás verziója. Az új verzió számos hasznos elemmel bővül, illetve a már meglévőek finomodtak. Ilyen például a DVD rippelő rész, vagy a hangsávszerkesztő.

➲ http://applications.linux.com/applications/07/03/05/181224.shtml?tid=13&tid=39

100 dolláros laptop Romániának is?

Nicholas Negroponte bemutatta a román kormánynak a 100 dolláros laptopot (***OLPC***). Amennyiben a parlament jóváhagyja, a kormány 150 millió dollár értékben vásárol ősszel a termékből. A kormány elsődleges célja, hogy reményt és esélyt adjon az anyagilag kevésbé tehetős gyerekek számára. Románia emellett tervezi egy saját és ingyenes ***wifi*** hálózat, a ***C*Free*** felállítását. A ***Linux***ot futtató számítógép kijelzője egy 7,5 hüvelykes 1200x900 képpontos ***LCD***, mely a képpontok méretét tekintve jobb, mint a kapható laptopok nagy része. A beépített kamera 640x480 képpontos felbontással bír másodpercenként 30 képkockás frissítés mellett. Processzora pedig egy ***AMD Geode*** GX-500@1.0W, mely 366 MHz-es sebességen üzemel.

➲ http://www.cfree.ro/
➲ http://wirelessisfun.com/2007/03/14/olpc-in-romania/

Debian unstable Live CD a Sidux

Megjelent a ***Sidux***, mely egy ***Debian Sid*** alapú ***LiveCD***. Elsődleges célja, bemutatni a ***Debian*** fejlődési irányvonalát, de azoknak is mankót adhat, akik nem boldogulnak egyes programok legfrissebb verzióinak forrásból történő fordításával.

➲ http://www.linux.com/article.pl?sid=07/02/28/1017239

Medve Zoltán
(e-medve@e-medve.hu)
2001-ben kezdett „Linuxolni", de már korábban is ismerkedett a szabad szoftverek világával. Ha éppen nem a gép előtt ül, akkor fotózgat, olvasgat vagy bicajozik.

Mi újság a rendszermag fejlesztése körül?

A kernel hamarosan támogatni fogja az eddiginél nagyobb méretű lefordított állományok futtatását is. Ehhez már csupán néhány apróbb problémát kell kiküszöbölni. ***Eric Biederman*** már jó ideje elkészített néhány ezt a célt szolgáló foltot, amelyek eddig ott lebegtek a felszín közelében a feldolgozandó feladatok között. Eddig annyi történt, hogy az ***IBM***-nél dolgozó ***Vivek Goyal*** tesztelte a benyújtott módosításokat. ***Vivek*** szerint elérkezett az idő arra, hogy az ő javaslataival illetve hibajavításaival kiegészítve a foltokat szélesebb a közönség elé tárják, illetve hogy bekerüljenek az ***Andrew Morton*** által karbantartott -mm fába. Ami a dolog fogadtatását illeti, számos fejlesztő a leghatározottabban örül ezeknek a módosításoknak, sőt sokan már dolgozni is kezdtek a jobbításukon. A munka egy része jelenleg a kód tisztázására irányul, némelyek azonban azzal a meglehetősen trükkös kölcsönhatásrendszerrel foglalkoznak, amely az említett kód és a ***swsusp*** nevű szoftveres felfüggesztési kód között fennáll. Mindent egybevetve úgy tűnik, hogy a foltok előbb vagy utóbb de biztosan elkészülnek, és bekerülhetnek a hivatalos kernelfába.

Úgy tűnik, sikerült megoldani az ***ar5k*** vezeték nélküli meghajtóval kapcsolatos valamennyi szerzői jogi problémát – legalábbis ezt állítja a ***Software Freedom Law Center (SFLC)***. A meghajtó, amit eredetileg az ***OpenBSD*** rendszerek számára fejlesztett ***Reyk Floeter*** egyesek szerint az ***Atheros*** lapkakészletet támogató meghajtó részleteit tartalmazta. És bár soha semmit nem sikerült rábizonyítani az ***ar5k*** meghajtó fejlesztőire, maga a skandalum elég volt ahhoz, hogy a fejlesztők

ódzkodjanak a kódnak a hivatalos ***Linux*** rendszermagba való befogadásától. Az ***SFLC*** most napvilágos látott véleménye szerint ugyanakkor a legfontosabb kérdéseket sikerült megnyugtatóan rendezni, így a kernelfejlesztők mostantól csak a kódolás hibáit, vagy magát a megközelítést kritizálhatják. Hogy mi is történjen a kóddal pontosan, arról egyelőre eléggé megoszlanak a vélemények. Egyesek szerint be kell venni a rendszermag kódjába úgy, ahogy van, vagy legfeljebb kisebb módosításokkal kiegészítve, míg mások szerint az egészet ki kell hajítani, és csupán a hardver kezelésére vonatkozó referenciaként szabad felhasználni egy teljesen új meghajtó megírásához. Akármi is lesz a végső döntés, az meglehetősen valószínűnek látszik, hogy nem is olyan sokára megjelenik a rendszermagban egy az ***Atheros*** lapkakészleteket támogató nyílt forrású kód, ami leválthatja az ***Atheros*** által fejlesztett, szabadon letölthető, ám zárt forrású ***MadWifi*** meghajtót.

Mostanában számos olyan folt jelenik meg, amelyek célja a kernel hibás, vagy karbantartó nélkül maradt részeinek eltávolítása, illetve az eltávolítás megfelelő türelmi idővel kiegészített ütemezése. ***Andrew Morton*** úgy döntött, hogy 2007 júniusában kikerül a fából a ***FUTEX_FD*** kód, mivel „javíthatatlan versenyhelyzeteket” tartalmaz. Egész pontosan a ***DevFS*** alrendszerrel van már egészen régi időkre visszanyúló problémája. Ráadásul a folt megalkotója, ***Rusty Russel*** sem tűnik különösebben motiváltnak azzal kapcsolatban, hogy megoldja a helyzetet, sőt azt állítja, hogy ha a kérdéses kódrészlet egyszer kikerül a kernelből, akkor az egész dolog sokkal egyszerűbben megoldható lesz. Namármost ha szegény ***FUTEX_FD***-nek ilyen barátai vannak, akkor ellenségekre tulajdonképpen már nincs is szüksége, melynek következtében az sem biztos, hogy azt a türelmi időt élve kihúzza, amit ***Andrew Morton*** még adott neki.

Adrian Bunk hasonlóan számos olyan foltot tett közzé, amelyek egyes, évek óta működésképtelennek tekinthető meghajtó eltávolítását célozzák. Ilyen a *VIDEO_ZR36120*, az *SKMC* meghajtók, illetve a *MAC89x0*, az *ATARI_BIONET*, valamint az *ATARI_PAMSNET* nevű kódrészletek. Ami az utóbbiakat illeti, elképzelhető, hogy *Geert Uytterhoeven* képes lesz átadni *Matthias Ulrichsnak* egy olyan foltot, ami megjavítja a *MAC89x0* meghajtót. Ezt a javítást ugyanakkor még tesztelni kell, illetve jóvá kell hagyatni. *Adrian* közzétett néhány egyéb foltot is az *FB_CYBER*, *FB_VIRGE*, *FB_RETINAZ3*, *FB_ATARI*, *FB_SUN3* és *FB_PM3* meghajtók eltávolítására, itt viszont úgy tűnik lesz más megoldás is, mivel *James Simmonds* jelezte, hogy átvenné a karbantartást, maga *Geert* pedig szeretne egy javítást benyújtani az *FB_ATARI* meghajtóhoz. Egyszóval ezek a *framebuffer* meghajtók feltehetőleg megmaradhatnak a hivatalos kernelben.
Mindeközben a *sysctl* kód – amelynek nyaka már tényleg régóta a tönkön nyugszik és csak arra vár, hogy végre lesújtson a bárd – úgy tűnik megint megússza. *Eric Biedermannek* – ő jelölte nevezettet nyugdíjazásra – ugyanis sikerült találnia olyan felhasználókat, akiknek fontos ez az alrendszer. Ennek következtében nemcsak hogy leveszik a kérdéses kódrészletről az „elavult” jelzést, hanem a közeljövőben alapértelmezésként belefordítják az összes kernelbe. *Eric* eközben minden terjesztés fejlesztőivel megpróbálja kidolgozni a *sysctl* részleges kiiktatásának módozatait, majd körülbelül egy év múlva ismét benyújtja az eltávolítására irányuló javítást.
A *Sparse C* kódelemző (*code parser*) azon különleges rendszerek egyike, amelyeknek csak azt követően jelenik meg az első hivatalos kiadása, miután az eredeti fejlesztő már átadta valaki másnak a karbantartás jogát. Ezt a programot eredetileg *Linus Torvalds* írta 2003-ban saját használatra, egészen pontosan arra, hogy megkönnyítse vele a kernelfoltokban történő hibakeresést. Aztán szabadon letölthetővé tette, de úgy, hogy soha nem adott neki verziószámot, és a változatok követését sem szervezte meg.

Amikor a felhasználók elkezdtek olyan szolgáltatásokat kérni, amelyekre *Linusnak* magának nem volt szüksége, azt javasolta, hogy vegye át valaki a fejlesztést, és valósítsa meg azokat. Ekkor állt elő *Josh Triplett*, aki végül kiadta a *Sparse 0.1*-es változatát, amely az eredetihez képest már néhány új szolgáltatást is tartalmazott. Ezt rövidesen követte a *0.2*-es változat, ami viszont jobbára csak hibajavításokat tartalmazott. Aztán hogy ez most egy régi projekt, ami 2003-ban indult, vagy egy vadonatúj, aminek épp most jelent meg az első hivatalos kiadása, nos azt döntse el mindenki maga.

Linux Journal 2007., 156. szám

Zack Brown

PR

Sherlock Holmes kalandjai: a co-location eset

Sherlock Holmes és Dr. Watson már több mint egy évszázada nyűgözik le történeteikkel a krimirajongókat. A zseniális nyomozó – aki csupán pontos megfigyeléseire és kristálytiszta logikájára támaszkodva derítette fel a legrejtélyesebb ügyeket – és hű társa most a szerverkihelyezés dzsungelében fog segítséget nyújtani az eligazodáshoz.

Holmes órák óta szótlanul ült, hosszú, keskeny hátát a billentyűzet fölé görnyesztve. A szalonban ismerkedett új számítógépünkkel, míg én korábbi kalandjainkról készült jegyzeteimet rendezgettem. Fejét mélyen leszegte, ahogy a monitorra összpontosított. Onnan, ahol én ültem, úgy festett, mint egy fakószürke hajú, egérpucolásban megfáradt anorexiás rendszergazda. Hirtelen megszólalt a kapucsengő.

– Nos ***Watsonom*** – nézett fel – már ismeri a módszereimet, így biztos meg tudja mondani ki csengetett?
– De hisz ez pofonegyszerű! Véletlenül jól megfigyeltem a lépteket, ketten jöttek, férfiak. A csöngetés rövid volt, de határozott, igaz mielőtt csöngettek volna, egy rövid ideig tétováztak, informatikusok lehetnek.
– Mire alapozza a megállapításait kedves ***Watson*** – kérdezte ***Holmes*** meglepetten.
– Ó, még a nevüket is meg tudom mondani.
– Elkápráztatna vele doktorom, de kötve hiszem, hogy erre az ördöngösségre képes volna – mondta, de nagy örömömre némi tanácstalanságot véltem felfedezni barátom arcán.
– Pedig ha a szimatom nem csal, akkor épp ***Vashegyi Zoltánt*** és kollégáját, ***Nagy Konrádot*** engedi be a házvezetőnő.

Holmes nem kapott levegőt. Vékony, merész metszésű orra, amely oly elszánt kifejezést kölcsönzött arcának, dühödten rezgett. Csak kiváló neveltetése miatt volt képes uralkodni magán. Ekkor két úriember jelent meg a szalon ajtajában. Tovább fokoztam a hatást.

– Uraim, ***Dr. Watson*** vagyok. Hadd mutassam be önöknek ***Sherlock Holmesot*** a nagy detektívet! ***Holmes***, ***Vashegyi*** úr, szerver hoszting üzletág igazgató és ***Nagy*** úr, technikai igazgató, mindketten az ***Enternet 2001 Kft***-től.
– Elképeszt ***Watson***, most már ne csigázzon, honnan tudta mindezt – nézett rám szúrós szemekkel.
– Nyugodjon meg kedves ***Holmes***, egyszerű a dolog. Tegnap én kértem meg a két urat e-mailben, hogy némi felvilágosítással segítsenek szerver hoszting ügyben.
– Ah, így már értem! Ezek a ***XX.*** századi technikai csodák elkápráztatnak – sóhajtott fel megkönnyebbülten ***Holmes***. Uraim, kérem, foglaljanak helyet! Mi ez a szerver hoszting dolog – fordult a vendégeinkhez, s látszott, újra elemében van.
– A kiszolgáló számítógépeket, a szervereket vagy a cégek telephelyén, vagy úgynevezett szerverhotelekben üzemeltetik – kezdte a választ ***Vashegyi Zoltán***, ám ***Holmes*** türelmetlenül közbe kérdezett. Megnyugodva láttam, a barátom a régi.
– Miért jó, ha a szerver kikerül egy szerverhotelbe?
– Két fontos ok miatt – válaszolta ***Nagy Konrád***. Az első, ha egy szerver el kell érni a vállalaton kívülről, a kollégák a terepen dolgoznak. Pláne, ha több telephely van. Ha 10-15 telephelyről szeretnénk elérni egy adott vállalati szervert, akkor azt érdemesebb egy co-location központban elhelyezni. Itt a ***BIX***-re csatlakozva a legtöbb szolgáltató száz megabites sebességet allokál a szervereknek, ami azt jelenti, hogy sávszélességi problémák csak extrém igények esetén léphetnek fel. Sávszélesség: nagyobb cégek bérelt vonal, tíz megabit, ami komolyabb felhasználószámnál már akár kevés is lehet. ***ADSL***: reménytelen a feltöltési sebesség alacsony volta miatt.
– Egy pillanat – szakította félbe ***Nagy Konrád*** válaszát ***Holmes***. Watsonom, kérem gyűjtse össze az interneten a magyarországi szerver hoszting szolgáltatókat – fordult hozzám minden idők talán legnagyobb detektívje, majd a folytatásra bíztatta az ***Enternet*** szakemberét.
– A második a biztonság – folytatta a technikai igazgató a válaszadást. A legtöbb szervezetnek nincs lehetősége arra, hogy felépítsen magának egy minden igényt kielégítő szerverszobát, központot. A minden igénybe beleértem a tűzvédelmet, a klimatizációt, az olyan szintű szünetmentes áramellátást, amire a ***co-location*** szolgáltatók képesek. A vállalatoknál gyakori, hogy a szervert egy 500-1000 VA-es szünetmentes tápegységgel – úgynevezett ***UPS***-sel – „védik”, ami öt-tíz percig képes árammal ellátni a számítógépet, ez legtöbbször elég a szerver leállásához, de amikor nem, akkor nagy tud lenni a baj. A kihelyezett szerverközpontokban kétszintű tápellátásnak kell lennie.

Ugyanúgy van ***UPS***, mint az előbbi példában, de itt sok száz kilo-voltamperes teljesítménnyel. Ennek a szintnek ráadásul csak annyi a feladata, hogy addig tartson ki, amíg beindul a második szintű védelem, az aggregátor. Innentől pedig már csak a dízel mennyiségén múlik, hogy meddig tarthat az áramszünet. Ilyen üzembiztonság megteremtése gazdaságosan, csak a legnagyobb vállalatoknál lehetséges.
– És mi szól a saját telephelyen történő üzemeltetés mellett – előzött meg a kérdéssel ***Holmes***.
– A közvetlen hozzáférés. Ez a legfontosabb érv, ami miatt sok rendszergazda ódzkodik a szerverkihelyezéstől. Ha ugyanis hiba, leállás esetén nem lehet ***IP*** alapon elérni a szervert, akkor az valóban gondot okozhat. Korábban ilyen esetben telefonon kért az ügyeletestől egy újraindítást, és ha szerencséje volt, akkor elhárult a probléma. Ha nem, akkor indulhatott a szerverközpontba megbirkózni a problémával. Persze ma már ez sem igazi érv a saját telephelyen való üzemeltetésre, mert léteznek olyan konzolmegoldások, amelyek olcsók, és függetlenek az ***IP*** kapcsolattól, vagyis akár otthonról is úgy érheti el a szervert, mintha a gép előtt ülne. Ezekkel a konzolmegoldásokkal akár a gép ***BIOS***-át is lehet állítgatni, azaz jóval többet tudnak, mint a jól ismert terminál programok – mondta ***Nagy Konrád***.
– Rendben, értem. Tegyük fel, hogy elhatároztuk a szerverünk kihelyezését. Mi alapján válasszunk ***co-location*** szolgáltatót – érdeklődött tovább ***Holmes***.
– Az legfontosabb az, hogy tudjuk miért akarjuk kihelyezni a szervert. Másmás igényt támaszt például egy játékszerver, egy adatbázisszerver, vagy egy webszerver. A különböző típusoknál különböző a rendelkezésre állás, a sávszélesség igénye. Nagyon nem mindegy az ár. A spektrum ezen a téren a havi tízezer forinttól a csillagos égig terjed. Nagy hiba azonban csak az ár alapján dönteni. Van olyan szolgáltató, amely a meghirdetett – és nagyon olcsó – árba nem számítja bele a szünetmentes tápellátást, vagy a hálózati kapcsolatot.
– Igen. Több szolgáltatótól kell ajánlatot kérni. Ilyenkor azt tapasztalhatjuk, hogy egyes ajánlatok értelmezhetetlenek lesznek. Ezeknél nem lehet pontosan megállapítani, hogy milyen szolgáltatásokat kínálnak mekkora összegért. Ezekkel ne foglalkozzunk. Aztán kaphatunk olyan ajánlatot is, amikor a konzultációk során az az érzés hatalmasodhat el rajtunk, hogy ennek a szolgáltatónak, mi nem is vagyunk igazán fontosak. Gondolom azért, mert már annyira megy a szekerük, hogy úgy gondolják, diktálhatnak nekem. Érdemes tudakozódni, hogy

1. táblázat *Co-location árak Magyarországon – Dr. Watson tapasztalatai a hazánkban uralkodó viszonyokról*

Szolgáltató	Webcím	Szerverterem címe	Havi díj (ÁFA nélkül)
1B Telekom	www.1btelekom.hu	1132 Budapest Victor Hugo u. 18-22	10900*
3 IN 1 Hosting	megacp.com	1132 Budapest Victor Hugo u. 11-15 (nem BIX)	11000
T-Online (volt Axelero)	www.adatpark.hu	1052 Budapest, Petőfi Sándor u. 17-19.	19 900 + 15 000 egyszeri díj**
B&B Web – A tárhelyszolgáltató	www.bbweb.hu	(AdatPark, Szerver Hotel, DataHOME)	9666
ColdFusion Magyarország	www.cfm.hu	1132 Budapest Victor Hugo utca 11-15 és 18-22	12 000***
Covysoft	www.covysoft.net		15 000*
Cyber Systems	www.cyber.hu	1132 Budapest, Victor Hugo u. 18-22	14 990
Csucsnet	www.csucsnet.hu	1132 Budapest, Victor Hugo u. 18-22	20000
Dataland.hu	www.dataland.hu	1132 Budapest Victor Hugo u. 11-15.	7900****
Datanet	home.datanet.hu	2040 Budaörs, Ipartelep u. 13-15., és 1134 Budapest, Váci út 37/a	15 900*
Datatrans	www.datatrans.hu	Székesfehérvár, Budai út 9-11	15000
DENINET	www.deninet.hu	Budapest XIII. Victor Hugo u. 18-22.	12900*
Domainrendeles.hu	www.domainrendeles.hu	1132 Budapest, Victor Hugo u. 18-22	16000*
Dravanet	uj.dravanet.hu	7624 Pécs, Budai Nagy Antal u.1	30000
DreamNet	www.dreamnet.hu	1132 Budapest, Victor Hugo utca 11-15.	12000
DWO Serverhosting	www.dwo.hu	1132 Budapest, Victor Hugo utca 11-15.	12900*****
EasyHosting	www.easyhosting.hu	1132 Budapest, Victor Hugo u. 18-22	17800
Enternet	www.enternet.hu	1132 Budapest, Victor Hugo u. 18-22	24000
Invitel (Euroweb)	www.euroweb.hu	XIV.ker Ilka utca 31.	20000*
eWorld	eworld.hu		15000
GIGANEThosting webdesign	www.giganet.hu	Nyíregyháza, Szarvas u. 1-3.	16000

milyen belföldi és nemzetközi sávszélességgel rendelkezik az adott szolgáltató. Döntő lehet, hogy e kapcsolatok redundánsak-e? Ezen kívül még számtalan paramétert, és azt venném figyelembe a döntés előtt, hogy milyen többletszolgáltatásokat képesek nyújtani – javasolta ***Nagy Konrád***.

– A számtalan paraméterre, többletszolgáltatásokra mondana példát – kérdezte ***Holmes*** némi tűnődés után.

– Mielőtt bárkivel szerződnénk, rá kell kérdezni a tápellátás stabilitására, a klímahelyzetre. Ezek döntőek a szolgáltatás minőségében – jelentette ki ***Vashegyi Zoltán***. – Apróságnak, mármár kukacoskodónak tűnik, de ilyen paraméter például, hogy ki végezte a kábelezést az adott szolgáltatónál. Nagyon-nagyon kevés olyan céget ismerünk ***Magyarországon***, amelyik ezt az alapvető munkát kifogástalan minőségben, minőségi eszközökkel, jó áron eltudja elvégezni, úgy, hogy komoly referenciákat is fel tudjon mutatni. Olyannyira kevés ilyen céget ismerünk, hogy annak számossága nem haladja meg az egyet. A plusz szolgáltatásoknál fontos tudni, hogy kínálnak-e például mentési, vagy rendszergazdai szolgáltatást. Még, ha az induláskor nincs is szükségem ezekre, jó tudni, hogy ha nő az igényem, akkor azt ki tudják elégíteni.

– Az állva maradt ajánlatok között hogyan érdemes dönteni – kotyogtam közbe, de láttam ***Holmes*** nem nehezttel. Büszkeséggel töltött el, hogy talán Ő is ezt kérdezte volna.

– Keresni kell olyan ismerőst, akinek vannak tapasztalatai az adott cég tevékenységéről. Ha nincs ilyen, vagy ellentmondóak a tapasztalatok, érdemes internetes fórumokon keresgélni, itt egészen őszinte véleményekre lehet időnként bukkanni. Mi – amikor másfél éve elkezdünk ezzel a tevékenységgel foglalkozni –, azt a feladatot adtuk a rendszergazda kollégáknak, hogy gyűjtsék össze a fórumokról azokat a véleményeket, amelyek a szerverhoszting szolgáltatók hibáit taglalják. Ismerni akartuk ezeket, hogy nálunk ilyen hiányosságok ne forduljanak elő – jelentette ki ***Vashegyi Zoltán***.

– Milyen egyéb tényező befolyásolhatja a döntést – érdeklődött tovább ***Holmes***.

– Nagyon fontosak – ugyanúgy, mint a hétköznapi életben – az első benyomások. Nagyon nem mindegy ugyanis, hogy a szerződés előtti személyes találkozón érezzük azt, hogy a szolgáltató segít feltérképezni az igényeinket. Ha azt érezzük, hogy nem mi vagyunk a fontosak, hanem csak a pénztárcánk, álljunk tovább azonnal.

Persze a valódi tapasztalatokat csak a szerződés megkötése után szerezhetjük. Komoly gondoktól szabadulhatunk meg, ha nem kötjük le magunkat hosszabb időre. Kössünk határozatlan időre szerződést, és teszteljük a szolgáltatót. Próbáljuk ki, mennyi idő alatt és hogyan teljesítik a kéréseinket. Teszteljük a szolgáltató operátorait, ha szakmai, vagy kommunikációs problémákat tapasztalunk, azonnal váltsunk szolgáltatót. Ha viszont kulturáltan és gyorsan teljesítik a kéréseket, akkor nyugodtan kössünk hosszú távú, s ráadásul olcsóbb szerződést.

– Ahogy nézem ***Watson*** barátom táblázatát, nagyon sok társaság foglalkozik ***co-locationnel***. Ahogy látom az ***Enternet*** is. Uraim, önök kit ajánlanának jó szívvel, magukon kívül természetesen – tette fel az utolsó kérdését a barátom.

– De drága ***Holmes***! Ez az a kérdés, amire nem lehet jól válaszolni – mentettem ki a netes szakembereket a válaszadás kötelezettsége alól.

Joe Girnt

Szabadúszó újságíró, rendszergazda, aki ez úton is kéri Sir Arthur elnézését.

ICNnet	icnnet.hu	9400 Sopron, II. Rákóczi F. u. 5.	14500/ 15900* ******
Interware	www.interware.hu	1132 Budapest, Victor Hugo u. 18-22	18000*
iStorm-bird ISP	istorm-bird.hu	1132 Budapest, Victor Hugo u. 11-15.	12000
Mega gold	m-g96.com	1132 Budapest, Victor Hugo u. 11-15.	12000
Myhosting (Ultraweb)	www.myhosting.hu	1117 Budapest, Hauszmann Alajos u.3	18000
Panic Button Team	www.pbt.hu	1132 Budapest, Victor Hugo u. 18-22	30000 + 5000 egyszeri díj
Senorg	www.serverfarm.hu	1132 Budapest, Victor Hugo u. 18-22	19900
Services.hunetwork	services.hu	1132 Budapest, Victor Hugo u. 18-22	21990/17900******
SzerverDoktor	www.szerverdoktor.hu	1132 Budapest, Victor Hugo u. 18-22	9900*
Szerver Hosting.NET	www.szerverhosting.net	1132 Budapest, Victor Hugo u. 18-22	15900
Szerver hosting	www.easyhosting.hu		17800
Szerverhoszting.com	www.ultranet.hu	1132 Budapest, Victor Hugo u. 11-15.	10000*
Szervernet.hu	www.szervernet.hu	1132 Budapest, Victor Hugo u. 18-22	10000*
Tvnet	www.tvnet.hu	1132 Budapest, Victor Hugo u. 18-22	19900*

* Egy éves hűségnyilatkozattal
** Két éves hűségnyilatkozattal
*** Nincs feltüntetve, hogy az ár nettó-e vagy bruttó, egy éves hűségesküvel
**** 10 megabites kapcsolat
***** Havi díj egy évre előre fizetve
****** Szerverpolcon/rack szekrényben

Az adatok a szolgáltatók honlapjáról származnak, a gyűjtés időpontja 2007. 03. 10.

PR

Új SUSE Linux vékonykliens megoldás

A Novell értékesítési partnerei a SUSE Linux Enterprise Desktop rendszeren alapuló kész vékonykliens megoldást kínálnak az ügyfelek felügyeleti kihívásainak és szoftverköltségeinek csökkentése, valamint az adatbiztonság növelése érdekében.

A *Novell* bemutatta a *SUSE Linux Enterprise Thin Client* termékét, amely a *SUSE Linux Enterprise Desktop* rendszerből és egy lemezképkészítő eszközből áll, ezáltal a *Novell* értékesítési partnerei kész vékonykliens megoldást kínálhatnak ügyfeleiknek.
A hagyományos, jogvédett platformú munkaállomásokról a vékonykliens-eszközökre és a *Novell SUSE Linux Enterprise* rendszerét futtató munkaállomásokra történő áttérés az ügyfelek számára alacsonyabb költségeket, nagyobb adatbiztonságot és nagymértékben javuló felügyelhetőséget biztosít. A *SUSE Linux Enterprise Thin Client* az egyetlen nagyvállalati szintű *Linux* megoldás, amellyel a munkaállomások lemezképei egyszerűen beállíthatóak és felügyelhetőek. A munkaállomások felügyeleti feladatait, például az alkalmazások frissítését úgy egyszerűsíti le, hogy az alkalmazásokat a kiszolgálón tartja karban, miközben a kényes adatok központi tárolásával növeli a biztonságot. Emellett egyesíti a *Linux* vonzó költségelőnyeit és a *SUSE Linux Enterprise Desktop* egyedi innovatív képességeit.

A kaliforniai Windsor Unified School District 2000 számítógépet állított át vékonykliens-architektúrára

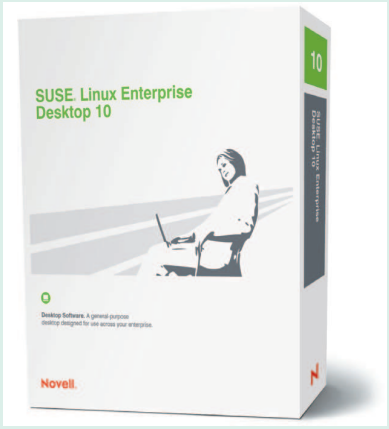
A kaliforniai *Windsor Unified School District* (windsori egyesült iskolakörzet) 2000 számítógépet állított át vékonykliens-architektúrára és *SUSE Linux Enterprise Desktop* rendszerre, így 75 százalékkal csökkentette hardverköltségeit, és majdnem 95 százalékkal szoftverköltségeit. *Heather Carver*, a *Windsor Unified School District* technológiai és információs szolgáltatásaiért felelős igazgatója elmondta, hogy amióta a *SUSE Linux Enterprise Desktop* rendszert és vékonykliens-képességeit használják, biztosak abban, hogy ez volt a legjobb választás. *Novell* megoldások azonnal nyereséget hoztak, hiszen radikálisan csökkentek a szoftverlicencek és a hardverek költségei. *Windsor Unified School District*-nél már nagyon várják, hogy kipróbálhassák a *SUSE Linux Enterprise Thin Client* új lemezképkészítő és felügyeleti képességeit is.

Az IDC szerint a Novell bejelentése tovább erősítette a Linux rendszerek használatát

Chris Ingle, az *IDC* tanácsadási és kutatási igazgatója szerint a vékonykliens alkalmazásokhoz épített célhardverek piaca évről évre 20 százalékos növekedést mutat, szemben a hagyományos munkaállomások 2 százalékos növekedésével. Számos szállító használja vagy próbálja ki a *Linux* rendszereket, hogy csökkentse költségeit vagy az egyes ügyfelek konkrét igényeihez szabott termékeket hozzon létre. A szakértő hozzátette, hogy a *Linux* használatát korábban a szakértelem és a támogatás hiánya korlátozta, a *Novell* támogatási bejelentése azonban tovább erősítette a *Linux* rendszerek használatát a gyorsan növekvő piacon.

A SUSE Linux Enterprise Desktop forradalmasította a Linux munkaállomások piacát

Roger Levy, a *Novell* alelnöke és a nyílt platformmegoldásokat fejlesztő részlegének vezetője szerint a *Novell* 2006 júliusában a *SUSE Linux Enterprise Desktop* megjelentetésével forradalmasította a *Linux*



munkaállomások piacát. Azóta számos szervezet vezette be ezt a megoldást, hagyományos munkaállomás, és testre szabott vékonykliens formájában egyaránt. Ezen ügyfélvisszajelzések alapján alkották meg vékonykliens-stratégiánkat a *SUSE Linux Enterprise Thin Client*, a díjnyertes *SUSE Linux Enterprise* platformra épülő új termék megjelentetésével. A *Novell* technológiájának és a partnerek szakértelmének köszönhetően az ügyfelek egy új, költséghatékony, rugalmas megoldáshoz jutnak.
A *SUSE Linux Enterprise Thin Client* munkaállomás-szoftvert és egy lemezképkészítő, -telepítő és -karbantartó eszközt kínál a különböző vékonykliens környezetekhez, beleértve a hívóközpontokat, termelőüzemek munkaállomásait és egyéb rögzített funkciójú és tranzakciós alkalmazásokat. A lemezképek számos különféle eszközre telepíthetők, például régebbi *PC*-kre, új *PC*-kre és vékonykliensekre specializálódott céleszközökre.

Az adattárházról egyszerűen – pillantás a fekete dobozba?

Az adattárház az Oracle kínálatában nem fekete doboz. Nagyon is transzparens, hiszen csakis általánosan elterjedt és szabványos kapcsolódási felületekkel rendelkező alkatrészekből áll.

Az adattárház definíciója szerint adatokat gyűjt, tárol és kínál fel másodlagos (tehát nem elsősorban tranzakciós típusú) hasznosításra. Több témakör adatait felöleli – a vállalati adattárház alapvetően az összes elemzendő témakörből –, és teszi elérhetővé elsősorban jelentésekben, lekérdező és elemző eszközökön keresztül.
A másodlagos hasznosítás itt nem az adattárház környezetvédelmi szerepére utal, hanem arra, hogy az alaptevékenység adminisztrálásán (pl. számlavezetés, számlázás, események regisztrálása, stb.) túlmenően a vállalatok és szervezetek által gyűjtött adatokból egy „mélyebb értelmezéssel", elemzéssel a vállalat, vagy intézmény működési hatékonyságának, az ügyfelek viselkedésének jellemzőit tudjuk szintetizálni; majd üzleti intelligencia eszközökkel megjeleníteni.
Ezen elemzési igények kiszolgálása az informatika fejlődési irányait elemző cégek szinte egyöntetű véleménye szerint elsődleges prioritássá vált az utóbbi évben (pl. ***2007 Gartner EXP – CIO Survey***, ***2006 Merrill Lynch CIO Survey***).
Ne titkoljuk el, hogy sok jelentési, elemzési feladatot meg lehet oldani adattárház nélkül is, hiszen az ***Oracle*** kínálatában ma már léteznek olyan üzleti intelligencia eszközök, amelyek közvetlenül az operatív adatbázisok adatain is képesek akár federatív módon, több tématerület feldolgozását támogató, adatbázisokból egyidejűleg, kombináltan is lekérdezéseket végrehajtani, akár heterogén környezetben is. Vannak azonban esetek, amikor nem tudjuk elkerülni az adattárház építését. Néhány példa ezekre az esetekre:

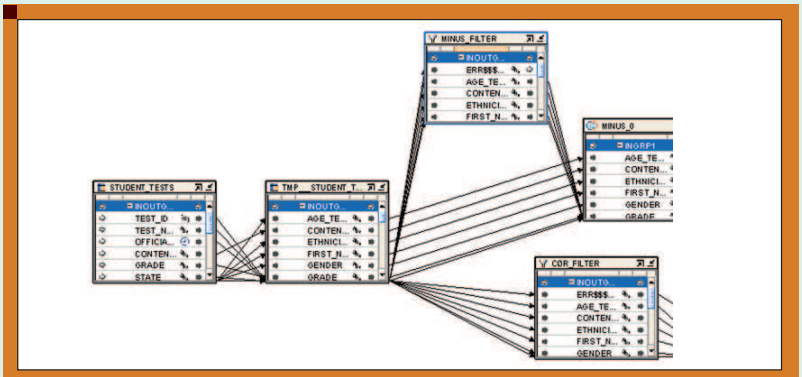
- az adatok nem őrződnek meg kellően hosszú ideig a tranzakciós rendszerek adatbázisaiban
- az elemző lekérdezések akadályozzák, lassítják a tranzakciós munkát
- az alaprendszeri adatséma túl bonyolult a lekérdező eszközök számára
- az egyszerű aggregációkon és kalkulációkon túlmenően bonyolult algoritmusokat is be akarunk vetni új információ szintetizálására
- jelentős adatminőség javításra van szükség
- biztonsági okokból nem akarunk hozzáférést adni egy csoport számára egy, vagy több alaprendszeri adatbázishoz

Amennyiben csak egy témakörrel foglalkozik nevezhetjük adatpiacnak; jelentés adatbázisnak – ha elsősorban jelentések készítését támogatja (mi mást tehetne); vagy integrációs platformnak, ha az adatintegrációs képességét akarjuk kihangsúlyozni; vagy használhatjuk szívünk kedvenc szavát is, amennyiben az adattárház szóhoz a kiszemelt belső szponzorok azonnal horribilis költségeket asszociálnának.
Az adattárház építés kétségtelenül az informatika egyik legizgalmasabb, nagy áttekintést ígérő és ugyanakkor legkockázatosabb vállalkozása.
Ha fenti pontok alapján adattárház építésbe kell kezdenünk, akkor pillantsunk bele dobozunkba. Érdemes, hiszen, az adattárház intenzív kapcsolatban van a környezetével. Egy kiterjedt adattárháznak rengeteg ki és bemeneti ponton van kapcsolata az alap informatikai rendszerekkel, a felhasználókkal, fejlesztőkkel és üzemeltetőkkel. Ilyen értelemben nem érdemes, és nem is lehet, fekete dobozként kezelni.
Az adattárház belsejét érdemes elterjedt, és szabványos alkatrészekkel kitölteni:

- egyrészről azért, mert a ***TPC-H benchmarkok*** (lásd. ***tpc.org***) tanúsága szerint az elterjedten használt általános célú relációs adatbázis kezelők jobban szerepelnek teljesítmény és ár/érték jellemzők alapján is, mint a specializált adatbázis motorok
- másrészt mert a hosszú távú költségszámításokban (***Total Cost of Ownership – TCO***) már nem elhanyagolható, hogy az egyébként is elterjedten használt adatbázis kezelőnkhöz házon belül, és a piacon kiterjedten férhető hozzá a tudás, amivel a rendszer életben tartása, és továbbfejlesztése megoldható.
- szintén nem elhanyagolható erejű érv az, hogy az általános célú adatbázis-kezelő security, hordozhatóság és skálázódás szempontból jelenleg éveket ver rá specializált társaira.

Linuxon az ***Oracle*** a következő ***TPC-H*** rekordokat tartja:

- 300 gigabyte
- 3 terabyte



1. ábra Megfelelő adatelrendezés az Oracle Warehouse Builder segítségével

A *Winter Corporation* által auditált rendszerek között a világ legnagyobb *Linuxon* futó adattárháza az *Amazon* által működtetett több mint 24 terabyte-os *Oracle* alapú adattárház. Az egyéb (nem adattárház és nem *OLTP*) kategóriában is az *Oracle* viszi el a pálmát a *Max Planck intézet Linuxon* futó 222 terabyte-os adatbázisával.

Az *Oracle* adatbázis-kezelője több piackutató cég szerint egybehangzóan 40%-át fedi le a relációs adatbázis alapon épített adattárházak piacának (pl. *IDC* 2006 adattárház piac).

Az *Oracle* adatbázis-kezelője a különböző cég méretekre és adatbázis-kezelési igényekre alapozva többféle változatban is kapható. Különböző méretű adattárházi, vagy adatpiaci projektekben sikerrel alkalmazzák az ingyenes legkisebb adatbázis kezelőtől az *Enterprise* változatig bezárólag mindegyik verzióját. Ugyanakkor mindegyik *Oracle* adatbázis-kezelő ugyanazt a kódot futtatja, melyek között az áttérési lehetőséget a cég egyszerűen biztosítja, amellett, hogy az egyik változatba fektetett munka nem vész el az áttéréskor.

Természetesen az *Oracle Database* és az *Oracle* üzleti intelligencia eszközök is számos platformon, így *Linuxon* is működnek. Ha valaki például szeretne áttérni *Linux* platformra, roppant egyszerűen migrálhat *Linuxra*.

Újdonság, hogy az adatfeltöltésre – és a megfelelő adatelrendezést biztosító adat-átalakítások elvégzésére ajánlott *Oracle Warehouse Builder* eszköz ingyenes kiegészítőként bekerült az *Oracle* adatbázis-kezelő minden megvásárolható kiadásába. Bonyolultabb esetekre, amikor a fejlett adattisztítás, vagy a több környezetben történő futtatás is igény, az ingyenes alap mellé kiterjedt funkcionalitással rendelkező fizetős opciókat is kínál a cég, csakúgy mint az *OLAP* és adatbányászati és particionálási funkciók támogatására.

A *Warehouse Builder Data Correction* varázslója automatikusan generál adat javító függvényeket. Az alábbi példában az `SDX_CONTENT_AREA` függvény látható.

```
type t_arr is table of
➥varchar2(2000) index by
➥binary_integer;
s t_arr;
begin
  s(0):= 'MATH';  s(1):=
➥'READING';  s(2):=
➥'SCIENCE';  s(3):= 'WRITING';

  - check for equality first.
➥Function may be called when
➥there is no error.
  for i in s.first..s.last loop
    if (s(i) = CONTENT_AREA)
➥then
      return CONTENT_AREA;
    end if;
  end loop;

  for i in s.first..s.last loop
    if (soundex(s(i)) =
➥soundex(CONTENT_AREA)) then
      return s(i);
    end if;
  end loop;
  return CONTENT_AREA;
end;
```

A függvény lefuttatását követően például az angolul hibásan írt `WRITEING` automatikusan javításra kerül a helyes `WRITING` szóra.

Az *Oracle* alapú adattárházban gyűjtött adatok kiaknázását számos gyártó lekérdező és elemző eszközkészlete támogatja. Érdemes mégis megismerni, és kipróbálni az *Oracle* kínálatában teljesen megújult üzleti intelligencia csomagokat, melyek egy dashboard keretben biztosítják a jelentéskészítést és -megtekintést, és az ad-hoc lekérdezések és a különböző eszközökre küldhető figyelmeztetések kiküldését.

A jól megválasztott szoftver eszközpark persze csak egy összetevője a sikeres adattárház projektnek. Mivel a kockázatok csökkentésében jelentős szerepe van a szabványos elemek használatának, ne csak nézzünk bele a dobozba, hanem válogassunk addig a számunkra szükséges alkatrészekből, amíg a problémának megfelelő rendszert sikerül összerakni.

Radnai Szabolcs
(szabolcs.radnai@oracle.com)

Radnai Szabolcs az Oracle Hungary Kft. 1993-as megalakulása óta dolgozik az Oracle-nél. Tíz éve foglalkozik üzleti intelligencia és adattárház megoldási javaslatok kidolgozásával. Jelenleg az Üzleti Intelligencia és Adattárház üzletág vezetője.

KAPCSOLÓDÓ CÍMEK

- http://oracle.com/bi
- http://www.gartner.com/it/page.jsp?id=501189
- http://www.gartner.com/press_releases/asset_152619_11.html

Hol szorít a cipő?

Mostanában annyi pingvinekről szóló film készül, tán okkal gondolhatnánk, a legújabb animáció, a Táncoló talpak ihletett meg néhány fiókát, hogy színes cipőkben illegjen a kövezeten, hátha egy arra sétáló rendező felfigyel tehetségére.

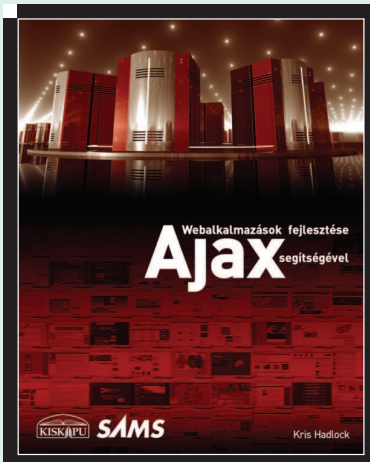
Persze a valóság nem erősíti meg ezen gondolatokat. A cipellőket viselő ifjú madárkák egy új-zélandi menhely lakói, a tetszetős lábbelik pedig sérült végtagjaikat hivatottak védeni. A felsebzett lábú csemetéknek balzsamozást és sós-vizes lábfürdőt is elrendelt a doktor, s noha e szolgáltatást valószínűleg nem vibrációs, hidromasszázs kiskádakban vehetik igénybe a kerti szikla peremén ücsörögve, a direkte számukra tervezett cipők is mutatják, mennyi gondosságban van részük. Ugyanakkor a gondoskodás sós-vizes lábfürdő ***ellen*** is mutatkozik. Szintén egyedi tervezésűek azon lábbelik, miket ebeknek szánnak, hogy a télen sóval felszórt latyakban ne sebesedjen ki tappancsuk. Egyes tervezőknek és vásárlóiknak nem ártana megszámolnunk lábaikat, biztosan nem öszvérek-e, mert bár igaz, hogy annak idején, valamikor a 17. században a magassarkú topánok elsősorban az utcai mocsokból való felszínen maradást szolgálták, s nem pedig a pipiskedést, mégis túlzás ily fazonokat ráhúzni négylábú csahosokra. Még csakcsak megértem a sötétben világító modelleket, a különc, strasszokkal kirakott szandálkákat, de a fűzős sportcipőknél már gondolkodóba esem, és fejben számolom a láb/cipőfűző kötési időhányadost.

Kínában nem lacafacáznak: az ebet cserélik le cipőre. Nevezetesen vakvezető ebeket tesznek munkanélkülivé olyan cipő kifejlesztésével, melyben mini komputer jelzi vibrálva viselőjének, ha akadályt észlel a haladásban, s e lábbelit műholdas helymeghatározó rendszerrel is megspékelik.

Lehet, hogy ilyen ***GPS***-el volt felszerelve egy körömágy-gyulladt elefántleány ormánya is, hisz ápolói mentem cipellőt húztak őreá, midőn észlelték, hogy eme megnyúlt testrészével tépegeti le a kötést lábáról. Talán neki szánták a mellékelt fotón látható bakancsot, (túl) szorgosan ügyelve arra, nehogy szorítson az a cipő…

Halusz Léna

Hatékonyabb interaktív webalkalmazások



Cím: Webalkalmazások fejlesztése Ajax segítségével
Szerző: Kris Hadlock
Kiadó: Kiskapu (SAMS)
Oldalszám: 288
Ár: 3980 Ft

Manapság ha megkérdezünk egy webfejlesztőt, hogy milyen technológiákra támaszkodva készít honlapokat, két választ fogunk nagyon gyakran kapni: ***Flash*** és ***AJAX***.
Már korábban is voltak az ***AJAX***-hoz hasonló próbálkozások – ***IFRAME*** a ***Microsoft Internet Explorer 3.0***-ban és a ***LAYER*** a ***Netscape 4***-ben – de maga az ***AJAX (Aszinkron Javascript és XML)*** betűszó és megvalósítás csak 2005-ben robbant be a köztudatba.
A legnagyobb sajtóvisszhangot talán a ***Google Mail*** kapcsán kapta a technológia.
Azóta már számos angol nyelvű könyv jelent meg a témában, de magyarul eddig csupán egy-két könyv (***Javascript zsebkönyv*** és a ***Tanuljuk meg a JavaScript használatát 24 óra alatt***) egy-egy fejezete támogatta a vállalkozó kedvűeket.

Jelen könyv tehát hiánypótló lehet azok számára, akiknek nehézséget jelent az angol nyelvű könyvek vagy a témába vágó egyéb szövegek megértése.
A könyv hat – teljesen logikusan kijelölt határokkal bíró részre tagolódik:

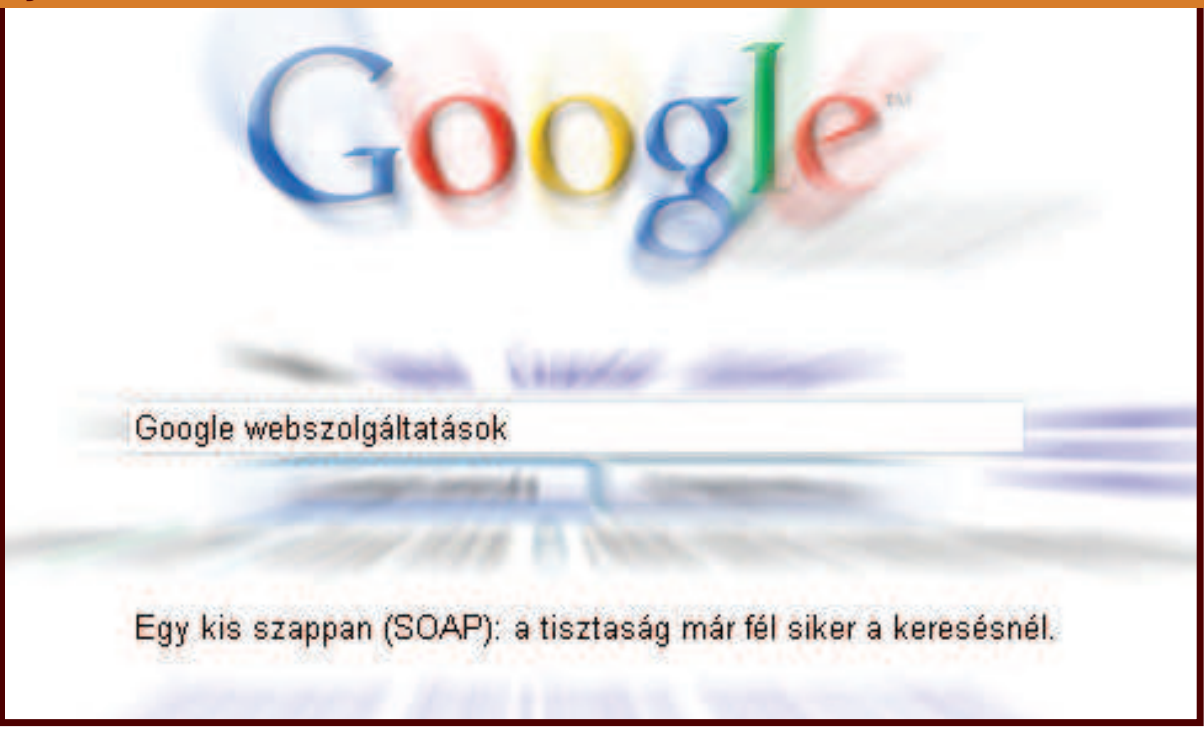
- ***Az első lépések*** – avagy induljunk el a kályhától. Pár szó az ***AJAX*** történetéről és az alapokról.
- ***A Javascript-motor létrehozása és használata*** – a kliens oldali rész (***Javascript***) létrehozása.
- ***Újrahasznosítható elemek létrehozása*** – a könyv szerzője létrehoz pár olyan alkalmazásrészletet (fanézet, ügyfél oldali ellenőrzés, adatrácsok), amely minden alkalmazásban benne van, így nem kell minden projektnél újra feltalálni a kereket.
- ***Ajax-minták*** – Jelen rész a különböző ***AJAX*** fejlesztési mintákat tárgyalja számba véve az előnyeiket és hátrányaikat.
- ***Együttműködés a kiszolgálóoldallal*** – a rész címe magáért beszél, azonban elég rövid, így amennyiben nem rendelkezünk biztos és megfelelő alappal a témában, érdemes más könyveket is beszereznünk. (Például a ***Tanuljuk meg 24 óra alatt*** sorozatból a ***PHP***-t és a ***MySQL***-t)
- ***Végső simítások*** – az utolsó és talán legrövidebb rész kitér az alkalmazásunk biztonságosabbá tételének lehetőségére, illetve pár gyakorlati jó tanáccsal is ellát.

Mindazonáltal a könyv mellé hasznos kiegészítő lehet és segíthet a biztonság fokozásában az áprilisi ***Linuxvilág***ban bemutatott ***Hogyan törjünk fel webhelyeket*** című könyv.

Az ***AJAX*** mindenképp olyan technológia, amely nem hiányozhat egy webes alkalmazások fejlesztésével foglalkozó szakember eszköztárából. Életképességét mutatja, hogy a ***Google*** számos ***AJAX***-os alkalmazást adott ki (***Gmail***, ***Calendar***, ***Docs & Spreadsheet***, stb.), amelyek mármár egy teljes irodai környezetté állnak össze és már jelenleg is használható alternatívát nyújtanak a lokálisan futtatott – kereskedelmi vagy nyílt forrású – megfelelőikkel szemben.
Persze azt sosem szabad elfelejtenünk, hogy az ***AJAX*** nem valami ***Szent Grál***, hiszen számos webböngésző létezik, amely nem – vagy nem megfelelően – kezeli az ***AJAX***-os oldalakat, vagy esetleg a felhasználó biztonsági okok miatt letiltotta a ***Javascript***-ek futtatását.
Hasznos és kedvcsináló olvasmány lehet a könyv megvásárlása előtt ***Radics Péter Kommunikáció „fű alatt”*** című cikksorozata, melynek első része a ***Linuxvilág*** 2007. februári számában jelent meg.

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.



Az elmúlt hónapokban több olyan az ***Amazon*** által rendelkezésre bocsátott webszolgáltatását vizsgáltunk, amelyek lehetővé teszik, hogy viszonylag könnyedén keressünk annak katalógusában. Az ***Amazon*** jó néhány évvel ezelőtt úgy döntött, hogy a webszolgáltatásainak nagy részét ingyenessé teszi, azt feltételezve, hogy így többen lesznek azok, akik végül vásárolnak a webhelyükről. Valóban, ma már rengeteg fejlesztő használja az ***Amazon*** webszolgáltatásait különböző dolgok létrehozására, az egyedi könyvesboltoktól kezdve a könyvesbolt-kezelést segítő programokig.

Az ***Amazon*** nem az egyetlen kereskedelmi webhely, amely megnyitotta a katalógusát a külvilág számára. Az internet másik ***King Kongja***, a ***Google*** néhány évvel ezelőtt szintén közzétette a web ***API***-jait (alkalmazásprogramozó illesztőfelület). Ezek az ***API***-k lehetővé teszik a ***Google*** terjedelmes webes katalógusában történő keresést. Lehetetlen megmondani, hogy a világ legnagyobb katalógusáról van-e szó, de véleményem szerint ez nem annyira lényeges. A ***Google*** katalógusa elég nagy és elég gyakran frissül ahhoz, hogy az esetek döntő többségében azt használjam elsődleges keresőmotorként.

A ***Google*** az utóbbi néhány évben számos különböző ***API***-t tett elérhetővé. Ebben a hónapban a legegyszerűbbet vizsgáljuk, amely a webes archívumban történő alapvető keresésekre szolgál. Megvizsgáljuk, hogy a ***Google*** hogyan reklámozza a webszolgáltatásait a ***WSDL (Web Service Description Language – Webszolgáltatás-leíró nyelv)*** segítségével, és hogy miként indíthatunk olyan ***SOAP*** hívásokat, amelyekkel a saját céljainknak megfelelő kereséseket végezhetünk a ***Google*** hatalmas könyvtárában.

Az első lépések

A ***Google API***-k első lépései nem tartogatnak különösebb meglepetést azok számára, akik dolgoztak már az ***Amazon*** webszolgáltatásaival. Kezdéskor mindkét cég elvárja a regisztrációt a szolgáltatások használatához. A – mindkét esetben ingyenes – regisztráció során kapunk egy azonosítót, amely a kiszolgálóhoz küldött összes kérésben szerepel majd. Ahhoz, hogy ***Google*** kulcsot szerezzünk, először ***Google*** fiókot kell nyitnunk. Nos, egy ideje már rendelkezem „Google fiókkal”, amelyet a ***Gmail*** illetve a testreszabott híroldal szolgáltatáshoz használok. Úgy tűnik azonban, hogy az ***API***-k egy másik fiókcsoporthoz tartoznak.

Kissé furcsának találtam, a „fő” Google fiókba történő bejelentkezés után az ***API*** rendszerben még külön regisztrációra és bejelentkezésre van szükség.

Mindezek ellenére a fiók létrehozása egyszerű és magától értetődő. Menjünk a ***Google API***-k főoldalára (➲ ***www.google.com/apis***), kattintsunk a ***„create Google account”*** (***Google*** fiók létrehozása) szövegre, majd töltsük ki az űrlapot. A ***HTML*** űrlap elküldése után nem sokkal a fiók létrehozását visszaigazoló e-mailt kapunk a ***Google***-től, amely tartalmazza a ***Google*** kulcsot, valamint a fiók létrehozásának megerősítéséhez szükséges ***URL***-t. A fiók létrehozásának jóváhagyása után máris elkezdhetjük használni a ***Google*** kulcsot, és olyan programokat készíthetünk, amelyek kihasználják a ***Google*** webszolgáltatásainak előnyeit.

Azonban mielőtt ezt megtennénk, érdemes megfontolnunk, hogy a ***Google*** milyen módon korlátozásokat alkalmaz a szolgáltatásnál, és hogy milyen adatokat kapunk azon keresztül. Az ***Amazon*** másodpercenként csak egy ***API*** hívást engedélyez a tagoknak, ami egy adott 24 órás időtartamot tekintve legfeljebb 86400 hívás. Ezzel szemben a ***Google*** 24 óránként legfeljebb 1000 hívást tesz lehetővé a felhasználók számára.

Ezen felül a határértékek meghatározásának módja jelzi, hogy miként kezelik a korlátok túllépését.
A *Google* akkor ad vissza hibaüzenetet, ha az elmúlt 24 órában több, mint 1000 lekérdezést végeztünk, míg az Amazon csak olyankor jelez, amikor egy lekérdezés ugyanabban a másodpercben érkezik, mint az előző.
A hibaüzenet küldését megelőzően egyik szolgáltatás sem figyeli ezeket a számokat, az Amazon korlátozásának túllépését azonban nyilvánvalóan könnyebb orvosolni (egy másodperc felfüggesztéssel, majd újbóli próbálkozással), mint a *Google*-ét (mivel előfordulhat, hogy a programot akár 24 óráig fel kell függeszteni, mielőtt újra próbálkoznánk).
Jogi szempontból a két webhely szolgáltatásai számos ponton különböznek. Az *Amazon* állt elő elsőként azzal az ötlettel, hogy társul a webes kereskedőkkel, az adatbázisához kapcsolódó kereskedelmi szolgáltatások létrehozására buzdítva az embereket. Ezzel szemben a *Google* kifejezetten megtiltja a felhasználóknak, hogy a keresési találatokra építve kereskedelmi szolgáltatást hozzanak létre. (Ha valaki internetes keresési adatokra épülő kereskedelmi szolgáltatást szeretne létrehozni, érdemes megtekinteni az *Amazon Alexa Web* keresőplatform szolgáltatását, amely nem alkalmaz ilyen korlátozásokat. Ugyanakkor 1000 kérésenként 25 centet kell fizetni, ami egy népszerű webhelyen hamar megsokszorozódhat.)
Végül, a két webhely között technikai eltéréseket is tapasztalhatunk. Az *Amazon API*-jai *SOAP*-on és *REST*-en keresztül egyaránt működnek, így a fejlesztők választhatnak a két formátum közül. A *Google* viszont csak *SOAP* felületet biztosít a keresőmotorjához, tehát a keresőrendszer létrehozásához *SOAP* ügyfélkönyvtárat kell telepítenünk és alkalmaznunk. Szerencsére a legtöbb nyelv rendelkezik olyan magas szintű könyvtárakkal, amelyek lehetővé teszik a *SOAP* hívásokat.

SOAP::Lite

A *SOAP* – amelyet korábban *Simple Object Acces Protocol*-nak (*egyszerű objektum-hozzáférési protokoll*) neveztek, a betűszó azonban ma már hivatalosan nem jelent semmit – segítségével viszonylag könnyen küldhetünk *XML*-be tokozott lekérdezéseket a kiszolgálónak. Ezután a kiszolgáló *XML*-ként kódolt választ küld vissza. Az évek során a *SOAP* messzire kalandozott az egyszerű gyökereitől. Annak ellenére, hogy a *SOAP* még mindig könnyebben érthető, megvalósítható és alkalmazható, mint bizonyos bonyolultabb protokollok (például a *COBRA*), sokkal nehézkesebb, mint amennyire azt a legtöbben elismerik. Amennyiben lehetséges, én személy szerint jobban szeretek *XML-RPC*-t használni a webszolgáltatásokhoz. Az *XML-RPC* ugyan nem biztosítja a *SOAP* összes szolgáltatását, de sokkal könnyebb vele dolgozni.
Mindenesetre a *Google* a *SOAP* használatát várja el, és mivel manapság számos jó *SOAP* ügyfélkönyvtár létezik, nem kell, hogy féljünk a használatától. A *Perl* programozóknak egy különösen erős megvalósítás áll rendelkezésére, amelyet *SOAP::Lite*-nak hívnak. A cikk programozási példáiban a *Perlt* és a *Soap::Lite*-ot alkalmazzuk. Fontos, hogy a modulnév *Lite* (könnyű) része azt mutatja, hogy a programozók mennyire könnyen megvalósíthatják a webszolgáltatásokat, és nem a *SOAP* lebutított változatát jelöli. A következő paranccsal telepíthetjük a *SOAP::Lite* legfrissebb változatát a *CPAN*-ból:

```
perl -MCPAN -e 'install
➥SOAP::Lite'
```

A *SOAP::Lite* telepítés megkérdezi, hogy milyen ellenőrzéseket szeretnénk végezni a modul telepítése előtt, ha egyáltalán akarunk tesztet futtatni. Én általában elfogadom az alapbeállításokat, de az igényeinknek megfelelően esetleg néhányat hozzáadhatunk vagy eltávolíthatunk.
A *SOAP::Lite* telepítése után ideje írnunk egy olyan programot, amelyhez *Google* szükséges. Ehhez azonban ismernünk kell a szolgáltatás *URL*-jét, azt, hogy milyen eljárást hívunk meg a *Google* számítógépén, valamint a küldeni kívánt összes paraméter típusát és nevét. Ezeket kézzel is megadhatjuk, de ez sok munkát jelentene számunkra. Ezen kívül a *Google* jelenleg az ➲ *api.google.com/search/beta2* címre várja a *SOAP* kéréseket. Ha a *Google* valaha figyelmeztetés nélkül megváltoztatja ezt az *URL*-t, valószínűleg sokan meglepődnek és bosszankodnak majd.
Szerencsére a *Google* rendelkezésre bocsátott egy *WSDL* fájlt, amely leírja a *Google API*-k nyújtotta szolgáltatásokat, valamint a rendszer által elfogadott kérés és válasz paramétereket. Emellett meghatározza a lekérdezések célállomását, ami (elméletben) lehetővé teszi a *Google* számára, hogy a fejlesztők előzetes értesítése nélkül változtasson a szolgáltatáson. Természetesen ez azt feltételezi, hogy maga a *WSDL* fájl azonos helyen marad, továbbá, hogy a szolgáltatások neve sem változik és azok mindegyike valahol dokumentált, mivel a meghívott eljárások kiválasztása továbbra is emberi beavatkozást igényel.
A *WSDL XML*-ben íródott, és meglehetősen könnyen érthető, ha felismerjük, hogy csupán egy adott kiszolgálón elérhető különböző webszolgáltatásokat jellemzi: a bemenetek számát, nevét és típusát írja le.
A `doGoogleSearch` *WSDL* bejegyzése így a következő:

```
<message name="doGoogleSearch">
<part name="key"
➥type="xsd:string"/>
<part name="q"
➥type="xsd:string"/>
<part name="start"
➥type="xsd:int"/>
<part name="maxResults"
➥type="xsd:int"/>
<part name="filter"
➥type="xsd:boolean"/>
<part name="restrict"
➥type="xsd:string"/>
<part name="safeSearch"
➥type="xsd:boolean"/>
<part name="lr"
➥type="xsd:string"/>
<part name="ie"
➥type="xsd:string"/>
<part name="oe"
➥type="xsd:string"/>
</message>
```

Ahhoz, hogy egy *SOAP::Lite*-ot használó *Perl* programon belül *WSDL*-t alkalmazzunk, a `SOAP::Lite->service` parancsot hívjuk meg az *WSDL* fájl

1. Lista google-query.pl

```
#!/usr/bin/perl
use strict;
use diagnostics;
use warnings;
use SOAP::Lite;
# --------------------------------
# Get the Google key from ~/.google_key
my $google_key_file = "/Users/
➥reuven/.google_key";
open GOOGLE_KEY, $google_key_file or die
➥"Cannot read
'$google_key_file': $! ";
my ($google_key) = <GOOGLE_KEY>;
chomp $google_key;
close GOOGLE_KEY;
# --------------------------------
# Get the command-line argument
if ($#ARGV != 0)
{
    print "$0: Invoke with a single argument,
    ➥your Google search term.\n";
    exit;
}
my $query_string = shift @ARGV;
# --------------------------------
# Get the WSDL file
my $google_wsdl = "http://api.google.com/
➥GoogleSearch.wsdl";
my $query = SOAP::Lite->service($google_wsdl);
# --------------------------------
# Use the WSDL to make the query
my $starting_page = 1;
my $max_results = 10;
my $filter = 'false';
my $geographic_restriction = '';
my $safe_search = 'false';
my $language_restriction = '';
my $results =
    $query->doGoogleSearch($google_key,
                $query_string,
                $starting_page,
                $max_results,
                $filter,
                $geographic_restriction,
                $safe_search,
                $language_restriction, 'utf-8',
                ➥'utf-8');
my @results = @{$results->{resultElements}};
if (@results)
{
    # Iterate through each result we got
    my $counter = 1;
    foreach my $result (@results)
    {
    print "Result $counter of ", $#results + 1,
    ➥":\n";
    foreach my $key (sort keys %{$result})
    {
        my $value = $result->{$key};
        # Is this a hash value?  If so,
        Vdisplay it accordingly
        if (UNIVERSAL::isa($value, 'HASH'))
        {
        print "\t'$key':\n";
        foreach my $subkey (sort keys
        ➥%{$value})
        {
            print "\t\t'$subkey' => '$value->
            ➥{$subkey}'\n";
        }
        }
        # Display the value as a simple string
        else
        {
        print "\t'$key' => '$value'\n";
        }
    }
    $counter++;
    }
}
else
{
    print "There were no results for your query
    ➥of '$query_string'.\n";
}
```

URL-jével. Ha a fájl a helyi fájlrendszerben található, ügyeljünk, hogy az ***URL*** előtt `file:` álljon. Például:

```
my $google_wsdl =
➥"http://api.google.com/
➥GoogleSearch.wsdl";
my $query = SOAP::Lite->service
➥($google_wsdl);
```

Ezután a ***SOAP::Lite*** elég okos ahhoz, hogy végignézze a ***WSDL***-t és dinamikusan elérhetővé tegye az összes hirdetett szolgáltatást, hogy a következőt tehessük:

```
my $results =
$query->doGoogleSearch
➥($google_key,
                $query_string,
                $starting_page,
                $max_results,
                $filter,
                $geographic_
                ➥restriction,
                $safe_search,
                $language_
                ➥restriction,
                ➥'utf-8',
                ➥'utf-8');
```

Mi történik itt? A ***WSDL***-ben meghatározott bemenetek és a `$query->doGoogleSearch()`-nek átadott paraméterek egy az egyben leképezhetők egymással.

Egyszerű keresések a doGoogleSearch segítségével

Láttuk tehát a *Perlben* írott *Google* keresőprogram magját. Már csak annyi van hátra, hogy áttekintsük a bemeneti paramétereket és a `$results` tartalmát, amelyben a *Google* által visszaadott találatok szerepelnek.

Az *API* ➲ *www.google.com/apis/reference* címen található dokumentációja leírja a bemeneti paramétereket. Mindegyik kötelező, de vannak olyanok, amelyek fontosabbak, mint a többi: a *Google kulcsot* (`google_key`) és a lekérdezés-karakterláncot (`query_string`) általában megadjuk, a többit pedig egyszerűen alapértékkel adjuk meg, ahogy azt az *1. Listában* láthatjuk.

Az emberek többsége (magamat is beleértve) általában a lehető legtöbb weboldalt akarja keresni a lekérdezésekkel, mégis vannak olyan esetek, amikor célszerűbb, ha csak egy bizonyos terület vagy nyelv kiszolgálóiról gyűjtünk adatokat. Mivel a *Google* ezt lehetségessé, sőt, magától értetődővé teszi, számos különféle érdekes alkalmazás előtt nyílik meg az út.

Ahogy *SOAP*-ban kódolt *XML*-ként küldjük a lekérdezést a *Google*-nak, ugyanúgy, *SOAP*-ban kódolt *XML*-t kapunk eredményként. Mivel azonban a *SOAP::Lite* megkímél bennünket attól, hogy akár egy hajszálnyi *XML*-t is kelljen írnunk a lekérdezéshez, a válasz hasonlóképpen elszigetelődik. A `$results` változó *Perl* kezelőfelületet biztosít a válaszként kapott adatokhoz.

Pontosan milyen adatokat is kapunk? Ahhoz, hogy ezt megtudjuk, ismét meg kell néznünk a *WSDL* fájlt. Többek között azt mutatja, hogy az alábbihoz hasonló eredmények halmazaként kapjuk meg a válaszokat:

```
<xsd:complexType name=
➥"ResultElement">
<xsd:all>
    <xsd:element name="summary"
    ➥type="xsd:string"/>
    <xsd:element name="URL"
    ➥type="xsd:string"/>
    <xsd:element name="snippet"
    ➥type="xsd:string"/>
    <xsd:element name="title"
    ➥type="xsd:string"/>
    <xsd:element name=
    ➥"cachedSize" type=
    ➥"xsd:string"/>
    <xsd:element name=
    ➥"relatedInformationPresent"
    ➥type="xsd:boolean"/>
    <xsd:element name=
    ➥"hostName" type=
    ➥"xsd:string"/>
    <xsd:element name=
    ➥"directoryCategory" type=
    ➥"typens:
    ➥DirectoryCategory"/>
    <xsd:element name=
    ➥"directoryTitle" type=
    ➥"xsd:string"/>
</xsd:all>
</xsd:complexType>
```

Más szóval, a *Google*-től visszakapott eredmények (legfeljebb tíz) mindegyike biztosítja az összes olyan információt, amely a *Google* találati oldalával megegyező oldal létrehozásához szükséges. Ezen felül kiválaszthatjuk és meghatározhatjuk a megjeleníteni kívánt elemeket, amelyek például csak a címet és a `dmoz` könyvtárkategóriát illetve címet mutatják. Azt is megtehetjük, hogy a keresett oldal egy rövid részletét jelenítjük meg. Esetleg a felsoroltak közül az összeset. Vagy egyiket sem.

A `doGoogleSearch` nem az egyetlen *WSDL* fájl által leírt eljárás. Számos más eljárás is létezik, amelyek például a *Google* tárolt oldalait kezelik, illetve az egyes szavak helyesírását ellenőrzik. Amikor a webszolgáltatásokat elérhetővé tették a nyilvánosság számára, gyakran említették példaként, hogy a szövegszerkesztők így távoli webszolgáltatást hívhatnak meg a helyesírás-ellenőrzéshez, és nem kell beépített rendszerrel felszerelni azokat. Erre még sokat kell várnunk, de elképzelhető, hogy a *Google API*-ját használjuk egy ilyen szolgáltatás kísérleti változatánál.

Mindezek mellett a kimeneteket bemenetként használhatjuk egy másik webszolgáltatás-hívásnál, helyben és távolról egyaránt. Egyre népszerűbb az a gyakorlat, amelynek során több webhelyről származó adatokat vegyesen alkalmaznak, különösen a *Google* térkép *API*-jai esetében. Bámulatos látni, hogy mi minden történik amikor ilyen módon kombináljuk a szolgáltatásokat – ezt a következő hónapokban vizsgáljuk majd.

Összegzés

Ebben a hónapban röviden áttekintettük a *Google* kereső *API*-ját. Sikerült létrehozni a *Google* keresőoldalának parancssor-változatát, olyan egyszerű eszközök segítségével, mint a *Perl SOAP::Lite* modulja.

A következő hónapokban a *Google* térkép *API*-jával foglalkozunk majd, és kiderül, hogyan készíthetünk vegyes szolgáltatásokat, amelyek többféle adatforrást ötvöznek.

Linux Journal 2006., 145. szám

Reuven M. Lerner régóta web illetve adatbázis tanácsadó, jelenleg az Illinois állambeli Evanstonban található Northwestern University PhD hallgatója, Learning Sciences szakon. A feleségével nemrég ünnepelték Amotz David fiuk születését.

KÓDOK

A cikkben szereplő kódok a ➲ ftp.scc.com/pub/lj/listings/issue145/8866.tgz címen találhatók.

KAPCSOLÓDÓ CÍMEK

➲ www.linuxjournal.com/article/8881

Python kód beágyazása C programokba

C, ismerd meg Pythont.
Python, ő C.

Meglepően kis erőfeszítéssel beépíthetjük a ***Python*** értelmezőt programjainkba, így rendkívül gyorsan valósíthatunk meg olyan funkciókat, amelyek *C* nyelven hónapokat vennének igénybe.
A nagyméretű, nagy teljesítményű alkalmazások nyelve ***Linuxon*** szinte mindig a *C*, esetleg a *C++*. Mindkettő igen hatékony nyelv, segítségükkel gyors, natív, lefordított programokat hozhatunk létre. Futásidejű rugalmasságuk viszont elég csekély. A lefordított kód statikus, ez pedig komoly akadály is lehet. Ha például bővítőmodulokkal akarjuk bővíteni a program funkcionalitását, dinamikus kapcsolási és egyéb problémák sorával kell szembenéznünk. Nem beszélve arról hogy a bővítéshez elengedhetetlen a *C*, vagy a *C++* ismerete, ami nagymértékben szűkíti azon felhasználók körét, akik bővítményeket tudnak írni.
Érdemes tehát valamilyen parancsnyelvet biztosítani, amellyel a bővítmények létrehozhatók.
A parancsnyelvek nagyobb futásidejű rugalmasságot és rövidebb fejlesztési időt biztosítanak, továbbá a tanulási idejük is rövidebb – mindezek biztosítják, hogy több felhasználó tudjon bővítményeket írni.
Egy parancsnyelv létrehozása azonban nem egyszerű feladat, és akár a program fejlesztésének központi eleme is lehet. Szerencsére nem kell saját parancsnyelvet létrehoznunk. A ***Python*** értelmezőt közvetlenül beépíthetjük alkalmazásainkba, így viszonylag kevés többletmunkával a ***Python*** teljes hatékonysága és rugalmassága elérhetővé válik.

A Python beépítése saját alkalmazásunkba

A címben említett feladat nagyon egyszerű. A ***Python Python.h*** nevű fejállományában minden meghatározás megtalálható, ami az értelmező beépítéséhez szükséges. Hatékonyabb lehetne a fordítás, ha a felületnek csak azokat az elemeit kellene beépítenünk, amelyekre szükségünk van, a ***Python*** azonban ezt nem teszi lehetővé. Ha megnézzük a ***Python.h*** fájlt, láthatjuk, hogy fontos makrókat határoz meg és számos fejállományt beemel, amelyekre később az egyes összetevőknek szükségük lesz.
Hogy fordításkor hozzákapcsoljuk a ***Python*** értelmezőt alkalmazásunkhoz, futtatnunk kell a `python-config` programot. Ez kilistázza a kapcsolási opciókat, melyeket át kell adnunk a fordítónak. Nálam ezek az alábbiak:

```
-lpython2.3 -lm -
➥L/usr/lib/python2.3/
➥config
```

Egy egyszerű beágyazott alkalmazás

Mennyi kód kell tehát, hogy a ***Python*** értelmezőt *C* alkalmazásunkból futtathassuk? Igazán kevés. Mint az *1. Lista* mutatja, három sor kód is elegendő – beállítjuk az értelmezőt, átadjuk neki a végrehajtandó Python kódot egy karakterlánc formájában, majd lezárjuk az értelmezőt.
A `Py_Main()` függvénnyel beépíthetünk programunkba egy interaktív ***Python*** terminált is, ahogy az a *2. Listában* látható. Ez ugyanúgy nyit meg egy értelmezőt, mintha a ***Pythont*** közvetlenül a parancssorból futtattuk volna. Az értelmező bezárásakor a vezérlés visszatér az alkalmazásunkhoz.

1. lista A Python beágyazása, mindössze három sor kóddal

```
void exec_pycode(const char*
➥code)
{
  Py_Initialize();
  PyRun_SimpleString(code);
  Py_Finalize();
}
```

2. lista Interaktív Python értelmező beépítése

```
void exec_interactive_
➥interpreter(int arg,
➥char** argv)
{
  Py_Initialize();
  Py_Main(argc, argv);
  Py_Finalize();
}
```

A Python környezet

Három sor kód elég az értelmező beágyazásához, de valljuk be, ***Python***

3. Lista A környezet másolása

```
// Hivatkozást szerzünk a
// main modulra.
PyObject* main_module =
    PyImport_AddModule
    ➥("__main__");
// Hozzáférünk a main modul
// szótárához és másolatot
// készítünk róla.
PyObject* main_dict =
    PyModule_GetDict
    ➥(main_module);
PyObject* main_dict_copy =
    PyDict_Copy(main_dict);
// Két különböző fájlt
// külön környezetben hajtunk
// végre.
FILE* file_1 = fopen
➥("file1.py", "r");
PyRun_File(file_1,
➥"file1.py",
            Py_file_input,
            main_dict,
            ➥main_dict);
FILE* file_2 = fopen
➥("file2.py", "r");
PyRun_File(file_2,
➥"file2.py",
            Py_file_input,
            main_dict_copy,
            ➥main_dict_
            ➥copy);
```

parancsok végrehajtása a programon belül nem túl szórakoztató és nem is igazán hasznos. Ám ez még közel sem minden, amit a ***Python*** kínál. Mielőtt azonban végignéznénk a lehetőségeket, vegyük szemügyre a ***Python*** környezet beállításának folyamatát.

A ***Python*** értelmező futtatásakor a környezet tulajdonságait a `__main__` modul névtér szótára tartalmazza. Minden globális függvény, osztály és változó megtalálható ebben a szótárban. Az interaktív értelmező, vagy egy parancsfájl futtatásakor többnyire nem kell foglalkoznunk ezzel a globális névtérrel. A beágyazott értelmező futtatásakor azonban gyakran kell ehhez a könyvtárhoz fordulnunk végrehajtandó függvényekre való hivatkozásokért, vagy létrehozandó osztályokért. Néha az egész globális könyvtárat másolnunk kell, hogy különböző kódok másmás környezetben futhassanak, például ha minden betöltött bővítménynek új környezetet akarunk biztosítani.

A `__main__` modul szótárának eléréséhez először egy hivatkozásra lesz szükségünk. Ehhez a `PyImport_AddModule()` függvénysegítségével juthatunk hozzá, amely megkeresi a modult a megadott név alapján és visszaad egy `PyObject` mutatót az objektum címével. Miért `PyObject`? Minden ***Python*** adattípus a `PyObject`-től származik, ami tehát a közös nevező. A ***Python*** értelmező legtöbb függvénye tehát nem valamely speciális ***Python*** adattípussal, hanem a `PyObject` objektumokkal dolgozik.

Ha megvan a `__main__` modulra mutató `PyObject`-ünk, a `PyModule_GetDict()` függvény által visszaadott `PyObject` mutatóval férhetünk hozzá a main modul szótárához. Ezután már más ***Python*** parancsok futtatásakor átadhatjuk ezt a hivatkozást.

A *3. Lista* mutatja, hogyan másolhatjuk le a globális környezetet két különböző ***Python*** fájl külön környezetben való futtatásához.

Hamarosan részletesen is bemutatjuk, hogyan működik a `PyRun_File()`, ám előbb nézzük meg a *3. Listát*, ami tartalmaz némi érdekességet. A `PyRun_File()` függvénynek, amellyel a fájlokat futtatjuk, kétszer adjuk át a környe zetet. Az első a globális környezet, amelyről beszéltünk már, a második pedig a helyi, amely a helyileg meghatározott változókat és függvényeket tartalmazza. Ez esetben a kettő ugyanaz, mivel a végrehajtandó kód felsőszintű. Ha egy függvényt dinamikusan akarnánk végrehajtani több ***C*** hívással, valószínűleg érdemesebb lenne egy helyi környezetet létrehozni és használni a globális szótár helyett. Többnyire persze nyugodtan átadhatjuk a globális környezetet globális és helyi paraméterként egyaránt.

Python adatszerkezetek elérése C-ből/C++-ból

Érdemes megfigyelni a *3. Lista* `PyDECREF()` függvényhívásait is. Ezek a memóriakezeléshez nyújtanak segítséget. A ***Python*** értelmező automatikusan gazdálkodik a memóriával, a hivatkozásokat pedig a programozó számára láthatatlanul kezeli. Amint észleli, hogy egy adott memóriaterületre való összes hivatkozás megszűnt, felszabadítja a feleslegessé vált területet. Ez pedig probléma forrása is lehet a ***C*** oldalon, mivel a ***C*** nem kezeli helyettünk a memóriát, ezért ha ***C***-ből hivatkozunk egy adatszerkezetre, a ***Python*** nem lesz képes automatikusan nyilvántartani a hivatkozásokat. A ***C*** alkalmazás bármennyi hivatkozást létrehozhat és tárolhat, anélkül, hogy a ***Python*** tudna róla.

A megoldás az, hogy a ***C*** kód, amely egy ***Python*** adatszerkezetre hivatkozik, maga számolja a hivatkozásokat. Amikor a ***Python*** visszaad egy objektumot a ***C*** programnak, az növeli a hivatkozásszámlálót eggyel. A ***C*** kód így azt tehet az objektummal, amit akar és nem áll fenn a veszélye, hogy az váratlanul törlődik. Ha pedig nincs már vele több dolga, törölheti a hivatkozást a `Py_DECREF()` függvény meghívásával.

Fontos, hogyha a ***C*** programban másolunk egy mutatót, a másolat élettartama pedig hosszabb, mint az eredetié, a hivatkozásszámlálót magunknak kell növelnünk a `Py_INCREF()` meghívásával. Ha például másolatot készítünk egy `PyObject` mutatóról, hogy egy tömbben tároljuk, valószínűleg meg kell hívnunk a `Py_INCREF()`-et, hogy megakadályozzuk, hogy a mutatott objektum a szemétgyűjtő áldozatává váljon az eredeti hivatkozás megszűnése után.

Kód végrehajtása fájlból

Lássunk egy hasznosabb példát arról, hogy hogyan építhetjük be a ***Pythont*** egy igazi alkalmazásba. A *4. Listában* egy kisebb program látható, amely parancssorból vesz át matematikai kifejezéseket, kiszámolja az eredményt, majd megjeleníti azt a kimeneten. Hogy még érdekesebb legyen a dolog, megadhatunk egy fájlt is, benne ***Python*** kódokkal, amelyet a program beolvas, mielőtt a kifejezéseket kiértékelné. Így meghatározhatunk függvé-

4. Lista Egyszerű kifejezéskalkulátor

```
#include <python2.3/Python.h>
void process_expression(char*
➥filename,
                        int num,
                        char**
                        ➥exp)
{
    FILE*        exp_file;
    // Létrehozunk egy
    // globális változót
    // a kifejezés
    // eredményének
    // megjelenítéséhez.
    PyRun_SimpleString("x =
    ➥0");
    // Megnyitjuk és
    // végrehajtjuk a
    // függvényeket
    // tartalmazó fájlt, hogy
    // a függvények
    // elérhetőek legyenek a
    // felhasználói
    // kifejezések számára.
    exp_file = fopen
    ➥(filename, "r");
    PyRun_SimpleFile
    ➥(exp_file, exp);
    // Végighaladunk a
    // kifejezéseken és
    // végrehajtjuk őket.
    while(num-) {
        PyRun_SimpleString
        ➥(*exp++);
        PyRun_SimpleString
        ➥("print x");
    }
}
int main(int argc, char**
➥argv)
{
    Py_Initialize();
    if(argc != 3) {
        printf("Usage: %s
        ➥FILENAME
        ➥EXPRESSION+\n");
        return 1;
    }
    process_expression
    ➥(argv[1], argc - 1,
    ➥argv + 2);
    return 0;
}
```

5. Lista Meghívható függvényhivatkozások

```
#include <python2.3/Python.h>
void process_expression(int
➥num, char* func_name)
{
    FILE*        exp_file;
    PyObject*    main_module,
    ➥* global_dict, *
    ➥expression;
    // Beállítunk egy
    // globális változót
    // a kifejezés
    // eredményének
    // megjelenítéséhez.
    PyRun_SimpleString("x =
    ➥0");
    // Megnyitjuk és
    // végrehajtjuk a Python
    // fájlt.
    exp_file = fopen(exp,
    ➥"r");
    PyRun_SimpleFile
    ➥(exp_file, exp);
    // Hivatkozást szerzünk
    // a main modulra és
    // a globális szótárra.
    main_module =
    ➥PyImport_AddModule
    ➥("__main__");
    global_dict =
    ➥PyModule_GetDict
    ➥(main_module);
    // Kinyerjük a
    // "func_name" függvényre
    // való hivatkozást
    // a globális szótárból.
    expression =
        PyDict_GetItemString
        ➥(global_dict,
        ➥func_name);
    while(num-) {
        // Meghívjuk az
        // "expression"
        // által hivatkozott
        // függvényt.
        PyObject_CallObject
        ➥(expression,
        ➥NULL);
    }
    PyRun_SimpleString("print
    ➥x");
}
```

nyeket, amelyek a parancssorból átadott kifejezésekből is elérhetők. Két alapvető ***Python API*** függvényt használtunk ebben a programban, a `PyRun_SimpleString()`-et és a `PyRun_AnyFile()`-t.
A `PyRun_SimpleString()`-gel már találkoztunk. Mindössze annyit csinál, hogy végrehajtja a megadott ***Python*** kifejezést a globális környezetben.
A `PyRun_SimpleFile()` hasonlít a korábban bemutatott `PyRun_File()` függvényhez, kivéve, hogy alapból a globális környezetben fut, így minden általa végrehajtott kifejezés, vagy kifejezéscsoport elérhető a később végrehajtandók számára is.

Meghívható függvényobjektumok

Kifejezés-kalkulátorunkkal nem csak kifejezések sorozatát értékelhetjük ki, de akár betölthetünk mondjuk egy `f()` függvényt is a megadott ***Python*** fájlból, és azt a parancssorban átadott számnak megfelelően többször végrehajtva a kifejezések összegét is kiszámolhatjuk. A függvényt a `PyRun_SimpleString("f()")` függvénnyel hajthatjuk végre, bár ez nem túl hatékony, mivel az értelmezőnek így be kell olvasnia és ki kell értékelnie a karakterláncot a függvény minden meghívásakor. Jobb lenne hivatkozni a függvényre, hogy közvetlenül meghívhassuk.
A ***Python*** az összes globálisan meghatározott függvényt a globális szótárban tárolja. Ha van egy hivatkozásunk erre a szótárra, onnan hivatkozhatunk bármely függvényre is.
A ***Python API*** kínál is függvényeket erre a célra. Működésük a *5. Listában* látható.
Hogy megkapjuk a függvényhivatkozást, a program először létrehoz egy hivatkozást a main modulra, „importálva” azt a `PyImport_AddModule("__main__")` függvénnyel. Ha ez megvan, a `PyModule_GetDict()` függvényhívással megszerzi a szótárát. Innentől csak a `PyDict_GetItemString(global_dict, "f")` hívásra van szükségünk, hogy a szótárból megkapjuk magát a függvényt.
Most, hogy megvan a hivatkozás a függvényre, meghívhatjuk a `PyObject_CallObject()` függvénnyel. Mint látható, egy mutatót

vesz át a meghívandó függvényre. A függvény már létezik a *Python* környezetben, így le is van fordítva. Meghívásakor tehát nincs beolvasás, vagy fordítás, így a függvény elég gyorsan végrehajtható.

Adatok átadása a függvényeknek

Most nyilván arra gondolunk, hogy ***„jaj, de jó, de mennyivel jobb lenne, ha ezeknek a függvényeknek adatokat is át tudnánk adni”***. És tudunk is. Egyik megoldás a titokzatos `NULL` érték használata - ezt adtuk át az *5. Listában* a `PyObject_CallObject()` függvénynek. Röviden bemutatjuk, hogyan is működik, de előbb lássunk egy sokkal egyszerűbb módot a függvények paraméterezésére *C*/*C++* adattípusok és a `PyObject_CallFunction()` segítségével. Ennek a függvénynek, ahelyett, hogy a *C* és *Python* közötti átalakítással kellene törődnünk, átadhatunk egy formátum karakterláncot és egy változó hosszúságú paraméterlistát, hasonlóan a `printf()` függvénycsaládhoz.
Visszatérve a kifejezés-kalkulátorunkhoz, tegyük fel, hogy egy nem folytonos értékek tartományból álló kifejezést akarunk kiértékelni.
Ha a kifejezést egy, a betöltött *Python* fájlban levő függvényben határoztuk meg, szerezhetünk rá hivatkozást a szokásos módon, majd végighaladhatunk a tartományon.
Minden értéknél hívjuk meg a `PyObject_CallFunction(expression, "i", num)`-ot. Az `"i"` karakterlánc azt jelzi, hogy egyetlen egész számot akarunk átadni. Ha a függvényünk két egész számot és egy karakterláncot venne át, ily módon hívnánk meg:
`PyObject_CallFunction (expression, "iis", num1, num2, string)`. Az esetleges visszatérési értéket a `PyObject_CallFunction()` függvény visszatérési értékeként egy `PyObject` mutató formájában kapjuk meg.
Python függvényeknek legegyszerűbben így adhatunk át argumentumokat, ám a legrugalmasabb megoldás nem ez. Gondoljunk csak bele, mi van, ha dinamikusan választjuk ki a meghívandó függvényt.
A nehézséget ekkor az okozza, hogy a különböző függvények más-más paramétereket várnak.
A `PyObject_CallFunction()`-nek viszont már a fordításkor meg kell adnunk az argumentumokat, ez pedig nem az a rugalmasság, amit egy parancsnyelvtől elvárnánk.
A megoldás kulcsa a `PyObject_CallObject()`. Ez a függvény beéri egyetlen *Python* objektumokból álló *tuple*-lel (tetszőleges, nem módosítható elemeket tartalmazó szekvencia) a változó hosszúságú *C* értékekből álló paraméterlista helyett. Hátránya viszont, hogy a natív *C* értékeket *Python* objektumokká kell alakítanunk, ám amennyit veszítünk sebességben, annyit nyerünk rugalmasságban. Persze mielőtt *Python* objektumok *tuple*-jeként átadnánk az értékeket, tudnunk kell, hogyan hozzuk létre ezt a *tuple*-t. Erről szólunk a következő részben.

Átalakítás Python és C adattípusok között

A *Python* értelmező `PyObject` objektumok formájában vesz át és ad vissza adatszerkezeteket. Ahhoz, hogy a megfelelő típust kapjuk, típusátalakítást kell végrehajtanunk. Átalakíthatunk például egy `PyObject` objektumot `PyIntObject()` objektummá. Ha nem ismerjük azonban a változó típusát, az átalakításnak katasztrofális következményei is lehetnek. Ilyen esetben meghívhatjuk valamelyik `Check()` függvényt, hogy megállapítsuk, az objektum típusa megfelelő-e. A `PyFloat_Check()` például igaz (true) értékkel tér vissza, ha az objektum átalakítható `float` (lebegőpontos) típusúvá. Más szóval igazat ad, ha az objektum `float`, vagy annak altípusa. Ha a pontos egyezést akarjuk vizsgálni, a `PyFloat_CheckExact()` függvényt használhatjuk.
A homályos `PyObject` adatszerkezet nem túl hasznos egy *C* program számára. Ahhoz, hogy a *Python* adatait elérhessük a programunkban, különböző átalakító függvények állnak rendelkezésre, amelyek natív *C* adatokat adnak vissza. Ha egy `PyObject` objektumot `long int`-té (duplapontos egész) akarunk alakítani, csak meg kell hívnunk a `PyInt_AsLong()` függvényt. A `PyInt_AsLong()` biztonságos függvény, ellenőrzött típusátalakítást fog végezni `PyIntObject`-té, mielőtt visszaadná a `long int` értéket.
Ha biztosan tudjuk, hogy az átalakítandó érték valóban int, felesleges lehet az ellenőrzés, különösen egy ciklus belsejében.
A *Python* függvények gyakran vesznek át *Python* objektumokat *tuple*-k, vagy listák formájában. Ezeknek a típusoknak nincsen *C* megfelelőjük, de a *Python* biztosít függvényeket, amelyekkel létrehozhatók *C* adattípusokból. Lássuk most egy *tuple* létrehozását - erre szükségünk lesz, ha a `PyObject_CallObject()`-tel akarunk függvényeket meghívni.
Az első lépés egy új *tuple* létrehozása a `PyTuple_New()` függvénnyel, ami átveszi a *tuple* hosszát és visszaad egy, az új *tuple*-t jelölő `PyObject` mutatót. Ezután a `PyTuple_SetItem()`-mel értéket adhatunk az egyes elemeknek, minden elemet egy `PyObject` mutatóként megadva.

Összefoglalás

Mostanra elegendő alapismeretre tettünk szert, hogy elkezdhessünk *Python* szkripteket használni saját programjainkban. Lásd még a *Python* dokumentációjának ***„Extending and Embedding the Python Interpreter”*** (***A Python értelmező bővítése és beágyazása***) című részét, ami további információval szolgál, de bemutatja a másik irányt is, vagyis a *C* függvények beépítését Pythonba. Hasznos forrás még a ***„Python/C API Reference Manual”*** (***Python/C API Referencia Kézikönyv***), amely részletes leírást tartalmaz a *Python* beágyazásához használható függvényekhez. A *LinuxJournal* archívumában megtalálható *Ivan Pulleyn* kitűnő cikke a *Pythont* beágyazó többszálú programokról.

Linux Journal 2006., 142. szám

William Nagel

Vezető szoftvertervező egy kis szoftverfejlesztő cégnél, ahol Linux alapú real-time rendszereket fejleszt. Ő a szerzője a „SubversionVersion Control: Using the Subversion Version Control System in Development Projects” című könyvnek.

KAPCSOLÓDÓ CÍMEK

➲ www.linuxjournal.com/article/8714

A $HOME virtuális utcái

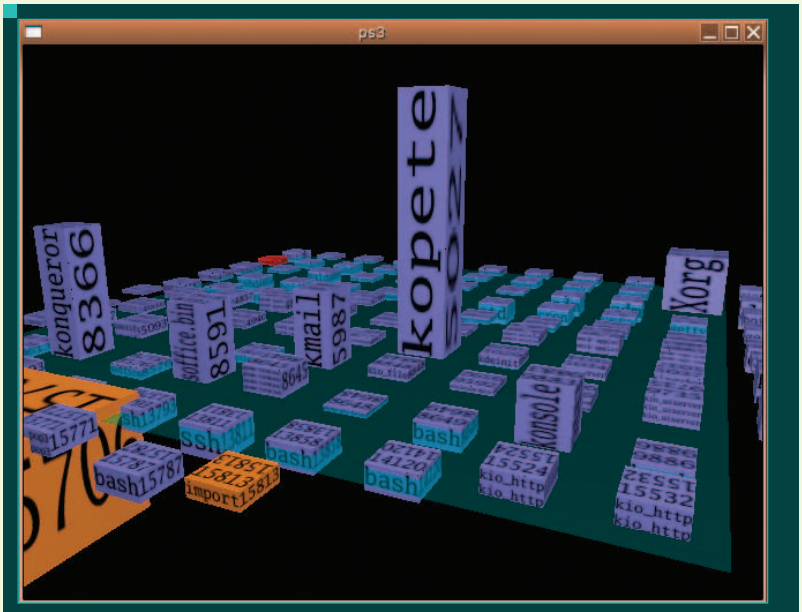
Avagy miként látogassunk meg a Linux segítségével virtuális konzolokat, városokat és egyéb csatatereket...

Mit keresel a ***Freshmeat***-en ***Francois***? Hogyan? Egy programot, ami digitalizál téged, hogy be tudj lépni a számítógépedbe? Igen, én is pontosan tudom, hogyan néz ki az ilyesmi a filmekben, de ki kell hogy ábrándítsalak: a virtuális valóság megvalósítása még elég messze van a késztől. Az igazat megvallva azt hittem erre sikerült kellőképpen rávilágítanom pár hónapja, amikor a világítási ciklusokról volt szó. Na, szóval kedve ***Francois***, én nem gondolom, hogy az emberek a közeljövőben számítógépekben fognak élni. Nem, kedvesem, nem nevetlek ki. Csak úgy elcsodálkoztam ezen ez egészen. Na, szóval tényleg sajnálom, hogy ki kell ábrándítsalak, de nem hiszem, hogy vannak városok meg emberek a ***Linux*** rendszereden. De ezt azt hiszem később is megbeszélhetjük. Most viszont csipkedjük magunkat mert a vendégeink perceken belül itt lesznek, nekünk pedig el kell készülnünk addig.
Mint mondasz kérlek? Hogy máris itt vannak? Hát akkor futás ***Francois***, irányítsd vendégeinket az asztalukhoz.

Isten hozott mindenkit ***Marcel*** éttermében, ahol a finom borok a kivételes ***Linux*** alkalmazásokkal találkoznak, szeretett vendégeink legnagyobb megelégedésére. Te pedig ***Francois*** ha befejezted a vendégek ültetését, akkor irány a borospince és hozd fel nekünk azt a 2002-es ***Chates du Roussillon Villeges***-t.
Francois-val épp a számítógépek belsejében megvalósítható virtuális világok lehetőségéről beszélgettünk, amely – szerintem – egyelőre csak a fantázia szüleménye. Az persze igaz, hogy fantasztikus haladás történt azóta, amióta először láttam számítógépet. Ezek egyike pedig éppen az Önök számítógépén levő ***Linux*** operációs rendszer. És mivel a ***Linux*** egy nyílt rendszer, segítségével szabadon felfedezhetünk olyan dolgokat, amelyek máshol egyszerűen nem is léteznek. Jómagam úgy gondolom, hogy a számítástechnika úgy egészében véve még mindig gyerekcipőben jár. Persze az is lehet, hogy egyszerűen csak túl sok tudományos-fantasztikus filmet láttam, és ettől irreálisan magasak az elvárásaim. Emlékezzünk csak vissza például a ***Tron*** című filmre.
A nyitójelenetben ***Flynn***, a főhős beküld a rendszerbe egy ***CLU*** nevű programot, amivel hiányzó fájlokat szeretne megtalálni. ***CLU***, a program ugyanúgy néz ki, mint ***Flynn*** maga, egy ***3D*** tankkal halad, miközben manapság egy attribútumbit csak igent vagy nemet képes tárolni információ gyanánt. ***Flynn*** körül meg ehhez képest még mindenféle felhőkarcolóra emlékeztető toronyházak meg efféle szerkezetek is voltak navigáció közben. Nos, ez az a virtuális világ, amit fiatal koromban szerettem volna megtapasztalni.
Ah, úgy látom ***Farancois*** visszajött a pincéből a borral! Kérlek tölts a vendégeinknek. Apropó had javasoljam mindenkinek, hogy próbáljon meg ennek az igen kiváló vörösbornak a rejtett ízeire ráérezni.
Bár rendszerünkben nincsenek rejtett világok, azért számos olyan dolog van benne, ami az avatatlan felhasználó szeme elől rejtve marad. A virtuális konzolok például számos olyan információt jelenítenek, meg, amit nem láthatunk, ha egyszer a grafikus felület elindult. Persze ott van a ***Ctrl-Alt-F1*** kombináció, amivel kiléphetünk a grafikus munkamenetből, de van erre jobb módszer is. Ha látni akarjuk egy virtuális konzolon megjelenő rejtett tartalmat, adjuk ki a következő parancsot egy héjban:

```
cat /dev/vcs1
```

Látható, hogy bár mi magunk eddig talán nem is tudtunk róla, a rendszerünk



1. ábra Ha a rendszeren futó folyamatokat a ps3-mal jelenítjük meg, úgy érezhetjük magunkat, mintha egy virtuális város utcáin barangolnánk.

azért számos információt nyilvántart a virtuális konzolokkal (ezek 1-től 6-ig számozódnak) kapcsolatban. Minden egyes konzolnak egy-egy */dev/vcsX* nevű eszközfájl felel meg, ahol *X* a konzol sorszáma. Az én Ubuntu tesztrendszeremen például az első konzol tartalma ebben a pillanatban a következő:

```
* Starting OpenBSD Secure Shell
➥server...
[ ok ]  * Starting Bluetooth
➥services...  hcid sdpd
➥[ ok ]  * Starting RAID
➥monitoring services...
[ ok ]  * Starting
➥anac(h)ronistic cron: anacron
[ ok ]  * Starting deferred
➥execution scheduler...
[ ok ]  * Starting periodic
➥command scheduler...
[ ok ]  * Checking battery
➥state...
[ ok ]  * Starting TiMidity++
➥ALSA midi emulation...
[ ok ]
Ubuntu 6.04 "Dapper Drake"
➥Development Branch francois
➥tty1
```

Nos, bár ezek itt meglehetősen érdekes információk, azért a „virtuális világ” fogalmát meg sem közelítik, sőt attól tartok azt a bizonyos Wow! faktort sem érik el, amit kedves pincérünk, ***Francois*** annyira kereset. No, de függetlenül attól, amit ***Francois***-nak mondtam, a szó bizonyos értelmében mégiscsak vannak olyan módszerek, amelyekkel egész városokat fedezhetünk fel Linux rendszerünkön belül. Kicsit erőltetett ugyan a dolog, de léteznek – egyelőre erősen kísérleti jelleggel – olyan alkalmazások, amelyek a rendszeren futó folyamatokat, a memóriát és persze a programokat amolyan valóságosnak tűnő virtuális objektumokként jelenítik meg. Közülük az egy a ***Rudolf Hersen*** által fejlesztett ***ps3*** (lásd az on-line információforrásokat), amelynek a használatához egy ***3D*** gyorsítással is rendelkező videókártyára lesz szükségünk.
A program lefordítása egészen egyszerű. Tulajdonképpen csak annyi megjegyzést érdemes hozzáfűzni, hogy a végrehajtáshoz szükségünk lesz az ***SDL*** fejlesztői könyvtárakra. Ha ezek fönt vannak, akkor a futtatható állomány előállítása csupán a következő parancsokból áll:

```
tar -xjvf ps3-0.3.0.tar.bz2
cd ps3-0.3.0
make
```

A program futtatásához a parancssorból adjuk ki a `./ps3` parancsot (ehhez ugyanabban a könyvtárban kell lennünk, ahol a bináris állomány keletkezett). A nyitóképen a futó folyamatok ***3D*** reprezentációját láthatjuk. Elképzelhető, hogy induláskor nem a leghasználhatóbb látvány fogad bennünket, de semmi gond: a ps3-nak pont az a lényege, hogy a ***3D*** látványt szabadon átszabhatjuk. A képet elforgathatjuk mindhárom tengely körül, így a folyamattáblát felülről sőt alulról is megszemlélhetjük. Ha a folyamatoknak megfelelő „épületek” túl magasak, egyszerűen állítsuk át a megjelenítési skálát valamilyen kényelmesebb üzemmódra. Valamennyi folyamat mellet megtaláljuk a hozzá tartozó program nevét, és persze a folyamatazonosítót is.
A ***ps3*** képernyőjén való mozgáshoz elegendő csak az egeret használni. Kattintsunk a bal egérgombbal és nyomva tartva húzzuk az egeret a kép elforgatásához. Beállíthatjuk vele a magasságot, és a vízszintes forgatás sebességét is. A jobb egérgombot nyomva tartva az egérrel a képet függőleges és vízszintes irányban egyaránt elforgathatjuk. Az egér görgőjével ráközelíthetünk a képre, vagy térben távolodhatunk tőle.
Ha ki akarunk lépni a ***ps3***-ból, nyomjuk meg a *Q* billentyűt.
Ami azt illeti, a rendszeren futó folyamatoknak a ***ps3*** által generált megjelenítése nem különösebben tudományos igényű, ellenben annál fantasztikusabb, mi több, szemet gyönyörködtetően szórakoztató. Szóval ha akarjuk, mégiscsak vannak épületek meg virtuális városok valahol mélyen a gépünkben. Hát, akkor már csak egy dolog hiányzik: a tank. Ami azt illeti, erre is van ám ötletem. Úgy hívják hogy ***BZFlag***, de mielőtt belevágnék, azt hiszem jó lesz, ha ***Fancois*** újratölti a poharainkat. Persze csak ha Önök is egyetértenek hölgyeim és uraim.
A ***BZFlag*** nem más, mint egy többfel-

Megjegyzés

Ha valakinek nem ugrana be, miről is szólt az a bizonyos csatározásokkal kapcsolatos cikki, keresse fel a ➲ ***www.linuxjournal.com/article/6638*** webhelyet, ahol elolvashatja ***Marcel „Csaták a számítógép belsejében”*** című cikkét.

2. ábra Bármilyen időpontban próbálunk is csatlakozni, több tucat aktív BZFlag kiszolgálót látunk majd, amelyeken összességében emberek százai játszanak éppen.

3. ábra A Join képernyőn beállíthatjuk a becenevünket (callsign), illetve hogy melyik csapathoz szeretnénk tartozni.

használós *3D* tankcsata-játék, amit az internet keresztül játszhatunk másokkal, avagy mások ellen. (A ***BZFlag*** kódját jelenleg ***Tim Riker*** tartja karban, az eredeti szerző azonban ***Chris Schoeneman***). A ***BZFlag*** név különben egy rövidítés, aminek a visszafejtése ***Battle Zone capture Flag*** (kb. szerezd meg a zászlót a harcmezőn). A névválasztás persze nem véletlen, a játék lényege tényleg egy zászló megszerzése. Ha valakinek kedve szottyant kipróbálni, nem kell messzire menni érte. Tekintettel arra, hogy a játék népszerűsége egyre nő, a legtöbb terjesztés telepítőanyaga eleve tartalmazza. Ez a módszer persze csak a kezdőknek való. Aki a legfrissebb változatra vágyik, az látogasson el a ***BZFlag*** webhelyére (lásd az on-line forrásokat). Itt találhatunk bináris csomagokat, forráskódot, sőt, még más operációs rendszerekre lefordított kódot is, hogy mindenkit bevonhassunk a játékba, digitális felekezetre való tekintet nélkül.

Hacsak másként nem rendelkezünk a ***BZFlag*** teljes képernyős üzemmódban indul. Aki ez zavar, az a `-window` kapcsolóval felülbírálhatja ezt az alapértelmezett viselkedést. A játék a ***Join Game*** (csatlakozás a játékhoz) képernyővel indul. Mielőtt kiválasztanánk egy megfelelő játékkiszolgálót (az első pont a listában), változtassuk meg a ***Callsign*** (becenév) mező tartalmát. A többi beállítást ráérünk azután megadni, hogy kapcsolódtunk a szerverhez. Most, hogy van egy jól hangzó nevünk, vigyük a kurzort a ***Find Server*** fülre, és üssük le az ***Entert***.

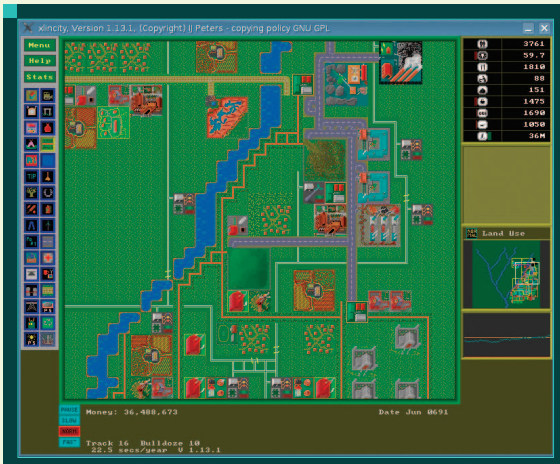
A játékostársak megtalálása nem fog gondot okozni, ebben a műfajban ismeretlen a pangás. Kapunk egy listát, amelyen az éppen elérhető játékkiszolgálókat látjuk (*2. ábra*). Nézzük végig a neveket, és válasszunk ki egy olyant, ami leginkább tetszik. A szempontok sokfélék lehetnek. Érdekes paraméter az aktív játékosok száma, a kiszolgáló foglaltsága és az, hogy hány csapat játszik éppen az adott csatatéren. A szerver megválasztásakor mindenképpen figyeljünk arra, hogy milyen típusú játék játszható azon a helyen. Egyeseken ugyanis a csapat alapú játékot részesítik előnyben, míg máshol inkább a szabad stílus dívik. Előfordulhat az is, hogy csak adott számú lövésünk van, szóval nem árt az óvatosság.

Ha kiválasztottuk a nekünk megfelelő játékszervert, üssük le az ***Entert***. Ekkor visszakerülünk a ***Join*** képernyőre, de már ki lesz jelölve egy kiszolgáló (*3. ábra*). Ekkor akár el is kezdhetünk játszani, de előbb talán mégis érdemes néhány apróbb beállítást megejteni, és csak aztán bepöccenteni a tank motorját. Először is menjünk el a Team (csapat) címkére, és nyomjuk meg a billentyűzeten jobbra vagy a balra mutató nyilat. Alapértelmezésként mindenképpen bekerülünk egy csapatba, de a program választását itt felülbírálhatjuk. Ahelyett, hogy csatlakoznánk egy csapathoz, választhatjuk a megfigyelői (***Observer***) pozíciót is. Ez nem rossz ötlet azoknak, aki még soha nem játszották a játékot, mivel így lehetőségünk nyílik arra, hogy békés szemlélődőként figyeljük, miként bánnak (el) a többiek egymással, vagy éppen önmagukkal.

A ***Join*** képernyőn akár kézzel is megadhatjuk egy kiszolgáló nevét, vagyis a gyakorlottaknak nem muszáj a listában keresgélni. Akkor is hasznos az a funkció, ha privát csatateret üzemeltetünk egy helyi hálózaton. Ha már az üzemeltetésnél tartunk, gondolom mindenki felfigyelt már a lista alján szereplő ***Start Server*** (kiszolgáló indítása) pontra. Most pedig hajrá, csatlakozzunk a játékhoz. Menjünk vissza a képernyő tetején található ***Connect*** pontra, és üssük le az ***Entert***.

Remélem mindenki felkészült lélekben, mert a csata pillanatokon belül kezdődik, és néhány játékostársunk bizony igencsak gyakorlott. A saját tankunkat az egérre tudjuk irányítani, a tüzeléshez pedig kattintsunk a bal egérgombbal. Ügyeljünk rá, hogy a tank igen mozgékony jószág, sőt egyes játékokban még ugrani is tud (ez utóbbira a Tab gomb megnyomásával lehet rávenni). Apropó ha bármikor látni szeretnénk az összes lehetséges billentyűparancs jelentését, nyomjuk meg az sc billentyűt és válasszuk a ***Help*** pontot. Játék közben a ***BZFlag*** egy meglehetősen részletes összefoglaló képernyőt is megjelenít, amelyen mindenféle statisztikákat láthatunk. Leolvashatjuk róla például, hogy melyik játékos milyen statisztikákkal

4. ábra A csata gyors és heves. Tankok robbannak mindenütt, ahova csak nézünk. Szóval csak óvatosan, nehogy a miénk legyen a következő.



5. ábra A Lincity egy számítógépes városszimulátor, ami segít az embernek megérteni, mitől is olyan nehéz létrehozni Utópiát.

rendelkezik, hány embert ölt meg, hány személyi pontot gyűjtött, hány pontja van a csapatának, és így tovább (*4. ábra*). Aztán az sem árt, ha egyik szemünket mindig a bal alsó sarokban látható térképen tartjuk, mivel itt lehet először fölfedezni, ha ellenség közelít. Namármost ha valaki képes egyszerre vezetni, lőni és gépelni, akkor nyomja meg az N gombot, amivel szöveges üzenetet küldhet a csoport többi tagjának. Az ***M*** billentyű jelentése hasonló, de ezzel csak a csapattagokkal kommunikálhatunk. Ha pedig azt látjuk, hogy közelít a munkahelyi főnökünk, akkor villámgyorsan nyomjuk meg az ***F12***-őt, amivel azonnal kiléphetünk a játékból. Persze most csak vicceltem. Ha bárki kérdezi, nem én mondtam, hogy a munkahelyen is lehet játszani.

Nos, kedveseim, az idő már megint gyorsan elszállt, de nem akarom, hogy azzal az érzéssel lépjen ki innen bárki, hogy a rendszereinkben teremtett virtuális világok mindegyike csak a pusztításra épülő pokol. Aki építeni szeret, az létrehozhat egy egész virtuális civilizációt, városokkal, farmokkal, gyárakkal, piacokkal és minden olyasmivel, ami egy modern vagy premodern civilizációnak része lehet. Ehhez nem kell mást tennünk, csak letölteni a ***Lincity*** nevű játékot, és elkezdeni az építkezést. (Ezt a programot is érdemes először az általunk használt terjesztés telepítőlemezein keresni.) Ennek a megkapó, és meglehetősen időigényes játéknak az alapötlete az, hogy fel kell építenünk egy várost, majd működtetnünk is kell azt, úgy, hogy etetjük és ruházzuk a lakóit. Munkahelyeket kell teremtenünk, hogy építkezni tudjunk, vagyis végső soron egy egész virtuális gazdaságot irányítunk. Az is jó ötlet, ha megújuló energiaforrásokba ruházunk be, különben nem jön össze a nagy ***Utópia*** (***5. ábra***). Ahogy aztán a dolgok egyre jobban és jobban állnak, természetesen el is menthetjük a játék aktuális állapotát, és később folytathatjuk az ideális világ építgetését. Nos, rendben, elismerem, hogy a dolog azért nem olyan egyszerű, mint ahogy kimondja az ember. Az óra ketyeg, a hónapok pedig gyorsan leperegnek. Ha nem figyelünk, a világunk szép lassan megölheti magát a maga termelte mérgekkel. Aki a semmiből kezdi el építeni a tökéletes világot, az gyakorta akár pusztítani is kezdhetne, legalábbis a végeredményt szemlélve. Na de miért is ne kezdenénk a dolgot valamilyen ideális állapotból? A játék indítása után kattintsunk a bal felső sarokban látható ***Menu*** gombra. A kinyíló főablakban láthatunk néhány érdekes funkciót. Többek között itt lehet elmenteni a játék állapotát, illetve betölteni egy régi állást. Két alaphelyzetet maga a frissen telepített játék is tartalmaz. Az egyik a ***Good Times*** (szép idők) a másikat meg gondolom mindenki kitalálná: ez a ***Bad Times*** (rossz idők). Én az elsőt javaslom annak, aki most kezdi. Aztán ha már úgy érezzük, hogy annyira beletanultunk az építkezésbe, hogy bármit képesek vagyunk megjavítani, akkor kipróbálhatjuk a másikat is. Szerintem nem kis feladat kihúzni a kátyúból egy 25 millió dollárra eladósodott várost...

Nos, kedves vendégeim, az óra szerint lassan zárnunk kell. Viszont tekintettel a nehéztüzérség egyre fokozódó akcióira, meg a munkaállomások hangszóróiból áradó robbanások gyakoriságára azt hiszem, ezúttal kicsit még nyitva tarthatunk... ***Francois*** még egyszer készségesen újratölti a poharakat, mielőtt búcsút mondunk egymásnak, és a csatatérnek. Figyelem, a játék virtuális ugyan, a bor viszont valódi, és nem szeretném mindannyiszor kilöttyenteni, valahányszor valaki elsüti az ágyút. Emeljük tehát poharainkat, és igyunk inkább egymás egészségére! Bon appetit!

Linux Journal 2006., 145. szám

Marcel Gagné
(mggagne@salmar.com)
Mississaguában, Ontario államban él.
Ő a szerzője a Kiskapu kiadásában tavaly szeptemberben megjelent Linux-rendszerfelügyelet (ISBN 96-9301-40) című könyvnek.

KAPCSOLÓDÓ CÍMEK

➲ www.linuxjournal.com/article/8882

Felhasználói Módú Linux

A Felhasználói Módú Linux olyan trükköket tesz lehetővé, mint egy biztonságosan elszigetelt Debian 3.1 futtatása Fedora 4 felett.

A számítógépek programokat futtatnak. A kezdetek irdatlan monstrumaitól kezdve, melyekbe bele volt huzalozva a program, egészen napjaink asztali és hordozható gépeiig minden számítógép valamilyen programot futtat. ***Neumann János*** rukkolt ki azzal az elvvel, hogy tároljuk a programokat is ugyanúgy, mint bármely más kis adathalmazt, és a beledrótozott funkciók zártságával ellentétben tegyük lehetővé a számítógépek többcélúvá válását. Hamarosan megszületett az operációs rendszer fogalma: egy olyan programé, amely a közös rendszerszintű részletek számára hoz létre egy elvonatkoztatási réteget, mint amilyen pl. az eszközök kezelése vagy a programok végrehajtása. Nem kellett sok idő, hogy néhány ravasz rendszerprogramozó tudatosítsa, hogy egyetlen processzor látszólag egyetlen pillanatban végrehajthat több feladatot is. Ez adott lendületet az első időosztásos és többfeladatos/többfelhasználós operációs rendszerek kialakulásához. Az összes modern számítógép is ugyanezen a „tárolt program" elven működik. Egy mai számítógép bekapcsolásakor először a ***BIOS***-ban tárolt programokat futtatja (vagy a firmware-t), amely aztán átadja a stafétabotot a többlépéses betöltéskezelőnek, amely végül betölti az operációs rendszer magját. A rendszermag hajtja végre és állítja be az operációs környezetet, amelyben a rendszer erőforrásai (mint pl. a processzoridő, memória és az eszközök) használhatóak a programok által, melyek a kernel fölött elindulnak. Mindez tárolt programok hosszú láncolata.

A kernel is olyan program, mint bármelyik másik (jóllehet meglehetősen összetett, szép példány). Így hát mi akadályozhatna meg abban, hogy úgy futtassuk a rendszermagot, mint bármely más programot? Végül is semmi. Pontosan erről szól a ***Felhasználói Módú Linux*** (***UML***). A ***Linux*** kernel általában speciális jogosultságokkal fut, mert szükség van arra, hogy direkt módon el tudja érni a hardvert. A ***Felhasználói Módú Linux*** lehetőséget ad arra, hogy úgy fordítsuk le a normál ***Linux*** kernelforrást, hogy egyszerű bináris programként lehessen elindítani a gazda ***Linux*** kernel felett. Amikor egy kernelt egy másik, „gazda ***Linux*** kernel" felett indítunk el, akkor igazából (egy vagy több) „vendég" ***Linux*** rendszert futtatunk, speciális jogosultságok nélkül. (Vannak azért kivételek – néhány programot rendszergazdaként kell telepíteni a ***Felhasználói Módú Linux*** számára, hogy jól működjenek).
Ezek a ***Linux*** vendégrendszerek teljes operációs rendszerek, melyek egy (szinte) teljesen biztonságos környezetben futnak.

Hogyan kell megcsinálni?

Cikkem hátralevő részében erre adok receptet: hogyan kell lábra állítani egy ***UML*** rendszert ***Linuxot*** futtató gazdagépünkön. Ezek után körbejárjuk néhány tulajdonságát és szórakozunk egy kicsit. A bemutatóhoz ***Fedora Core 4*** gazdagépet használok, ***Intel P4***-en 1 GB memóriával – azonban a módszer szinte bármilyen rendszeren és disztribúción működik, amennyiben friss 2.6-os kernel fut rajta, és van legalább 256MB memóriája.

Egy vendég ***UML*** rendszer olyan, mint bármilyen más ***Linux*** rendszer. A „***Linux***" a kernelen kívül sajátos gyűjteménye kisebb programoknak, programkönyvtáraknak és megfelelően szervezett fájloknak, melyek lehetővé teszik az operációs rendszer rendeltetésszerű működését. Ezek két külön egységben állnak rendelkezésünkre: a kernel és a fájlrendszerképmásfájl formájában – ez utóbbi nem más, mint egy virtuális merevlemez-partíció; ezt csatoljuk fel és használjuk majd ***UML*** rendszerünk gyökér fájlrendszereként. Megválaszthatjuk, hogy előállítjuk-e ezt a két összetevőt, vagy letöltjük az internetről, ahol a népszerűbb disztribúciók számára már előre elkészítették ezeket. A gyorsabb siker érdekében most nem hagyjuk el a járt utat a járatlanért. Mindazonáltal vessünk egy pillantást az ***UML Wikire*** is; megtudhatjuk, hogy hogyan készíthetjük el fájlrendszereinket saját magunk (lásd a cikkhez tartozó forrásokat).
A rendszermagoknak és gyökér fájlrendszereknek számos disztribúcióhoz elkészített változatai elérhetőek; például ***Red Hat***, ***Fedora Core***, ***Debian*** és számos más speciális célú képmásfájl vár letöltésre. A bemutatóhoz most ***Debian 3.1***-est fogok használni.

A tehenek (COWs) a barátaid

A ***Felhasználói Módú Linux*** üzemszerű működtetésének van egy sajátos lehetősége, a ***Copy-On-Write*** (***COW***; „írásnál másold") fájl használata. Az „írásnál másold" egy elterjedt számítástechnikai mechanizmus, amely lehetővé teszi, hogy egy nagy sereg

adat megmaradhasson csak-olvasható állapotban, azonban a változtatás előnyeit sem kell feladni: ezek egy másik helyen levő adathalmazban vannak nyilvántartva. A letöltött fájlrendszer-képmásfájl megmaradhat csak-olvashatónak; az *UML* rendszerünkben elkövetett változtatások csak a *COW* fájlban lesznek rögzítve. Ez arra is lehetőséget nyújt, hogy egyszerre több *UML* rendszert állítsunk lábra ugyanabból a (csak-olvasható) fájlrendszer-képmásfájlból, feltéve, hogy mindegyikhez saját *COW* fájl tartozik. Egy másik előny: ha az *UML* rendszerünk elromlik, egyszerűen törlendő a *COW* fájl, és minden kezdhető elölről. A *COW* fájl tipikus példája a „szellős" (*sparse*) fájloknak; bár a mérete ránézésre (amit az `ls -l` mutat) óriási, tényleges merevlemezhelyet (amit a `du` mutat) csak a nemnulla adatrészek foglalnak.

Az alkotóelemek összegyűjtése

Rakjuk össze a szükséges összetevőket egy frissen, ropogósan elkészített üres könyvtárba. Győződjünk meg arról, hogy van elegendő hely; végül is egy teljes értékű *Debian* telepítést szeretnénk tető alá hozni. Három gigabájt bőségesen elegendő lesz az alaprendszer számára. Töltsük le a *Debian-3.1-x86-root_fs.bz2* fájlt a ➲ *uml.nagafix.co.uk* weblapról, valamint a *2.6.14.3-bs3 UML* kernelt a ➲ *www.user-mode-linux.org/~blaisorblade/binaries* helyről. Végül szerezzük be az *UML* segédprogramok forrását innen: ➲ *user-mode-linux.sourceforge.net/dl-sf.html*.
Ha ezek közül valamelyik már nincs ott, igyekszem megadni az új lelőhelyet a cikkhez tartozó források között.
Álljon itt egy szkript a fenti műveletsor megvalósítására. Összegyűjti a szükséges alkotóelemeket és lefordítja az *UML* segédprogramokat, melyek csak forrásban érhetőek el.
Ha nem lesz szükségünk hálózatra, elhagyható az *uml_utilities* tarlabda beszerzése és lefordítása. A parancsok megvalósíthatóak normál felhasználóként, kivéve az *UML* segédprogramok telepítését, mely rendszergazda jogokat igényel (ezért a `su root`):

```
mkdir /tmp/UML-Demo
cd /tmp/UML-Demo
wget http://uml.nagafix.co.uk/
➥Debian-3.1/
-->Debian-3.1-x86-root_fs.bz2
bunzip2 Debian-3.1-x86-
➥root_fs.bz2
wget http://www.user-mode-
➥linux.org/~blaisorblade/
➥binaries/
-->2.6.14.3-bs3/uml-release-
➥2.6.14.3-bs3.tar.bz2
tar -xvjf uml-release-2.6.14.3-
➥bs3.tar.bz2
cp um32-2.6.14-release-mod/
➥vmlinux-2.6.14.3-bs3  .
wget http://mirror.usermode
➥linux.org/uml/
-->uml_utilities_
➥20040406.tar.bz2
tar -xvjf
➥uml_utilities_20040406.
➥tar.bz2
cd tools
make all
su root
make install DESTDIR=/
exit
cd ..
```

Ezzel megvannak a szükséges összetevők, táncra készen állunk. Minden Linux rendszernek van egy kernel parancssora. A legtöbb rendszerben ezt a betöltéskezelő hívja meg (*GRUB*, *LILO* stb.). Ebben az esetben viszont mi magunk fogalmazzuk meg a parancssort, amivel utasítjuk a rendszermagot a *Debian* gyökér fájlrendszer képmásfájl (és a hozzá tartozó, *Debian1.cow* nevű *COW* fájl) használatára – e kettő együttesen hozza létre a fájlrendszer gyökerét.
Az aktuális terminálunk lesz a vendég *UML* rendszer konzolja:

```
cd /tmp/UML-Demo
./vmlinux-2.6.14.3-bs3 ubd0=
➥Debian1.cow,Debian-3.1-x86-
➥root_fs root=/dev/ubda
```

Miután lefutott a parancs, ismerős *Linux* kernelüzeneteket látunk megjelenni, végül ott villog a *Debian* bejelentkezési parancssor végén a kurzor. Beléphetünk *root*-ként (nincs jelszó), és kémleljünk kicsit körbe – magam az *1. Lista* szerint tettem meg ezt.
Elég dögös, nem igaz? Íme a legszemélyesebb *Debian 3.1* homokozónk, melyben kedvünkre építhetünk vagy rombolhatunk. El lehet tekinteni a *hwclock*ra és a *tty0*-ra vonatkozó figyelmeztetésektől; ezek a legtöbb *UML* rendszerben megjelennek, hiszen néhány hardverjellemzőt nem támogat az *UML* rendszermag.
Ezen a ponton átállítható a *root* jelszó is, ha szükséges.
Lássuk a hálózati kapcsolat felhúzását. Szükség van két statikus *IP* címre, mégpedig az alagút két vége számára, melyet a korábban lefordított UML segédprogramjaink által fogunk létrehozni. Példámban a 192.168.1.100-at és a 192.168.1.101-et használom, de bármi más is megfelel, ami illeszkedik a helyi hálózat adottságaihoz. Az induláshoz az alábbi módon indítsuk el *Debian UML* rendszerünket:

```
cd /tmp/UML-Demo
./vmlinux-2.6.14.3-bs3
➥ubd0=Debian1.cow,
➥Debian-3.1-x86-root_fs
➥root=/dev/ubda eth0=
➥tuntap,,,192.168.1.100
```

Ha *Debian* vendégrendszerünk lábra állt, jelentkezzünk be újból *root*-ként, és módosítsuk a hálózati konfigurációt.
Kezdjük a */etc/network/interfaces* fájl szerkesztésével, melynek csak ennyit kell tartalmaznia:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
 address 192.168.1.101
 netmask 255.255.255.0
 gateway 192.168.1.1
```

A választásunknak megfelelő gépnevet írjuk be a */etc/hostname* fájlba, végül másoljuk a gazdagép */etc/resolv.conf* névfeloldási beállításait Debian vendégrendszerünk azonos helyére.
Indítsuk újra a vendégrendszert. Miután újra betöltődött, tudnunk kell pingelni őt a helyi hálózat bármely pontjáról.
Lenne néhány javaslatom az újonnan hálózatba kötött *Debian* rendszerünkre vonatkozóan. Először is telepítsük az *OpenSSH*-t, majd frissítsünk minden telepített csomagot. Ezt az alábbi paranccsal tudjuk

1. Lista Debian 3.1 UML vendégrendszer betöltése

```
Debian GNU/Linux testing/unstable (none) tty0
(none) login: root  <ENTER>
Linux (none) 2.6.14.3-bs3 #7 Fri Dec 16 17:47:00 CET 2005 i686
GNU/Linux
The programs included with the Debian GNU/Linux system
are free software;
the exact distribution terms for each program
are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY,
to the extent
permitted by applicable law.

(none):~# ps -ef <ENTER>
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 22:56 ?        00:00:00 init [2]
root         2     1  0 22:56 ?        00:00:00 [ksoftirqd/0]
root         3     1  0 22:56 ?        00:00:00 [watchdog/0]
root         4     1  0 22:56 ?        00:00:00 [events/0]
root         5     1  0 22:56 ?        00:00:00 [khelper]
root         6     1  0 22:56 ?        00:00:00 [kthread]
root         7     6  0 22:56 ?        00:00:00 [kblockd/0]
root         8     6  0 22:56 ?        00:00:00 [pdflush]
root         9     6  0 22:56 ?        00:00:00 [pdflush]
root        11     6  0 22:56 ?        00:00:00 [aio/0]
root        10     1  0 22:56 ?        00:00:00 [kswapd0]
root        12     1  0 22:56 ?        00:00:00 [kjournald]
root       299     1  0 22:56 ?        00:00:00 /sbin/syslogd
root       305     1  0 22:56 ?        00:00:00 /sbin/klogd
root       343     1  0 22:56 ?        00:00:00 /usr/sbin/exim4
                                                ➥-bd -q30m
root       348     1  0 22:56 ?        00:00:00 /usr/sbin/inetd
daemon     361     1  0 22:56 ?        00:00:00 /usr/sbin/atd
root       364     1  0 22:56 ?        00:00:00 /usr/sbin/cron
root       379     1  0 22:56 tty0     00:00:00 /bin/login --
root       380   379  0 22:56 tty0     00:00:00 -bash
root       384   380  0 22:57 tty0     00:00:00 ps -ef

(none):~# df -h <ENTER>
Filesystem            Size  Used Avail Use% Mounted on
/dev/ubda             1008M  264M  694M  28% /
tmpfs                  768M     0  768M   0% /tmp
tmpfs                   14M     0   14M   0% /dev/shm

(none):~# halt <ENTER>
Broadcast message from root (tty0) (Sun Jan 15 22:57:17 2006):
The system is going down for system halt NOW!
```

megtenni (válaszolva a menetközben felmerülő egyszerű kérdésekre):

```
apt-get update
apt-get install openssh-server
apt-get upgrade
```

Innentől a lehetőségek szinte korlátlanok. Bármely hálózati szolgáltatás vagy alkalmazás futtatható eme ***Debian UML*** vendégrendszer alól. Egyetlen gépen többféle kernelváltozat vagy disztribúció alatt ki lehet próbálni alkalmazásokat. A fájlrendszer képmásfájlt és a ***COW*** fájlt fel lehet tölteni egy pendrive-ra, melynek révén stabil, számítógép-független fejlesztési környezetben dolgozhatunk akárhol. A ***Felhasználói Módú Linux*** könnyűvé és fájdalommentessé teszi olyan rendszerváltoztatások kipróbálását, melyek egyébként a betölthetetlenségig porba döntenék a rendszert.

A pusztítás öröme...

Rendben, ezek után már mindenki tudja, hogy már a burokban is erre vágyott tulajdonképpen. Íme, itt a lehetőség. Indítsunk el egy különálló ***Debian UML*** vendégrendszert, és engedjünk el egy `rm -rf /.` parancsot. Ha hozzám hasonló érzésvilágú a kedves Olvasó, akkor valószínűleg remegő ujjakkal üti majd le ezt a néhány billentyűt. Indítsuk tehát el újdonsült ***Debian*** vendégrendszerünket az alábbi paranccsal (közben tudatosítva azt, hogy most egy másik, beszédes nevű ***COW*** fájlt használunk, nem pedig azt, amin az imént véres verejtékkel beállítottuk a hálózati paramétereket):

```
cd /tmp/UML-Demo
./vmlinux-2.6.14.3-bs3  ubd0=
➥DangerDanger.cow,Debian-3.1-
➥x86-root_fs
-->root=/dev/ubda
```

Miután halálraítélt barátunk betöltődött sorsába, essünk túl az elkerülhetetlenen. Győződjünk meg arról kétszer(háromszor), hogy ezt ***Debian*** vendégrendszerünk terminálablakában gépeljük be (*2. Lista*), nem pedig a gazdagépen.
Elég rendesen elintéztük. Még egy `halt` parancsot sem bírunk kiadni, mivel maga ez a parancs is megsemmisült. Egy másik ablakból lőjük ki a rendszert ezzel a paranccsal:

```
killall -9 vmlinux-2.6.14.3-bs3
```

Most pedig nézzük meg, mi történne, ha újra be szeretnénk tölteni a rendszert a fenti paranccsal (*3. Lista*).
Elvérzett. Mindenesetre tanuljunk a leckéből, és ***ne*** tegyünk ilyet igazi

2. Lista A rombolás örömszerző jellegű, főleg, ha csak próbaképpen tesszük

```
Debian GNU/Linux
➥testing/unstable (none)
➥tty0
(none) login: root
Linux (none) 2.6.14.3-bs3 #7
➥Fri Dec 16 17:47:00 CET
➥2005 i686 GNU/Linux
The programs included with
➥the Debian GNU/Linux system
➥are free software;
the exact distribution terms
➥for each program are
➥described in the
individual files in
➥/usr/share/doc/*/copyright.
Debian GNU/Linux comes with
➥ABSOLUTELY NO WARRANTY, to
➥the extent
permitted by applicable law.

(none):~#  rm -rf /
rm: cannot remove
➥`//proc/meminfo': Operation
➥not permitted
rm: cannot remove
➥`//proc/uptime': Operation
➥not permitted
(... Many warnings about
➥read-only filesystems
➥omitted ...)
rm: cannot remove
➥`//proc/loadavg': Operation
➥not permitted
rm: `//proc/self' changed
➥dev/ino: Operation not
➥permitted
(none):~# df -k
-bash: df: command not found
(none):~# ps -ef
-bash: ps: command not found
(none):~# halt
-bash: /sbin/halt: No such
➥file or directory
(none):~#
```

rendszerben. Szerencsénkre ez csak egy *UML* vendégrendszer *COW* fájllal, így egyszerűen törölhetjük a *DangerDanger.cow* fájlt (*jelentése: VeszélyVeszély*), és vendégrendszerünk vissza is állt eredeti egészségi állapotába.

3. Lista Egy igen beteg Debian UML vendégrendszer

```
(... boot messages omitted ...)
EXT3-fs: INFO: recovery required on readonly filesystem.
EXT3-fs: write access will be enabled during recovery.
kjournald starting.  Commit interval 5 seconds
EXT3-fs: ubda: orphan cleanup on readonly fs
EXT3-fs: ubda: 66 orphan inodes deleted
EXT3-fs: recovery complete.
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Warning: unable to open an initial console.
Kernel panic - not syncing: No init found. Try passing init=
➥option to kernel.
EIP: 0073:[<a01c6691>] CPU: 0 Not tainted ESP: 007b:b7f3afac
➥EFLAGS: 00000282
Not tainted
EAX: 00000000 EBX: 000012eb ECX: 00000013 EDX: 000012eb
ESI: 000012e8 EDI: 00000000 EBP: b7f3afb8 DS: 007b ES: 007b
a10afb80:  [<a0032d2a>] show_regs+0x21a/0x230
a10afbb0:  [<a0016c8c>] panic_exit+0x2c/0x50
a10afbc0:  [<a004a275>] notifier_call_chain+0x25/0x40
a10afbe0:  [<a0037501>] panic+0x71/0x100
a10afc00:  [<a000e2c0>] init+0x100/0x170
a10afc20:  [<a002bf59>] run_kernel_thread+0x39/0x50
a10afce0:  [<a001c3d4>] new_thread_handler+0xc4/0x120
a10afd20:  [<b7f3b420>] 0xb7f3b420
```

Még néhány megjegyzés a COW fájlokkal kapcsolatban

A *uml_moo* segédprogram az *UML* segédprogramok egyike. Segítségével megtehetjük azt, hogy egy fájlrendszer-képmásfájlt és a hozzá tartozó *COW* fájlt beolvasunk, és egy olyan fájlrendszer-képmásfájlt generálunk, mely e kettő egybeolvasztásával születik (ahogy azt a vendégrendszer is látja futás közben). Ez egyszerűvé teszi a működő, jól beállított vendég-fájlrendszer képmásfájlok sokszorosítását.

Összefoglalás

A *Felhasználói Módú Linux* jó szórakozás, de van néhány igazi alkalmazása is. Ki lehet vele próbálni ismeretlen vagy megbízhatatlan alkalmazásokat, miközben a nem a teljes gazdagép lesz a veszélyeztetett terület, hanem csak a *COW* fájl. Virtuális hálózatokat is létrehozhatunk az egyszerre elindított *UML* rendszerekkel, így minimális ráfordítással lehet „laboratóriumi körülményeket” teremteni, ahol ki lehet próbálni mindazokat az eszement linuxos trükköket, amelyeket félne az ember „rendes” számítógépeken elkövetni.

Linux Journal 2006., 145. szám

Matthew E. Hoskins vezető UNIX rendszeradminisztrátor a New Jersey Technológiai Intézetben, ahol a testületi adminiszratív feladatok jó részét ő látja el. Szeret azzal kísérletezni, hogy egészen eltérő rendszereket és szoftvereket bír együttműködésre, többnyire egy vékony Perl ragasztóréteggel (amit a helyiek „MattGlue” néven szoktak emlegetni; glue=enyv). Amikor nem számítógép-rendszerekkel trükközik, akkor a konyhában teszi ugyanezt. Matt a Hivatásos Újságírók Társaságának a tagja. Szomjúhozza az Olvasók visszajelzését – elérhető a matt@njit.edu e-mail címen.

KAPCSOLÓDÓ CÍMEK

➲ www.linuxjournal.com/article/8883

Pendrive-ok automatikus mentése

Használjuk ki az udev képességeit és készítsünk biztonsági mentést Flash meghajtónk tartalmáról naponta, vagy akár minden egyes csatlakoztatáskor...

Tegye föl a kezét, aki még soha nem vesztette el az ***USB Pendrive***-ját – vagy legalább nem gyanított ilyesmit. Aztán amikor az ember már órákat töltött az elbitangolt jószág előkerítésével, akkor egyszer csak kinyitja a mosógép ajtaját, és hopp, az eszköz a lábai elé hullik. Nos, igen... Már megint elfelejtettük akkurátusan kiforgatni az összes nadrág zsebeit, mielőtt betettük volna ezt az adagot. De semmi gond, hiszen minden adatunkról van biztonsági másolatunk. Ugye van? Ugye bizonyos időközönként rutinszerűen bemásoljuk a Flash meghajtó teljes tartalmát a merevlemezre? Nem?! Hát, akkor gond van. A rendszeres adatmentés ilyetén kivitelezése természetesen számomra is borzalmasan unalmasnak tűnik. Mellesleg pont az ilyen feladatok automatizálására találták ki a számítógépeket. Meg aztán az sem mellékes, hogy az automatikus és főként fájdalommentes adatmentési módszer kidolgozása maga is élvezet egy vérbele felhasználó számára.
Na, de nem ártana tisztázni, mit is értek pontosan az alatt, hogy „fájdalommentes"? Mit szólna például a nyájas olvasó egy olyan megoldáshoz, amelynél az ember csak odasétál egy linuxos géphez, csatlakoztatja a pendrive-ját, megvárja „a mentés elkészült" hangot, aztán kiveszi az eszközt a csatlakozóból, és továbbsétál. Na és persze azt se felejtsük el, hogy néha nem árt, ha egy ilyen rendszernek van egy viszonylag rövid távú emlékezete is, vagyis mondjuk az utolsó hét mentés tartalma legyen bármikor hozzáférhető. Ja igen, és mondjuk kezelje gond nélkül a titkosított meghajtókat is.
Na és még valami: ha arra kerülne a sor, hogy tényleg vissza kelljen állítani valamit a mentéből, akkor legyen képes teljes és fájlonkénti visszaállításra is. Na, szóval ennyi lenne. Aki ***Linuxot*** használ, annak mindez nem gond. Valamennyi fent leírt funkciót megvalósíthatjuk az udev képességeire támaszkodva egy egyszer héjprogrammal. A szükséges eszközök már ott vannak valamennyiünk rendszerén. Ebben a cikkben és a ***CentOS 4.3*** rendszeren fogom bemutatni a probléma megoldását, amely 2.6-os kernelt használ, de minden bemutatott dolog természetesen más rendszereken is megvalósítható.

Az udev és az adatmentés

Az ***udev*** egy modern eszközkezelő, amely a 2.4-es ***Linux*** kernelben található ***devfs*** rendszert váltotta le. Az ***udev*** képes valamennyi eszközt a rendszerbe leképezni, beleértve természetesen a menet közben csatlakoztathatókat (hotplug) is. Az egyik legnagyszerűbb szolgáltatása az, hogy akár saját eseménykezelőket is írhatunk hozzá.
És ezzel el is érkeztünk a cikk témájához, hiszen a következőkben arról lesz szó, hogyan írhatunk meg egy olyan eseménykezelőt, amely az ***USB*** pendrive csatlakoztatásakor lementi annak teljes tartalmát a merevlemezre. A szabályok a */etc/udev/rules.d* könyvtárban találhatók. (Aki más terjesztést használ, az ellenőrizze a */etc/udev/udev.conf* fájlban az `udev_rules=` kezdetű sort. Elvileg ez tartalmazza a szabályokat hordozó könyvtár nevét.) Ebben a könyvtárban gyakorlatilag bármilyen ***udev*** szabályt elhelyezhetünk. A rendszer az új előírásokat ráadásul azonnal használni kezdi, anélkül, hogy a gépet újra kellene indítanunk.

Az eszközök azonosítása

Ahhoz, hogy megírhassunk egy szabályt az ***udev*** rendszerhez, először is rendelkeznünk kell egy olyan módszerrel, amivel egyértelműen azonosítani tudjuk az egyes ***USB*** eszközöket.
A legtöbb pendrive-nak van sorozatszáma, bár ez nem mindegyikről mondható el. Szerencsére az ***udev***-nek arra az esetre is van megoldása, ha nincs az eszköznek ilyen egyedi azonosítója. Jómagam két pendrive-ot használok: egy ***JetFlash JF110***-et, amelynek a tartalma ***TrueCrypt*** segítségével van titkosítva, és egy ***Corsair Flash Voyager*** típusút. A ***JetFlash***-nek van egyedi azonosítószáma, a ***Corsairnak*** azonban nincs. Csatlakoztassuk a pendrive-ot, majd adjuk ki a

```
cat /proc/scsi/usb-storage/*
```

parancsot. Erre egy ehhez hasonló információhalmazt kapunk:

```
   Host scsi5: usb-storage
       Vendor: Unknown
      Product: USB Mass Storage
➥Device
Serial Number: 85a5b1f2c96492
     Protocol: Transparent SCSI
    Transport: Bulk
       Quirks:
```

Ha a kiírt adatok között van sorozatszám, akkor az ebbe a szakaszban leírtakat ki is hagyhatjuk, és nekiláthatunk magának a szabálynak a megírásához. Ha azonban a ***„Serial Number"*** rovatban a ***„None"*** kifejezés díszeleg, akkor sincs minden veszve, mert az `udevinfo` segítségével rendelhetünk az eszközhöz egyedi azonosítót. Ehhez a következőket kell tennünk.

1. Nézzük meg a `dmesg` parancs kimenetét. Egy tipikus részlet a következőképpen fest:

```
usb-storage: waiting for device
➥to settle before scanning
  Vendor: Corsair   Model:
➥Flash Voyager     Rev: 1.00
  Type:   Direct-Access
  ANSI SCSI SCSI device sde:
➥2031616 512-byte hdwr sectors
➥(1040 MB)
[...]
sde: assuming drive cache:
➥write through
 sde: sde1
Attached scsi removable disk
➥sde at scsi12, channel 0, id
➥0, lun 0
Attached scsi generic sg4 at
➥scsi12, channel 0, id 0, lun
➥0,  type 0
```

A fenti szöveg szerint eszközünk a `/dev/sde` ponton keresztül kapcsolódik a rendszerhez.

2. Futtassuk a következő parancsot:

```
udevinfo -a -p $(udevinfo -q
➥path -n /dev/sde)
```

majd vizsgáljuk meg a kimenetét. A következő sorokat kell keresnünk:

```
BUS=="scsi"
SYSFS{model}=="Flash Voyager  "
SYSFS{vendor}=="Corsair "
```

A kezelési szabály megírása

Ha kezünkben a sorozatszám, vagy a gyártó/modell páros, akkor nekiláthatunk a kezelési szabály megírásának. A szkript úgy fog működni, hogy létrehoz egy az eszközre mutató szimbolikus láncot a */dev* fában (esetünkben ez lehet például */dev/corsair_drive*), majd meghívja a */usr/local/bin/backup-thumb.sh* szkriptet, amit mindjárt részletesen is megvizsgálunk. Jelentkezzünk be rendszergazdaként (`su -`) majd a */etc/udev/rules.d* könyvtárban hozzunk létre egy *95.backup.rules* nevű szövegfájlt. Természetesen nem kötelező a 95-ös számot használni, de ne felejtsük el, hogy az *udev* ábécé sorrendben dolgozza fel a szabályokat, a helyi műveleteket pedig jobb a sorozat végére tenni.

1. Lista Az adatmentéshez használt szkript

```
#!/bin/bash
# Pendrive mentésére szolgáló
# szkript
##############################
# CONFIG szakasz
# Hol akarjuk tárolni a mentett
# adatokat
BACKUP_DIR=/backups/thumb
# Hány visszamenőleges
# változatot szeretnénk tárolni
GENERATIONS=7
# Naponta csak egy mentés
# készüljön
# Ha azt szeretnénk, hogy
# a pendrive minden egyes
# csatlakoztatásakor
# lefusson a folyamat, írjunk
# ide nullát
BACKUP_ONCE_DAY=1
# Ha a mentés kész, a következő
# nagállományt játssza le
# a rendszer
SOUND=/usr/share/sounds/KDE_Bee
➥p_ClockChime.wav
# END CONFIG
##############################
# Főprogram
# Megvárjuk, míg az eszköz
# üzemkész állapotba kerül
sleep 10
# Gondoskodunk róla, hogy
# a mentést senki más ne tudja
# lemásolni.
umask 077
# Ellenőrizzük, hogy létezik-e
# a könyvtár
DEVICE=$1
if [ ! -d ${BACKUP_DIR} ] ;
then
        mkdir -p ${BACKUP_DIR}
fi
# Naponta csak egyszer mentünk
if [ ${BACKUP_ONCE_DAY} -gt 0 ]
# ; then
       DIDTODAY=${BACKUP_DIR}/$
➥{DEVICE}.did_today
        find ${BACKUP_DIR} -
➥name ${DEVICE}.did_today -a
➥-mtime +1 -delete
        if [ -f ${DIDTODAY} ] ;
➥then
                exit
        else
                touch $
➥{DIDTODAY}
        fi
fi
# Mentések cseréje (rotációja)
cd ${BACKUP_DIR}
let GENERATIONS=${GENERATIONS}-1
while [ ${GENERATIONS} -ge 0 ]
➥; do
  let NEWFILE=${GENERATIONS}+1
        if [ -f ${DEVICE}.
➥backup.${GENERATIONS} ] ; then
          mv -f ${DEVICE}.
➥backup.${GENERATIONS}
${DEVICE}.backup.${NEWFILE}
        fi
  let GENERATIONS=$
➥{GENERATIONS}-1
done
# A mentés tényleges
# végrehajtása
dd if=/dev/${DEVICE} of=$
➥{BACKUP_DIR}/${DEVICE}.backup
➥.0 > /dev/null 2>&1
# Hangjelzés küldése a folyamat
# végén
aplay ${SOUND} > /dev/null 2>&1
```

Ha eszközünk rendelkezik sorozatszámmal, akkor a szabálynak valahogy így kell kinéznie (az egésznek egyetlen sorban kell lennie):

```
BUS="usb", SYSFS{serial}=
➥"85a5b1f2c96492", SYMLINK=
➥"jet_drive",
RUN+="/usr/local/bin/
➥backup-thumb.sh jet_drive "
```

Ha nem tartozik az eszközhöz egyedi numerikus azonosító, akkor a gyártó/modell párost kell használnunk erre a célra. Ebben az esetben az új szabály a következőképpen fest:

```
BUS="scsi", SYSFS{vendor}==
➥"Corsair ", SYSFS{model}==
➥"Flash Voyager  ",
SYMLINK="corsair_drive", RUN+=
➥"/usr/local/bin/
➥backup-thumb.sh
corsair_drive"
```

Érdemes talán megjegyezni, hogy szükség esetén tetszőleges számú `SYSFS{}` bejegyzés fűzhető össze, a lényeg, hogy a végeredmény valóban egyedi módon azonosítsa az eszközt. A most létrehozott új szabály mostantól minden alkalommal lefut, valahányszor csatlakoztatjuk a kérdé-

ses pendrive-ot. Szintén érdemes mindehhez hozzátenni, hogy egy eszközhöz több szabály is rendelhető. Ilyenkor az *udev* azokat a fájlban felülről lefelé haladva hajtja végre.

Az adatmentő szkript megírása

Esetünkben az egész automatikus rendszer lelke a ***backup.thumb.sh*** szkript lesz, ami a tényleges adatmentést elvégzi. Az imént létrehozott szabály tulajdonképpen semmi egyebet nem tesz, csak meghívja ezt a programot, miközben átadja neki annak az eszközfájlnak a nevét (esetünkben az imént említett szimbolikus láncot), amelyen keresztül a kérdéses hardverelem elérhető.
Minden más beállítás a `CONFIG` szakaszban található. A mentőszkript kódját az *1. Lista* tartalmazza.
Mentsük ezt a szkriptet */usr/local/bin/backup-thumb.sh* néven, és persze ne felejtsük el rá kiadni a `chmod +x` parancsot. A következő lépésben saját igényeinknek megfelelően szerkesszük át a beállításokat. A beállítható paraméterek a következők:

`BACKUP_DIR` : Hova akarjuk menteni a pendrive tartalmát.
`GENERATION` : Hány napra visszamenőleg szeretnénk megőrizni a lementett változatokat. A mentések 0-tól kezdődően számozódnak (0 a mindenkori legfrissebb változat) egészen a megadott határig. Persze ne feledkezzünk meg arról sem, hogy ha ezt az értéket magasra állítjuk, akkor rendelkeznünk kell az ehhez szükséges mennyiségű szabad kapacitással a háttértáron. Ha egy közönséges 1 GB-os USB eszközt használunk, a `GENERATIONS` értékét pedig 7-re állítottuk, akkor 7 GB helyre lesz szükségünk a merevlemezen.
`BACKUP_ONCE_DAY` : Ha naponta többször is csatlakoztatjuk és leválasztjuk a pendrive-unkat, akkor bizonyára nem szeretnénk, hogy minden egyes alkalommal mentés készüljön róla, megelégszünk egyetlen változattal is.
A ***backup-thumb.sh*** szkript egy jelzőfájlt használ, amivel megoldható, hogy ilyenkor naponta csak egyszer fusson le a kód. Ha meg szeretnénk változtatni ezt a viselkedésformát, vagyis kifejezetten azt akarjuk, hogy minden egyes csatlakoztatáskor készüljön másolat az adatokról, akkor sincs más dolgunk, csak nullára állítani ennek a változónak az értékét.
`SOUND` : Ebben a példában egy a ***KDE*** rendszerrel érkező hangeffektust használtam a mentés végének jelzésére, de természetesen bármilyen más ***WAV*** fájl megteszi. Ha pedig nem ***WAV***, hanem ***MP3*** kódolású hangállományt szeretnénk lejátszatni, akkor a kódban jelenleg használt `aplay-t` cseréljük le `madplay`-re.

Hogyan működik

A ***backup-thumb.sh*** szkript először is tíz másodpercig vár a rendszer indulásakor, mivel meg kell várnia, amíg a kernel befejezi a pendrive pásztázását. Ha csatlakoztatunk egy ilyen eszközt, majd azonnal kiadjuk a `dmesg` parancsot, akkor a „`waiting fo device to settle`” üzenetet fogjuk látni. Nos, erről van itt szó. Tíz másodperc ugyanakkor még az „idősebb” rendszereken is elegendő kell legyen arra, hogy a kernel befejezze ezt a műveletet.
A következő lépésben a ***backup-thumb.sh*** meglehetősen szigorú jogosultsági korlátokat léptet életbe, mégpedig úgy, hogy a mentéseket csak a root legyen képes olvasni. Ellenkező esetben bármelyik kíváncsi embertársunk átmásolhatná a lemezképeket egy másik gépre és ott szabadon turkálhatna bennük.
A szkript lényege tulajdonképpen a `dd` parancs megfelelő paraméterekkel való végrehajtása. A pendrive tartalmának mentése összességében csak ennyiből áll, ráadásul ez a módszer akkor is működik, ha az eszköz tartalma titkosított. Amikor a folyamat befejeződött, hangjelzést foguk kapni. Ha gépünk ***USB 2.0***-ás portokkal rendelkezik, akkor 1 GB adat lementése körülbelül egy percig fog tartani.

Adatvisszaállítás

Ha bekövetkezett a baj, vagyis elvesztettük vagy kimostuk jó öreg pendrive-unkat, akkor nincs más dolgunk, mint a legutóbbi lementett képet a `dd` paranccsal visszaírni egy másik eszközre, valahogy így:

```
dd if=corsair_drive.backup.0
➥of=/dev/corsair_drive
```

Ha pedig csak egyes fájlokat szeretnénk visszaállítani a mentés alapján, a következőt kell tennünk:

```
mkdir /mnt/thumb
mount -o loop corsair_drive.
➥backup.0 /mnt/thumb
```

Ezek után a */mnt/thumb* könyvtárban található bármilyen fájlról másolatot készíthetünk.
Ha a pendrive tartalmát a ***TrueCrypt*** segítségével titkosítottuk, akkor a lemezkép becsatolását a következőképpen kell végeznünk:

```
truecrypt corsair_drive.
➥backup.0 /mnt/thumb/
```

Látható tehát, hogy a visszaállítás gyakorlatilag ugyanolyan egyszerű, mint a rendszeres mentések elkészítése.
Aki pedig még arra is lusta, hogy odaballagjon az asztali gépéhez, bedugja a pendrive-ot és megvárja, amíg csippant a gép, hogy kész a mentés, nos az ... maradjon távol a mosógépektől.
Ez a cikk tulajdonképpen csak a felületet karcolta. Az ***udev*** szabályok segítségével az itt bemutatottaknál sokkal több, és sokkal bonyolultabb dolgot is meg lehet oldani. A menet közben csatlakoztatható (hot plug) eszközökhöz tartozó bármilyen eseményhez rendelhetünk udev szabályokat, amelyekkel gyakorlatilag bármilyen művelet elvégzését előírhatjuk. Automatizálhatjuk például az eszközön található fájlrendszer becsatolását, digitális fényképezőgépről a képek átmásolását, vagy egy hálózati kapcsolat fölállítását. Az ***udev***-szabályok megírására használatos nyelv ráadásul igen hajlékony. Használhatunk többek között a ***printf*** vezérléséhez használatosakhoz hasonló helyettesítőkaraktereket, illetve a kezelt objektumokhoz a legkülönbözőbb jogosultságokat adhatjuk meg.
Az ***udev*** szabályok írásának rejtelmeiről talán a legjobb összefoglalót ***Daniel Drake*** készítette el. Ez az anyag ***„Writing udev Rules”*** címen a ➲ ***www.reactivated.net/writing_udev_rules.html*** címen érhető el.

Linux Journal 2006., 154. szám

Andrew Fabbro ugyan Oracle DBA fokozattal rendelkezik, de azért otthon még nem mindig ő viseli a nadrágot. Aki íri szeretne neki a témával kapcsolatban, az az andrew@fabbro.org címen érheti el.

A kereshető webhely

A Webglimpse használata weboldalon történő kereséshez és a keresésen alapuló hirdetések elhelyezéséhez.

Régen, amikor még kíváncsi diák voltam, ellátogattam egy előadásra, amely egy kicsi képzeletbeli lényről, a ***Maxwell*** démonról szót. Ez az okos kis lény ki tudja válogatni a magas hőmérsékletű részecskéket a levegőből, feltéve, ha ismeri azok helyét és sebességét. A démon alapvetően a tudást alakítja energiává (pontosabban fogalmazva az információt és a hőt munkává, illetve hasznosítható energiává). Azt hiszem, hogy azért maradt meg ez annyira a fejemben, mert kézzelfogható módon szemléltette az információ értékét, különösen a rendszerezett információét.
Egy gazdag tartalommal ellátott weboldal vonzza a látogatókat az értékes információn keresztül. Egy keresőmotor hozzáadása ezt az értéket megsokszorozza. És ha nem rendelkezünk ilyen tartalomban gazdag weblappal? Inkább egy bizonyos tárgykörrel kapcsolatos linkgyűjteményünk van? A ***Webglimpse*** segítségével olyan űrlapokat helyezhetünk el az oldalunkon, amellyel a linkgyűjtemény által hivatkozott oldalakon kereshetünk. Így a témakör keresésével, a hasznos információ válogatással, linkgyűjteménnyé szervezésével töltött munkánk segíthet a látogatókat saját oldalunkra csábítani. A továbbiakban ismertetem, hogyan használjuk a ***Webglimpse***-t az oldalunkon más kiválasztott lapokon történő kereséshez, és hogyan jussunk látogatottságunknak köszönhetően gyors hirdetési bevételhez.

A Webglimpse története

A ***Webglimpse*** több összetevő keveredéséből jött létre. Egy ***Perl***-ben íródott letapogató (spider) és kezelőfelületből és a ***C*** nyelven írott ***Glimpse*** a fő kereső és indexelő algoritmusból áll.
A ***Glimpse***-t először ***Udi Manber*** és ***Sun Wu*** számítástudomány-professzorok készítették, akik az ügyes kereső-algoritmus a korábban ***Sun Wu*** által még ***Manbert*** tanítványaként kifejlesztett (és ***agrep*** néven kiadott) úgynevezett fuzzy-minták illesztésére akarták használni.
A ***Glimpse*** eredetileg teljes fájlrendszerek keresésére alkalmas eszközként íródott 1993-ban, amely minden olyan felhasználó számára hasznos, akinek kavarodott már el dokumentuma vagy egy régi e-mail üzenete a merevlemezen.
A ***Webglimpse*** pár évvel később burkolódott a ***Glimpse*** köré, alkalmazva a hatékony kereső és indexelő algoritmust a teljes weben történő keresés helyett a böngészés és az egyéni weboldalakon történő keresés összekapcsolásához. A ***Pavel Klark*** és ***Michael Smith*** végzős hallgatók által ***Webglimpse*** megismertette a weboldalak „szomszédságában" vagyis a hivatkozott oldalakon történő keresés fogalmát. Időközben ***Manber*** és egy másik hallgató ***Burra Gopal*** új szolgáltatásokat írt hozzá, tovább finomítva a ***Glimpse***-t, hogy minél jobban megfeleljen az új környezet elvárásainak.
Ezen a ponton léptem színre én. Épp kiléptem az assembly hálózati kódok javítgatását végző állásomból az ***Artisofttól***, hogy saját céget alapítsak az interneten található információk felfedezésére és rendszerezésére. Az 1996-os korai Web időszakában úgy tűnt, a ***Webglimpse*** a legígéretesebb keresőeszköz. Teljesen új volt, még eléggé sorjás, szóval amikor ***Udi Manber*** elfogadta az ajánlatomat, hogy segítsen a projektben, az első dolgom az volt, hogy újraírtam a telepítést. Egyre inkább beleástam magam a ***Webglimpse***-be, új szolgáltatások hozzáadásával, felhasználók támogatásával, és 2000 januárjában az Arizonai Egyetem

kizárólagos licencet biztosított a cégem számára az értékesítéshez. Nem éreztem úgy, hogy ha a ***Webglimps*** nyílt forráskódú lenne, akkor tovább élhetne mint életem fő célja, de meghoztam a döntést, hogy mindig adjuk ki a teljes forráskódot, és a legtöbb non-profit szervezet számára tartsuk meg ingyenesnek. Ennek eredményeképp számos felhasználó akart visszajelzést adni és foltokat küldeni, és lehetővé vált hogy ingyenes, sőt továbbadható licenceket biztosítsak mindenkinek, aki segített a projektben.

Fogjuk munkára

A sok évnyi munkának köszönhetően a ***Webglimpse*** szinte minden ***Linux*** változaton és terjesztésen fut. Az egyetlen előfeltétel csupán az 5.004-es vagy újabb változatú ***Perl***-lel ellátott webkiszolgáló, és persze parancssori hozzáférés a kiszolgálóhoz.
A minden részletre kiterjedő útmutató elérhető a ***Webglimpse*** honlapjáról (lásd az on-line erőforrásoknál), így itt csak pár tanácsot említenék.
Ha a rendszerünkön már telepítve van a ***Webglimpse***, ellenőrizzük a változatszámát. A legtöbb előretelepített példány már régi (v 1.6 1998 környékéről), ilyen esetben jó ha van jogosultságunk a frissítéshez.
A ***Webglimpse*** változatszámát a legegyszerűbb módon úgy ellenőrizhetjük, hogy lefuttatunk egy keresést, és megnézzük az eredmény oldal forrását. A változatszám egy megjegyzésben található a keresési eredmények legelején.
A cikk írásának pillanatában a ***Webglimpse 3.0 FTP***-n keresztül telepíthető változatának béta tesztje zajlik. Kipróbálhatjuk ezt, vagy választhatjuk a régebbi 2.0-s változatot, ha van ***SSH*** hozzáférésünk a kiszolgálóhoz. Az ***SSH***-n keresztüli telepítéshez először töltsük le a kipróbálható változat tar fájljait a weboldalról. Kövessük a letöltési oldal tetején hivatkozott telepítési útmutatót, amely elmondja, hogy először le kell fordítani a ***Glimpse***-t, majd telepíteni a szokásos módon:

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

Aztán fel kell tenni a ***Webglimse***-t a telepítő parancsfájl segítségével

```
./wginstall
```

A parancsfájl végigmegy a szokásos kérdéseken: hová telepítse a programot, hová tegye a ***CGI*** szkripteket. Megpróbálja még elemezni az Apache fájlt (ha megtalálja), és megerősítést kér az elsődleges tartománynév és a weboldalgyökér beállításokkal kapcsolatosan. Mivel a ***Webglimpse*** beindexeli a helyi merevlemezen található fájlokat ***URL***-ekké alakítva az elérési útvonalat, ennélfogva meg kell adni, hogy hogyan írhatja át az elérési útvonalakat ***URL***-ekké. Ez egy olyan kulcsfontosságú dolog, hogy a webes adminisztrációs felület egy egész képernyőt szentel az ***URL*** >> fájl, fájl >> ***URL*** fordítások ellenőrzésére, hogy megbizonyosodjunk, ezt a részt jól beállítottuk.
Más beállítások a telepítés folyamán a biztonsághoz kapcsolódnak. Annak érdekében, hogy az archívumkezelő webfelületről futhasson, írhatóvá kell tenni az archívumkönyvtárat a webkiszolgáló számára. Ennek legbiztonságosabb módja nem az, hogy mindenki számára írhatóvá tesszük, hanem hogy csak a webszerver felhasználó számára engedjük ezt meg, ami azt a felhasználót takarja, akinek a nevében a webszerver fut. A leggyakrabban ez a ***www***, vagy a ***nobody*** felhasználó. Ezt a folyamatok listájából deríthetjük ki:

```
ps aux | grep httpd
```

Ami valami hasonló pár sort fog mutatni:

```
nobody      873  0.1  0.5 16492
➥11416 ?
    S    18:03   0:00 /usr/
➥local/apache2/bin/httpd
nobody      874  0.0  0.5 16492
➥11416 ?
    S    18:03   0:00 /usr/
➥local/apache2/bin/httpd
nobody      875  0.0  0.5 16552
➥11620 ?
    S    18:03   0:00 /usr/
➥local/apache2/bin/httpd
```

Az első oszlop a felhasználó neve, akinek a nevében a webkiszolgáló fut, ebben az esetben: ***nobody***. Most már megválaszolhatjuk a ***wginstall*** kérdését, és ha olyan felhasználó nevében futtatjuk a parancsfájlt, aki meg tudja változtatni a fájlok tulajdonosát, akkor az be is állítja azokat. Ha nem, váltsunk rendszergazdai módra a telepítés után, és változtassuk meg kézzel. Feltételezve, hogy az alapértelmezett ***/usr/local/wg2*** területre telepítettük, az alábbi parancsot kell futtatni, hogy az archívumkönyvtárat webről írhatóvá tegyük:

```
chown -R nobody
/usr/local/wg2/archives
```

Ha a telepítés véget ért, itt az ideje, hogy kijelöljük az indexelendő fájlokat és elkészítsük a kereső űrlapot. A ***Webglimpse*** szóhasználatával ez egy archívum beállítása.

Egy archívum beállítása

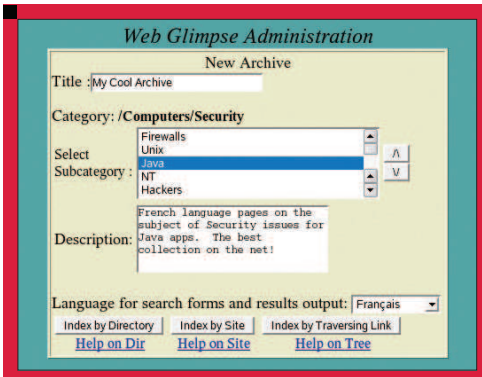
A telepítés végeztével az alábbihoz hasonló sorokat kell látnunk

```
*******************************
Done with install! You may use
 http://mycoolserver.com/
 ➥cgi-bin/wg2/wgarcmin.cgi
```

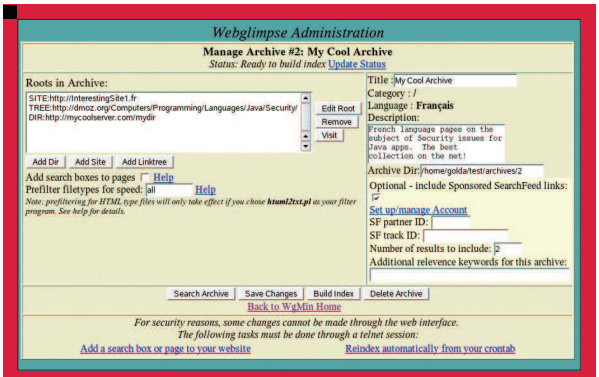
vagy:

```
/usr/local/bin/wgcmd
to configure archives at any
➥time.
(The web version currently has
➥more features)
Run wgcmd to create new archive
➥now? [Y]:
```

Ha már összebarátkoztunk a ***Webglimpse***-szel, a parancssoros eszköz nagyon kézhez álló megoldás a számos archívum kezeléséhez és újak létrehozásához. Első alkalommal, azt javaslom, használjuk a webes felületet. Úgyhogy nyomjunk ***N***-t hogy ne induljon el a ***wgcmd***, ehelyett nyissuk meg a ***wgarcadmin.cgi URL***-t a böngészőnkben, majd írjuk be a felhasználónév-jelszó párost amit a telepítés során választottunk. Ezután betöltődik az archívumkezelő, ami később majd az összes létrehozott archívumot mutatni fogja. Ha ez az első telepítés, akkor a lista üres, tehát nyomjuk meg az ***Új Archívum***



1. ábra Az Új Archívum képernyő lehetővé teszi, hogy megadjunk egy nevet és egy leírást, kijelöljük a kategóriát és kiválasszuk azt nyelvet, amin a keresési eredménynek megjelenik.



2. ábra Az Archívum szerkesztése képernyő lehetővé teszi a különböző helyekről származó weboldalak összekapcsolását egy kereshető indexállományba.

Hozzáadása (*Add New Archive*) gombot.
Most a az *Új Archívum* (*New Archive*) képernyőt kellene látnunk, ahogy az *1. ábra* mutatja.
Itt megadhatunk egy nevet egy leírást és lehetőségünk van kategóriát és nyelvet választani. A nyelv nem korlátozza a keresendő oldalakat, de meghatározza a keresési eredmény oldal sablonjának nyelvét és a karakterkódolási beállításokat. Ezek után kattintsunk az oldal alján található gombok egyikére:

3. ábra A Webglimpse sablonjai módosíthatók, hogy minél jobban illeszkedjenek a weboldalunkba. Mindössze az 'action' címke és a beviteli mezők nevei kötöttek.

- ***Index by Direcotry (Mappákon történő indexelés)***: indexeli a webkiszolgáló meghatározott könyvtárában elhelyezett fájlokat.

- ***Index by Site (Webhelyeken történő indexelés)***: indexeli a megadott weboldalt, akár a mi szerverünkön található, akár valamelyik másikon. A saját weboldalunk dinamikus tartalmainak beindexeléséhez is ezt kell használnunk.

- ***Index by Tree (Webhely bejárásán alapuló indexelés)***: indexeli a megadott kezdőoldalról hivatkozott összes oldalt az olyan beállításoknak megfelelően, mint hogy milyen mélységig és mennyi átugrással kövesse a hivatkozásokat.

Miután megadtuk az indexelendő könyvtárat vagy *URL*-t, és beállítottuk az olyan lehetőségeket, mint például az oldalak legnagyobb megengedett száma, az archívum fő vezérlőoldalára jutunk. Itt újabb oldalforrásokat adhatunk meg az indexelőnek, azaz egy archívumon belül keveredhetnek a helyi fájlok, távoli oldalak, teljes távoli webhelyek, ezekből akár több is, ha a helyzet úgy kívánja. A *2. ábra* az indulásra kész archívumot mutatja.
Amikor rákattintunk az Index felépítése (*Build Index*) gombra, beindul a weboldal-letapogató összegyűjti az oldalakat, kiszűri a *HTML* címkéket, és futtatja a *Glimpse* indexelőjét hogy létrehozza a gyors kereséshez szükséges fordított blokkszintű indexállományt.
Végezetül a a *Keresőlap vagy doboz hozzáadása az oldalhoz* (*Add a search box or page to your website*) linkre kattintva elkészíthetjük az oldalunkba illesztendő kereső űrlapot. Ez egy olyan oldalra visz, ahol háromféle keresőűrlap-forrást találunk ehhez az archívumhoz, az egyszerű keresőmezőtől a szabályos kifejezéseket is támogató, minden lehetőséget felfedő részletes keresőoldalig. A egyszerű keresőűrlap támogatja az *összes*, *bármely*, *pontos egyezés* keresési módokat, ahogyan a *3. ábra* is mutatja.
Ugyanezeket a keresőűrlapokat kapjuk, ha rákattintunk a *Keresés ebben az archívumban* (*Search this Archive*) gombra, vagy beírjuk a közvetlenül a *Webglimpse cgi*-re mutató *URL*-t (*http://mycoolserver.com/cgi-bin/wg2/webglimpse.cgi?ID=2*).
Alaphelyzetben ezek az archívum megadott nyelvén jelennek meg, de itt az angol változatot mutatja.

Tegyük kifizetődővé

Nos, van már egy kereshető archívumunk az adott témában fellelhető legtutibb hivatkozások gyűjteményére vonatkozóan. A felhasználók szerte a világon hasznot húznak az összegyűjtött tartalomból, és az oldalunkra látogatnak, hogy keressenek az igen hatékonyan indexelt tartalomban.
Ha szeretnénk, most már lehetőségünk van hirdetések megjelentetésére, hogy bevételhez jussunk, és működtetni tudjuk az oldalunkat. Ha visszamegyünk a *2. ábrán* látható *Archívum szerkesztése* képernyőre, jelöljük be az opcionális *include Sponsored SearchFeed links* feliratú jelölőnégyzetet. Ezután kattintsunk a *Fiók létrehozása/kezelése* (*Set up/manage Account*) hivatkozásra, amely a *Searchfeed* fiókkezelő oldalára mutat. On-line hirdetéskezelő és tartalomszolgáltató társaságként a Searchfeed.com olyan fizetett keresési eredményeket közvetít, amelyek relevánsnak számítanak

a felhasználók által az oldalunkon megadott keresőszó alapján. Miután elkészült a fiókunk, egyszerűen adjuk meg a partnerazonosítót és a forgalomazonosítót (***track ID***), amit a ***Searchfeed.com***-on kaptunk, és adjuk meg, hogy mennyi hirdetés jelenhet meg a keresési eredményeink elején. Nagyon egyszerű beállítani. Hogy a legtöbbet hozzuk ki a hirdetésekből, használhatjuk a ***Searchfeed*** on-line eszközkészletét, hogy követni tudjuk milyen keresőszavakat használnak a felhasználók, milyen hirdetésekre kattintanak, és mennyi jut nekünk az egyes kattintásokból.

Testreszabás

Függetlenül attól, hogy akarunk-e fizetett hivatkozásokat a keresési eredmény elejére vagy sem, nagyon valószínű, hogy rá akarjuk húzni az oldalunkra jellemző kinézetet, navigációt. Ennek elérése érdekében szerkesszük a ***wgoutput.cfg*** állományt az archívum könyvtárban. (Az archívum könyvtárának helye az Archívumkezelő képernyőn látszik.) A fájl egy ***HTML*** kódrészletet tartalmaz, amely az egyes keresési eredmények elé, közé és mögé kerül. Azt is megtehetjük, hogy a saját fejlécünket és láblécünket behívjuk (***include***), ahelyett, hogy ***HTML***-be irkálnánk bele. Néhány esetben szükségünk lehet az eredményhalmaz rangsorolásának testreszabására is. A ***Webglimpse*** más keresőmotorokkal ellentétben nem szándékozik meghatározni, hogy egy oldal hány százalékban hasznos a felhasználónak. Ehelyett betekintést enged a színfalak mögé, hogy hogyan számítja ki mennyire fontos egy találat, és ha úgy akarjuk, megadhatjuk az általunk készített, személyre szabott rangsoroló képletünket. Egyszerűen csak szerkesszük a ***wgrankhits.cfg*** fájlt, ami szintén az archívum könyvtárban található, és egy ***Perl*** kódrészletet tartalmaz, amely az alábbi változókat használja (itt magyarra fordítva):

```
# A használható változók:
#
# $N            # a kereső-
# szavak előfordulásának száma
# $LineNo       hol fordul elő
# a szüvegben a keresőszó
# $TITLE        # előfordulások
# száma a CÍM címkében
# $FILE         # előfordulások
# száma a fájl elérési útjában
# $Days         dátum (hány
# napja készült a fájl)
# $META         keresőszavak
# összes előfordulása bármilyen
# META címkében
# $LinkPop      hivatkozás
➥népszerűsége az oldalon (hány
➥másik oldal hivatkozik rá)
# %MetaHash     A szavak
# előfordulásának hash értéke
#  az egyes meta címkékben, a
#  NAME= paraméter alapján
#  indexelve
# $LinkString  A hivatkozás
# akutális URL-je
# A következő komment nélküli
# sorok
# az aktuális rangsoroló
# képletet alkotják
# Ez az alapértelmezett
# rangsorolás, magas
# súlyt ad a címben előforduló
# kulcsszavaknak,
# súlyozza az általános
# találatokat, népszerűséget
# és az időbeniséget.
$TITLE * 10 + $N + $LinkPop +
➥5/($Days + 1)
```

Hibaelhárítás

Mostanra van némi sejtésünk arról, hogy mi az erőssége, illetve mi a gyengéje a ***Webglimpse***-nek. Egy csomó ügyes sajátosság, amelyeket a felhasználó közvetlenül testre szabhat, és egy csomó ügyes sajátosság valamiféle alkalomszerű módszerrel összekeverve. A ***Webglimpse*** – a nézőponttól függően – meg van áldva, vagy épp átkozva a jó sok trükközéssel, ami lehetővé teszi a sok-sok különálló feladatot. A következő változat, amely a cikk írásának időpontjában javában készül, a tervek szerint egyszerűbben telepíthető és kezelhető lesz, nem beszélve a az ***FTP***-n keresztüli telepítési lehetőségről, azon felhasználók számára, akik nem rendelkeznek héj hozzáféréssel a kiszolgálón. Bárhogy is lesz, a leggyakoribb problémák, amikbe a jelenlegi változat használata során belefuthatunk, a következők.

1. ***Jogosultsági problémák:*** Akkor fordul elő, ha időnként újraindexeljük az archívumot a webfelületről, és néha a héjból vagy ***crontab***-ból történő újraindexelés esetén is. Bármely archívum újraindexelhető, ha a webfelületen az 'Index építése' gombra kattintunk, vagy ha kiadjuk héjból a `./wgreindex` parancsot az archívumkönyvtárban. A legjobban tesszük, ha ragaszkodunk a kiválasztott újraindexelési módhoz, és az archívumot annak a felhasználónak a tulajdonába adjuk, aki a szkriptet futtatni fogja.

2. ***URL/fájl átalakítási problémák:*** Akkor fordulhat elő, ha a `DocumentRoot` változó nincs helyesen beállítva. A webfelület főoldalán található ***Elérési útvonalak átalakításának tesztelése*** gombra kattintva ellenőrizhetjük, hogy az egyes fájlok milyen URL-lé alakulnak át, és fordítva. Minden, a helyi és távoli tartományokra alkalmazható beállítás a ***/usr/local/wg2/archives/wgsites.conf*** fájlban van tárolva. Közvetlenül, és az Archívumkezelő képernyő ***Tartomány szerkesztése*** gombra kattintva egyaránt szerkeszthetjük a ***wgsites.conf*** fájlt.

Még több hibaelhárítási tipp található ***Dokumentációk*** és hogyanok oldalon (lásd az on-line erőforrásoknál).

Köszönetnyilvánítás

A szerző köszönetét fejezi ki ***Udi Manbernek***, amiért rábízta ezeket a nagyszerű kreálmányokat! Még mindig úgy próbálom kezelni, ahogyan megérdemlik. Köszönet továbbá ***Sun Wunak***, ***Burra Gopalnak***, ***Michael Smith***-nek, és ***Pavel Klarknak***, a ***Webglipse*** és ***Glimpse*** társfejlesztőinek, és az összes felhasználónak, aki hibajelentéseket, foltokat, fordításokat és javaslatokat küldött az évek során.

Linux Journal 2006., 147. szám

Golda Valez az Arizonai Tucsonban él, programozó és édesanya. Ő a Webglimpse vezető fejlesztője 1997 óta. Emellett tulajdonosa és alapítója az 1995-ben alapított Internet Workshop nevű tárhelyszolgáltató és tanácsadó cégnek.

KAPCSOLÓDÓ CÍMEK

➲ www.linuxjournal.com/article/9021

Központosított hitelesítés és vállalati címtár megvalósítása (1. rész)

Ti Leggett sorozatában arról lesz szó, hogy hogyan lehet biztonságos vállalati címtárat létrehozni, amely támogatja az egyszeri bejelentkezést, és amely felhasználók ezreit képes kiszolgálni.

Vállalati címtárat szeretnénk tehát, de nincs vállalati költségvetésünk. Az egyszeri bejelentkezés előnyeit szeretnénk élvezni, mely mind az adminisztrátor, mind a felhasználók életét megkönnyíti. Ha ez a cél, és ráadásként elfogadnak egy biztonságos és egységesített hitelesítési és személyazonosság-nyilvántartási rendszert, akkor érdemes kitartóan végigolvasni e sorokat. Elindítom az olvasót a rendszergazda-mennyországba vezető úton. Ebben a cikksorozatban megmutatom, hogyan lehet építkezni a már előállított és helyretett alapelemekből, hogyan lehet újabbakat is beépíteni, és miként lehet ezek együttesét munkára bírni. A hitelesítést intéző kiszolgálóktól kezdve a levélkézbesítésig és kliensgépek integrálásáig (beleértve a ***Microsoft Windows*** vagy ***OS X*** operációs rendszert futtató gépekig) mindent végiggondolunk. Sok mindenről kell szót ejtenünk, úgyhogy kezdjünk is bele!

A korábbi építőkockák felhasználása

Gentoo Linuxon futtatott ***MIT Kerberos V v1.4.1***-et használunk hitelesítésre és ***OpenLDAP v2.1.30***-at címtárként, azaz személyazonosság-kezelésre. Három kiszolgálónk lesz: ***kdc.pelda.com***, ***ldap.pelda.com*** és ***mail.pelda.com***. (A beszédes nevek az alábbiakra utalnak: ***KDC=Key Distribution Center, Kulcselosztó Központ; LDAP=Lightweight Directory Access Protocol,*** címtár szolgáltatások elérését szabályozó protokoll) Mielőtt továbblépnénk, érdemes elolvasni a ***Linuxvilágból*** a ***„Központosított hitelesítés Kerberos 5-tel”*** cikksorozat első részét ➲ ***linuxvilag.hu/node/3002370***, valamint az ***„OpenLDAP mindenütt”*** című cikket ➲ ***linuxvilag.hu/node/3001551***, illetve ***Jászberényi József*** esettanulmányát 2006 szeptemberében és októberében. (Lásd az on-line forrásokat, valamint a ***Google-beli „LDAP OpenSUSE” keresőkifejezés is jó magyar bevezetőt ad az LDAP témához – a ford.***)
Onnan lépünk tovább, ahová ezek az írások eljutottak. Azt érdemes még szem előtt tartani, hogy a mi ***Kerberos*** tartományunk (***realm***) a ***CI.PELDA.COM***, így bázis ***DN***-ünk az ***o=ci***, ***dc=pelda***, ***dc=com*** lesz. (***DN = Distinguished Name***, megkülönböztető név; ***o = organization***, szervezet; ***dc = domain component***, tartománykomponens)
A cikkben hivatkozott összes konfigurációs fájl megtalálható a megjelölt on-line források között.

SSL Tanúsítvány Hatóság (Certificate Authority, CA) létrehozása

A továbbiak megértéséhez ezt a részt nem kötelező elolvasni, de olyan hálózatok építői számára, akik több szerveren is ***SSL***-t használnak, melegen ajánlott. Bár minden kiszolgáló alá tudná írni a maga számára a tanúsítványt, így elveszne valami az egységességből és a saját ***CA*** futtatásával járó sokféle lehetőségből. Az ***OpenSSL*** részletei iránt érdeklődőknek jó szívvel tudom ajánlani a ***Network Security with OpenSSL*** című könyvet (***John Viega, Matt Meisser, Pravir Chandra – O'Reilly***).
Legyen a ***/etc/ssl/pelda.com*** könyvtár az alapkönyvtár, ahol az összes aláírt tanúsítványt, visszavont tanúsítványlistát (***certificate revocation list, CRL***) és azonosítói információt tartjuk.
Ha e könyvtár elkészült, hozzuk létre benne a ***certs, crl, newcerts*** és ***private*** alkönyvtárakat, valamint ugyanitt egy üres ***/etc/ssl/pelda.com/index.txt*** fájlt. Írjuk egy „01”-t a ***/etc/ssl/pelda.com/serial*** nevű fájlba.
Ez utóbbiakat megtehetjük pl. az alábbi módon:

```
# touch /etc/ssl/
# pelda.com/index.txt
# echo '01' > /etc/ssl/
# pelda.com/serial
```

Végül hozzunk létre a ***CA*** számára egy ***OpenSSL*** konfigurációs fájlt ***/etc/ssl/pelda.com/ca-ssl.cnf*** néven.
Jelentkezzünk be olyan felhasználói néven, aki a ***/etc/ssl/pelda.com*** könyvtár rekurzív tulajdonosa (valószínűleg a ***root*** felhasználó). Az ő nevében az alábbiakat kell tennünk egy ön-aláírt ***CA*** tanúsítvány létrehozásához:

```
# export OPENSSL_CONF=
# /etc/ssl/pelda.com/
# ca-ssl.cnf
# openssl req -x509 -days 3650
# -newkey rsa \
 -out /etc/ssl/pelda.com/
 ➥ci-cert.pem -outform PEM
```

```
# cp /etc/ssl/pelda.com/
# ci-cert.pem /etc/ssl/certs
# /usr/bin/c_rehash
# /etc/ssl/certs
```

Az ***openssl req*** parancs részleteire vonatkozóan eligazítást ad a ***req(1)*** kézikönyv (man) oldal.
Fontos, hogy a ***CA*** kulcs jelszava (***passphrase***) igen biztonságos helyen legyen, mert ha a ***CA*** titkos kulcsa kitudódik, megbízhatatlanná válnak az általa aláírt tanúsítványok. Az is fontos, hogy a ***CA*** számítógép maga, illetve az elérése is biztonságos legyen. Ennek a biztonságnak a szintje az adminisztrátortól és az általa képviselt igényektől függ, de mihelyt egy illetéktelen felhasználó fizikailag vagy a hálózaton keresztül hozzá tud férni e géphez, már meg is szerezte a ***CA*** titkos kulcsát. Ahogy már fentebb említettem, a ***CA*** titkos kulcsának veszélyeztetése lerombolja az egész bizalmi láncot, bizonytalanná és megbízhatatlanná válik az összes aláírt tanúsítvány. Egyesek szerint az a legjobb megoldás, ha a ***CA*** gép fizikailag el van különítve mindenféle hálózattól.
A tanúsítványok aláírására úgy kerülhet sor egy ilyen környezetben, hogy ***Tanúsítványregisztrációs Központok (Registration Authorities, RA)*** fogadják a beérkező ***Tanúsítványaláírási Kérelmeket (Certificate Signing Request, CSR)***. Itt ezeket a ***CSR*** kéréseket egy megbízható adathordozón átviszik a CA gépre, ahol sor kerül az aláírásra és a tanúsítványoknak az adathordozóra történő visszaírására, amit a ***Tanúsítványregisztrációs Központtól*** a végfelhasználó átvehet.
Amennyiben erre van szükségünk, akkor az ***OpenCA*** projekt ilyen biztonsági szintet céloz meg; támogatja az aláírt tanúsítványok ***LDAP***-ben való eltárolását is.
Elkészítettük immár a ***CA*** számára az ***OpenSSL*** konfigurációs fájlunkat, de ez csak egyetlen tanúsítvány igénylésére és aláírására alkalmas. Elő kell állítanunk még egy olyan konfigurációs fájlt is, amelyet mostantól normál gazdagép- és felhasználói tanúsítványok igénylésére is lehet használni. Erre szolgál a ***/etc/ssl/pelda.com/ssl.cnf***. A kliensgépek konfigurálása a ***CA***-nál egy kissé összetettebb feladat, mivel többféle tanúsítvány-változatot kell kell kezelnie.

Most, hogy már van egy kliens konfigurációs fájlunk, generáljunk egy gazdagép tanúsítványt is az ***LDAP*** kiszolgáló számára. A ***CSR (Tanúsítvány-aláírási Kérelem)*** előállítása normál felhasználóként történhet:

```
# export OPENSSL_CONF=
# /etc/ssl/pelda.com/
# ssl.cnf
# openssl req -new -nodes
# -keyout ldap-key.pem \
 -out ldap-req.pem
```

Az ***openssl req*** kapcsolói lényegében ugyanazok, mint amiket a ***CA CSR*** legyártásához használtunk.
Az egyetlen újdonság a `-nodes` opció, ami egy kódolatlan titkos kulcsot gyárt (***man req***).
A nyilvános tanúsítvány elkészítéséhez vezető következő lépés a ***CSR*** aláíratása a ***CA***-val. Ezt ismét ***root***-ként kell megtenni:

```
# export OPENSSL_CONF=
# /etc/ssl/pelda.com/
# ssl.cnf
# openssl ca -policy
# policy_anything -out \
 ldap-ckert.pem -in ldap-
 ➥req.pem
```

Ebben a pillanatban három fájlunk van: az ***ldap-cert.pem***, a nyilvános tanúsítvány ; az ***ldap-key.pem***, a titkos kulcs; valamint az ***ldap-req.pem***, a ***CSR (Tanúsítvány-aláírási Kérelem)***.
Ez utóbbi eldobható, mihelyt a tanúsítványt aláírta a ***CA***. Itt is hangsúlyoznunk kell, mennyire fontos a titkos kulcs védelme, annál is inkább, mert nincs kódolva. Érdemes ezt a ***root*** birtokában tartani és 0400 jogosultsággal ellátni.

Legyen biztonságos az LDAP

Annak ellenére, hogy nincsenek jelszavak az ***LDAP*** címtárban, mégis van itt sok más értékes információ. Vélhetőleg a felhasználók nem örülnének, ha az interneten nyilvánosságra hoznák a telefonszámaikat, e-mail címeiket vagy munkavállalói azonosító számaikat. Ha olvasták az ***„OpenLDAP mindenütt”*** című írást és van egy működő ***LDAP*** kiszolgálójuk, akkor szükség van arra is, hogy biztonságosan történhessen az adatok átvitele és a címtár elérése.

Az adatátvitel biztosításának első lépése az ***OpenSSL*** használata.
Először is másoljuk át az aláírt tanúsítványunkat illetve kulcsunkat a ***/etc/openldap/ssl/slapd-cert.pem*** és a ***/etc/openldap/ssl/slapd-key.pem*** helyekre.A ***slapd.conf***-ban meg kell adnunk öt adatot: ***TLSCipherSuite*** (opcionális), ***TLSCACertificatePath***, ***TLSCertificateFile***, ***TLSCertificateKeyFile*** és ***TLSVerifyClient***. A ***slapd.conf(5)*** kézikönyv oldalak eligazítást adnak ezek mibenlétéről. (***TLS: Transport Layer Security; a protokoll elsődleges célja a titkosság és az adatintegritás biztosítása***)
Most, hogy biztonságosakká tettük a kábeleken áthaladó adatforgalmat, biztosítanunk kell a ***Kerberos KDC*** által használt hitelesítést is.
Az ***OpenLDAP*** „kerberizált”; ***SASL***-t (***„Simple Authentication and Security Layer”; „Egyszerű hitelesítési és biztonsági réteg”***) hitelesítési mechanizmust használ. Először is tudatnunk kell a ***slapd***-vel, hogy hol találja a kulcsokat tartalmazó ***keytab*** fájlt. Ezt a ***/etc/conf.d/slapd*** szerkesztésével tehetjük meg, vagy a ***slapd*** indítása előtt a megfelelő indító szkriptben létrehozott `KRB5_KTNAME` változó segítségével. A ***slapd.conf***-ban meg kell még adni a ***sasl-secprops*** és a ***sasl-regexp*** értékét.
E pillanatban mind a ***TLS***, mind a ***SASL*** mechanizmus használható, de ezek akár nélkülözhetőek is.
Még két opció van a ***slapd.conf***-ban (***security*** és ***allow***), melyek arra használhatóak, hogy megadjuk a biztonsági mechanizmust és a titkosítás erősségét, melyet néhány művelet elvégzése megkövetel. Győződjünk meg arról is, hogy az ***hozzáférés-szabályozó listák (ACL; Access Control List)*** megfelelően be vannak-e állítva. Javasolt kézikönyv-oldalak: ***slapd.access(5)***.

A Kerberos biztonságos átvezetése

Kezdjük azzal, hogy átvezetjük („replikáljuk”) a ***Kerberos*** adatbázisunkat a ***kdc.pelda.com***-ról az ***ldap.pelda.com***-ra. Ha valami miatt a ***kdc.pelda.com*** elromlana, az ***ldap.pelda.com*** át tudja majd venni a szerepét. Emlékeztetnék rá, hogy egy időpontban csak egyetlen ***kadmin*** kiszolgáló lehet a hálózati

tartományunkban. Máskülönben nem lehetne tudni, ki az illetékes az adatbázis-frissítések ügyében.
A *Kerberos* tartalmazza a *kprop* és *kpropd* programokat. Ezek megfelelő módon, biztonságosan el tudják terjeszteni a *Kerberos* adatbázist.
Először is meg kell adnunk a *kpropd*-t, mint ismert szolgáltatást. Írjuk a */etc/services* fájlba:

```
krb5_prop                754/tcp
```

Definiálnunk kell egy *ACL* fájlt is, a */etc/krb5kdc/kpropd.acl*-t, ami felvilágosítja a *kpropd*-t arra vonatkozóan, hogy mely gépek jogosultak a az adatterjesztésre. Igazából csak egy *KDC* főfiókot kell megadnunk a fájlban, de megadható akár az összes *KDC* gép is. Ekkor hiba esetén választhatunk egy új főfiókot, elindíthatjuk rajta a *kadmin* szolgáltatást, és tőle kezdődhet az adatterjesztés a szolgagépek felé.
Ezek után a szolgagépeken megadjuk az *xinetd* szolgáltatás-definíciót a */etc/xinetd.d/kpropd* fájlban; (újra)indítjuk az *xinetd*-t; kiírjuk az adatbázist a *kdc.pelda.com* gépen; és átvisszük a szolgagépekre az alábbi kezdőkonfigurációval:

```
# /usr/sbin/kdb5_util dump
# /etc/krb5kdc/slavedump
# /usr/sbin/kprop -f
# /etc/krb5kdc/slavedump \
 ldap.pelda.com
```

Végül minden szolgagépen létrehozunk egy biztonsági (*stash*) fájlt, mégpedig annak a mesterkulcsnak a segítségével, amit a *kdc.pelda.com* adatbázisának beállításakor használtunk; aztán elindítjuk a *KDC* szolgáltatást:

```
# /usr/sbin/kdb5_util stash
# /etc/init.d/mit-krb5kdc start
```

A *KDC* adatbázis rendszeres elterjesztése érdekében indítsunk egy *cron* parancsot a *kdc.pelda.com*-on. *Jason Garman*nek (és az *O'Reilly* által megjelentetett *„Kerberos: The Definitive Guide"* című könyvének) köszönhetően kezünkben van egy működő *cron* parancs.
Kézenfekvő, hogy ezt a szkriptet óránként indítsuk a */etc/cron.hourly* könyvtárból. Ezek után *Kerberos* adatbázisunk biztonságosan vezetődik át a főfiókból a szolgagépek sokaságára. Ha a főfiók elromlik, lehetőségünk van arra, hogy valamelyik szolgagép könnyen-gyorsan átvegye a feladatát, minimális adatvesztéssel (vagy szerencsés esetben anélkül).
Ha már át tudjuk vezetni a *Kerberos* változásokat egy-egy szolgagépre, akkor nyilvántartásba vehetjük őket a *krb5.conf* fájlban, mint érvényes *KDC*-ket.

Az OpenLDAP biztonságos átvezetése

Minden fontos rendszerben kerülnünk kell az egy pontból eredő hibaforrásokat, *SPOF*-okat (*Single Point of Failure; egypontos hiba*). Problematikus lenne csak egyetlen helyen tárolni az *LDAP* címtárunkat; nem kevés kritikus információ veszne el hiba esetén, sőt a felhasználóink még be sem tudnának jelentkezni, lehetetlenné válna az e-mailek megnézése és számos egyéb napi teendő. Az *LDAP* címtár átvezetése ezt küszöböli ki.
Replikáljuk tehát az *LDAP* címtárat az *ldap.pelda.com*-ról a *kdc.pelda.com*-ra. Az OpenLDAP-nak van is egy háttérprogramja (démonja), ami pontosan ezért felel: a *slurpd*. Sajnos a *slurpd*-nek nincs olyan beállítási lehetősége, amivel meg lehetne neki adni, melyik *Kerberos keytab* fájlt kellene használnia, így szükség lesz egy kis kézimunkára. Szerkesszünk bele a *slapd.conf* fájlba a *ldap.pelda.com*-on, megadva a *replogfile* és *replica* opciókat, majd indítsuk újra a *slapd*-t.
Létre kell hoznunk egy *Kerberos* alapú *LDAP* főszolgáltatást, egy *SSL* tanúsítványt és egy kulcsot a *kdc.pelda.com* számára, ahogyan azt a *ldap.pelda.com* esetében is tettük, és a *slapd.conf* fájlt is be kell állítanunk ugyanitt. Ez szinte ugyanolyan, mint amilyet az *ldap.pelda.com*-on készítettünk, néhány kulcsfontosságú különbséggel.
Ugyanabból az okból, mint ami miatt csak egyetlen *Kerberos* főkiszolgálónk van, itt is csak egyetlen *LDAP* címtárat tartunk naprakészen, és ezen hajtjuk végre a változtatásokat. Az egyetlen felhasználó, akinek lesz jogosultsága írni a szolgagépek címtárába, az alábbi módon írható le:

```
uid=host/ldap.pelda.com,cn=GSSA
➥PI,cn=auth
```

Ő nem más, mint a főfiók *Kerberos* gazdája; így a szolgagépek hozzáférés-szabályozó listáit (*ACL*-jeit) jóval szigorúbbra kell szabni. A *slapd*-nek arról is kell tudnia, hogy az *updatedn* és *updateref* opciók által megadott módon *ki* fog a *slurp* révén frissítéseket küldeni.
Most újra irányítsuk figyelmünket az *ldap.pelda.com*-ra. Létre kell hoznunk a */etc/conf.d/slurpd* fájlt, vagy be kell állítanunk a `KRB5CCNAME` változót, mielőtt a megfelelő szkript elindítja a *slurpd*-t.
Ezek után beszerezzük az indításhoz szükséges *Kerberos* igazolványokat (*credentials*):

```
# KRB5CCNAME=/var/
# run/slurpd.krb5cache
# /usr/bin/kinit -k
```

Majd az egész címtárat kiíratjuk egy fájlba:

```
ldap# /etc/init.d/slapd stop
ldap# /usr/sbin/slapcat -l
➥/tmp/slavedump.ldif
ldap# /etc/init.d/slurpd start
```

Minthogy a *slurpd* csak a főfiókra hat, nekünk kell benépesítenünk a szolgagépek címtárait a főfiók tartalma alapján. Ezt úgy tesszük meg, hogy a */tmp/slavedump.ldif* fájlba kiírt főfiók-adatbázist (amit az imént már elkészítettünk) átmásoljuk a *kdc.pelda.com*-ra, ahol a fájl beolvasása után elindíthatjuk a *slapd*-t:

```
kdc# /usr/sbin/slapadd -l
➥slavedump.ldif
kdc# /etc/init.d/slapd start
ldap# /etc/init.d/slapd start
```

Ellenőrizzük, hogy a szolgagép címtára megfelelő-e:

```
# ldapsearch -H
# ldap://kdc.pelda.com -ZZ
```

Próbáljuk ki, hogy jól működik-e az átvezetés. Módosítsunk vagy adjuk hozzá új adatot az *ldap.pelda.com* címtárához, majd keressünk rá a *kdc.pelda.com*-on, hogy megbizonyosodjunk arról, hogy a változtatások átvezetődtek-e.
Ha meggyőződtünk arról, hogy a *slurpd* működik, hozzunk létre egy

alkalmas ***cron*** parancsot az ***ldap.pelda.com***-on, hogy meggátoljuk az igazolványok elévülését. Az igazolványok érvényességének alapértelmezett ideje tíz óra, így ha pl. nyolc óránként futtatjuk a ***cron*** parancsot, az megfelelő lesz. Végül fel kell vennünk a ***kdc.pelda.com***-ot az érvényes *LDAP* kiszolgálók közé az ***nss_ldap*** számára. Azaz: be kell illesztenünk a ***kdc.pelda.com***-ot abba a kiszolgálófelsorolásba, ami a */etc/ldap.conf „host”* („gazdagép”) opciójában szerepel.

A Postfix levélkezelő beállítása

Postfix levélkezelőt (***mail transport agent, MTA***) fogunk használni. A 2.1.5-ös verziójú ***Postfixben*** már jól kiépített támogatás található az SASL hitelesítésre, valamint olyan *LDAP* finomságok támogatására, mint az álnevek (***aliasok***). Mivel a ***Postfix*** beállításának alapoktól történő bemutatása túlmutatna e cikk keretein, most csak azzal foglalkozunk, hogy hogyan lehet rávenni a programot az ***SASL*** és a ***TLS*** használatára. A ***Postfix*** részletes beállítására vonatkozóan: információk a cikkhez tartozó források között.
A ***Postfixnek*** két fő konfigurációs fájlja van: a */etc/postfix/main.cf* és a */etc/postfix/master.cf*. A ***main.cf*** elsősorban a bejövő levelek fogadásáért felelős, míg a ***master.cf*** inkább a levélkézbesítő programok (***mail delivery agent, MDA***) működtetéséért.
Egy példa ***main.cf*** megtekinthető a cikkhez tartozó források között, de a részletek megértéséhez érdemes ismerni a ***Postfix*** dokumentációját és weboldalát.
Három fő kulcsszó határozza meg azt, hogy *SMTP* (***Simple Mail Transfer Protocol – kommunikációs protokoll az e-mailek továbbítására***) kiszolgálónk hogyan értekezzen más *SMTP* kiszolgálókkal: ***smtp_sasl_auth_enable, smtp_use_tls*** és ***smtp_tls_note_starttls***. Ha *SMTP* kiszolgálónk ki lesz téve az internet viharainak, akkor ezeket a változókat a lehető legrugalmasabb módon kell beállítani, hogy biztosan sikerüljön más *SMTP* kiszolgálókkal a kapcsolatfelvétel. Ha ez csak egy belső *SMTP* kiszolgáló, akkor viszont biztonságosabbra lehet szabni ezeket a beállításokat.
Az érdekesebb feladat annak beállítása, hogy miként adjuk meg a felhasználóink és számítógépeink kapcsolódását a levélkezelőnkhöz a levelek elküldésekor. Néhány egyéb opció, amit ezzel kapcsolatban jó ismerni: ***smtpd_sasl_auth_enable, smtpd_sasl_security_options, smtpd_sasl_tls_security_options, smtpd_use_tls, smtpd_tls_cert_file, smtpd_tls_key_file*** és ***smtpd_tls_auth_only***.
Ha *IMAP* rendszerű levélkézbesítést használunk, akkor győződjünk meg arról, hogy be van-e állítva a ***master.cf***-ben a ***mailbox_transport*** változó értéke, valamint az ***smtp*** és ***cyrus*** átviteli (***transport***) mechanizmus.
Az *OpenLDAP*-hez hasonlóan a ***Postfix*** is kerberizált; *SASL*-t használ a hitelesítési képesség-egyeztetéskor és *SSL* segítségével tudja biztosítani az adatátvitelt. A ***Postfix*** biztonságossá tételéhez, az *SASL* használatára való beállításához lesz néhány teendőnk a ***main.cf*** módosításán túl is. ***Először*** létrehozunk egy *SSL* tanúsítvány/kulcs párt és elhelyezzük e két összetevőt a */etc/ssl/postfix/smtp-cert.pem* és a */etc/ssl/postfix/smtp-key.pem* fájlba, miközben megbizonyosodunk arról, hogy a ***postfix*** felhasználó és a ***mail*** csoport tulajdonában vannak, és hogy a kulcs csak a ***postfix*** felhasználó számára olvasható.
Ezután elkészítünk egy főfiókot a ***mail.pelda.com*** számára, és elmentjük a normál helyére. Egy főszolgáltatást is létrehozunk, ***„smtp/mail.pelda.com@CI.PELDA.COM”***-ként, ezt pedig a */etc/postfix/smtp.keytab*-ba mentjük el. Ezt a fájlt a ***root*** felhasználó tulajdonában kell tartani, és ugyanolyan jogosultságokkal felruházni, mint az ***smtp-key.pem*** fájlt. Ezek után még létre kell hoznunk egy *SASL* konfigurációs fájlt */etc/sasl2/smtpd.conf* néven, és át kell szerkesztenünk a */etc/conf.d/saslauthd*-t. A ***Postfix*** a ***saslauthd*** háttérprogramot használja arra, hogy információt kapjon a hitelesítési mechanizmusokról. A fenti két fájl adja az *SASL* tudtára, hogy hogyan ellenőrizze a jelszavakat, milyen mechanizmusok támogatottak, és mi legyen a minimálisan használt biztonsági szint. A ***minimum_layer*** felvehető értéke megegyezik az *OpenLDAP*-ben megadható biztonságossági faktoréval (***„Security Strength Factor”, SSF***).
Végül pedig a */etc/conf.d/postfix* fájllal megmondjuk a ***Postfixnek***, merre találja a ***Kerberos keytab*** fájlját. (Vagy, mint ahogy eddig már többször láthattuk: a ***Postfix*** indítása előtt a megfelelő indító szkriptben létrehozott `KRB5_KTNAME` változó segítségével is megtehetjük ugyanezt). Ha mindezekkel végeztünk, elindíthatjuk a ***saslauthd***-t és a ***Postfixet*** indító szkripteket.
Az *LDAP* nem pusztán a személyazonosság-szervezés és hitelesítés miatt hasznos, hanem amiatt is, mert a ***Postfix*** számára átadható ***álnevek*** (***aliases***) szótárát is tudja kezelni. Egyszerű használni és karbantartani, és feleslegessé teszi azt, hogy minden változáskor újrageneráljuk az álnévadatbázist. Címtárunk akkor válik először igazán erős eszközzé, amikor az álnévszótárat is rendelkezésére bocsátjuk. Ezt azzal tehetjük meg, hogy átadjuk a ***misc.schema***-t a ***slapd*** konfigurációnak, majd létrehozunk a címtárban egy elágazást (***branch***) az álnevek számára. Használjuk ezt:

```
ou=aliases,o=ci,dc=pelda,dc=com
```

(Itt `ou` = ***organizational unit***; a többi rövidítést lásd fentebb). Az utolsó feladatrész abból áll, hogy megmondjuk a ***Postfixnek***, hogy az *LDAP*-ot használja az álnevek beazonosításához. Ezt az ***ldap:/etc/postfix/aliases.cf***-nek a ***main.cf***-beli ***alias_maps*** opciójához való beírásával tehetjük meg, valamint ezzel párhuzamosan a */etc/postfix/aliases.cf* fájl létrehozásával, ami megadja, hogy hogyan kell az *LDAP*-hez kapcsolódni, s hogy hol is vannak az álnevek az *LDAP*-ben. Újraindítjuk a ***slapd***-t, majd a ***Postfix***-et; íme, készen állunk arra, hogy létrehozzunk egy e-mail álnevet. Hozzunk létre egy *LDIF* fájlt, nevezetesen az ***alias.ldif***-et, és vegyük fel a címtárba. Tá-dáááám! Készen vagyunk!

A cyrus IMAP levélkézbesítő beállítása

A ***cyrus IMAP*** levélkézbesítő program 2.2.10-es verzióját fogjuk használni. (***A cyrus egy jól skálázható vállalati levelezőrendszer, mely megbirkózik sokféle szabványra épülő technológiával; használható***

egyedülálló gépeknél, de akár óriási, centralizált intézeti hálózatban is – a ford.) A *cyrus IMAP* kiszolgáló részletes beállításának bemutatása túlmutatna e cikk keretein, de működőképes példák fellelhetők a források között. A *cyrus IMAP* kiszolgálót ugyanaz a csoport fejlesztette, aki a *cyrus SASL*-t is, így az *SASL* és az egyszeri bejelentkezés az elvárásoknak megfelelően működik.

A *Postfixhez* hasonlóan a *cyrus IMAP*-nak is két konfigurációs fájlja van: a */etc/imapd.conf* és a */etc/cyrus.conf*. Most csak a */etc/imapd.conf*-fal foglalkozunk. Itt is adott néhány előfeltétel: *SSL* tanúsítvány/kulcs pár, főfiók és főszolgáltatás; ez utóbbit hívjuk így: *„imap/mail.pelda.com@CI.UCHICAGO.EDU”*, és tároljuk el a */etc/imap.keytab* fájlban. Az *SSL* beélesítéséhez adjuk meg alkalmas módon a *tls_ca_path, tls_cert_file* és *tls_key_file* opciókat. Az *SASL* használatához meg kell adnunk a *sasl_pwcheck_method, sasl_mech_list* és *sasl_minimum_layer* opciókat is. Ezek értékei egyezzenek meg azzal, mint amiket a *Postfix* számára megadtunk a */etc/sasl2/smtpd.conf* fájlban.

A *Postfixhez* hasonlóan a *cyrus IMAP* számára is meg kell mondanunk, hogy hol van a *keytab* fájlja, mégpedig a */etc/conf.d/cyrus* fájl révén. (Vagy, mint ahogy ez már szinte a könyökünkön jön ki: az *IMAP* háttérprogram indítása előtt a megfelelő indító szkriptben létrehozott KRB5_KTNAME változó segítségével is megtehetjük ugyanezt). Ha mindezzel elkészültünk, győződjünk meg arról, hogy valóban fut-e a *saslauthd* program, és ha igen, akkor futtassuk az *IMAP* indító szkriptjét.

Vízre bocsátás

Meglehetősen nagyot kaszáltunk rövid idő alatt, de megérte a fáradozás: egy biztonságos és jól skálázható vállalati címtár lett az eredménye. Lábra állítottunk egy rendszert, amely akár néhány, egy helyre tömörült felhasználót/számítógépet, akár a világban szétszóródott tízezernyit is képes kiszolgálni. Következő cikkemben azzal fogunk megbirkózni, hogy munkánk gyümölcseként miként tudunk *Linux* és *Apple OS X* kliensgépeket is hálózatunkba kapcsolni.

Köszönetnyilvánítás

Munkámban segítséget nyújtottak: *Matematikai, Információtechnológiai és Számítástudományi tanszék (Office of Advanced Scientific Computing Research, Office of Science)*, az *Amerikai Energiaügyi Minisztérium* a *W-31-109-ENG-38* számú szerződés szerint. További támogatást kaptam a *Chicagoi Egyetem Számítástudományi Intézetétől* és a *Nemzeti Tudományos Alaptól*.

Linux Journal 2006., 140. szám

Ti Leggett (leggett@mcs.anl.gov) a Futures Laboratory of the Mathematics and Computer Science Division rendszergazdája az Argonne National Laboratory-ban; emellett a Chicagoi Egyetem Számítástudományi Intézetében is dolgozik.

KAPCSOLÓDÓ CÍMEK

➲ www.linuxjournal.com/article/8581

Hálózati szolgáltatások futtatása Felhasználói Módú Linuxon (1. rész)

Hálózati háttérprogramok elszigetelése a Linux kernel virtualizációs képességeinek kiaknázásával.

2006. májusában a ***Linux Journal Paranoid Pingvin*** rovatában már kifejtettem a ***Debian 3.1*** kitűnő képességeit virtualizációs környezetek támogatására, ide értve a ***Felhasználói Módú Linuxot*** (***User Mode Linux; UML***) is. Ugyanebben a számban ***Matthew Hoskins „Felhasználói Módú Linux”*** című cikkében olvashattunk egy gyors és boszorkányos receptet a ***Felhasználói Módú Linux*** kipróbálására, előre lefordított ***UML*** kernel és előre elkészített gyökér-fájlrendszer képmásfájlok segítségével. (***UML: User-Mode Linux, Felhasználói Módú Linux, FML***)

Ha ezek a cikkek felkeltették étvágyukat egy átfogóbb és biztonságosabbra tervezett ***FML*** rendszerre, akkor lássanak hozzá: az elkövetkezőkben belecsobbanunk a ***Felhasználói Módú Linux*** átélésébe, gondosan megvizsgálva minden mozzanatot (beleértve a kiadandó parancsokat is), hogy ki-ki elkészíthesse legszemélyesebb Felhasználói Módú Linux változatát, amivel például hálózati szolgáltatások végezhetőek.

Célok

Miért is tesszük mindezt, minek az elérésében reménykedünk? Ahogy azt egy korábbi írásomban említettem, a virtualizáció hasonlít a ***chroot jail*** elvéhez („elkülönített börtön”, melyben a látszólagos gyökérkönyvtár eltér a fájlrendszer tényleges gyökérkönyvtárától). Ez bezár egy processzt vagy háttérprogramot (démont) az operációs rendszerkörnyezet egy részébe, ezáltal igencsak megnehezítve a támadó dolgát, aki hatalma alá szeretné vonni a teljes operációs rendszert. Ezt akkor sem fogja tudni megtenni, ha sikerül rést találnia a processz vagy háttérfolyamat működésében.

A ***chroot*** leszűkíti tehát a folyamat hatáskörét a gazdagép (valódi) fájlrendszerének egy részére.

A virtualizáció ehhez hasonló, de ott a leszűkítés egy teljesen virtuális számítógépre történik, amely a (valódi) gazdagépen fut. Ez azt jelenti, hogy a merevlemez, a memória és a kernel, sőt még az olyan rendszereszközök, mint a hálózati és hangkártyák is csak virtuálisan léteznek a számára. A ***Felhasználói Módú Linuxban*** ez úgy valósul meg, hogy a vendég (virtuális) rendszermag a gazdagépen levő (valódi) rendszermag felhasználói terében lévő processzként fut.

Mivel mind a vendég, mind a gazda rendszermag ***Linux*** kernel, a ***Felhasználói Módú Linuxszal*** történő virtualizáció gyors és hatékony.

A vendég kernelnek nem kell adminisztrátori jogosultságokkal futnia a gazdagépen, ezért a vendég rendszeren valamely háttérfolyamatot megtörő és ebből (a vendég rendszeren) adminisztrátori jogokat megszerző támadó is csak alacsony jogosultságú héj elérést tud szerezni a gazdagépen.

Ez nem jelenti azt, hogy lehetetlen adminisztrátori jogosultságot szerezni a gazdagépen. Ha a támadónak valóban sikerül odáig eljutnia, hogy héj elérése legyen a gazdagépen, ezt erősebb jogosultságúvá alakíthatja, ha talál valami alkalmas sebezhetőséget a gazdagép rendszermagjában vagy valamely, ott futó felhasználói programban. (Emlékeztetőül: semmilyen helyi sebezhetőség nem pusztán ***helyi***, ha már hálózatba van kötve a rendszer!)

De ezzel eléggé megnehezítettük a támadó dolgát efféle sebezhetőségek kiaknázására, különösen akkor, ha ez a rés nincs ott a vendég (virtuális) rendszerben.

Most már sejthető, hogy hogyan érdemes terveznünk: a vendég rendszernek oly csupasznak kell lennie, amennyire csak lehetséges. Semmilyen felesleges programot nem kell rá telepítenünk, semmi felesleges erőforrást nem kell rendelkezésére bocsátanunk, hogy ezáltal a minimálisra csökkentsük a támadásra alkalmas felületet. Ha például ***DNS*** szerverként működik a gépünk, akkor csak az alapvető hálózati támogatást, ***BIND***-ot (vagy más ***DNS***-kiszolgáló csomagot) kell feltennünk rá, és lehetőleg semmi mást. Sem ***X Window*** rendszert, sem Apache webszervert – semmi többet, mint amire a szoros értelemben vett ***DNS***-kiszolgálónak szüksége van.

Ha igazán paranoiások vagyunk, akkor még az ***SSH*** kiszolgálót is kihagyhatjuk, ehelyett egy virtuális soros konzolról intézve az adminisztrációt. Bár, őszintén szólva, egy olyan megoldás, hogy csak egy adott ***IP***-címről: a gazdagépéről fogadja el a szerver az ***SSH*** kapcsolatot, talán egy ésszerű középutat jelenthet. Az is egy járható út, hogy ***SELinux*** alól futtatjuk a ***Felhasználói Módú Linuxot***; azonban ennek tárgyalása már túlmutatna jelen sorozatunk keretein.

Tekintsünk egy olyan helyzetet, hogy egy „bástya” szervernek több hálózati szolgáltatást is futtatnia kell: ***Apache***-ot, ***BIND***-ot. Ekkor érdemes két különböző vendég rendszert futtatni ugyanazon a gazdagépen. Az egyiken csak az Apache (és függőségei) fussanak, a másikon hasonlóképpen: csak a ***BIND*** legyen telepítve. Ily módon egy ***BIND*** biztonsági rés nem vezet közvetlenül weboldalunk elcsúfításához, vagy egy ügyetlenül elkészített webes alkalmazás megtörése nem nyit utat a ***BIND*** meghamisításához.
Összefoglalásként: két fő tervezési elvünk az lehet, hogy minden hálózati szolgáltatásnak külön virtuális gépet érdemes készíteni, valamint, hogy ezek mindegyike legyen olyan puritán és biztonságos, amennyire csak lehetséges. Remélhetőleg ez olyan szakaszolt, bástyaerős szerverfelépítéshez vezet, amely a lehető legtöbb absztrakciós réteget kihúzza a támadó és a teljes adminisztrátori birtokba vétel közé.
Jelen cikksorozat hátralevő részeiben egy olyan példát fogok bemutatni, amelyben egyetlen vendég rendszer működik: ez ***BIND***-ot futtat. Mind a gazdagép, mind a vendég rendszer ***Debian 3.1***-esen alapul, mivel a ***Debian*** igen népszerű az ***FML*** vendég rendszerek terén (sokszor nevét is adja más, lecsupaszított telepítéshez is, akárcsak a ***Slackware***). Ezzel együtt a továbbiak alkalmazhatóak egyéb disztribúciókra is, mind gazdagép, mind a vendég rendszer tekintetében.

A teendőink:

1. Gazda rendszermag készítése ***Felhasználói Módú Linux*** vendégrendszerek futtatására optimalizáltan
2. A gazdagépen futó egy (vagy több) vendég-rendszermag készítése
3. A vendég rendszerek számára készült gyökér-fájlrendszer testre szabása
4. Vendég rendszerek futtatása, konfigurálása és lezárása biztonságos ***DNS*** szolgáltatáshoz

A gazdagép előkészítése

Először is meg kell győződni arról, hogy a gazdagépen megfelelő kernel fut-e. Nagy valószínűséggel újat kell majd fordítani a rendszermagot. Néhány ***Linux*** disztribúció rendszermagja alapértelmezetten tartalmazza a ***Felhasználói Módú Linuxot***. Nem biztos azonban, hogy a kiválasztott disztribúcióba a ***SKAS*** foltot is beleforditották. (A ***SKAS*** betűszó: ***Separate Kernel Address Space***, elkülönített kernel-címtér.) Az alapértelmezett kernelben sajnos általában nincs ***SKAS*** támogatás. Annak ellenére, hogy a kernel forrásában már a 2.6.9-es verziótól kezdve jelen van az ***FML*** támogatás, a ***SKAS*** foltot még külön tartják karban (***Linus Torvalds*** ugyanis nem fogadta el a beillesztését).
Fontos szerepe van a ***SKAS*** foltnak. Nagyban megnöveli az ***FML*** teljesítményét és biztonságát, ha a vendég rendszer kernelét az egyéb folyamataitól elkülönített címtérben futtatjuk (mint ahogy ezt a gazdagép is teszi). A ***Felhasználói Módú Linux SourceForge***-on található weboldala részletesen elmagyarázza a ***SKAS*** folt szükségességét. (Lásd a cikkhez tartozó forrásokat. ***Lényege, hogy segítségével a gazdagép kernelének nem kell foglalkoznia az FML processzeivel; az FML nem adja át a processzadatokat – a ford.***)

A kernelek és a vendég rendszerek elkülönítése

A ***Felhasználói Módú Linux***, a ***VMware*** és más virtualizációs rendszerek szövegkörnyezetében sajátos jelentése van a gazdagép és vendég rendszer kifejezéseknek. A gazdagép az a rendszer, ami a virtualizációs környezetet futtatja; azaz végső soron egy vagy több virtuális gép gazdájaként működik. A vendég rendszerek olyan virtuális gépek, amelyek a gazdagép felett futnak.
A gazda kernel és a vendég kernel említésekor tehát tudnunk kell, hogy a vendég kernel a gazda kernelen fut. A ***Felhasználói Módú Linuxban*** a gazda kernel a normál kernel, melyet célszerűen az adott hardverre optimalizáltak (***Intel x86***, ***IBM PowerPC*** stb.), és amelybe beleforditották a ***Felhasználói Módú Linux*** támogatást (beleértve az opcionális ***SKAS*** foltot).
A vendég kernelt úgy kell lefordítani, hogy fusson a virtuális „hardveren”: az ***UM*** (***Felhasználói Módú***) architektúrán. De ezen túl ennek a kernelnek ***nincs*** szüksége a SKAS foltra vagy a Felhasználói Módú Linux támogatásra. Hacsak nem akar valaki a vendég rendszeren belül is újabb vendég kernel(eke)t futtatni, amely, bár lehetséges (ezt beágyazásnak (***nesting***) hívják), ennek tárgyalása túlmutat ezen cikk keretein.
Három összetevőt különíthetünk el minden ***FML*** virtuális gép példányban: a vendég kernelt, a vendég gyökér fájlrendszert és a ***COW*** fájlt (***Copy On Write***, íráskor létrejövő másolat). A gyökér fájlrendszer lehet egy lemez-képmásfájl, mely magán a kernelen kívül a virtuális gép minden fájlját tartalmazza. A vendég kernel futtatásakor a gyökér fájlrendszer fájlja pontosan úgy csatolható fel, mint ahogy bármely más képmásfájlt, például egy ***CD .iso*** fájlt is csatolni szoktunk. Ahogy a ***CD-ROM***, ez is csak olvasható lesz. Minden változtatás, ami a virtuális fájlrendszert érinti az ***FML*** munkamenet során (beleértve fájlok törlését és létrehozását is), a ***COW*** fájlban tárolódik. Ennek a „varázslatnak” köszönhetően lehetséges több példányban is futtatni ugyanazt a vendég kernelt akár ugyanazzal a vendég gyökér fájlrendszerrel együtt – elegendő pusztán annyit tenni, hogy különböző ***COW*** fájlt nevezünk ki az egyes ***FML*** példányok számára.
A kernel forrásának beszerzéséhez a legjobb választás valószínűleg az adott disztribúcióhoz elkészített kernelforrás-csomag telepítése. Arra azonban figyelni kell, hogy legalább 2.6.9-es rendszermag álljon rendelkezésre, mert az ***FML*** támogatás csak ettől kezdve tekinthető biztonságosnak. A ***Debian 3.1*** még ***2.6.8***-as rendszermagot használ, így arra az elhatározásra jutottam, hogy a hivatalos ***Debian*** kernelforrás-csomag helyett letöltöm a ***2.6.17***-es változatot a ***kernel.org*** webhelyről. A ***kernel-package*** csomag segítségével ebből a hivatalos kernelforrásból normál ***Debian*** (***.deb***) csomagot lehet készíteni.
A rendszermag forrásán kívül még a ***SKAS*** foltra is szükségünk lesz, melyből a legfrissebb elérhető a ***Blaisorblade*** webhelyen

(lásd a források között). A folt letöltésekor szem előtt kell tartani, hogy mely sorszámú kernel forrását szeretnénk megfoltozni.
Debian gazdagépemen kicsomagoltam a letöltött hivatalos kernelforrást a */usr/src/linux-2.6.17.3* könyvtárba, majd azon nyomban át is neveztem */usr/src/linux-2.6.17.3-host* nevűre, finom utalásféleképpen a készületben lévő gazdarendszerre. Ezek után a *SKAS* tarlabdát is a helyére tettem (a *skas-2.6.17-rc5-v9-pre9.patch.bz2*-t a */usr/src*-be). Meg kellett változtatnom a */usr/src/linux-2.6.17.3-host* könyvtár jogosultságát egy nem-adminisztrátori felhasználóéra azon lelkiismereti elv érvényesítése miatt, hogy „csak akkor legyünk adminisztrátorok, ha nagyon muszáj": a kernel felépítését ugyanis egy normál felhasználó is vígan el tudja végezni.
Az alábbi parancsokat adtam ki adminisztrátorként:

```
host:/usr/src/# tar -xjvf
➥./linux-2.6.17.3.tar.bz
host:/usr/src/# mv ./linux-
➥2.6.17.3 ./linux-2.6.17.3-host
host:/usr/src/# chown mick
➥./linux-2.6.17.3
host:/usr/src/# su - mick
```

A *SKAS* folt „odavarrásához" nem-adminisztrátor felhasználóként a */usr/src/linux-2.6.17.3-host* könyvtárba lépve az alábbi parancsot adtam ki:

```
host:/usr/src/linux-2.6.17.3
➥-host$ bunzip2 -c ../skas-
➥2.6.17-rc5-v9 -pre9.patch.bz2
➥| patch -p1
```

Majd ugyanebből a könyvtárból kiadtam egy `make menuconfig` parancsot is. A rendszermag Felhasználói Módú Linux számára történő konfigurálásához az alapértelmezett értékek többnyire megfelelőek – „csak" arra kell figyelni, hogy az adott gép hardveréhez is illeszkedjenek. Ezen kívül bölcs dolog kétszer is ellenőrízni az alábbiakat:

- A „Processzor típusa és egyéb jellemzői" részen a */proc/mm* támogatása legyen bekapcsolva.

- A „Hálózati beállítások" alatt az *IP:tunneling* (*IP-alagút*) és a *802.1d Ethernet Bridging* (*Ethernet híd létrehozása*) is szükséges. Ha az *iptables* által is szabályozni szeretnénk a vendég rendszerünk viselkedését, akkor ellenőrizni kell a „Hálózati csomagszűrés" szekciót; be kell kapcsolni az alábbiakat: *Core Netfilter Configuration, IP: Netfilter Configuration* és *Bridged IP/ARP packets filtering* opciókat *(Általános Hálózati Szűrő Beállítása, IP:Hálózati Szűrő Beállítása, Hídkapcsolt IP/ARP csomagszűrés)*.

- A „Hálózati eszközök"-nél kapcsoljuk be a *„Universal TUN/TAP device driver"* támogatást (*„Általános TUN/TAP eszköz meghajtó"*).

- Végül bizonyosodjunk meg arról, hogy bele legyen drótozva (ne pedig csak modulként elérhetően létezzen) azon fájlrendszerek támogatása, ami a rendszerünk gyökér-fájlrendszere (például *ext3*, *ReiserFS*).

Ettől a ponttól kezdve ugyanúgy folytatódik minden, mint bármely más kernelfordításkor: adjuk ki a

```
make bzImage; make modules;
```

parancsokat, majd adminisztrátorként a

```
make modules, make
➥modules_install
```

és

```
make install
```

utasításokat. *Debian* esetén használható a `make-kpkg` parancs is (azaz *„make Kernel-PacKaGe"*, *„csinálj kernelcsomagot"*) ugyanilyen céllal, majd `dpkg`-vel telepíthető a kapott kernel csomag.
Ha elkészült friss-ropogós gazdarendszermagunk, indítsuk újra számítógépünket. A lábra álló gazdarendszer már alkalmas lesz *Felhasználói Módú Linux* vendég rendszerek futtatására.

A vendég-rendszermag előállítása

Ezek után, hogy készen áll gazdagépünk az *FML* használatára, még szükségünk van vendég-rendszermagra is. Ez egy fokkal egyszerűbb, mint a gazda-rendszermag előállítása, mivel nem kell foglalkoznunk a *SKAS* folttal.
Először térjünk vissza ahhoz a könyvtárhoz, ahol a kernelforrásunk tarlabdáját tartjuk, és újra tömörítsük ki. Talán emlékszünk még arra, hogy ezek után egy átnevezéssel utaltunk a kernelforrás hovatartozására – ezt most is hasonlóképpen fogjuk tenni. A gazda- és vendég-rendszermagunk könyvtárait elkülönítetten kell használnunk.
Debian tesztrendszeremen a */usr/src/linux-2.6.17.3* könyvtárba tömörítettem ki a tarlabdát, majd ezt a könyvtárat */usr/src/linux-2.6.17.3-guest*-nek neveztem át. Újfent gyengítsük le a könyvtár jogosultságát egy kiskaliberű felhasználóéra, és váltsunk bele.
Ezúttal eltekinthetünk a SKAS folt beépítésétől. Mivel kernelünk nem egy normál (például *x86*-os) architektúra számára készül, hanem egy speciális *UM* (*User Mode, Felhasználói Módú*) architektúra számára, a forrás megfelelő előkészítéséhez javaslom az alábbi parancsok kiadását:

```
host:/usr/src/linux-2.6.17.3-
➥guest$ make mrproper ARCH=um
host:/usr/src/linux-2.6.17.3-
➥guest$ make defconfig ARCH=um
host:/usr/src/linux-2.6.17.3-
➥guest$ make menuconfig ARCH=um
```

A `make mrproper` parancs mindenféle konfigurációs előzményt és objektumot eltakarít a kódkönyvtárból; a `make defconfig` elkészíti a friss alapkonfigurációs fájlt az *um* architektúra számára; végül a `make menuconfig`, elvárásainknak megfelelően, lehetővé teszi eme konfigurációs fájl finomhangolását.

Különösen is figyeljünk az alábbiakra:

- Egyszerűbb az élet monolitikus kernellel (modulok betöltögetése helyett). Ha netán mégis ez lenne a cél, érdemes elolvasni a *User-Mode Linux HOWTO* (*Felhasználói Módú Linux Hogyan*) 2.2-es fejezetét (lásd a források közt).

- A „Processzor típusa és egyéb jellemzői" részen győződjünk meg (kétszer is) arról, hogy architektúraként az *um*-et (*User Mode,*

Felhasználói Módú) adtuk-e meg, s hogy a */proc/mm* támogatás be van-e kapcsolva.

- A „Hálózati beállítások” alatt az ***IP:tunneling*** (***IP-alagút***) és a ***802.1d Ethernet Bridging*** (***Ethernet híd létrehozása***) legyen bekapcsolva.

- A „Hálózati eszközök”-nél kapcsoljuk be a „***Universal TUN/TAP device driver***” támogatást („***Általános TUN/TAP eszköz meghajtó***”).

- Az összes lehetséges hardver kernelmodult kapcsoljuk ki; ez a kernel virtualizált hardveren fog futni, így nem lesz szükség semmi egzotikus drótnélküli LAN támogatásra, sem őskori párhuzamos portra stb.

A friss konfigurációs fájl elmentése után az alábbi parancsot mezei felhasználóként adjuk ki (tehát nem adminisztrátorként):

```
host:/usr/src/linux-2.6.17.3-
➥guest$ make linux ARCH=um
```

Talán feltűnt, hogy ***nem*** javasoltam a (***zip*** vagy ***bz2*** módon) tömörített rendszermag-képfájl (***image***) előállítását. Gondoljunk arra, hogy ez a kernel úgy fog futni, mint bármely más, normál felhasználó által kiadott parancs – nem érdemes tehát tömöríteni. A kész rendszermag a forráskód főkönyvtárában helyezkedik el a fordítás befejeztével (***/usr/src/linux-2.6.17.3-guest*** a fenti példában), és nemes egyszerűséggel ***linux*** lesz a neve. Ezt célszerű átnevezni valami kifejezőbbre, például ***uml-guestkernel-2.6.17.3***-re, valamint érdemes át is mozgatni valahova máshová, pl. a ***/usr/local/uml/*** könyvtárba. Ne ijesszen meg senkit a vendégrendszermag mérete. A bithalmaz mérete nagyrészt szimbólumokból áll, ami nem töltődik be a memóriába a végrehajtáskor.

Tanulság

Gazdagépünk immár teljes mértékben támogatja a ***Felhasználói Módú Linuxot***, és egy futtatásra készen álló vendég-rendszermagot is elkészítettünk. Már csak egy megfelelő gyökérfájlrendszer képmásfájl beszerzése vagy elkészítése van hátra, melyet a vendég-rendszermaggal fogunk tudni használni. Innen folytatjuk következő alkalommal.

Linux Journal 2006., 151. szám

Mick Bauer
(darth.elmo@wiremonkeys.org)
az Egyesült Államok egyik legnagyobb banki hálózatában felelős a hálózati biztonságért. Ő a szerzője az O'Reilly által kiadott Linux szerverek biztonsága, Linux Server Security c. mű második kiadásának. (Korábban a Biztonságos szerverek építése Linuxon, Building Secure Servers With Linux címet viselte a könyv.) Időnként előad információ-biztonsági konferenciákon. Ő a szerzője a „Hálózatmérnöki Polkának” („Network Engineering Polka”).

A CIKK FORRÁSAI

➲ www.linuxjournal.com/article/9260

D-Link
Building Networks for People

Hálózati eszközök (2. rész)
Vezeték nélküli eszközök telepítése és használata

A sorozat előző részében egy átfogó képet adtunk a vezeték nélküli hálózatok rejtelmeiről. következik a Wi-Fi eszközök telepítése és a kapcsolat beállítása a NetworkManager segítségével.

Szakácskönyvünk ezúttal úgy kezdődik, hogy: *„Fogjunk egy Wi-Fi eszközt, és csatlakoztassuk a számítógéphez”*. Ez az esetek kétharmadában egy *USB*-s vagy *PCMCIA*-os kártya betolása a gép oldalába. Kivétel a *PCI*-os eszköz, amihez szét kell szedni a gépet, de ilyen kártyát általában olyan felhasználók választanak, akik – legalábbis a hardver tekintetében – tudják mit csinálnak. Beépített eszközzel rendelkező notebooktulajdonosok a fenti lépést kihagyhatják, de a beépített *Wi-Fi* adapterek teljesítménye és sebessége rendszerint nem éri el az ismertebb márkájú vezeték nélküli kártyákét.
Ha ezzel a rövid mozzanattal megvolnánk, mindenekelőtt telepítenünk kell a csatlakoztatott eszközt.
Talán még emlékszünk az előző cikkben emlegetett három ügyféloldali vezeték nélküli hálózati kártyára. Mi most ezeken keresztül mutatjuk be a módszert, ám a folyamat minden vezeték nélküli kártya esetében ugyanez. A tesztkörnyezetünk egy *Ubuntu Linux 6.10*-ből áll, *2.6.17-11*-es rendszermaggal, és a lehető legfrissebb csomagokkal, amely egy notebookon, illetve egy mezei asztali *PC*-n fut. Nem muszáj pont ezt a terjesztést pont ezen változatát használni, az *Ubuntu* mellett számos más jó rendszert találunk (*Slackware*, *Fedora*, *SuSE*, stb.), de törekedjünk mindig a legfrissebb változat használatára, bármilyen terjesztés is található a számítógépen.

A hálózati eszközről röviden

Az eszközöket a *D-Link Magyarországtól* kaptuk kipróbálásra. Mindhárom a cég legújabb, *Wireless N* termékcsaládjába tartozik, amely már kapható a hazai boltokban. Közös jellemzőjük, hogy a legújabb (elfogadás alatt álló) *IEEE 802.11n* szabványnak megfelelően működnek, amely lehetővé teszi a *802.11g* szabványhoz képest akár tízszer gyorsabb adatátvitelt, és kétszer akkora hatósugarat. Az n szabvány ereje az úgynevezett *MIMO* (*multiple input, multiple output*) technológia, amely egyszerre több antennát használ a jeltovábbításhoz. Ezzel próbálja meg elérni a rádióhullámok többutas terjedéséből és visszaverődéséből adódó jelgyengülés, jelirány változás minimalizálását, vagyis kiküszöböli a több irányból visszaverődő, és kisebb-nagyobb késéssel érkező jelek okozta problémákat. A *802.11n* még csak tervezet, de a gyártók részéről hatalmas a nyomás, nem titkoltan a *HD* videók vezeték nélküli továbbítását szeretnék minél hamarabb elérni az új szabvány segítségével, ezért is jelentetni meg több gyártó a hivatalos véglegesítés előtt az erre épülő termékét.

Telepítés

Eszközeink *NDIS*-megfelelőek, tehát a windowsos meghajtóprogramokat fogjuk használni. Ellátogathatunk a gyártó honlapjára, de úgy láttam, hogy a cikk írásának pillanatában a legújabb elérhető vezérlőprogram megegyezik a mellékelt *CD*-n találhatóval, így az egyszerűség kedvéért alkalmazzuk ezt. Mindhárom eszközhöz szükségünk van az *NdisWrapper* csomagra, amelyet *Debian* alapú rendszereken az

```
apt-get install ndiswrapper-
➥utils
```

paranccsal telepíthetünk. Tegyük is meg rögtön a legelején, aztán nézzük, hogyan tovább. (Minden lépést rendszergazdaként hajtsunk végre!)

Ha a már meglévő eszközünket csatlakoztatjuk a számítógéphez, mielőtt a telepítési folyamatba bonyolódnánk, bizonyosodjunk meg arról, hogy valóban nem ismerte fel a rendszerünk egyből az eszközt. Ezt a legegyszerűbben parancssorból ellenőrizhetjük ha kiadjuk az `ifconfig` parancsot. A parancs listázza a számítógépben található hálózati eszközöket. Alapesetben látni fogunk egy eth0 eszközt, valamint egy `lo` eszközt, és ha ezen felül találunk `wlan0`, `wifi0`, `ath0`, vagy hasonló nevű eszközt, akkor a kártya működik.

D-Link RangeBooster DWA-142, avagy a zöld szemes ostoros

A dobozt kibontva egy érdekes formatervű eszközt találunk: egy jégkoronghoz hasonló szerkezet, amelyből egy ***USB*** kábel lóg ki. Az ***USB*** csatolófelületnek köszönhetően nagyon egyszerű csatlakoztatni, ennélfogva azok számára a legjobb választás, akik nem szeretik szétszerelni a masinájukat.
Ha rádugtuk a gépünk egyik ***USB*** kapujára, adjuk ki héjból az `lsusb` parancsot, amellyel a számítógéphez csatlakoztatott ***USB***-s eszközök listáját kérhetjük le. Ha látjuk az alábbi sort az eszközök listájában, akkor minden rendben van:

```
Bus 001 Device 004: ID
➥07d1:3b10 D-Link System
```

Az eszköz tehát látszik a ***Linux*** rendszermag számára, másoljuk át a ***CD***-n lévő windowsos vezérlőprogramokat egy könyvtárba.

```
sudo mkdir /usr/lib/wlan
cp /cdrom/setup/Drivers/
➥Driver/* /usr/lib/wlan/
```

Adjuk meg a vezérlőprogram elérési útját az `ndiswrapper` számára:

```
ndiswrapper -i /usr/lib/wlan/
➥netmw245.inf
```

Ellenőrizzük, hogy valóban összepárosította-e az eszközt a meghajtóprogrammal

```
ndiswrapper -l
```

Ezt kell látnunk:

```
Installed drivers:
netmw245                driver
➥installed, hardware present
```

azaz a hardver és a vezérlőprogram is a helyén van.

Az ***Ubuntu 6.10***-es változatában lévő ***ndiswrapper*** csomag sajnos hibás, ki kell cserélnünk egy régebbi változatra, amit az itt leírt parancsok rendszergazdaként történő kiadásával meg is tehetünk:

```
apt-get install ndiswrapper-utils-1.8
mv /usr/sbin/ndiswrapper /usr/sbin/ndiswrapper_bak
cp /usr/sbin/ndiswrapper-1.8 /usr/sbin/ndiswrapper
```

Ezután már próbálkozhatunk a

```
modprobe ndsiwrapper
```

utasítással.

Töltsük be az `ndiswrapper` modult, amely az eszközünk vezérlését végzi

```
modprobe ndiswrappert
```

Ha mindent jól csináltunk, zölden villogni kezd az eszközön elhelyezett ***LED***, készen vagyunk. Jó hír, hogy a legközelebbi csatlakoztatásnál már nincs szükségünk végigjárni a folyamatot, a csatlakozás után az eszköz azonnal működik.

D-Link RangeBooster DWA-547

Egy ***PCI***-os eszközről van szó, azoknak ajánljuk, akik nem rajonganak érte, hogy még egy eszköz lógjon ki a gépükből, foglalva ezzel is egy ***USB*** kaput. A dobozt kinyitva megtaláljuk a három mellékelt antennát, de addig ne csavarozzuk fel, amíg be nem szereltük az eszközt, különben nehéz dolgunk lesz, hogy beférjen. A három antenna természetesen a már említett ***MIMO*** technológia velejárója, így éri el a sebesség- és hatótávolság növekedést. Ha beszereltük a kártyát, kapcsoljuk be a számítógépet, kezdődhet a telepítés.
Ellenőrizzük, hogy a kártyánk látszik-e a rendszermag számára, és szerezzük meg az eszköz azonosítóját:

```
lspci -v
```

Az utolsó egy ***Atheros Inc. (Unknown device 0023)*** típusú eszköz lesz, amelynek hardverazonosítója: `168c:3a6b`. Ezt a későbbiekben még fel fogjuk használni.
Másoljuk át a ***CD***-n lévő windowsos vezérlőprogramokat egy könyvtárba.

```
sudo mkdir /usr/lib/wlan
cp /cdrom/setup/Drivers/
➥Driver/* /usr/lib/wlan/
```

Adjuk meg a vezérlőprogram elérési útját az `ndiswrapper` számára:

```
ndiswrapper -i /usr/lib/wlan/
➥net5416.inf
```

Ellenőrizzük, hogy valóban összepárosította-e az eszközt a meghajtóprogrammal:

```
ndiswrapper -l
```

Ezt kell látnunk:

```
Installed drivers:
net5416          driver
➥installed, hardware present
```

azaz a hardver és a vezérlőprogram is a helyén van.
Rendeljük hozzá a vezérlőprogramot a megfelelő hardvereszközhöz a már megszerzett azonosító alapján:

```
ndiswrapper -d 168c:3a6b
➥net5416
```

Töltsük be az `ndiswrapper` modult, amely az eszközünk vezérlését végzi

```
modprobe ndiswrappert
```

Ellenőrizzük, hogy működik-e az eszköz

```
ifconfig
```

A megjelenő listában legalul egy wlan0 címkéjű eszközt kell lássunk.

D-Link RangeBooster DWA-645

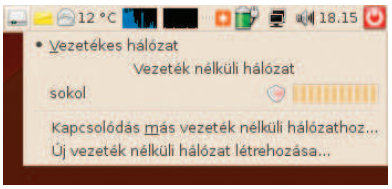
Ez az eszköz hozza meg legin-kább a ve-zeték nélküli internetezés szabadságát. Egy notebookba szánt *PCMCIA* csatolófelü-lettel rendelkező *PC*-kár-tyáról van szó. Ha notebookunk van (és nincs beépített wifi kártyánk), akkor az ilyen típusú eszköz az egyetlen jó választás. A számítógép oldalába tolva csak az antenna látszik ki a gépből, innen indul a telepítés, amely teljes mértékben megegyezik a *PCI*-os eszköz telepítésével, ugyan-azt a vezérlőprogramot használja, ugyanúgy *PCI*-os eszközként látjuk. Egyetlen különbség a kártya hard-verazonosítója, amely itt `07d1:3a09`, ezt kell behelyettesíteni az előbb részletezett képletbe.
Itt is igaz az, hogy a következő csatlakoztatásnál már nem kell külön telepíteni az eszközt, azonnal működésre kész.

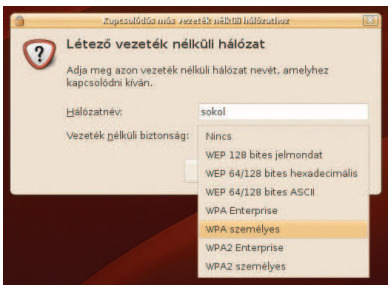
A telepített eszközök beállítása

A következő lépés elvégezni a kártyánk hozzáférési ponthoz történő beléptetését. Ez történhet otthon, a helyi hálózathoz, vagy egy kávézóban, egyetemen, cégnél, stb. Konkrét példát adni itt nem lehet, mivel minden hálózat egyedi jellemzőkkel bír, ehelyett inkább megismerkedünk egy okos kis alkalmazással, aminek a neve ***NetworkManager Applet***, és a *Gnome* felület tálcájára ül rá. Feladata a hálózati kapcsolatok kézben tartása, kapcsolódási profilok kezelése, átállás egyik hálózatról a másikra, stb. *Ubuntu* alatt az

```
apt-get install network-
➥manager-gnome
```

paranccsal telepíthető, de a következő változatban már alapértelmezetten benne lesz.
A telepítést követően legegyszerűbb, ha újraindítjuk a grafikus felhasználói felületet, és az ***applet*** már ott lesz a tálcánkon.
Ha vezeték nélküli hálózatra szeretnénk váltani, kattintsunk bal gombbal az ikonra. A lenyíló menüben látszanak a környezetünkben elérhető hozzáférési pontok és mellettük a jelerősség. Ha olyan hálózathoz szeretnénk csatlakozni, amelynek nyilvános a beállított szolgáltatásazonosítója (*SSID*), akkor nyert ügyünk van, mert látszódnia kell a listában. Rákattintva egy párbeszédablakot kapunk, amelyen meg kell adni a kapcsolat jellegét, illetve a kapcsolódáshoz szükséges esetleges jelszót. A kapcsolat jellege többnyire kimerül a hozzáférési pont által használt titkosítási módszerben. Ettől függ az a bizonyos jelszó is. Annak érdekében, hogy illetéktelenek ne férjenek hozzá a hálózathoz, kulccsal/jelszóval védik, ezt a jelszót/kulcsot fogja kérni a kiválasztott titkosításnak megfelelően. Ezekről bővebben a vezeték nélküli hálózatok biztonsági kérdéseiről szóló cikkben értekezünk.
Ha beírtuk a szükséges adatokat, a program megkezdi a csatlakozást. Bontja az előző kapcsolatot, ha volt ilyen, és felépíti az újat.
A folyamat állását egy animáció mutatja, sikeres a kapcsolódás, ha mind a helyi, mind a távoli pont jelzőfénye zöldre vált, ezután egy üzenetet kapunk a sikeres (vagy épp sikertelen) kapcsolódásról, és a megjelenő ikon a vezeték nélküli hozzáférési pont jelerősségét mutatja egy grafikon segítségével.
Ha a hozzáférési pont által sugárzott *SSID* rejtett, nem fogjuk látni a hálózatot, ebben az esetben ismernünk kell a nevét, és az Új vezeték nélküli kapcsolat vagy Kapcsolódás vezeték nélküli hálózathoz menüpontra kattintani (mindkettő ugyanazt csinálja).
A megjelenő párbeszédablak ugyanaz, leszámítva, hogy nincs





kitöltve a hozzáférési pont neve.
Ha kitöltöttük, innen lásd az előző bekezdést.
Ezek után az egyes hálózatok között úgy váltogathatunk, hogy kiválasztjuk a ***NetworkManager*** menüjéből az ott található hálózatokat. Megjegyzem, nagy szükségünk nincs rá, ugyanis amikor az egyik kapcsolat megszűnik (például kihúzzuk a gépből a hálózati kábelt), automatikusan átáll a másik kapcsolatra, ha az elérhető, és természetesen fordítva. Ilyenkor az addig aktív hálózatot mindig leállítja, így nekünk semmivel sem kell foglalkozni, egyetlen konfigurációs fájlhoz sem kell hozzányúlni, s mindeközben az ***applet*** folyamatosan tájékoztat bennünket az állapotváltozásokról.
A már beállított kapcsolatokat a hozzáférési pontok által sugárzott *SSID*-hez rendelve eltárolja, többször nem szükséges megadni, a jelszavakat pedig az alapértelmezett kulcskarikánkra fűzi.
A fentiekhez természetesen ismernünk kell a kapcsolódási pontok jellegét, amiről általában a szolgáltatás igénybevételének helyén (szálloda portáján, gyorsétterem pultjánál, stb.) tájékozódhatunk (az otthoni hálózatunkat pedig feltehetően ismerjük). Ezekről még fogunk szólni a biztonságról szóló fejezetben is.
A következő részben beállítjuk az otthoni hálózatunk központi elemét, egy *ADSL*/Kábel routerrel egybeépített hozzáférési pontot, megosztjuk vele az internetet, és csatlakozunk hozzá a vezeték nélküli kártyánkkal. ■

A SUSE 10.0 biztonsági szolgáltatásai

A SUSE terjesztés a biztonság híve – rengeteg biztonsággal kapcsolatos eszközt kínál.

Az évek során tapasztalhattuk, hogy egyre több és jobb biztonsági szolgáltatást építenek be a kedvenc ***Linux*** terjesztéseinkbe. A terjesztésfüggő biztonságtudatosság számos formában jelenik meg, például

- A biztonságnövelő alkalmazások elérhetőségében.
- A telepítő parancsfájlok „megerősítő" működésében
- A foltok kezelésének módjában
- A hálózati alkalmazások alapértelmezett beállításaiban

Ebben a hónapban egy három cikkből álló sorozatot indítunk a ***SUSE Linux***, a ***Debian GNU/Linux*** és a ***Red Hat Enterprise Linux*** terjesztésre jellemző biztonságáról. Ezzel a három terjesztéssel szereztem a legtöbb tapasztalatot, és sokak szerint ez a három a legnépszerűbb.
Kezdjük a ***SUSE 10.0***-val. A ***SUSE*** egy általános célú, kereskedelmi kiadású ***Linux*** terjesztés, amelyet 32 és 64 bites ***Intel*** környezetekhez fejlesztettek.
A ***Novell*** tulajdonába került ***SUSE*** eredetileg ***Németországból*** származik, és jórészt ma is ott fejlesztik. Számos különböző ***SUSE*** termék létezik: a ***SUSE Linux***, amely kiskereskedelmi forgalomban elérhető „személyes" változat; a ***SUSE Linux Enterprise Server***, egy „vállalkozás-szintű" változat, amely közvetlenül a ***Novelltől*** szerezhető be; és az ***OpenSUSE***, amely gyakorlatilag megegyezik a ***SUSE Linuxszal***, de telepítő hordozók (csak az interneten keresztül lehet telepíteni), nyomtatott kézikönyvek és telepítési támogatás nélkül.
A cikk alapjául a ***SUSE Linux 10.0***, vagyis a kereskedelmi „személyes felhasználású" változat szolgál. Az itt leírtakat elvileg azonos módon lehet alkalmazni az ***OpenLinux 10.0*** változatban, és a ***SUSE Enterprise*** változataiban is jórészt helytállónak kell lenniük. Az ***Enterprise*** változatok feltételezhetően további biztonsággal kapcsolatos csomagokat és szolgáltatásokat tartalmaznak.

A SUSE Linux 10.0 telepítése

A rendszerbiztonság a telepítésnél kezdődik. Itt van először lehetőségünk lényeges döntéseket hozni arról, hogy milyen szerepet szánunk a rendszernek, hogy milyen programrendszert futtatunk majd, illetve hogy milyen beállításokat határozunk meg a rendszerben. Ezért nem árt, ha a telepítési folyamatnál kezdjük a SUSE biztonságának tárgyalását.
A ***SUSE*** összes változata a ***YaST***-ot (***Yet Another Setup Tool***) használja, a rendszer először telepítéséhez és a folyamatos rendszerfelügyelethez egyaránt. Az évek során a ***YaST*** egyszerű ***RPM*** felhasználói felületből (front end) egy moduláris, átfogó felügyeleti eszközzé vált, amely nem csak alacsony szintű rendszerprogramok, hanem összetett kiszolgáló alkalmazások, például az ***Apache*** és a ***Postfix*** beállítására is alkalmas.
Rövidesen bővebben is szót ejtünk a ***YaST***-ról, az operációs rendszer első telepítésekor azonban a legsürgősebb probléma annak eldöntése, hogy melyik programcsomagot telepítsük.
Ha valaki a biztonságot tartja a legfontosabbnak, akkor ez kellemes probléma. A ***SUSE Linux 10.0*** széles választékot kínál a biztonsági alkalmazásokból – ezek közül válogathatunk.
Megítélésem szerint ezek az alkalmazások két csoportra oszthatók: rendszer-biztonság-alkalmazásokra és biztonság-pásztázó alkalmazásokra. Az előbbiekhez tartoznak a hatékony biztonsági szolgáltatásokkal rendelkező általános célú alkalmazások – a ***Postfix*** a legkézenfekvőbb példa –, valamint az olyan alkalmazások, amelyek fő célja az, hogy biztonsági vezérlőeszközöket biztosítsanak más alkalmazások vagy a mögöttes operációs rendszer számára – tipikusan ilyen például a ***tcwrappers***. Az *1. Táblázat* a ***SUSE Linux 10.0*** biztonságot erősítő csomagjait sorolja fel.
Az *1. Táblázat* hosszú csomaglistája valójában csak bizonyos személyes kedvenceket és néhány ***SUSE***-ra jellemző választást tartalmaz. A ***SUSE***-ban sokkal, sokkal több biztonsági eszköz található, például a ***tcpd (tcpwrappers)***, az ***openssl***, a ***chkrootkit***, a ***sudo*** és a ***wipe***. A ***SUSE Linuxba*** épített csomagok teljes felsorolását a ➲ ***www.novell.com/products/linuxpackages/professional/index_all.html*** címen tekinthetjük meg.
Azon kívül, hogy biztonságossá tesszük azt a rendszert, amelyre a ***SUSE***-t telepítjük, érdemes lehet más rendszerek vagy akár teljes hálózatok biztonságát is ***SUSE*** rendszerrel megerősíteni. A ***SUSE*** remekül alkalmazható erre a feladatra. A *2. Táblázat* olyan ***SUSE Linux 10.0*** csomagokat mutat be, amelyeket biztonsági pásztázásra használhatunk. Ügyeljünk, hogy ezeket a csomagokat (talán a ***Snort*** kivételével) soha nem szabad internetre kapcsolódó kiszolgálón telepíteni. Ilyen környezetben a csomagok a támadóknak nagyobb hasznot jelentenek, mint nekünk. A programok pásztázását olyan rendszerről kell végezni, amely egyébként nincs veszélyben.
Annak, aki most ismerkedik a ***SUSE***-

1. táblázat *Néhány SUSE Linux 10.0-hoz tartozó biztonságnövelő csomag*

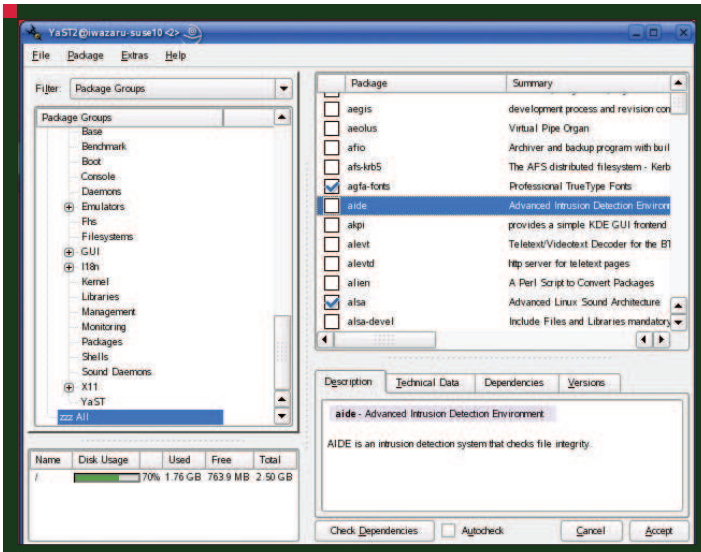
Csomag neve	Leírás
aide, fam	A fájlok épségét ellenőrzik - mindkettő a hasonlít a Tripwire-re
bind-chrootenv	Automatikusan létrehoz egy chroot környezetet, amelyben a BIND (a DNS démon)biztonságosabban futhat.
clamav, antivir	Vírusölő csomagok – a clamav teljesen ingyenes, de az antivir kereskedelmi program (személyes használatra ingyenes).
cracklib	Olyan könyvtárak és eszközök, amelyek megakadályozzák, hogy a felhasználók könnyen kitalálható jelszavakat válasszanak.
gpg, gpg2, gpa	A GNU Privacy Guard (gpg) egy sokoldalú és elterjedt e-mail-, illetve fájltitkosító eszköz.
ipsectools, openswan	IPsec alapú virtuális magánhálózat építésére szolgáló eszközök.
openldap, freeradius	Nyílt forráskódú hitelesítő démonok.
proxy-suite	A SUSE által fejlesztett biztonsági FTP-proxy.
seccheck	A SUSE által testreszabott cron parancsfájlok, amelyek különböző biztonsági ellenőrzéseket futtatnak naplófájlokon, a rendszerállapoton és egyebeken, majd e-mail jelentéseket küldenek.
subdomain-utils, subdomain-profiles, mod-change-hat és más csomagok	Az AppArmor egy kötelező hozzáférés-vezérlés (Mandatory Access Control – MAC) rendszer, amely korlátozza bizonyos bináris fájlok viselkedését. A SUSE ezt használja az SELinux helyett, amelyre nagyon hasonlít.
squid, SquidGuard	A Squid népszerű HTTP/HTTPS proxy. A SquidGuard hozzáférés-vezérlést és egyéb biztonsági szolgáltatásokat kínál.
SUSEfirewall	A SUSE kényelmes felhasználói felülete a Linux netfilter-hez illetve iptables-hez.
syslog-ng	A syslogd-nél jóval hatékonyabb fejlett rendszernaplózó. A syslog-ng a SUSE alapértelmezett naplózója.
tinyca2	Az OpenSSL felhasználói felülete, amely a tanúsítvány hatóságok kezelésére szolgál.
yast2-firewall	Tűzfal szolgáltatás.
vsftpd	Very Secure FTP Daemon
xen, FAUmachine, uml-utilities, bochs	A Xen, a FAUmachine, a User Mode Linux és a BOCHS virtuális gép (virtual machine) környezetek.

val, tudnia kell, hogy a ***YaST*** alapértelmezésben a ***Selections (Válogatás)*** szűrőt (nézetet) alkalmazza, amelyben a rendelkezésre álló csomagok csak egy kisebb részét kínálja fel. Ha valami, ami számunkra szükséges – például a ***nessus-core*** – nem szerepel ebben a nézetben, a ***Pacakage Groups (Csomagcsoportok)*** szűrő segítségével a kategóriák egy teljesebb halmazát jeleníthetjük meg. Amennyiben az összes csomagot egyetlen listában, betűrendben szeretnénk látni, egyszerűen válasszuk a szűrő Package Groups beállítását, majd kattintsunk a *zzz* ***All (Összes)*** csoportra (*1. ábra*). A szűrő ***Search (Keresés)*** beállításával név vagy kulcsszó alapján kereshetünk csomagokat.

2. táblázat *A SUSE Linux 10.0-ban használt biztonsági pásztázók*

Csomag neve	Leírás
ethereal, tcpdump	Kitűnő csomagfigyelő
fping	Elárasztásos ping (több célpontú ping)
john	John the Ripper – jelszófeltörő eszköz (gyenge jelszavakat azonosítására használható jogszerűen).
kismet	Vezeték nélküli LAN-figyelő
nessus-core, nessu-libraries	Nessus – általános célú biztonsági pásztázó
snort	Kiváló csomagfigyelő, csomagnaplózó és betörésérzékelő rendszer

Az összes programcsomag kijelölése és telepítése után a ***YaST*** lehetővé teszi a rendszergazda jelszó beállítását és az első (nem rendszergazda) felhasználói fiók létrehozását. A ***SUSE*** alapértelmezésben a ***Blowfish***-t



1. ábra A YaST összes rendelkezésre álló csoportjának megtekintése

használja a jelszavak titkosításához, a ***YaST*** pedig ellenőrzi a begépelt jelszó bonyolultságát. (A túl egyszerű jelszavakat a támadók könnyen kitalálhatják vagy nyers erővel feltörhetik.)
A helyi tűzfal-parancsfájlok (alaphelyzetben engedélyezett), valamint az ***SSH*** és a ***VNC*** távoli héj démonok (alaphelyzetben mindkettő tiltott) engedélyezésére is lehetőségünk van. Megjegyzendő, hogy az utóbbi kettő közül az ***SSH*** a legalkalmasabb a bástya gépek (megerősített internetes kiszolgálók) felügyeletére – egyebek mellett azért, mert – hacsak nincs valamilyen nagyon különleges és nagyon megkerülhetetlen indokunk – bástyagépeken nem szabad ***X Window*** rendszert használni. Ki kell emelni, hogy a ***YaST*** tökéletesen fut szöveges (***ncurses***) üzemmódban, az X változattal pontosan megegyező modulokkal és lehetőségekkel. Ráadásul, a ***SUSE***-hez tartozó ***VNC*** távoli asztal változat, a ***tightvnc*** csak a hitelesítési adatokat titkosítja, a munkamenet-adatokat nem.
Azt is meg kell említeni, hogy a telepítéskor nincs lehetőségünk a helyi tűzfalbeállítások testreszabására. Kezdetben egy alapértelmezett parancsfájl érvényesül, amely egy egyszerű „összes kimenő művelet engedélyezése, összes nem helyben kezdeményezett bejövő forgalom tiltása” szabályrendszert biztosít. Más szóval, az alapértelmezett ***SUSEfirewall*** parancsfájl tökéletesen megfelel a legtöbb asztali rendszeren, de kiszolgálón történő használathoz nem alkalmas. A ***YaST Firewall (Tűzfal)*** modulját futtatva ezt később megváltoztathatjuk.
A ***YaST*** ezután a következő módszereket biztosítja a nem rendszergazda felhasználók hitelesítésére:

- helyi /etc/passwd fájl (alapértelmezett)
- LDAP
- NIS
- Samba (Windows NT tartományok)

A ***SUSE Linux 10.0*** az ***Active Directory*** hitelesítést és lehetővé teszi, a ***Kerberoson*** keresztül.
A hitelesítési módszer kiválasztása után létrehozhatjuk az első nem rendszergazda felhasználói fiókot. Ügyeljünk, hogy az ***Automatic Logon (Automatikus bejelentkezés)*** lehetőséget hagyjuk tiltott állapotban, kivéve, ha a rendszer biztonsági követelményei valóban nagyon alacsonyak. Ha engedélyezzük ezt a lehetőséget, a számítógép induláskor automatikusan belépteti a nem rendszergazda felhasználót. (Csak a nyilvános – ***kiosk*** – típusú rendszerek esetében tudom elképzelni, hogy ez hasznos lehet.)

3. táblázat ***Biztonsághoz kapcsolódó YaST modulok***

YaST csoport	Modul neve	Leírás
Software (Programok)	Online Update (Hálózati frissítés)	Kézi és automatikus programfrissítések beállítására szolgál
	Software Management (Programkezelés)	Csomagok telepítésére és eltávolítására szolgál.
	Virtual Machine Installation (Virtuális gép telepítés – XEN)	Virtuális gépeket hoz létre a Xen 3 virtuális gép környezet számára
System (Rendszer)	/etc/sysconfig Editor (Szerkesztő)	Démonok indítási paramétereit szerkeszti
	System Services (Runlevel) – Rendszerszolgáltatások (Futási szint)	Indító parancsfájlok kezelésére szolgál
	Powertweak	További rendszermag-paramétereket, például TCP timewait aljzatokat állít be
Network Services (Hálózati szolgáltatások)	DNS Server (DNS kiszolgáló)	A BIND beállítására szolgál

3. táblázat *folytatás*

YaST csoport	Modul neve	Leírás
Network Services (Hálózati szolgáltatások)	HTTP Server (HTTP kiszolgáló)	Az Apache beállítását végzi
	LDAP Client (LDAP ügyfél)	LDAP hitelesítést és kereséseket határoz meg
	Mail Transfer Agent (Levéltovábbító ügynök)	Beállítja a Postfixet vagy a Sendmailt
	Kerberos Client (Kerberos ügyfél)	Kerberos hitelesítés, többek között Active Directory beállítására szolgál
	Remote Administration (Távoli felügyelet)	Beállítja a TightVNC-t
Novell AppArmor	Various (Vegyes)	Az AppArmor kötelező hozzáférés-vezérlés kezelésére szolgál meghatározott bináris fájlokon.
Security and Users (Biztonság és felhasználók)	Firewall (Tűzfal)	Netfilter/iptables beállítások kezelésére szolgál.
	Local Security (Helyi biztonság)	Meghatározza a jelszavak bonyolultságát illetve hosszát, a jelszó-öregedést, fájl-hozzáférés sémákat és egyéb más rendszerbiztonsági paramétereket.
	Group Management (Csoportkezelés)	Csoportfiókok létrehozására, szerkesztésére és törlésére szolgál.
	User Management (Felhasználó-kezelés)	Felhasználói fiókok létrehozására, szerkesztésére és törlésére szolgál (valójában megegyezik a Group Management modullal, ami kettős célt szolgál).

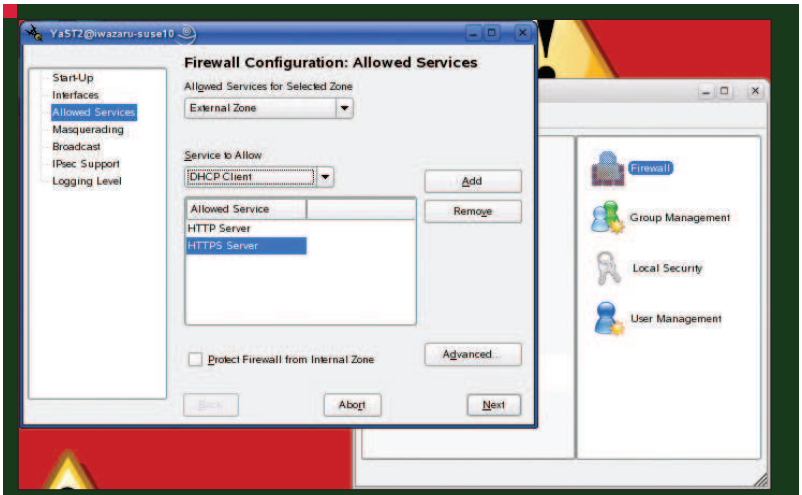
Készen is vagyunk: befejeződött a ***SUSE*** telepítése. A biztonságtudatos rendszergazda feladatai azonban nem érnek véget itt.

Biztonsághoz kapcsolódó YaST modulok

Az újonnan felhúzott ***SuSE Linux*** rendszer első indításakor azonnal jelentkezzünk be a jogokkal nem rendelkező felhasználóval, és hívjuk meg a ***YaST***-ot. Ha ezt a ***KDE***-ből vagy a ***GNOME***-ból tesszük, a rendszer automatikusan kéri a rendszergazda jelszót, de szövegkonzolos munkamenetben a `su -c` parancsot kell alkalmazni az ***/sbin/yast*** fájl hívásához.
Ahogy korábban említettem, a ***YaST*** számos beépített biztonsági szolgáltatással rendelkezik. A *3. Táblázat* a rendszerbiztonság szempontjából kifejezetten fontos ***YaST*** modulokat sorolja fel.
A fenti ***YaST*** modulok közül az ***Online Update*** az egyik legfontosabb. Érdemes azonnal alkalmazni az automatikus foltletöltések, valamint – ha a rendszerben nem működik változtatásvezérlő folyamat – az automatikus folttelepítések beállításához. A ***YaST Online Update*** volt az első, egy nagyobb ***Linux***-terjesztés által kínált automatikus folt eszköz, és még ma is az egyik legjobb. A használatával élvezhetjük a ***SUSE*** azon kiváló gyakorlatának előnyeit, amely során friss és jól ellenőrzött foltokat bocsát rendelkezésre.
A ***Firewall*** modul (*2. ábra*) szintén rendkívül hasznos, különösen akkor, ha valaki nem szívesen hoz létre és kezel saját tűzfal-parancsfájlokat (elismerem, hogy kevés az olyan ember, aki hozzám hasonlóan izgalmasnak és szórakoztatónak találja ezt).
A ***Group/User Management*** hasonlóképpen feleslegessé teszi, hogy valaha kézzel kelljen szerkeszteni a ***/etc/group*** vagy a ***/etc/passwd*** fájlt.
A ***Virtual Machine*** modul és a ***Novell AppArmor*** csoport szintén különösen említésre méltó. Olyannyira, hogy érdemes némi időt szánni a ***SUSE*** virtuális gép illetve kötelező hozzáférés-vezérlés rendszereinek részletesebb tárgyalására.

Virtuális gépek a SUSE Linuxban

Lehet, hogy vannak, akik emlékeznek a ***„The Future of Linux Security" (A Linux biztonság jövője – Linux Journal, 2005. augusztus)*** című cikkemre, amelyben virtuális gép környezeteket és hypervisorokat (más néven biztonsági megfigyelőket) vizsgáltam a rendszerbiztonság fontos, új irányvonalaként. Ha valaki nem emlékezne, összefoglalva az volt az indok, hogy a ***MAC*** (kötelező hozzáférés-vezérlés) sémákat, például az ***SELinuxot*** sokan túl bonyolultnak tartják. Egyszerűbb megoldás, ha minden nagyobb alkalmazást vagy szolgáltatást saját virtuális gépen futtatunk. Így ha az a virtuális gép, amelyben például a ***Sendmail*** fut, veszélyeztetett, az ***Apache2***-t fizikailag azonos vason futtató virtuális



2. ábra A YaST Firewall modulja

gép nem kerül azonnali vagy közvetlen veszélybe.
A virtuális gépek ezért hatékony és könnyen érthető módszert biztosítanak a bonyolult alkalmazások egymástól történő elkülönítéséhez.
A ***SUSE Linux*** pedig nem kevesebb, mint három különböző virtuális gép eljárást tartalmaz.
A ***SUSE*** „technológiai előzetesként" biztosítja a ***Cambridge University*** berkeiből származó ***Xen 3*** környezetet. Amennyire meg tudom állapítani, ez csupán annyit jelent, hogy mivel a ***Xen 3*** kiforratlan és esetleg ingatag alkalmazás, a ***SUSE*** egyszerűen nem akar hiú reményeket kelteni az emberekben a használhatóságát illetően – a ***Xen 3 SUSE Linux 10.0***-ban szereplő változata nem különleges előzetes–, illetve próbaváltozat vagy ehhez hasonló. A ***Xen 3*** a ***Linux***, a ***FreeBSD***, a ***NetBSD*** és a ***Plan9*** „vendég-" (virtuális) rendszereket támogatja.
Másik lehetőségként a ***FAUmachine*** virtualizáló környezet olyan ***RPM*** csomagokat tartalmaz, amelyek a ***SUSE 9***, a ***Debian 3.0***, az ***OpenBSD 3.5/3.6*** és a ***Red Hat 9*** vendégrendszer támogatását teszi lehetővé. A ***FAUmachine*** egyik előnye a ***Xen 3***-mal szemben az, hogy a ***FAUmachine***-nál a vendégrendszerek magjai nem rendszergazda (jogokkal nem rendelkező felhasználó) hozzáféréssel futnak a gazdarendszeren.
A ***User Mode Linux*** egy újabb virtualizáló környezet, amelyet a ***SUSE Linux 10.0*** az ***uml-utilities*** csomagon keresztül biztosít.
A ***FAUmachine***-hoz hasonlóan a vendég rendszermagok rendszergazda jogok nélkül futnak.

Novell (Immunix) AppArmor

Ennek ellenére nem mindenki mondott le a ***MAC***-alapú rendszerbiztonságról, ráadásul a ***SUSE*** az ***Immunix AppArmor*** (más néven ***Subdomain***) alkalmazásának megvásárlásával és újracsomagolásával elegánsan lefedte ezt a területet. Az ***AppArmor az SELinuxhoz*** hasonlóan lehetővé teszi bizonyos folyamatok viselkedésmódjának korlátozását, hasonló eredménnyel, de hatékonyabban, mint ha ***chroot*** ketrecekben futtatnánk azokat. (Érdemes megjegyezni, hogy ugyan a ***SUSE***-ben rendelkezésre áll a ***libselinux*** csomag, az alapértelmezett rendszermagban pedig szerepel az ***SELinux*** működés, hivatalosan a ***SUSE*** Linux mégsem támogatja az ***SELinuxot***.
Az ***SELinux SUSE Linuxban*** történő futtatásához a ➲ ***www.cip.ifi.lmu.de/~bleher/selinux*** címen elérhető csomagokra van szükség.
A ***subdomain-docs*** csomagban található ***/usr/share/doc/packages/subdomain-docs/ug_apparmor.pdf*** dokumentum az ***AppArmor*** használati útmutatója, ami az ***AppArmor*** beállításával és használatával kapcsolatos összes tudnivalót leírja. Egyelőre elég annyit elmondani, hogy ha egyszerűen futtatjuk a ***YaST AppArmor Control Panel (Vezérlőpult)*** modult, és engedélyezzük az ***AppArmort***, betöltődik egy alapértelmezett profil, amely számos elterjedt démon és parancs – például ***netstat***, ***ping***, ***traceroute***, ***firefox***, ***evolution***, ***gaim***, ***syslogd***, ***acroread***, ***ethereal***, ***appropos***, ***procmail***, ***postfix*** (***smtpd*** és számos más), ***Apache2*** (***httpd2-prefork***), ***nscd***, ***identd***, ***ntpd***, ***sshd*** és ***squid*** – beállításait tartalmazza.
Ez az ***AppArmor*** egy korlátozott szolgáltatásokkal ellátott változata, tehát nyilvánvalóan a teljes, 1250 dolláros US változatban elérhető szolgáltatásoknak csupán egy részét biztosítja. Számomra viszont nem teljesen világos, hogy pontosan mi a különbség – minden, amit a ***SUSE Linux 10.0*** változattal kipróbáltam, rendesen működött, tehát nem valószínű, hogy ez egy jelentősen lebutított kiadás lenne. Elképzelhető, hogy a teljes változatban több előre beállított alkalmazás található.

Összegzés

A cikkben nem szerepel a ***SUSE Linux 10.0*** összes biztonsági szolgáltatása. Nem beszéltem arról, hogy sok alkalmazás mennyire biztonságos alapértelmezett beállításokkal rendelkezik (általában elég biztonságosak – amikor csak lehetséges, a démonok nem rendszergazda hozzáféréssel futnak, az olyan hálózati figyelők, mint az ***sshd*** általában alapértelmezésben tiltottak, és még sorolhatnám).
A ***SUSE Linux*** ezen változata valóban nagyon biztonságbarát. Ne feledjük azért, hogy a valódi biztonság kulcsa a mi kezünkben van – a ***SUSE*** biztonsági lehetőségeinek csak kis része valósul meg, amíg azokat nem állítjuk be vagy legalább engedélyezzük saját magunk! Remélhetőleg ez a cikk segít az olvasóknak képet alkotni, hogy milyen nagyok ez a lehetőségek.
A következő hónapban a ***Debian 3.1***-ről lesz szó. Addig is mindenki vigyázzon magára!

Linux Journal 2006., 144. szám

Mick Bauer
(darth.elmo@wiremonkeys.org)
Hálózatbiztonsági mérnök az Egyesült Államok egyik legnagyobb bankjánál. Az O'Reilly kiadó gondozásában megjelent Linux Server Security 2. kiadásának (korábbi címén Building Secure Servers With Linux) szerzője, időnként előadóként vesz részt informatikai biztonsági konferenciákon, valamint a „Network Engineering Polka" zeneszerzője.

A Logical Volume Manager

Vajon hányan ültek már büszkén az újonnan beüzemelt szerverük előtt azon gondolkodva, hogy ezzel a rendszerrel bizony évekig semmi komolyabb gondja nem lesz. Három vagy még több merevlemez, örökké 80% alatt terhelt fájlrendszerek...

Aztán telnek-múlnak a hónapok, szabad hely pedig egyre csak fogy, és rendszergazdánk lassan már elgondolkodik, hogy mihez is kezdjen a szervere csordulásig telt fájlrendszereivel, és hogyan is tudná felhasználni az alig használtakat. Mennyivel könnyebb lenne az élet, ha egy eredetileg túlméretezett partíció rovására növelhetnénk a helyhiánnyal küszködő partíciónkat, vagy egyszerűen csak kibővítenénk egy újonnan beüzemelt merevlemezzel a megterhelt fájlrendszerünket, és mindezt persze újra telepítés, megkockáztatom: újraindítás nélkül... Röviden erre (is) jó az *LVM*.

Unix örökség

Az *LVM* (*Logical Volume Manager*) ötlete még a jó öreg *Unix* rendszereken valósult meg elsőként (azóta szinte az összes operációs rendszer támogatja). A kilencvenes évek elején már a *HP-UX* része volt, később az *OS/2*-ben illetve a *Solarisban* (*Solaris Volume Manager*) is megvalósult. *AIX*, **BSD* mellett természetesen *Linuxon* is elterjedt eszközzé vált.

Az LVM az mi az?...

Azt azért tudni kell, hogy nem csak *Linux*/*Un*x* környezetben használatos az *LVM*. Lefogadom, sokan találkoztak már az *LDM* (*Logical Disk Manager*) fogalommal, amit a *Microsoft* a *Windows* 2000-ben vezetett be. Vagyis a „*Basic*", *DOS*-ból örökölt lemezkezelés mellett megjelent a dinamikus lemezkezelés, amit eredetileg egy *VERITAS* nevű cég fejlesztett ki Unix alá, majd ültette át *Windows NT*-re (későbbiekben licencelte a *Microsoft* a 2000-es változathoz). Funkcióiban találhatunk hasonlóságot a *Linux* alatt fellelhető *LVM* és az *LDM* között, de megvalósításban teljesen különbözőek (igaz, az *LDM* is a fizikai partíciókon tárol minden leíró információt (*metadata*) a logikai kötetről, sőt ugyanúgy *UUID* (*Universally Unique Identifier*) alapján azonosítja őket). Vicces, de meg kell jegyeznem, hogy az *LDM* óta képes a *Windows* is „mountolni" partíciókat. Ami késik nem múlik.
Térjünk vissza az *LVM*-hez. Az első *LVM* meghajtó a 2.3.47-es kernelben kapott helyet, ezért az ebből a fából kiemelt 2.4.x sorozat már aktívan tartalmazta. A 2.6-os kernelfa már az *LVM2*-t (is) támogatja, melyet a kernelbe bekerült *device-mapper* valósít meg.
LVM segítségével a gépünkben található háttér tárakat akár egyben, egy logikai háttértárként használhatjuk, melyet utólag particionálhatunk igényeinknek megfelelően. Hatalmas előnye, hogy akár *RAID* tömbre is felépíthető az *LVM*, sőt több *RAID* tömböt is összefoghatunk egy logikai eszközzé. Új merevlemez hozzáadásakor egyszerűen csak megnöveljük partíciónk méretét, vagy akár teljes lemezterületeket mozgathatunk át rá.

Alapok

A fentebb említett előnyeit az *LVM*-nek nem túl nagy ördöngösség kihasználni, igaz, nem árt néhány alap fogalmat tisztázni, mielőtt az „építkezést" elkezdenénk.

- *PV (Physical Volume)*: Ez az alap és egyben legkisebb egysége az *LVM*-nek. A *PV* nem más, mint egy inicializált *LVM* típusú partíció. Itt még nincs logikai kapcsolat felépítve a fizikai kötetek között.
- *VG (Volume Group)*: *PV*-ket összefoglaló, névvel ellátott logikai kötetcsoport. Általában egy *VG* több, akár különböző háttér tárakon elhelyezkedő *PV*-ből épül fel.
- *LV (Logical Volume)*: Logikai kötet, tulajdonképpen egy partíció az *LVM* felett. Létező *VG*-re épül. Egy *VG*-re több *LV* is létrehozható (képzeljük el úgy, hogy a *VG* a fizikai lemez, az *LV*-k pedig partíciók).

A cikkben az *1. ábrán* látható *LVM* szervezetet fogjuk lépésről lépésre létrehozni, vagyis kezdésként négy *PV*-ből létrehozunk egy *VG*-t, majd azt kettéosztva fogunk kapni két darab *LV*-t. Ez hát a cél...

Építkezés lépésről lépésre

Ez az a pont ahol meg kell jegyeznem, hogy az itt leírtakat éles rendszerben *ne* alkalmazzuk. Csak kipróbált, agyontesztelt *LVM* konfigurációt állítsunk csatasorba, sok-sok kellemetlen adatvesztéstől mentesülve ezáltal.
Első lépés, elővesszük kedvenc partíció kezelő programunkat, és elkezdünk *Linux LVM* (0x8E) típusú partíciókat gyártani.
Ha ezzel megvagyunk, jöhet a partíciók inicializálása a `pvcreate` segítségével:

```
# pvcreate /dev/hda[5-8]
  Physical volume "/dev/hda5"
  ➥successfully created
  Physical volume "/dev/hda6"
  ➥successfully created
  Physical volume "/dev/hda7"
  ➥successfully created
  Physical volume "/dev/hda8"
  ➥successfully created
```

Jó esetben hiba nélkül létrejött a négy *PV*, jöhet a csoportba szervezés a

```
# vgcreate vgteszt /dev/hda
# [5-7]
  Volume group "vgteszt"
  ➥successfully created
```

paranccsal.
A *„VG Size”* értékéből látható, hogy maximum 9,31Gb méretű *LV*-t hozhatunk létre. Készítsünk egy 2 GB *LV*-t:

```
# lvcreate -L2048 -nlvkotet01
# vgteszt
  Logical volume "lvkotet01"
  ➥created
```

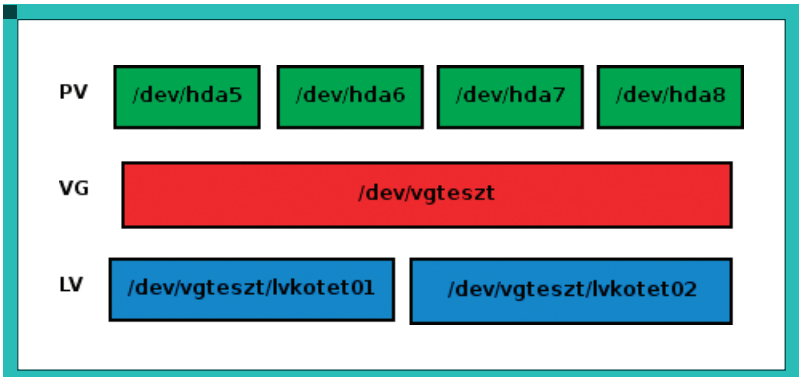
A maradék 7,3 GB *VG* kapacitáson pedig készítsük el második *LV* kötetet:

```
# lvcreate -L7,3G -nlvkotet02
# vgteszt
```

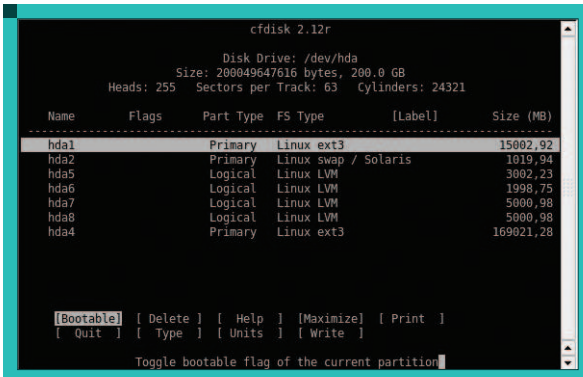
Nincs más hátra, már csak használatba kell venni újdonsült logikai köteteinket, formázzuk le őket kedvenc fájlrendszerünkkel, mondjuk használjunk *ext3*-at:

```
# mkfs.ext3 /dev/vgteszt/
# lvkotet01
# mkfs.ext3 /dev/vgteszt/
# lvkotet02
```

Gondolom, mindenkinek feltűnt, hogy a *VG* létrehozásakor „véletlenül” kihagytam a negyedik *PV*-t, a `/dev/hda8`-at. A megoldásnak az *LV*-k és a *VG* törlése, majd új, mind a négy *PV*-t tartalmazó csoport létrehozása, ami ugyan hatékony lehet üres kötetnél, de mégis van ettől egy egyszerűbb *LVM* eszköz, a `vgextend`. Ezzel az eszközzel egy már meglévő kötetcsoporthoz adhatunk hozzá újabb fizikai kötetet, ezzel is bővítve kapacitását. Szóval adjuk hozzá a hiányzó *PV*-t a *„vgteszt”* csoporthoz:



1. ábra PV-VG-LV viszony. Nem is annyira bonyolult...



2. ábra Igaz, egy merevlemezen, de elkészült a négy darab „nyers” PV kötet.

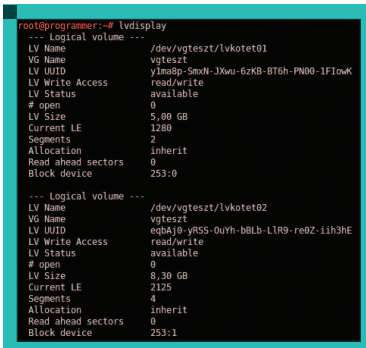
```
root@programmer:~# vgdisplay
  --- Volume group ---
  VG Name               vgteszt
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               9,31 GB
  PE Size               4,00 MB
  Total PE              2383
  Alloc PE / Size       0 / 0
  Free  PE / Size       2383 / 9,31 GB
  VG UUID               4eORU6-6rE8-S0PM-BNhX-O6FK-8qOA-EvdoXZ
```

3. ábra A létrehozott vgteszt nevű kötetcsoportról a vgdisplay parancs által kaphatunk némi információt.

```
# vgextend vgteszt /dev/hda8
  Volume group "vgteszt"
  ➥successfully extended
```

Ezek után meg kell növelnünk a logikai kötetek méreteit is. Azt szeretnénk, ha a `lvkotet01` mérete 2 GB-ról 5 GB-ra növekedne, míg az `lvkotet02` méretét 1 GB-tal szeretnénk növelni:

```
# lvextend -L5G
# /dev/vgteszt/lvkotet01
  Extending logical volume
  ➥lvkotet01 to 5,00 GB
  Logical volume lvkotet01
  ➥successfully resized
# lvextend -L+1G /dev/
# vgteszt/
# lvkotet02
```



■ *4. ábra* Az lvdisplay segítségével ellenőrizzük munkánkat.

```
  Extending logical volume
  ➥lvkotet02 to 8,30 GB
  Logical volume lvkotet02
  ➥successfully resized
```

Már majdnem kész vagyunk, de a fájlrendszerek átméretezése még hátravan. Ha a `resize2fs`-nek nem adjuk meg paraméterként az új méretet, akkor a fájlrendszert magában foglaló partíció maximális méretéhez igazít:

```
# resize2fs /dev/vgteszt/
# lvkotet01 && resize2fs
/dev/vgteszt/lvkotet02
```

Elvileg kész. A

```
# mount /dev/vgteszt/lvkotet0x/
# elérési/út
```

paranccsal már használatba is vehetjük *LV* kötetünket.

Rombolunk, majd újjáépítünk

Egy szép nap észrevesszük, hogy a `smartd` csúnyákat írogat log fájlunkba, és rájövünk, hogy az egyik, *LVM* alatt zakatoló merevlemez bizony-bizony a végét járja. Persze nem ijedünk meg, elegáns mozdulattal elővesszük tartalék meghajtónkat a fiókból (ha nincs, akkor eszeveszetten lavírozunk a legközelebbi számítástechnikai boltba), és nemes egyszerűséggel beüzemeljük haldokló szerverünkbe. Jön a már megismert `pvcreate`, majd újdonsült fizikai kötetünket a

```
# vgextend vgteszt /dev/hda9
  Volume group "vgteszt"
  ➥successfully extended
```

paranccsal hozzáfűzzük létező *LVM* csoportunkhoz. Most már csak az adatokat kell átmozgatunk a beteg kötetről az újra:

```
# pvmove /dev/hda6 /dev/hda9
  /dev/hda6: Moved: 8,2%
  /dev/hda6: Moved: 17,0%
  /dev/hda6: Moved: 25,4%
  ...
  /dev/hda6: Moved: 96,7%
  /dev/hda6: Moved: 100,0%
```

A `pvmove` nem kapkodja el, PV méretétől függően akár kávézni is elmehetünk. Ha a parancs futtatásakor az

```
  Insufficient free space: 715
  ➥extents needed, but only 24
  ➥available
  Unable to allocate temporary
  ➥LV for pvmove.
```

hibaüzenethez hasonlót kaptunk, akkor biztosak lehetünk benne, hogy a régi *PV extent*-jei nem férnek el az újonnan csatolt fizikai köteten. Az is előfordulhat, hogy a `pvmove` parancsra a

```
mirror: Required device-mapper
➥target(s) not detected in
➥your kernel
```

hibaüzenet jelenik meg. Ilyenkor jól jön, ha tudjuk, hogy a `dm-mirror` modulunk nincs betöltve. Ezen segít a

```
#modprobe dm-mirror
```

parancs. Már majdnem készen vagyunk, már csak a jó öreg betegeskedő PV-től kell megszabadulnunk:

```
# vgreduce vgteszt /dev/hda6
  Removed "/dev/hda6" from
  ➥volume group "vgteszt"
```

Ha minden jól ment, akkor merevlemez ki, selejtezés (*torrentek* hazahurcolására meg jó lehet).

Átszabjuk

Természetesen nem csak *LVM* bővítésére, hanem átméretezésére is igény lehet. Első lépés, fájlrendszereket lecsatoljuk (nekem az `/mnt/lv0x` mappába lettek csatolva):

```
#umount /mnt/lv0[1-2]
```

Jöhet a fájlrendszer csökkentése, melyre szintén a `resize2fs` parancsot használhatjuk. Mondanom sem kell, az említett példák *ext2/3* fájlrendszerre vonatkoznak, *reiserfs*, *xfs* esetén már parancsokat kell használnunk (főleg *xfs* esetén, mivel ezt a típusú fájlrendszert a *jfs*-hez hasonlóan nem lehet csökkenteni).

```
#resize2fs /dev/vgteszt/
#lvkotet01 2G
resize2fs 1.39 (29-May-2006)
Resizing the filesystem on
➥/dev/vgteszt/lvkotet01 to
➥524288 (4k) blocks.
The filesystem on /dev/
➥vgteszt/lvkotet01 is now
➥524288 blocks long.
```

Most jön a logikai kötet csökkentése/növelése a `lvreduce/lvextend` paranccsal:

```
# lvreduce -L2G /dev/
# vgteszt/lvkotet01
  WARNING: Reducing active
  ➥logical volume to 2,00 GB
  THIS MAY DESTROY YOUR DATA
  ➥(filesystem etc.)
Do you really want to reduce
➥lvkotet01? [y/n]: y
  Reducing logical volume
  ➥lvkotet01 to 2,00 GB
  Logical volume lvkotet01
  ➥successfully resized

# lvrextend -L+2G /dev/
# vgteszt/lvkotet02
  Extending logical volume
  ➥lvkotet02 to 10,30 GB
  Logical volume lvkotet02
  ➥successfully resized
```

Az `lvreduce` figyelmeztet, hogy a művelet adatvesztéssel járhat. Ezt vegyük komolyan, de mi nyugodtan folytassuk a redukálást, hiszen előtte már lecsökkentettük a fájlrendszer méretét. A `lvkotet02` fájlrendszert szintén a `resize2fs` segítségével növeljük. A biztonság kedvéért érdemes a fájlrendszereinket ellenőrizni:

```
# fsck.ext3 -f /dev/
# vgteszt/lvkotet01 ;
# fsck.ext3
# -f /dev/vgteszt/lvkotet02
```

`pvremove`	Használaton kívüli (egyetlen csoporthoz sem tartozó) fizikai köteteket tudunk "LVM-mentesíteni".
`pvresize`	Segítségével VG-hez csatolt, sőt aktív LV alatt működő fizikai kötetet tudunk átméretezni. Leginkább PV-k csökkentésére érdemes használni, bár én ha nem muszáj, nem nyúlnék éles rendszerben ehhez az eszközhöz.
`pvs`	Formázott riportot kapunk fizikai köteteinkről. A vgs parancs nagyon hasonlóan működik, használatával a VG-ről kapunk információt, míg a lvs az LV-k adatairól ad hasonló riportot.
`pvscan`	Minden partíciót megvizsgál, PV-k után kutatva, a vgscan kötetcsoportokat keres, míg a lvscan inaktív logikai köteteket.
`vgcfgbackup`	A kötetcsoportok leíró adatairól (metadata) készít másolatot szöveges fájlba általában az /etc/lvm/backup (amit a -f kapcsolóval megváltoztathatunk) mappába. Félreértések elkerülése végett, ez nem a felhasználói adatokat archiválja! Szorosan ide kapcsolódik a vgcfgrestore parancs, mely a mentett szöveges állományból állítja vissza a csoport metaadatait.
`vgchange`	Kötetcsoportok néhány tulajdonságát befolyásolhatjuk. A -a kapcsolóval a csoport elérhetőségét engedélyezhetjük/tilthatjuk, -A kapcsolóval a metadata automatikus mentését befolyásolhatjuk illetve a csoport átméretezhetőségét és a maximum LV számot is megadhatjuk.
`vgconvert`	Kötetcsoport leíró adatait konvertálhatjuk át más formátumúvá (nagyon hasznos, ha LVM1 formátumú VG-t akarunk LVM2-ként használni).
`vgmerge`	Két létező kötetcsoportot tudunk vele összekapcsolni. A csatolandó kötetcsoportnak mindenképp inaktívnak kell lennie (vgchange -a n). Hasznos lehet a -t kapcsoló, mivel így a folyamatot valós csatolás nélkül tesztelni. Ellentéte a vgsplit, ami két részre oszt egy VG-t (ne felejtsük, a VG-n található LV-t nem lehet "kettévágni").
`vgck`	A kötetcsoportok metaadatainak konzisztenciáját ellenőrizni. Örüljünk, ha nincs a programnak semmilyen kimenete.
`vgmknodes`	Ellenőrzi, hogy az aktív csoportoknak megfelelő bejegyzések léteznek-e a /dev-ben. Ha nem, létrehozza őket (nem jelent jót, ha használnunk kell).
`lvchange`	Logikai kötet aktiválásán/leállításán kívül alapvető jellemzőket állíthatunk vele.
`lvmdump`	Hasznos debug eszköz. Minden hasznos és haszontalan információt összeszed rendszerünkről (futó folyamatok, eszközlista, LVM konfiguráció stb.).

Figyelem, felvétel indul!

Rendkívül hasznos, mégis alul dokumentált funkciója az ***LVM***-nek a ***snapshot*** készítés. Segítségével a logikai köteten elhelyezkedő fájlrendszerről készíthetünk egy „pillanatfelvételt”, melyet akár archiválhatunk, természetesen az adatok konzisztenciája megőrzése mellett. Kernelünk nagyon egyszerű, de nagyszerű módon oldja meg ezt a szolgáltatást. A ***snapshot LV*** létrehozásakor igazából csak egy „tükörkép” jön létre (semmi duplikáció) az archiválandó logikai kötetről. A ***snapshot*** élete alatt bármilyen tranzakció hajtódik végre a forráson, akkor az eredeti adatok átkerülnek a ***snapshot*** kötetre, és csak ezután hagyja a kernel a módosult adatok kiírását. Így a forrás logikai kötet illetve a ***snapshot*** logikai kötet adatait összevetve a ***snapshot*** készítés időpontjában létezett állapotot kapjuk (vagyis a két ***LV*** tartalmából kernelünk készíti el a „pillanatfelvétel” tartalmát, mivel ő pontosan tudja, mely blokkok tartalma módosult). Természetesen ahhoz, hogy a ***snapshot LV***-t létrehozzuk, szabad kapacitásnak is kell lennie a ***VG***-n. Próbaképp csináljunk egy 50Mb méretű „felvételt” az `lvkotet01`-ről:

```
# lvcreate -L50M -s -n
# lvpillfelv /dev/vgteszt/
# lvkotet01
  Logical volume "lvpillfelv"
  ➥created
```

Ha most felcsatoljuk az újonnan létrejött logikai kötetet (`/dev/vgteszt/lvpillfelv`), akkor az `lvkotet01` tartalmához hasonlót találunk rajta. A ***snapshot*** méretére figyeljünk oda. Bizonyos módosításszám után könnyen megtelhet, oda a „felvétel”. Az optimális méret függ a forrás kötet nagyságáról, fájlok átlagos méretétől, felhasználók és az általuk indított tranzakciók számától. Ideális, igaz helypazarló megoldás, ha a ***snapshot*** mérete megegyezik a forrás ***LV*** méretével, így biztos, hogy soha nem fogunk abból kifutni a mentés alatt. Pazarló, de egyszerű napi mentést is képesek vagyunk létrehozni az ***LVM snapshot*** funkciójával, hiszen nincs más dolgunk, mint minden nap egy új „pillanatfelvétel”-t készíteni heti forgásban (vagyis hétfőn töröljük az előző hétfőt, majd létrehozzuk az újat), amit ***cron*** démon segítségével éjszakára is ütemezhetjük.

Ha költözik az LVM

Egyszer eljön a pillanat, mikor egy működő ***LVM*** rendszert valamilyen úton-módon át kellene költöztetni egy másik gépbe, esetleg teljesen különböző disztribúció alá. A ***PV***-k merevlemezeit szépen átköltöztetjük az új gépbe, és eljön a varázslás ideje, `pvscan`, `vgscan` stb-stb...

Ekkor jön egy újabb hasznos funkció, az export-import. A régi gépen kiadjuk a

```
# vgchange -a n && vgexport -a
```

parancsokat, melyek hatására az összes *VG* szépen megáll, a `vgexport` hatására pedig bekerül az `EXPORTED` bejegyzés az `/etc/lvm/backup/vgteszt` fájlba (ide is, meg a kötetleíróba is), magyarul a logikai kötet leválasztható rendszerükről. Nincs más dolgunk, mint a merevlemezeket átszerelni új gépünkbe, majd kiadni a

```
# vgimport -a && vgchange -a y
```

parancsokat.

Vegyes-vágott, ami kimaradt

Az *LVM* jó néhány tulajdonságáról illetve a kezelésére szánt bináris csomagban megtalálható néhány segédprogramról nem beszéltünk még, ezért jöjjön néhány (általam hasznosnak vélt) funkció.
`pvchange`: Kevés dokumentációban említik, pedig hasznos programocska. Segítségével paraméterként megadott *PV*, vagy akár az összes (`-a` kapcsoló) „kioszthatóságát” engedélyezni illetve tiltani tudjuk. A *man* oldalon is megemlítik, hogy ez „rosszalkodó” merevlemezek esetén hasznos, hiszen nem tudjuk a letiltott *PV*-ket újra felhasználni. Akinek nehezére esik ezeket a programokat megjegyezni, annak ajánlom az `lvm` parancsot, hiszen az összes fent említett programocska az `lvm` szimbolikus láncait. Csak gépeljük be hogy `lvm`, és már láthatjuk is az elérhető funkciók listáját.

Végszó

Manapság nagy divat lett az *EVMS* (*Enterprise Volume Management System*), mely igazából a *LVM* leváltására készült, de azért nem szabad lemondani az *LVM*-ről sem (amúgy is az *EVMS* kezeli az *LVM* köteteket is, így később is át lehet állni az új megoldásra). Ha egyszer megszoktuk a *PV, LV, VG* elnevezéseket, akkor már biztonságosan és könnyedén leszünk képesek ezt a technológiát használni. Az *LVM* egyik súlyos problémája, hogy hihetetlenül alul dokumentált projekt, ráadásul a bináris segédprogramok üzenetei is legtöbbször semmit mondóak (ezért írtam le a legtöbb program kimenetét, sokszor nehéz észrevenni, hogy valami nem stimmel). Remélem, használható anyagot sikerült összehoznom, és még jobban remélem, hogy sok *Linux* szerető ember hasznára válnak az itt leírtak.

Gráma Tibor
(tibor.grama@hoya.lmh.hu)

1997 óta „Linuxozik”, UHU hívő. Szabadidejében gyermekeivel és vizsla kutyáival játszik, ha éppen nem kertészkedik vagy horgászik.

KAPCSOLÓDÓ CÍMEK

- http://cs.ubc.ca/~anirbans/cs508/508-report.pdf
- http://www.sulinet.hu/tart/cikk/Rda/0/25831/1
- http://tldp.org/HOWTO/LVM-HOWTO/
- http://librenix.com/?page=LVM

Adatok, tűzön-vízen át

Jó ideje már, hogy kinőttem a hagyományos mobilrackekből. A Linux, illetve BSD operációs rendszereket folyamatos üzemre találták ki. Márpedig egy hagyományos mobilrackhez le kell állítani a gépet. SCSI merevlemezeknél ugyan van hotswap, de az a hétköznapi felhasználók számára – ára miatt – elérhetetlen. Nem marad más, mint USB-s mobilracket használni.
Vagy van jobb alternatíva is?

A válasz: igen. Körülbelül fél éve az akkori ***USB***-s mobilrack-em elektronikája megadta magát (szerencsére a merevlemezt nem vitte magával). Garancia már nem volt rá. Mostanáig vártam a cserével. Persze találtam volna azóta utódot, de gondoltam egy merészet: nekem olyan rack kell, ami tud ***FireWire***-t is kezelni.

Cipőt a cipőboltból...

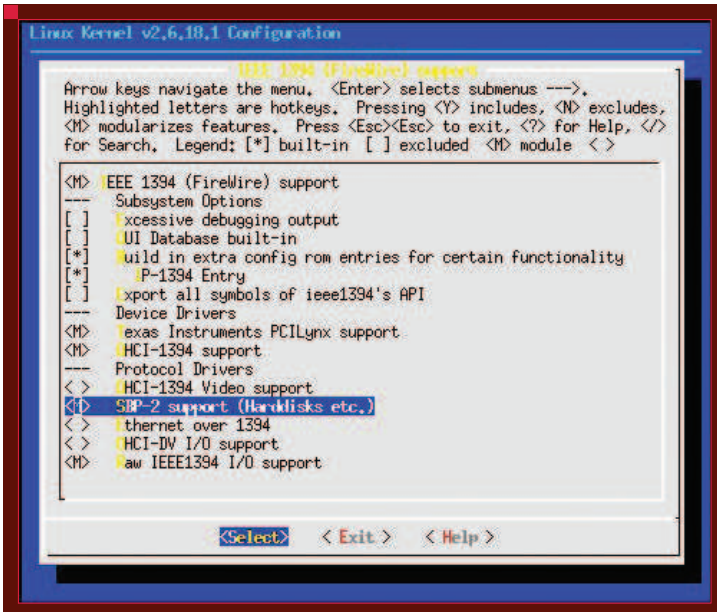
Mobilracket pedig a számítástechnikai üzletből. ***Szegeden*** elég érdekesnek indult a bevásárlás, hiszen legjobb tudomásom szerint csak pár helyen árulnak csak ***USB2.0*** és ***FireWire*** csatlakozóval ellátott racket. Az egyik boltban egyenesen elhajtottak azzal, hogy nem is létezik ilyen rack. Ha nem lett volna nálam két modell adatlapja kinyomtatva, akkor talán el is hiszem. Egy másik boltban pedig már hallottak róla, de náluk csak ***USB2.0***-eseket keresnek. Végül az egyik országos bolthálózattal rendelkező cégtől vásároltam egy ***Spire*** márkájú racket.

1. ábra 6 pólusú FireWire aljzat közelről

A meghajtóprogramokat tartalmazó ***CD***-t azt hiszem poháralátétnek fogom használni, hisz se a ***USB***, se ***FireWire*** esetén nem lesz rá szükség ***Linux*** alatt.

2. ábra Merevlemez már a helyén

3. ábra Összerakott rack, jól láthatóak a csatlakozók (USB, 2x FireWire, tápegység)



4. ábra IEEE-1394 almenü a kernelkonfiguráláskor

Egy új kernel és más semmi

Miután hazavittem, első dolgom volt belehelyezni a merevlemezt, amihez vettem. ***USB***-n kipróbáltam, rendben van az elektronika, így megejtettem a sebességtesztet is. Aztán az ***USB***-t elraktam és elővettem a ***FireWire*** kábelt. Amin meglepődtem, hogy miért van két ***FireWire*** csatlakozó az eszköz hátulján? Ez a szabványról szóló keretes részből azonban ki fog derülni. Minthogy nem biztos, hogy támogatja a ***FireWire***-t a kernelünk, elképzelhető, hogy fordítanunk kell egyet.
A ***FireWire***-hez szükséges pontokat a ***Device Drivers*** menü ***IEEE 1394 (FireWire) support*** almenüjében találjuk meg.
Belépve az almenübe érdemes az alábbi elemeket legalább modulnak bejelölni:

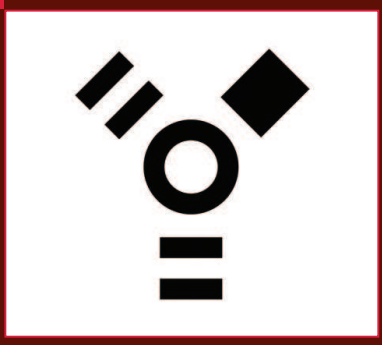
```
IEEE 1394 (FireWire) support
```

1. Lista Tipikus dmesg részlet firewire rack bekapcsolása után

```
ieee1394: Current remote IRM is not 1394a-2000 compliant, resetting...
ieee1394: Error parsing configrom for node 0-00:1023
ieee1394: Node changed: 0-00:1023 -> 0-01:1023
ieee1394: Node resumed: ID:BUS[0-00:1023]  GUID[0050770e7778a64e]
scsi6 : SBP-2 IEEE-1394
ieee1394: sbp2: Logged into SBP-2 device
ieee1394: Node 0-00:1023: Max speed [S400] - Max payload [2048]
  Vendor: Maxtor 4   Model: D040H2             Rev:
  Type:   Direct-Access-RBC                    ANSI SCSI revision: 04
SCSI device sdh: 80043264 512-byte hdwr sectors (40982 MB)
sdh: Write Protect is off
sdh: Mode Sense: 00 13 00 00
SCSI device sdh: drive cache: write back
SCSI device sdh: 80043264 512-byte hdwr sectors (40982 MB)
sdh: Write Protect is off
sdh: Mode Sense: 00 13 00 00
SCSI device sdh: drive cache: write back
 sdh: sdh1
sd 6:0:0:0: Attached scsi disk sdh
sd 6:0:0:0: Attached scsi generic sg7 type 14
```

1. táblázat ***Sebességteszt***

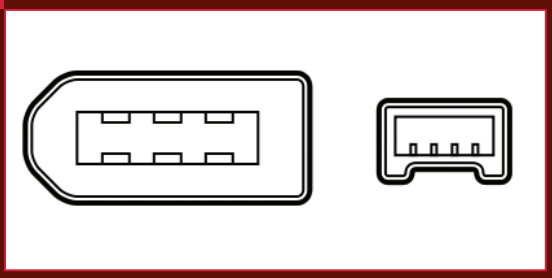
	USB2.0	FireWire
Maximális buszsebesség (gyökér eszköznél)	480 megabit/sec	400 megabit/sec
hdparm -t /dev/sdh (min/max négy mérés alapján)	21/26 megabájt/sec	25/26 megabájt/sec
valós másolás rack-re (Debian Sarge cd iso)	16 megabájt/sec	18 megabájt/sec
iso fájl másolása közben a processzorterheltség (min-max)	20-30%	10-15%



5. ábra A FireWire logó

FireWire

A ***FireWire (IEEE-1394)*** szabványt az ***Apple*** hozta létre ***1995***-ben. Az adatátviteli módot tekintve soros, akárcsak az ***USB***. A megvalósítás egyik nagy előnye, hogy a közös kábelre felfűzött eszközök képesek egymással a processzor igénybevétele nélkül is adatot cserélni (például szkenner és nyomtató). Egy rendszeren belül ***maximum 63 eszköz*** lehet, azonban további korlátozás, hogy ***egy gyökérre maximum 16 eszköz*** fűzhető fel sorosan. Az eszközök közötti ***kábelhossz nem lehet hosszabb 4.5 méternél***. Ennek azonban ellentmondani látszik, hogy 5 méteres a leghosszabb kábel, ami kapható. A szabvány ***100, 200 és 400 megabites*** névleges sebességet tud, a tényleges azonban kb. 2%-al kevesebb. Az eszközökön ***4 vagy 6 pólusú csatlakozót*** találunk. A 6 pólusúban az ***USB***-hez hasonlóan tápfeszültség is van, így kisebb eszközök tápegység nélkül is üzemeltethetőek. 4 pólusú csatlakozót jellemzően notebookokon és videókamerákon találhatunk.
A növekvő sávszélesség igényeknek megfelelően ***2003-ban*** megjelent a ***800 megabites*** szabvány ***(IEEE-1394b)*** is. Mint ahogy sejthető, a csatlakozó új dizájnt kapott (***9 pólusú*** lett), azonban a 800 megabites eszközök ***képesek*** a régebbiekkel ***kommunikálni megfelelő*** átalakító ***kábellel***.



6. ábra FireWire csatlakozók (6-os és 4-es)

6. ábra FireWire csatlakozók a valóságban (4-es és 6-os)

```
Texas Instruments PCILynx
➥support
OHCI-1394 support
SBP-2 support (Harddisks etc.)
```

Az elemeknek vannak függőségei is, ilyen például az ***I2C***, a ***PCI alrendszer*** és a ***SCSI***. Az első elem a globális ***FireWire*** támogatást kapcsolja be, a következő kettő a meghajtóprogram, az alaplapok nagy része a kettő közül valamelyikkel üzemeltethető, míg az utolsó a háttértárak (***DVD*** olvasó, merevlemez) kezelésért felel.
Ha a kernel lefordult és beraktuk ízlés szerint a ***grub***-ba vagy a ***lilo***-ba, akkor az új kernellel indítva a rack bekapcsolása után látnunk kell a ***dmesg***-ben az eszközt. Az alábbi információkat kaptuk jelenesetben: közli a buszsebességet (jelenesetben: ***Max speed, 400 megabit másodpercenként***), megadja a merevlemez típusát és méretét, valamint végül közli, hogy milyen eszközként hivatkozhatunk rá (jelenesetben: ***sdh***) Van három darab ***SATA*** merevlemezem – természetesen ***RAID***-ben –, valamint egy négy slot-os kártyaolvasó, így az ***sdh*** a legelső szabad eszköznév. Ezek után már teljesen úgy kezelhető az eszköz, mintha ***USB***-s mobilrack, kártyaolvasó vagy pendrive lenne.

Miért éri meg a FireWire?

Bizonyára felmerült a kérdés az Olvasóban, vajon miért éri meg ***FireWire***-t használni, hiszen az ***USB*** sokkal elterjedtebb. Az *1. Táblázatban* szereplő adatok a ***dmesg***-ben szereplő 40 gigabájtos ***Maxtor*** merevlemezzel és egy ***AMD Athlon64 3000+*** processzort tartalmazó rendszeren születtek.
A táblázatból jól látszik: noha ugyanolyan sebességet ad gyakorlatilag mindkét megoldás, a ***FireWire*** lényegesen kevésbé veszi igénybe a processzort. Továbbá szubjektív véleményem a ***Debian*** cd másolása kapcsán: nem ingadozik az átviteli sebesség annyira ***FireWire*** esetén, mint ***USB***-nél. A ***FireWire*** lehetősége nem drágította jelentősen az eszközt, hiszen ***USB***-vel ***4200 forint***-ért kapni az említett gyártótól mobilracket, míg ***FireWire*** és ***USB*** párossal ***6000 forint*** (az árak 2007. januári árak).

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

Hasznos információk a FireWire technológiáról:
➲ http://en.wikipedia.org/wiki/Firewire

Qmail – Öreg ember, nem vén ember

Már nem is emlékszem pontosan mikor, talán még 1998-ban telepítettem először a Qmail-t. Noha sok év eltelt már azóta, de még ma is jól használható SMTP kiszolgálót írt D.J. Berstein. A legújabb funkciók (például DSN, TLS, …) némelyike ugyan hiányzik belőle, de az alapfeladatát, a levelek biztonságos továbbítását, még mindig ugyanolyan jól végzi.

Szerencsére a *Qmail* lelkes tábora különböző foltokat (***patch***) írt, amelyekkel felokosítható a program, hogy a jelen kor igényeinek is megfelelhessen. A támogatott ***RFC***-k listája a http://cr.yp.to/qmail.html címen olvasható. Ebben az írásban egy vírus- és spam védelemmel ellátott levelező kiszolgáló kialakítását mutatom be. Szükségünk lesz az ***ucspi-tcp*** csomagra, amely egy ***inetd*** alternatíva, illetve tartalmazza az ***rblsmtpd*** programot, amellyel feketelistákat (***RBL***) használhatunk. Ennek telepítése részletesen le van írva a http://cr.yp.to/ucspi-tcp/install.html címen.

Hogy a telepítés és a használat minél kényelmesebb legyen, néhány önkéntes elkészítette a ***netqmail-1.05*** csomagot, amelyben a gyári ***qmail-1.03*** szerepel néhány ajánlott folt társaságában. Töltsük ezt le a http://www.qmail.org/netqmail/ címről, majd telepítsük az *1. listában* látható parancsokkal, amelyek létrehozzák a szükséges felhasználókat és csoportokat, beállítják az alapértelmezett hosztnevet és azt a felhasználót, aki a ***root***, a ***postmaster*** és a ***MAILER-DAEMON*** leveleit olvassa. A telepítés részletei az ***INSTALL*** dokumentációban vannak.

A ***Qmail*** alapértelmezett könyvtára a ***/var/qmail***. Ha ezt módosítani akarjuk, akkor fordítás előtt szerkesszük a ***conf-qmail*** állományt, illetve ennek megfelelően módosítani szükséges az imént létrehozott felhasználók könyvtárait.

1. Lista A qmail telepítése

```
# a szükséges csoportok és felhasználók létrehozása
groupadd nofiles
useradd -g nofiles -s /nonexistent -d /var/qmail/alias alias
useradd -g nofiles -s /nonexistent -d /var/qmail qmaild
useradd -g nofiles -s /nonexistent -d /var/qmail qmaill
useradd -g nofiles -s /nonexistent -d /var/qmail qmailp
groupadd qmail
useradd -g qmail -s /nonexistent -d /var/qmail qmailq
useradd -g qmail -s /nonexistent -d /var/qmail qmailr
useradd -g qmail -s /nonexistent -d /var/qmail qmails

tar zxvf netqmail-1.05.tar.gz
cd netqmail-1.05
mkdir /var/qmail
./collate.sh

make setup check

# ez az FQDN nevünk
./config-fast mail.aaaa.hu

# bela kapja meg a postmaster, a MAILER-DAEMON és a root leveleit
cd ~alias
echo bela > .qmail-postmaster
echo bela > .qmail-mailer-daemon
echo bela > .qmail-root
chmod 644 ~alias/.qmail*
```

A ***Qmail*** indításához készítsük el a *2. Listában* látható ***/etc/rc.d/rc.qmail*** héjprogramot (***shell script***), amely elindítja mind a helyi levéltovábbítást, mind pedig a ***qmail-smtpd*** programot. A ***qmail-smtpd root*** felhasználóként fog futni, legfeljebb 100 (alapértelmezésben 40) konkurens kapcsolatot tud kiszolgálni, az ***sbl-xbl.spamhaus.org*** és a helyi ***rbl.aaaa.hu*** feketelisták adatbázisával veti össze a kliens IP-címét, illetve a ***syslog*** naplóba kerül minden üzenet. A ***softlimit*** paraméter után

megadott szám (20 M) a ***Qmail*** számára elérhető memória méretét korlátozza. Az eredeti értéket (2 M) azért növeltem meg, hogy a később ismertetett vírusellenőrzés se üsse be a fejét memória limitbe.
Ha ***Maildir*** formátumú postafiókot akarunk, akkor a */etc/rc.d/rc.qmail* állományban a cseréljük le a „*./Mailbox*” sztringet „*./Maildir/*”-re, és adjuk ki minden felhasználó nevében a

```
cd ~;/var/qmail/bin/maildirmake
➥Maildir
```

utasítást.
Ha mindent jól csináltunk, akkor a programot lefuttatva, a *3. Listában* felsorolt folyamatokat (***process***) láthatjuk. Ne felejtsünk el kedvenc ***Linux*** disztribúciónk valamelyik indító szkriptjében hivatkozni rá, például ***Slackware*** esetében a */etc/rc.d/rc.local* állományban.
A figyelmes olvasónak bizonyára feltűnt a `-x` kapcsoló után szereplő ***cdb*** fájl. Ezzel adhatjuk meg azt a hálózatot és/vagy ***IP***-címet, amelyek számára ***relay*** szolgáltatást nyújtunk.
A *4. listában* egy lehetséges megoldást láthatunk, amely a ***localhost*** illetve a 192.168.1.0/24 hálózat számára engedi a levél továbbítását, és amelyből a

```
tcprules /etc/tcp.smtp.cdb
➥/etc/tcp.smtp.tmp < /etc/
➥tcp.smtp
```

utasítással hozható létre a szükséges ***cdb*** állomány.
A ***Unix*** programok egy része a ***sendmail*** programot használja levélküldésre. A ***Qmail*** rendelkezik egy burkoló (***wrapper***) programmal, amelyet az

```
ln -s /var/qmail/bin/sendmail
➥/usr/lib/sendmail
```

és az

```
ln -s /var/qmail/bin/sendmail
➥/usr/sbin/sendmail
```

parancsokkal tehetünk például a ***pine*** számára elérhetővé.
Az *5. Listában* egy levél nyomát láthatjuk a naplóban. A bejegyzésből azt tudhatjuk meg, hogy sj@aaaa.hu 2765 byte hosszú levelet küldött a bela@mail.aaaa.hu címre. A levél a 219962 (ez valójában egy ***i-node*** száma) sorszámot kapta meg ***queue*** azonosítóként, ill. hogy a levél küldése sikeres volt.
Ha a levél küldője rajta van az ***RBL*** listáink valamelyikén, akkor az ***rblsmtpd*** program 451-es hibával elutasítja a levelet az ***RCPT TO*** fázisban, amint az a *6. Listában* is látható, ill. az esemény a naplóba is bekerül – lásd a *7. Listát*, az adott ***IP***-címhez tartozó ***TXT*** rekorddal együtt.

2. Lista A /etc/rc.d/rc.qmail tartalma

```
#!/bin/sh

QMAILDUID=`id -u qmaild`
NOFILESGID=`id -g qmaild`
MAXSMTPD=100
LOCAL=`head -1 /var/qmail/control/me`

exec /usr/local/bin/softlimit -m 20000000 \
    /usr/local/bin/tcpserver -v -R -l "$LOCAL" -x
/etc/tcp.smtp.cdb -c "$MAXSMTPD" \
        -u "$QMAILDUID" -g "$NOFILESGID" 0 25
➥/usr/local/bin/rblsmtpd -r sbl-xbl.spamhaus.org \
        -r rbl.aaaa.hu /var/qmail/bin/qmail-smtpd 2>&1 |
➥/var/qmail/bin/splogger smtpd 3 &

exec env - PATH="/var/qmail/bin:$PATH" qmail-start ./Mailbox
➥splogger qmail &
```

3. Lista Ezeknek a qmail programoknak kell futniuk

```
qmaild    8311  0.0  0.2   1356   292 pts/0    S    15:04   0:00
➥/usr/local/bin/tcpserver -v -R -l mail.aaaa.hu -x /etc/
➥tcp.smtp.cdb -c 100 -u 1000 -g 102 0 25 /usr/local/bin/
➥rblsmtpd -r rbl.aaaa.hu -r sbl-x
root      8312  0.0  0.3   1472   444 pts/0    S    15:04   0:00
➥/var/qmail/bin/splogger smtpd 3
qmails    2657  0.0  0.2   1500   360 pts/0    S    12:17   0:00
➥qmail-send
qmaill    2658  0.0  0.3   1468   440 pts/0    S    12:17   0:00
➥splogger qmail
root      2659  0.0  0.2   1468   304 pts/0    S    12:17   0:00
➥qmail-lspawn ./Mailbox
qmailr    2660  0.0  0.2   1464   304 pts/0    S    12:17   0:00
➥qmail-rspawn
qmailq    2661  0.0  0.2   1456   316 pts/0    S    12:17   0:00
➥qmail-clean
```

4. Lista A /etc/tcp.smtp fájl tartalma

```
192.168.1.:allow,RELAYCLIENT=""
127.:allow,RELAYCLIENT=""
```

Ha ***tinydns***-t használunk, a *8. Listában* szereplő bejegyzésekre lesz szükség.
A ***Qmail*** nem rendelkezik hagyományos konfigurációs állománnyal, hanem környezeti változók, ill. a */var/qmail/control* könyvtárban

5. Lista Egy levélhez tartozó naplóbejegyzések

```
Mar 21 14:25:30 aaa qmail: 1174519530.809151 new msg 219962
Mar 21 14:25:30 aaa qmail: 1174519530.809802 info msg 219962:
➥bytes 4769 from <sj@aaaa.hu> qp 2765 uid 1000
Mar 21 14:25:30 aaa qmail: 1174519530.837840 starting delivery
➥2: msg 219962 to local bela@mail.aaaa.hu
Mar 21 14:25:30 aaa qmail: 1174519530.838335 status: local 1/10
➥remote 0/20
Mar 21 14:25:30 aaa qmail: 1174519530.890881 delivery 2:
➥success: did_1+0+0/
Mar 21 14:25:30 aaa qmail: 1174519530.891472 status: local 0/10
➥remote 0/20
Mar 21 14:25:30 aaa qmail: 1174519530.891820 end msg 219962
```

6. Lista Az rblsmtpd elutasított egy spammert

```
220 rblsmtpd.local
HELO aaaa.hu
250 rblsmtpd.local
MAIL FROM: <sj@aaaa.hu>
250 rblsmtpd.local
RCPT TO: <sj@aaa.acts.hu>
451 spammer vagy
```

7. Lista Az rblsmtpd épp megfogott egy spammert

```
Mar 21 14:35:38 aaa smtpd:
➥1174520138.145382 rblsmtpd:
➥192.168.1.22 pid 2795: 451
➥spammer vagy
```

8. Lista DNS rekordok a tinydns-ben

```
+22.1.168.192.rbl.aaaa.hu:
➥127.0.0.2:1800
'22.1.168.192.rbl.aaaa.hu:
➥spammer vagy:1800
```

található egyszerű szöveges fájlok segítségével hangolhatjuk a működését, amelyek részletes leírása a http://www.lifewithqmail.org/lwq.html#config-files címen található. Ezek segítségével beállíthatjuk például

9. Lista A checkpassword telepítése

```
tar zxvf checkpassword-0.90.
➥tar.gz
cd checkpassword-0.90
echo cc -O2 -include
➥/usr/include/errno.h >
➥conf-cc
make setup check
```

virtuális domaineket, az üzenetek max. méretét, mennyi ideig várjon a távoli *SMTP* ügyfélre, a *HELO* paraméter értékét, stb.

POP3 szolgáltatás

A *Qmail* csomagban egy teljes értékű *POP3* démon (*qmail-popup*) is található, amely (csak) M*aildir* formátumú postafiókkal tud együtt működni. A használat előtt telepítsük a *checkpassword* programot, amelyet a http://cr.yp.to/checkpwd/install.html címről tölthetünk le, telepíteni pedig a *Berstein* programjainál megszokott, a *9. Listában* látható módon lehet.
A http://cr.yp.to/qmail/toaster.html címen egyéb jelszó ellenőrző programokat is találunk, amelyekkel – ha úgy kívánjuk – a *checkpassword* kiváltható.
Futtassuk a

```
tcpserver 0 110 /var/
➥qmail/bin/qmail-popup
➥mail.aaaa.hu /bin/
➥checkpassword /var/qmail/bin/
➥qmail-pop3d Maildir
```

utasítást, ahol 0 a root felhasználó numerikus azonosítója (*userid*), a 110 pedig a *POP3* szolgáltatás *TCP* portja. Ezután a levelező ügyfelek rögtön elérhetik postafiókjukat. Természetesen más *POP3* (sőt *IMAP4*, *webmail*, stb.) kiszolgáló is használható, ha úgy akarjuk. Szerencsére nem kell érvényes héjprogramot (*shell*) beállítani a csak *POP3* szolgáltatást használó felhasználók számára.

Statisztika

Egyetlen alkalmazás sem lehet meg kimutatások nélkül, amelyből kiderül, hogy ki, kinek, mikor és mennyit küldött. A *qmailanalog* program segítségével (http://cr.yp.to/qmailanalog.html) részletes statisztikát készíthetünk mindezekről.
Nézzünk meg először egy általános statisztikát a

```
grep qmail /var/log/maillog |
➥awk '{$1="";$2="";$3="";
➥$4="";$5="";print}' |
➥/usr/local/qmailanalog/bin/
➥matchup | /usr/local/
➥qmailanalog/bin/zoverall
```

parancs segítségével, amely a *10. Listához* hasonló eredményt ad, ahol láthatjuk a továbbított üzenetek számát, hogy abból mennyi volt sikeres, illetve sikertelen, meddig tartott átlagosan egy üzenet továbbítása, illetve a különböző késleltetések, méretek stb. is szerepelnek a jelentésben.
A *zsuids* parancs táblázatos formában mutat meg statisztikát az egyes felhasználókról, amint az a *11. Listában* látható.
A `zrecipients` parancs a címzettekről mutat meg összesítést, lásd a *12. Listát*.
A további parancsokról az előbbi web oldal közöl információkat.

Biztonság

Ma már egy magára valamit is adó levelező rendszernek biztosítania kell a vírusok- és spam elleni védelmet.
A *Qmail-Scanner* egy lehetséges megoldás, amelyet a http://qmail-scanner.sourceforge.net/ címről tölthetünk le. Az alkalmazás többféle vírusirtóval is együtt tud működni, például *Clamav, Sophos, McAfee*, stb. ill. a *SpamAssassin* spamszűrővel.

10. lista Általános információk a qmail futásáról

```
Completed messages: 4010
Recipients for completed messages: 3955
Total delivery attempts for completed messages: 3963
Average delivery attempts per completed message: 0.988279
Bytes in completed messages: 46955358
Bytes weighted by success: 46738103
Average message qtime (s): 0.582822

Total delivery attempts: 3963
  success: 3940
  failure: 15
  deferral: 8
Total ddelay (s): 2326.556067
Average ddelay per success (s): 0.590496
Total xdelay (s): 798.067730
Average xdelay per delivery attempt (s): 0.201380
Time span (days): 2.07719
Average concurrency: 0.00444682
```

11. Lista Az egyes felhasználói azonosítók tevékenysége

```
mess     bytes    sbytes    rbytes  recips  tries      xdelay  uid
   2       388       388       388       2      4    0.373292  0
  90    407842    338807    338807      63     67    4.335345  1000
   1       294       294       294       1      3    0.275610  1003
  10     41879         0     41879      10     10    0.763050  1006
   1      3496         0      3496       1      1    0.080386  1009
3593  42561854  42459009  42472581    3565   3565  733.164487  1010
```

12. Lista Statisztika címek szerinti bontásban

```
  sbytes  mess  tries  xdelay  recipient
    7009     2      3    0.12  local.bela@aaa.acts.hu
46413806  3878   3878  792.39  local.bela@mail.aaaa.hu
       0     1      1    0.08  local.maildir@mail.aaaa.hu
       0     5      5    0.31  local.postmaster@mail.aaaa.hu
       0     5      5    0.46  remote.sj@aaaa.hu
```

13. Lista A qmail-scanner telepítése

```
groupadd qscand
useradd -g qscand -s /bin/
➥false -d /nonexistent
➥qscand
tar zxvf qmail-scanner-
➥2.01.tgz
cd qmail-scanner-2.01
./configure --log-details
➥syslog --domain mail.aaaa.hu
```

14. Lista Setuid wrapper program telepítése

```
gcc -O2 -o qmail-scanner-
➥queue qmail-scanner-queue.c
cp qmail-scanner-queue /var/
➥qmail/bin/qmail-scanner-queue
chown qscand:qscand
➥/var/qmail/bin/
➥qmail-scanner-queue
chmod 6755 /var/qmail/bin/
➥qmail-scanner-queue
chmod -s /var/qmail/
➥bin/qmail-scanner-queue.pl
chown -R qscand:qscand
➥/var/spool/qscan
```

15. Lista A /etc/tcp.smtp fájl tartalma módosítás után

```
192.168.1.:allow,RELAYCLIENT=
➥"",QMAILQUEUE="/var/qmail/
➥bin/qmail-queue"
127.:allow,RELAYCLIENT="",
➥QMAILQUEUE="/var/qmail/bin/
➥qmail-queue"
```

A példában feltételezzük, hogy a ***clamav*** antivírus csomag már telepítve van, és gépünkön fut a ***clamd***. Adjuk ki az *13. Listában* szereplő utasításokat, majd kövessük a megjelenő instrukciókat.
Ha minden rendben ment, akkor a `./configure --install` következik, és kövessük ismét az utasításokat. A telepítő létrehozza a */var/spool/qscan* könyvtár alatt a szükséges állományokat és könyvtárszerkezetet, illetve a `qmail-scanner-queue.pl` programot a */var/qmail/bin* alá másolja.
Ha a telepítő arra panaszkodik, hogy a ***Perl*** nem tud ***setuid*** programot futtatni, akkor telepítsük a ***contrib*** könyvtárban található ***qmail-scanner-queue.c*** programot az *14. Listában* látható utasításokkal, majd a */var/qmail/bin/qmail-scanner-queue.pl* első sorából töröljük a `-T` kapcsolót.
Már csak azt kell tudatni a ***Qmail***-lel, hogy ne a gyári ***qmail-queue*** programot használja, hanem a mi ***setuid*** programunkat – amely a */var/qmail/bin/qmail-scanner-queue.pl* programot futtatja, hogy az elvégezze a vírusellenőrzést, majd átadja a levelet a ***qmail-queue*** programnak, hogy aztán a levél tovább

16. Lista Egy vírusmentes levél naplóbejegyzése

```
Mar 22 09:21:03 aaa qmail: 1174587663.700520 new msg 64906
Mar 22 09:21:03 aaa qmail: 1174587663.701110 info msg 64906:
➥bytes 2932 from <sj@aaaa.hu> qp 31739 uid 1000
Mar 22 09:21:03 aaa qmail: 1174587663.731470 starting delivery
➥40: msg 64906 to local sj@mail.aaaa.hu
Mar 22 09:21:03 aaa qmail: 1174587663.732095 status: local 1/10
➥remote 0/20
Mar 22 09:21:03 aaa qmail-scanner[31732]:
➥Clear:RC:1(192.168.1.22): 0.221937 2642 sj@aaaa.hu
➥sj@mail.aaaa.hu RE:_torles
➥<E8C8116587EBD6119ACB00508B63C4DD0A65003B@bxxxxx01.xxxx.xxx.h
➥1174587663.31734-0.aaa:424 orig-aaa117458766354031732:2642
Mar 22 09:21:03 aaa qmail: 1174587663.843804 delivery 40:
➥success: did_1+0+0/
Mar 22 09:21:03 aaa qmail: 1174587663.844481 status: local 0/10
➥remote 0/20
Mar 22 09:21:03 aaa qmail: 1174587663.844789 end msg 64906
```

17. Lista Egy vírusos levélhez tartozó naplórészlet

```
Mar 22 19:28:24 aaa clamd[31833]:
➥/var/spool/qscan/tmp/aaa117458810454031841/photo-9502.zip:
➥Worm.Sober.G FOUND
Mar 22 19:28:25 aaa clamd[31833]:
➥/var/spool/qscan/tmp/aaa117458810454031841/
➥orig-aaa117458810454031841: Worm.Sober.G FOUND
Mar 22 19:28:25 aaa qmail-scanner[31841]: Clear:RC:1(127.0.0.1):
➥0 1100 sj@mail.aaaa.hu <>
➥virus_found_in_sent_message_"FwD:_hey_dude!_(fwd)"
➥aaa117458810554031841-sj@mail.aaaa.hu quarantine-
➥event.txt:1000
Mar 22 19:28:25 aaa qmail-scanner[31841]:
➥CLAMDSCAN:Worm.Sober.G:RC:1(192.168.1.22): 0.935775 69075
➥sj@aaaa.hu sj@mail.aaaa.hu FwD:_hey_dude!_(fwd)
➥<ab.20609153921.K49148@aaaabbbb.xxxx.hu> photo-9502.zip:49821
```

18. Lista A /var/spool/qscan/quarantine.log-ba írt bejegyzés

```
Thu, 22 Mar 2007 19:31:52
➥AKST  sj@aaaa.hu
➥sj@mail.aaaa.hu FwD: hey
➥dude! (fwd)    Worm.Sober.G
➥clamdscan: 0.90.1/2892.
```

mehessen. Ehhez állítsuk be a `QMAILQUEUE` környezeti változót a */etc/tcp.smtp* állományban, a *15. Lista* szerint, majd frissítsük a hozzá tartozó ***cdb*** fájlt a korábban leírt módon.

Ha egy vírusmentes levél érkezik, akkor a *16. Listában* látható bejegyzés kerül a naplóba

Ha vírust talál a levélben, akkor azt karanténba helyezi, és a *17. Listában* látható bejegyzés kerül a naplóba (csak a lényeges részt mutatva), ill. a *18. Listában* a */var/spool/qscan/quarantine.log* fájlba írt bejegyzés látható. Bár véleményem szerint jobb lenne, ha a levelet az ***SMTP*** párbeszéd során 55x hibával eldobná. Egy kis módosítással azonban ez is elérhető.

Vírusos levél esetén – ha a ***debug*** opciót engedélyezzük (alapértelemzésben be van kapcsolva) – a */var/spool/qscan/qmail-queue.log* állományba egy igen részletes bejegyzés kerül, ahol mindent megtudhatunk, hogy a levél feldolgozása során pontosan mi történik.

Ami a spam elleni védelmet illeti, ha már telepítettük a ***SpamAssassin***-t, akkor azt felismeri és használja a ***qmail-scanner***. Ha még nem, akkor érdemes egy olyan statisztikai elven működő programot választani, amely használható a ***maildrop*** vagy ***procmail*** programokon keresztül.

Hogyan tovább?

A ➲ http://www.lifewithqmail.org/lwq.html#related-packages címen további hasznos kiegészítő programokat találunk. Bizonyára telepíteni akarjuk a ***dot-forward*** csomagot, amely ***sendmail***-kompatibilis ***.forward*** támogatást biztosít.

Ha nem akarunk megbarátkozni a ***qmail*** saját álnév (***alias***) mechanizmusával, akkor a ➲ http://cr.yp.to/fastforward.html címről töltsük le a ***fastforward*** csomagot.

Ajánlom még az Olvasó figyelmébe a http://www.qmail.org/top.html és a ➲ http://www.lifewithqmail.org/lwq.html címeket, ahol részletes információk, leírások, foltok ill. segédprogramok találhatóak, amelyek megkönnyítik életünket.

A ***Qmail*** nagy teljesítményű, biztonságos, a többi ***djbware*** programhoz hasonlóan moduláris, és néhány segédprogram segítségével ***sendmail*** kompatibilissé tehető. ***Berstein*** a ***TODO*** részben felveti, hogy esetleg az egészet újraírja a nulláról. Hátha akkor már belekerülnek a hiányolt-, ill. azok a funkciók is, amelyeket jelenleg csak kissé körülményesen lehet biztosítani.

Sütő János
(jsuto@freemail.hu)

1997 óta használ Slack ware Linux-ot. Szabadidejében a postfix clapf nevű vírus- és spam szűrőjét polírozza.

Soul Ride

Snowboard, Linuxon

A sportjátékok fejlesztői (és kiadói) nem igazán rajonganak a Linuxért, így csak kevés klasszikus megoldás fut Szabad rendszeren. A ritka kivételek sajnos tovább erősítik az előbb írt szabályszerűséget, de legalább bepillantást engednek a stílus világába.

Az idei tél igen szeszélyesre sikeredett. Úgy *Magyarországon*, mint a világ bármely hasonló éghajlatú vidékén. Ebből talán nekünk jutott a szerencsésebb rossz, hiszen amíg máshol komoly hóviharok tomboltak, addig mi hónapokon át élvezhettük a tavaszi időjárást (az persze más lapra tartozik, hogy ennek a következményei hová vezetnek majd). Nos, biztosan akad, aki nagyon örült a januári melegnek, másoknak viszont csak kesergésre adott okot. Bárhogyan is volt: igazi havat már régen láthattunk, ezért megpróbálom a hiányérzetet csökkenteni, a magam módján. Egy téli sportjátékot fogok röviden bemutatni: a kiszemelt program 2003-ban jelent meg, készítői *Soul Ride* címre keresztelték. Gyakorlatilag egy extrém téli sportág, a *snowboard* számítógépes szimulációjáról van szó, mely *Win32* és *Linux* felületen egyaránt fut.

Lélekvesztő lejtőkön

Rögtön az elején le kell szögeznem egy dolgot. Ami engem illet, elképzelhetetlennek tartom, hogy hódeszkán állva, 35-40 fokos lejtőkön csúszkáljak. Ezt a szórakozást nem nekem találták ki: se affinitásom, se tehetségem nincs ehhez az egyébként látványos dologhoz. Ellenben a deszkázás népszerűsége nem lehet kérdés tárgya, hiszen rengetegen vállalkoznak a veszélyes mutatványra. Aki komolyabb sérülés nélkül szeretné megúszni a sporttal történő ismerkedést, annak nincs más választása: el kell utaznia valamelyik hófedte, lejtős terepre, és hozzácsapódni egy segítőkész csapathoz. A másik lehetséges utat a *Soul Ride* kínálja, hiszen székben ülve, képernyő előtt talán ez érintettek minden csontja épen fog maradni. Persze az ismerkedés ebben az esetben csak képletesen értendő, hiszen a játékprogram nem alkalmas valódi tapasztalatok szerzésére (erre a pontra azért még visszatérek), csupán a speciális, akrobatikus lesiklás hangulatát szimulálja tökéletesen.

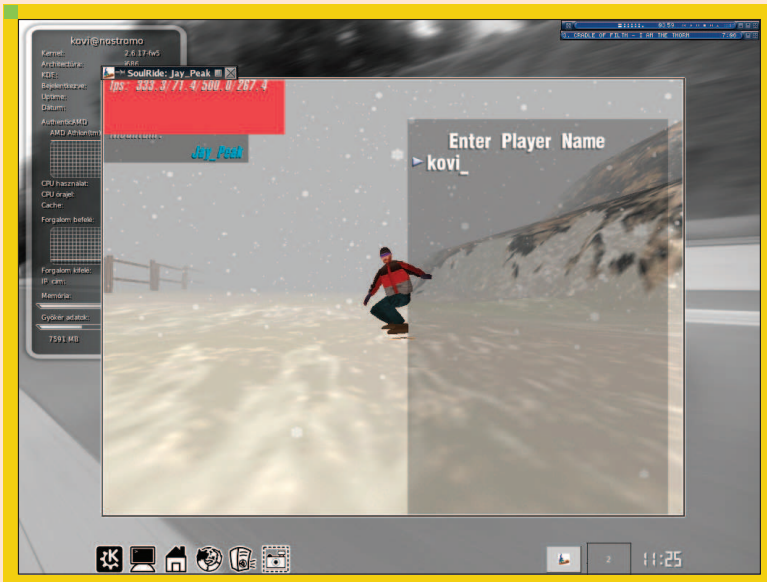
Fájl: config.txt

```
TerrainMeshSlider = "1"
TextureDetailSlider = "10"
SurfaceNodeBuildLimit = "3"
DetailMapping = "1"
Clouds = "1"
Fog = "1"
ShowFrameRate = "1"
SFXVolume = "0"
CDAudioVolume = "0"
OGLModeIndex = "2"
Fullscreen = "0"
Music = "1"
Sound = "1"
DefaultMountain = "Jay Peak"
MasterVolume = "85"
```

1. ábra A program paraméterei

Üzembe állítás

A *Soul Ride* linuxos verzióját a *Linux Game Publishing* csapata terjeszti: a 32 bites *x86* (és *PPC*) alapú *Linux* rendszerekre készült natív változat már csekély 14 *Euro* áron megrendelhető, a ➲ *http://www.linuxgamepublishing.com* oldalon keresztül (a szoftver vásárlás előtt kipróbálható, mivel a demó példány ugyanitt elérhető). Engedelmetekkel feltételezem a megrendelés tényét: ahogyan kézhez kaptuk a (linuxos) *CD*-t, fűzzük be a rendszerünkbe, majd a lemez gyökerében található *setup.sh* szkriptet futtassuk le *root* jogosultsággal. A játék alapértelmezés szerint a */usr/local/games/soulride* útra települ,



2. ábra Íme, így siklok - KDE asztalon

3. ábra A szlalom nem túl nagy kihívás...

indító kötéssel a */usr/local/bin* mappában. Ha minden állomány a helyére került, felhasználóként kiadott soulride paranccsal léphetünk be az extrém sportok világába...

A számunkra érdekes kiadás használatának feltételét egy ***500 MHz*** órajelű központi egység, valamint egy ***32 MByte*** memóriával szerelt ***3D*** grafikus hardver jelenti (betöltött ***GLX***, illetve ***DRI*** meghajtóval). A teljes telepítés szerény helyigénye hasonlóan mosolyogtató, hiszen éppen csak eléri az 50 ***Mbyte***-ot: összességében bátran kijelenthetem, hogy a leendő felhasználónak szinte biztosan nem lesz olyan problémája, ami a számítógépe gyengeségéből ered.

A ***Soul Ride*** kódja minden felhasználói beállítást egy speciális, személyhez kötődő fájlban tárol (***/home/$/lgp/soulride/config.txt***). Mivel a leképezett belső világ képfelbontása sajnos nem változtatható meg a játék grafikus menüiből, így az említett szöveges állomány „***OGLModeIndex=#***” sora kitüntetett figyelmet érdemel. Íme, a használható „#” értékek, jelentésükkel együtt: ***1=640x480px***, ***2=800x600px***, ***3=1024x768px***, ***4=1280x1024px.***

Tippek, trükkök, tapasztalatok

A menürendszer nem érdemel hosszabb magyarázatot, még ismeretlenül sem fog komoly fejtörést okozni senkinek. Szerencsére átlátható és lényegre törő lehetőségeket biztosít, érintve a képi leképezést, hangszolgáltatást, játékmódokat és az időjárás beállítását. Ellenben, a monitoron száguldozó deszkás emberke irányítása elsőre talán bonyolult lehet. Nézzük tehát, hogyan kell kezelni a programot! A lesikló modell a kurzorbillentyűkkel vezérelhető jobbra és balra (az irányváltás mellett ez lassításként is funkcionál). Alapértelmezés szerint álló helyzetben egyensúlyoz, a ***CTRL*** billentyűt lenyomva pedig leguggol. Ekkor például az előre- vagy hátra gombot nyomva tartva, a ***CTRL***-t felengedve, a megfelelő irányú ***flip***-et (a szaltó megfelelője) kezdeményezünk, aminek sikere a sebességen, és a megfelelő pillanatban történt „dobbantáson” múlik. Ugyanezt a módszert szem előtt tartva, az oldalirányú mozgásért felelős billentyűkkel is eljátszható a trükk.

A levegőben kieszközölt minden akrobatikus mozdulat (***flip***, dupla- és tripla ***flip***, ***degree***, ***air***, ***nice-*** és ***big air***) azonnal pontozásra kerül, majd a pálya teljesítése után a futamidővel együtt meghatározza a számszerű eredményt.

A program összképe nagyon kedvező, összesen csak két-három dologgal nem tudtam megbarátkozni. Elsőként említeném a pályák havas fenyőinek szerény kidolgozottságát. Szerencsére a ****.psd*** grafikák kicserélhetőek jobbakra, szebbekre: a fő elérési út ***/data*** könyvtárában lehet megtalálni az eredeti, kis felbontású képeket (a többi teendő már adja magát). A másik hiányosság viszont nem gyógyítható a felhasználó által. Arról van szó, hogy az esetleges balesetet szenvedett karakter a fizika szabályait meghazudtolva és nevetségesen pörög a levegőben, vagy éppen a hóban hasalva csúszik tovább. Pedig az

4. ábra ... Az első dupla flip viszont órákba is telhet.

összetett mozgáskultúra minden egyéb fázisban élvezhetőre sikeredett... Végül íme az utolsó, kellemetlen apróság: a játék nehezen „pörög fel". Ezen a ponton barátilag javaslom, hogy az első, lassú hegyoldalak senki kedvét se vegyék el a továbbiaktól! Később olyan gyors és „sűrű" terepeket kell majd időre teljesíteni, amit ép eszű ember az ellenségének sem kívánna.

Végül

A rovat elején azt állítottam, hogya ***Soul Ride*** nem alkalmas valódi technikák elsajátítására. Nos, azért ennyire mégsem eszik forrón a kását... Sőt, (az irányíthatóságot tekintve) talán a kategória egyik legérdekesebb opcióját vásárolhatjuk meg a játékhoz! A kétkedőknek barátilag tanácsolom, hogy látogassanak el a ➲ ***http://www.soulride.com*** oldalra, és csodálják meg azt a deszkát, melyre ráállva beépített nyomásérzékelőkkel vezérelhető a képernyő minden története... Ha még ezek után is kételyek lebegnének a látogató szemei előtt, bemutató videók bizonyítják a jópofa ***Catapult*** irányító létjogosultságát...

5. ábra Sűrű erdő közepén, jeges vízbe fulljak´ én?

Kovács Zsolt (kovi@linuxforum.hu)

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

MythTV videófeldolgozás felsőfokon

Kifinomult módszerek a lejátszás közbeni deinterlacing megvalósítására, és a videó hordozhatóvá tételére.

Stabilitása és kiterjeszthetősége miatt a ***Linux*** gyakran megtalálható a számítástechnika határterületein. Ígéretes platformnak mutatkozik a házimozi audiovizuális alkalmazásai esetében is. Saját tévéhasználatomat jelenleg egy speciális célú ***Linux PC*** vezérli, amin ***MythTV*** fut. A ***MythTV*** telepítésekor mindkét fő probléma, amivel találkoztam, a videófeldolgozáshoz kapcsolódott. Az első kihívás a sima videólejátszás konfigurálása, a második a felvett műsorok hordozhatóvá tétele volt.

Deinterlacing videólejátszás közben

Napjaink elektronikájának korlátai miatt a televíziós képkockák két külön félkép formájában kerülnek adásra. Egy félkép a képnek vagy a páros, vagy páratlan számú vízszintes soraiból áll. Lejátszásnál a félképek páros és páratlan számú sorai összefésülésre kerülnek, amit a képernyőtől kellően távol levő néző folyamatos mozgásként érzékel.
Két egymást követő félkép összefüggő, de nem azonos. A kamera gyors oldalirányú mozgása közben egy adott félkép kicsivel előtte lesz az őt megelőzőnek, és egyenetlen kontúrok jelenhetnek meg a képernyőn gyorsan váltakozó képeken. Az *1. ábra* egy 1080i videómódot használó, nagy felbontású adás képét mutatja. A jelenetben a kamera balról jobbra pásztáz, ezért a kép objektumai gyorsan mozognak a képernyőn. Minden egyes félkép kicsit eltérő pozícióban van, ami fűrészfog-szerű torzulást okoz, amit fésűsödésnek, csipkézettségnek, illetve fogazottságnak is hívnak. Azoknál a jeleneteknél, ahol nagy az oldalirányú mozgás, a torzulás miatt hihetetlenül nehezen követhetővé válhat a műsor tartalma.
Ahhoz, hogy a *1. ábrán* láthatóhoz hasonló videót nézhetővé tegyük, sima képpé konvertálható egy deinterlacing néven ismert eljárással. A ***MythTV*** felhasználók a deinterlacing számos módszere közül választhatnak:

- ***Egy félkép*** – Ahelyett, hogy két félképből állítana össze egy képkockát, ez a rendkívül egyszerű módszer csak az egyiket használja fel. Minden második félkép állóképként jelenik meg, a nem használtakat pedig figyelmen kívül hagyjuk.

- ***Lineáris*** – Ez a módszer összekeveri a közvetlenül szomszédos sorokat, amelyek váltakozó félképekből származnak. Enyhe szellemkép tűnhet fel, de a fűrészfog-szerű torzulás megszűnik.

- ***Kernel*** – Az előzővel ellentétben nem csak a szomszédos sorokat, hanem számos más sort is összekever. Nincs szellemkép, bár az egyik félképből visszamaradhatnak halvány részletek.

- ***Bob*** – Ez a legbonyolultabb módszer. Minden egyes félkép sorai megduplázódnak, majd az így létrehozott képkockákat dupla sebességgel játsszuk le.

A deinterlacing jelentős számítási teljesítményt követel, de a legtöbb modern ***CPU*** rendelkezik multimédiás utasításkészlettel, ami csökkenti a processzorra jutó terhelést. Ha ***Intel*** processzort használunk ***MMX*** vagy ***SSE*** utasításokkal, vagy ***AMD*** processzort ***3Dnow!*** támogatással, a deinterlacing nem jelenthet gondot. A ***Bob*** a szinkronizált TV kimenet előállítására alkalmas legjobb deinterlacing eljárás, bár egy kevésbé fejlett gépen nehézségeket okozhat. A saját ***MythTV*** rendszerem egy 2 GHz-es ***AMD Athlon64*** processzort rejt, ami bőven elég ***Bob*** módszerrel deinterlace-elt nagy felbontású videó megjelenítésére egy analóg TV-n. Bár a ***CPU*** terhelése nagyobb, mint más deinterlacing módszereknél, még mindig bőven a rendszerem teljesítőképességén belül van.
A lineáris és a kernel deinterlacing alkalmazásával hasonló képi hatás érhető el, míg az utóbbi kissé jobban igénybe veszi a ***CPU***-t. Kevésbé erőforrás-igényesek, mint a Bob, ami jól jöhet gyengébb ***CPU*** esetén. A kettő közül személy szerint a kernel deinterlacinget kedvelem jobban, mert több szomszédos sort kever és kiküszöböli a szellemképet, így a kapott kép élesebb lesz.

Videó exportálása más rendszerekbe

Az egyik alapvető oka annak, hogy felállítottam egy ***MythTV*** rendszert, az volt, hogy a tévéműsoraimat szerettem volna utazáskor magammal vinni. Most a „televízió" számomra fájlokat jelent a merevlemezen, így sokkal könnyebb tárolni, szállítani, és ott nézni, ahol éppen kényelmes. Mint a sokat utazó szakembereknél általában, a laptop számomra is egy extra tartozék lett, ami utazás közben ideális eszköz a videó lejátszására, főleg mostanában, amióta sok repülőgépen

1. ábra Lejátszás közbeni fésűsödés

megjelentek a hordozható elektronikus eszközök energiaellátásához szükséges csatlakozók.
A *MythTV* videó exportálásakor alapvető kérdés, hogy megtaláljuk az egyensúlyt a méret és a feldolgozási idő között. A digitális TV adás szabványai leírják, hogyan továbbítsunk *MPEG-2* videó adatfolyamot egy TV csatornára, így a *MythTV* a digitális tévéműsorokat „természetes" *MPEG-2* formátumukban tárolja. Az MPEG-2 digitális TV adatfolyam konvertálása egy másik *MPEG-2* alapú videóformátumra viszonylag könnyű, és nem igényel hosszú feldolgozási időt. A digitális tévéadás *MPEG-4* formátumra konvertálása már sokkal nagyobb számítási teljesítményt követel, de az eredményként kapott videó fájl jóval kisebb lesz.

A MythTV felvételek feldolgozása

Bár a digitális TV felvételek *MPEG-2* videó adatfolyamok, a *MythTV* által használt *NuppelVideo* konténer formátum speciális *MythTV* sajátosság, és a legtöbb videólejátszó szoftver nem ismeri. Amennyiben a videókat nemcsak a *MythTV* kezelőfelületén keresztül szeretnénk megtekinteni, olyan formátumra kell őket konvertálni, amit több lejátszó is támogat.
A videó exportálását tovább bonyolítják a *MythTV* felvételek tárolásánál használt fájlnevei. A fájlnév első része a *MythTV* által használt csatornaszám, a két hosszú szám pedig a kezdés illetve a befejezés időpontja:

```
myth:/video$ ls -lh *.nuv
-rw-r--r--  1 mythtv users 2.8G
➥Sep  3 08:00 1000_2005090307
➥3000_20050903080000.nuv
-rw-r--r--  1 mythtv users 9.1G
➥Aug 31 23:30 1007_200508312
➥20000_20050831233000.nuv
-rw-r--r--  1 mythtv users 808M
➥Sep  3 01:30 1008_200509030
➥10000_20050903013000.nuv
-rw-r--r--  1 mythtv users 1.8G
➥Sep  1 09:00 1009_200509010
➥80000_20050901090000.nuv
-rw-r--r--  1 mythtv users 3.7G
➥Aug 28 22:00 1019_200508282
➥10000_20050828220000.nuv
```

A *MythTV* felvételek szélesebb körben támogatott formátumra konvertálásához szükséges eszközök készen elérhetők. A *MythTV* disztribúció része a *mythtranscode*, egy program, ami képes a *MythTV NuppelVideo* fájlokat más alkalmazások által is támogatott szabványos videó adatfolyammá alakítani. A *mythtranscode* kimenetét tetszőleges kódoló program bemenetére irányítva bármilyen típusú videó fájlt létrehozhatunk. A gyakran használt kódoló programok egyike az *FFmpeg*, ami számos elterjedt videóformátumot támogat. A két program összekapcsolása a megfelelő kimeneti fájl létrehozásához elméletileg lehetséges, de az összes parancssori opción, és a rendszerben tárolt felvételeken átvergődni nem egyszerű feladat.
Használjuk inkább a *nuvexport* nevű *Perl* szkriptet, ami igazán felhasználóbarát módon kezeli a folyamatot. A *nuvexport* összegyűjti azokat a parancssori opciókat, amelyek a *mythtranscode* és az *FFmpeg* futtatásához szükségesek. Egy szövegalapú kezelőfelületen keresztül kiválaszthatjuk az exportálni kívánt műsort, majd beállíthatjuk a konverziós programok számára szükséges paramétereket.
A *nuvexport* telepítése előtt szükségünk lesz néhány további eszközre. Az *FFmpeg* az alapértelmezett program, amivel a videó a választott célformátumra kódolható. (A *nuvexport* támogatja az átkódolást is, de az sokkal lassabb.) A *MythTV* fájlokat az *MPlayerrel* dekódoljuk a konverzióhoz. A konverzióval gyakran együtt jár a zaj megjelenése is, amit ki kell szűrnünk. A *nuvexport* a *yuvdenoise* programot használja, ami az *MJPEG* eszközök egyike. Mindhárom programot széles körben használják, és valószínű, hogy az általunk használt *Linux* disztribúcióhoz is elérhetők az őket tartalmazó csomagok. A *nuvexport* a *DateManip* modult is használja, úgyhogy szerezzük be ezt is a disztribúciónk weboldaláról, vagy a kedvenc *CPAN* tükröldalunkról.
A *nuvexport* elindításakor első dolgunk az exportált videófájl formátumának kiválasztása. Az alapvető kérdés itt az, hogy minimális fájlmérettel rendelkező, ám hosszabb feldolgozási időt igénylő *MPEG-4* alapú formátumot válasszunk, vagy használjunk nagyobb méretű, de könnyebben létrehozható formátumot. A főbb lehetőségek az alábbiak:

- *Video CD (VCD)*: *MPEG-1* videó adatfolyamból áll (1.150kbps), egy különálló *MPEG layer 2 (MP2)* formátumú hangsávval (224 kbps).

- *Super Video (SVCD)*: változó bitsűrűségű *MPEG-2* videó, *MP2* audiosáv mellett. A *VCD*-vel ellentétben a hangsáv többcsatornás lehet, így 5.1-es vagy 7.1-es hang is tárolható ebben a formátumban.

- ***DVD***: az ***MPEG-2*** videón alapul, a hangsávra többféle opcionális lehetőség van. Magasabb felbontású bármely ***Video CD*** formátumnál.

- ***DivX***: ***MPEG-4*** alapú formátum, ami kis fájlokat eredményez; a fájlméretet a minőség jelentős romlása nélkül minimalizálja. A ***DivX*** állandó illetve változó bitsűrűségű videó létrehozására is képes.

- ***XviD***: ***MPEG-4*** alapú formátum, a ***DivX*** fejlesztésének egyik oldalhajtása. Egy 2001-ben kiadott nyílt forráskódú ***DivX*** kodek fejlesztésen alapul. Alapértelmezésben a ***nuvexport*** változó bitsűrűségű videókódolást használ az ***XviD***-nél, és felkínálja az egy- illetve többmenetes kódolás lehetőségét. A többmenetes kódolás jobb minőségű videót biztosít, a számítási idő növekedése árán.

- ***Advanced Streaming Format (ASF)***: a ***Microsoft*** által kifejlesztett általános médiaformátum, rendszerint a ***Windows Media Audio (WMA)*** és ***Windows Media Video (WMV)*** fájloknál használják. A ***WMV*** a ***Microsoft MPEG-4*** bővítményein alapul.

Tapasztalataim szerint a ***VCD*** és az ***SVCD*** kodekek jó minőséget nyújtanak gyors feldolgozási idővel, míg a ***DivX*** és az ***XviD*** kodekek adják a legkisebb fájlméretet, de létrehozásuk hosszabb ideig tart. Miután kiválasztottuk a videóformátumot, a ***nuvexport*** szövegalapú menürendszerét használva kiválaszthatjuk az exportálni kívánt filmrészleteket, és beállíthatjuk a kodek paramétereit. Miután kijelöltük az exportálásra váró részleteket, a ***nuvexport*** általános kérdéseket tesz fel, mint például hogy hová kerüljön az exportált fájl. Felajánlja a ***MythTV*** egyik szolgáltatását, amivel az exportált videóból kivághatjuk a reklámokat. A zajcsökkentés és a deinterlacing opcionálisak. Bár alapértelmezés szerint mindkét opció be van kapcsolva, általában kikapcsolom őket a megnőtt számítási idő miatt. Sok videólejátszó képes a deinterlacing elvégzésére lejátszás közben, és nem találtam a konverzióból származó rendkívüli zajt.

Néhány formátumnál további kodekspecifikus kérdések is felmerülnek. Mind a ***DivX***, mind az ***XviD*** lehetővé teszi a bitsűrűség és a felbontás állítását. Az alapértelmezett bitsűrűség 128 kbps a hang illetve 960 kbps a videó esetében, ami elegendő arra, hogy a legtöbb számítógépes megjelenítőn jó minőségű videót eredményezzen.
A videó méretének beállításakor a ***nuvexport*** először megkérdezi a kívánt szélességet, majd javasol egy magasságot, ami a felvétel képméretarányán alapul. Figyeljünk arra, hogy a szélesség a felvétel szélességén belül maradjon. Az alapértelmezett szélesség 624, ami általában jó videót eredményez, de 1920x1080 méretű felvételeknél nagyobb is lehet. A ***VCD*** és az ***SVCD*** nem kérdeznek rá a felbontásra, mert ezek a formátumok fix felbontásúak.
Az *1. Táblázat* a különböző formátumokra jellemző feldolgozási időket és tárhelyszükségletet hasonlítja össze, köztük a ***MythTV*** saját formátumát és az én kereskedelmi ***DVR***-emet is. Forrásként egy órányi hosszúságú felvételt használtam a ***PBS*** Nova sorozatából, amit 704x480 felbontásban sugároznak. A „kereskedelmi DVR" bejegyzés a ***DVR***-ről a laptopomra másolt videóra utal. Hozzávetőleges számítás szerint az ***MPEG-2*** körülbelül 1 GB-ot igényel óránként, míg az ***MPEG-4*** csak 350-450 MB-ot.
A táblázatban feltüntetett feldolgozási időbe nem értendő bele a deinterlacing illetve a zajcsökkentés; mindkét feladatot a videólejátszóval oldom meg, lejátszás közben.

Bár az ***ASF*** mérete a legkisebb, ez is néz ki messze a legrosszabbul. Az ***ASF*** fájlban gyakoriak a tömörítésből adódó hibák, ami nagyon idegesítővé teszi a nézését. Bár a kis fájlméret vonzó, a gyenge képminőség miatt kizárhatjuk ezt a formátumot. A ***DVD*** videó rendelkezik a legjobb felbontással, de ez foglalja a legtöbb helyet is.
A két véglet közötti kompromisszumként a ***VCD***-t és a ***DivX***-et használom; egy fájl gyors létrehozására az előbbi alkalmas, az utóbbival pedig a lehető legkisebb fájlméret érhető el.
A videó átkódolása erősen igénybe veszi a ***CPU***-t. Alapértelmezés szerint a ***nuvexport*** alacsony prioritási értékkel futtatja a segédfolyamatokat, hogy megelőzze a rendszeren futó egyéb műveletekkel (például videólejátszás vagy felvétel) való ütközéseket.
Az utóbbi időben kiadott összes ***Linux*** disztribúcióban található olyan szoftver, ami lehetővé teszi a ***CPU*** órajelének megváltoztatását, eleget téve a feldolgozási teljesítményt érintő igényeknek. A ***CPU*** sebességvezérlőt használva az órajelet a lehető legalacsonyabb értéken tartom, ami még elegendő a szükséges munka elvégzéséhez.
Sok sebességvezérlő program nem veszi figyelembe a csökkentett prioritású folyamatokat, de beállíthatók az ilyen folyamatok figyelésére. Az én ***Linux*** disztribúcióm a ***CPUfreq*** kernelmodult használja, amit be kell konfigurálni a csökkentett prioritású folyamatok felügyeléséhez. Egy kisméretű indító szkript az alábbi két parancsot futtatja:

```
echo "ondemand" >
➥/sys/devices/system/cpu/
```

1. táblázat *Az exportálás eredménye (jellemző értékek)*

Formátum	Felbontás	Fájlméret	Kódolási idő
Myth formátum	704x480	1,756MB	N/A
Kereskedelmi DVR	N/A	1,236MB	N/A
VCD	352x240	596MB	15 perc
SVCD	480x480	601MB	25 perc
DVD	720x480	899MB	34 perc
DivX	624x464	432MB	35 perc
XviD (egymenetes)	624x464	451MB	1 óra, 39 perc
XviD (kétmenetes)	624x464	472MB	2 óra, 35 perc
ASF	320x240	143MB	18 perc

```
➥cpu0/cpufreq/scaling_governor
echo "1" > /sys/devices/
➥system/cpu/cpu0/cpufreq/
➥ondemand/ignore_nice
```

A második parancs arra utasítja a ***CPUfreq*** vezérlőt, hogy számolja a csökkentett prioritású folyamatoktól érkező feldolgozási kérelmeket. Az *1. Táblázatban* felsorolt idők egy csúcsteljesítményen működő ***2.0 GHz-es AMD Athlon64***-ra érvényesek. Az 1.0 GHz-es minimum sebességen a feldolgozási idő ennek körülbelül három-négyszerese. A ***MythTV*** videó exportálása azonban csak az egyik fele a dolgoknak. Amint kész a videó, a lejátszáshoz átvihető egy másik gépre. A visszajátszásra használt platformon jelenlevő alkalmazásokon kívül létezik két figyelemre méltó nyílt forráskódú lejátszó: az ***Mplayer*** és a ***Video LAN Client (VLC)***. Én az ***MPlayert*** használom, mert a beépített deinterlacing képességek simább képet eredményeznek, mint a ***VLC***. Mindkét alkalmazás elérhető ***Linux*** és ***Windows*** alatt is.
Az ***MPlayer*** parancssora a különböző operációs rendszereken megegyezik. A cél az éles, teljes képernyős lejátszás elérése. Az `-fs` opció a videót teljes képernyőn játssza le, nincs ablak a kép körül.
A videószűrők a lejátszás megváltoztatására használhatók, és a `-vf` paraméterrel aktiválhatók. Két `-vf` opciót használok. Az egyik egy kis fekete keretet hoz létre a képernyő körül az `expand` szűrővel. Az `expand` szűrő több paramétert is kaphat, egy negatív számmal határozzuk meg a keretet. A `-vf expand=0:-50` szűrő utasítás egy 50 pixeles szegélyt helyez a képernyő aljára, miközben a videó a kereten belül középre igazítva marad. Az éles videóhoz szükséges a deinterlacing is: ezt az ***MPlayer*** az utófeldolgozó szűrővel (`postprocessing` filter, röviden `pp`) aktiválja. Általános szabály szerint négy utófeldolgozó szűrőt kapcsolok be: a vízszintes blokktördelés (`hb`), a függőleges blokktördelést (`vb`), a de-ringing-et (`dr`), és a fényerő/kontraszt korrekciót (`al`).
Az eredményként kapott szűrő a `-vf pp=hb/vb/dr/al` paranccsal aktiválható. Az egészet összerakva az alábbi parancssort kapjuk:

```
mplayer -fs -vf pp=hb/vb/dr/
➥al -vf expand=0:-50
➥(filename)
```

Lejátszás alatt az ***Mplayer*** kiterjedt billentyűparancsai lehetővé teszik a pillanatnyi szünetet, a gyors előretekerést és a képbeállítást, csakúgy, mint a kivetített megjelenítést.

Linux Journal 2006., 140. szám

Matthew Gast a vezetéknélküli hálózatokról szóló kiváló szakkönyv (802.11 Wireless Networks: The Definitive Guide (O'Reilly Media)) szerzője. Jelenleg nagyon sok időt tölt a MythTV-vel. Elérhető a matthew.gast@gmail.com email címen, de csak ha éppen a tengerszint közelében tartózkodik.

FORRÁSOK

➲ www.linuxjournal.com/article/8585