Hírek

Motorola és a pingvin (lowend)

A *Motorola* bejelentette, hogy befejezik a *Razr* (razor, magyarul borotva) fantázianevű készülék gyártását, és helyét a *Linux*os *Scpl* (*scalpel*, magyarul szike). A *Motofone F3 Indiá*ban már kapható. Noha alsó kategóriás telefon (*50 dollár körüli áron* lesz kapható), mégis a kijelző a megszokott *LCD helyett elektronikus papír*, mellyel így hihetetlen készenléti (körülbelül 400 óra) és beszélgetési (körülbelül 8 óra) idő érhető el. A telefonon *MontaVista Linux* fut.

➲ http://www.linuxdevices.com/news/NS5567800205.html

ISO szabvány az OpenDocument

ISO szabvánnyá nyilvánították az *OpenDocumentet (ISO/IEC 26300)*. Ez a gyakorlatban annyit jelent, hogy jogdíj nélkül bárki implementálhatja a szoftverében, legyen az zárt vagy nyílt forráskódú.
Az *OpenDocument* az alábbi végződéseket használja: *.odt* szöveges állományhoz, *.ods* táblázatokhoz, *.odp* prezentációkhoz, *.odg* grafikához és *.odf* matematikai képletekhez.

➲ http://en.wikipedia.org/wiki/OpenDocument

Motorola és a pingvin (highend)

Linux iránt talán a *Motorola* az egyik legelkötelezettebb gyártó. Mi sem bizonyítja, hogy *E6* néven piacra dobott egy mobiltelefont, mely 2.4 hüvelykes *QVGA kijelzővel* (240x320) bír, kapott *2 megapixeles kamerát* és *2 gigabájtig* bővíthetjük *SD* kártyával. Helyet kapott benne *Microsoft kompatibilis irodai csomag*, illetve szinte *bármilyen zenei* formátumot lejátszik, melyet akár

*Bluetooth headset*en is hallgathatunk. Üröm az örömben: *nincs benne 3G*, illetve nem négy, hanem csupán három sávos. Egyelőre csak *Kínában* kapható.

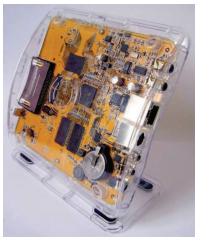
➲ http://www.slashgear.com/motorolas-sexy-linux-smartphone-052771.php

4G próba már az idén

A *Telecom World 2006* kiállításon a *Qualcomm* felfedett pár részletet a *4G* mobil adatátvitelről. Az *UMB (Ultra Mobile Broadband)* lehetővé fogja tenni akár a *280 megabites átvitelt* is. Ez azonban *elméleti* szám, kérdés, milyen lesz a való életben. Az persze már bizonyos, hogy *100 megabites letöltés* és *50 megabites feltöltés* lesz a kitűzött cél a *3GPP*-s hosszútávú terveknek megfelelően.

➲ http://www.eetimes.com/rss/showArticle.jhtml?articleID=196601876

Mini mindenes



A *Neuros Audio* bemutatta apró digitális videó rögzítőjét, az *OSD*-t (*Open Source Device*), amellyel felvehetünk vagy lejátszhatunk videókat. Az eszköz maga nem tartalmaz háttértárat, azonban rendelkezésünkre áll számos kártyahely *(CF, Microdrive, SD/MMC, MemoryStick)*, illetve nem szabad megfeledkezni az *USB háttértárakról*, valamint a netes kapcsolat (*NFS, Samba*) lehetőségéről sem. Az eszköz természetesen *Linux*os és *hardveresen rögzít MPEG4* formátumba. Lejátszáskor viszont nem jön zavarba *xvid*, *asf* vagy *wmv* formátum esetén sem.

➲ http://www.linuxdevices.com/news/NS9045680371.html

➲ http://www.linuxlookup.com/review/neuros_osd_review

Playstation3 támogatás a Linux kernelben

A *2.6.20-as kernelverziótól* már találunk egy *„PS3 console and other devices”* pontot is kernel fordításkor. Ez azt jelenti, hogy minden *IBM Cell processzor*ral szerelt eszköz használható lesz *Linuxszal*. A kódrészletet a *Sony* írta.

➲ http://osnews.com/story.php/16644/Sony-Adds-PS3-Support-to-Linux-Kernel/

64 Studio

Ez nem egy újabb ***hollywood***-i stúdió neve, hanem egy ***Debian*** alapú ***Linux*** disztribúcióé, mely a ***hang és videó szerkesztésre*** lett kihegyezve.
A rendszer alapját a ***Debian 64 bites*** (***Amd*** és ***Intel*** is) verziója adja, de régebbi ***PC***-kre ***elkészült a 32 bites*** változat is.
Grafikus felületet a ***Gnome*** biztosít. ***2.6.17-es kernelét*** felkészítették ***valós idejű SMP-re*** is, így nem jelenthet gondot a többprocesszoros számítógép sem. A ***CD***-n helyet kaptak zenei (pl. ***Ardour***, ***Hydrogen***, ***Jamin***, stb.) és videós alkalmazások, két és háromdimenziós grafikus alkalmazások, webböngésző és irodai alkalmazás.
A disztribúció egyelőre ***ftp*** és ***http*** protokollokon tölthető le.

➲ http://desktoplinux.com/news/NS5486057047.html
➲ http://64studio.com/wiki/DownloadAnInstaller

PostgreSQL 8.2

Egy éves fejlesztői munka után végre megjelent a népszerű adatbázis-kezelő legfrissebb verziója, mely több mint ***200 újdonsággal*** és javítással kecsegtet. A gyakorlatban ez azt jelenti, hogy egyre inkább felzárkózik a kereskedelmi adatbázis-kezelőkhöz. Az egyik legjelentősebb újítás a ***Warm Standby Databases***, amely lehetővé teszi, hogy egy klaszteren belül legyen mentésünk az adatbázisról. Mindezek mellett a ***teljesítmény is javult*** 10, bizonyos esetekben pedig akár 20 százalékkal is.
A ***PostgreSQL*** következő verziója (8.3) előreláthatóan 2007 nyarán jelenik meg.

➲ http://www.internetnews.com/dev-news/article.php/3647376

Metanol alapú tápellátás

A ***Hitachi*** 2007-től kezdi el gyártani a ***metanol alapú*** üzemanyag celláját, mellyel elsősorban kis energiaigényű mobil eszközöket (***PDA, mobiltelefon***) kívánnak táplálni. A készülék 10 ezer órás élettartammal bír, és ***négyzetcentiméterenként 100 milliwattot*** fog leadni. Attól persze nem kell félnünk, hogy megissza a kedvenc sörünket, hiszen abban ***etanol***, míg az üzemanyagcellában ***metanol*** van.

➲ http://www.i4u.com/article7357.html

Novell-Microsoft egyezmény

Richard Stallman, a ***GPL*** atyja, a ***tokiói GPL konferencián*** kijelentette, hogy a ***Novell-Microsoft*** megállapodás nem sérti a ***GPL 2***-es verzióját. Az is kiderült, hogy a 3-as verzió jelenlegi tervezete sem szabna gátat az ilyen megállapodásnak, ***Stallman*** szerint azonban még nincs késő módosítani.

➲ http://www.eweek.com/article2/0,1759,2065479,00.asp

Wizpy, a linuxos MP3 lejátszó

A japán ***TurboLinux*** bemutatta a ***linux***os ***Wizpy***-t, mely ***4 gigabájtos tárhelyével*** lehet vonzó. Az eszköz az ***1.7 hüvelykes kijelzőjének*** köszönhetően ***videót is lejátszhat*** a közismert hangformátumok (***OGG***, ***MP3***, ***WMA***, ***AAC***) mellett, sőt ***rádió***ként és ***diktafon***ként is használhatjuk. Ezenkívül a ***Linuxnak*** köszönhetően szinte bármilyen alkalmazás telepíthető rá. Előreláthatóan ***2007 februártól*** kerül a polcokra kb. ***51 ezer forintos*** áron.

➲ http://www.i4u.com/article7177.html

Firefox: Muszáj hamarosan frissíteni

Jelenlegi tervek szerint a ***Mozilla 2007 április 24***-ig támogatja az ***1.5-ös Firefox***ot, tehát legkésőbb akkor mindenkinek ajánlják a váltást. A tervek között szerepel az 2007 végén megjelenő ***Firefox 3.0*** is.

➲ http://www.pcadvisor.co.uk/news/index.cfm?RSS&newsid=7552

Előtérben a háttértárak

A *Toshiba* 2007 elejétől szállítja legújabb, *100 gigabájtos* merevlemezét. A dolog azért jelentős, mert ez a merevlemez csupán *1.8 hüvelyk* széles. Elsősorban hordozható számítógépekben és médialejátszókban láthatjuk őket viszont.

A *SanDisk* piacra dobta *32 gigabájtos SSD*-jét (Solid State Drive), azaz olyan merevlemezét, mely nem tartalmaz mozgó alkatrészt. Az *1.8 hüvelykes eszköz* akár *62 megabájt másodpercenkénti* sebességgel is képes olvasni. A megoldás nagy előnye a kisebb *energiafogyasztás (0.4 watt)* mellett a nagyobb megbízhatóság is.

A *Hitachi* elkészítette *3.5 hüvelykes*, *1 terabájtos* merevlemezét, mely 2007 első negyedévében kerül a boltokba *399 dolláros* áron.

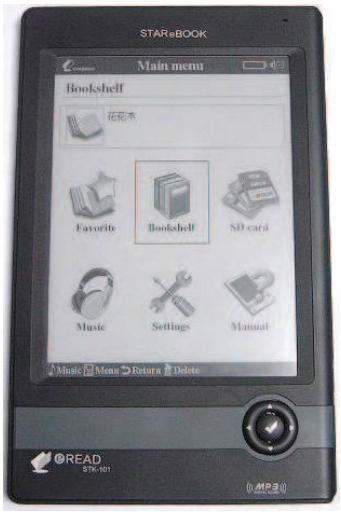
A *Seagate* sem ül a babérjain, még 2007 első felében piacra dobja ő is *3.5 hüvelykes* méretben az *1 terabájtos* merevlemezét. A bejelentés szerint a meghajtó kevesebb lemezt tartalmaz majd, mint a 750 gigabájtos elődj, így *lényegesen sűrűbben* lesznek rajta az adatok.

Mégis csak megfelel a BitTorrent?

Hamarosan nem csak *Linux* telepítőmédiát és nyílt forrású játékokat tölthetünk le legálisan *BitTorrent*en keresztül, de filmeket, sorozatokat is. A megállapodás értelmében a *20. Century Fox*, az *MTV Networks*, a *Paramount Pictures* és még sokan mások elérhetővé tesznek tartalmakat kisebb nagyobb korlátozásokkal. Ilyen korlátozás lehet például, hogy bizonyos filmeket csak néhányszor lehet majd megnézni, viszont egy TV sorozat egy-egy epizódja várhatóan *1 dollár körüli* áron lesz elérhető.

➲ http://arstechnica.com/news.ars/post/20061129-8321.html

Linuxos E-Book olvasó



A *taivan*-i *eRead* kínálatában már szerepel a *Star eBook STK-101*, amely *elektronikus papírt használ*. A gyártó szerint akár 60 könyv (kb. 6-10 ezer oldal) is elolvasható egy feltöltéssel. Az eszköz emellett *hangoskönyvet is képes* lejátszani, mint ahogy *MP3*-at is.

Az eszköz kijelzője *6 hüvelykes*, *800x600* képpont felbontással bír és *16 szürkeárnyalatot* tud megjeleníteni. *200 MHz-es ARM-920T* processzor bújik meg a burkolat alatt. A *Linux* egy 4 megabájtos *ROM*-ból indul és működés közben 16 megabájt memóriával gazdálkodhat. Felhasználói állományok tárolására a beépített *64 megabájtos DataFlash*-t használhatjuk, de *SD* vagy *MMC* kártyával akár *1 gigabájtig* is bővíthetjük. Az eszköz alapterületre valamivel nagyobb, mint egy postai képeslap (188 x 118 mm), vastagsága pedig csupán 8 mm.

A készülék ára *400 dollár*, de ezért kapunk a készülék mellé hálózati adaptert, fülhallgatót, akkumulátor, bőrtokot, illetve egy 512 megabájtos *SD* kártyát.

➲ http://www.linuxdevices.com/articles/AT5277698708.html

Linux? Oui!

A francia csendőrség és a kulturális minisztérium már *Linuxot* használ. Hamarosan a *francia parlament* is nyílt forrásra vált, melynek keretében több, mint *ezer munkaállomásra kerül Linux*, *OpenOffice.org*, *Firefox* és nyílt forrású email kliens.

➲ http://news.zdnet.com/2100-3513_22-6138372.html

Évfordulók

35 éves az Intel 4004-es processzor

Az *Intel 4004*-et jellemzően számológépekben alkalmazták. Az eltelt 35 év alatt az integráció foka hihetetlenül megugrott, hiszen a *4004-es csupán 2300 tranzisztort* tartalmazott, míg egy mai *Core Duo több, mint 291 milliót*.

A *4004 órajele 740 kHz* volt, míg *4 kilobájt memóriát* tudott *megcímezni*.

A 35. születésnap alkalmából az *Intel* elérhetővé tette a processzor kapcsolási rajzát és a kézikönyvét.

➲ http://www.linuxdevices.com/news/NS8934778390.html

➲ http://en.wikipedia.org/wiki/4004

Idén 16 éves a web

16 évvel ezelőtt készült az *első weboldal. A* weboldalak alapvetően meghatározták az *Internet* történetét. *Internet* persze már korábban is volt (levelezés, hírcsoportok, stb.), de egészen a 1990-ig csupán a lakosság kis része használta napi rendszerességgel. A robbanásszerű gyarapodás bizonyíthatóan a weboldalaknak köszönhető.

➲ http://www.w3.org/History.html

15 éves a PGP

Phil Zimmerman 1991-ben adta ki *PGP*-jét (*Pretty Good Privacy*), mely bárki számára lehetővé tette a biztonságos és titkosított információ továbbítást.

➲ http://www.net-security.org/secworld.php?id=4410

10 éves

10 éves a *CSS*. A *CSS* nem más, mint stíluslap, amely lehetővé teszi, hogy az oldalaknak pillanatok alatt más megjelenést adjunk. A korábbi megoldáshoz képest különvált a tartalom (*html*) és a dizájn (*css*).

➲ http://www.w3.org/Style/CSS10/

Dizájnos TV/PC

A *hongkong-i Quataris* bemutatta *Ottimo* modelljét, mely a belsejét tekintve egy *Pentium 4-es* gép *LCD monitorral* szerelve. Persze ez még nem lenne érdekes, de az eszköz előre telepített *SUSE Linux*ot futtat. A tárhelyről *80 gigabájtos merevlemez* gondoskodik, míg az operatív *memória 256 megabájt*, amely 2 gigabájtig bővíthető. Az eszköz rendelkezik gigabites *hálózati csatolóval*, *tv tunerrel* és *DVD olvasóval*. A képernyője *16:10 képaránnyal* bír, és 15, 17 vagy 19 hüvelykes változatban is kérhetjük.
Az eszköz ára *1000 euró* körül várható.
➲ http://www.linuxdevices.com/news/NS9175573311.html

Java 6

A kétéves fejlesztés eredményeként már elérhető a népszerű *Java* platform *6*-os verziója. A Sun *330 külső fejlesztővel* és *160 céggel működött együtt* annak érdekében, hogy biztosítsák a kompatibilitást a korábbi rendszerekkel, illetve a stabilitást és a megfelelő teljesítményt.
➲ http://osnews.com/story.php/16689/Java-6-Released/

Debian Etch – befagyasztva

Andreas Barth, a *Debian* kiadásokért felelős csapat tagja, bejelentette, hogy befagyasztották a *Debian Etch-et,* mely *a* következő stabil verzió lesz. A kiadásig azonban még pár hibát javítani kell, de új csomag (vagy egy csomag újabb verziója) már nem kerül bele.
➲ http://www.debian.org/News/weekly/2006/42/

Kernel szintű virtualizáció

Andrew Morton szeretné a *2.6.20-as kernel verzióba* beletenni a *kernel szintű virtualizációt.* Ez a gyakorlatban azt jelentené, hogy a *VMWare*, *Xen*, *qemu* mellett maga a *Linux kernel* is képes lenne futtatni módosítatlan *Linux* vagy *Windows* rendszereket.
➲ http://osnews.com/story.php/16746/KVM-To-Be-Merged-Into-Linux-Kernel-2.6.20/

Indiai VoIP kérdés

Elképzelhető, hogy hamarosan nem telefonálhatunk *Indiából* vagy *Indiába* számítógép segítségével (*Skype, Yahoo,* stb.) a kormány hatására. Hogy ez befolyásolja-e a hagyományos telefonról indított *VoIP* hívásokat, arról nem szól a híradás.
➲ http://economictimes.indiatimes.com/articleshow/726843.cms

Biztonságos laptop akkumulátorok

A *Matsushita* (a *Panasonic* termékek gyártója) kifejlesztette a *biztonságosabb* litium-ion *akkumulátort* laptopok számára. Korábban az anódot és a katódot csupán egy vékony réteg választotta el, így az akkumulátorok esetleges sérülése vagy túlmelegedése esetén a könnyen kigyulladhattak és felrobbanhattak. A *Matsushita* újítása abban áll, hogy az anód és a katód közé jobb szigetelési képességgel bíró anyagot helyezett, amely meggátolja a túlmelegedést még akár rövidzárlat esetén is.
➲ http://arstechnica.com/news.ars/post/20061218-8442.html

Bináris kernel modulok

Korábban felröppent a hír, miszerint 2008 januártól *kizárnák a bináris kernel modulok* használatát. (Jelenleg például jó néhány *WIFI* kártya rendelkezik ilyen meghajtóval.) *Linus Torwalds* azonban ezt az ötletet badarságnak tartja. A felvetésre válaszolva a *DRM*-hez (digitális jogkezelés) hasonlította a dolgot, ahol az ember nem szabad (free). Véleményében megjegyezte azt is, hogy amennyiben a nagyobb *Linux* disztribúciók nem ellenzik ezt a tiltást, úgy ő nem akadályozza meg a lépést. Az viszont meglehetősen valószínű, hogy a bináris modulok tiltása visszavetheti a telepített *Linuxok* számának növekedését.
➲ http://arstechnica.com/news.ars/post/20061215-8428.html

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.

Helyesbítés

Olvasónk hívta fel a figyelmünket arra, hogy a Linuxvilág 2007. januári számában megjelent, az RSSH-ról szóló cikk egyik állításával ellentétben a FreeBSD illetve NetBSD rendszerben már megtalálható a wordexp() függvény, illetve van rá áthidaló megoldás. Az OpenBSD port gyűjteményében szereplő rssh csomag szintén tartalmazza azt. Köszönjük az észrevételt!

Mi újság a rendszermag fejlesztése körül?

Michael Halcrow olyan javítócsomagokat készített, amelyek támogatják a nyilvános kulcsú titkosítást az *eCryptFS*-ben. Úgy tűnik, az *Andrew Morton* féléknek tetszik az ötlet, bár *Andrew* arra hívja fel a figyelmet, hogy új kód írása helyett talán szerencsésebb lenne a rendszermagban már megtalálható kulcskezelést kibővíteni az *eCryptFS* nyilvános kulcsú jellemzőinek támogatására. *Michael* azonban úgy érzi, jó úton halad, és úgy tűnik, az *eCryptFS* nyilvános kulcsú támogatásának ideje elérkezett, függetlenül attól, hogy végül milyen kód valósítja majd meg.

Alon Bar-Lev 255-ről 2048 karakterre bővítette ki a rendszermagbetöltő parancssor hosszát, hogy az elmúlt években felgyülemlett kapcsolók, például modulparaméterek, *initramfs*, *suspend*, *resume* kényelmesen elférjenek. Sajnos az is kiderült, hogy a rendszermagbetöltő parancssort nem lehet csak úgy megváltoztatni. Az assemblyben írt kód olyan furmányos és rosszul megtervezett, hogy még az egyszerűbb változtatások is nagyobb javításokat eredményeznek. *Andy Kleen*, *H. Peter Anvin*, *Alon* és mások azonban mindezt lehetőségnek tekintették, hogy végre felszámolják a kupit. A tisztogató akció valószínűleg késleltetni fogja a parancssor hosszának 2048 karakterre bővítését, de egyúttal olyan lehetőségeket is feltárhat, amelyekre eddig még csak nem is gondoltak.

Greg Kroah-Hartman és *Thomas Gleixner* szerint a felhasználói térben működő *PCI*-eszközmeghajtók írása bolondság, tehát úgy döntöttek, valamit tenniük kell. A folyamat megállításához, *Thomas* infrastruktúrakódot dobott össze, amit még *Greg* is bütykölt. Miután közzétették, egy maroknyi csapat, köztük *Andrew Morton*, elkezdte olyan formába önteni, hogy a rendszermagba kerülhessen. Úgy tűnik, a kód végül nem egyszerűen *PCI*-, hanem általános eszközmeghajtó-alrendszer lesz a felhasználói térben, ami már be is indította a keresztelő előkészületeit – egy sor elnevezés született, a *User Space Driver*-től (*USD*, *eszközmeghajtó a felhasználói térben*) a *Framework for Userspace Drivers*-ig (*FUD*, *keretrendszer a felhasználói eszközmeghajtókhoz*). A magam részéről egy *FUD* nevű alrendszert látnék szívesen. *Manu Abraham* már türelmetlenül várja a teljes implementációt, mert simábbá tenné az utat az *Andrew de Quincey*-vel közösen végzett munkájuk számára, és még mások is járnak hasonló cipőben.

Neil Brown kissé frusztrált amiatt, hogy manapság hány különféle módon lehet paramétereket átadni a rendszermagnak. *sysctl*, *SysFS*, modulparaméterek, rendszermag-paraméterek vagy (csak suttogni is alig merem) *ProcFS* legyen a választás, ha új modul készül? Tanácstalanságában tanácsért folyamodott. Azon túl, hogy *Horst von Brand* a *sysctl*-t javasolta, a kibontakozó vita nem adott *Neilnek* egyértelmű és megnyugtató választ, de legalább sikerült *Oleg Verych*-et vallomásra bírni az új, *etab* (*External Text and Binary*) nevű, paraméter-interfész ügyében. Az *etab* a paramétereket kulcs-érték párban tárolja, és *Oleg* szerint sok esetben használható, mikor a rendszermagot kell paraméterezni.

Joerg Roedel implementálta az *RFC 3378*-ban definiált protokollt, ami az *Ethernet* keretek *IPv4*-es csomagokban való küldésére (tunelling) szolgál. *Philip Craig* rámutatott, hogy *Joerg* kódjának logikus helye a már létező *iproute2*-ben lenne. *Joerg* ezzel egyetértett, de hangsúlyozta, hogy a kódot az *iproute2*-től függetlenül, kísérleti céllal írta. Tervezi az integrálását, mihelyt a kód elegendően stabil lesz.

Arjan van de Ven az *Intelnél* bejelentette, hogy elkészült a *Linux Firmware* fejlesztőcsomag első változatával. Az *Intelnek* ez a nyílt forráskódú kezdeményezése egy sor tesztet is tartalmaz annak vizsgálatára, hogyan működik együtt egy rendszer *BIOS*-a a *Linuxszal*. *Arjan* reményei szerint ez segítséget nyújt majd a *BIOS*-fejlesztőknek, hogy ez a cél minél jobban megvalósuljon. Az *Intel* is reménykedik, hogy a fejlesztők használják majd az új eszközt, segítségével hibákat javítanak és további *BIOS*-ok támogatásáról gondoskodnak.

Linux Journal 2006., 153. szám

Zack Brown

Személyes adataink védelme

Már Magyarországon is szinte mindenhol elérhető a szélessávú internet, legyen az ADSL, kábeltv vagy vezeték nélküli – legyen az mobiltelefonos vagy WIFI. Ez sok terhet levesz a felhasználó válláról, hiszen nem kell például egy átutalásért a bankba rohangálni. Viszont sokkal sebezhetőbbé váltunk.

Az ideális bank

Egy bank esetében nagyon fontos, hogy a tranzakciókról – legyen az akár ezer forint vagy akár több millió – csupán a küldő és a fogadó félnek legyen tudomása.

1. ábra Így néz ki egy Netlock tanúsítvány

Ennek megfelelően a bankok titkosított oldalakon (***https***) vagy saját titkosító programmal biztosítják a megbízható internetes banki ügyintézéseket. Persze ez ahhoz, hogy egy bank megbízható legyen, szüksége van ***hiteles tanúsítvány***ra. Ilyet ***Magyarország***on például a ***Netlock***, külföldön a ***Verisign*** adhat, de számos egyéb tanúsítvány kezelő cég van. A tanúsítvány egyfajta virtuális személyigazolványként is felfogható.

Phising

Phisingnek, vagy más néven ***adathalászat***nak hívják azt, amikor ***személyes*** – sokszor ***bizalmas*** – adatokat próbálnak ***megszerezni*** tőlünk.

Nyilvános és titkos kulcsok

A dolog lényege, hogy két ***kulcs*** van a rendszerben: egy ***privát*** és egy ***publikus***. A mechanizmus röviden úgy működik, hogy minden banki tranzakció esetén a bank publikus kulcsával titkosítja a böngésző a küldött csomagokat. Ezt a titkosított adatcsomagokat ezek után azonban már csak a bank tudja megnézni a privát kulccsal, éppen ezért az ilyen típusú titkosításnál a privát kulcsot titokban kell tartani, míg a publikus kulcs bárkinek kiadható. Vagyis nem igaz a következő egyenlet: adat + publikus kulcs - publikus kulcs = adat. A megoldás briliáns, noha nem feltörhetetlen. Megfelelő számítási kapacitással – ne tízezer gépes klaszterben gondolkodjon az Olvasó – fel lehet törni, de anyagilag egyelőre nem biztos, hogy megéri.

Kis kitérő: néha fel-felbukkannak hírek az úgynevezett ***kvantum számítógép***ekről, azonban ezek még a gyakorlatban nem használhatóak, viszont ha valaha is elkészülnek, nagyságrendekkel gyorsabbak lesznek a mostaniaknál. Tehát amihez most kell egy tízezer gépes klaszter, ahhoz a kvantumszámítógépek korában egy is bőven elég lesz. Ennek megfelelően a jelenleg használt kódolási megoldások is értéktelenné válhatnak. Persze ehhez elég sok víznek kell lefolynia a ***Dunán***.

Whois – Állj, ki vagy?!

www.budapestbank.net

```
domain:       budapestbank.net
owner:        Hugh Funderburg
email:        vrwerfverw41@yahoo.com
address:      12913 Melrose Rd
city:         Caledonia
state:        -
postal-code:  61011
country:      US
phone:        +815.8851429
admin-c:      CNET-626553 vrwerfverw41@yahoo.com
tech-c:       CNET-626553 vrwerfverw41@yahoo.com
billing-c:    CNET-626553 vrwerfverw41@yahoo.com
nserver:      ns1.willbe-one.com
```

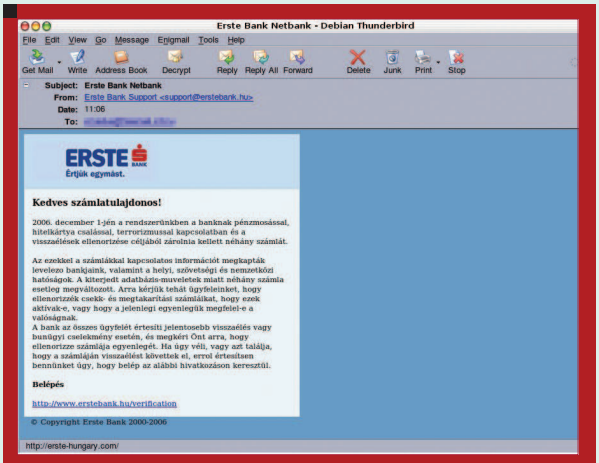
folytatás

```
nserver:        ns2.willbe-one.com
status:         hold,infringe-3rd-parties
created:        2006-11-30 22:07:00 UTC
modified:       2006-12-05 11:06:38 UTC
expires:        2007-11-30 22:07:00 UTC

contact-hdl:    CNET-626553
person:         Hugh Funderburg
email:          vrwerfverw41@yahoo.com
address:        12913 Melrose Rd
city:           Caledonia
state:          –
postal-code:    61011
country:        US
phone:          +815.8851429
```

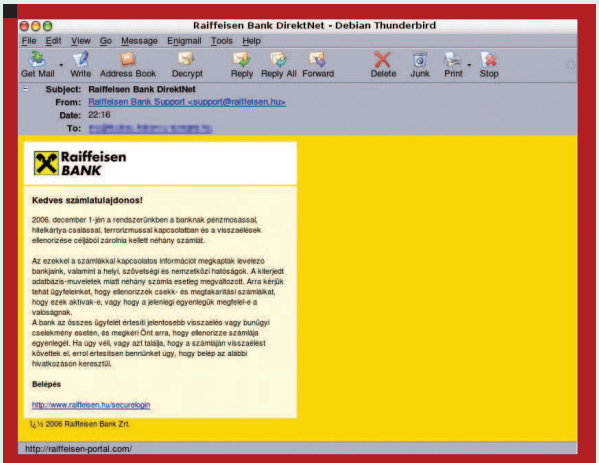
www.budapestbank.hu

```
domain:         budapestbank.hu
org:            org_name_eng: Budapest
                ➥Bank Co.
org:            org_name_hun: Budapest
                ➥Bank Rt.
address:        Váci út 188
address:        H-1138 Budapest
address:        HU
phone:          +36 1 4506000
fax-no:         +36 1 4506032
hun-id:         0990603035
admin-c:        2000243871
tech-c:         2000275450
zone-c:         2000275451
domain_pri_ns:  serval.bbrt.hu[195.56.141.43]
registered:     1999.06.03 20:22:53
changed:        2004.02.05 15:17:50
registrar:      1990930008

. . .
. . .
. . .

org:            org_name_eng: T-Online
                ➥Hungary Co.
org:            org_name_hun: T-Online Magyar
                ➥ország Zrt. (Registrar)
address:        Postafiók 204.
address:        H-1364 Budapest
address:        HU
phone:          +36 1 3713400
fax-no:         +36 1 3713405
hun-id:         1990930008
```

Mindkét lekérdezés 2006. december 7-én történt. Melyik hihetőbb?



2. ábra Hamis Erste Bankos e-mail



3. ábra Hamis Raiffeisen Bankos e-mail

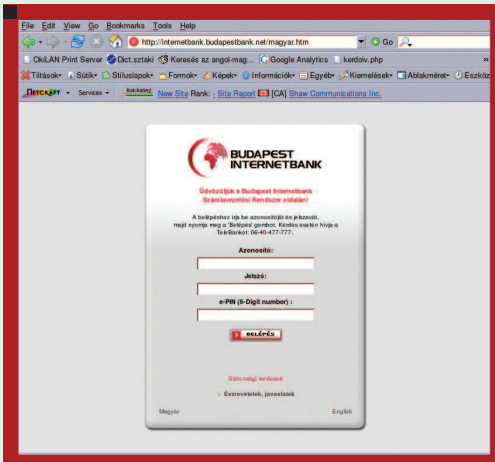
A ***phising*** célpontja leggyakrabban banki adat szokott lenni, de gyakran van példa más számra (például *IMEI*).

A sikeres ***phising*** azon áll vagy bukik, hogy a felhasználót sikerül-e meggyőzni, hogy ő most ténylegesen az adott oldalt nézi, és nem csak egy csali oldalt. Ezt legtöbbször a hasonló dizájnnal és tartománynévvel rendelkező oldal tudja elhitetni, de nem kell túl paranoiásnak lennünk, hogy ezt egyszerű eszközökkel kivédjük.

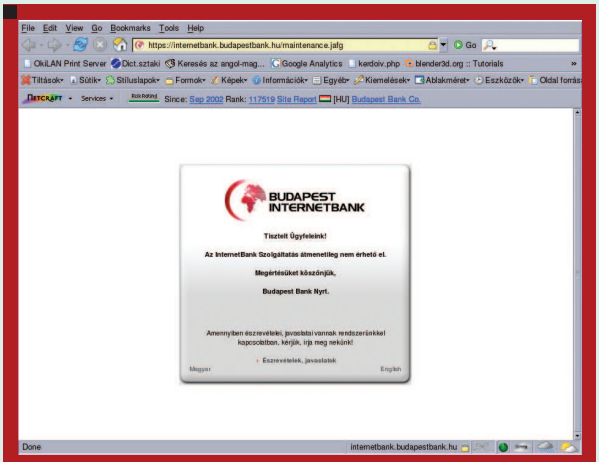
Hogyan védekezhetünk?

A ***phising*** ellen többféle védelem is létezik, de az alábbiakból minél többet alkalmazunk, annál nagyobb biztonságban érezhetjük magunkat. Persze teljes biztonság nincs.

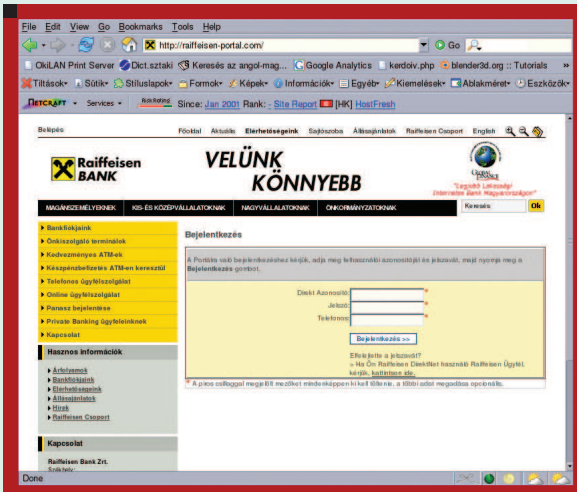
Az első és legfontosabb: ***jegyezzük*** meg fejből az online ***bankunk internet címét***, vagy legalábbis az elejét. Ezzel elkerülhető, hogy esetleg egy nem megfelelő oldalon adjuk ki adatainkat.



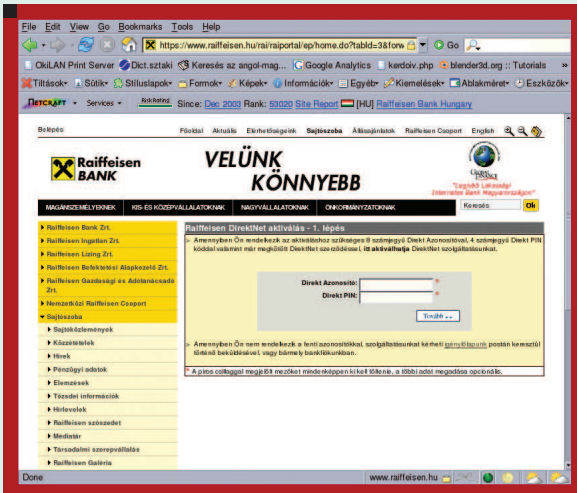
4. ábra A hamisított Budapest Bank nyitó oldal (2006.12.05.)



5. ábra Az igazi Budapest Bank nyitóoldala (2006.12.05.)



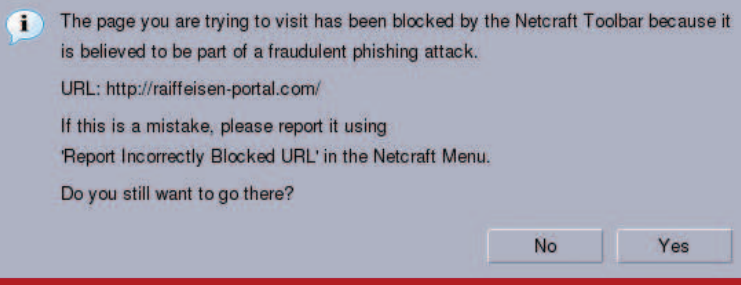
6. ábra Hamis Raiffeisen Bank nyitó oldal (2006.12.06.)



7. ábra Valós Raiffeisen Bank nyitó oldal (2006.12.06.)

A bank ***e-mailben*** és ***sms-ben nem kér adatokat***, illetve ***nem kéri***, hogy az e-mailben szereplő ***linkre kattintsunk***. Amennyiben ez történik, nézzük meg, hogy az e-mailben szereplő link milyen címre mutat. Fontos, hogy mire mutat és nem az, amit kiír. Ezt például ***Thunderbird***-ben úgy tudjuk megnézni, hogy a ***link fölé visszük az egeret*** és alul a ***státusz sorban*** megjelenik a hivatkozás. Ha a hivatkozás ***https*** helyett ***http***, akkor mindenképp gyanakodjunk.

E-mail esetén hasznos lehet az email forrásai is. Egész pontosan az utolsó ***Received from*** mező. Ritkán jön (értsd: ***soha***) magyar banki e-mail ***demon.nl***, ***ninja.com***, vagy ***wanadoo.fr*** tartományokból, mint ahogy az decemberben történt.
A banki weboldalakon ***nem kérik egyszerre*** az ***azonosítót***, a ***jelszót*** és a ***telebank kódját***. Ezzel is elég jól szűrhető az adathalászat, amennyiben nem rutinból töltjük ki a bejelentkezési ablakot.
Ha a bankunk támogatja, érdemes ***SMS értesítést kérni*** az online felületen végrehajtott ***műveletekről***, mint például a belépés, az átutalás kezdeményezése, illetve a jelszóváltás.
Ha nem is telepítünk minden frissítést, de legalább a ***titkosító könyvtárak frissítéseit***, illetve az online bankhoz használt ***böngésző frissítését*** tegyük meg.
Nyilvános helyen – például ***netkávézóban***, vagy más ***közös használatú gépen*** – ha lehet, ***mellőzzük*** az internetes ***banki ügyintézést***.



8. ábra Ha a Netcraft adatbázis már tud a csalásról, akkor az oldal megnyitása előtt jelez

9. ábra Egy 70 dolláros billentyűleütés figyelő (keylogger). Akár félmillió leütést is tárolhat.

A gépen bármi lehet, amely elárulja, mit csináltunk. Lehet akár egy ***trójai***, vagy egy ***keylogger*** (billentyűleütéseket megjegyző program), de akár ***külső keylogger*** is, amely a számítógép és a billentyűzet köze csatlakoztatható. Ugyanide tartozik: noha könnyedséget okoz, ***mégse mentsük le a jelszavakat*** a böngészőbe.

Gyakori hiba még, hogy ***nem titkosított VNC*** vagy ***rdesktop*** kapcsolaton keresztül jelentkezik be valaki. Ez a gyakorlatban talán nem akkora rizikó, de inkább legyünk elővigyázatosak. Ha egy e-mailben küldött banki weboldalra klikkelünk, mindenképp ***ellenőrizzük a tanúsítványt***. Ezzel például kideríthető, hogy valamelyik nagy tanúsítvány ***cég adta-e***, vagy csak egy ***saját készítésűvel*** van dolgunk...

Végül pedig, amennyiben módunkban áll, használjuk a ***Netcraft Toolbar***t. Igaz, hogy pár hasznos pixellel csökkenti a képernyőt, de oldalak böngészésénél olyan információkat ad, melyek segíthetnek leleplezni az adathalászokat. Minden oldallátogatás esetén ***kiad például egy zászlócskát*** annak megfelelően, hogy ***melyik országban*** van az adott szerver. Éppen ezért, ha egy ***magyar bankhoz kanadai***, vagy ***hongkongi zászlót ad be***, akkor mindenképp ***gyanakodnunk*** kell. Ha nem szeretnénk a ***Netcraft Toolbar***t használni, akkor ***gyanús domain esetén*** használjuk a ***whois*** parancsot.

A legegyszerűbb a végére: ha nem vagyunk biztosak a dolgunkban, ***hívjuk fel bátran a bank ügyfélszolgálatát***. Az ügyfélszolgálat telefonszáma leggyakrabban a bankkártyánk hátulján található.

2006 decemberében számos bank ügyfelét megpróbálták becsapni, azonban a saját bőrömön csupán három hamis banki email jutott el (időrendben: ***Budapest Bank***, ***Raiffeisen Bank***, illetve az ***Erste Bank***). Az ***Erste Bankról*** hamis változatáról azért nincs képernyő mentés, mert a hivatkozott oldal nem volt elérhető.

Jelenleg egyik fenti banknál sincs számlám. Ez azonban nem jelenti azt, hogy a jövőben nem fordulhat elő hasonló eset, hiszen akár a számlámat vezető bank is lehet a következő. Tehát vigyázzon a kedves Olvasó, az adathalászok már a spájzban vannak...

IMEI

Az ***IMEI*** szám a ***mobiltelefonok egyedi azonosítója*** – olyan mint a hálózati kártyáknál a ***MAC cím*** – amit azonban megfelelő programokkal módosíthat a felhasználó, azonban ez jelenleg ***Magyarországon*** és a világ nagy részén törvénytelennek számít.

Jó pár ***phising*** oldalt láttam már, amely az ***IMEI*** szám, az ország és a szolgáltató ismeretében vállalta, hogy megmondja a készülék hálózatfüggetlenítéséhez szükséges kódot. Ez azonban a legtöbb esetben csak ígéret, viszont a megadott ***IMEI*** szám más országban használható, hiszen az ***IMEI*** számok ***nyomonkövetése*** és ***tiltólistája*** csak egy-egy országon belüli szolgáltatók esetén közös legtöbbször. Előfordulhat például az, hogy ha mondjuk külföldön felbukkan a mi ***IMEI*** számunk és valami törvénytelen dolgot hajtanak végre vele – majd letiltják abban az országban –, úgy később esetleg, Mi, a jogos ***IMEI*** tulajdonosok se tudjuk használni a készülékünket.

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

Netcraft Toolbar
➲ http://toolbar.netcraft.com/

Többplatformos alkalmazások készítése a Mono fejlesztői keretrendszerrel

2005 utolsó negyedévében a Linux kiszolgálókból származó bevétel elérte az 1,6 milliárd dollárt. Ez már a 14-ik egymást követő negyedév, amelyben kétszámjegyű növekedést, az előző évhez képest összesen 20,8 százalékos bővülést regisztráltak ezen a területen. A teljes árbevétel tekinetében a Linux a harmadik helyen áll: a felhasználók egyre inkább bővítik a linuxos kiszolgálók felhasználási körét, kereskedelmi és műszaki területeken egyaránt. A Linux az asztali rendszereken is egyre népszerűbbé válik, hiszen a különféle szervezetek stabilabb, biztonságosabb és költséghatékonyabb alternatívákat keresnek.

A szoftverfejlesztés tekintetében, a ***Linux*** megjelenése az adatközpontokban és az asztali gépeken azt eredményezi, hogy a vállalati fejlesztők és a független szoftverfejlesztők (***ISV***-k) egyaránt keresik a hatékony, minél kevésbé költséges módját annak, hogy kritikus fontosságú alkalmazásaikat átírhassák más-más platformokra.
A ***SUSE Linux Enterprise Server 10*** és ***SUSE Linux Enterprise Desktop 10*** termékekben megtalálható ***Mono*** keretrendszer minden olyan eszközt biztosít a fejlesztők számára, amellyel nagyteljesítményű, többplatformos alkalmazásokat írhatnak mind a kiszolgálókra, mind az asztali rendszerekre, miközben korábbi beruházásaikat, a szaktudásba, kódba és meglévő hardverbe fektetett értékeiket megőrzik.

A .NET keretrendszer kialakulása, elemei

A ***.NET*** keretrendszert, mint szoftverfejlesztői platformot 2001-ben hozta létre a ***Microsoft*** – válaszul a ***Java*** egyre növekvő népszerűségére. A ***.NET*** keretrendszer segítségével lehetővé válik egyidejűleg a több nyelven történő fejlesztés, ráadásul számos olyan technológiát tartalmaz, amely nyílt szabványon keresztül biztosítja a kölcsönös együttműködést.

Mire jó a Mono?

A ***Mono*** projekt egy nyílt fejlesztési kezdeményezés, a ***Microsoft .NET*** keretrendszer nyílt forráskódú, ***Novell*** által szponzorált változata. A ***Mono*** segítségével a vállalati informatikusok és a független szoftverszállítók fejlesztői meglévő ***.NET*** alkalmazásaikat migrálhatják ***UNIX***-ra, és minden eddiginél nagyobb termelékenységgel fejleszthetnek új ***Linux***-alapú és többplatformos alkalmazásokat.
Az eddig hagyományosan ***Microsoft*** eszközökön dolgozó fejlesztők meglévő tudásuk és képességeik birtokában a ***Mono*** keretrendszer segítségével könnyedén elérhetővé tehetik ***.NET***-es munkaállomás- és szerveralkalmazásaikat a gyorsan növekvő ***Linux*** piac számára.
A ***Mono*** minden szükséges szoftvert tartalmaz a ***.NET*** kliens- és kiszolgálóalkalmazások készítéséhez és futtatásához ***Linux***, ***Solaris***, ***MacOS X***, ***Windows*** és ***UNIX*** környezetekben. Lehetővé teszi a vállalatok számára, hogy korábbi beruházásaikat – az alkalmazottak szaktudása, fejlesztői nyelvek ismerete, meglévő hardverek és egyéb eszközök – költséghatékonyabb módon használják ki a jövőben a szoftverfejlesztések során.
A ***.NET*** alkalmazások létrehozásához és működtetéséhez szükséges ***Mono*** környezet magja az alábbi fő összetevőket tartalmazza:

- ***Common Language Runtime*** (közös nyelvi futtatórendszer, ***CLR***)
- ***C#*** fordító
- ***.NET Class Library*** implementáció
- ***GNOME***, ***Mono***, ***UNIX*** függvénytárak
- Fejlesztőeszközök

A Mono UNIX-támogatása

Mint ***UNIX***-platformokhoz szánt fejlesztői környezet, a ***Mono*** néhány igen hasznos funkciót biztosít, ezek közül az alábbiakban felsoroljuk a legfontosabbakat.

- *Mono.Posix* – *UNIX*-specifikus alkalmazások, például démonok vagy *UNIX*-kiszolgálók előállítására szolgáló függvénytár.

- *GTK#* – Egy *API*, olyan linuxos asztali alkalmazások készítéséhez, amelyek funkcionalitásában megtalálhatóak a grafikus alkalmazások, a nyomtatás, a nemzetközi működésre való felkészítés (*internationalization*), konfigurációfelügyelet, témák kezelése és rajzolás.

- *LDAP-támogatás* – *LDAP*-hozzáférés *.NET* alkalmazások számára a *Mono.Directory.LDAP* vagy *Novell.Directory.LDAP* segítségével.

- *Adatbázisok támogatása* – Osztálykönyvtárak átfogó halmaza a legtöbb (ingyenes és jogvédett) adatbázis támogatásához.

- *Biztonsági csomag* – A *Mono* saját, az alapoktól újraírt biztonsági rendszert használ, amelyben minden szükséges eszköz megtalálható a kriptográfiai algoritmusoktól kezdve egészen a tanúsítványkezelésig és a felsőbb szintű protokollok (például *SSL* vagy *TLS*) megvalósításáig.

Mono-támogatás a Microsoft API-khoz

A fenti bővítéseken túl a *Mono* projekt támogatja a fejlesztők és felhasználók által használt *Microsoft API*-kat is, így a *UNIX*, *Solaris*, *MacOS X* és *Linux* gépek egyetlen kódalappal is elérhetővé válnak. Ezek a könyvtárak binárisan kompatibilisek a *Microsoft* csomagjaival, tehát újrafordításra sincs szükség.

A Mono fejlesztés előnyei

A *Mono* lényegesen lecsökkenti a *Linux* platform támogatási költségeit a szoftverfejlesztők számára. Hagyományosan egy új platform támogatásával kapcsolatos döntéshez alaposan össze kellett vetni a fejlesztés költségeit a várható megtérüléssel. A *Mono* jelentősen leegyszerűsíti ezt a döntést, mivel egy olyan többplatformos keretrendszert biztosít, amely együttműködik a legfontosabb *.NET*-kompatibilis összetevőkkel, mint például a *C#*-fordító és egy portolható végrehajtó rendszer, valamint egyesíti számos modern programozási nyelv előnyeit. A Windows és *Linux* fejlesztők meglévő tudásukra alapozva, eltérő platformon és különböző nyelven, de számukra ismerős eszközökkel készíthetik el a többplatformos alkalmazásokat.

> A *Windows* és *Linux* fejlesztők meglévő tudásukra alapozva, eltérő platformon és különböző nyelven, de számukra ismerős eszközökkel készíthetik el a többplatformos alkalmazásokat.

A Mono beszerzése

A *Mono* fejlesztői keretrendszer fontos része a *SUSE Linux Enterprise Server 10* és *SUSE Linux Enterprise Desktop 10* termékeknek. A *Mono* számos egyéb platformon – *SLES9*, *Windows*, *Mac*, *Solaris* és más *Linux*-disztribúciók – is elérhető. A program letölthető a *Mono* projekt weboldaláról (➲ *mono-project.com/downloads*) is.

Asztali rendszerek programozása

A *Mono* igen sok olyan alkalmazást tesz elérhetővé, amely hozzájárul a linuxos asztali rendszerek egyre növekvő népszerűségéhez. Az olyan ismerős alkalmazások, mint az *F-Spot* fényképkezelő, a *Beagle* keresőmotor, a *Tomboy* jegyzetkészítő, a *Banshee* zenelejátszó és még sok egyéb program, mind ki tudják használni a *Mono*/*GTK#* keretrendszer biztosította *API*-kat a kiváló funkcionalitás érdekében.

Linuxon

A *GTK#* a *GTK+* eszközkészlethez való *.NET* hozzárendelések, valamint válogatott *GNOME* függvénytárak együttese. A *GTK#* egy eseményvezérelt rendszer, csakúgy, mint bármely más ablakkezelő könyvtár, és lehetővé teszi teljesen natív grafikus *GNOME* alkalmazások készítését a *Mono* felhasználásával. A *GTK#* része például az a funkcionalitás is, amely a grafikus alkalmazások készítéséhez, nyomtatás kezeléséhez, nemzetközi működésre felkészítéshez, konfigurációfelügyelethez, témák kezeléséhez és rajzoláshoz szükséges.
A *GNOME*-ot használó linuxos asztali rendszerek natív eszközkészlete lévén a *GTK#* használatával készült alkalmazások legjobban *Linux* alatt futnak, de természetesen működnek majd más platformokon is, például *Windowson* vagy *MacOS X* alatt. Ezen felül a *Glade* felhasználóifelület-szerkesztővel és a *Glade#* hozzárendelésekkel egyszerűen készíthetők el grafikus felületű alkalmazások.

System.Windows.Forms

Windows Forms-nak nevezik a *Microsoft .NET* fejlesztői keretrendszer *GUI* részét, amely hozzáférést biztosít a natív windowsos ablakelemekhez a meglévő *Win32 API* felügyelt kódba ágyazásával. A *Mono* átfogóan támogatja jelen kiadásában a *Windows.Forms 1.1*-et, és 2006 végére a 2.0-s változatot is.

Linuxon és Windowson is futó alkalmazások fejlesztése

Érdemes odafigyelni néhány részletre a fejlesztési fázisban, hogy az alkalmazás helyesen fusson *Windows* és *Mono* alatt is:

- Elérési utak elválasztó karakterei: törtvonal vagy fordított törtvonal

- A fájlnevek érzékenysége a nagybetűkre (a *Windows* nem foglalkozik vele, de a *UNIX* érzékeny rá)

- Az alkalmazás *Mono* alatti tesztelése

A Mono ütemterve és a .NET 2.0

Ma már a vállalati és a tranzakciós szolgáltatások kivételével a teljes *.NET 1.1* specifikáció *Mono* alatt is fut, beleértve a *Windows.Forms* támogatást is. Ezen felül a *Mono* magában foglalja a *C# 2.0*-t, az általános támogatást, valamint a *.NET 2.0* számos *API*-jának támogatását.
A cikk bővített változata teljes terjedelmében a *Novell Connection* 2006. évi decemberi számában, vagy a ➲ *http://www.novell.com/hungary/nc/* weboldalon olvasható.

Villámgyors alkalmazás-fejlesztés Oracle adatbázison

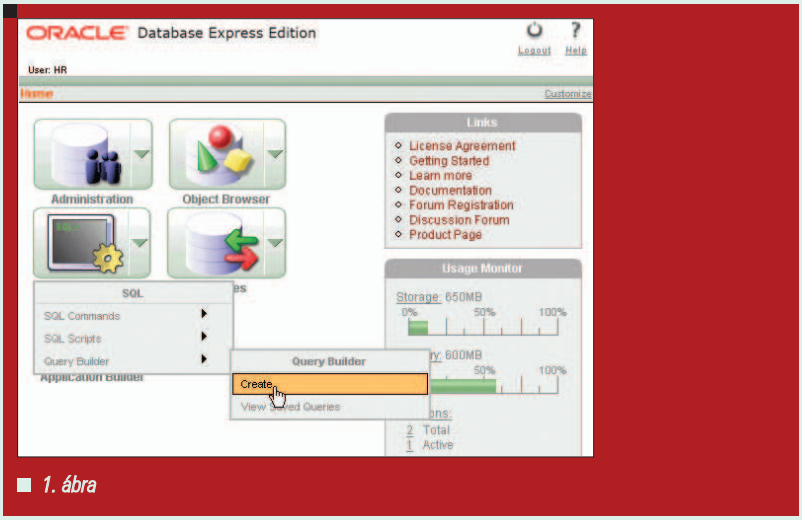
Az *Oracle Application Express* web-alkalmazás fejlesztő eszköz része az *Oracle Database 10g Express Edition* (továbbiakban *XE*) ingyenes adatbázis-kezelőnek. Az *Oracle Application Express* segítségével rendkívül gyorsan lehet *Oracle* adatbázisra webes alkalmazásokat fejleszteni. Mindössze egy böngészőre és némi alkalmazás fejlesztői tapasztalatra van szükség, hogy egyszerűen, professzionális alkalmazásokat készítsünk akár kódolás nélkül, melyek egyaránt biztonságosak és gyorsak. Az elkészített alkalmazásoknak csupán egy böngészőre van szükségük, valamint egy *Oracle* adatbázis-kezelőre, melyen elérhető az *Oracle Application Express*. Az *Application Express* nem csak az *XE* része, bármely támogatott Oracle adatbázis-kezelőre telepíthető.
Az alábbiakban egy példa alkalmazás elkészítését ismertetem.

Előfeltételek

A példa elkészítéséhez előzetesen az alábbi követelmények teljesítése szükséges:

1. Az Oracle Database 10g XE telepítése. A telepítő letölthető a http://www.oracle.com/technology/software/products/database/xe/index.html oldalról.
2. Az SQL*Plus-t elindítva lépjünk be **SYSTEM** felhasználóként (XE telepítésekor meg kellett adni a jelszavát), majd hajtsuk végre az alábbi parancsot, hogy az alapértelmezetten letiltott hr felhasználót engedélyezzük:

```
alter user hr identified by hr
➥account unlock;
```



1. ábra

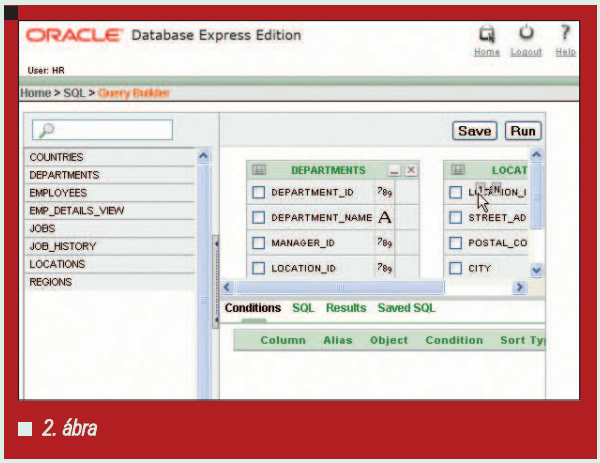
SQL lekérdezés készítése

Ebben a példában egy 3 táblára épülő lekérdezést készítünk el a Query Builder segítségével. Hajtsuk végre az alábbi lépéseket:

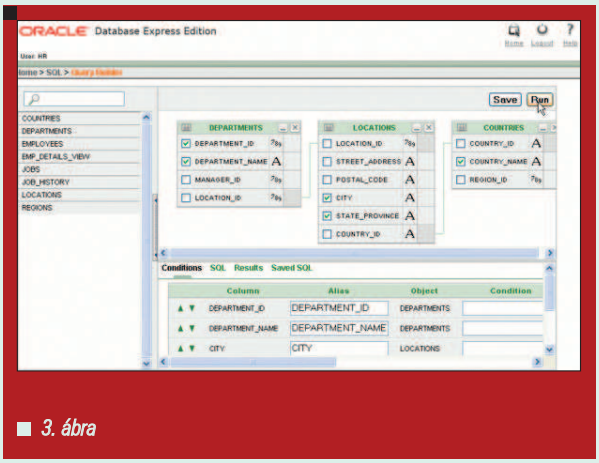
1. Böngészőben a http://127.0.0.1:8080/apex URL-t megadva indítsuk el az Oracle Application Express felületét.
2. A belépéshez adjuk meg az alábbi adatokat, majd klikkeljünk a *Login*-ra.

```
Username: hr
Password: hr
```

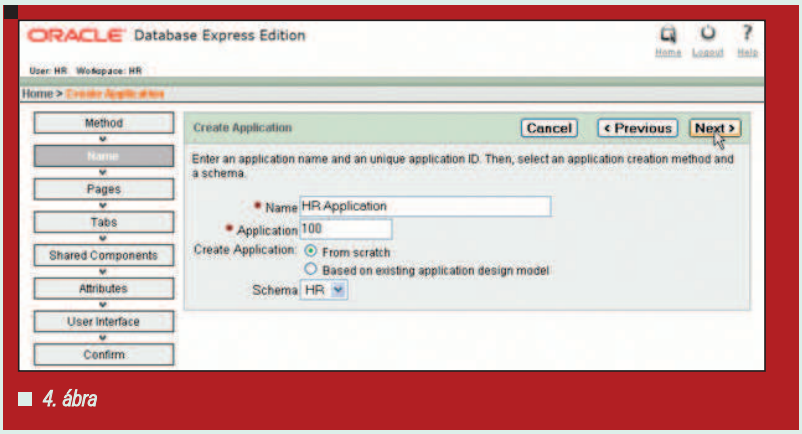
3. Lekérdezés készítéséhez klikkeljünk az *SQL* ikon melletti nyílra, és válasszuk a *Query Buildert*, azon belül pedig a *Create*-t. (*1. ábra*)
4. Klikkeljünk a *Departments*, *Locations* és *Countries* táblákra, ezzel azok a szerkesztő mezőbe kerülnek.
5. A táblák között létre kell hozni egy összekapcsolást (join), hogy az adatok lekérdezhetők legyenek. Klikkeljünk a *DEPARTMENTS.LOCATION_ID*-ra, és *„drag and drop"* módszerrel húzzuk rá az egérrel a *LOCATIONS.LOCATION_ID* oszlopra. (*2. ábra*)
6. Ezt követően megjelenik a két oszlopot összekötő vonal. Ugyanígy kell elkészíteni a *LOCATIONS* és *COUNTRIES* táblák közötti kapcsolatot is. A *LOCATIONS.COUNTRY_ID*-t kell összekötni a *COUNTRIES.COUNTRY_ID* oszloppal.
7. Miután ezzel kész vagyunk, már csak meg kell jelölni, mely oszlopok tartalmát szeretnénk lekérdezni. Ezt az adott oszlop neve előtt lévő jelölőmező bejelölésével tehetjük meg, majd klikkeljünk a *Run* gombra. (*3. ábra*)



2. ábra



3. ábra



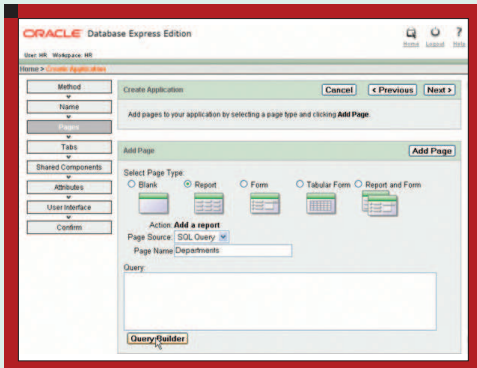
4. ábra

8. A lekérdezés eredménye megjelenik a képernyő alsó részén. A riport kilistázza az összes osztályt elhelyezkedésével (ország, megye, város szintjén) együtt. Amennyiben szeretnénk felhasználni ezt a lekérdezést alkalmazás fejlesztése során, a *Save* gombra klikkelve, például *Departments Locations* néven elmenthetjük, mely aztán a *Saved SQL* fülön meg is jelenik.

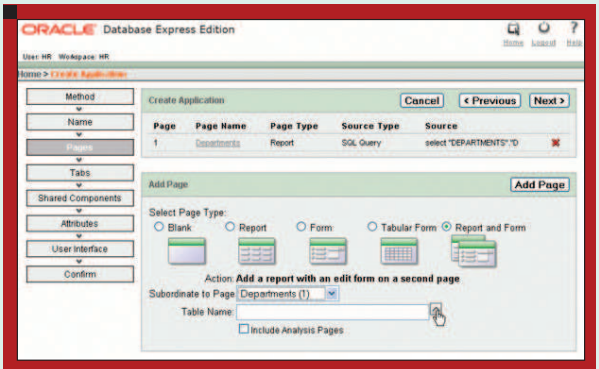
Riport alkalmazás készítése

Az imént elmentett lekérdezés felhasználásával most készítsünk egy alkalmazást, mely tartalmaz egy riportot a *hr* adatbázis osztályairól.

1. Az *XE* főoldalán az *Application Builder* ikon melletti nyílra klikkelve válasszuk a *Create Application*-t, majd ismét a *Create Application*-t.
2. Az alkalmazás nevének adjuk meg a *HR Application* nevet, és klikkeljünk a *Next* gombra. (*4. ábra*)
3. Ezután válasszuk a *Report* oldal típust (*Page Type*), és ha az elmentett lekérdezés alapján akarunk riportot készíteni, akkor válasszuk az *SQL Query*-t az oldal forrásaként (*Page Source*), és oldal névnek adjuk például a *Departments* szót, végül klikkeljünk a *Query Builder* gombra. (*5. ábra*)
4. A *Query Buildernél* az előbbi példából ismerős oldalon a *Saved SQL* fület kiválasztva megtaláljuk a korábban elmentett *Departments Locations* lekérdezést. Ha erre ráklikkelünk, és a *Return* gomb segítségével visszatérünk az oldalkészítő formra, akkor a *Query* mezőben megjelenik a lekérdezés *SQL* kódja. Az *Add Page* gombra klikkelve a riport oldallal elkészültünk.
5. Most készítsünk egy másik oldalt, melynek típusa legyen *Report and Form*. Ez valójában két oldal lesz. A riport tartalmazni fogja egy



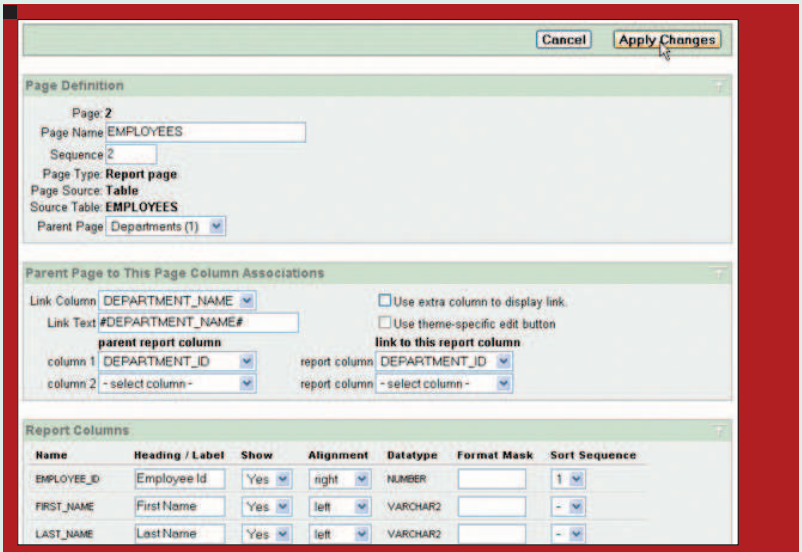
5. ábra



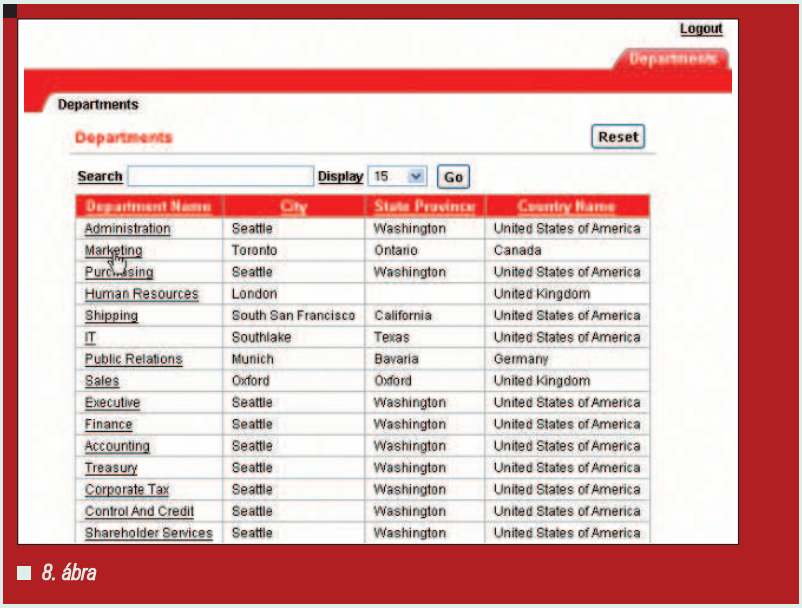
6. ábra

adott osztály alkalmazottainak listáját, míg a *Form* egy alkalmazott szerkesztő oldala lesz.

6. A *Subordinate to Page* paraméternél (melyik oldal alá kötjük be) válasszuk a *Departments (1)*-et, majd a táblanév mező melletti felfelé mutató nyílra klikkeljünk, és válasszuk ki az *EMPLOYEES* táblát. (*6. ábra*)
7. Az *Add Page* gombra klikkelve ezzel is kész vagyunk.
8. A képernyő felső részén lévő oldal struktúrán klikkeljünk a 2-es oldal *EMPLOYEES* megnevezésére.
9. A megjelenő ablakban a *Link Column* paraméternek válasszuk ki a *DEPARTMENT_NAME* mezőt, a *Parent Report Column 1* értékének pedig válasszuk a *DEPARTMENT_ID*-t, és ennek megfelelően a *link to this report column* értéke is legyen *DEPARTMENT_ID*. Ezzel beállítottuk, hogy az osztály nevére klikkelve, annak azonosítója alapján fogunk átlépni a megfelelő osztály szerkesztő oldalára (*Form*). Befejezésül klikkeljünk az *Apply Changes* gombra. (*7. ábra*)
10. Most klikkeljünk az 1-es oldal *Departments* megnevezésére ismét az oldal struktúrában, és a riport oszlopainál a *DEPARTMENT_ID* esetén a *Show* paramétert állítsuk *No*-ra, hiszen az azonosító megjelenése nem szükséges.
11. Az összes szükséges oldallal elkészültünk, klikk a *Next*-re.
12. Egy sornyi fül (tab) elegendő lesz, klikk *Next*.
13. Nem használunk osztott (shared) komponenseket, így a következő oldalon ismét csak klikk *Next*.
14. Elfogadjuk az alapértelmezett authentikációt, így ismét klikk *Next*.
15. Elfogadjuk az alapértelmezett megjelenést (theme) is, klikk *Next*.
16. Az utolsó, egyben összegző oldalon, ha mindent rendben találunk, akkor klikk *Create*.
17. Elkészült az alkalmazás 3 oldallal, plusz a bejelentkezési oldallal, így klikkeljünk a *Run Application* logóra, és *hr* felhasználóként lépjünk be az alkalmazásba.
18. Az első oldalon az osztályok listája látható, mely lekérdezést még a *Query Builder*-ben raktunk össze. Az osztály nevére klikkelve megtekinthetők lesznek az adott osztályban dolgozó alkalmazottak listája, majd az alkalmazottak adatait is szerkeszthetjük a 3. oldalra eljutva. (*8. ábra*)



7. ábra



8. ábra

Sárecz Lajos
lajos.sarecz@oracle.com

2003 óta dolgozok az Oracle Magyarországnál értékesítési tanácsadóként. Szakterületem az adatbázis-kezelő, azonban feladatom az Oracle Linuxos tevékenységének követése is.

KAPCSOLÓDÓ CÍMEK

Az ismertetett példa továbbfejlesztési lehetőségeit az Oracle Technology Network oldalán, a
➲ http://www.oracle.com/technology/obe/xe/getstarted/getstarted.htm címen lehet megtalálni.
Az Application Express-ről további információk:
➲ http://www.oracle.com/technology/products/database/application_express/index.html

Mikor szállítják házhoz a jegesmedvét?

Miután a Linuxvilág szerkesztősége örökbefogadta Tóbiást, a Budapesti Állatkert madárleányát, többen többször nekem szegezték a kérdést: *„És hová tettétek a pingvint?”* Én csodálkozva néztem, s feleltem: hát elvittük egy körre és néhány tripla leszúrt rittbergerre a műjégpályára, majd estebédeltünk és koccintottunk vele egy halászcsárdában, de mikor már idétlen rendőrvicceket kezdett mesélni, bezártuk a fridzsiderbe, azóta is jégkockákat darabol. Talán elhamarkodott volt tréfákat űznöm, sosem lehet tudni, fejletlenebb érzékű humorral ki meríthet ötleteket.
Leendő szülők örökbefogadási kérelmei között kutakodva aranyos írásokat leltem, mint például: *„Sz'al, nekem kéne egy mókusmajom, de ha örökbefogadom, az hogy jut el hozzám?”*

Mások egyből keresztszülőkké válnának: *„Én egy jegesmedvét szeretnék örökbefogadni! A neve Lady lenne!”*

Vannak igen bátor (?) egyedek is: *„Szeretnék egy hím oroszlánkölyköt felnevelni!”*

És vannak, akik nem sokat lacafacáznak: *„Egy krokodil kellene, küldjetek már egy emailt, hogy mennyibe kerül és mikor szállítják a megadott címre. Üdülési csekkel lehet fizetni?”*

Azt hiszem, nem is akarnám megtudni, mi az összefüggés az üdülési csekk és a krokodil házhoz szállítása között, de azért kétlem, hogy a leendő hüllőszülő mondjuk balaton-parti sétákon andalogna pár méteres csemetéjével.
Így hát indokoltnak látszik a kérdés: milyen kötelezettségekkel, illetve milyen jogokkal jár, ha örökbefogadunk például egy 99-es évjáratú *rókakuzut*, vagy ugye esetünkben egy 98-as pápaszemest? Bizonyára egzotikusabb látványt nyújtanának reggelente az ország utcái, amint az emberek kávé után és munkafelvétel előtt leruccannának röpke lakóparki sétára egy-egy zsiráffal vagy elefánttal, és valószínűleg megnőne az ázsiója a köztisztasági vállalatok dolgozóinak; azonban ezzel egyidejűleg emelkedne a veszélyeztetett és kihaló fajok száma – állatok és emberek közt egyaránt.
Állatkerti nevelőszülőként elsősorban a *zoo* lakóinak nyújtható anyagi segítség, életkörülményeik, egészségük megőrzéséhez, javításához. Az örökbefogadó támogatásáért cserébe egy névre szóló oklevelet kap, továbbá szülői értekezleten vehet részt, melyen megérdeklődheti, hogyan s miként éldegél állati sarja, mivel tölti napjait és bendőjét. (Dicséretes, mennyire lelkiismeretes szülőkké válnak egyesek.
A legutóbbi értekezlet után volt, kinek feltűnt, hogy örökbefogadott pápua levelibékája soványabbá vált, fel is merült a gyanú, hogy tán nem csemetéje pislogott reá a terráriumban.)
Fentiek után tehát tisztázzuk a kérdést, vagyis a választ: nem, nem tartunk pingvint a szerkesztőség hűtőszekrényében, és nem, nem szállítják házhoz a jegesmedvét. Még munkanapokon 16:00 és 17:00 óra között sem...

Halusz Léna

Linux bevetés közben – Második küldetés

Cím: Linux bevetés közben – Második küldetés
Kapcsolatok, megfigyelés, hibakeresés
Szerző: Bill von Hagen, Brian K. Jones
Kiadó: Kiskapu (O'Reilly)
Oldalszám: 544
Ár: 6980 Ft

Kezdjük talán rögtön a lényeggel. Ez az a könyv, ami – bár csak nemrég vásároltam – már kétszer is megmentett attól, hogy úgy fél napot elvesztegessek az életemből dokumentációk olvasgatására. Valami azt súgja, hogy ez nem lehet véletlen.
A szerzők (***Bill von Hagen*** és ***Brian K. Jones***) mindketten tapasztalt rendszergazdák, jó néhány könyv és szakmai újságcikk szerzői. Saját bevallásuk szerint nem kívánták követni az első kötetben ***Flickenger*** által lerakott, szigorúan rendszergazdai alapokat, helyette inkább egy olyan kötetet szerettek volna megjelentetni, amiben kevesebb a kifinomult trükk és több az általános megoldás az gyakran felmerülő problémákra. Ezen az alapvető célkitűzésen kívül még néhány egyéb célt is szem előtt tartottak az anyag összeállítása során.
Ezek közül az első azon ismeretek közzététele, amelyekkel szerintük minden magára valamit is adó rendszergazdának rendelkeznie kell(ene). A fejezetek szervező elve ennek megfelelően nem a ***Probléma – Megoldás*** kettősök bemutatása, hanem a ***Probléma – Elméleti háttér – Megoldás*** hármasok tárgyalása.
A könyv másik fontos célja a problémamegoldó gondolkodás fejlesztése. Ennek megfelelően a bemutatott ötletek és megközelítések, más helyzetben, némi változtatással ugyanúgy alkalmazhatók, vagyis a szerzők nem kifogják nekünk a halat, hanem megtanítanak halászni. Bár a kiszolgálók üzemeltetése – általában – nem a Linuxszal most ismerkedők feladata, a fenti írói szemléletnek köszönhetően a kötetet a kezdők is ugyanolyan haszonnal forgathatják, mint a már tapasztalt, sokat látott rendszergazdák.
Nézzük meg egy kicsit részletesebben, mit tartogat számunkra az 515 oldalas mű. A könyv 10 fejezetben tartalmazza az újabb 100 fogást. A főbb témakörök a következők:

- Hitelesítés Linux rendszeren
- Távoli grafikus kapcsolatok
- Rendszerszolgáltatások
- Eszközök és ötletek rendszergazdáknak
- Tárfelügyelet és biztonsági mentés
- Erőforrások megosztása, összehangolása
- Biztonsági eszközök és tanácsok
- Hibaelhárítás és teljesítménynövelés
- Naplózás és a hálózati forgalom megfigyelése
- Rendszermentés, adatok helyreállítása és javításuk

A kötetben elméleti háttérrel kiegészítve szerepel a ***PAM***, az ***LDAP***, a ***Kerberos***, a ***VNC***, a ***CUPS***, az ***LVM***, ***Samba*** kiszolgáló telepítése, az ***rsync*** használat, ***NAS***, ***DHCP***, ***NTP***, a naplózó fájlrendszerek használata, a ***/proc*** könyvtár, a ***sysctl***, az ***IDS*** rendszerek, az ***SNMP*** valamint az ***MRTG.*** Esik szó a ***syslog*** üzeneteinek asztalra irányításáról, a rootkitekről és a ***dd_rescue*** programról. A rendszermentésről és az adatok visszaállításáról szóló fejezetet minden ***Linux*** felhasználónak, legyen akár rendszergazda vagy egyszerű otthoni felhasználó, érdemes elolvasnia és elsajátítania az abban leírtakat, mert később rengeteg kellemetlenségtől kímélheti meg ez a tudás!
Az írók nem átallottak olyan alapokig sem visszamenni, mint a ***/etc/inittab***, a ***/etc/hosts.allow*** és ***hosts.deny*** fájlok, a ***top***, ***df***, ***ps***, ***lsof*** és ***du*** parancsok, mindezzel egy apró, kijózanító fricskát adva azoknak, akik a legegyszerűbb problémákat is a legbonyolultabban próbálják megoldani. A könyvet olvashatjuk a szokásos módon, de kézikönyvként, ide-oda lapozgatva is kiválóan használható.
Kezdőként én haszonnal és nagy megelégedéssel forgattam a könyvet, remélem másoknak is segít a könnyebb munkavégzésben! Ha valaki 2007-ben csak egy szakmai könyvet akar elolvasni, javaslom, ez legyen az!

Leszkoven Csaba
(leszkovencsaba@gmail.com)

Múzeum a város szélén (2. rész)

Folytatjuk a januári cikkünkben elkezdett múzeumlátogatást. Ebben a részben már olyan gépek is bemutatásra kerülnek, amelyeket a mai huszonéves korosztály valószínűleg már ismer vagy akár használt is.

A múlt havi részben megnéztük a nagyvadakat, akarom mondani a nagyvasakat, most ebben a részben tehát a kisebbek kerülnek sorra.

Az energiatakarékos számológép

Az egyik személyes kedvencem a múzeumban az egyik mechanikus számológép, melyből nekem is van egy saját példány. Valamikor a kilencvenes évek közepén tettem szert rá.
Általában az összes ilyen mechanikus számológép kezelése hasonló, de leírom, mert elsőre nem triviális. A gép baloldalán egy, a jobb oldalán két kar van, amit a megfelelő irányba húzva nullázhatjuk a kijelzőket. Mindezt persze csak akkor, ha a jobb oldalon látható műveletvégző kar alsó állásban van. A gép sikeres kinullázása esetén a példa szorzást (128 x 64) a következőképp hajtjuk végre: a számok segítségével üssük be például a 128-at. Tekerjük meg a jobb oldali kart négyszer, majd a baloldalon lévő narancssárga gomb egyszeri megnyomása után hatszor. Ha mindent jól csináltunk, akkor a jobb felső kijelzőn 64 lesz, a bal felsőn pedig a végeredmény. Az összeadás és a kivonás ennél sokkal egyszerűbb. A szám beírása után tekerjük meg egyszer a kart (bal felül megjelenik a bevitt szám), majd a jobb alsó karral kinullázzuk a beviteli mezőt és beírjuk a második tagot. A kar egyik irányba tekerésével hozzáadjuk, a másik irányba tekeréssel kivonjuk a bevitt számot. Mondanom se kell, áramszünet esetén is tökéletesen működik, unatkozó irodisták pedig súlyzózhattak vele.
A 80-as években számos házi számítógép került piacra, ilyen például a *ZX 81*, a *Commodore 64*, illetve a magyar gyártmányú *Videoton TV Computer*. Számunkra e két utóbbi lesz fontosabb.

Nagy emberek nagy tévedései

A világhálón számos olyan idézet kering, amelyeket ma már igencsak megmosolygunk, pedig a maguk korában tökéletesen helytállónak tűnhettek. Lássunk néhányat időrendben!

„I think there is a world market for maybe five computers."
„Jó, ha a világon öt számítógépet el tudunk adni."
Thomas Watson – az ***IBM elnöke*** mondta 1943-ban

„Where a calculator on the ENIAC is equipped with 18000 vacuum tubes and weighs 30 tons, computers of the future may have only 1000 vacuum tubes and perhaps weigh 1□ tons."
„Míg az ENIAC 18 ezer elektroncsövet tartalmaz és 30 tonnát nyom, addig a jövő számítógépei talán már csak ezer csövet fognak tartalmazni és nem lesznek nehezebbek másfél tonnánál."
A ***Popular Mechanics*** 1949 márciusi számában írták

„There is no reason anyone would want a computer in their home."
„Nincs olyan ok, amiért bárki is akarhatna otthonra egy számítógépet."
Ken Olson, a ***Digital Equipment Corporation*** alapítója és elnöke mondta 1977-ben

„640K ought to be enough for anybody."
„640 kilobájt mindenkinek elég lesz."
Bill Gates 1981-ben

1. ábra Mechanikus számológép

2. ábra

A magyar iskola-számítógép

A külföldi gépek beszerzési nehézségei miatt a magyar ***Videoton*** cég elkezdett iskolai számítógépeket gyártani,

3. ábra Napelemes zsebszámológép bankkártyaméretben a kilencvenes évek elejéről

4. ábra Videoton TV Computer

5. ábra Videoton TV Computer billentyűzete

6. ábra BRG magnetofon, a Videoton TV Computer jellemző szalagos egysége

7. ábra Sinclair ZX81

A CoCom lista

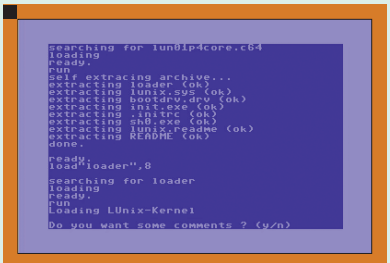
A hidegháború alatt szinte lehetetlen volt nyugati számítógéphez jutni többek között ***Magyarországon*** is. Volt egy lista azokról a – többnyire kelet-európai – országokról, akiknek a ***17 CoCom tagország*** – köztük ***Japán*** és az ***USA*** – és néhány pártoló ország nem szállított elektronikai eszközöket.

A rendszerváltáshoz közeledve egyre inkább érezhető volt a ***CoCom*** embargó gyengülése. Jó példa erre a ***80-as évek végén*** a ***„Commodore 64-turizmus"*** ***Bécs***be. Az se volt ritka, hogy egy-egy számítógépet éppen az embargó miatt mezőgazdasági gép alkatrészként hoztak be az országba.

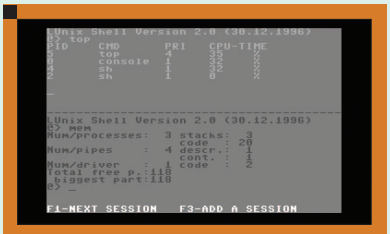
mely ***Videoton TV Computer*** néven került az üzletekbe ***32*** vagy ***64 kbyte memóriával***. A gép lelke egy ***3 MHz***-es ***Z80***-as processzor. A gép tömeges elterjedését a viszonylag magas eladási ár (az akkori ára 14-16 ezer, míg egy diplomás fizetése 4-5 ezer forint volt), illetve a piacra kerülő ***Commodore***-ok népszerűsége gátolta.

Commodore 64 és a Unix

Világhálón kalandozva egy érdekes projektbe botlottam: ***Unix Commodore 64***-re. Igen, jól olvasta a kedves Olvasó. ***LUnix*** néven elkészült egy ***Unix kompatibilis*** rendszer, azonban pár megszorítás adódik a gép képességei kapcsán: ***32 processz***, illetve maximum



8. ábra A LUnix indulása VICE emulátorban



9. ábra Két processz LUnixban (top és mem)

10. ábra Commodore 64 és egy hajlékony lemez meghajtó

11. ábra Commodore 64 billentyűzet

12. ábra Commodore gépek

13. ábra

14. ábra Commodore 64 és a néhány különböző hajlékony lemez-meghajtó hozzá

15. ábra Néhány Apple gép

16. ábra MicroVax

7 prioritási szint. Mindez egy ***64 kbyte memóriával*** rendelkező gépre, melynek sebessége ***1 MHz***.
A gép ***gyártását 1994-ben befejezték***, ennek ellenére azonban most is rengeteg jó állapotú kerül elő a szekrények mélyéről. A konstrukció sokoldalúságát jelzi, hogy ***Szegeden*** sokáig (évtizedekig) egy ***Commodore 64*** vezérelte a ***Dugonics téri zenélő szökőkutat***.
(A zene alapján előre programozva változtatta a különböző vízsugarak magasságát.)
A másik érdekesség, hogy a géphez készült kiegészítés, mellyel akár ***internetre*** is ***csatlakoztathatjuk***, illetve hagyományos ***IDE merevlemezeket*** is használhatunk a ma már kevésnek tűnő 170 kbyte-os hajlékony lemeze mellett.

Egzotikus darabok

Találkozhatunk olyan darabokkal is a gyűjtemény keretében, amelyeket hétköznapi ember nem nagyon használ. Ilyenek például a ***SUN*** és ***Apple*** gyártmányú gépek. Akik komolyan foglalkoznak ***Linux*** és ***BSD*** disztribúciókkal, annak nem kell mondanom, hogy ezeken a gépeken a saját operációs rendszerük mellett (***Solaris*** és ***Mac OS***) a fentiek is futtathatók, így például egy mai szemmel fapados ***Sun Sparc Classic*** is beállítható akár otthoni szervernek.
Persze vannak még ennél is egzotikusabb gépek, ilyen például a ***DEC*** gyártmányú ***MicroVax***, illetve a ***francia Telmat*** gyártmányú szerverek.
A kiállított ***Telmat*** szerverek némelyikéhez személyes élményem is fűződik, hiszen 2001 tavaszán ilyenen ismertem meg az ***AT&T Unix***-ot.
A mai ***BSD***-k (***Free***, ***Net***, ***Open***) az egykori ***AT&T Unix szabad implementációi***.

Telekommunikáció

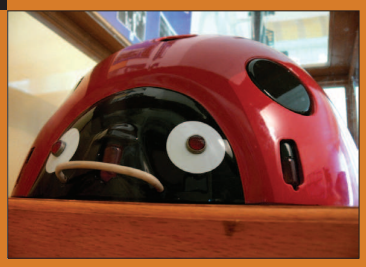
Természetesen nem lenne teljes a múzeum telekommunikációs eszközök nélkül. Helyet kapott néhány különböző generációs – leselejtezett – ***telefonközpont***, ***telefon***, ***mobiltelefon***.
A legnépszerűbb ebben a kategóriában mégis a ma is működő

17. ábra Két Telmat szerver

18. ábra A Telmat logója

Katicabogár

A múzeumban megtekinthető a ***katicabogár*** is, melyet ***Muszka Dániel*** tervezett és épített meg ***Kalmár László*** támogatásával ***1957-ben***. A szerkezet 60 centi hosszú, 40 centi széles, 25 centi magas. Az eszköz ***célja a feltételes*** és ***feltétlen reflexek modellezése***, különlegessége, hogy mind a mai napig működőképes. A katicabogár viselkedését és mozgását fény, hang és fizikai kapcsolat (érintés, ütközés) befolyásolja.



19. ábra A katicabogár

20. ábra Telex gép

21. ábra Leselejtezett telefonközpont rész

és kipróbálható ***két darab telex gép***, melyek egymás között a múzeumban üzenetet is tudnak váltani.
A ***telex***et a legegyszerűbb egy ***villanyírógép*** és ***nyomtató*** párosnak elképzelni, melyek akár több száz kilométerre is lehettek egymástól. Minthogy a ***fax***szal szemben nem képpontok, hanem csupán betűk kerültek átvitelre, így zajos vonal esetén is meglehetősen gyorsan lehetett adatot küldeni. Természetesen a gyors alatt itt csupán 100 ***baud*** körüli sebességet kell érteni. Ez másodpercenként körülbelül 10 bájtnak felelt meg. A megvalósítás előnye a korábbi távírógépekkel szemben: használatához ***nem kellett ismerni*** a ***Morze ABC***-t.
Ezzel véget ért a múzeumlátogatásunk. Nyilván minden kiállított darabról szólhatna egy-egy cikk, de történelemből egyelőre elég ennyi, hiszen képek és elbeszélések alapján elég nehéz megtapasztalni az informatika múltját. Érdemes tehát mindenképp meglátogatni a múzeumot. Egyszer legalábbis.
Végül pedig ha a szekrényben vagy a padláson porosodik egy-egy ritka informatikatörténeti relikvia, amit már az Olvasó nem szívesen pakolgat, de kidobni sajnálja, akkor annak itt a helye.

Látogatás és támogatás

Mint minden non-profit vállalkozás, így az ***Informatikatörténeti Múzeum Alapítványa*** is szívesen fogadja a támogatásokat, felajánlásokat, legyen az munka vagy anyagi jellegű. Anyagi jellegű támogatást az ***Alapítvány számlájára*** utalhatunk (11670009-07803500-70000002), végül pedig befizetett személyi jövedelemadónk (***SZJA***) ***1%-nak felajánlásával*** is támogathatjuk a alapítványt. Ebben az esetben a rendelkező nyilatkozatra ezt a számot írjuk: 18036170-1-01
Segítség felajánlását és látogatási igény bejelentését a múzeum honlapján (➲ ***http://www.infmuz.hu***) található elérhetőségeken várja ***dr. Bohus Mihály***, ***Csorba Béla*** és ***dr. Muszka Dániel***.

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

Videoton TV Computer
➲ http://tvc.homeserver.hu/

Informatikatörténeti Múzeum
➲ http://www.infmuz.hu/

LUnix
➲ http://lng.sourceforge.net/

Egyszerű weboldalak készítése DocBook XML és CSS használatával

Hogyan építsünk egyszerű tartalomszolgáltató weboldalakat DocBook XML és CSS használatával?

Eredetileg arra találták ki a világhálót, hogy információinkat könnyen elérhetővé tudjuk tenni. Manapság a webfejlesztők a stílusra és a marketingre koncentrálnak, mégis ugyanúgy jelen van az igény a könnyen és gyorsan összeállítható weboldalak létrehozására, mint amikor ***Tim Berners-Lee*** először megálmodta a ***HTML*** szabványt. Az ő szemléletét vettem át a ***DocBook XML*** és a ***CSS*** használatával, illetve kerestem néhány könnyen elérhető linuxos eszközt, melyek lehetővé teszik, hogy egyszerű, tartalomra koncentráló weboldalakat hozzak létre – a „szegény ember tartalomkezelő rendszerét".
Beágyazott szoftvereket fejlesztek. A ***HTML***, az ***XML***, a ***CSS*** és maga a web általában nem lényeges a munkámhoz. Nem vagyok olyan bensőséges viszonyban a ***HTML*** szabvány furcsa kiszámíthatatlanságával, mint a processzorokkal, hálózati csatolókártyákkal, soros portokkal. Mégis, ma mindenbe beszivárog a világháló. Egy beágyazott processzor ***Linux*** alatti futási képességeiről gyakran úgy győződünk meg, hogy rákeresünk az interneten.
Én az ügyfeleimet, s ők engem a világhálón keresnek meg. Bár a ***JavaScript***, a böngészőfüggetlen ***HTML***, a ***CSS***, ***PHP***, ***Ruby on Rails*** stb. szaktudás nem túl lényeges, mégis, egy bizonyos szintű HTML ismeret (illetve néhány alapvető eszköz használatának képessége, amivel egyszerű, de hasznos weboldalakat lehet gyártani) egyre inkább elengedhetetlen a szoftverfejlesztéshez csak úgy, mint sok más munkához. A ***DocBook XML*** eszközt ad a kezünkbe, amivel a tartalomra koncentráló dokumentációt tudunk készíteni, oly módon, hogy ezt sokféleképpen, például weboldalakon is könnyedén fel tudjuk használni.

Ez a megközelítés számos összetevőből áll, melyek nem teljesen függetlenek egymástól. Még ha nem is fogadható el általánosságban a szemléletmódom, egyes részei könnyen kiemelhetőek és használhatóak más összefüggésben is.
Magamat szoftveres eszközöket használó embernek tartom. Valószínűleg van számos webes fejlesztőkörnyezet, amely mindent megcsinál, ha megismeri valaki a használatát. Bizonyára van erre számos ***Eclipse*** bővítmény is. A hatékony célszerszámoknak azonban általában meglehetősen meredek tanulási görbéje van, ami csak akkor kifizetődő, ha sokat dolgozik valaki az adott eszközzel.
Ez a cikk nem a ***DocBook XML***-ről szól, hanem arról, hogy hogyan építsünk egyszerű módon weboldalakat, melyek ***DocBook XML*** szerkezetéhez a megjelenítést ***CSS*** adja meg. Nem vagyok webfejlesztő, és olyan eszközök megtanulására vállalkozom csak, melyek széles körben elterjedtek. Azaz: ***vim*** a szerkesztőprogramom, ***m4*** vagy ***Perl*** a makrófeldolgozóm és ***HTML tidy*** segítségével ellenőrzöm a helyes szintaxist – azaz ugyanazokkal az eszközökkel írom a szoftvert, mint amivel a dokumentációt. Az elmúlt néhány év folyamán az alap ***XML***-t, különösen is a ***DocBook XML***-t egyre inkább az alapvető ismereteim közé soroltam be.
Egy egyszerű ***DocBook XML*** cikk szövegsablont mindig készenlétben tartok a ***vim***ben, hogy egyből elő tudjam venni, ha indíttatást érzek valami olyan technikai szöveg megírására, ami hosszabb egy e-mailnél. A ***DocBook XML*** sablon használatával a tartalomra, a lényegre tudok koncentrálni. Világos és kifejező dokumentumokat tudok készíteni anélkül, hogy különösebb erőfeszítésbe kerülne a megjelenítés.
Nemrégiben fedeztem fel, hogy némi ***CSS*** segítséggel a ***DocBook XML*** dokumentumok közvetlenül megtekinthetőek bármely ***CSS***-képes böngészővel – mindenféle ***HTML***-lé alakítás nélkül, ami nagyban megkönnyíti a weboldalamon történő publikálást. Bonyolultabb szövegek esetén pl. az ***OpenOffice.org***-ból is lehet ***DocBook XML***

kimenetet előállítani. Egyre több alkalmazás képes manipulálni és előállítani ***DocBook XML*** formátumot, például az ***OpenOffice.org*** közvetlenül tudja ezt olvasni, sőt tetszőleges elterjedt formátumba át is alakítja, mint például ***HTML***, ***PDF***, ***Microsoft Word*** stb. Az ***XML*** egyik fő célja (melyet meglehetősen nehéz lenne tetten érni az egymással versengő ***XML*** alapú szövegszerkesztő-formátumokban) a tartalom és megjelenés elkülönítése. Szívemből tudom támogatni ezt az alapelvet.
A weboldalakon meg szoktam különböztetni navigációs és tartalmat mutató részeket. Ezeket szándékosan, fájl szinten is elkülönítem egymástól. A tartalomra koncentráló weboldalak legtöbbször nem igényelnek a navigációt, különálló dokumentumként működnek. Ma már ezeket ***DocBook XML***-ben írom meg. Régebben a ***HTML*** szabványt használtam, de akkor is törekedtem arra, hogy a tartalom és a navigáció ne keveredjen. Első lépésben egy ***HTML*** alapú megjelenítő és navigációs keretrendszert hozok létre. Elkészítem a weboldal ***HTML*** indexfájlját, amelyben ***HTML*** kereteket (***frame***) használok a három részegység: a fejléc, a menü és a fő rész elkülönítéséhez. A keretek a legtöbb webfejlesztő szemében nem számítanak szalonképes megoldásnak, talán azért sem, mert könnyen elérhető velük olyan megjelenítés, hogy a más által elkészített honlaptartalmat valaki a sajátjaként tálalja. A navigációt is gátolja a használatuk, és a fogyatékkal élő embertársaink számára is kevéssé barátságosak. Mindazonáltal én nem tudok ehhez mérhető weblap-összeállító módszerről, amellyel a navigáció és a megjelenítés ilyen egyszerűen, frappánsan elkülöníthető lenne. Vannak eszközök, melyekkel hasonló hatások érhetőek el, de az általam ismertek mindegyike beolvasztja a navigációs és a megjelenítési elemeket a „tartalom” kategóriájába. Az a célom, hogy a honlap tartalmi részeit ***DocBook XML*** struktúrában tudjam fejleszteni, melyet csak néhány stíluslap erejéig módosítok. Fontos számomra, hogy valahol máshol különüljön el egymástól a megjelenítés és a navigáció.
Van egy másik eretnek mellékhatása is ennek a megközelítésnek: nevezetesen az, hogy egyáltalán nincs szüksége webszerverre. Mindezt egyetlen böngésző segítségével is fel lehet építeni, ki lehet próbálni – nem kell webszervert telepíteni. Amikor készen vagyunk, csak kiírjuk az egészet egy ***CD***-re, és bárhol megtekinthető az eredmény egyetlen webböngészővel.
Az index oldalam lényegében ennyi:

```
<frameset class="frame" cols="140,*" bordercolor=
➥"#000000"
frameborder="0" framespacing="0">
 <frame class="frame" src="margin.html"
 ➥name="Margin" scrolling="no"
marginwidth="0" marginheight="0"
 <frameset class="frame" rows="100,*"
 ➥bordercolor="#000000"
frameborder="0" framespacing="0">
   <frame class="frame" src="header.html"
   ➥name="Header" scrolling="no"
marginwidth="0" marginheight="0" />
   <frame class="frame" src="home/index.xml"
   ➥name="Body" scrolling="auto"
marginwidth="0" marginheight="0" frameborder=
➥"0" />
 </frameset>
</frameset>
```

Ez három részt határoz meg a böngésző felületén. A menü van baloldalt, a fejléc fent és a tartalmat hordozó fő rész a fennmaradó (jobb alsó) területen. Fontos szerepe lesz a továbbiakban a `name` által meghatározott címkéknek (`margin`, `header`, `body`), hiszen ezzel hivatkozhatunk majd rájuk.
A fejlécem meglehetősen visszafogott, alapvetően csak ennyi:

```
<body class="header" id="body-header">
 <div class="header">
  <h1 class="header">My Title</h1>
 </div>
</body>
```

A `class` és az `id` kulcsszavak teszik majd lehetővé a stílusok alkalmazását a ***CSS*** révén.
A menüsáv számára létrehozott margó is hasonlóan egyszerű:

```
<body class="margin" id="body-margin">
 <div class="menu-box">
  <div class="menu" id="home">
    <a href="home/index.xml"
    ➥target="Body">Home</a>
  </div>
...
 </div>
</body>
```

A `class` és az `id` kulcsszavak megint csak a ***CSS*** stíluslapok alkalmazhatóságát szolgálják. Minden menüpontot körülveszünk egy `menu-box` stílusú téglalappal – ezeket tetszőleges számban ismételhetjük egymás után. Ennek stílusát egyéni ízlésünknek megfelelően bármikor átalakíthatjuk a ***CSS*** segítségével. A kódrészlet szerint a hivatkozás tartalma a menüpontokban megadott célpontban (`target`), azaz a “***body***” címke által meghatározott keretben (a keretrendszer fő részében) jelenik meg, lecserélve az ott korábban megjelenített tartalmat.
Az alábbi ***CSS*** kódot használom a kivilágítható menügombok előállítására:

```
div.menu-box {
 display: block;
 border-width: 2pt;
 border-color: color_bkgr !important;
 border-style: inset ;
}
div.menu {
 border-style: inset ;
 border-width: 5px ;
 background: color_menu_bkgr1 !important;
 border-color: color_menu_bkgr !important;
 color: color_bkgr !important ;
 font-weight: bold;
```

```
 font-size: 8pt;
 height: 14pt ;
 Width: 110pt;
 vertical-align: middle;
 x-margin: 5pt;
 x-padding: 5pt;
 text-align: center;
 padding-left: 5pt;
}
div.menu:hover {
 position: relative;
 top: 1px;
 left: 1px;
 border-color: color_menu_bkgr1;
 background-color: color_menu_bkgr;
}
a.menu { text-decoration: none }
```

Ezek tehát a nem-tartalmi rész kulcsfontosságú összetevői. A menüsávok hierarchikusan egymásba ágyazhatóak. Egyegy menüpont céljának (`target`) `margin`-ra állítása azt eredményezi, hogy a hivatkozott tartalom (praktikusan egy másik menürendszer) az oldalsó margó sávjában, azaz az eredeti menürendszer helyén jelenik meg. Ez a menüváltás akárhányszor megismételhető. A Microsoft Internet Explorer *CSS* kezelése (különösen is a pozicionálás) sajnos esetenként hibás, így a használatakor némi megjelenítésbeli eltérés észlelhető a megfelelően működő böngészőkhöz képest. Böngészőfüggetlen pozicionálás ugyan megoldható, de szörnyen bonyolult – az is nehezíti, hogy az *MSIE 7*-es verziója úgy old meg több *CSS* kérdést, hogy a legtöbb korábbi probléma-megkerülés használhatatlanná válik.
A háttérszínekkel kapcsolatban is óvatosságra intenék mindenkit. Életem egy szakaszát azzal töltöttem, hogy próbáltam kitalálni, miként lehet kiküszöbölni azt a kis fehér csíkot, ami a menüsáv és a fő rész között keletkezik a Microsoft Internet Exploreren, amikor háttérszínt állítok be – de hiába. Ez a cikk nem arról szól, hogy hogyan váljunk szakértőkké a különleges böngészőfüggetlen webfejlesztés terén, hanem egy egyszerű utat ajánl a tartalom szép megjelenítéséhez, függetlenül attól, hogy milyen böngészőben nézik. A pixelről pixelre megegyező, böngészőfüggetlen *CSS* megoldások számos böngésző számára igen nagy kihívást jelentenek.
Eddig nem szóltam a *HTML* fejlécről, sem olyan apróságokról, mint például az, hogy a ***color_menu_bkgr*** nem *HTML*/*CSS* szín, holott a kódban úgy szerepel, mintha az lenne.
HTML oldalaink, mint az ***index.html***, a ***header.html*** és a ***margin.html*** érvényes *HTML* fejlécet igényelnek, amiben egy vagy több hivatkozás is szerepel a kívánt *CSS* stíluslapra, például ennek megfelelően:

```
<link rel="stylesheet" type="text/css"
➥href="/css/stylesheet.css"
title="default">
```

A korábban idézett *CSS* részlet éppen ebben a ***stylesheet.css***-ben található. Ez a kód hívhat még egyéb *CSS* fájlokat is, amikre pl. azért lehet szükség, hogy egyes alapértelmezett *DocBook CSS* értéket felülírjunk vagy kiegészítsünk. Számos *CSS* stíluslap elérhető a *DocBook XML* számára – ezek közül néhány a *DocBook Wiki* oldalakon szerepel. Magam a ***badgers-in-foil***-t használom (lásd a cikk végén, a hivatkozások közt). A ***badgers-in-foil*** stíluslap tette lehetővé számomra, hogy különösebb erőfeszítés nélkül jelenítsek meg *DocBook XML* cikkeket különböző böngészőkben.
Minden *XML* oldalon két stíluslap-hivatkozást kellett beillesztenem az *XML* fejlécbe:

```
<?xml-stylesheet href="/css/docbook-
➥css/driver.css" type="text/css"?>
<?xml-stylesheet href="/css/stylesheet.css"
➥type="text/css"?>
```

A második sor nem feltétlenül kötelező, de erősen ajánlott: ennek révén lehetséges az eredeti *DocBook XML* stíluslapok alapértelmezett értékeinek kibővítése vagy felülírása anélkül, hogy megváltoztatnánk az eredeti stíluslap-fájlt.
A keretrendszer, az *XML*, a *HTML* körítések és számos ismétlődő összetevő előállítására ***m4*** makrófeldolgozót használok. A feladat hasonló könnyedséggel elvégezhető lenne *Perl* vagy ***bash***/***sed*** segítségével is. Ez teszi lehetővé számomra standard fejlécek, színek és más szövegrészek ügyes helyettesítését ***m4*** makrók segítségével. Például a `color_bkgr` is egy ***m4*** makró, amely lehetővé teszi az ugyanilyen nevű szövegrészek lecserélését az általam választott háttérszínre a honlaprendszer összes oldalán.
Ugyanazt a keretrendszert használom fel újra és újra, amikor egy-egy új honlap elkészítésére kell vállalkoznom. Ha egy új honlapot eltérő tartalommal, címekkel, színekkel szeretnék elkészíteni, akkor néhány változtatást kell eszközölnöm a makrókon. Ezek aztán addig bonyolódtak, míg végül azon kezdtem gondolkozni, hogy az előfeldolgozást illetően átálljak *Perl* használatára az ***m4*** helyett. Remek lehetőségnek tartom a feldolgozást követően a *HTML* ***tidy*** általi ellenőrzést, mivel az *XML* és a *HTML* kód előállítását automatizáltan végzem, és ez rejthet buktatókat.
Először is telepítenünk kell a *HTML* ***tidy*** és ***m4*** programokat. Alapvetően ***Debiannal*** és leszármazottaival dolgozom, így a telepítés számomra ennyiből áll:

```
apt-get install tidy
apt-get install m4
```

A legtöbb disztribúció előre csomagoltan tartalmazza az ***m4***-et és a *HTML* ***tidy***-t, de ha forrásból szeretné valaki lefordítani, akkor e programok honlapja megtalálható a cikk hivatkozásai közt.
Haladjunk tovább: egy ***pages.list*** nevű szövegfájlból indulok ki, amely az egyes oldalak alapneveit hordozza a megfelelő típussal együtt, ami lehet *CSS*, *HTML* és *XML*.

```
stylesheet,css
index,html
header,html
margin,html
home,xml
...
```

Az ***m4*** és ***HTML tidy*** futtatására egy rövid héjprogramot használok. Ez minden oldalt végignéz, és – hacsak nem fut hibára – előállítja a kívánt eredményt a megfelelő fájlokba:

```
#!/bin/sh
# $Id:
# $URL:
#dest=../test
dest=..
lname=pages.list
dopage() {
 echo "$1"
 if [ "$2x" == "xmlx" ]; then
  if ! [ -d $dest/$1 ]; then
   mkdir $dest/$1
  fi
  m4 -D_xml $1.m4 | tidy -i -xml >$dest/$1/
  ➥index.xml
 elif [ "$2x" == "htmlx" ]; then
  m4 $1.m4 | tidy -i  >$dest/$1.html
 elif [ "$2x" == "cssx" ]; then
  m4 -D_css $1.m4 >/var/www/share/css/$1.css
 else
  echo "Whoops $1 $2"
 fi
}
if [ -f $lname ]; then
 list=`cat $lname | grep -v '#' | awk '{print
 ➥$1}' | tr '\n' ' '`
 for argv in $list ; do
  page=""; fmt=""
  page=`echo $argv | awk -F "," '{print $1}'`
  fmt=`echo $argv | awk -F "," '{print $2}'`
  dopage ${page} ${fmt}
 done
fi
```

Ezzel az ***m4*** révén a standard fejlécek, a stíluslapokra mutató hivatkozások, a makróhelyettesítések és a színek helyettesítése stb. megoldódott, sőt még maguk a menüpontok is automatikusan állíthatóak elő a megfelelő makrókkal.
A fejléc részért felelős ***header.m4*** fájl így fog tehát kinézni:

```
define(_page,header)dnl
include(defs.m4)dnl
include(hdr.m4)dnl
<div class="header">
 <h1 class="header">_title</h1>
</div>
include(ftr.m4)dnl
```

Mint már szó volt róla, nem szükséges webszerver ahhoz, hogy ezeket a keretrendszereket vagy az előállított tartalmak bármelyikét meg tudjuk nézni. A weboldalak megjelenítését azonban többnyire webszerver végzi. Ennek beállításaiban tulajdonképpen semmit sem muszáj megváltoztatni; mégis, ha az alábbi ***CSS***-beállító sorokat beillesztjük a ***/etc/apache2/conf.d*** megfelelő fájljába, akkor ezzel létrehozunk egy megosztott CSS könyvtárat. Ennek révén több weblap is tudja majd használni ugyanazokat a stíluslapokat, függetlenül a weblapok elérési útjától a webszerver fájlrendszerében.

```
Alias /css /var/www/share/css/
<Location /css>
 Order allow,deny
 Allow from all
 Options Indexes FollowSymLinks MultiViews
</Location>
```

Amit itt láttunk, az egy szoftveres eszközökre építő megközelítés. Ha valaki csak néhány weboldalt készít el minimális tartalommal, nyilván nincs értelme a ***HTML***, ***XML*** fájlok vagy a fejlécek, láblécek automatikus előállításával bajlódni. Ha viszont megnő a szolgáltatott tartalom mennyisége, a változtatások gyakorisága, vagy sok különböző honlapot kell létrehozni, akkor már meghálálja magát az automatizálásra szánt idő.
Igazából alig érintettem a ***DocBook XML***-t. A szövegfeldolgozást már középiskolás éveim alatt elkezdtem egy H8-as gépen, olyan formázóprogramok használatával, mint a ***runoff***, ***nroff*** és ***text***. A tartalom és megjelenés elkülönülése számomra nem más, mint magától értetődő visszatérés nem-***ALAKHŰ*** (***nem-WYSIWYG***) gyökereimhez. (***A betűszó fantáziadúsan magyarított jelentése: Azt Látod, Amit Kapsz, HŰen.***)
Az ***XML*** alapú dokumentumok ***ALAKHŰ*** feldolgozását támogató eszközök is léteznek. Ha kényelmesebbnek tűnik az ***ALAKHŰ*** szövegszerkesztő, akkor a legkézenfekvőbb ***OpenOffice.org***-ot használni, mely ***DocBook XML*** formátumban is tud menteni. Azonban az ***OpenOffice.org DocBook XML*** képességei sem korlátlanok. Amikor egy szépen megformált ***OpenOffice.org*** vagy ***Microsoft Word*** formátumú fájlt ***DocBook XML*** struktúrájúvá alakítunk, a megjelenés néhány jellemzője megváltozhat. A normál ***DocBook XML*** szabvány sokkal inkább a tartalmat és a struktúrát veszi célba, semmint a megjelenítés részleteit. Az ***OpenOffice.org*** nem társít stíluslapot az elmentett ***DocBook XML*** dokumentumhoz, így néhány stíluselem (mint pl. a betűtípus, a betűméret, a beljebb kezdés mértéke stb.) csak a felhasználó által használt ***DocBook XML CSS*** alapján helyettesítődik be. Ha ez nem megfelelő, akkor módosítható vagy felülbírálható a stíluslap egy új, „sorba kapcsolt stíluslappal” (innen az angol név: ***Cascading Style Sheet***, CSS), melyben a kívánt értékek vannak megadva.
Ahogy korábban említettem, a magam részéről teljesen elégedett vagyok a ***badgers-in-foil*** stíluslappal. Ehhez alig kellett hozzányúlnom. Inkább az a célom, hogy minél fájdalommentesebben készíthessem el jól olvasható dokumentumaimat, majd ezek átkerüljenek egy honlapra vagy más, igény szerinti formátumba. Arról is szó esett már, hogy többnyire egyszerű ***DocBook XML*** cikk sablonból szoktam kiindulni, melybe a szövegtartalmat egyszerűn ***Vim*** segítségével írom be. Ez a sablon a ***DocBook XML***-nek csak igazán minimális részét alkalmazza. Néhány ***XML*** alapelvtől eltekintve (mint pl. a kezdő és záró kulcsszavak illeszkedése) az általam beírt szöveg sem tartalmaz a triviális formázásokon túl szinte semmit.

Gyakorlott *DocBook XML* használók az *XML* struktúrák egész arzenálját vethetik be, de a kezdők is egyre kifinomultabb írásműveket készíthetnek, apránként megismerve a használható kulcsszavakat. Megítélésem szerint a *DocBook XML*-t sokkal könnyebb használni, mint a *HTML* szabványt. Mivel az *XML* mereven ragaszkodik a kezdő és záró kulcsszavak egyeztetéséhez és a beágyazás szabályaihoz, így alig lehet váratlan következetlenségbe botlani. A struktúra és a megfelelő szövegfelépítés az *XML* dolga (felsorolások, táblázatok, bekezdések, fejezetek, szakaszok stb. jelölésével). A megjelenítésre és látványra vonatkozó információk már a stíluslapokból derülnek ki. Hozzáértő *CSS* fejlesztők kezei közt az egyszerű *DocBook XML* cikkek elegáns öltözetet nyernek. A célom azonban mégsem elegáns dokumentumok és weboldalak készítése, hanem a szövegtartalmak megjelenítése kifejező és olvasható módon, könnyen és gyorsan, különböző formátumokban. A *DocBook XML* egyre növekvő népszerűségnek örvend a webes dokumentum-formátumok között. Számos nyílt forrású projekt, mint például maga a *Linux kernel* is egyre inkább a *DocBook XML*-re támaszkodik, hiszen ez egy standard dokumentációs forma. A *Linux Dokumentációs Projekt* ajánl egy szerzői útmutatást és egy minta cikksablont, amit gyakran használok magam is, csak úgy, mint más online *DocBook XML* forrásból beszerezhető sablont. *Eric Raymond „DocBook mítosztalanító HOGYAN"-ja („DocBook Demystification HOWTO")* kitűnő magyarázatot ad arra, miért fontos a *DocBook XML*, és miért alkalmas arra, hogy a legtöbb nyílt forrású dokumentációs formátum helyébe lépjen. *Michael Smith* könyve, a *„Fogadd meg a tanácsom: Ne tanuld meg az XML-t"* (*„Take My Advice: Don't Learn XML"*) ehhez hasonlóan értékes: elmagyarázza, hogy a *DocBook XML* érdemben való használatához nem kell *XML*-profivá válni, sem a témához közel álló *XML* technológiák özönét mélységében megismerni. *Norman Walsh* és *Leonard Muellner "Alapvető Kalauz"* (*„The Definitive Guide"*) című műve sokkal több mindent elmond, mint amire valaha is szükségünk lesz, valamint beszél néhány fontos mozzanatról olyan esetekben, amikor kifinomult *DocBook XML* használatba kezdünk bonyolódni. Végül remélem, hogy ez a cikk is világossá teszi, hogy a *DocBook XML* hatékony használata nem bonyolult, és csak minimális mennyiségű új ismeret megtanulását igényli.

Linux Journal 2006., 151. szám

Dave Lynch szoftvertanácsadó. A webfejlesztés, az XML, a CSS és a HTML csak érintőleges kapcsolatban vannak a beágyazott- és rendszerszoftverekkel, melyeket többnyire Linux alá ír, lényegében hiábavaló megélhetési próbálkozásként. Egy másik életében építész – jelenleg épp saját házán dolgozik, amikor épp nem weboldalainak romba döntésével vagy ügyfelei programjainak írásával van elfoglalva.

A CIKK FORRÁSAI

➲ www.linuxjournal.com/article/9263

Programozzunk Pythonban (2. rész)
A szolid óriáskígyó

Műveletek akár a matematikában

Használtunk egyszerű műveleteket korábbi példáink során is, nézzük, mit tartogat még számunkra a ***Python***. Az osztással már találkoztunk, pontosítsuk kissé a már megismert adattípusok segítségével tudásunkat:

/ – az osztás eredménye, egész számot ad, ha egész számokat kap bemenő értékként, egyébként lebegőpontos eredményt kapunk
% – az osztás maradéka, ugyanolyan logikával dolgozik, mint az előző
// – az osztás egész számú eredménye, lebegőpontos számok osztásakor is mindig egészekre kerekít
Azaz például:

```
a=10.2
b=5
print('a%b='),a%b      #eredménye
                       #0.2
print('a/b='),a/b      #eredménye
                       #2.04
print('a//b='),a//b    #eredménye
                       #2.0
```

A számok világánál maradva megemlíthetjük még a hatványozást, melyet a ** jellel tudunk használatba venni, azaz például 13**2 jelenti 13 négyzetét.
A korábbi, mindennapokban is általánosan használt matematikai műveleteken túl a ***Python*** természetesen támogatja az informatika világára jellemzőbbeket is. Úgynevezett bitszintű művelet a >> és a <<, előbbi jobbra tolja el a biteket, utóbbi balra. Megértéséhez a kettes számrendszer alapjait kell kissé megismernünk, ami nem olyan bonyolult, mint hinnénk, csak egy csepp gondolkodást igényel.
A helyiértékek fogalmával már biztosan találkoztunk tízes számrendszerben, ebből induljunk most ki, vegyük például a 169-es számot:
169 jelentése: $1*100 + 6*10 + 9*1$
másképp megfogalmazva $1*10^2 + 6*10^1 + 9*10^0$

Szemléletesen:

számjegy:	1	6	9
helyiérték:	10^2	10^1	10^0

azaz a tízes számrendszerben a helyiértékek 10 megfelelő hatványait jelentik, jobbról balra haladva folyamatosan növekedve (mindig 10^0 az első jobbról, majd balra haladva 10^1 következik, utána 10^2 , 10^3 stb.), s ezeket szorozzuk meg a számrendszerben érvényes számjegyek (0-9) egyikével.
Kettes számrendszerben is gondolkodhatunk ugyanígy, azaz kettő megfelelő hatványai következnek jobbról balra, s ezeket kell szoroznunk a számrendszerben érvényes számjegyekkel, melyek jelen esetben 0 vagy 1 lehetnek.
Jobbról balra haladva a helyiértékek így alakulnak, segítségképp a legalsó sorban kiszámolva őket:

számjegy:	[0]	[1]	[0]	[1]	[0]	[1]	[0]	[0]	[1]
helyiérték:	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
helyiérték:	256	128	64	32	16	8	4	2	1

Ha egy tízes számrendszerbeli számot kell átváltanunk, akkor azt kell megnéznünk, melyik az a legnagyobb szám ezek közül, amely még osztható vele – annak a helyiértékén 1-es számjegy fog szerepelni –, majd az osztás maradékát újra el kell osztanunk kettő hatványai közül azon legnagyobbal, mely még éppen osztója a maradéknak – az ennek megfelelő helyiértékre is 1-et írunk –, s az osztás maradékával ugyanezt kell tennünk egészen addig, míg olyan osztót nem kapunk, mely maradék nélkül megvan az éppen aktuális számban. Nézzük lépésről lépésre mindezt a 169 esetén:
Melyik az a legnagyobb kettő hatvány, ami még osztja? 128. Tehát a 128-nak megfelelő pozíciójú helyiértékre kell 1-est írnunk, azaz jobbról a 8. számjegy már megvan. Az előtte lévő pozíciókon biztosan 0 szerepel (körülbelül úgy, mintha tízes számrendszerben azt írnánk 0000000000000128, ami ugyanaz mint 00128 és természetesen egyenlő 128-al). Ezekre a bevezető nullákra csak akkor van szükség, ha például kifejezetten 16 számjeggyel akarjuk ábrázolni a számot, kettes számrendszerben ugyanennyi számjegy esetén ezt úgy hívják, hogy 16 biten ábrázoljuk. A bevezető nullákat kiírva, és a még ismeretlen helyiértéken lévő számjegyeket x-el jelölve ennyi biztos már az ábrázolásból:
0 0 0 0 0 0 0 0 1 x x x x x x x
Előző osztásunk maradéka 41, így most ehhez keressük a legnagyobb még osztónak megfelelő kettő hatványt. Ez a szám a 32, ami egyben azt jelenti, hogy jobbról a hatodik helyiértéken is 1-es szerepel, s mivel az osztandók egyre kisebb számok, ezért az előző helyiérték és e között lévő számjegyek már biztosan nem jönnek szóba, tehát azok 0-t vesznek fel:
0 0 0 0 0 0 0 0 1 0 1 x x x x x
Az előző osztásból maradt 9, az ehhez legközelebbi helyiérték a 8 lesz, azaz jobbról a negyedik pozícióba is 1-et írunk (s 0 kerülne az előző pozíciójú számjegy és e közé – ha lenne ilyen):
0 0 0 0 0 0 0 0 1 0 1 0 1 x x x
1 kettő hatványos osztóját keresve feltűnik, hogy ez az 1 és ezúttal nincs maradék, ami módszerünk szerint az is jelenti, hogy végeztünk, azaz
0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1

állapotból tovább már nem léphetünk. Így a 169 kettes számrendszerbeli megfelelője 16 számjegyet felhasználva (16 biten ábrázolva) 0000000010101001. Ha nem ragaszkodnánk a 16bithez, akkor rövidebben (8 biten ábrázolva): 10101001.

Négyzetrácsos papír helyett monitor

Írjunk egy rövid programot ***Pythonban*** az előző munkamenet alapján, ami helyettünk a tízes számrendszerből átváltja az adott számot kettesbe:

```
szam=169
bitek_szama=8        #azaz hány
# biten ábrázoljuk a számot

for i in range(bitek_szama,0,
➥-1):
   bit=szam//2**(i-1)      #a
# kiírandó számjegyek balról
# jobbra haladva
   print bit,        #kiírjuk a
# kiszámolt értéket, majd
   szam=szam%2**(i-1)      #a
# maradékkal folytatjuk a
# ciklust
```

A programban van egy számunkra még ismeretlen vezérlési szerkezet, mely lehetővé teszi, hogy addig ismétlődjön a számjegybitek kiszámítása, amíg ki nem írtunk a `bitek_szama` változóban beállított mennyiségű számjegyet. Valójában nem is egy, hanem két egymásba ágyazott szerkezetről van szó, melyből a `for` több programnyelvben általánosan használt ciklusvezérlő. A ***Python*** kissé más logika szerint használja mint például a ***C*** nyelv, inkább egy karakterlánc (string) típusú felsorolás, lista esetén tudnánk hatékonyan kihasználni képességeit, mivel előre megadott értékek mentén képes ciklust szervezni. Például a következőképp tudunk egy lista elemein végigmenni, és kiíratni azokat:

```
listank=[1, 'kutya', 'macska',
➥13, 'pingvin']
for ciklusvaltozo in listank:
   print listank
```

Ha pedig csak arra lenne szükségünk, hogy egytől százig írja ki a számokat az értelmező, akkor legegyszerűbben talán így kérhetnénk tőle:

```
range(100)
```

Gyakorlatilag csak a felső korlátot adtuk meg, a kezdőértéket (1) és a lépésközt (1) a beépített függvény „kitalálta”. Természetesen megadhatunk más záróértéket és lépésközt, ezt tettük a példaprogramban is, azaz a ciklusmag számol visszafelé (lépésköz -1) egészen 0-ig, a megadott `bitek_szama` változótól kezdve, s az éppen aktuális értékét az `i` változóban tároljuk.
A kettes számrendszer fent részletezett logikájából következik, hogy a 8 biten ábrázolható legnagyobb szám $2^{(8-1)}$, ezért láthatjuk azt, hogy a ciklusban már csak $2^{(i-1)}$. hatvánnyal osztunk.
A `print bit,` utasítás után szereplő vessző szándékos, segítségével a számjegyek nem sorban egymás alá (különben a `print` automatikusan sortöréssel dolgozna, mintha `\n` sorvégjelet kapott volna), hanem szóközzel elválasztva egymás mellé kerülnek a jobb átláthatóság érdekében. Igazi megoldást (szóköz nélkül, egy számként látszódva ábrázolni) a `print` helyett más függvény használata jelentene, erre a későbbiekben kitérünk.

Túlcsordulunk

Mi történik, ha kevesebb biten próbáljuk meg ábrázolni a számot, mint amennyit az igényelne? Ha például a `bitek_szama` változó értékét 4-re állítjuk, a következő eredményt kapjuk: 21 0 0 1

Mi változott az eddig jól működő programban? Semmi, pusztán elfelejtettünk felkészülni a megszokott normálistól eltérő szituációkra. A program futtathatóságát, azaz nyelvtani helyességét illetően nincs aggályunk (szintaktikailag helyes), viszont így már a logikája rosszul működik (szemantikailag hibás). Tegyünk bele egy egyszerű ellenőrző rutint:

```
szam=169
bitek_szama=8        #azaz hány
➥biten ábrázoljuk a számot

for i in range(bitek_szama,0,
➥-1):
   bit=szam//2**(i-1)      #a
# kiírandó számjegyek balról
# jobbra haladva
   if bit<=1:
      print bit,
   else
      print "Szerintem nem lehet
➥ennyi biten ábrázolni!"
      break       #kilépünk
# a ciklusból és a program
# véget ér
   szam=szam%2**(i-1)      #a
# maradékkal folytatjuk
# a ciklust
```

Az újonnan elhelyezett feltételvizsgálatban `break` utasítás megszakítja a ciklus működését, és kilép a programból ha kettes számrendszerbe nem illő számjegy keletkezne, viszont a feltételes vezérlés (`if`) használata miatt ha elegendő a megadott bitszám, akkor az utána következő művelet végrehajtását ez egyáltalán nem befolyásolja.
Ezek után könnyedén fel tudjuk írni például a 25-öt kettes számrendszerben 8 bittel ábrázolva:
00011001

A << műveleti jellel balra toljuk egy bittel az egész számsort, így binárisan 00110010-t kapunk:

```
print('25 egy bittel balra
➥forgatva:' ,25<<1)
50
```

Ahol eddig nem szerepelt számjegy, oda 0-t írunk (jobb szélső érték), ill. az egész számsort balra csúsztattuk el. Mindez ugyanazt eredményezi, mintha kettővel szoroztunk volna, hiszen a kapott szám éppen 50. Ha nem egy, hanem két helyiértékkel csúsztattuk volna balra (`50<<2`), akkor az megfelel a néggyel való szorzásnak (100 az eredmény), `50<<3` ugyanaz, mintha nyolccal szoroznánk stb., azaz a jel jobb oldalán szereplő számot kettő n-edik hatványkitevőjeként használva $50*2^n$ értékeket számoltathatunk ki e példában.
Ennek nagyjából ellenkezője történik a jobbra csúsztatás során, ha kiadjuk:

```
50 >> 1
```

az eredmény 25 lesz, mivel a bitek jobbra csúszva 00001100 -t adnak kettes számrendszerben, s az eredetileg jobbszélső helyiértéken szereplő 1-es csonkolásra került, más néven túlcsordult. Programunk esetén is azt tapasztalhattuk, hogy a szám nagysága és az ábrázolandó bitek száma közötti

összefüggés miatt előfordulhat, hogy nem tudjuk a számot kettes számrendszerbe korrekten átszámítani. Még látványosabb ugyanez a művelet, ha nem eggyel, hanem négy helyiértékkel csúsztatjuk jobbra az 50-et:

```
print '50 >> 4 eredménye: ',
➥50>>4
50>>4 eredménye: 3
```

Ugyanazt az eredményt kaptuk, mintha maradék nélkül osztunk 2^4 -el, vagyis kiadtuk volna:

```
50//16
```

Megemlíthetjük még a bitszintű tagadást, jele a ~, értékét pedig a következő egyenlőség mutatja:

```
~szám = -1*(szám-1)
```

Mindez a negatív számok tártakarékosabb, úgynevezett kettes komplemens ábrázolásával függ össze, s ezért nem egyszerűen ellenkezőjére váltja a biteket.
Segítségül hívhatjuk a ***Pythont*** akkor is, ha vissza akarunk váltani egy számot kettes számrendszerből tízesbe, de akár nyolcasból tízesbe is, mégpedig a már ismert `int()` konverziós függvény segítségével:

```
a=int('0110',2)      #a második
# argumentum jelzi, hogy hányas
# számrendszerből
print 'a értéke szerintünk
➥tízes számrendszerben 6,
➥a Python szerint:' , a
b=int('644',8)    #nyolcas
# számrendszerből alakít át
print 'b értéke szerintünk
➥tízes számrendszerben 420,
➥a Python szerint:',b
```

Figyeljük meg, hogy az `int()` hívásakor egyszeres vagy kétszeres idézőjeleket használtunk, különben a konverzió nem sikerül, mivel egyszerű karakterláncot, és nem „igazi számot" vár tőlünk az értelmező.

És a jogaink? Vagy a jogaink?

Hol láthatjuk még hasznát a kettes számrendszernek? A bitszintű operátorok használatakor például áramkörök működésének megértésekor, vagy akár gyorsítás céljából alacsony szintű programozás esetén, háromdimenziós grafikánál, s más alkalmazásokban.
Bitszintű ***és*** művelet jele ***&***, értéke két 1-es számjegy találkozásakor 1, minden más esetben 0.
Kettes számrendszerben felírva a két számot:

egyik szám:	1001 (tízes számrendszerben értéke 9)
másik szám:	0101 (tízes számrendszerben értéke 5)
eredmény:	0001 (tízes számrendszerben értéke 1)

Bitszintű ***vagy*** művelet jele | , jelentése pedig az, hogy azonos helyiértéken két 0 számjegy találkozása ad 0-t, minden más esetben 1 fog szerepelni.

egyik szám:	1001 (tízes számrendszerben értéke 9)
másik szám:	0101 (tízes számrendszerben értéke 5)
eredmény:	1101 (tízes számrendszerben értéke 13)

Mire használhatjuk ezt a tudást a gyakorlatban? Nézzünk egy egyszerűnek tűnő példát, a fájljogosultságokat.
Ha egy adott könyvtárban kilistázzuk az állományokat, a következőhöz hasonló kimenetet kapunk:

```
[valaki@localhost]$ ls -l
total 24
-rwxr-xr-x 1 valaki valaki 462
➥Oct 17 10:21 pelda.py
-rwxr-----  1 valaki mas
➥80 Oct 10 23:02 teszt.txt
```

Az első oszlopblokk reprezentálja a jogosultságokat, ***Unix/Linux*** felhasználók számára feltehetően ismerős séma szerint, azaz az első érték – vagy d lehet (utóbbi jelenti a könyvtárat), ezt a továbbiakban figyelmen kívül hagyjuk; a következő három a felhasználó hozzáférési jogosultsága, majd három karakter jelzi a csoportét, végül három a többiekét.
Számunkra lényeges, hogy e három különálló csoportot viszonylag egységesen lehet kezelni, hiszen ha kiadjuk a `chmod - x pelda.py` parancsot, akkor mindhárom csoportnak egyszerre szüntetjük meg az addig meglévő futtatási jogát az adott fájlra. A ***chmod*** parancs azonban ennél finomabban is szabályozható (csoportra, egyénre), legegyszerűbben talán úgy, ha a jogokat számokkal reprezentáljuk. E számokat úgy kell elképzelnünk, mintha az ***rwx*** jelzők mindegyike egy-egy bitet jelentene, s a karakterhármas együttesen egy oktálisként (nyolcas számrendszer, gyakorlatilag mintha 3 egymás mellett álló bit képezne egy számot) mutatná a jogokat. Szemléletesebben az előző példánk a ***pelda.py*** fájlra, ahol 1 az adott bit értéke, ott van beállítva a jogosultság:

```
d r w x r w x r w x
0 1 1 1 1 0 1 1 0 1
```

Minden számhármas értelmezhető egy-egy háromjegyű bináris számként, melynek maximális értéke számhármasonként átszámítva 7 (111 binárisan), minimális értéke pedig 0 (000 binárisan), a köztes értékeket pedig aszerint veszik fel, hogy mely jogosultság van beállítva. Például a tulajdonosnak van futtatási, a csoportnak írási és olvasási joga, mindenki másnak csak írási joga, és nem könyvtárról van szó:

```
oktálisan: 0   1   6   2
binárisan: 000001110010
leírva:    d   rwx   rwx   rwx
```

Tételezzük fel, hogy egy fájlunk hozzáférési jogosultságát kell beállítanunk úgy, hogy a tulajdonos számára írható legyen, de minden más attribútuma változatlan maradjon. Ezt úgynevezett maszkolással lehet könnyen megoldani (lásd ***umask*** működése), ahol egy olyan bitmaszkkal végzünk ***vagy*** műveletet, mely minden más bitje 0 értékű (így a ***vagy*** művelet működéséből következően nem változtatja meg az eredeti értékeket) a beállítani kívánt pedig 1.

```
#!/usr/bin/python
import os,stat,sys

fajlnev=sys.argv[1] #parancs-
# sori indítás paramétere
informacio=os.stat(fajlnev)[ST_
➥MODE]  #modulból stat futtatás
# jogosultságbitekre
print "Jogosultságok oktális
➥számmal:' , oct(jogok)   #ki-
# íratjuk oktális számokkal
```

Az `import` utasítással betöltjük a használni kívánt – a nyelv alapkészletét bővítő – modulokat, melyekkel a viszonylagos hordozhatóságot (`os`) és az operációs rendszer parancsaihoz hozzáférést (`stat, sys`) tudjuk biztosítani. Működésük pontosabb megértését segítheti a dokumentáció böngészése, de hatékony használatukhoz szükségünk lesz a később ismertetendő objektum-orientált tervezési elvek ismeretére is, egyelőre csak vegyük igénybe őket. Az `ST_MODE` segítségével a jogosultságbiteket és egyéb jellemzőket (***sticky bit, fájltípus*** stb.) tudjuk akár makrószerűen egyesével lekérdezni. Minket most csak a három csoport jogosultságai érdekelnek, ezért egyszerűsítjük a kiíratást:

```
eredeti_oktalis_jogok=oct(jogok
➥& 0777)
print "Könyvtár,tulajdonos,
➥csoport,egyéb oktálisan:' ,
➥eredeti_oktalis_jogok
```

Láthatjuk, hogy a megjelenített érték egyezik az előző, szűretlen kiíratás utolsó három számjegyével (a számkezdő 0 csak azt jelzi, hogy oktális szám, és nem tízes számrendszerbeli), hiszen ***bináris és*** műveletet hajtottunk végre egy csupa 1-esekből álló (777 oktálisan) számmaszkkal, ezáltal az eredeti bitek értékét nem változtattuk. Pontosabban csak a jobb oldali három bit értékét, az első négy bitet ugyanis pont ezzel – a bevezető nullákat is oda kell gondolni – nulláztuk le.

A `pelda.py` fájl `1 1 1 1 0 1 1 0 1` jogosultságbitjeit (lásd feljebb) oktálisra váltva 755-öt kapunk, ezt kapcsoljuk össze ***és*** műveleten keresztül a 777-es maszkkal, melynek utolsó három számjegye binárisan 111111111. Egyszerűsítve:

eredeti pelda.py fájl oktálisan: 0100755
maszk oktálisan: 0000777
---
eredmény oktálisan: 0000755

Ez a maszk egyelőre csak annyit csinált, hogy „levágta” a fölösleget, a számunkra jelenleg érdektelen bevezető bitekkel így nem kell foglalkoznunk. Ejtsünk néhány szót a `fajlnev=sys.argv[1]` sorról is. Eddigi példáinkban a változókat jobbára előre feltöltöttük értékkel, rugalmatlan programozási megoldást választva. Ha annak eldöntését, mely fájl jogosultságait írassuk ki s módosítsuk majd, a felhasználóra szeretnénk bízni, legegyszerűbb, ha parancssori argumentumként kérjük be. A gyakorlatban ez annyit tesz, hogy

pythonprogramnév argumentum1 argumentum2 argumentum3 … argumentumn

alakban kell elindítani a programot, s mi a `sys` modul segítségével ezen argumentumok értékeit egy számozott listából ki tudjuk olvasni.
A legelső argumentum maga a futtatott fájl – függetlenül attól, hogy megadjuk-e külön, erre `argv[0]`-ként lehet hivatkozni, míg a többi sorban a következő sorszámú elem, azaz az első „igazi” parancssori argumentum az `argv[1]` lesz.
Esetünkben így lehet a szkriptet a ***pelda.py*** fájlra értelmezve futtatni:

```
pythonprogramneve pelda.py
```

Folytassuk rövid programunkat a lekérdezés után a jogosultságbeállítással. A használandó maszknak csakis a tulajdonos írási bitjét szabad 1-re változtatnia (ez jobbról a 8. bit), az összes többi bitet változatlanul kell hagynia, akármilyen értékük is volt eredetileg. Erre alkalmas a ***vagy*** művelet és egy ilyen bináris bitmaszk: 0 1 0 0 0 0 0 0 0, ami oktálisan kifejezve 200.

```
maszk=oct(0200)    #tulaj_rwx=2
# csoport_rwx=0 mások_rwx=0
beallitott=int(eredeti_oktalis_
➥jogok,8)   | int(maszk,8)
# tízes számrendszerbe váltva
# VAGY
os.chmod(fajlnev,oct(beallitott
➥& 0777)    #a művelet eredménye
# szerint állítjuk be
```

Amint látjuk, a biztonság kedvéért átváltottuk egyazon számrendszerbe a műveletek előtt az oktálisokat, majd átadtuk a kapott értéket az operációs rendszer használó modulnak.
A ***chmod*** két paramétere értelemszerűen a fájlnévből és az új jogosultságból áll, ami „tartalmazza” a korábbit, és gyakorlatilag egyenértékű a `chmod u+w` parancs kiadásával.
Az előző ***vagy*** műveletet ***megengedő vagy***ként is emlegetik, ellentétben a ^ jellel jelölt ***kizáró vagy***-gyal, mely a számjegyek találkozásakor csak akkor ad 1-est eredményül, ha két különbözőről van szó, azaz 0 és 1 vagy 1 és 0 ugyanazon a pozíción:

A	B	A ÉS (&) B	A VAGY(|) B	A KIZÁRÓVAGY(^) B
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

1. ábra Igazságtábla

egyik szám: 1001 (tízes számrendszerben értéke 9)
másik szám: 0101 (tízes számrendszerben értéke 5)
eredmény: 1100 (tízes számrendszerben értéke 12)

Végezetül használjuk fel ezen tudásunkat egy egyszerű titkosítóprogram írására, melyhez számokat és a ***kizáró vagy*** műveletet hívjuk segítségül.

```
adat=17
kulcs=35
titkositva=adat^kulcs
print"Az adat titkosítva így
➥néz ki:' , titkositva  #ez
# azért nem túl biztonságos
# titkosítás...
print "Visszafejtése:
➥titkosított adat kizáró_vagy_
➥művelet kulcs, azaz:"
print titkositva^kulcs
```

Alacsony szintű, bináris műveletekkel ritkábban találkozunk közvetlenül, mint például a későbbiekben ismertetett logikai és,vagy vagy egyéb kifejezésekkel. Nem árt tudnunk azonban, hogy egy magas szintű utasítás végrehajtása mögött milyen elemi lépések lehetnek, illetve bizonyos helyzetekben kifejezetten jól jön ez a tudás sebességnövelésre, egyszerűsítésre. A ***Python*** szerencsére egyre segítőkészebb ahogy haladunk vele előre, s a magasabb szintű funkciók esetében könnyebb dolgunk lesz.

Tóth Virgil Zoltán
(m_v@c2.hu)

Kommunikáció „fű alatt” – AJAX (1. rész)

Melyik webfejlesztő nem ábrándozott még arról, hogy anélkül kommunikáljon a szerverrel, hogy az oldalt újra kellene hívni? Az AJAX , mint a JavaScript és az XML nyerő párosa váltja valóra ezt az ábrándot. Ismerjük meg közelebbről ezt a megoldást, melynek segítségével talán hamarosan csökkenni fog a honlap és alkalmazás kifejezések közötti különbség.

Egyszer volt , hol nem volt …

Hogy igazán megértsük, miért is fontos és nagy áttörés az ***AJAX***, valamint , hogy mi is ez valójában, érdemes áttekinteni a ***World Wide Web (WWW)*** történetét napjainkig. Az első név, melyet meg kell említenünk ***Tim Berners-Lee***, aki lefektette a ***WWW*** alapjait a 1990-ben azzal, hogy megálmodott egy hálózatot. Egy olyan hálózatot, melyben az adatok rendszerezetten ***hypertext*** formátumban érhetők el, úgynevezett ***URI-k (Uniform Resources Identifiers)*** segítségével. Ekkor bontakozott ki a jó öreg ***HTML (Hypertext Markup Language)***, melynek segítségével formázni lehetett az adatokat, valamint linkek alkalmazásával összekapcsolni az egyes dokumentumokat. Ezeken felül nem sok tudása volt a ***HTML***-nek, az adatok statikus megjelenítése volt a jellemző.

Az üzlet azonban hamarosan közbeszólt. A reklámok, hirdetések, egyéb üzleti elképzelések vezettek oda, hogy fellépet az igény arra, hogy a felhasználók szabadabban kezelhessék a megjelenített adatokat. A ***HTML*** azonban nem támogatott eddig semmilyen eszközt, mellyel a megjelenített adatok manipulációját eszközölni lehetett volna. Ekkor a semmiből bukkant elő egy vállalat, aki igen csak nagyot lendített a ***WWW*** fejlődésén, megoldást találva a fent leírt problémára és melyet úgy hívtak, hogy ***Netscape***.

Az első igazi böngésző, mely igazán elterjedt a ***Netscape Navigator*** volt. A ***Netscape*** ebben a böngészőjében tette lehetővé először a ***JavaScript*** használatát, melynek eredeti neve ***LiveScript*** volt. Fejlesztőjét ***Brendan Eichnek*** hívták, az első verzió pedig az 1995-ben kiadott ***Netscape Navigator 2.0***-ban látott napvilágot. Ezekben az időkben ha valaki 28.8 Kbps-os modemmel rendelkezett az nagy jelentőséggel bírt. A lényeg, hogy nem kellett minden interakció után a szerver felé kérést intézni, hanem a ***JavaScript*** segítségével helyben is lehetett módosítani az adatokat, vagy ugyancsak a szkripten keresztül csak bizonyos adatokat fogadni illetve küldeni, melynek segítségével csökkent az adatforgalom és nőtt a kiszolgálás sebessége. Ez kulcsfontosságú volt az ***AJAX*** történetében.

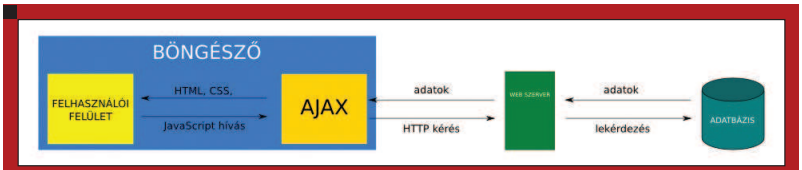
A ***HTML*** „ősi” verziója csak „egyke” dokumentumokat tudott megjeleníteni. A ***frame-ek*** bevezetése, ugyancsak nagy áttörés volt. Segítségükkel egy dokumentumot, több független részre lehetett osztani, így megint csak külön lehetett választani a statikus elemeket, és változó, dinamikus adatok másik ***frame***-be kerülhettek. Ez azért jelentős, mert nem az egész dokumentummal kellett dolgozni, csak egy bizonyos részével, megint csak csökkentve a feldolgozott adatok mennyiségét és a feldolgozás idejét. A ***Netscape*** megelőzve a ***HTML 4.0*** szabvány bevezetését, saját kezébe vette az irányítást, és a már említett 2.0-ás verziójú böngészőjét frame-es támogatással dobta piacra. A ***JavaScript*** és a ***frame***-ek használata többféle kliens-szerver kommunikációra adott lehetőséget, főleg azután, hogy a 90-es évek végén a „böngésző háborúban” (***Netscape Navigator – Internet Explorer***) a ***frame*** és ***JavaScript*** lehetőségek a versenyhelyzetben egyre jobban csiszolódtak, fejlődtek.

Az ***AJAX***-nak még be kellett várnia az ***XML*** forradalmát is, hogy végre úttörő lehessen a kliens-szerver kommunikációban a böngészők és a webfejlesztés terén. Végül 2005-ben látott napvilágot a kifejezés: ***AJAX***, mely nem egy új nyelv, mint majd látni fogjuk, hanem több alkalmazás együttesét takarja.

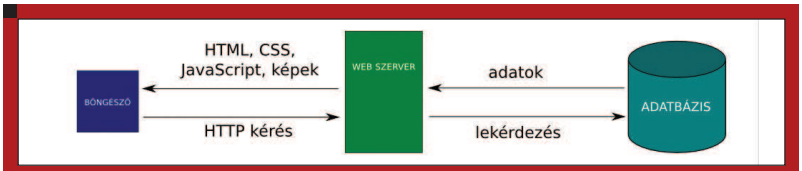
Az AJAX előtti technikák

Még mindig húzzuk az időt egy kicsit, olyan technikák megismerésével amelyekkel megvalósítható az oldal újratöltése nélküli kommunikáció a szerverrel. ***Frame*** és ***JavaScript*** alkalmazásával könnyen tudunk a háttérben kommunikálni a szerverrel. Ezt nevezzük „rejtett frame” módszernek. Az alapelv a következő: az oldalunk legalább két ***frame***-ből áll, melyek közül az egyik rejtett és ezen a ***frame***-en keresztül kommunikálunk a szerverrel. A rejtett ***frame*** annyit takar, hogy a szélessége vagy magassága 0 pixel. Az adatáramlást a két frame között a ***JavaScript*** fogja biztosítani.

A rejtett ***frame*** tartalmaz egy ***formot***, amit a látható ***frame***-ből származó adatok valamint a ***JavaScript*** segítségével kitöltünk és elküldünk. Amikor az információ megérkezik a rejtett ***frame***-be akkor azt kimásoljuk a látható területre.



1. ábra Az AJAX



2. ábra Általában így kommunikálnak a honlapok a szerverrel

A következő változást a ***DHTML (Dynamic HTML)*** és a ***DOM (Document Object Model)*** megjelenése hozta. Ezen két technológia segítségével lehetővé vált, hogy ***JavaScript*** programozással a kliens oldalon lehessen módosítani a ***HTML*** dokumentum szerkezetét, bármely apró részének tulajdonságát, tartalmát. Ez megint egy lépés volt az ***AJAX*** fejlődésében.
Bár a *„**rejtett frame**”* módszer nagyon elterjedt, hamarosan mégis egy új közkedvelt megoldás érkezett.
Az ***iframe*** elem 1997-ben futott be a ***HTML 4.0*** részeként. Használata sokkal egyszerűbb volt, mint a ***frame***-eké. Az oldalban bárhol el lehetett helyezni ezt az elemet, és ***CSS*** segítségével ***(Cascading Style Sheets***) láthatatlanná tenni. Amikor pedig az ***Explorer 5***-ös és a ***Netscape 6***-os böngészők hozták a ***DOM***-ot, a *„**rejtett frame**”* módszernek végleg befellegzett.
A ***DOM*** segítségével a fejlesztők ***JavaScripttel*** bármikor be tudtak szúrni a dokumentumba egy ***iframe*** elemet a kliens oldalon, anélkül hogy eleve a dokumentumban létezett volna ilyen elem egyáltalán. Azaz a szerverrel való kommunikáció végleg a ***JavaScript***-re maradt, nem kellett már kombinálni, előre megírt ***HTML*** elemekkel, hiszen azokat is létre tudja már hozni futási időben.
Az utolsó fejlemény, ami talán végleg megpecsételte az ***AJAX*** sorsát – hiszen valamilyen szempontból innen is ered maga a kifejezés: ***AJAX*** – az a ***Microsoft XMLHttp ActiveX*** objektumának megjelenése volt 2001 környékén. Ezen az objektumon keresztül lehetővé vált, hogy kérést küldjön a ***JavaScript*** a szerver felé, valamint hogy kezelje a ***HTML*** státusz kódokat, fejléceket is. Lehetővé vált végre, hogy ***frame***-ek illetve ***iframe***-ek használata nélkül valósuljon meg a kommunikáció. Persze a ***Mozilla*** projekt később sajátot fejlesztett ki ebből az objektumból, ami hordozza az ***XMLHttp*** főbb jellemzőit: ***objectXMLHttpRequest***. Még fogunk vele találkozni a cikk során.

Végre: AJAX

Ennyi történet után már remélhetőleg valami körvonalazódik az olvasó fejében, hogy mi is lehet az az ***AJAX***.
Az ***AJAX*** tehát annak a technológiának a neve, mely a ***háttérben*** valósítja meg a kommunikációt a szerverrel és lehetővé teszi a kapott információk által a felhasználói felület módosítását. Lényeges eleme, hogy a háttérben várja meg az adatok megérkezését, tehát nem áll meg az oldal működése, amíg a szerver felé a kérés le nem zárul. Az ***adatáramlásban bármilyen adat szerepelhet***: sima szöveg, ***XML*** vagy amire éppen a fejlesztőnek szüksége lehet. Végeredményben egy ***kommunikációs rétegről*** beszélhetünk: *1. ábra*. Az általában megszokott kommunikációs forma: *2. ábra.*
Ennek a megoldásnak nagyon sok előnye van. Mivel csak a szükséges adatokat mozgatja a szerver és a kliens között ezért minimalizálódik az áramlott adatok mennyisége, nincs felesleges kommunikáció. Felesleges lehet például mindig újratölteni egy oldalon a ***CSS***-t, képeket, ***HTML*** elemeket, ha csak a tartalom változik.
A kevesebb adatforgalom gyorsabb kommunikációt eredményez, ezáltal gyorsabb a felhasználók igényeinek kiszolgálása is. Mivel nincs szükség az oldal újratöltésére, szinte egy alkalmazás látszatát kelthetjük a felhasználóban, miközben egy ***HTML*** dokumentumot kezel.
Természetesen hátrányai is vannak ennek a módszernek. A programozás során elég bonyolult kód keletkezhet, ha elég jól értünk a ***JavaScripthez*** akkor nem lehet akadály az ***AJAX*** programozás. Persze az ***AJAX*** technológia nem csak kizárólag ***XML***-re és ***JavaScriptre*** épít, ahhoz, hogy megfelelően kezelhessük az ***AJAX***-ot ismernünk kell az ***XHTML***, ***HTML*** nyelveket a ***CSS***-t, ***DOM*** ismeretekre fokozottan szükségünk van, nem árt ha képben vagyunk az ***XML*** alapjaival, az ***XMLHttp*** és az ***objectXMLHttpRequest*** kezelése is szükséges a ***JavaScripten*** felül.
Már ezekből is látszik, hogy egy bonyolult rendszer keletkezhet egyszerűbb ***AJAX*** oldalból is, de majd látni fogjuk, hogy az eredmény kárpótol mindenért, és később már nem is fog olyan nehéznek tűnni.

AJAX akcióban

Lássunk végre példát a módszer alkalmazására. Mi is lehetne más a példa, mint a jó öreg *„**Hello World**”* alkalmazás. Kicsit persze alakítunk rajta, hogy két irányú adatáramlásról legyen szó. Készítünk egy egyszerű ***HTML*** oldalt, melyben szerepelni fog egy input mező valamint egy gomb. A gomb megnyomásakor az input mező tartalma elküldjük a szervernek ott feldolgozzuk egy ***PHP*** szkript segítségével és visszaadjuk a böngészőnek úgy hogy elé fűzzük a *„**Hello**”* szót, ezt pedig egy mondjuk egy ***SPAN***-ban fogjuk megjeleníteni. Az alkalmazás három részből fog állni: a ***HTML*** felület, ***JavaScript***, szerver oldali szkript (ez most ***PHP*** lesz).
Először készítsük el a ***HTML*** felületet, ami mivel most szemléltetési eszköz lesz, igen egyszerűen fog kinézni: *1. lista.* A gombnak és a beviteli mezőnek itt szerintem érdemesebb ***id***-t adni mint nevet (***name***). Véleményem szerint ez alapján könnyebb őket visszakeresni a ***DOM*** segítségével (`document.getElementById` függvény). A gomb ***onclick*** metódusa a ***Kuld*** eljárás lesz.

1. lista HTML felület

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C
➥//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/
➥xhtml1/DTD/xhtml1-
➥strict.dtd">
<html xmlns="http://
➥www.w3.org/1999/xhtml">
  <head>
    <title>AJAX -
    ➥LINUXVILÁG</title>
    <script language=
    ➥"javascript"
type="text/javascript">

        // ide jön majd a
➥JavaScript kódunk: 2. lista

    </script>
  </head>
  <body>
     Név:<input type="text"
     ➥id="nev" />
    <input type="button"
    ➥onclick="Kuld()"
    ➥value="Ok" />
    <br />
    Ide fog kerülni a
    ➥szerverről visszaérkező
    ➥adat: <span id="box">
    ➥</span>
   </body>
</html>
```

A kommunikációs réteg programozása *JavaScript* segítségével: 2. *lista*. Ahogyan nagyon sok *JavaScript* alkalmazás működik, itt is figyelnünk kell, hogy melyik alternatívát használó böngészővel van dolgunk, és ez alapján hozzuk létre az `XMLHttpRequest` vagy az `XMLHttp` objektumunkat. Ezen az objektumon keresztül fog történni a kommunikáció. A `Kuld` eljárás végzi a kommunikációs feladatokat. Kivesszük a beviteli mezőből a beírt nevet, értéket. Az `Ajax.onreadystatechange` egy olyan függvény vagy eljárás, mely akkor fut le ha az adatok megérkeztek, ez lesz nekünk most a `Kesz` eljárás. Az `open` tag fogja nyitni a csatornát a szerver felé. Most *GET* módszerrel fogunk küldeni adatot a *hello.php*-nak (a küldendő anyagot a *URL*-hez fűzzük), a harmadik paraméterben pedig megadjuk hogy a futás ne szakadjon meg amíg az adatok érkeznek (`false` érték esetén pedig a futás a kommunikáció idejére megszakad). A `send` tag paramétere csak *POST* típusú küldés esetén érdekes, ezért ezt most null értékre állítjuk. A `Kesz` függvényben a feltétel vizsgálja a kommunikáció eredményességét. A `readyState` több értéket is felvehet, amivel az adatforgalom feletti ellenőrzést lehet megvalósítani:

0 – még nem indult el az adatcsere,
1 – töltés,
2 – töltés kész,
3 – feldolgozás,
4 – kész.

A feltételünk persze az *ActiveX* objektummal is számol, ezért vizsgálja a `complete` egyezést is. Amint megjött az adat a szerverről,amit a `responseText` tag tartalmaz, elhelyezzük a *SPAN* tagon belül. A szerver oldalon pedig annyira egyszerű ebben az esetben a kód, hogy külön listát kár lenne rá pazarolni. A *hello.php* tartalma a szerveren: `<?php echo 'Hello '.$_GET['nev'].'!'; ?>`. Láthatjuk, hogy egy egyszerűbb példa is kicsit bonyolultra sikerülhet, de az eredmény véleményem szerint nagyszerűen hat. Nem kell az oldalt újratölteni, csak a lényeges elemek változnak. Kisebb adatok esetén, szinte egy helyben futó alkalmazás érzetét kelti a honlap. A jó öreg *Google* a *Gmail* nevű szolgáltatásánál is előszeretettel használja ezt a technológiát. Még külön *API*-t is le tudunk tölteni a honlapjukról (➲ *code.google.com*), ami az *AJAX*-ra épít és felhasználhatjuk honlapok fejlesztéséhez.

Ami következik

A következő alkalommal, természetesen *AJAX*-ot használva felépítünk egy honlapot, hogy komolyabb alkalmazásra is lássunk példát. Addig is ha valakit elkapott az ihlet, akkor kellemes időtöltés kívánok az *AJAX*-hoz. Véleményem szerint hamarosan eljön az idő, amikor alkalmazás és honlap egybe fog olvadni. Ez a technológia az első lépések egyike. A *Google* már ilyen úton jár a *Docs & Spreadsheets* nevű szolgáltatásával is.

2. lista AJAX JavaScriptben

```
      var Ajax = null;

        // Mozilla
        if (window.XMLHttp
        ➥Request) { Ajax =
        ➥new XMLHttpRequest
        ➥(); }

        // Explorer
        if
(window.ActiveXObject)
        {
           Ajax = new
➥window.ActiveXObject("Micro
➥soft.XMLHTTP");
        }

        function Kuld()
        {
          var nev    =
➥document.getElementById
➥('nev').value;

          Ajax.onready
➥statechange = Kesz;
          Ajax.open('GET',
➥'hello.php?nev='+nev,true);
          Ajax.send(null);
        }

        function Kesz()
        {
          if (Ajax.ready
➥State == 4 || Ajax.ready
➥State == "complete")
          {
            document.get
➥ElementById('box').inner
➥HTML = Ajax.responseText;
          }
        }
```

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

A legjobb biztonsági rendszer... ugat!

Hála az égnek, a Guarddog nagyobbat harap, mint amilyen hangosan ugat.

Nem, ***François***, nem tűnt fel, hogy valami baj lenne az internet-kapcsolattal. Ja, értem, szóval új tűzfalat telepítettél, ami most nem enged ki. Hm, lássuk csak a beállításokat. Azt hiszem, tudom, mi a gond, ***mon ami***. A beállítások rendkívül szigorúak, viszont kiváló biztonságot nyújtanak. Semmi nem juthat be, de semmi sem jut ki. Tökéletesen biztonságos.

Igen, ***François***, csak vicceltem.

Ha a biztonságról van szó, mindenki túlzó hasonlatokat használ. Egyszer azt hallottam, hogy egy kiszolgálót úgy lehet a legbiztonságosabbá tenni, ha kihúzzuk, és a szekrénybe zárjuk. Ha már viccelődünk, miért ne menjünk tovább? Öntsük betonba a kiszolgálót, és temessük egy 15 méter mélyen fekvő, ólommal bélelt pincébe. A tréfát félretéve, ***mon ami***, meg kell találni a helyes egyensúlyt az elfogadható biztonság és a teljességgel használhatatlan rendszer között. Ez szerepel a mai étlapon, és amikor a vendégeink végre megérkeznek, felszolgálunk néhány tetszetős tűzfal-alkalmazást.

De már itt is vannak, ***François***. Mindenkit üdvözlünk ***Marcelnál***, ahol a remek ***Linux*** fogások és a kitűnő borok tökéletes harmóniát alkotnak. Hűséges pincérem az asztalhoz kíséri önöket, utána pedig behozza a bort. A 2002-es ***Sonoma Belle Glos Pinot Noir*** remekül hangzik – azt hiszem, ***François***, az északi szárnyban találod meg, ***Henri*** éppen ott tölti fel a készletet.

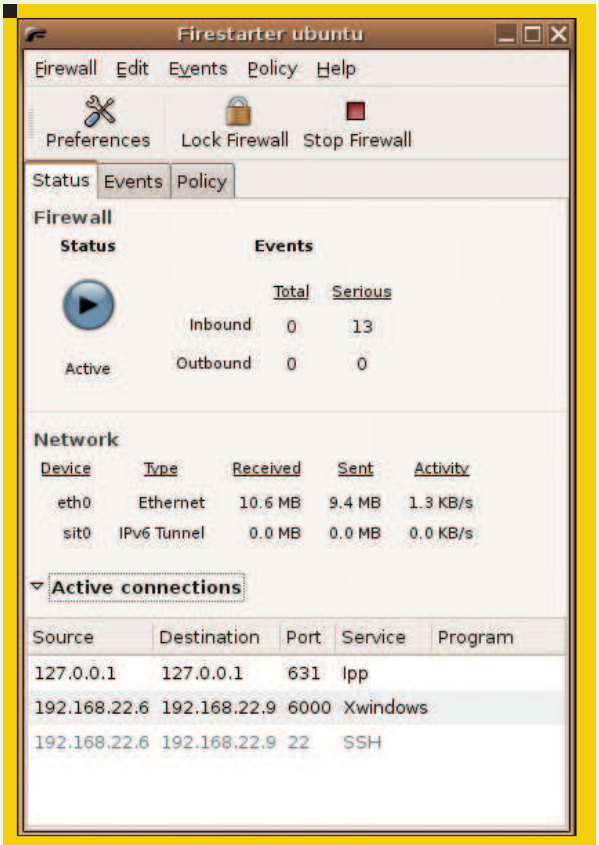
A ***Linuxot*** forgalmazó csoportok általában biztosítanak valamilyen tűzfalat a terjesztéshez, de nem mindegyik. Ezeket rendes estben a gyártó által rendelkezésre bocsátott rendszervezérlő eszközön keresztül érhetjük el. Időnként a tűzfal eszközök lényegében a parancssori ***iptables*** programok. Semmi gond nincs azzal, ha kizárólag a parancssor alkalmazásával húzunk fel tűzfalat, de sokan vannak, akik nagyon szívesen fogadnak egy kis irányított, egyszerűsített, grafikus segítséget. A ma tárgyalt tűzfalépítők az egyszerű használhatóság és beállíthatóság mellett rendelkeznek még egy előnnyel: mindkettő lehetővé teszi a tűzfal valósidejű módosítását. Mindegyik nagyon szigorú beállításokat alkalmaz a bejövő forgalomra (minden adatcsere tiltott, hacsak nem engedélyezik kifejezetten), továbbá nem terjesztésfüggők.

Ha úgy döntünk, hogy terjesztést váltunk, használhatjuk ugyanazokat az eszközöket.

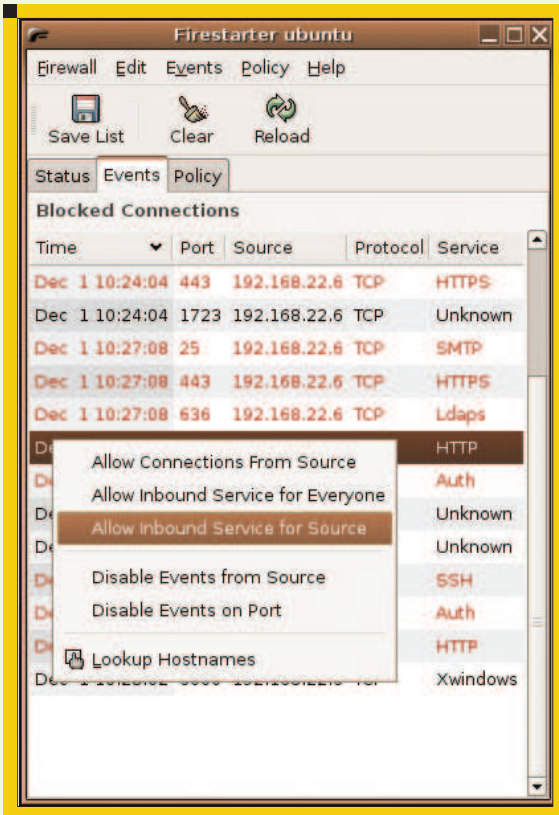
François, jó, hogy visszatértél. Kérlek, tölts a vendégeinknek!

A mai első fogás ***Thomas Junnon*** kitűnő ***Firestarter*** alkalmazása, egy könnyen használható, grafikus tűzfalprogram, amely lehetővé teszi a biztonsági szabályok valósidejű válaszát és beállítását. A ***Firestarter*** futtatásakor (parancsnév, ***firestarter***), meg kell adnunk a fő jelszót. Az első futtatáskor a program elindít egy tűzfalvarázslót (***Firewall Wizard***), amely segít a kezdeteknél. Mivel az első képernyő gyakorlatilag csak egy üdvözlés, olvassuk el az üzenetet, majd a ***Forward (Tovább)*** gombra kattintva lépjünk tovább.

Ezután az internet-kapcsolat megosztását kezelő képernyőre jutunk. Az egyfelhasználós gépeknél nem kell emiatt aggódnunk, egyszerűen továbbléphetünk a következő képernyőre. Ha azonban az otthoni vagy az irodai ***PC*** más ***PC***-k ***NAT*** (címfordító) átjárójaként szolgál, kattintsunk az ***Enable Internet connection sharing (Internet-kapcsolat megosztásának engedélyezése)*** jelölőnégyzetre. Ismét megadhatjuk, hogy alapértelmezésben milyen ***Ethernet*** kártyát (vagy telefonos kapcsolatot) használunk az internethez való csatlakozáskor. Közvetlenül ezalatt ***DHCP*** kiszolgálón keresztül adhatunk meg címeket. Ha a rendszeren nem telepítettünk ***DHCP*** kiszolgálócsomagot, ez a lehetőség szürkén jelenik meg.



1. ábra A Firestarter felülete világos és könnyen kezelhető



2. ábra A Firestarter Events lapján folyamatosan lehet létrehozni a forgalmat engedélyező szabályokat

A beállítások elvégzése után kattintsunk a ***Forward*** gombra, és tulajdonképpen végeztünk is a varázslóval. Mielőtt a ***Save (Mentés)*** és a ***Quit (Kilépés)*** gombra kattintanánk, nézzük meg figyelmesen a képernyőt. Észrevehetjük a ***Start firewall now (Tűzfal azonnali indítása)*** gombot, amely eleve ki van pipálva. A varázsló által létrehozott alapértelmezés szerint a ***Firestarter*** szabályai meglehetősen korlátozzák a bejövő forgalmat (ahogy ezt elvárnánk), és ez általában nem jelent problémát. Ahogy azonban a képernyőn megjelenő tipp is tájékoztat, ha távolról végezzük a telepítést, ez gondot okozhat.

Ha nem a szóban forgó munkaállomásnál ülünk, kapcsoljuk ki az azonnali indítás lehetőségét. Készen is vagyunk. Kattintsunk a ***Save*** gombra, és a ***Firestarter*** élesíti az új tűzfalat, valamint megjeleníti az állapotablakot (*1. ábra*).

A kezelőfelület egyszerű, három lapból áll. A lapok címkéje ***Status (Állapot)***, ***Events (Események)*** és ***Policy (Szabályok)***. Az állapotnézet az, amelyik nagy valószínűséggel rendszeresen érdekelhet bennünket. A kijelző megjeleníti a tűzfal futásának állapotát (***Active – Aktív vagy Disabled – Kikapcsolt***), a bejövő és kimenő kapcsolatokat, valamint a különböző felületeken keresztül zajló forgalmat. Az ablak alján található ***Active connections (Élő kapcsolatok)*** rész alaphelyzetben be van zárva. A címke melletti nyílra kattintva megtekinthetjük ezeket a kapcsolatokat.

Ki kell emelni, hogy a ***Firestarter*** először minden elképzelhető bejövő szolgáltatást kizár. Ennek következményeként, ha nem az asztali számítógépen, hanem a kiszolgálón futtatjuk, azt tapasztaljuk, hogy senki semmit nem tud majd futtatni, még a ***biztonságosnak*** gondolt szolgáltatásokat sem, például a webkiszolgálót. Ennek ellenére minden kimenő forgalom megengedett, tehát ez nem érinti a hagyományos asztali szolgáltatásokat, vagyis az e-mail olvasást, a webes böngészést és az ***IM*** (azonnali üzenetküldő) ügyfelekkel történő csevegést. A fent említett ***Active connections*** ablak azonnal megjeleníti a kapcsolódási kísérleteket, de ezek pár másodperc múlva elhalványodnak és eltűnnek.

Az ***Events*** lapra kattintva megtudhatjuk, milyen kapcsolódási események történtek, így eldönthetjük, hogy melyiket engedjük be. Itt megtaláljuk a számítógép teljes forgalmának naplóját (*2. ábra*).

Ha a jobb gombbal valamelyik bejegyzésre kattintunk, megjelenik egy felugró menü a kapcsolatokra vonatkozó lehetőségekkel. Amennyiben például a 80-as kapu (***HTTP*** szolgáltatás) egy eseményéről van szó, érdemes bejelölni az ***Allow inbound Service for Everyone (Belső szolgáltatás engedélyezése mindenki számára)*** lehetőséget. Más a helyzet viszont a 22-es kapun érkező biztonságos héjszolgáltatás (***secure shell***) kapcsolattal, amelynél csak az ***Allow inbound Service for Source (Beérkező szolgáltatás engedélyezése a forrás számára)*** jelölőnégyzetet pipáljuk ki. Egy adott ***IP*** cím (például egy belső hálózaton található

PC) engedélyezéséhez válasszuk az ***Allow Connections From Source (Forrástól érkező kapcsolatok engedélyezése)*** lehetőséget. Azt is megtehetjük, hogy egy adott gazdagép, kapuszám vagy szolgáltatás kapcsolatait nem naplózzuk.

Én személy szerint nem hiszem, hogy a rendszergazdák először megnézik, ki kopog, és csak azután engednek bizonyos szolgáltatásokat a rendszerb. Aki webkiszolgálót üzemeltet, annak valószínűleg előnyösebb, ha engedélyezi a 80-as kaput. Ugyanez az elv érvényes akkor, ha egy ***Samba*** kiszolgálón engedélyeznünk kell, hogy az irodában dolgozók hozzáférjenek a kiszolgálón tárolt megosztásokhoz. A ***Policy*** lapra kattintva elkerülhetjük, hogy a bekövetkezésük pillanatában kelljen kezelni az eseményeket. Ez az ablak két vízszintes részre vagy táblára van osztva.

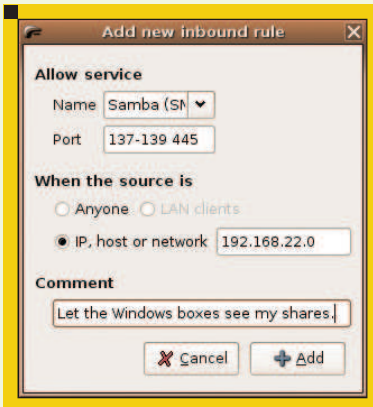
A felső egy adott gazdagép vagy gazdagép-csoport általános kapcsolatait kezeli, az alsó tábla pedig az egyes szolgáltatásokkal illetve azokkal a kapukkal foglalkozik, amelyeken ezek a szolgáltatások futnak. Az ***Events*** lapon hozzáadott szolgáltatások itt jelennek meg. Ha úgy akarunk további szabályokat hozzáadni, hogy ne kelljen végiglépkedni az ***Events*** párbeszédablakon, kattintsunk a jobb gombbal a felső vagy az alsó táblára, majd a felugró menüben válasszuk az ***Add rule (Szabály hozzáadása)*** lehetőséget. Megjelenik egy kis barátságos párbeszédablak, amely megkönnyíti a folyamatot (*3. ábra*).

Adjunk hozzá egy olyan szabályt, ami engedélyezi a helyi hálózatra csatlakozó ***PC*** számára, hogy elérje a ***Samba*** szolgáltatást. A párbeszédablak tetején szerepel egy lenyíló lista, amely a lehetséges szolgáltatásokat tartalmazza (például ***DHCP***, ***BitTorrent***, ***IMAP*** és a többi). Én a ***Sambát (SMB)*** választom a listából. Észrevehetjük, hogy az ismert szolgáltatásoknál a kapu (vagy kapuk) mezője automatikusan kitöltődik. Ezután a ***When the source is (Amikor a forrás...)*** címke alatti választógombbal engedélyezhetjük az összes kapcsolatot, illetve egy meghatározott gazdagépet vagy hálózatot. Ebben az esetben a saját ***C*** osztályú hálózatomat állítom be. Végül, a lent szereplő mezőben valamilyen megjegyzést is hozzáadhatunk. Kattintsunk az ***Add (Hozzáadás)*** gombra, és készen is vagyunk. Az új szabály megjelenik a ***Policy*** ablakban. Kattintsunk a ***Firestarter*** fő ablakának tetején található ***Apply Policy (Szabály alkalmazása)*** gombra az új szabály érvényesítéséhez.

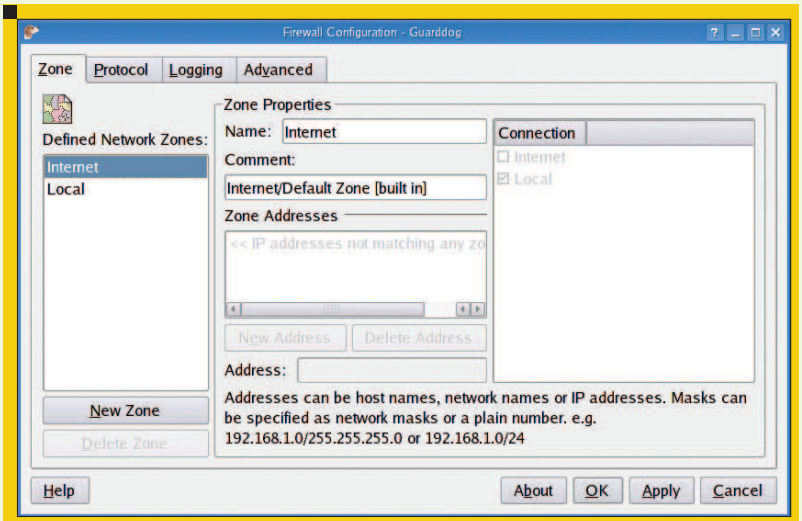
A most létrehozott szabályokhoz mellesleg a ***Firestarternek*** nem feltétlenül kell futnia. A program az */etc/rc.firewall* fájlban tárolja a tűzfal adatait. Mivel ez egy indítás szintű parancsfájl, valahányszor újraindítjuk a rendszert, a tűzfal már futni fog.

Itt a kiváló alkalom, hogy kis szünetet tartsunk és lazítsunk, amíg ***François*** mindenki poharát újratölti. Ezalatt hadd meséljek el még egy bölcseletet a biztonsággal kapcsolatban. Sok évvel ezelőtt valaki azt mondta, hogy a lehető legjobb biztonsági riasztórendszer, amit egy házhoz beszerezhetünk, az a kutya. Azt mondták, felejtsem el a csillogó elektronikus ketyeréket, és vegyek magamnak egy nagy német juhászt. Valószínűleg ez a gondolat sugallta a mai étrend második fogását. ***Simon Edwards Guarddog (Őrkutya)*** programja egy grafikus tűzfalbeállító eszköz, amely kutya biztonságossá teszi a ***Linux*** rendszert. A ***Guarddog*** az asztali számítógépeken nagyszerű, viszont még a bonyolult kiszolgáló összeállításokon is ideális eszköz.

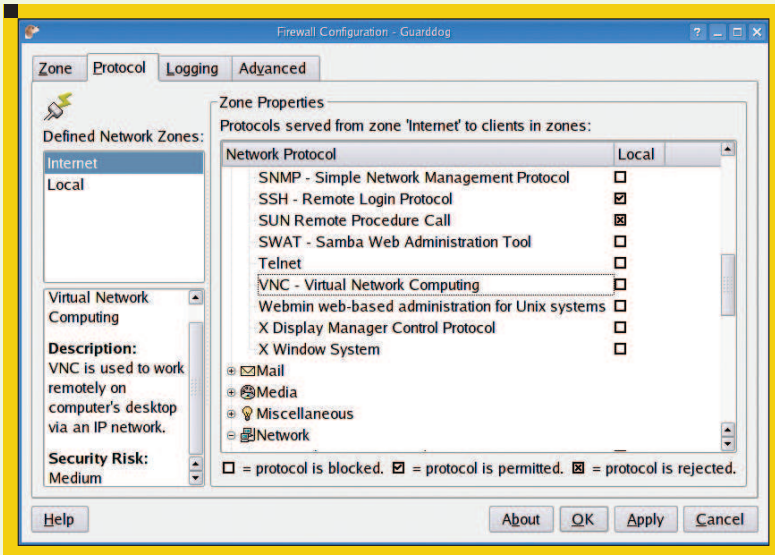
Mielőtt bemutatnám a ***Guarddogot***, ki kell emelni, hogy nem rendszergazdaként is futtatható, viszont a végrehajtott változtatásokat nem lehet menteni. Ez azért van, mert a tűzfal szabályainak megváltoztatásához rendszergazda hozzáférés szükséges. Egyértelműen jobb, ha rendszergazdaként futtatjuk az alkalmazást, kivéve, természetesen, ha előbb el szeretnénk sajátítani a működését. Ez hasznos lehet; nemsokára elmondom, miért. Még valamire fel szeretném hívni a figyelmet. A ***Guarddog*** a */etc/rc.firewall*-ban tárolja a tűzfalszabályokat, és így lehetséges (bár nem valószínű), hogy ezen a helyen már létezik egy fájl. Ebben az a furcsa (és talán egy kicsit mulatságos), hogy a ***Guarddog*** egy ilyen nevű fájlt telepít, és lehet, hogy induláskor beleütközik. Ez nem túl nagy gond, de ha erre utaló üzenetet látunk, jó, ha tudjuk, hogy valószínűleg minden



3. ábra Bejövő forgalomra vonatkozó szabály hozzáadása a Firestarterrel



4. ábra A Guarddog tűzfal program fő ablaka



5. ábra A Guarddog protokollokat egyetlen egérkattintással tilthatjuk, engedélyezhetjük vagy visszautasíthatjuk

rendben van. Adjuk meg a rendszergazda jelszót, és indítsuk a programot úgy, hogy teljes hozzáféréssel rendelkezik a tűzfalhoz.

A ***Guarddog*** fő kezelőfelülete négy lapból áll, amelyek ***Zone (Tartomány)***, ***Protocol (Protokoll)***, ***Logging (Naplózás)*** és ***Advanced (További beállítások)*** címkével rendelkeznek (*4. ábra*).

A ***Zone*** lapon két előre meghatározott tartomány szerepel. A ***Local (Helyi)*** a helyi címhez érkező forgalmat jelöli, az internet pedig a rendszert elhagyó, az internetre irányuló forgalmat. Ez nagyon fontos. A ***Guarddog*** a semleges terület (***Demilitarized Zone – DMZ***) beállítások, különböző kártyák és ehhez hasonlók alkalmazásával viszonylag könnyűvé teszi a bonyolult tűzfalak létrehozását. Egyelőre összpontosítsunk az alapvető asztali tűzfalbeállításra, ami egy internetre kapcsolódó gép.

Ahogy elindítjuk a ***Guarddogot*** (parancsnév guarddog) és az ***Apply (Alkalmazás)*** gombra kattintunk, azonnal működésbe lép a tűzfal, és minden bejövő illetve kimenő forgalmat visszatart. Ez egy erősen korlátozó beállítás, ezért nagy biztonságban vagyunk. Talán egy túlságosan is: semmi sem jut ki vagy be – ez az egyik nyomós érv, amiért érdemes először nem rendszergazdaként kísérletezni. Ez nem olyan furcsa, mint amilyennek először tűnik. A ***DMZ***-ben található rendszerek összetettebb tűzfalait, rendszerint kizárják a belső hálózatról, és csak néhány külső szolgáltatást kapcsolnak be. Ha túlságosan védett területre kerülnénk, kattintsunk az ***Advanced*** lapra, majd pipáljuk ki a bal felső sarokban szereplő ***Disable firewall (Tűzfal kikapcsolása)*** jelölőnégyzetet, ezután pedig kattintsunk a jobb alsó sarokban található ***Apply*** gombra. Szintén az ***Advanced*** lapon található az a gomb, amellyel visszatérhetünk a ***Guarddog*** alapértelmezett, mindent tiltó gyári beállításaihoz.

Így vagy úgy, engedélyeznünk kell bizonyos forgalmat. Kattintsunk a ***Protocol*** lapra, és megjelenik a különböző forgalomtípusokat képviselő kategóriák listája: ***Chat (Csevegés)***, ***Data Serve (Adatszolgáltatás)***, ***File Transfer (Fájltovábbítás)***, ***Game (Játék)***, ***Interactive Session (Interaktív munkamenet)***, ***Mail (Levelezés)***, ***Media (Média)***, ***Miscellaneous (Vegyes)***, ***Network (Hálózat)*** és ***User Defined (Felhasználói beállítás)***.

Minden kategória mellett látható egy plusz jel, az önálló protokollok pedig almenükben jelennek meg. Kattintsunk az egyes protokollokra, és a bal alsó táblában megjelenik a rövid leírásuk, valamint az általuk képviselt biztonsági kockázat becsült mértéke.

A protokollok neve mellett egy-egy jelölőnégyzet látható, amelyekre kattintva letilthatjuk, engedélyezhetjük vagy visszautasíthatjuk az adott protokollt (*5. ábra*). Ahogy korábban említettem, a kapu alaphelyzetben tiltott. Egyszer kattintva engedélyezzük a protokollt, ha pedig újból kattintunk, visszautasítjuk a csomagot.

A tűzfal korlátozó jellegét tekintve először végignéztem az internet tartomány protokolljait, és engedélyeztem mindent, ami szükséges (például azonnali üzenetküldés, e-mail, webböngészés és így tovább). Ezekre van szükség egy olyan asztali munkaállomáson, ahol lényegében minden kimenő forgalom engedélyezett.

A változtatások elvégzése után kattintsunk a fő ablak alján található ***Apply*** gombra az új tűzfalbeállítás érvényesítéséhez. Egy kis felugró ablak figyelmeztet, hogy a működő tűzfalon végrehajtott módosítások hatással lehetnek a meglévő kapcsolatokra. Kattintsunk a ***Continue (Folytatás)*** gombra a tűzfal újbóli élesítéséhez.

Ha valamilyen kiszolgálót működtetünk (például ***Samba*** fájlmegosztást), szinte hallhatjuk a kedves kiskutya morgását, ugye? Esetleg biztonságos héjszolgáltatás (***SSH***) is fut annak érdekében, hogy ezt a gépet egy másik, otthoni vagy irodai számítógépről elérhessük. Kattintsunk a ***Local*** tartományra, és válasszuk ki azokat a protokollokat, amelyekkel forgalmat bonyolítunk. Ne feledjük, hogy ez most bejövő forgalom, tehát nem szabad túl nagylelkűnek lenni.

Ha már erről van szó, attól tartok, nemsokára záróra. Azért nem kell sietni. Hűséges pincérem, ***François*** szívesen újratölti a poharukat még utoljára, mielőtt elbúcsúznánk. Kérem, emeljék magasra a poharat, ***mes amis***, és igyunk egymás egészségére.

A ***votré Santé! Bon appétit!***

Linux Journal 2006., 143. 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.

A CIKK FORRÁSAI

➲ www.linuxjournal.com/article/8745

SuperKaramba újratöltve

A SuperKaramba már többször szerepelt a Linuxvilágban, ezért most inkább azokhoz szólnék, akik az érdekességet nem a projekt felélesztésében vélik felfedezni, sokkal inkább az egyedi, látványos és hasznos témák kreálásában.

Mint a bevezetőben leszögeztem, ebben a cikkben nem célom a közkedvelt *„Karamba”* bemutatása. Még a feléleszésre sem térek ki: a sokszor bemutatott és csontig rágott témával senkit nem nem szeretnék feleslegesen untatni. A szokatlan bevezetést egy huszáros vágással le is zárom: a továbbiakban feltételezem, hogy az Olvasó (kedvenc disztribúciójának csomagkezelőjével, esetleg forráskódból) már feltelepítette ezt a kiváló programot, majd felhasználóként kiadta a `superkaramba` parancsot. A művelet eredményeképpen, néhány pillanat múlva a szoftver már kérte is a megnyitandó ***.theme*** leíró állományokat.

Szép, szebb, leghaszontalanabb

Amikor néhány éve először kutakodtam ***SuperKaramba*** témák után, némi iróniával „rosszmájú” következtetéseket vontam le: voltak szép munkák, akadtak szebbek, és teljesen felesleges szenzorokkal felszerelt giccses darabok. A méltán híres ➲ ***http://kde-look.org*** honlap ide vonatkozó választékából az egyik kezemen meg tudtam volna számolni az értelmes munkákat. Nem tudtam mire vélni a jelenséget, csak csóváltam a fejem: „Létezik olyan ember, aki azt szereti, ha az aktuális időjárást a monitorról olvassa le ahelyett, hogy kinézne az ablakon? Ennyire gyakori, ha valakinek kevés a rendszertálcán lévő óra, ezért egy éktelen nagy időmérőt tesz a képernyőre második eszközként?” De a legjobb csak ezután jött! Egy rendszerparaméter bűbájos kijelzésétől majdnem „üvöltve futottam ki a világból”: az alkotó neon-sárga színű információkat tett rikító piros háttérre. Utóbbi attrakciót hosszú ideig egy nagyon rossz tréfának véltem, miszerint talán olyan szándékkal született, hogy a felhasználó szeme órákon át káprázzon tőle.

Végül aztán rádöbbentem, nincs ezekben semmi különös, csupán nem vagyunk egyformák. Triviális felismerés volt, de ahogyan két ízlés egyezése annyira valószínű, mint két azonos pofoné, úgy más irányban kerestem a megoldást. A szubjektív megítélés szülte elégedetlenségemet munkába fojtottam, és csináltam magamnak egy olyan ***„Karamba-témát”***, amitől nem rázott ki a hideg. Ez a hajlandóság szerencsére mind a mai napig meg van bennem, mára azonban kissé más megközelítésben: az előbb leírt linket aktívan használom, bizonyos kreátorok ***GPL*** munkáit keresve. Ezeket a műveket aztán jellemzően átalakítom – saját ízlésemnek megfelelően.

Mire kell törekedned?

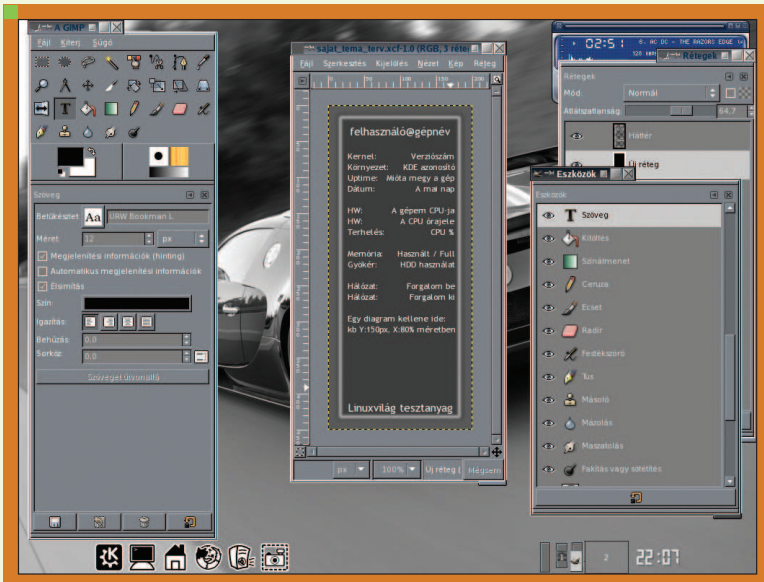
Nehéz kérdés! A választ nem is tudnám úgy megadni, hogy ne szóljon közbe a részrehajlás ténye. Inkább olyan megoldást választok, ahol leírom, hogy én milyen szempontok szerint képzelem a stílusos és hatékony ***SuperKaramba*** környezetet. Reményeim szerint ebből az Olvasó le tudja majd szűrni a tanulságokat, aminek hatására (nézeteim helyeslése vagy ellenzése által) szilárdulhat a véleménye – feltéve, hogy még csak „tapogatózik”. A hatékony munkaasztal megítélése a felhasználó beállítottságától függ. Van, aki esküszik a villámgyors ***BlackBox***-ra, van aki a ***GNOME***-ban

1. ábra A munkaasztalom, „pucéron”

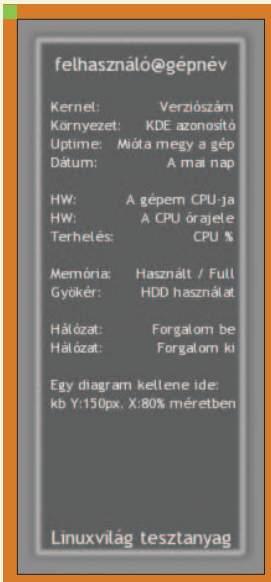
talál rá a számításaira, hogy a többi felületet ne is említsem. Mindenki másképp csinálja: ismerősen cseng az öregecske hazai sláger – és micsoda igazságot rejt! Én sem vagyok kivétel: évek óta ***KDE***-párti vagyok, az „elefánt méretű” környezet a kezdetektől fogva lenyűgöz engem is. No nem a méretével – nem is az igényeivel, sokkal inkább az összetett képességeivel: az extráktól roskadó asztal mérhetetlenül sok segédprogramot és lehetőséget biztosít a rálátásomhoz mérten. Így a példaként bemutatandó téma felépítésénél a ***KDE*** asztalom megjelenését tartottam szem előtt...

Íme, a formák és a színvilág

Személy szerint leginkább a fehér, a fekete és a kék színekkel vagyok kibékülve – ezeket (és árnyalataikat) huzamosabb ideig is el tudom viselni a képernyőmön. Pontosítanék: kizárólag csak ezeket tudom elviselni... A mindenható eléggé kényes látást „programozott” nekem: sajnos még egy csodaszép, színes természetfotó is zavaróan hat rám hosszú távon, így a tetszetős munkákat legtöbbször telítetlenné teszem.



2. ábra Készül a téma, épül a skicc



3. ábra A vázlat alapján dolgozom...

Az asztalom „alapját” rendszerint a következőképpen alakítom ki: három-négy szürkeárnyalatos képet bemásolok egy mindenki által olvasható (e célra fenntartott) mappába, majd készítek egy apró szkriptet, ami a grafikákat meghatározott szisztéma szerint „körkörösen” átnevezi a számítógép indulásakor. A ***KDE*** asztal hátterét aztán az egyik ilyen grafika tallózásával határozom meg, így minden egyes indítás után más háttérképpel fogad a profilom. Igyekszem az ikonszettekben is követni a kissé különc ízlésvilágomat, tehát a két színre szűrt remekműveket favorizálom. Természetesen a menüszerkezetet és a konzolokat is ehhez igazítom, miközben bizonyos fokú áttűnésre törekszem.

Külcsín...

Nos, egy ilyen „szürke világra” nem nehéz egyszerű és elegáns ***SuperKaramba*** témát készíteni, hiszen csak a színekben csökkentett környezet szabályait kell figyelembe venni. Képzeljünk el egy képet, ami az egész vizualitás alapját fogja nyújtani, lehetőleg áttetsző háttérréteggel. Nyúljunk bátran a ***GIMP***-ért! Kezdjünk nagyjából ***200x450px*** méretű áttetsző rajzfelületen, ennek sarkait kerekítsünk le, és lágy szélekkel öntsük ki a területet fekete színnel. A sötétre festett réteg áttetszőségét állítsuk hatvan százalékosra, majd egy új rétegen készítsünk egy fehér színű keretet.

Az eredményt ezután (összefésülve) el kell mentenünk tetszőleges névvel, mondjuk ***PNG*** formátumban (esetemben ***sajat.png***). Ezzel a résszel meg is volnánk, de még ne zárjuk be a képmanipulátort: a műveletek közben szerepe lesz!

A belbecs ereje

A történet most kezdődik igazán: a stílusos munka belbecsben is meg kell nyilvánuljon. Márpedig a lekérdezett (és kijelzett) paraméterek terén szükséges némi rálátás a ***Linux***/***Unix*** parancsokra, a ***SuperKaramba*** dokumentációjában leírt megoldások mellett. Utóbbi leírás egyébként nem szűkölködik a lehetőségek ismertetésében! Hálózati és rendszer információk, merevlemez „monitorozás”, egyéb programvezérlések egyaránt beleszőhetőek a témába. Az elképzelt megoldásomban a hálózat, a merevlemez és a memória adatait fogjuk előtérbe helyezni, emellett a rendszer paraméterei (kernel, felhasználó, gépnév, asztali környezet) is szerepet kapnak majd. A programvezérléshez nem fogunk görcsösen ragaszkodni, de az ***XMMS*** alapfunkcióit például integrálhatnánk... No, mindegy, erre a részre azért még visszatérek!
Az előző blokkban leírtam, hogyan lehet egyszerűen elkészíteni a munka grafikai alapját. Tényleg nem volt nagy dolog, viszont most komolyan végig kellene gondolni az alkotás megjelenését: a ***GIMP*** még fut, így egy újabb rétegen érdemes lehet kipróbálni néhány lehetőséget, szövegblokkokat létrehozva és türelmesen rendezgetve (valójában a kész témát szeretnénk előzetesen modellezni). Az elfogadhatónak ítélt skiccet hagyjuk nyitva!
A vázlaton szereplő, megjelenítendő paramétereket rögzíteni kell a ***SuperKaramba*** új témájának fő állományában. Tetszőleges szerkesztővel hozzunk létre egy szöveges fájlt, mondjuk ***sajat.theme*** néven! Első sorát (saját példám szerint) kezdjük így:

```
karamba x=30 y=30 w=200 h=450
```

Ez a következőket jelenti: „***Karamba***” témáról van szó, aminek alapértelmezett elhelyezkedése az asztal „***x,y***” pozíciójába kéretik, a kijelzőn lévő információk helyzetét pedig egy „***w,h***” pixelméretű mátrixon fogjuk megadni (ez rendszerint egyezik a létrehozott ****.png*** háttérkép dimenzióival).
A következő két sor így fest:

```
defaultfont color=255,255,255
➥fontsize=9 font="Tahoma"
➥shadow=2 bgcolor=0,0,0
image x=0 y=0 path="sajat.png"
➥name="background"
```

Ezzel azt közlöm, hogy az alapértelmezett szöveg színe fehér lesz (***RGB 255,255,255***), típusa ***Tahoma***, kilences méretben. A betűknek vastag fekete

4. ábra Az asztal bal oldalán a kész téma!

árnyékot szeretnék (***RGB 0,0,0***)! A téma grafikai alapját az előzőekben készített egyszerű ***sajat.png*** kép fogja biztosítani, amit a pixelmátrix ***x,y(0,0)*** pontjába kell elhelyezni (így a mátrix és a ***PNG*** grafika pontosan „lefedi” egymást).
A bevezetés ezzel meg is volna, következzenek kijelző alkotóelemei.
A *GIMP*-pel létrehozott skicc első szöveges sorában a büszke felhasználó és a számítógép neve „feszít”. Nagyon fontos, hogy ez a szöveg milyen pozícióban helyezkedik el! Váltsunk át a képmanipulátorra, és vigyük a mutatót a rétegelt kép első gépelt sora elé. A szerkesztő ablak bal alsó sarkában leolvasható a pontos pozíció: esetemben ez ***x,y(42,40)*** érték. Rögzítsük ezt is a ***theme*** fájlban, tehát a következő sornak nagyjából így kell kinéznie:

```
text x=42 y=40 sensor=program
➥program="echo ${USER}@
➥`hostname`" fontsize=14
```

Ugyebár említettem, némi gyakorlat szükséges a ***Linux*** parancsok terén: az `echo` parancs az aktuális szintaktikában itt a ***felhasználó@gépnév*** kimenetet adja eredményül. A pozicionálás nem szorul magyarázatra, viszont a ***sensor*** szó igen: így kell jelölni azokat az értékeket, melyek nem általunk rögzített szövegek, hanem „szenzorszerűen” kiolvasott adatok. A ***program*** nevű vezérlő pedig arra utal, hogy egy külső programból nyerjük ki az adatokat, nem pedig a „***Karamba***” beépített lehetőségeit használjuk (e külső program jelen esetben az említett ***echo***). A használandó betűtípus méreténél csúnyán eltértem az alapértelmezettől – azért, hogy ez a sor azonnal szemet szúrjon mindenkinek!
Ettől kezdve kicsit kuszább lesz a helyzet, mivel a kijelző ugyanazon sorában lesz a szenzoros érték és az általunk megadott szöveg is, a példában a „Kernel:” felirat, rögtön utána a a mag jelölése. Ezt a következőképpen tudjuk megoldani:

```
text x=30 y=76 value="Kernel:"
text x=116 y=76 sensor=program
➥program="uname -r"
```

A pozicionálás itt sem érdemel különösebb értelmezést, hiszen a *GIMP* folyamatosan „súg”: a magassági értékek értelemszerűen egyeznek, „***x***” irányban azonban el vannak tolva.
Az első sorban olvasható ***value*** jelölés szolgál arra, hogy a „Kernel:” felirat statikus legyen. Eközben az `uname -r` parancs kiolvassa a mag verziószámát. Most több elképzelést valósítsunk meg „egy körben”!

```
text x=30 y=91
value="Környezet:"
text x=140 y=91 sensor=program
➥program="kde-config -version
➥| grep KDE | sed -e 's/.*:
➥//'"
```

```
text x=30 y=106 value="Uptime:"
text x=102 y=106 sensor=uptime
➥format="%Hh:%Mm:%Ss"
➥interval=1000
text x=30 y=121 value="Dátum:"
text x=104 y=121 sensor=time
➥format="yy.MM.dd ddd"
➥interval=60000
```

Nos, így kell lekérdeznünk a ***KDE*** környezet verzióját, majd a ***sed*** által érdemes levágunk az információ felesleges részét. Egyúttal megmérjük a bekapcsolás óta eltelt időt, hogy adott formátumban ki tudjuk jelezni (ehhez nem kell külső program, így itt kihagytam a programvezérlőt). Az `uptime` szenzor frissítését ***1000 ms***-os (tehát egy másodperces) időközökre állítjuk, az `interval=1000` érték segítségével.
Végül, de nem utolsó sorban a dátum is a témára fog kerülni, amit percenként leellenőrzünk (ezen a ponton kérhetnénk jóval lomhább frissítést is, viszont egy esetleges éjféli „átfordulás” esetén a kijelző csúnyán késne).
A következő blokk a processzorra, a gyökér partícióra és a memória foglaltságára fog vonatkozni. Kezdjük a központi egységgel! A ***CPU*** adatainak lekérdezéséért egyaránt felel a „külsős” `cpuinfo` és az `uname -p` parancs, valamint a ***SuperKaramba*** beépített ***CPU*** szenzora. A terhelés mérése másodperces ciklusokban esedékes, az órajel kijelzése fél percenként frissül:

```
text x=30 y=150 value="HW:"
text x=126 y=150 sensor=program
➥program="uname -p"
text x=30 y=165 value="Órajel:"
text x=120 y=165 sensor=program
➥program="cat /proc/cpuinfo |
➥grep 'cpu MHz' | sed -e
➥'s/.*: //'| sed -e
➥'s/\..../MHz/'"
➥interval=30000
text x=30 y=180
➥value="Terhelés:"
text x=148 y=180 sensor=cpu
➥format="%v%" interval=1000
```

Ezek után már semmiség a gyökér és a központi tár mérése, kijelzése. Erre a célra szintén a beépített szenzorokat használhatjuk legegyszerűbben:

```
text x=30 y=212
➥value="Memória:"
```

```
text x=98 y=212 sensor=memory
➥format="%um / %tm Mb"
text x=30 y=227 value="Gyökér
➥foglalt:"
text x=146 y=227 sensor=disk
➥format="%up%" mountpoint="/"
```

A hálózati vezérlő sem maradhat ki a listából, a befelé és a kifelé mutató adatforgalom sebességét szeretném monitorozni:

```
text x=30 y=260 value="Hálózat
➥(le):"
text x=128 y=260 sensor=network
➥device="eth0" format="%in
➥kB/s" interval=2000
➥decimals=1
text x=30 y=275 value="Hálózat
➥(fel):"
text x=128 y=275 sensor=network
➥device="eth0" format="%out
➥kB/s" interval=1000
➥decimals=1
```

Most már csak egy dolog van hátra. Egy szép, aktív grafikont gondoltam ki a példa elején, ami a ***CPU*** terhelését mutatja folyamatában (a szándékot a skiccen megjelenítettem). Erre is van beépített lehetőség, amit a következők szerint kell használni:

```
graph x=50 y=312 w=100 h=80
➥sensor=cpu points=100
➥color=255,255,255
```

Tehát az ***x,y*** koordinátákra kérünk egy fehér színű aktív grafikont, ami ***100x80*** pixel szélességben ***100*** lehetséges értéket tud megjeleníteni. Persze az analizátor jobban mutat, ha van mögötte egy erre a célra készült mutatós háttérgrafika. Ezt most nem részletezem, a ***GIMP*** segítségével gyorsan összeütök egy ***130x100*** pixel méretű gráfot. Ha elkészült, akkor a ***sajat.theme*** három bevezető sora után beszúrom negyediknek a friss-ropogós képet:

```
image x=36 y=300
➥path="racs.png"
```

Nos, ha mindennel végeztünk, akkor a létrehozott két képet, valamint a ***sajat.theme*** állományt mozgassuk ugyanabba a mappába – majd a `superkaramba` parancs kiadása után tallózzuk ki a munkánkat. Remélem, tetszik a letisztult a téma. A pozicionálás pontosítása ugyan még hátravan, de ez már teljesen személy- és asztal függő. Így ha valakinek nem tetszik egy kijelzett adat helye, akkor az ***x,y*** koordinátákon változtasson bátran. A változtatások idejére azonban az aktuális témát mindig be kell zárni, különben a ***SuperKaramba*** (bizonyos verziókban) futása hibával megszakad. Említettem, hogy akár az ***XMMS*** vezérlését is rátehetnénk az elkészült műre. Engedelmetekkel nem követném el ezt a „bűnt” (nem látom értelmét), de aki mindenképpen szeretné megoldani, annak először is készítenie kell egy gombsor grafikát. Olyan képet kell szerkeszteni, ahol a népszerű lejátszó alapfunkcióihoz (előző szám, lejátszás, állj, következő szám) tartozó gombok egymás mellett sorakoznak, a téma szélességéhez igazodva. Nevezzük a képet ***xmms_irany.png*** néven! Ekkor a vezérlést biztosító sorok valahogy így festenek:

```
image x=60 y=400
➥path="xmms_irany.png"
clickarea x=60 y=400 w=20 h=20
➥onclick="xmms -rew"
clickarea x=90 y=400 w=20 h=20
➥onclick="xmms -play"
clickarea x=120 y=400 w=20 h=20
➥onclick="xmms -stop"
clickarea x=150 y=400 w=20 h=20
➥onclick="xmms -fwd"
```

A sorok működése rém egyszerű: meghívjuk a gombsoros képet ***x,y(30,400)*** pozícióba, majd a ***WEB*** kapcsán ismerős ***Imagemap*** eljáráshoz hasonlóan területeket definiálunk. Az első aktív sor így értelmezhető: a ***x,y(60,400)*** ponttól jobbra, lefelé irányban ***20x20px*** méretű területen figyeljük az egérgomb lenyomását. Ha megtörténik, akkor kiadjuk az `xmms --opció` parancsot (mivel a lejátszó futása például egy konzolról is változtatható a leírt utasításokkal). A teljes igazságoz azonban tartozik még két apró információ, amivel tartozom. Az egyik dolog egy fontos képesség „elhallgatása”: amennyiben a témát tartalmazó mappában létrehozunk egy (bizonyos szabályok szerint felépülő) ***Python*** szkriptet, akkor az interaktivitás magasabb szintre emelhető. Például a kész kijelzőn jobb egérklikket nyomva adott dolog történhet: akár a háttérgrafika cseréje,

5. ábra RAM adatai borzasztóak, valami nem ürít...

vagy éppen a teljes kinézet átváltozása. Amennyiben igény mutatkozik rá, akkor vissza fogunk majd térni erre egy későbbi számban – ahol egy nagyon összetett, sok blokkos, több ablakos interaktív téma létrehozásával fogok megpróbálkozni.
A másik apróság arról szól, hogy a létrehozott munka természetesen egyszerű példa jellegű. Személy szerint a cikk elején olvasható linkről töltöm le a nekem tetsző remekműveket, majd saját ízlésem szerint „rombolom szét” őket, hogy a fekete-fehérre szűrt asztalomhoz illeszkedjenek...
Aki kedvet érezne saját munkák alkotásához, miközben nem szeretné begépelni a leírt sorokat (vagy éppen nem áll kezére a ***GIMP***), látogasson el fél percre a ➲ ***http://kovi.uw.hu/lv2007*** címre! Ide töltöttem fel a cikkben szereplő grafikákat nyers formákkal, és az aktuális ***theme*** állományt – segítségképpen. Egyet kérek szépen mindössze (mondjuk úgy, cserébe): ha valaki alkotott egy tetszetős és hasznos ***SuperKaramba*** témát, azt ne felejtse el önzetlenül megosztani a többiekkel, tehát irány a ➲ ***http://kde-look.org***! Tartalmas kikapcsolódást!

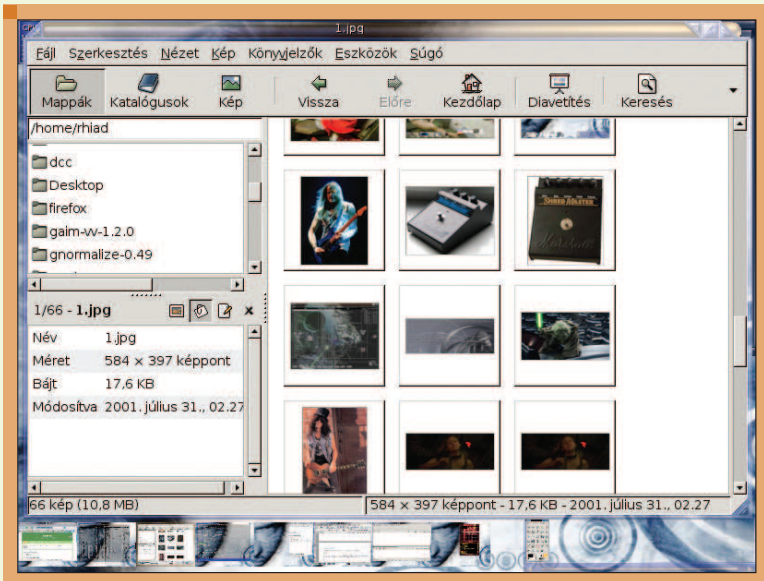
Kovács Zsolt (kovi@linuxforum.hu)

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

gThumb – Képnézegetés egyszerűen

Régen, környezetünk leírására/dokumentálására könyvek, útleírások szolgáltak. Manapság a képek és filmek korszaka van. Hogy kinek mi a jobb, azt maga döntse el. Mindenesetre ebben a cikkben egy GNU/Linuxra írt, könnyen kezelhető képnézegetővel fogunk megismerkedni, mely kis mérete ellenére is sok hasznos funkciót lát el, és mindkét tábor örömére szolgálhat.

Ha a *Kedves Olvasó* figyelemmel kíséri cikkeimet, akkor tudhatja, hogy ezek általában személyes tapasztalatok és benyomások útján születnek. Most sem lesz ez másképp. Személy szerint nekem az egész „képnézegetősdi" a *windowsos ACDSee-vel* kezdődött. Már a neve is megfogott ennek az alkalmazásnak, hiszen minden keményebb zenéket kedvelő egyén rögtön az *AC/DC-re* asszociálhat belőle, ahogy tettem ezt én is. Előtte nem is gondoltam volna, hogy létezik olyan program, mely képes az egy könyvtárban lévő képeket egymás után „lejátszani".
Nos, a *windowsos* idők hamar elmúltak, az ablakos rendszert felváltotta a *Linux*. Kellet találni egy hasonló programot, mely megfelel a legtöbb elvárásnak.



1. ábra gThumb az indítás után

Képnézegető programok

Nagyon egyszerű egzakt definíciót találni a képnézegető programra. Egy program, mellyel képeket lehet nézegetni. Igen ám, de ennél azért egy picit bonyolultabb a helyzet. Egy képet akár egy böngészővel is meg tudunk nézni. Sőt, ha jók az információim, akkor minden operációs rendszer alatt az alapértelmezett képnéző program egy webböngésző. Természetesen ezt csak elnagyoltan jelenthetem ki, de az biztos, hogy *Windows* alatt az *Internet Explorer*, míg a linuxos grafikus felületek favoritja, a *KDE* alatt a *Konqueror* az alapértelmezett képnéző alkalmazás.

A képnézegető programok azonban többre hivatottak, mint szimplán képek nézése. Lehet velük képeket manipulálni, forgatni, átméretezni, stb. Számtalan ilyen alkalmazás létezik, számtalan operációs rendszerre. Jelen esetben e család egy pehelysúlyú tagját fogjuk górcső alá venni.

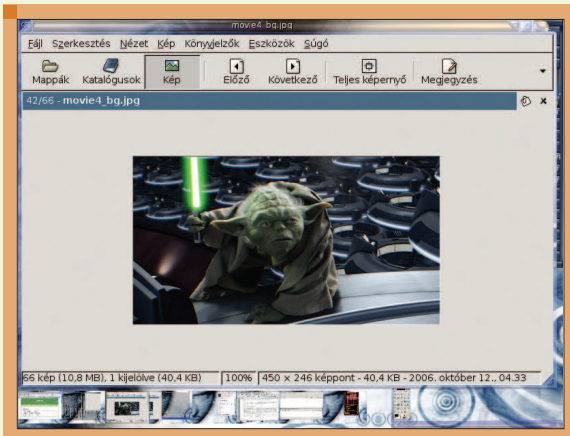
gThumb

A *gThumb* nevű alkalmazás egy képnézegető és böngésző program, mely beleépül a *Gnome* grafikus környezetbe. Természetesen, – mint már megszokhattuk – ez nem jelenti azt, hogy más grafikus felületek alatt ne tudnánk használni. Kinézete homogén, köszönhetően a *GTK-nak*, így bármely ablakozó sémájához illik. Jelenlegi stabil verziója a 2.6.9., ami a hivatalos honlapról letölthető (➲ *http://gthumb.sourceforge.net/*).

Telepítés

Minden *Linux* disztribúció, amely tartalmazza a *Gnome* felületet, tartalmazza a *gThumb-ot* is. Ha ez mégsem állna fenn, akkor az alkalmazást telpítenünk kell. Legegyszerűbben disztribúciónk *ftp-szerveréről* telepíthetünk. Például *Debian GNU/Linux* alatt az

```
apt-get install gthumb
```



2. ábra Kép megtekintése

3. ábra Nagyítás és pozícionálás

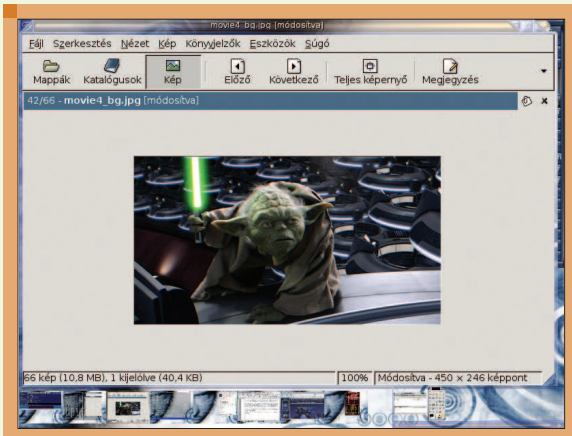
paranccsal. Ilyenkor a telepítő (***Debian*** alatt az ***apt***) magával húzza a ***gThumb*** függőségeit is, ezzel egyszersmind gördülékenyebbé teszi a folyamatot.
Természetesen forrásból is telepíthetünk, itt azonban nagyon kell vigyázni arra, hogy minden függőség fent legyen a procedúra megkezdése előtt. Személy szerint az ***ftp-s*** megoldást javasolnám, már amennyiben rendelkezünk csomagtelepítővel (***apt***, ***slapt***, ***urpmi***, ***emerge***, stb.).

Használat és funkciók

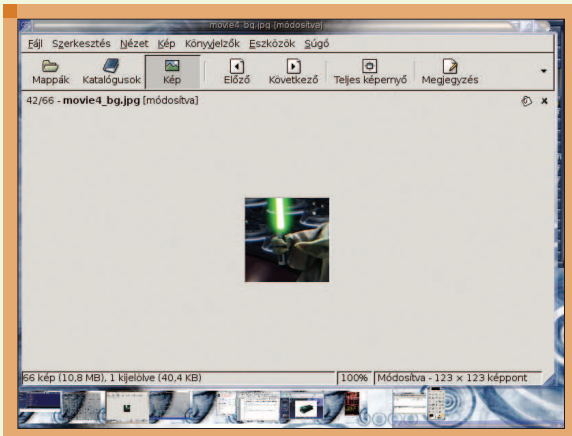
A ***gThumb*** indítható menüből, ikonból, illetve bármilyen grafikus terminálba beírt

```
gthumb
```

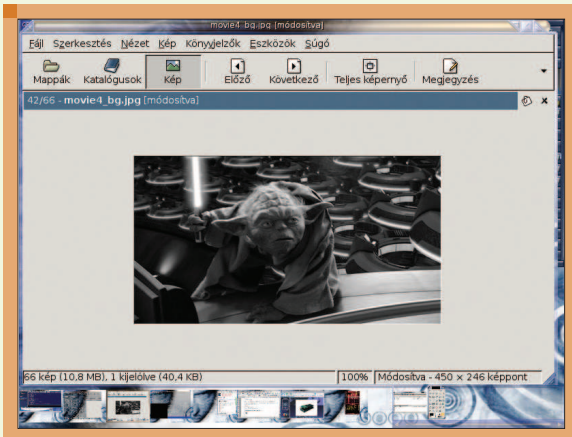
paranccsal. Ezzel kezdetét veheti a képnézegetés. Rögtön láthatjuk, ha egy képeket tartalmazó könyvtárba lépünk, hogy az alkalmazás ezen képeket megjeleníti kicsiben. Nézzük meg egy kicsit jobban ezt az ábrát! A fő ablak három részre van osztva: a bal felső részben láthatjuk a könyvtárainkat, melyekben kedvünkre böngészhetünk, a bal alsó részben a kijelölt képről találhatunk hasznos információkat, illetve az ablak legnagyobb részében képeink miniatűrjeit figyelhetjük meg. Ez az elrendezés teljesen logikus, egy ablakban történik minden. A képek kicsinyített mása megkönnyíti a gyors keresést, a könyvtárstruktúra-ablak megkönnyíti a gyors könyvtárváltást, az információs ablak pedig elárulja nekünk a kép nevét, felbontását, illetve a fájl méretéről és az utolsó módosítás időpontjáról is tájékoztat. Tehát a program alaphelyzetével megismerkedtünk, lássuk most a tényleges használatát. Egy kicsinyített képre kattintsunk kétszer, így az megjelenik eredeti méretben.
Megfigyelhetjük, hogy a ***Yoda-t*** ábrázoló kép nem túl nagy, így teljes méretben is az ablak közepén helyezkedik el. De előtte vizsgáljuk meg az ablak legalsó információs sorát. Most ez helyettesíti az imént említett felosztott ablakban a bal alsó információs részt. Balról jobbra haladva láthatjuk az adott könyvtárban lévő képek számát és méretét, az adott kép méretét, illetve nagyítását (most ez pont 100%-os), majd pedig a felbontását és az utolsó módosítás dátumát. Az ablak fejlécében pedig megjelenik a kép neve.
Lehetőségünk nyílik kicsinyíteni vagy nagyítani a képet. Ezen műveletekre a '**-**' és a '**+**' billentyűk szolgálnak.
A túl nagy képben pedig mozoghatunk a billentyűzetünkön található nyilak segítségével. Nézzünk egy példát erre. Tegyük fel, hogy csak ***Yoda*** fénykardjának a „pengéje" érdekel minket. Nyomjuk meg annyiszor a '**+**' billentyűt, amekkorára szeretnénk nagyítani a képet, illetve a nyilakkal pozicionáljuk magunkat a pengére. Természetesen nem csak kis méretben nézhetjük végig az egy könyvtárban lévő képeket, hanem nagyban is. Kattintsunk az első képre (de persze bármelyikre lehet) kétszer, ekkor megjelenik nagyban. Ha bárhova kattintunk a képen bal egérgombbal, akkor megjelenik a következő kép. Így szépen végigmehetünk az összesen.
Persze visszafelé is lapozhatunk, erre a középső egérgomb szolgál. „Gyengébbek kedvéért" vannak lapozást segítő ikonok is, azokat is használhatjuk. Most, hogy a legfőbb, mindennapos funkciókon átküzdöttük magunkat játékosan, nézzük egy kicsit sematikusabban a ***gThumb*** szolgáltatásait. Az alkalmazás képnéző része egyenként jeleníti meg a képeket, beleértve az animációkat is, amik általában ***GIF*** képek. Támogatott képformátumok még: ***BMP, JPEG (JPG), PNG, TIFF, ICO, XPM***.
EXIF-adatokat (amik ***JPG*** képekhez vannak csatolva) nézhetünk és ezeket módosíthatjuk is. A megjelenített képet forgathatjuk, tükrözhetjük, illetve invertálhatjuk a ***Kép/Átalakítás*** menüben. Minden átalakítás után, ha továbblépünk, a program rákérdez, hogy mentse-e a módosított fájlt. Ezen felül használhatjuk a teljes képernyős módot az '***f***' billentyű lenyomásával. Normál módba az ***Esc*** billentyűvel válthatunk vissza. Persze itt is könnyítenek a helyzetünkön a következőképpen: teljes képernyős módban, ha megmozdítjuk az egeret, előjön egy úgynevezett navigációs menü. Ez tartalmaz normál módba visszaváltó, lapozó és egyéb gombokat.
A képböngészés is bővelkedik extra funkciókban. A fentebb leirt módon böngészhetünk a képek között, törölhetünk, másolhatunk képeket, egész könyvtárakat, illetve létre is hozhatunk újakat. A ***gThumb*** automatikusan frissíti a könyvtárak tartalmát, ami egy nagyon-nagyon hasznos funkció. Nem kell folyton ki-be lépkednünk, hogy az új fájlokat láthassuk.



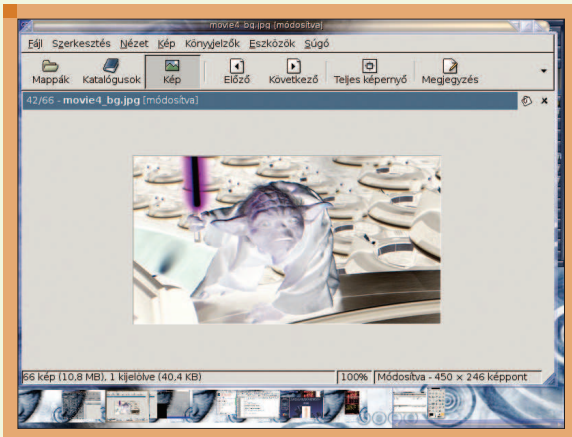
4. ábra Yoda feljavítva



5. ábra Levágás funkció



6. ábra Színes képből fekete-fehér, egy kattintással

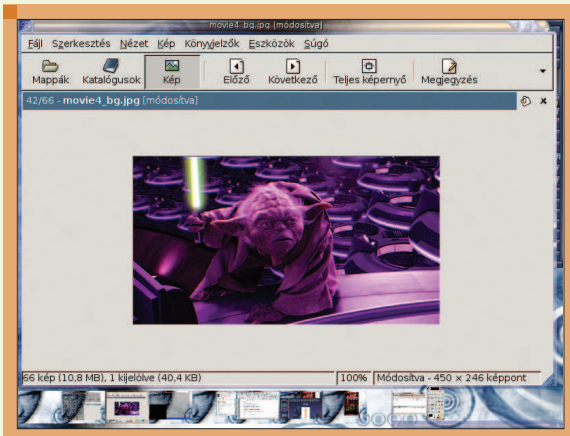


7. ábra Yoda negatívja is a jó oldalon áll

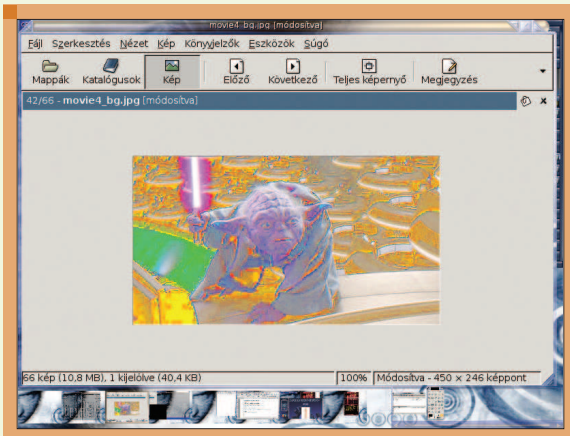
Ráadásul itt is – mint a webböngészőkben – felvehetünk kedvenceket. Persze értelemszerűen itt a kedvencek nem webhelyek, hanem képeket tartalmazó könyvtárak lehetnek. Ezt a ***Könyvjelzők*** menüben tehetjük meg. Készíthetünk katalógust is a képekről. Igaz, én ezt a funkciót nem használom, de nyilván hasznos, ha többezer képet szeretnénk rendszerezni. Fűzhetünk megjegyzéseket egyes képekhez, a katalógusokat külön könyvtárakba gyűjthetjük. Ezen felül, ami a legfontosabb, egy remek grafikus kereső is a segítségünkre van, ha nem akarunk minden képet megnézni ahhoz, hogy a keresettet megtaláljuk. Ezt a ***Szerkesztés/Keresés*** menüben találjuk, a keresési eredményt pedig elmenthetjük egy fájlba.
Elérkeztünk az egyik legfontosabb funkciójához a ***gThumb-nak***, ez pedig a képszerkesztő. A példákhoz vissza fogunk térni ***Yoda mesterhez***.

Az összes képszerkesztési funkció a ***Kép*** menüben érhető el. Nézzük őket sorjában:

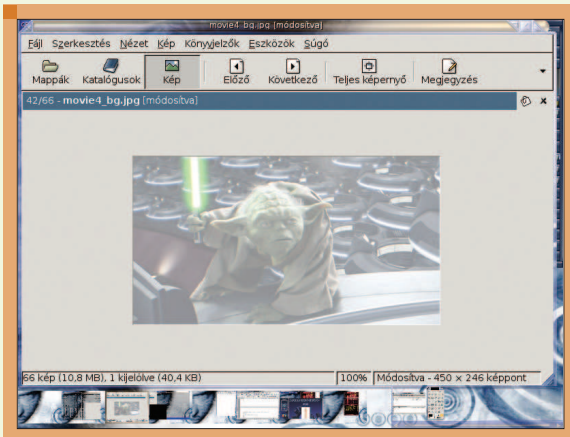
- ***Javítás***: Ez az opció egy erősebb kontrasztot ad a képnek, tehát vehetjük úgy, hogy egy „nagygenerál” jellegű javítást csinál.
- ***Átméretezés***: Nem fűznék hozzá sok kommentárt. Az eredeti kép méretét változtathatjuk meg.
- ***Levágás***: Lehetne téglalap kivágásának is nevezni ezt a funkciót. Többfajta sémából választhatunk, de a sablon mindig csak téglalap lehet.
- ***Átalakítás***: Erről már volt szó korábban.
- ***Szürkítés***: Magyarul színes képből fekete-fehér lesz.
- ***Negatív***: Szép színes képből csúnya absztraktszínes negatívot készít.
- ***Színegyensúly***: A kép ***RGB*** (piros-zöld-kék) értékeit változtathatjuk kedvünk szerint.
- ***Árnyalat és telítettség***: A kép árnyalatát, fényességét és telítettségét állíthatjuk.
- ***Fényerő és kontraszt***: Értelemszerűen sötét és életlen képeket javíthatunk fel ezzel az opcióval.
- ***Poszter***: Poszterszerűvé (durvábbá és színesebbé) varázsolhatjuk a képeket.
- ***Automatikus kiegyenlítés/normalizálás/kontraszt kifeszítése***: Mind olyan funkciók, amiket már tárgyaltunk. Itt viszont



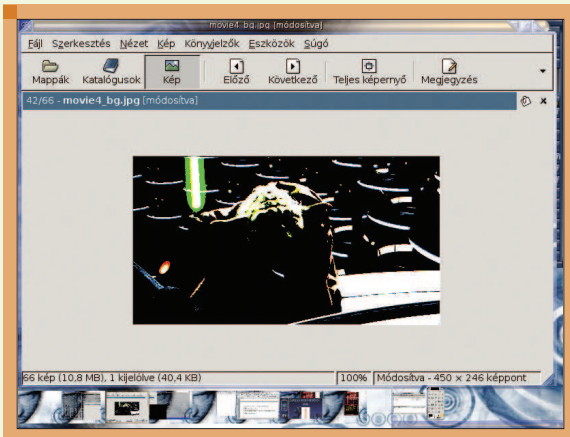
8. ábra „Zavart érzek a színegyensúlyban”



9. ábra Árnyalat és telítettség a maximumon



10. ábra A fényerő is Yoda-val van



11. ábra Yoda poszteren is jól mutat

lehetőségünk nyílik a programra bízni a döntést a helyes értékek kiválasztásában.

Fontos dolog, hogy programunk többféle formátumba képes elmenteni képeinket: ***JPEG (JPG), PNG, TIFF, TGA***. Konvertálni (***Eszközök/Átalakítás más formátumba*** menü) ugyanezek között tud, tehát például egy ***jpg-t png-be*** konvertálni ***gThumb-bal*** gyerekjáték.

Haladó funkciók

Mint minden komolyabb program, ***gThumb*** is rendelkezik néhány extra vagy haladó funkcióval. Pár szóban nézzük ezeket is.
A ***Fájl/Fényképek importálása*** menüben található egy varázsló, mely képes direkten, digitális fényképezőgépről képeket importálni a programba. Ez az eszköz el van látva egy intelligens fényképezőgép-felismerő opcióval is.

Érdekes, és manapság közkedvelt funkció az úgynevezett diavetítés, vagy angolul ***slide show***. A ***gThumb*** erre is képes. A diavetítés azt jelenti jelen esetben, hogy nem kell a képek „lejátszása” közben kattintgatnunk, a képek automatikusan követik egymást. A ***Nézet/Diavetítés*** menüre kattintva elkezdődik a lejátszás.
A diavetítés beállításait a ***Szerkesztés/Beállítások*** menüben találjuk. Állíthatjuk, hogy sima vagy teljes képernyős módot szeretnénk-e használni, milyen időközönként kövessék egymást a képek, illetve arról is dönthetünk, hogy a könyvtár elejéről vagy végéről kezdődjön-e a vetítés.
Az éppen aktuális „nézett” képet beállíthatjuk háttérképnek az asztalunkra. Ehhez el kell látogatnunk az ***Eszközök/Kép beállítása háttérképként*** menübe. Ez lehet középre igazított, mozaik, kifeszített vagy méretezett.
További extra funkciók még a web-album létrehozása, indexkép létrehozása, duplán létező képek keresése, veszteségmentes ***JPG*** mentése, a képek dátumának megváltoztatása, illetve a sorozatos átnevezés.
Remélem sikerült érdekes ízelítőt adnom ebből a programból. Egyszerű, mégis gazdagon bővelkedik funkciókban. Jómagam a kezdetek óta ezt használom, és bátran állíthatom, hogy teljes mértékben meg vagyok vele elégedve. Ajánlom mindenkinek, hogy tegyen egy próbát vele!

Apagyi György, (killall)
(killall@root.hu)

25 éves, jelenleg az ELTE programozó matematikus szakán másodéves hallgató. Hobbija a zene (gitározás), az olvasás (Stephen King) és a számítástechnika (Linux, Unix, VMS).

A DansGuardian tartalomszűrő és a pehelysúlyú Tinyproxy összehangolása és beállítása

Amikor a *Microsoft* felhasználók kezdenek *Linux* operációs rendszert használni, különböző elvárásokkal érkeznek; például olyan tartalomszűrőt keresnek, mint amilyet *Microsoft Windows XP* alatt is használtak. A *Linuxra* áttérők gyakran az otthoni, különálló számítógépükkel kísérleteznek. Mivel a legtöbb ember arra használja számítógépét, hogy megfelelő információkat, képeket töltsön le az internetről, a tartalomszűrő rendszer használata kulcsfontosságú – különösen akkor, ha a szülők és a gyermekek közös számítógépet használnak, és a felnőtt felügyelet nem mindig megoldott. A *DansGuardian* és a *Tinyproxy* használatával a szülők távollétükben is felügyelhetik az internetes tartalmakat. A *DansGuardian* sokoldalú tartalomszűrő; nyílt forrású szoftver, amelyet alapértelmezett beállításaival nem kereskedelmi használatra szántak. A kereskedelmi változathoz szánt konfigurációhoz hozzájuthatunk a megfelelő licenc (vagy a *„SmoothGuardian”*) megvásárlásával. A *Tinyproxy* együttműködik a *DansGuardiannal* – ez egy kicsiny, nyílt forrású program, amely képes értelmezni és kiértékelni a számítógépen áthaladó információkat. E két eszköz együtt olyan adminisztratív felügyeleti lehetőséget biztosít, mellyel hatékonyan gátat lehet vetni a célba vett internetes tartalmaknak.

Tartalomszűrés 5000 láb magasból

A *DansGuardian* nem más, mint megadott szavak, mondatok és képek által megfogalmazott áthaladásgátlók együttese, melyek révén egyes weboldalak letilthatóak. A *DansGuardian* szűrői az internet és a böngészőprogram (például *Firefox*) közé illesztett programként működnek. A *Firefox* a weboldalak lekérését a *DansGuardianhoz* intézi, ami ezt a *Tinyproxynak* továbbítja – ez tartja a közvetlen kapcsolatot az internettel. Az internetről érkező adatcsomagok a *Tinyproxyn* és a *DansGuardianon* haladnak keresztül, mielőtt elérkeznének a böngészőklienshez. Természetesen csak a jóváhagyott információk jutnak át a szűrőkön és jelennek meg a böngészőablakban; tiltott weboldalak esetén a *DansGuardian* egy *„access denied” („hozzáférés megtagadva”)* képernyőt jelenít meg.
Mindez természetesen a szűrési folyamatnak csak egy meglehetősen vázlatos leírása. Valójában ennél sokkal összetettebb és érdekesebb a *DansGuardian* és a *Tinyproxy* együttműködése. Aki erre kíváncsi, látogasson el a *DansGuardian „folyamatábra”* oldalára (*„Flow of Events”*, lásd a cikkhez tartozó forrásokat). Itt egy mélyrehatóbb tanulmányt olvashatunk arról, hogy hogyan működnek ezek a szűrők, és hogy miként továbbítódnak az adatok a két program és az internet között.

Amit viszont fontos tudnunk: a *DansGuardiannak* megadható soksok tilalom alá eső szó, kifejezés, *URL*. A weboldalakon található szövegek vizsgálatán túl a *DansGuardian* még képek alapján is tud szűrni, és meg tudja gátolni bizonyos fájlok letöltését. Ez a szűrőmódszer kombináció sokkal hatékonyabb, mint az olyanok, amelyek csak a tilalomlistán levő *URL*-ek alapján korlátozzák a böngészést. Kezdő *Linux*-felhasználók számára először bonyolultnak tűnhet a *DansGuardian* húszegynéhány konfigurációs fájlja – azonban világos útmutatást kapunk, hogy miként kell őket igényeinkhez mérten megszerkeszteni. Próbálkozásaim során alig kellett változtatnom ezeken, mert az alapértelmezett szűrési beállítások szinte tökéletesen megfelelnek családi használatra.

A telepítés

Először a *DansGuardiant* és a *Tinyproxyt* kell telepíteni és beállítani. Ezt követően igen fontos lépés, hogy úgy állítsuk be az asztali környezetünket, hogy a normál felhasználók ne tudják egyszerűen kikapcsolni a tartalomszűrést.
Telepítés előtt érdemes megvizsgálni, hogy disztribúciónk tartalmazza-e a *DansGuardian* és *Tinyproxy* csomagokat. Néhány esetben a legegysze-

rűbb ezeket egy grafikus felületű csomagkezelővel, például a *Novell SUSE YaST* programjával vagy a *Synaptic*-kal feltelepíteni. Debian Linuxban elendő a a *root* felhasználó által kiadott

```
apt-get install dansguardian
➥tinyproxy
```

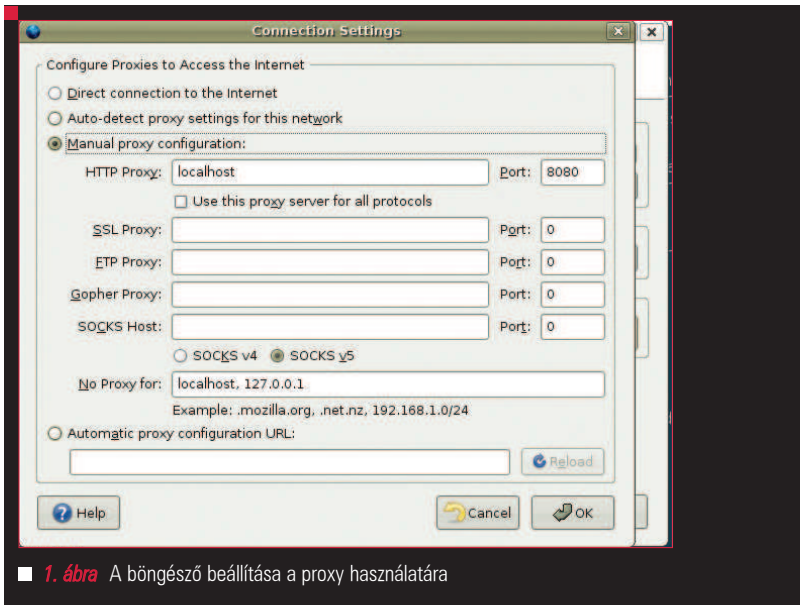
parancs.
Ha ezen programok bináris változatai netán mégsem lennének megtalálhatóak az adott disztribúcióban, le is lehet tölteni őket a megfelelő webhelyekről (lásd a cikkhez tartozó forrásokat). Letöltés után az *INSTALL* fájlban olvasható a telepítéshez szükséges eligazítás.

A DansGuardian és a Tinyproxy beállítása

Következő teendőnk a *DansGuardian* és a *Tinyproxy* konfigurációs fájljainak testre szabása. Tesztelési célokra *Ubuntu Dapper Drake*-et használok, így a könyvtárak és fájlnevek ezt a világot tükrözik. Nyilván más disztribúciók is többé-kevésbé hasonlóan szervezik fájljaikat; előfordulhat, hogy kicsit körül kell nézni, hogy hol is található a telepítési könyvtár. A jellemzők átszerkesztéséhez egy közönséges editor is elég, mint például a *GNOME gedit* programja.
Saját szerkesztőnkkel – root felhasználóként – nyissuk meg a */etc/dansguardian/dansguardian.conf* fájlt. Módosítsuk a *filterport*, a *proxyip* és a *proxyport* értékét az alábbiaknak megfelelően. Disztribúciónktól függően szükség lehet néhány `UNCONFIGURED` szóval kezdődő sor megjegyzéssé alakítására („kikommentezésére") a **#** jel segítségével.

```
# the port that DansGuardian
➥listens to.
filterport = 8080
# the ip of the proxy–default
➥is the loopback (this server)
proxyip = 127.0.0.1
# the port DansGuardian
➥connects to proxy on
proxyport = 3128
```

A *DansGuardian* általában a 3128-as portra kapcsolódik alapértelmezetten, mert ugyanezt a portot használja a méltán oly népszerű *Squid* proxy is.



1. ábra A böngésző beállítása a proxy használatára

Két utat választhatunk: vagy ezt az értéket állítsuk át a *Tinyproxy* által használt alapértelmezett portszámra (8888), vagy a *Tinyproxy* port értékét változtassuk meg a *DansGuardian* (azaz a *Squid* alapértelmezett port) értékére. Én ez utóbbit követtem.
A *Tinyproxy* testreszabásához – root felhasználóként – nyissuk meg szerkesztésre a */etc/tinyproxy/tinyproxy.conf* fájlt. Olvassuk végig, és győződjünk meg arról, hogy a *User, Group, Port* és *ViaProxyName* (*felhasználó, csoport, port, proxynkeresztül*) értékét szükséges-e megváltoztatnunk. Ha ezt az utat választjuk, akkor fontos, hogy a *Tinyproxy* port értékét módosítsuk úgy, hogy a *DansGuardian* által várt 3128-as portot használja:

```
# Port to listen on.
#
Port 3128
```

Ezek után egy terminálablakból adjunk ki egy

```
tinyproxy
```

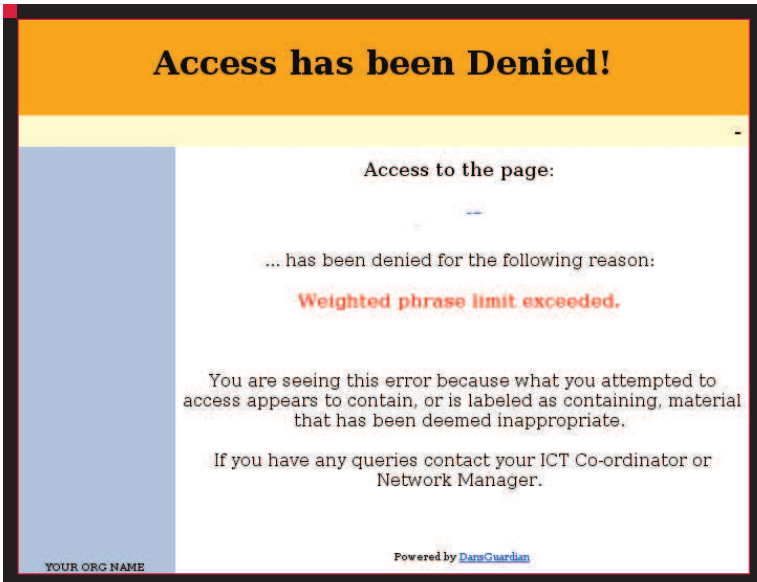
parancsot, vagy – *Debian* és *Ubuntu* alapú disztribúciók esetén – egy

```
sudo /etc/init.d/tinyproxy
➥start
```

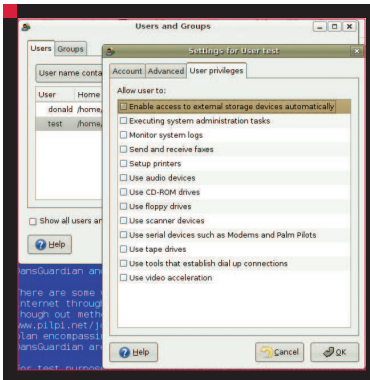
parancsot. Ez elindítja a proxyt, és innentől már csak a böngészőben kell beállítani egy-két dolgot a telepítés befejezéséhez. A fenti folyamat további tanulmányozásához érdemes átolvasni a *DansGuardian* dokumentációjának hivatkozásait (lásd a források közt).

A böngésző beállítása

Az *Ubuntu Linux* (és még néhány más disztribúció) a *Firefox* böngészőt ajánlja alapértelmezettnek, így az alább vázolt lépések is erre vonatkoznak. Nyilván a többi (összemérhető szintű) böngészőben is megvannak az ezzel analóg lehetőségek, amiket megtalálhatunk a megfelelő dokumentációban vagy a weben.
A telepítés ezen utolsó lépcsőfoka ráállítja a böngészőt a 8080-as port használatára, így ez csak a *DansGuardianon* és *Tinyproxyn* keresztül fog tudni adatokat küldeni. *Firefoxban* válasszuk a *Szerkesztés* menü | *Beállítások* almenü | *„Általános"* fül | *„Kapcsolat beállításai"* gombot. Az *1. ábrán* látható az ennek hatására kapott dialógusablak, valamint az is, hogy a „kézi proxybeállítás"-t miként lehet beállítani „localhost" *HTTP-proxy* és „8080"-as *port* értékre. Ez azt feltételezi, hogy a *DansGuardiant* és *Tinyproxyt* minden munkaállomáson használni fogjuk. Ha egy külön szerveren állítjuk be a *DansGuardiant* és *Tinyproxyt*, akkor a *HTTP-proxy* értékét értelemszerűen nem `localhost`-ra, hanem a *DansGuardiant* és *Tinyproxyt* futtató gép nevére vagy *IP*-címére kell állítani.



2. ábra Tipikus DansGuardian „hozzáférés megtagadva” oldal



3. ábra Ubuntu Dapper Drake felhasználói jogosultság-beállító

A böngésző újraindítása után ellenőrizhető, milyen jól működnek a szűrők. Egy-egy új szűrő kipróbálásakor a *2. ábrán* láthatóhoz hasonló, hozzáférést megtagadó képernyőt kell látnunk. Mielőtt továbblépnénk, érdemes körbejárni, milyen gondok adódhatnak az alapértelmezett beállítások miatt. Én pl. gyakran töltök le ***.tar*** és egyéb hasonló „végrehajtható” fájlokat. Az eredeti konfiguráció leállítja ezen fájlok letöltését. Ennek orvoslásához a ***bannedextensionlist.txt*** fájlt kell szerkeszteni; a sor elejére írt # jel által megjegyzésbe lehet tenni azon kiterjesztéseket, amiket át szeretnénk engedni a szűrőnkön.
Az érdeklődőbb olvasóknak azt javaslom, hogy rágják át magukat valamennyi ***DansGuardian*** konfigurációs ***.txt*** fájlon, hogy felelősségteljesen legyen testreszabva a szűrők működése. Nyilván nem lehet elképzelni az összes szituációt, amibe valaha is belefutunk, mégis, ez egy jó alkalom arra, hogy némileg belelássunk eme alkalmazás fantasztikus lehetőségeibe.

Sebezhetőségek

Nincs tökéletes rendszer. Van néhány nyilvánvaló megoldás, amellyel ki lehet játszani a ***DansGuardiant*** és a ***Tinyproxyt***; főleg, ha a felhasználók könnyedén ki tudják kapcsolni a proxyt és a szűrőket. Ha ezt nem akadályozzuk meg, akkor vissza lehet állítani a ***Firefox*** beállítási menüjében a közvetlen internetkapcsolatot, ami elkerüli a ***DansGuardian*** és ***Tinyproxy*** használatát. Innentől kezdve pedig korlátlan internetelérés áll a felhasználók rendelkezésére.
Van néhány módszer arra, hogy biztonságosabbá tegyük a ***DansGuardian*** szűrőit azáltal, hogy minden internetes adatáramlást a 8080-as porton kényszerítünk át. A ***DansGuardian*** webes dokumentációjában található egy hivatkozás, ami elmagyaráz egy remekül kigondolt módszert, amely a ***FireHol*** segítségével ezt a kényszerfeltételt minden internetes adatkapcsolatra érvényesíti (lásd a forrásokat).
A kezdő felhasználók készíthetnek egy egyszerűbb szűrési tervet is. Ez azon alapul, hogy vannak korlátozott jogosultságú felhasználók, akik számára rögzítünk bizonyos böngészőbeállításokat, valamint beállítjuk, hogy a számítógép bekapcsolásakor mindig elinduljanak a proxyszűrők.
Tesztelési célra készítettem egy új felhasználói azonosítót ***Ubuntu Dapper Drake***-et futtató számítógépemen (*3. ábra*). Bizonyos jellemzők megadásával szigorúan leszabályoztam eme felhasználó képességeit, oly módon, hogy ezek a jogosultságok azért bőven használhatóak legyenek bárki számára, aki nem különösebben járatos a számítógépes világban, vagy egyszerűen csak nem kellően megbízható. Az ***update-rc.d*** vagy ***fcconf*** segítségével meg lehet határozni, hogy mely programok induljanak el rendszerbetöltéskor. A magam részéről a ***BUM*** nevű programbetöltés-szervezőt használom a ***DansGuardian*** és a ***Tinyproxy*** elindítására.
Végül le kellett zárnom a ***Firefox*** beállításait. Ez nem olyan nagy feladat, mint amekkorának első hallásra tűnik. Olvastam egy részletes, régi (szerzői jogokkal védett) cikket ***Warren Togamitól*** (lásd a forrásokat) ***„Mozilla beállításlezárási HOGYAN LTSP Linux számára”***, (***„HOWTO Lock Down Mozilla Preferences for LTSP”***) címmel. Én azonban nem szeretnék mélyreható bitbuherálással zűrzavart teremteni, mikor ennek egyszerűbb módja is van. Miután beleástam magam a ➲ ***Mozilla.org*** weboldalának átolvasásába, arra jutottam, hogy elegendő egy ***lockPref*** utasítást beírni a ***Firefox*** konfigurációs fájljába ahhoz, hogy a felhasználók ne tudják megváltoztatni a kapcsolati beállításokat. Szerkeszteni kezdtem tehát a ***/usr/lib/firefox/firefox.cfg*** fájlt, ahogy az az *5. ábrán* is látszik. Az utolsó három sor kényszeríti ki a ***localhost*** kézi proxybeállítását a 8080-as portra.
E fájl elmentése és a ***Firefox*** újraindítása után már nem lehetséges a kapcsolati beállítások alapértelmezettre állítása. Természetesen megfelelő adminisztrátori jogosultság nélkül a felhasználók nem tudják ezeket a beállításokat átírva megkerülni a szűrőket.

Karbantartás

Miután ízlésünknek megfelelően testreszabtuk a szűrőket, fontos tudni, hogy bizonyos beállítások

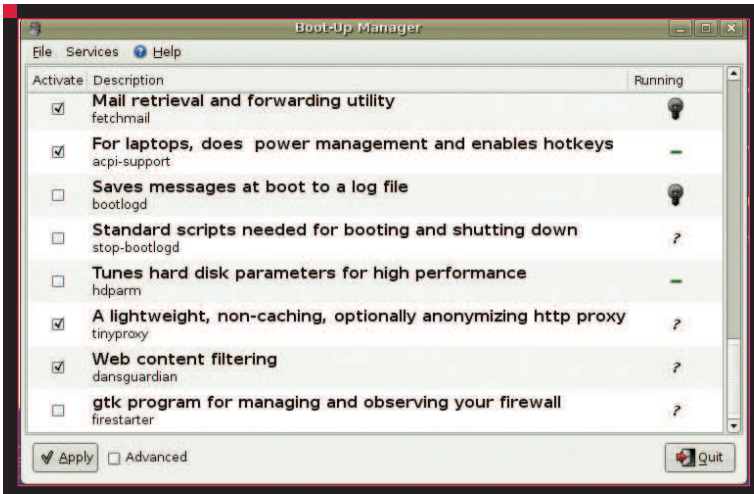
elévülhetnek. Egyes tiltott webhelyek és kifejezések hamarabb, mások később mennek ki a „divatból”. Gyakran új weboldalak jelennek meg, és bizonyos szókapcsolatok a régiek helyébe lépnek. A *DansGuardian* webhelyén van egy *„Extras” („Egyebek”)* hivatkozás, ahol a feketelistára tett oldalakról további információk olvashatók. Ráadásul találhatunk olyan szkripteket is, melyeket áldozatkész felhasználók azért publikáltak, hogy automatizálni lehessen velük a feketelisták készítését és naprakészen tartását.

Egy másik alternatíva az ➲ *URLblacklist.com* használata, mely az új felhasználók számára az első letöltést ingyenesen lehetővé teszi. Ha ez megtetszik, elő lehet fizetni a további remek frissítésekre. Ezeknek az adatoknak a *DansGuardian* számára való átalakítása megtalálható a weben. Érdemes figyelni, hogy a proxy és a szűrők mennyire lassítják a szörfözést, az oldalak betöltődését. Lehetnek helyzetek, amikor a felhasználóknak el kell viselniük bizonyos mértékű teljesítményromlást a *Tinyproxy* használatakor. Saját kísérletezéseim során észleltem némi késést, valamint néhány új tételt a böngésző gyorstárában. A *Firefox* gyorstárának *Ctrl-Shift-Del* gombkombinációval való törlése azonban rögtön orvosolta ezt a problémát. Előfordulhat, hogy szükséges lehet a *Tinyproxy* újraindítása az internetelérés sebességének növeléséhez. Ezek kissé zavaróak, mégis elfogadható kompromisszumot jelenthetnek a hétköznapi munka során.

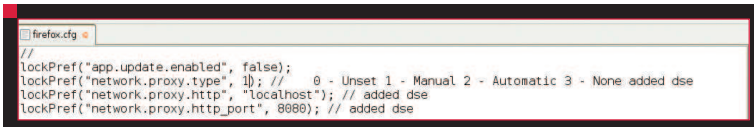
A naplófájlok áttekintése

Mind a *DansGuardian*, mind a *Tinyproxy* készít naplófájlokat, melyeket az adminisztrátornak célszerű időnként átnéznie. A */var/log*-on belül készül egy-egy könyvtár a *DansGuardian* és a *Tinyproxy* számára. Egy egyszerű editorral megnyitva a megfelelő fájlokat érdemes átpásztázni az adatokat, hogy tudjuk, mi történt a számítógépen. Az egymás utáni sorokban tárolt információk és világos megjegyzések teszik egyszerűbbé az események megértését.

A *DansGuardianhoz* készített valaki egy olyan szkriptet is, amivel a keresés és az adatok kijelzése sokkal inkább felhasználóbarát formátumban valósítható meg.



4. ábra A DansGuardian és a Tinyproxy beállítása – bootoláskor induljanak el



5. ábra A Firefox beállításainak lelakatolása – adminisztrátori jogosultság nélkül ezek nem változtathatóak meg

Sajnos a *DansGuardianból* hiányzik az a lehetőség, hogy e-mailben is elküldje a naplófájlokat (*ám ez könnyedén megoldható például a* cron *segítségével – a ford.*). Erős motiváció lehet bizonyos felhasználók viselkedésének kordában tartására az a tudat, hogy egy felelős ember időnként átvizsgálja az internetes tevékenységüket.

Záró gondolatok

Mielőtt döntést hozunk a tartalomszűrés fent vázolt technikai megvalósításáról, fontoljuk meg, milyen igényeket szeretnénk megvalósítani az elkövetkező hónapokban. Ha csak egyetlen számítógépről van szó, és szívesen bütyköl valaki konfigurációs fájlokat, akkor a *DansGuardian* valószínűleg jó választás. Azonban a *SmoothGuardian* is jó vásárnak tűnik a maga 90 dolláros árával. A szoftver felhasználóbarát webes kezelőfelületet és egy egyszerűsített telepítőt is tartalmaz. Mindamellett a *DansGuardian* és *Tinyproxy* beállítása egyáltalán nem mutat túl még a kezdő *Linux*-felhasználók képességein sem, és az ingyenes hozzájutás a legtöbb költségvetésben jól fest. Eme cikk és a megadott hivatkozások segítségével valószínűleg nem lesz probléma a telepítéssel és futtatással. Ha valami akadály fel is bukkanna, a *Google*-n bizonyára könnyen megtaláljuk a megoldást.

A *DansGuardian* honlapján (lásd a forrásokat) hivatkozás látható egy webes tartalomszűrési portálra, valamint egy témába vágó *IRC* csevegőhelyre is. Általánosságban elmondható, hogy a *DansGuardian* és a *Tinyproxy* a szabad szoftveres világ élharcosai – segítik az átállást a *Microsoft Windows* környezetből. Megítélésem szerint a rugalmas szűrőfeltételek megadásának lehetősége és a pehelysúlyú proxyterhelés jó választássá teszi e két program kombinációját kisebb hálózatokon.

Linux Journal 2006., 151. szám

Donald Emmack vezetőségi tag a The IntelliGents & Co. cégnél. Kiterjedt munkát végez íróként és üzleti tanácsadóként Észak-Amerikában. Elérhető a donald@theintelligents.com e-mail címen vagy a 2 méteres amatőr rádiócsatornákon.

A cikk forrásai

➲ www.linuxjournal.com/article/9291

Az Apache biztonságos beállítása

Egy neves IT cég hirdetése szerint naponta többen auditálják hálózatunkat – sajnos nem mindenki a mi megbízásunkból. A biztonság nem fölösleges luxus, hanem az a nélkülözhetetlen kiinduló állapot, ami nélkül ne is reméljünk jövedelmező online jelenlétet.

Az ***Apache*** része a legtöbb ***Linux*** terjesztésnek, így ma már mindenkinek lehet web kiszolgálója. De biztonságos web kiszolgálót nem is olyan egyszerű építeni. Írásomban egy (viszonylag) biztonságos ***LAMP*** (=***Linux, Apache, MySQL*** és ***PHP***) konfiguráció kialakítását mutatom be.
Már rögtön az elején jó szem előtt tartani, hogy a biztonság mindig csak relatív lehet, az abszolút biztonság, mint olyan nem létezik. Tisztában kell lennünk azzal, hogy ha egy támadónak megfelelő erőforrása (idő, pénz, számítási teljesítmény, …) van, akkor be tudja venni a várat.
Egy átlagos betörés jellemzően információ gyűjtéssel kezdődik, a támadó apró, lényegtelennek tűnő mozaikdarabokból összeállítja a cél profilját, kiválasztja a behatolási pontot, majd azon keresztül bejön, elveszi, amit akar, törli a nyomokat, majd angolosan távozik.
A biztonságos kiszolgálónkat ezért úgy alakítjuk ki, hogy a lehető legkevesebb információt szivárogtassuk ki, és a lehető legkevesebb dologhoz engedünk hozzáférést a világnak.
Bruce Schneier elhíresült mondása szerint a biztonság nem egy termék, hanem egy folyamat. A biztonság számtalan láncszemből áll össze, és talán nem árt hangsúlyozni azt a közhelyet, hogy olyan erős a lánc, mint a leggyengébb láncszem.
Sok éve már, hogy egy ifjú titán magabiztosan állította, hogy az ő gépén ***Mandrake*** fut, úgyhogy jobb, ha előre feladja bárki. Én ennél egy kissé óvatosabb vagyok, és azt tanácsolom, ne ringassuk magunkat abban a hamis illúzióban, hogy csak azért, mert ***Linux*** és ***Apache*** fut a gépünkön, nem érhet bennünket egy idegen nyelvű felirat kellemetlen híre egy borús reggelen.
Csábító a lehetőség, hogy telepítsük a ***Linux*** terjesztésünkben található ***Apache*** és ***PHP*** csomagot. Azonban ezek jellemzően (szinte) minden funkcionalitást tartalmaznak, ami nekünk nem jó. ***Wietse Venema*** írta valahol, hogy a ***Postfix*** kódjának minősége kb. 1 hiba (***bug***) 1000 soronként. Ha ezt átlagnak tekintjük, könnyen belátható: minél több (felesleges) funkció van egy programban, statisztikailag annál több (kihasználható) hiba lehet benne. Ezért csomagból csak a ***MySQL*** programot telepítsük.
Kedvenc ***Linux*** terjesztésünk telepítése során alakítsunk ki megfelelő méretű ***/***, ***/usr***, ***/var***, ***/tmp***, ***/home*** partíciókat, és készítsünk egy nagy ***/opt*** partíciót, ez utóbbi fogja tárolni a web kiszolgálónkkal kapcsolatos összes állományt.

Készítsünk ketrecet!

Miként a veszélyes állatokat ketrecben tartják, úgy a veszélyes programot is célszerű bezárni, hogy ha a támadónak sikerül – pl. egy programhiba segítségével – átvenni felette az ellenőrzést, ne okozhasson (még) nagyobb bajt, ne férhessen hozzá a gép más részeihez. A ***chroot()*** rendszerhívás ill. a ***chroot*** parancs hatására az adott program az argumentumban megadott elérési utat látja a továbbiakban gyökérkönyvtárként, és csak az az alatt található dolgokhoz férhet hozzá.
Készítsük el első lépésben a „ketrecet” a ***/opt/jail*** könyvtár alatt! Az *1. Listában* szereplő ***makejail.sh*** héjprogram segít ebben, argumentumként a kezdőkönyvtárat kell megadni, például az

```
sh makejail.sh /opt/jail
```

parancs a ***/opt/jail*** alatt hozza létre a szükséges környezetet. Az Olvasó gépén esetleg más állományokra is szükség lehet.
Ez a héjprogram létrehoz egy minimális környezetet, amely az ***Apache*** futtatásához feltétlen szükséges. Bizonyára feltűnt az Olvasónak, hogy a ***/etc*** könyvtárban nem szerepel jelszavakat tartalmazó állomány (például ***/etc/shadow***), és a ***/etc/passwd*** állományban is csak a nélkülözhetetlen felhasználók szerepelnek. Hogy még keményebb legyen a dió, a ***/etc*** könyvárban mindenre bekapcsoljuk az úgynevezett ***immutable*** attribútumot (`+i`), amely megakadályozza, hogy az itt lévő állományok megváltozhassanak. Ezután a program bemásol néhány könyvtárat (***library***), amelyek az ***Apache*** működéséhez szükségesek.
Nagyon fontos, hogy a ***chroot*** könyvtárba csak a valóban nélkülözhetetlen állományokat helyezzük el. Ne feledjük, minél kevesebb dolgot talál itt a támadónk, annál kevesebb kárt tud okozni. Azt is szeretném itt megjegyezni, hogy a ***root*** felhasználó ill. az ő jogaival futó programok kitörhetnek a ketrecből, ezért hacsak különösen jó okunk nincs rá, semmiképpen ne helyezzünk itt el ***setuid/setgid*** programokat.

1. Lista Az mkjail.sh héjprogram

```
#!/bin/sh
##
## mkjail.sh

if [ $# -ne 1 ]; then echo
➥"usage: $0 <jail directory>";
➥exit 1; fi

if [ ! -d $1 ]; then mkdir -p
➥$1; fi

cd $1 || exit 2;

mkdir dev etc lib usr www

# néhány device is szükséges
mknod dev/null c 1 3
chmod 666 dev/null
mknod dev/hwrandom c 10 183

cp /etc/resolv.conf etc
cp /etc/localtime etc

# csak a legszükségesebb
➥információt tesszük be
➥a kalitkába
echo "root::0:" > etc/group
echo "nobody::98:" >> etc/group
echo "nogroup::99:" >> etc/
➥group
echo "httpd::7002:" >> etc/
➥group

echo "root:x:0:0::/:/bin/sh" >
➥etc/passwd
echo "nobody:x:99:99:nobody:/
➥oblivion:/bin/false" >> etc/
➥passwd
echo "httpd:x:7002:7002::/
➥oblivion:/bin/false" >> etc/
➥passwd

echo "127.0.0.1 localhost" >
➥etc/hosts

# immutable attribútum
# bekapcsolása
chattr +i etc/*

cp /lib/ld-linux.so.2 lib
cp /lib/libc.so.6 lib
cp /lib/libcrypt.so.1 lib
cp /lib/libdl.so.2 lib
cp /lib/libm.so.6 lib
cp /lib/libnsl.so.1 lib
cp /lib/libnss_compat.so.2 lib
cp /lib/libnss_files.so.2 lib
cp /lib/libresolv.so.2 lib
cp /lib/librt.so.1 lib
cp /lib/libtermcap.so.2 lib

mkdir usr/lib usr/local usr/
➥local/bin usr/local/etc

mkdir www/log www/log/.tmp www/
➥data www/phpsessions www/
➥data/www.fiktivceg.hu
chmod 700 www/phpsessions
➥www/log
chown httpd:httpd www/
➥phpsessions
```

Telepítsük az Apache kiszolgálót

Hogy minél kevesebb állományt kelljen a kalitkába másolni, az *Apache*-ot statikusan fordítjuk le. Töltsük le az *Apache* 1.3.37 verzióját a ➲ *http://httpd.apache.org/download.cgi* címről, a hozzá tartozó *mod_ssl* nevű *SSL* foltot ➲ http://www.modssl.org/source/mod_ssl-2.8.28-1.3.37.tar.gz, webhelyről továbbá szükségünk lesz még a *PHP 4.4.4*-es verziójára a ➲ *http://www.php.net/downloads.php* oldalról. A *PHP* számára *MySQL* támogatást is biztosítunk, a példában feltételezzük, hogy a *mysql-standard-4.1.21-pc-linux-gnu-i686.tar.gz* állomány a /usr/local/mysql könyvtár alá lett telepítve.

Első lépésben csomagoljuk ki a letöltött állományokat:

```
tar zxvf mod_ssl-2.8.28-
➥1.3.37.tar.gz
tar zxvf apache_1.3.37.tar.gz
tar jxvf php-4.4.4.tar.bz2
```

Aztán alkalmazzuk az *SSL* foltot (*mod_ssl*):

```
cd mod_ssl-2.8.28-1.3.37
./configure --with-
➥apache=../apache_1.3.37
```

A *2. Listában* látható utasításokkal konfiguráljuk az *Apache*-ot, definiáljuk a *MySQL* telepítésének könyvtárát, hozzáadunk néhány extra könyvtárat (*library*), ill. beállítjuk a statikus fordítást. Most még ne fordítsuk le az *Apache*-ot, hanem konfiguráljuk és fordítsuk le a *PHP* modult, ahogyan az a *3. Listában* szerepel. Természetesen egyéb kapcsolókat is használhatunk, ha extra funkciókra is szükségünk van. Másoljuk át a szükséges állományokat az *Apache* megfelelő könyvtárába:

```
cd ../apache-1.3.37
cp ../php-4.4.4/sapi/apache/
```

2. Lista Az Apache konfigurációja

```
cd ../apache_1.3.37
CFLAGS="-static" \
LDFLAGS="-L/usr/local/mysql/
➥lib" \
LIBS="-lcrypt -lcrypto -lssl
➥-lmysqlclient_r -lz
➥-lresolv -lpthread -lm -lgd
➥-lstdc++" \
INCLUDES="-I/usr/local/mysql/
➥include" \
./configure \
-prefix=/usr/local/
➥apache-1.3.37 \
-enable-module=include \
-disable-module=so \
-disable-module=userdir \
-disable-module=info \
-enable-module=status \
-enable-module=ssl \
-activate-module=src/modules/
➥php4/libphp4.a
```

3. Lista Beállítjuk a PHP-t

```
cd ../php-4.4.4
./configure \
--prefix=/usr/local/php \
--with-zlib \
--with-openssl \
--with-config-file-path=/usr/
➥local/etc \
--with-apache=../
➥apache_1.3.37 \
--with-gd \
--with-mysql=/usr/local/mysql
make
```

```
➥mod_php4.* src/modules/php4
cp ../php-4.4.4/sapi/apache/
➥libphp4.module
src/modules/php4
cp ../php-4.4.4/sapi/apache/
➥apMakefile.libdir src/
➥modules/php4/Makefile.libdir
cp ../php-4.4.4/sapi/apache/
➥apMakefile.tmpl
src/modules/php4/Makefile.tmpl
cp ../php-4.4.4/.libs/libphp4.a
➥src/modules/php4/libmodphp4.a
```

Futtassuk újra az ***Apache*** konfiguráló programját, hogy az ***src/modules/php4*** könyvtárban is létrejöjjön a ***Makefile*** állomány, majd fordítsuk le és telepítsük. A figyelmes olvasónak feltűnhet, hogy a ***/opt/jail*** mint gyökérkönyvtár (/) alá telepítjük az alkalmazást:

```
sh config.status
make
su -c 'make install
➥root=/opt/jail'
```

A titkosítás a barátunk, de nem old meg minden problémát

*„**Ez a webhely biztonságos, mert SSL tanúsítvánnyal rendelkezik**”*. Egy időben több web oldalon is láttam ezt a némileg megtévesztő szöveget, amely azt az érzetet kelti a látogatóban, hogy itt nyugodtan megadhatja az adatait, mert az jó kezekben lesz, ott semmi baj nem történhet. A titkosított kapcsolat jó dolog, főleg ha éppen a személyes adatainkat adjuk meg egy vásárlás során, esetleg a bankszámlánkkal kapcsolatos ügyeket intézzük a fotelból. A valóságban azonban a kriptográfia (például ***SSL***) alkalmazása nem tesz egyetlen kiszolgálót sem biztonságosabbá. A kis lakat a böngészőben nem jelent se többet, se kevesebbet, minthogy a látogató gépén futó böngésző és a távoli web kiszolgáló között egy titkosított csatorna jött létre, és az ebben haladó adatok biztonságban vannak egy esetleges hallgatózó harmadik féllel szemben – hacsak nem az egyik kormány szuperszámítógépe érdeklődik valamelyik tranzakciónk iránt. Az elkészített ***Apache***-unk támogatja az ***SSL*** titkosítást, már csak egy kulcsot és egy tanúsítványt (***certificate***) kell készítenünk. A *4. Listában* az ehhez szükséges parancsok, ill. azok képernyőre írt üzenete látható. (A parancsok előtt a *$* prompt áll, amit nem kell begépelnünk) A ***CSR*** állomány készítésekor ügyeljünk arra, hogy a *„**challenge password**”* kérdésnél ne adjunk meg jelszót, ellenkező esetben az ***Apache*** (minden újra)indításakor be kell azt gépelnünk.

4. Lista SSL kulcs és tanúsítvány készítése

```
$openssl genrsa 2048 > /opt/
➥jail/usr/local/etc/
➥www.fiktivceg.hu.key

Generating RSA private key,
➥2048 bit long modulus
......+++
.............................
.............................
...................+++
e is 65537 (0x10001)


$openssl req -new -key /opt/
➥jail/usr/local/etc/
➥www.fiktivceg.hu.key > 1.csr

Country Name (2 letter code)
➥[AU]:HU
State or Province Name (full
➥name) [Some-State]:Hungary
Locality Name (eg, city)
➥[]:Budapest
Organization Name (eg, company)
➥[Internet Widgits Pty Ltd]:
➥Fiktiv Ceg
Organizational Unit Name (eg,
➥section) []:
Common Name (eg, YOUR name)
➥[]:www.fiktivceg.hu
Email Address
➥[]:info@fiktivceg.hu

Please enter the following
➥'extra' attributes
to be sent with your
➥certificate request
A challenge password []:
An optional company name []:


$openssl x509 -in 1.csr -out
➥/opt/jail/usr/local/etc/
➥www.fiktivceg.hu.cert -req
➥-signkey /opt/jail/usr/local/
➥etc/www.fiktivceg.hu.key
Signature ok
subject=/C=HU/ST=Hungary/
➥L=Budapest/O=Fiktiv Ceg/
➥CN=www.fiktivceg.hu/
➥emailAddress=info@fiktivceg.hu
Getting Private key

$chmod 400 /opt/jail/usr/local/
➥etc/www.fiktivceg.hu.key
$chmod 644 /opt/jail/usr/local/
➥etc/www.fiktivceg.hu.cert
```

A példában mi magunk írjuk alá a tanúsítványt, ami arra ugyan jó, hogy titkosított kommunikációt folytassunk egy belső hálózaton, de mindenképpen ajánlott egy hatósággal (***CA***) – például ***NetLock***, ***VeriSign***, ***Thawte*** – aláíratni a tanúsítványunkat, ha az Interneten akarunk megjelenni, ez segít elkerülni a közbenső ember (***man-in-the-middle***) támadásokat. Javaslom, hogy az ***Apache*** konfigurációs állományát (***httpd.conf***) mozgassuk át a ***/opt/jail/usr/local/etc*** könyvtárba, az indító héjprogramot (***apachectl***) pedig az ***/opt/jail/usr/local/bin*** alá, továbbá készítsünk egy szimbolikus linket az

5. Lista Egy apró korrekció az apachectl állományban

```
PIDFILE=/usr/local/apache/
➥logs/httpsd.pid
HTTPD="/usr/local/apache/bin/
➥httpsd -f
/usr/local/etc/httpsd.conf"
```

```
ln -sf /usr/local/apache-1.3.37
➥/usr/local/apache
```

utasítással. Ezek a kényelmünket szolgálják, verzió frissítésnél elég csak a szimbolikus linket módosítani. Az *5. Listában* látható módon módosítsuk az `apachectl` programban az alábbi két változót, hogy a megfelelő állományokra hivatkozzon.

6. Lista A httpd.conf állomány

```
# globalis rész
#

# httpd felhasználóként fog
# futni az Apache
User httpd
Group httpd

ServerName www.fiktivceg.hu
ServerAdmin info@fiktivceg.hu

# nem kérünk névfeloldást
HostnameLookups Off

LogLevel warn

ErrorLog /www/log/.tmp/
➥error.log

LogFormat "%h %v %l %u %t
➥\"%r\" %s %b %{Referer}i
➥\"%{User-agent}i\"" myformat
CustomLog "|/usr/local/apache/
➥bin/rotatelogs /www/log/.tmp/
➥access.log 10800" myformat

# minimális adatot adunk
# a kiszolgálóról
# ServerSignature Off
ServerTokens Minimal

<IfModule mod_alias.c>

    ....

# szigorú korlátozások
# a CGI-BIN könyvtáron
    <Directory "/www/cgi-bin">
        AllowOverride
        ➥AuthConfig Limit
        Options None
        Order allow,deny
        Allow from all
    </Directory>

</IfModule>


<IfModule mod_mime.c>
    # engedélyezzük a php
    # motort
    AddType application/
    ➥x-httpd-php .php
</IfModule>


# mod_ssl konfiguráció

SSLCertificateKeyFile /usr/
➥local/etc/www.fiktivceg.hu.key
SSLCertificateFile /usr/local/
➥etc/www.fiktivceg.hu.cert

AddType application/x-x509-
➥ca-cert .crt
AddType application/x-pkcs7-
➥crl    .crl

SSLPassPhraseDialog  builtin

SSLSessionCache        dbm:/
➥www/ssl/ssl_scache
SSLSessionCacheTimeout  300

SSLMutex  file:/www/ssl/
➥ssl_mutex

SSLRandomSeed startup builtin
SSLRandomSeed connect builtin

SSLLog none
SSLLogLevel warn

SSLCipherSuite ALL:!ADH:!
➥EXPORT56:RC4+RSA:+HIGH:
➥+MEDIUM:+LOW:+SSLv2:+EXP:
➥+eNULL
SetEnvIf User-Agent ".*MSIE.*"
➥nokeepalive ssl-unclean
➥-shutdown downgrade-1.0
➥force-response-1.0

<Directory "/www/cgi-bin">
    SSLOptions +StdEnvVars
    ➥+CompatEnvVars
</Directory>


#
# most csak egy virtuális
# kiszolgálót definiálunk

NameVirtualHost 1.2.3.4:80

<VirtualHost 1.2.3.4:80>
    ServerName www.fiktivceg.hu
    SSLEngine off

    DocumentRoot /www/data/
➥www.fiktivceg.hu

    <Directory /www/data/
➥www.fiktivceg.hu>
        Options Includes Indexes
        AllowOverride AuthConfig
        ➥Limit
        Order allow,deny
        Allow from all
    </Directory>

    php_admin_value
    ➥open_basedir
"/www/data/www.fiktivceg.hu/"
    php_admin_value doc_root
    ➥"/www/data/
    ➥www.fiktivceg.hu/"
    php_admin_flag
    ➥display_errors On
    php_admin_value
    ➥sendmail_path
"/usr/sbin/sendmail -t -i -f
➥info@fiktivceg.hu"

    # engedélyezzük a fájl
    ➥feltöltést
    php_admin_flag file_uploads
    ➥On
    php_admin_value
    ➥upload_tmp_dir
/www/data/www.fiktivceg.hu/
➥upload
    php_admin_value
    ➥upload_max_filesize 1000k

    ErrorDocument 404 http://
    ➥www.fiktivceg.hu/
    ➥error404.php

</VirtualHost>

NameVirtualHost 1.2.3.4:443

<VirtualHost 1.2.3.4:443>
    ServerName www.fiktivceg.hu
    SSLEngine on

    ....

</VirtualHost>
```

Az Apache és PHP beállítása

Indítás előtt szerkesszük az *Apache* konfigurációs állományát, a *6. Listában* szereplő változók kivételével a többi maradhat az alapértelmezett értékkel.

A *PHP* egy rendkívül sokoldalú programnyelv, amelynek a használata sajnos azzal a (biztonsági szempontból)

7. Lista Bekapcsoljuk az úgynevezett safe mode funkciót

```
safe_mode = On
safe_mode_gid = On

# nem kell mindenkinek tudni,
# hogy milyen verziójú PHP
# fut a gépünkön
expose_php = Off
```

rendkívül kellemetlen mellékhatással is jár, hogy a felhasználók gyakorlatilag olyan ***PHP*** nyelven megírt programokat futtathatnak, amilyet csak akarnak. Néhány óvintézkedést azonban megtehetünk, hogy az ebből eredő esetleges károkat minimalizáljuk. Másoljuk a ***PHP*** forrás könyvtárából a php.ini-recommended nevű állományt a ketrecbe:

```
cp php.ini-recommended /opt/
➥jail/usr/local/etc/php.ini;
➥chmod 600 /opt/jail/usr/
➥local/etc/php.ini
```

Itt szeretném felhívni a kedves Olvasó figyelmét a ***php.ini*** elején található megjegyzésekre, amelyek a javasolt beállításokat tartalmazzák, közülük többnek van biztonsági vonatkozása. Mindenki kedvére szabhatja testre a ***PHP*** konfigurációs állományát, a ***http.conf*** állományban minden virtuális kiszolgáló (***virtualhost***) esetében egyedi értékeket lehet beállítani.
A ***safe_mode*** – amelyet a *7. Listában* kapcsolunk be – több ***PHP*** függvényre is hatással van, korlátozza azok működését, például a ***fopen()*** függvény nem hajlandó megnyitni azokat az állományokat, amelyek kívül esnek az ***open_dir*** változó által megadott könyvtáron. Ez utóbbi igen hasznos funkció, így elkerülhetjük, hogy illetéktelenek megnézzék például a jelszó állományunkat (***/etc/passwd***).
A ***php.ini***-ben állítsuk be a

```
session.save_path = /www/
➥phpsessions
```

sort, hogy a ***PHP*** itt tárolja el az egyes kapcsolatokhoz ***(session)*** tartozó információkat.

8. Lista A modsecurity modul beállítása a httpd.conf állományban

```
<IfModule mod_security.c>

    # bekapcsoljuk a modult
    SecFilterEngine On

    # probléma esetén 403-as
    # HTTP státusz kódot ad
    # vissza, és elutasítja
    # a kérést
    SecFilterDefaultAction
    ➥"deny,log,status:403"

    # néhány józan
    # alapértelmezett beállítás
    SecFilterScanPOST On
    SecFilterCheckURLEncoding
    ➥On
    SecFilterCheckUnicode
    ➥Encoding Off

    # a 0 byte érték
    # kivételével mindent
    # elfogad
    SecFilterForceByteRange 1
    ➥255

    # álcázza a kiszolgáló
    # típusát és verzióját
    # SecServerSignature
    # "Microsoft-IIS/5.0"

    # az ideiglenes
    # állományokat itt tárolja
    SecUploadDir /www/tmp
    SecUploadKeepFiles Off

    # csak a lényeges adatokat
    # naplózza
    SecAuditEngine RelevantOnly
    SecAuditLog /www/log/.tmp/
    ➥modsec_audit.log

    # nem akarunk túl részletes
    # naplózást
    SecFilterDebugLevel 0
    SecFilterDebugLog /www/log/
    ➥.tmp/modsec_debug.log

    # csak azokat a kéréseket
    # fogadjuk el, amelyeket
    # kezelni is tudunk
    SecFilterSelective
    ➥REQUEST_METHOD
    ➥"!^(GET|HEAD)$" chain
    SecFilterSelective
    ➥HTTP_Content-Type
    ➥"!(^application/x-www
    ➥-form-urlencoded$|
    ➥^multipart/form-data;)"

    # GET és HEAD kéréseket nem
    # fogadunk el törzzsel
    SecFilterSelective
    ➥REQUEST_METHOD
    ➥"^(GET|HEAD)$" chain
    SecFilterSelective
    ➥HTTP_Content-Length "!^$"

    # minden POST kérésnél meg
    # kell adni a hosszát is
    SecFilterSelective
    ➥REQUEST_METHOD "^POST$"
    ➥chain
    SecFilterSelective
    ➥HTTP_Content-Length "^$"

    # csak ismert átviteli
    # kódolást fogadunk el
    SecFilterSelective
    ➥HTTP_Transfer-Encoding
    ➥"!^$"

</IfModule>
```

A `display_errors` direktíva bekapcsolását nem javasolja a ***PHP*** kézikönyve, mert az esetleges hibaüzenetek információt adhatnak a támadónak. Azért javaslom mégis, mert ezek a hibák elcsúfítják a honlapunkat, így rákényszerítik a web fejlesztőket, hogy olyan kódot írjanak, ami nem eredményez hibákat.
A `sendmail_path` változó levelezés esetén hasznos, például űrlap kitöltésének visszaigazolásakor, az ***SMTP*** kapcsolat ***MAIL FROM:*** paraméterét rögzíti fix értékre, így nyomon lehet követni, hogy az egyes leveleket melyik virtuális kiszolgáló küldte el. Az is hasznos lehet, ha letiltjuk a `phpinfo()` függvényt, amely túl sok információt tud, még azt is megmutatja, hogy a ***PHP***-t fordító felhasználónak mi volt a `$PATH` változója. Egy adott függvényt a `disable_functions` paraméternél felsorolva tudunk letiltani, például:

9. Lista Valaki bináris kérést küldött

```
==48835b61==================
==========
Request: www.fiktivceg.hu
64.56.74.94 - - [27/Dec/
➥2006:05:11:49 +0100] "\x04\
➥x01" 403 0 "-" "-" - "-"
--------------------

mod_security-action: 403
mod_security-message: Access
➥denied with code 403.
➥Pattern match
➥"!(^application/x-www
➥-form-urlencoded$|
➥^multipart/form-data;)" at
➥HEADER("Content-Type")
➥[severity "EMERGENCY"]

-48835b61-
```

```
disable_functions = phpinfo,
➥shell_exec, system
```

Egy visszatérő probléma a `register_globals` *PHP* változó használata. Noha már számos verzió óta ki van kapcsolva alapállapotban (nem véletlenül!), mégis számtalan web fejlesztő és alkalmazás követeli, hogy kapcsoljuk be. Ahelyett, hogy követnék a *PHP* újabb lehetőségeit, és hozzáigazítanák a programjaikat.
Itt nem foglalkozunk azzal, hogy több kódoló is létezik, amelyekkel az olvasható *PHP* kódból egy – akár titkosított – futtatható ***bytekódot*** lehet készíteni. Ez amellett, hogy megvédi a *PHP* kódunkat az illetéktelen szemektől, azzal az előnnyel is jár, hogy gyorsabban fog futni a *PHP* programunk. Azonban ez sem csodaszer: a rossz és hibás kód ellen ez sem véd meg.
Ha idáig eljutottunk, adjuk ki ***root*** felhasználóként a

```
chroot /opt/jail apachectl
➥configtest
```

parancsot. Ha minden rendben, akkor indítsuk el a ***chroot /opt/jail apachectl start*** utasítással. Gratulálok, elkészült egy relatíve biztonságos web kiszolgáló!

Egy biztonsági modul

Bár hosszú út áll mögöttünk, mégsem merítettük ki az összes lehetőséget, amivel a web kiszolgálónkat biztonságosabbá tehetjük. Időről időre bejárja az Internetet valamilyen egzotikus nevű féreg (***worm***), amelyet a megfertőzött gépek láncreakciószerűen terjesztenek tovább, hogy más kiszolgálókat is megfertőzzenek.
A ***modsecurity*** nevű ***Apache*** modul ➲ ***http://www.modsecurity.org/*** segítségével (amely elérhető az 1.3.x, 2.0.x és 2.2.x verziókhoz is) azonban bizonyos támadásokat megakadályozhatunk. A ***modsecurity*** modul segítségével bizonyos kérésekre 'hozzáférés megtagadva' (403) választ adhatunk. A modul az időponton és a kliens ***IP***-címén túl naplózza magát a problémás kérés adatait, a ***modsecurity*** válaszát és azt is, hogy az adott kérés melyik szabályon akadt fenn. A javasolt konfigurációs részlet a *8. Listában* látható.
Az ***Apache 2.x*** verziójához a ***modsecurity 2.x*** változata használható, amely jobb reguláris kifejezés támogatással és több előre definiált szabállyal rendelkezik, mint az 1.x.
Egy tipikus naplóbejegyzés a *9. Listában* látható.

És még mindig nincs vége!

A téma összetettsége miatt csak utalok néhány dologra, amelyek nélkül nem képzelhető el biztonságos web (tulajdonképpen semmilyen) kiszolgáló. Az első lépés a fizikai biztonság megteremtése, csak a jogosult személyek férhessenek a gép közelébe. A következő lépés az operációs rendszer telepítése. Fontos, hogy csak azokat a csomagokat telepítsük, amelyekre valóban szükségünk lesz. Például fordító (***gcc, make***) biztosan nem kell, inkább egy másik gépen fordítsuk le az ***Apache*** kiszolgálót, ott készítsünk belőle csomagot, és azt vigyük át a biztonságos kiszolgálónkra.
Nem lehet eleget hangsúlyozni, hogy minden biztonsági frissítést azonnal telepíteni kell. Minden terjesztés esetén jó, ha megerősítjük az operációs rendszert, pl. szigorítjuk némely konfigurációs állományhoz való hozzáférést (például ***/etc/lilo.conf***), töröljük a szükségtelen binárisokról a ***setuid/setgid*** jogosultságot (például ***/bin/mount***), eltávolítjuk a szükségtelen felhasználói fiókokat (például ***news, uucp, games***).
Ne feledjük az arany szabályt: minél kevesebb információt mutatunk meg a gépünkről, annak konfigurációjáról, minél kevesebb dologhoz férhetnek hozzá, annál kisebb az esélye egy betörésnek.
A gépen az összes felhasználói fióknak, és különösen a ***root*** felhasználónak erős jelszóval kell rendelkeznie. Itt nem bonyolódom bele például az egyszer használatos jelszavakba (***OTP***) vagy a biometrikus azonosítókba. Sokat segíthet egy körültekintően kialakított kötelező hozzáférés szabályozó (***MAC***) rendszer is. A fizikai biztonság része még az érzékeny adatokat tartalmazó mentésekhez való hozzáférés szabályozása, jó ha páncélszekrényben tartjuk azokat.
Szintén nem esett szó arról sem, hogy hiába egy biztonságosan kialakított kiszolgáló, ha a web fejlesztők nem kellő körültekintéssel írják meg az alkalmazásaikat. Találkoztam egy olyan kiszolgálóval, amelyiken egy sérülékeny verziójú ***phpBB*** fórum futott. Egy kreatív látogató módosította az alkalmazás konfigurációját tartalmazó ***SQL*** táblát, és egy furcsa dizájnt állított be. Magát a gépet ugyan nem törte fel, de pont elég kárt okozott azzal, hogy megváltoztatta a nyitó oldalt. Sokszor nem is szükséges teljes ellenőrzést szerezni a célpont felett, az pont elég lehet a vállalkozásunk csődjéhez, ha bizalmas üzleti információkat szerez meg a konkurencia. Amihez az is elég lehet, ha le tud másolni egy adatbázist.
Bár nem a legújabb ***LAMP*** verziókat használtam ebben a példában, de ezek az elvek a ***PHP 5.x*** ill. az ***Apache 2.x*** verziókkal is használhatóak. Sok sikert kívánok minden Olvasónak a saját biztonságos kiszolgálójához!

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

Személyi tűzfal használata Linux munkaállomáson (1. rész)

Tűzfalak használata Internetes környezetben ma már nélkülözhetetlen feltétele a biztonságos kommunikációnak. Ha intézményi hálózatról érjük el a „Net”-et, akkor a hálózat üzemeltetője minden bizonnyal megoldja a problémát helyettünk, kérés nélkül is gondoskodik a biztonságunkról, még ha ez sokszor kényelmetlenséget is okoz.

Otthoni munkaállomás esetén azonban nincs mentség, a biztonságos környezetet magunknak kell felépíteni. Első és legkézenfekvőbb intézkedés a magára a munkaállomásra telepített személyi tűzfal használata. Ha abban a kiváltságos helyzetben vagyunk, hogy otthoni gépünkön Linux operációs rendszert használhatunk, akkor a tűzfalért nem is kell messzire menni, be van építve a rendszermagba.
A cikk célja tehát ennek megfelelően:

- A tűzfalak fajtáinak rövid ismertetése
- A ***netfilter*** működésének bemutatása
- Az ***iptables*** használatának alapszintű, de részletes ismertetése sok példán keresztül
- Egyszerű otthoni tűzfal konfiguráció bemutatása

A tűzfalak típusai

Amint az zsenge ifjúkorunk óta valamennyiünk számára jól ismert, tehenekből három félét különböztet meg a tudomány, nevezetesen feketét, fehéret és tarkát (lásd még ***Egyszer volt egy Mehemed***). Így a tűzfaltechnikában kevéssé jártas olvasót sem érheti meglepetésként a tény, hogy tűzfal típusból is többfélét találhat. Mielőtt azonban az rövid bemutatásukra rátérnénk, egyáltalán mik azok a tűzfalak? A tűzfal olyan eszköz, mely az egymással kommunikáló végpontok (pl. felhasználói munkaállomás és távoli webszerver) között helyezkedik el, s bizonyos előre beállított szabályoknak megfelelően engedélyezi (átengedi) vagy tiltja (megszakítja) a forgalmat.
A hálózati kommunikáció legkisebb egységei az adatcsomagok (***IP*** csomagok), melyek fejrésze tartalmazza a csomag kézbesítéséhez, kezeléséhez szükséges kísérő információkat (például forrás cím, cél cím, portok, protokollok azonosítói, stb.), az adatmező rész pedig a ténylegesen átvinni kívánt felhasználói adatokat szállítja.
Az összetartozó adatcsomagok összefüggő adatfolyamot, kommunikációs csatornát alakítanak ki.
A legegyszerűbb tűzfal, az úgynevezett ***csomagszűrő***, mely adatcsomag szinten foglalkozik a rajta keresztül áramló információval, nem vizsgálja, hogy a csomag milyen kommunikáció (például ***TCP*** kapcsolat) része. A csomagszűrő tehát minden adatcsomag sorsáról külön hoz döntést, a megelőző forgalomtól függetlenül. Ennek megfelelően a szűrési szabályok is csak az aktuális csomag fejrészéből kiolvasható információkra hivatkozhatnak. Megadhatjuk például hogy mely ***IP*** címekre/címekről engedélyezzük a csomagok továbbítását, milyen protokollt engedünk át, illetve milyen portok elérését tesszük lehetővé a tűzfal két oldalán, stb. A csomagszűrő tűzfalak csak nagyon egyszerű elválasztást képesek megvalósítani, hatékony védelemre nem alkalmasak.

Fejlettebb megoldás jelentenek az úgynevezett ***állapotfigyelő tűzfalak*** (vagy állapotfigyelő csomagszűrők). Ezek jórészt szintén csak a csomagok fejrészében található információkat vizsgálják, viszont nyilvántartják, hogy az aktuális csomag melyik élő kommunikációs kapcsolat (például ***TCP*** csatorna, ***ICMP*** üzenetváltás, ***DNS*** lekérdezés, stb.) része, s amikor a csomag sorsáról döntenek (továbbítják/eldobják), a csomag kapcsolaton belüli szerepét is képesek figyelembe venni. Állapotfigyelő tűzfallal pl. megvalósítható az alábbi szűrési feltétel is, mely egyszerű csomagszűrővel nem megoldható:

- Kimenő ***ping (icmp-echo-request)*** engedélyezése (kiengedése) tetszőleges külső IP cím (hálózati eszköz) felé
- A kérésre érkező „válasz ***ping***” ***(icmp-echo-reply)*** beengedése

A „válasz ping”-et tehát csak akkor engedi át a tűzfal ha előtte kiment egy kérés csomag, s a választ csak arról a külső címről fogadja el, melyre az eredeti ***ping***-et küldték. A beérkező csomag sorsát tehát a csomagnak a tűzfal által nyilvántartott nyitott kommunikációs kapcsolatokhoz való viszonya is befolyásolja. (***Ping*** küldéskor a tűzfal „megjegyzi”, hogy erre választ várunk, s azt is, hogy a válasznak honnan kell érkeznie. A válasz beérkezésekor a „megjegyzést” törli, így

ha ugyanarról a külső címről egy újabb, kérés nélküli „válasz *ping*”érkezik, az már nem jut át.)
Állapotfigyelő tűzfalak szűrési szabályainak megadásakor tehát a normál fejrész információkon kívül a csomag kommunikációs kapcsolaton belüli szerepére is hivatkozhatunk.
A példában említett ***ping*** esete ugyan talán a legegyszerűbb kommunikációs kapcsolat, de bonyolultabb (például ***TCP***) csatornák esetén is érvényesül ugyanez az elv.
A hálózati forgalom feletti legmagasabb szintű kontrollt az ún. ***alkalmazási réteg szintű*** tűzfalak valósítják meg. Ezek már belelátnak a csomagok adatmezőibe is, s az ott szállított információkat összefüggő adatfolyamnak tekintve alkalmazásszintű szűrést is végezhetnek. Blokkolhatják a rosszindulatú tartalmakat szállító web kapcsolatokat, a veszélyesnek ítélt email adatátvitelt, azonnali üzenetküldő csatornákból kiiktathatják a fájl továbbítást, stb. A legmagasabb szintű kontroll azonban nyilvánvaló módon a legnagyobb bonyolultsággal is együtt jár, mind a működés, mind a konfiguráció szintjén.
Az intézményi tűzfalak rendszerint különálló gépen helyezkednek el, s az intézmény belső hálózatát választják el a külső, nem biztonságos hálózattól (például az internettől). A személyi tűzfalak ezzel szemben általában egy-egy munkaállomást védenek, s magára a védendő munkaállomásra telepítjük őket. Az alábbiakban megvizsgáljuk, hogyan alakítható ki egyszerű személyi tűzfal megoldás a Linux kernel beépített tűzfal funkcionalitására támaszkodva.
A megvalósítani kívánt védelem lényege tehát a következő: Saját linuxos munkaállomásunkon felélesztjük a beépített tűzfal szolgáltatást és azt a munkaállomás védelmére alkalmas szűrési szabályokkal látjuk el.

Kernelszintű tűzfalfunkciók a Linuxban

Amint arról korábban már szó esett, a ***Linux*** kernel beépített tűzfal funkcióval rendelkezik, melynek fontosabb jellemzői az alábbiak:

- A beépített tűzfal működhet egyszerű csomagszűrőként vagy állapotfigyelő tűzfalként. A két üzemmód nem lehet egyszerre aktív, a kernel fordításánál kell eldöntenünk, melyik megoldást választjuk. (Aki nem maga fordítja a kernelt, az ma már minden bizonnyal az állapotfigyelő funkciót találja a készen kapott bináris rendszermagban.)
- A kernel szintű csomagszűrő funkciót ***netfilter***nek nevezik, mely közvetlenül vagy modulként is belefordítható a kernelbe.
- A ***netfilter*** támogatja a ***NAT***-ot és a ***MAC*** alapú címzést is. (Ha valaki nem tudja, mit jelentenek ezek a rövidítések, beleértve az előző pontban említett „modulként való fordítás” lehetőségét is, semmi gond, a cikket azért még érdemes tovább olvasni.)
- Az állapotfigyelő csomagszűréshez legalább 2.4-es kernelre van szükség.
- A kernel szintű csomagszűrés konfigurálása (vagyis a tűzfal szűrési szabályainak kijelölése) két felületen, az ***ipchains***-en vagy az ***iptables***-en keresztül történhet. Az ***ipchains***-en az állapot független, az ***iptables***-en az állapotfüggő funkciók konfigurációja végezhető. Ha a kernel fordításakor az állapotfüggő tűzfal megoldás beépítése mellett döntöttünk, akkor értelemszerűen a szabályok kijelöléséhez az ***iptables*** alkalmazást kell használnunk.

A továbbiakban feltételezzük, hogy a kernel tartalmazza az állapotfüggő tűzfal funkcionalitást (ami a készen kapott rendszermagok esetében általános) és a konfigurálásához az ***iptables*** programot használjuk.

1. táblázat

INPUT	Azok a csomagok kerülnek ebbe a láncba, melyek kívülről, a hálózatról érkeznek és címzettjük a védett gép
OUTPUT	Azok a csomagok kerülnek ebbe a láncba, melyek forrása a védett gép, címzettje pedig egy másik, a hálózaton keresztül elérhető eszköz.
FORWARD	Azok a csomagok kerülnek ebbe a láncba melyek forrása és címzettje egyaránt valamely hálózaton keresztül elérhető külső eszköz.. A védett gép feladata csak ezeknek a csomagoknak a továbbítása. (Akkor érdekes ha az adott gépen a csomagtovábbítás engedélyezve van.)

Az állapotfigyelő csomagszűrés működése

A kernel a hálózatról érkezett vagy oda kilépni szándékozó csomagokat a ***netfilter*** feldolgozási láncaiba irányítja. A legfontosabb feldolgozási láncok a következők: *1. táblázat*.
Vannak még más előre definiált láncok is, és mi magunk is hozhatunk létre láncokat. A láncok kezelésével kapcsolatos fontosabb tudnivalók a következőkben foglalhatók össze:

- Az egyes láncokra szabályokat adhatunk meg, melyeket a ***netfilter*** érvényesít. A szabályok a láncba került csomagok tovább engedésére, eldobására vagy naplózására vonatkozhatnak.
- A szabályok sorrendje fontos. Amikor a rendszer a soron következő csomag sorsáról dönt, megvizsgálja, hogy a megadott szabályok közül illeszkedik-e valamelyik a csomagra. A keresést mindig a szabálylista elején kezdi, és az első illeszkedő szabálynál fejezi be. (Kivéve például a naplózást. Lásd később.)
- A szabály rendszerint egy feltételből és egy akcióból áll. Ha a feltétel illeszkedik a csomagra, akkor a rendszer végrehajtja az akciót, például eldobja a csomagot.
- Ha a csomagra egyik szabály sem illeszkedik, akkor a lánc alapértelmezett szabálya lép életbe. (Minden láncra vonatkozik egy alapértelmezett szabály.)

Mivel az otthoni munkaállomások rendszerint nem végeznek csomagtovábbítást, így az alábbiakban csak az ***INPUT*** és ***OUTPUT*** láncok kezelésével foglalkozunk.

A szűrési szabályok megadása

Annak érdekében, hogy a beépített tűzfal működését saját igényeinkhez illesszük, a ki és belépni szándékozó csomagokra (`INPUT` és `OUTPUT` lánc) szabályokat kell meghatároznunk.
A szabályok megadása hagyományos módon az `iptables` paranccsal (parancssor programmal) történik. Léteznek ugyan különböző grafikus felületek is az ***iptables*** fölé, melyek megkönnyítik a tűzfal szabályok kijelölését, ezek azonban rendszerint a lehetőségeket is szűkítik. Így a továbbiakban csak az `iptables` parancs használatára koncentrálunk.
Szabály megadásának formátuma az `iptables` paranccsal:

```
iptables parancs láncnév sor-
➥szám feltétel művelet
```

Lássuk az egyes paraméterek jelentését.
A `parancs` az új szabály felvételének módja a meglévő szabályok közé. Lehetséges értékei: *2. táblázat*.
A `láncnév` annak a láncnak a megadására való, melyre a szabály vonatkozik. Szokásos értékei: *3. táblázat*.
A `sorszám` annak a szabálynak a sorszáma, melyre a művelet vonatkozik. `-A` (hozzáfűzés) esetén nem használjuk.
A `feltétel` a kezelni kívánt csomag meghatározása. (Vagyis itt kell megadnunk, hogy a szabály milyen csomagokra vonatozzék.) A fontosabb lehetőségek: *4. táblázat*.
A feltételekből természetesen több is megadható ugyanabban a parancsban. A !-el a az illeszkedési feltételek általában invertálhatók.
Végezetül a `művelet` a feltételre illeszkedő csomagok kezelési módja. (Itt határozzuk meg, hogy a megadott feltételre illeszkedő csomaggal pontosan mit is kell csinálni). Megadása:

```
-j kezelési_mód
```

A `kezelési_mód` lehetséges értékei: *5. táblázat*.
Az `iptables` parancs a fenti formájában alkalmas arra, hogy jól meghatározott csomagokat válasszunk ki az átlépni szándékozó üzenetfolyamból és rendelkezzünk a kezelésük módjáról. Mi történik viszont azokkal a csomagokkal, melyekre a rendszer nem talál illeszkedő szabályt? Nos, ezek sorsáról a láncok alapértelmezett kezelési szabályai döntenek.

2. táblázat

`-I`	Szabály beszúrása az adott sorszámú helyre
`-D`	Az adott sorszámú szabály törlése
`-R`	Az adott sorszámú, már létező szabály cseréje a parancsban megadott szabályra
`-A`	A szabály hozzáfűzése a meglévő szabálylista végéhez

Láncok alapértelmezett kezelési szabályai

Ha egyik megadott kezelési szabály sem illeszkedik az éppen feldolgozásra váró csomagra, akkor a csomag sorsát az alapértelmezett kezelési mód dönti el. Az alapértelmezett kezelési mód beállítása a következőképpen történik:

```
iptables -P lánc_név
➥kezelési_mód
```

A `kezelési_mód` lehetséges értékei: `DROP, ACCEPT, REJECT, LOG`

Ennek értelmében (például) a beérkező csomagok feldolgozása az alábbi módon megy végbe:

1. A csomag megérkezik a hálózatról
2. A csomagot a kernel az `INPUT` láncba irányítja
3. A csomag feldolgozásakor a ***netfilter*** sorszám szerint növekvő sorrendben végignézi az `INPUT` láncra megadott szabályokat. Az első illeszkedő szabálynál megáll, végrehajtja a szabály által kijelölt akciót, majd veszi a következő csomagot. (Kivétel a `LOG` kezelési mód. Ebben az esetben a tűzfal naplózza a csomag megjelenését, s folytatja az illeszkedő szabály keresését.)
4. Ha a rendszer az `INPUT` lánc szabálylistájában nem talál illeszkedő kezelési utasítást, akkor a csomag sorsát az `INPUT` lánc alapértelmezett kezelési módja dönti el.

Lássunk két példát

Először nézzük meg, mit kell tennünk, ha azt szeretnénk elérni, hogy valamennyi hálózati kommunikáció alapértelmezés szerinti tiltva legyen.

3. táblázat

`INPUT`	Beérkező csomagok lánca
`OUTPUT`	Kilépni szándékozó csomagok lánca

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

A művelet eredményeként a rendszer eldob minden beérkező és távozni szándékozó csomagot, hacsak egy illeszkedő szabály másként nem rendelkezik. Működő tűzfalaknál rendszerint ez a beállítás használatos, alapértelmezés szerint minden kommunikáció tiltva van, amit mégis engedélyezni akarunk, arról (alkalmas illeszkedő szabály megadásával) külön rendelkezünk.
A második példában a beérkező csomagokat letiltjuk, a kimenőket azonban engedélyezzük, vagyis a hálózati kommunikáció számára amolyan egyirányú utcát alakítunk ki.

```
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
```

A művelet eredményeként a rendszer eldob minden beérkező csomagot, viszont a csomagok távozását engedélyezi. (Minden, a védett gépről induló csomagot kiengedünk, de hogy mely csomagok bejutását engedélyezzük, azt külön határozzuk meg, az ***INPUT*** láncra megadott szabályokkal.) Személyi tűzfalak esetén gyakori beállítás. Ilyenkor az ***INPUT*** láncra olyan csomagok átengedését engedélyezzük, melyek már egy létező (belülről kezdeményezett) kommunikációs kapcsolat részeként érkeznek.

4. táblázat

`-s forrás_IP_cím`	A forrás IP cím, ahonnan a csomagot küldték. Megadható pontozott decimális alakban (pl. 10.21.32.43) vagy domén neves formában (pl. mozdony.sihu.hu) Lehet hálózati cím is cím/maszk alakban. Ilyenkor a szabály valamennyi olyan csomagra vonatkozik, melynek forrása az adott hálózat. Pl. `-s 10.21.32.43` – azokat a csomagokat választja ki, melyeket a 10.21.32.43 IP című hálózati helyről küldtek.
`-d célcím`	Cél IP cím, ahová a csomagot küldték. A részletek megegyeznek az előző pontban írtakkal.
`-i bemenő_interfész`	Az az interfész (hálózati kártya), melyen a csomag belépett. Pl. eth0. (Csak az INPUT lánc esetén használható.)Pl. `-i eth0` – azokat a csomagokat jelenti, melyek az eth0 jelű hálózati interfészen (hálózati kártyán) keresztül lépnek be a védett gépre.
`-o kimenő_interfész`	Az az interfész (hálózati kártya), melyen a csomag távozni fog. (Csak az OUTPUT lánc esetén értelmes.)
`-p protokoll`	A csomag adatmezője által használt kommunikációs protokoll. Lehetséges értékek: tcp, udp, icmp, all (minden).Pl. `-p tcp` – a feltétel a TCP csomagokra illeszkedik
`--dport célport`	A cél port, ahol a csomagot várják. Lehet szám, vagy az /etc/services-ben megadott név. Port tartomány megadása: kezdő_sorszám:záró_sorszámPl. `-dport 80` – azokat a csomagokat választja ki, melyeket valamely hálózati eszköz 80-as portjára küldtek
`--sport forrásport`	A forrás port, ahonnan a csomagot küldték. (ld. fent)
`--tcp-flags maszk lista`	A maszk-ban felsoroljuk (vesszővel elválasztva), hogy a csomag fejrészében mely TCP jelzőbiteket (flag-eket) kívánjuk vizsgálni, a listában pedig megadjuk, hogy ezek közül melyiknek kell beállítva lennie. A lehetséges jelzőbitek: SYN, ACK, PSH, URG, RST, FIN, ALL Pl. `--tcp-flags SYN,ACK SYN` – Kiválasztja mindazokat a TCP csomagokat, melyek SYN flag-je be van állítva (1), ACK flag-je pedig törölt (TCP kapcsolat felvétel első fázisa).
`--icmp-type típus`	Ha a szabállyal ICMP csomagokat kívánunk kezelni, akkor itt adhatjuk meg az illeszkedő ICMP csomag típusát. A lehetséges típusok az iptables -p icmp -h paranccsal listázhatók ki. Csak ICMP protokoll (-p icmp) esetén használható.Pl. `--icmp-type echo-request` – a ping csomagok kiválasztása
`-m state --state állapotlista`	A szabállyal kezelni kívánt csomagok kommunikációs kapcsolaton belüli állapota határozható meg ezzel a paraméterrel. Lehetséges értékek (állapotok):· • NEW – A csomag egy új kapcsolatot indít· • ESTABLISHED – A csomag egy már létező kapcsolat része· • RELATED – A csomag egy új kapcsolatot indít, de társítható egy létező kapcsolathoz is. (Pl. FTP adattranszfer, vagy ICMP hibaüzenet, mely egy másik kommunikációs kapcsolat eredményeként keletkezett)· • INVALID – A fentiek szerint nem azonosítható csomag Pl. `-m state --state ESTABLISHED` – azokra a csomagokra illeszkedik, melyek egy, már létező kommunikációs kapcsolat részei.

5. táblázat

ACCEPT	A csomag tovább engedése. (Áthaladhat a tűzfalon.)
DROP	A csomag eldobása hibaüzenet nélkül. (A csomagot a tűzfal eldobja, anélkül, hogy értesítené az eldobott csomag küldőjét.)
LOG	A csomag feljegyzése a rendszernaplóba (syslog). (Ezt követően tovább folytatódik a listában az illeszkedő szabályok keresése, hiszen a csomag sorsa még nem dőlt el.)
DROP	A csomag eldobása ICMP hibaüzenet visszaküldése mellett. (A hibaüzenet típusa megadható.) A csomagot a tűzfal eldobja és értesíti a csomag küldőjét a műveletről.

Ezzel a ***netfilter*** és az ***iptables*** működésével, használatával kapcsolatos alapismeretek bemutatásának a végére értünk. A sorozat következő részében az itt leírtakat a gyakorlatban is alkalmazni fogjuk, konkrét célokat szolgáló szűrési szabályokat fogunk beállítani, és összeállítunk egy egyszerű, otthoni munkaállomás védelmét szolgáló szabálylistát is.

Nagy Sándor
(nasi63@gmail.com)

Tűzfalszabályok előállítása Perl program segítségével

Az internet gonosztevőinek köszönhetően már a legtöbb kezdő felhasználó is tisztában van egy biztonságos tűzfal szükségességével.

Mint a legtöbb ***Linux***-felhasználó, kezdetben magam is egyszerű ***Bash*** parancsfájlokkal készítettem el a gépem tűzfalszabályait. Lassan azonban belefáradtam, hogy ugyanazokat a kódsorokat írom újra és újra, így hát elkezdtem ciklusokat használni az ismétlések elkerülésére. El akartam különíteni magukat a szabályokat is a program többi részétől – erre a megoldás az, hogy a program külső beállításfájlokkal dolgozik. Mivel a ***Perlhez*** lényegesen jobban konyítok, mint a ***Bash***-hez, úgy döntöttem, Perlben fogom megírni a tűzfalszabályaimat. A cikkben bemutatott programot meg lehet írni ***Bash***-ben, vagy bármilyen parancsnyelven, de akár ***C++***-ban is. Nem a nyelv a fontos. A lényeg, hogy megírjuk a programot, teszteljük, és ezután már nincs más dolgunk, csak módosítani a beállításfájl biztonsági szabályait és újra lefuttatni a programot. A beállításfájlok szerkezete magáért beszél, így könnyen olvashatók, érthetők, módosíthatók.

Az *1. Listában* látható a ***Perl*** szkript. A fentről lefelé elvet követtem (***top-down programming***), úgyhogy az első néhány sorból már látszik, hogy mit csinál a program. Remélhetőleg nem csak ***Perl*** programozók számára lesz érthető.

Látható, egészen rövid program és nem is bonyolult. Elég rugalmas azonban ahhoz, hogy fekete- és fehérlistákat hozzunk létre egyes gépekből, vagy akár egész hálózatokból. Mint majd látni fogjuk, a `build_chains()` és az `add_rules()` függvényekben alkalmazott szabálykurtító

1. Lista firewall.pl

```
#!/usr/bin/perl
$default_policy = "DROP";
$iptables = "/sbin/iptables";
$work_dir = "/root/fw";
set_ip_forwarding(0);
load_interfaces();
$protocols{tcp}++; $protocols{udp}++; $protocols{icmp}++;
init();
set_default_policy();
add_good_hosts();
add_bad_hosts();
build_chains();
add_rules();
set_default_action();
set_ip_forwarding(1);
exit;
################################################
sub    load_interfaces {
    my($int, $name);
    local(*FILE);
    open FILE, "$work_dir/interfaces.conf";
    while (<FILE>) {
        chomp($_);
        if ($_ eq "") { next; }
        ($name, $int) = split(/\s*=\s*/, $_);
        $interface{$name} = $int;
    }
}
sub    init {
    iptables("-F");  # flush rules
    iptables("-t nat -F");
    iptables("-X");  # delete chains
    iptables("-Z");  # zero counters
    iptables("-t nat -A POSTROUTING -j MASQUERADE");
    iptables("-A INPUT -m conntrack -ctstate ESTABLISHED
        -j ACCEPT");
}
sub    set_default_policy {
    iptables("-P INPUT $default_policy");
    iptables("-P OUTPUT ACCEPT");
    iptables("-P FORWARD ACCEPT");
    return;
```

1. Lista folytatás

```
}
sub    build_chains {
    my($interface, $protocol, $chain);
    foreach $interface (keys %interface) {
        foreach $protocol (keys %protocols) {
            $chain = "$interface-$protocol";
            iptables("-N $chain");
            iptables("-A INPUT -i
            ➥$interface{$interface}
                -p $protocol -j $chain");
        }
    }
}
sub    add_rules {
    local(*FILE);
    open FILE, "$work_dir/ports.conf";
    while (<FILE>) {
        chomp($_);
        $_ =~ s/#.?//;
        if ($_ eq "") { next; }
        ($int, $proto, $port) = split(/\t/, $_);
        $i = $interface{$int};
        $chain = "$int-$proto";
        if ($proto eq "all") {
            foreach $proto (keys %protocols) {
                $chain = "$int-$proto";
                iptables("-A $chain -i $i -p
                ➥$proto -j ACCEPT");
            }
            next;
        }
        if ($proto eq "udp") {
            iptables("-A $chain -i $i -p udp
            ➥-dport $port
                -j ACCEPT");
            iptables("-A $chain -i $i -p udp
            ➥-sport $port
                -j ACCEPT");
        }
        if ($proto eq "tcp") {
            iptables("-A $chain -i $i -p tcp
            ➥-dport $port -syn
                -j ACCEPT");
            iptables("-A $chain -i $i -p tcp
            ➥-dport $port
                -j ACCEPT");
        }
    }
}
sub    set_default_action {
    my($interface, $protocol, $chain);
    foreach $interface (keys %interface) {
        foreach $protocol (keys %protocols) {
            $chain = "$interface-$protocol";
            iptables("-A $chain -j LOG
                -log-prefix
DEFAULT_$default_policy-$chain-");
            iptables("-A $chain -j
            ➥$default_policy");
        }
    }
}
sub    iptables {
    my($line) = @_;
    print "$iptables $line > /dev/null\n" if
    ➥($debug);
    $result = system("$iptables $line >
    ➥/dev/null");
    if ($result != 0) {
        print "X: ($result) iptables $line\n";
    }
}
sub    set_ip_forwarding {
    my($value) = @_;
    local(*FILE);
    print "Setting IP forwarding to $value.\n";
    open FILE, ">/proc/sys/net/ipv4/ip_forward";
    print FILE $value;
    close FILE;
}
sub    add_good_hosts {
    my($host, $comment);
    local(*FILE);
    open FILE, "$work_dir/good_hosts.conf";
    while (<FILE>) {
        ($host, $comment) = split(/\t/, $_);
        iptables("-A INPUT -s $host -j ACCEPT");
        iptables("-A OUTPUT -d $host -j
        ➥ACCEPT");
    }
}
sub    add_bad_hosts {
    my($host, $comment);
    local(*FILE);
    open FILE, "$work_dir/bad_hosts.conf";
    while (<FILE>) {
        chomp($_);
        ($host, $comment) = split(/\t/, $_);
        iptables("-A INPUT -s $host -j LOG
             -log-prefix $comment");
        iptables("-A OUTPUT -d $host -j LOG
             -log-prefix $comment");
        iptables("-A INPUT -s $host -j DROP");
        iptables("-A OUTPUT -d $host -j DROP");
    }
}
```

(*rule-pruning*) algoritmusnak köszönhetően a *Linux* rendszermagnak nem kell a lényegtelen szabályokkal foglalkoznia.

A `set_ip_forwarding()` függvény azt csinálja, ami a neve alapján elvárható, megmondja a rendszermagnak, hogy továbbítson, vagy éppen ne továbbítson *IP* csomagokat. A függvény egyetlen paramétert fogad, 0 vagy 1 értékkel attól függően, hogy akarunk-e *IP* továbbítást.

A szkript a tűzfalszabályok betöltésekor letiltja a továbbítást, majd futásának befejezte előtt újra engedélyezi azt. E plusz lépések célja, hogy az útválasztónk biztonságos állapotban legyen a szabályok

betöltésekor. Jobb minden forgalmat blokkolni, mint egyetlen támadást beengedni.

A `load_interfaces()` függvény beolvassa a hálózati eszközök neveit, majd könnyen megjegyezhető címkéket kapcsol hozzájuk. A további beállítások során már ezekkel a címkékkel hivatkozunk az eszközökre. Az olyan elnevezések, mint a `lan` (helyi hálózat), vagy a `vpn_to_work` (munkahelyi ***VPN***) csökkentik a hibalehetőségek számát, de így könnyebben is módosíthatjuk a tűzfalat, hogy azt máshol mások is használhassák. Gyakran csak az ***interfaces.conf*** fájlt kell módosítanom az adott hálózatnak megfelelően és máris egy jól használható tűzfalat nyújthatok át egy barátomnak.

A szkript négy beállításfájlt használ: ***interfaces.conf***, ***good_hosts.conf***, ***bad_hosts.conf*** és ***ports.conf***.

A *2. Listában* látható az ***interfaces.conf*** fájlom. Látható, hogy az útválasztómban hat hálózati eszköz van.

Az internetkapcsolat az `eth5`. A házon belüli hálózatot ***10/100TX Ethernet*** köti össze. Egy ***MythTV PVR Gigabites Ethernettel*** kapcsolódik az útválasztóhoz fájltárolás céljából. Van még egy csatoló ***Wi-Fihez*** és ***VoIP***-hoz, továbbá egy ***VPN*** kapcsolat néhány barátom számítógépéhez. Könnyebb megjegyezni, hogy a `lan` nevű eszköz a 10/100-as rézszálas hálózat, mint azt, hogy eth3 a ***VoIP***, vagy a ***Wi-Fi*** csatoló. Azt pedig a legkevésbé sem akarjuk, hogy valamelyik csatoló nem a neki készített szabályokat kapja.

Az `init()` függvény végzi az ***iptables*** beállításnak első lépéseit. Először kiürítjük (`flush`) vagy töröljük az összes szabályt és felhasználói láncot. Aztán lenullázzuk a számlálókat. Később ezekből tudhatjuk meg, hogy az egyes tűzfalszabályaink hány csomagot kaptak el. Ezután beállítjuk a címálcázást (***IP masquerading***). Az olyan forgalmat, amely egy létező kapcsolathoz tartozik, további ellenőrzés nélkül átengedjük a tűzfalon. Így nem kell minden csomagnak végigjárnia az összes szabályt, elég, ha csak az új kapcsolatokat vizsgáljuk.

A `set_default_policy()` megadja, hogy mi legyen az egyetlen szabályra sem illeszkedő csomagok sorsa. Ez esetben csak a bejövő forgalommal foglalkozunk, a házirend elfogad minden kimenő és továbbítandó csomagot, de kisebb módosítással minden irányhoz megadhatunk házirendet (***policy***). Az alapértelmezett házirend alapján a szkriptünk eldobja a csomagokat, tehát az adminisztrátornak külön ki kell jelölnie az engedélyezett forgalmat. Ez a legbiztonságosabb megoldás, és lényegesen jobb annál, mintha mindent átengednénk és csak az ártó szándékú forgalmat blokkolnánk. Előre úgysem tudhatjuk, mi az ártó, így a legjobb mindent tiltani és csak azt engedélyezni, ami tényleg szükséges.

Az `add_good_hosts()` a ***good_hosts.conf*** fájlban megadott gépekből és hálózatokból hoz létre szabályokat, melyek minden forgalmat engedélyeznek. Ezek a szabályok nem hálózati eszközhöz kötődnek. A fehérlistán levő gépekről és hálózatokból minden csatolón minden adatforgalmat beengedünk. Ezen a listán általában szerepel az otthoni gépem és a munkahelyi hálózat, így ha valami valami hibát vétek a beállítás közben, akkor sem zárom ki magam az útválasztó gépről, tehát otthonról és a munkahelyemről is javíthatom a hibát. Persze alapfeltétel, hogy mind az otthoni gép, mind a munkahelyi hálózat biztonságos legyen, és lehetőleg ez a lista se legyen hosszú.

Az `add_bad_host()` által létrehozott szabályok blokkolnak minden forgalmat a ***bad_hosts.conf*** fájlban megadott gépek és hálózatok felől. Ez a függvény majdnem ugyanúgy működik, mint az `add_good_hosts()`, egyetlen fontos kivétellel. A feketelistán levő helyekről érkező csomagokat az útválasztó naplózza, a bejegyzéshez pedig egy megjegyzést is fűz úgy, ahogy azt a ***bad_hosts.conf*** fájlban előre megadtuk. Így elég csak belepillantani a naplóba és rögtön látjuk, melyik csomag miért lett eldobva. Hasznos fejlesztés lenne, ha az `add_bad_hosts()` függvény külön láncba tenné a feketelista szabályokat, a láncot pedig egy korai szakaszban hívná meg. Így akár egy külső programból is kényelmesen törölhetnénk, vagy adhatnánk címeket ehhez a lánchoz, esetleg éppen a naplóbejegyzések függvényében.

A `build_chains()` tűzfalszabály láncokat hoz létre. Minden hálózati eszköz és protokoll-kombinációhoz külön láncot építünk. Például egy linuxos útválasztón négy eszközzel – `eth0`, `eth1`, `eth2`, `eth3` – a következő láncaink lennének: `eth0-tcp`, `eth0-udp`, `eth1-tcp`, `eth1-udp` és így tovább. Aztán létrehozzuk a szabályokat, amelyek a megfelelő lánchoz küldik a csomagokat. Eredményül egy faszerkezetet kapunk amely meghatározza az útválasztóra érkező hálózati csomagok útvonalát. Ellentétben egy lineáris tűzfalszabály listával, a rendszermagnak itt nem kell nyilvánvalóan lényegtelen szabályokkal foglalkoznia. Nem fogunk tehát ellenőrizni a ***WAN*** csatolón bejövő ***TCP*** csomagokat a ***Wi-Fi*** csatoló ***UDP*** csomagjaira vonatkozó szabályok alapján.

Nem vizsgáltam még meg, hogy ez a szabály-visszametsző eljárás eredményez-e számottevő teljesítménynövekedést. Nem kerül azonban semmibe ennek a döntési fának a létrehozása, tehát ha a teljesítményt nem is növeli jelentősen, nem is igényel több programozói munkát, tehát összességében mégiscsak megéri.

A munka oroszlánrésze az `add_rules()` függvényben történik, amely beolvassa az *5. Listában* látható ***ports.conf*** fájl tartalmát.

2. Lista interfaces.conf

```
lo = lo
gig = eth0
lan = eth1
wifi = eth2
voip = eth3
wan = eth5
tun = tun0
```

3. Lista good_hosts.conf

```
127.0.0.1     Loopback
224.0.0.0/8     Multicast
10.4.0.0/16     VPN
10.0.1.1/32     Home office
```

4. Lista bad_hosts.conf

```
216.250.128.12  My_comment
www.microsoft.com       Micro
                        ➥soft
```

5. Lista ports.conf

```
wan   tcp   22   # ssh
wan   tcp   25   # smtp
wan   tcp   80   # http
wan   udp   53   # dns
wan   udp   1194   # openvpn
wan   udp   5060   # sip
wan   udp   4569   # iax2
wan   udp   10000:20000   # rtp
lo   all
lan   tcp   22   # ssh
lan   tcp   25   # smtp
lan   udp   53   # dns
lan   tcp   53   # dns
lan   udp   67   # dhcp
lan   udp   68   # dhcp
lan   tcp   80   # http
lan   tcp   111   # portmapper
lan   udp   111   # portmapper
lan   tcp   143   # imap
lan   tcp   443   # https
lan   tcp   2049   # nfs
lan   udp   2049   # nfs
lan   tcp   3306   # mysql
lan   udp   4569   # iax2
lan   udp   5060   # sip
lan   tcp   5432   # postgresql
lan   tcp   10000   # webmin
lan   all
gig   all
tun   all
wifi   udp   1194   # openvpn
voip   udp   5060   # sip
voip   udp   4569   # iax2
voip   udp   53   # dns
voip   tcp   22   # ssh
voip   udp   10000:20000   # rtp
voip   tcp   80   # http
```

Mielőtt részletesebben megvizsgálnánk az `add_rules()` függvényt, lássuk a ***ports.conf*** fájlt.
A ***ports.conf*** minden szabályhoz tartalmaz egy sort. Minden sor három oszlopból és egy választható megjegyzésből áll, amelyet # karakter előz meg. Az első annak a hálózati eszköznek a neve, azaz címkéje, amelyre a szabály vonatkozik. A második oszlop a protokoll: `tcp`, `udp`, vagy mindegyik, azaz `all`. Utóbbi esetben olyan szabályt kapunk, amely minden csomagot beenged a kérdéses csatolón. Az harmadik oszlop a kapuszám, vagyis a port. Az első sor például egy olyan szabályt hoz létre, amely beengedi az ***SSH*** forgalmat a `wan` hálózati csatolón.
Látható, hogy van egy szabály, amely engedélyezi a `lo - loopback`, azaz visszacsatoló hálózati eszközön az adatforgalmat. Enélkül sok program nehezen felderíthető hibákat produkálna. Felvetődhet a kérdés, hogy miért van szükség ennyi szabályra a ***LAN*** eszközön, hogy aztán a végén egy `all`-lal minden csomagot beengedjünk. A fő ok, hogy amíg a gyerekek fel nem nőnek és el nem kezdik használni az internetet, az otthoni hálózatot biztonságosnak tarthatom. Ha azonban minden szolgáltatáshoz külön szabályt készítek, remek statisztikát kapok arról, hogy melyik szolgáltatás mennyi forgalmat generált. No meg a biztonság biztosítása folyamatos munka. Időről időre újabb szabályokat adok a tűzfalam szigorításához, aztán előbb utóbb az az `all` is eltűnik majd a házirendből.
Térjünk vissza az `add_rules()` függvényre. Ez a program leghosszabb függvénye, de azért még könnyen érthető. A kód `tcp` és `udp` szabályokkal foglalkozó része egyszerűen csak két szabályt készít a ***ports.conf*** fájl minden szabálya alapján. Egyik a cél-, a másik a forráskapuhoz kapcsolódik. Ez elsőre furcsának tűnhet, mivel csak a bejövő forgalommal foglalkozunk. Igazából azonban így biztosítjuk, hogy mind a bejövő, mind a kimenő kapcsolat engedélyezve legyen. Például egy a ***WAN*** eszköz 80-as kapujára érkező kapcsolat a saját web kiszolgálómat célozza meg, míg ugyanezen az eszközön a 80-as kapuról érkező adat egy külső kiszolgáló válasza egy, a belső hálózatról érkező kérésre.
Az `all` szabályokat kezelő kód speciális eset. Itt minden protokollra külön szabályt alkotunk az adott hálózati eszközön. Talán túl bonyolultnak tűnket, de van egy érdekes mellékhatása. Ha az útválasztó ismeretlen protokollú csomagot kap, mint mondjuk az ***IPSec***, az alapértelmezett házirendet fogja alkalmazni, még akkor is, ha az azt jelenti, hogy engedjen át minden adatforgalmat. A „minden protokoll" tehát „minden ismert protokoll"-t jelent, és szerintem ez jó.
A szkript nagyjából olyan sorrendben adja át a szabályokat a rendszermagnak, ahogy a ports.conf fájlban szerepelnek. Mondom csak nagyjából, mivel a szabályok a hálózati eszköz és protokoll alapján meghatározott láncba kerülnek. A szabályon belül azonban nem változik a sorrend.
A `set_default_action()` függvény meghatározza, hogy mi történjen azokkal a csomagokkal, amelyekre egyetlen korábbi szabály sem illeszkedik. Ez hasonlónak tűnhet a `set_default_policy()`-hez, de van egy apró különbség.
A `set_default_policy()` az alapértelmezett tűzfal házirendet állítja be, a `set_default_action()` tűzfalszabályokat hoz létre, amelyek elkapják a korábban el nem kapott csomagokat, lezárva az egyes láncokat, azelőtt, hogy a rendszermag az alapértelmezett házirendet alkalmazná.
Ha egy megfelel egy ilyen szabálynak, létrejön egy napló bejegyzés, majd életbe lép a megadott házirend, ez esetben a `DROP`. A naplóból később megállapítható, hogy milyen csomag és miért lett eldobva.
Nem mondom, hogy ez a program tökéletes és azt sem, hogy minden elképzelhető feladatot képes ellátni. Még akár hibák is lehetnek benne. Mire ez a cikk megjelenik bizonyára további fejlesztéseket végzek a szkripten.
Látható, hogy jelenleg még egyáltalán nem kezeli az ***ICMP*** protokollt, pedig jó lenne például engedélyezni a kimenő `ping` kéréseket, de tiltani a bejövőket. Jó lenne szabályozni a kimenő és a továbbított forgalmat is. Mivel használok ***VoIP***-ot, gondoltam már arra, hogy a szkript beállíthatná a szolgáltatásminőséget (***Quality of Service - QoS***) is. Aki kiegészíti hasznos funkciókkal a programot, kérem értesítsen.
Most olyan, amilyen. Kevesebb, mint 200 sornyi ***Perl*** kód, mégis rugalmasan és hatékonyan kezeli akár tűzfalszabályok százait, a szabályok módosítása pedig olyan egyszerű, hogy még az egészen kezdő ***Linux***-felhasználóknak sem okozhat gondot.

Linux Journal 2006., 143. szám

Mike Diehl a SAIC-nak dolgozik a Sandia National Laboratories-nál Albuquerque-ben, Új-Mexikóban, hálózati felügyeletei szoftvereket fejleszt. Feleségével és két kisfiával él. E-mailben elérhető a mdiehl@diehlnet.com címen.

A kismalac és a farkasok – Az nmap használata

Cikkemben megpróbálom a teljesség igénye nélkül, de a lehetőségek legszélesebb skáláját érintve bemutatni az ***nmap*** (***Network Mapper***) programot, vagyis egy olyan népszerű és sokat próbált hálózati eszközt, melynek rendszeres és szakszerű használatával jelentősen növelhetjük hálózatba kötött számítógépeink biztonságát.
Egyesek szerint a rendszergazdákkal szemben a hackerek egyik legnagyobb előnye az, hogy ők 24 órában tevékenykednek, s így mindig lépéselőnyben vannak. Ezen lépéselőny leküzdésére a rendszergazdák a hackerek által használt eszközöket vetik be saját hálózataik ellen, hogy kiderítsék, milyen sebezhetőségre lelhet rá egy betolakodó, amíg ők otthon éppen a legédesebb álmukat alusszák.
Ehhez a rendszerellenőrzéshez jelenleg az egyik legalkalmasabb eszköz az ***nmap*** pásztázó (portscanner), ami ellenőrzi, hogy egy adott számítógépen mely portok vannak nyitva, azokon milyen, és hányas verziószámú szolgáltatások üzemelnek, valamint milyen operációs rendszer fut rajta. Ezek alapján talán már érthető, miért elengedhetetlen az ismerete.

Mik azok a portok?

Mielőtt elmélyülnénk a program adta lehetőségekben, tegyünk egy rövid hálózatelméleti kitekintést. A portok vagy kapuk lehetővé teszik, hogy egy adott ***IP*** címmel rendelkező számítógépen egyszerre több különböző szolgáltatást is el lehessen érni. Minden démon és program, ami hálózati kapcsolatokat használ, legyen az ***TCP*** vagy ***UDP*** kapcsolat, portokon keresztül kommunikál a hálózaton. A kapuszámok 1-től 65536-ig terjednek. Ezen belül három csoportot lehet megkülönböztetni, az 1-1023-ig terjedő ún. rendszerkapukat, ahol csak rendszergazdai jogosultságú folyamatok működhetnek, az 1024-49151-ig terjedő skálát, ami a nem rendszergazdai jogokkal bíró folyamatok kapuit tartalmazza, és az ezen felüli ún. magánkapukat, amiken azok a szolgáltatások futnak, melyek nem kapcsolódnak az internethez. A ***TCP*** és az ***UDP*** protokollhoz külön-külön 65536 port érhető el, így előfordulhat, hogy ugyanazon a számú kapun szolgáltat egy ***TCP*** és egy ***UDP*** folyamat.
Szót kell még ejteni a hivatalos port-hozzárendelésekről, melyeket Linux rendszereken a ***/etc/services*** állomány tartalmaz. A lényege, hogy egy adott számú ***TCP*** vagy ***UDP*** kapun minden rendszeren ugyanaz a szolgáltatás fut. Például a 23-as ***TCP*** porton a ***telnet*** szolgáltatás érhető el. Ami miatt ez nem egy szabvány, az az, hogy egy szolgáltatást könnyen áthelyezhetünk egy másik kapura. Ezért ha azt látjuk, hogy egy számítógépen nyitott a 23-as kapu, azaz lehet rá csatlakozni, még nem biztos, csak valószínű, hogy azt egy ***telnet*** démon használja.
A számítógép egy másikhoz történő kapcsolódásakor forrás- és célkapukat használ. A célkapu az előbb említett némileg szabványos port-hozzárendelések közül az egyik, amin egy szolgáltatást szeretnénk elérni. A forráskapu – ahonnan a kapcsolatot kezdeményezzük – sorszáma előre nem meghatározott, alkalmazásfüggő, amit a rendszer dinamikusan oszt ki.

Mit jelent a kapupásztázás?

A kapupásztázás leegyszerűsítve annyit tesz, hogy megpróbálunk a célgép minden egyes portjára csatlakozni. Ha sikerrel járunk, elkönyvelhetjük, hogy a port nyitott, sikertelen csatlakozás esetén pedig azt, hogy a vizsgált kapu zárva van.
Legegyszerűbb példa erre ***TCP*** portok estében a ***telnet*** program, amivel minden egyes portra csatlakozva ellenőrizhetjük, fut-e azon valamilyen szolgáltatás. Beláthatjuk, hogy ez nem egy kényelmes és fejlett módja a felderítésnek.
Itt jön a képbe ***Fjodor*** és az általa írt ***nmap*** ➲ http://insecure.org/nmap/ pásztázó. Az ***nmap*** három különböző fázisban gyűjt információt a célrendszerről:

- Kiszolgáló felderítése (***host scan***): A hálózatban elérhető számítógépek felderítése.
- Kapupásztázás (***port scan***): Egy adott számítógépen futó szolgáltatások és azok verzióinak felderítése.
- Operációs rendszer ujjlenyomat általi azonosítása (***OS scan***): Egy adott számítógépen futó operációs rendszer felderítése.

Az nmap beszerzése és telepítése

Az ***nmap*** népszerűségének köszönhetően minden ***Linux*** terjesztésnek a részét képezi. Ezenkívül elérhető még ***Windows***, ***Free/Open/NetBSD***, ***MacOS X***, ***Solaris***, ***Amiga*** és ***HP-UX*** rendszerekre is. ***Linux*** alatt a legegyszerűbben az általunk használt disztribúció csomagkezelőjével telepíthetjük. Ha a legfrissebb stabil változatot szeretnénk használni, töltsük le a forrást a ➲ http://insecure.org/nmap/download.html oldalról, majd a kicsomagolás után a forráskód könyvtárába lépve a

```
./configure
make
su root
make install
```

parancsokkal telepítjük.

A grafikus felület kedvelőire gondolva az *nmap* rendelkezik grafikus kezelőfelülettel is, de mivel ez nem tartalmaz annyi lehetőséget és nem olyan finoman hangolható, mint parancssorból, ezért itt csak megemlítjük a létezését. Az *nmap* nem rendelkezik konfigurációs fájllal, minden opciót parancssori kapcsolók segítségével adhatunk meg. Mivel a program nagyrészt nyers csomagokat küld a hálózatra, futtatásához több esetben rendszergazdai hozzáférésre lesz szükségünk. Most pedig a három fő pásztázási fázist áttekintve ismerkedjünk meg a programban rejlő lehetőségekkel. Pataméterezésének az alaplogikája a következő:

```
nmap [Pásztázás típusa(i)]
➥[Kapcsolók] {Célgép(ek)}
```

Célgépek meghatározása

Mindenek előtt azt kell tudnunk, hogyan adhatjuk át az *nmap* számára a pásztázni kívánt gépek neveit. Mindez történhet egyszerű felsorolással, vagy egy már előre elkészített lista beolvasásával. Az *nmap* nagyon rugalmasan kezeli a célgépek IP címeit. A 10.200.21.8, 10.200.21-23.8, 10.200.21.0/24, 10.200.21.1,2,3,4,21,54 mind elfogadott formátum.

- `-iL <lista.txt>`: A listában felsorolt számítógépek ellenőrzése
- `-iR <szám>`: `<szám>` darabnyi véletlen *IP* címet generál, és azokat pásztázza végig. Akkor hasznos, ha az interneten olyan számítógépeket keresünk, amelyeken nyitva van egy adott port. Ha a szám értéke nulla, végtelen számú *IP* címet generál. A multicast és belső hálózati *IP* címek automatikusan át lesznek ugorva.
- `--exclude <ipcím1,ipcím2>`: Ide jönnek vesszővel elválasztva azoknak a gépeknek az *IP* címei, amiket ki akarunk hagyni a tartomány vizsgálatakor. Ez akkor hasznos, ha egy teljes hálózatot ellenőrzünk, de néhány gépre vagy alhálózatra nem mi felügyelünk.
- `--excludefile <kihagy.txt>`: Ugyanaz, mint az előző, csak egy fájlban szerepel az ellenőrzésből kihagyni kívánt számítógépek listája.

Kiszolgáló felderítése

- `-sP` ***(Ping scan)***: Végigpingeli azoknak a számítógépeknek a listáját, amit megadunk neki, ezáltal kideríthető, melyek a hálózaton aktuálisan elérhető gépek.
- `-P0` ***(P nulla)***: A kapupásztázást nem előzi meg az aktivitást ellenőrző ***ping***. Akkor hasznos, ha tudjuk, hogy a célszámítógép üzemel, illetve ha az úgy van „tanítva", hogy ne válaszoljon a ***ping***-re. Az ***nmap*** ezzel a kapcsolóval minden esetben megpróbálja pásztázni az adott gépet.
- `-PS <portlista>` ***(TCP SYN Ping)***: SYN jelzőbittel ellátott csomagot küld a gép adott portjára. Ez a port alapértelmezetten a 80-as port, de a <portlistában> megadható, hogy mely portokra küldje ezeket a csomagokat. Ha RST csomag érkezik vissza, zárva van a port, ha SYN/ACK csomag, akkor nyitva.
- `-PA <portlista>` ***(TCP ACK Ping)***: ACK jelzőbites csomagok használata a pásztázáskor. Működése megegyezik a ***SYN Pingével***. Érdemes mind a kettőt használni, mert sokszor a ***SYN*** csomagokat eldobják a tűzfalak.
- `-PU <portlista>` ***(UDP Ping)***: Üres ***UDP*** csomagot küld a 31338-es portra, ezzel ellenőrizve a gép állapotát.
- `-PR` ***(ARP Ping)***: Ethernet hálózatok ellenőrzésekor ez a javasolt eljárás. Olyannyira, hogy az ***nmap*** automatikusan erre vált át, ha azt érzékeli, ha helyi hálózaton pásztázunk működő gépek után.

Kapupásztázási technikák

Az ***nmap*** a portok állapotát 6 különféle csoportba sorolja.

- ***Nyitott (open)***: Valamilyen ***TCP*** vagy ***UDP*** protokollt használó szolgáltatás fut az adott porton. A hackerek számára ez jelenti a főnyereményt, hiszen minden nyitott port egy potenciális támadási lehetőség.
- ***Zárt (closed)***: Elérhető, de nem fut rajta semmilyen szolgáltatás.

1. Lista Egyszerű kapupásztázás pingelés nélkül

```
csaba@lcs:~$ nmap -sT -P0
➥scanme.nmap.org

Starting Nmap 4.20
➥( http://insecure.org ) at
➥2006-12-28 18:08 CET
Interesting ports on
➥scanme.nmap.org
➥(205.217.153.62):
Not shown: 1691 filtered
➥ports
PORT      STATE     SERVICE
22/tcp    open      ssh
25/tcp    closed    smtp
53/tcp    open      domain
70/tcp    closed    gopher
80/tcp    open      http
113/tcp   closed    auth

Nmap finished: 1 IP address
➥(1 host up) scanned in
➥68.421 seconds
csaba@lcs:~$
```

- ***Szűrt (filtered)***: Az ***nmap*** nem tudja eldönteni, hogy az adott port nyitva van-e, ugyanis valamilyen csomagszűrési eljárás megakadályozza, hogy elérje a kaput.
- ***Nem szűrt (unfiltered)***: A port elérhető, de az ***nmap*** nem tudja eldönteni, hogy nyitva van-e vagy sem. Csak a tűzfalszabályok felfedezésére használatos ***ACK*** pásztázás sorolja ebbe a csoportba a portokat.
- ***Nyitott|Szűrt (open|filtered)***: nem lehet eldönteni, h a port nyitva van, vagy szűrt. Az ***UDP***, ***IP*** protokoll, ***FIN***, ***Null***, és ***Xmas*** pásztázás sorolja ebbe a csoportba a kapukat.
- ***Zárt|Szűrt (closed|filtered)***: Nem eldönthető, hogy a port zárt vagy szűrt. Az ***Idle pásztázás*** által használt jelölés.

  - ***-sS (TCP SYN scan)***: Az alapértelmezett pásztázási eljárás, ami SYN csomagokat használ. Gyors és pontos. A ***SYN*** pásztázás nem fejezi be a háromlépéses ***TCP*** kézfogást.

A második lépés, a ***SYN/ACK*** csomag után ***RST*** csomaggal zárja a kapcsolatot.

- `-sT` ***(TCP Connect() scan)***: Az alapértelmezett pásztázási eljárás abban az esetben, ha nincs megfelelő jogosultságunk ahhoz, hogy nyers csomagokat tudjunk küldeni a hálózatra, illetve ***IPv6*** hálózatok pásztázásakor. A `Connect()` rendszerhívást használva teljes kapcsolatot épít fel az ellenőrizni kívánt géppel, ezáltal könnyebb felfedezni.
- `-sU` ***(UDP scan)***: Annak ellenére, hogy a legnépszerűbb hálózati szolgáltatások ***TCP*** protokollt használnak, az ***UDP***-t használó programok közül is akad néhány híresebb. Ilyen például a ***DNS***, a ***DHCP***, ***SNMP***. Úgy működik, hogy küld egy üres ***UDP*** fejlécet a célportra, és a válaszból következtet a port állapotára. Működik a ***TCP*** pásztázásokkal együtt.
- `-sF` ***(TCP FIN scan)***: A ***FIN*** bit van beállítva a csomagban, vagyis a kapcsolat lezárását jelzi a célgépnek.
- `-sN` ***(TCP Null scan)***: Jelöletlen ***TCP*** csomagot küld a célkapura.
- `-sX` ***(Xmas scan)***: A ***FIN***, ***PSH***, ***URG*** jelzőbitek vannak beállítva a pásztázáshoz használt csomagokban.

Ez az utóbbi három pásztázási módszer ugyanarra az ***RFC 793***-ban leírt eljárásra támaszkodik, amelynek lényege, hogy a zárt portok ***RST*** választ küldenek, míg a nyitott portok eldobják a küldött csomagot.

- `-sA` ***(TCP ACK scan)***: Arra való, hogy feltérképezzük egy adott tűzfal szabályait. Ez a gyakorlatban úgy néz ki, hogy ha tűzfallal nem védett portokat pásztázunk, a nyitott és a zárt kapuk is ***RST*** csomaggal válaszolnak, amit az ***nmap*** a nem szűrt (unfiltered) csoportba sorol, függetlenül attól, hogy nyitott vagy zárt kapuról van szó. Azonban a tűzfallal védett portok vagy nem, vagy ***ICMP*** hibaüzenettel válaszolnak, így ezek a szűrt (filtered) csoportba sorolódnak.
- `-sW` ***(TCP Window scan)***: Hasonló, mint az ***ACK*** pásztázás, azzal a különbséggel, hogy a nyitott és a zárt kapukat képes megkülönböztetni egymástól.
- `--scanflags`: Mi adhatjuk meg, hogy milyen jelzőbiteket szeretnénk beállítani a pásztázáshoz használt csomagokban. Az ***URG***, ***ACK***, ***PSH***, ***RST***, ***SYN***, és ***FIN*** jelzőbitek használata a megengedett.

Pásztázni kívánt portok szűkítése

- `-p <port tartomány>`: Csak bizonyos portok vizsgálata. Megadhatunk konkrét portszámokat, de tartományokat is. Ezen kívül szűkíthetjük a vizsgálatot ***TCP*** vagy ***UDP*** kapuk szerint is. A ***TCP*** portok 21-től 80-ig, míg az ***UDP*** kapuk közül csak az 53-as ellenőrzése: `-p T:21-80,U:53`.
- `-F` ***(Fast (limited port) scan)***: Gyorsabb ellenőrzés azáltal, hogy az ***nmap*** csak azon kapukat vizsgálja, melyek szerepelnek az ***nmap-services*** fájlban.

2. Lista FIN, UDP és RCP pásztázás portszűkítéssel

```
csaba@lcs:~$ sudo nmap -sFUR
➥-p 20-25,53,68,110-140 lcs

Starting Nmap 4.20
➥( http://insecure.org ) at
➥2006-12-28 18:26 CET
Interesting ports on lcs
➥(192.168.1.1):
Not shown: 71 closed ports
PORT    STATE   SERVICE VERSION
21/tcp  open|filtered  ftp
22/tcp  open|filtered  ssh
25/tcp  open|filtered  smtp
139/tcp open|filtered
➥netbios-ssn
68/udp  open|filtered  dhcpc
137/udp open|filtered
➥netbios-ns
138/udp open|filtered
➥netbios-dgm

Nmap finished: 1 IP address
➥(1 host up) scanned in
➥4.865 seconds
csaba@lcs:~$
```

3. Lista A szolgáltatások verzióinak kiderítése

```
csaba@lcs:~$ sudo nmap -sSURV -P0 -T5 lcs

Starting Nmap 4.20 ( http://insecure.org ) at 2006-12-28 18:33 CET
Interesting ports on lcs (192.168.1.1):
Not shown: 3177 closed ports
PORT     STATE     SERVICE      VERSION
21/tcp   open      ftp          ProFTPD 1.3.0
22/tcp   open      ssh          OpenSSH 4.3p2 Debian 5ubuntu1
                                ➥(protocol 2.0)
25/tcp   open      smtp         Postfix smtpd
139/tcp  open      netbios-ssn  Samba smbd 3.X (workgroup: MSHOME)
445/tcp  open      netbios-ssn  Samba smbd 3.X (workgroup: MSHOME)
68/udp   open|     dhcpc
         filtered
137/udp  open      netbios-ns   Microsoft Windows XP netbios-ssn
138/udp  open|filtered netbios-dgm
Service Info: Host:  lcs; OSs: Unix, Linux, Windows

Service detection performed. Please report any incorrect results
➥at http://insecure.org/nmap/submit/ .
Nmap finished: 1 IP address (1 host up) scanned in 51.379 seconds
csaba@lcs:~$
```

4. Lista A számítógépen futó operációs rendszer kiderítése

```
csaba@lcs:~$ sudo nmap -sS -O -T4 -P0 scanme.nmap.org

Starting Nmap 4.20 ( http://insecure.org ) at 2006-12-28
➥18:27 CET
Interesting ports on scanme.nmap.org (205.217.153.62):
Not shown: 1691 filtered ports
PORT       STATE      SERVICE
22/tcp     open       ssh
25/tcp     closed     smtp
53/tcp     open       domain
70/tcp     closed     gopher
80/tcp     open       http
113/tcp    closed     auth
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.15-27-686 (Ubuntu Dapper, X86)
Uptime: 11.093 days (since Sun Dec 17 16:14:38 2006)

OS detection performed. Please report any incorrect results at
➥http://insecure.org/nmap/submit/ .
Nmap finished: 1 IP address (1 host up) scanned in 45.142
➥seconds
csaba@lcs:~$
```

Verzióinformációk

Ahogyan azt már említettem a cikk elején, attól függetlenül, hogy létezik egy szabvány arra vonatkozóan, melyik számú *TCP* illetve *UDP* kapun milyen szolgáltatás figyel, még nincs rá garancia, hogy a vizsgált számítógépen is hasonló a helyzet. Egy sima pásztázással csak annyit tudhatunk meg, hogy az adott kapu nyitva van-e vagy sem, de a mögötte lévő szolgáltatás rejtve marad.
Ehhez nyújt segítséget az `nmap -sV` kapcsolója, amely a szolgáltatások nevét és verziószámát hivatott kifürkészni. Itt említhető meg a `-sR` kapcsoló is, aminek a segítségével az úgynevezett *RPC* portok és a szolgáltatásaik verziószáma tudható meg. Ez a kapcsoló is használható együtt a *TCP* és *UDP* pásztázásokkal.

Az operációs rendszer felderítése

Az *nmap* az alapján, hogy a célgép bizonyos csomagokra milyen válaszcsomagot (ujjlenyomatot) generál, képes megállapítani az adott gépen futó operációs rendszer típusát, a kernelverzió számát, valamint az üzemidőt. Talán nem kell túlzottan részleteznem, mennyire megkönnyítheti egy támadó dolgát, ha tudja, milyen operációs rendszerrel van dolga. Ehhez mindössze egy nyitott és egy zárt kapura van szüksége a célrendszeren.
Az ehhez használt kapcsoló a `-O`. Ilyenkor a folyamat a második generációs felderítéssel kezdődik, majd annak sikertelensége esetén az első generációs is lefut. Kérhetjük a programot, hogy kifejezetten csak első vagy második generációs felderítést használjon. Erre valók az `-O1` illetve az `-O2` kapcsolók.
Előfordulhat, hogy az *nmap* nem tudja megállapítani a használt operációs rendszer típusát, ilyenkor a beérkezett ujjlenyomatot jeleníti meg a képernyőn. Ha segíteni akarjuk *Fjodor* fejlesztői munkáját, a ➲ http://insecure.org/nmap/submit oldalon elküldhetjük ezt az ujjlenyomatot a pásztázott rendszer pontos paramétereivel együtt.

Kimenet és egyéb érdekességek

Előfordulhat, hogy a pásztázás eredményét a későbbiekben még látni szeretnénk, ha másért nem, a rendszerbiztonság megerősítése utáni összehasonlítás végett. Ezen kívül ebbe a bekezdésbe kerültek azok a kapcsolók, melyek hasznosak, de nem tartoznak szervesen egyik tárgyalt témához sem.

- `-oN <fájlnév>`: A kimenet egy egyszerű szövegfájlba történő mentése.
- `-oX <fájlnév>`: A kimenetet *XML* formátumba menti.
- `--append-output <fájlnév>`: A fájlt nem írja felül, hanem hozzáfűzi az újabb kimenetet.
- `-v`: *Verbose*, azaz beszédes kimenetet biztosít. Kétszeri ismétlésére az *nmap* további részleteket árul el.
- `-T<0-5>`: A kapcsoló és a mögötte álló szám segítségével közölhetjük a programmal, milyen időközönként küldjön csomagokat a célgépnek. A 0 érték jelenti a legnagyobb időközt, ami 15 perc minden elküldött csomag után, míg az 5 a legkisebbet, ami 5ms várakozási időt jelent. *IDS* rendszerek ellen a 0 és az 1 érték a javasolt, míg a hétköznapokban a 4 tűnik jó választásnak.
- `-D <csalétek1, csalétek2, csalétek3...>`: A célgép szemszögéből úgy látszik, mintha a csalétek gépek pásztáznák a portjait.
- `-6`: Ipv6 protokoll használata. Csak a Ping scan (`-sP`), TCP Connect() scan (`-sT`) és List scan (`-sL`) esetén működik.
- `-A`: Operációs rendszer detektálás és verzióinformációk lekérése. Ugyanaz, mint a `-sV -O` páros. *Fjodor* a későbbiekben tervezi ennek a kapcsolónak a további szolgáltatásokkal történő bővítését, azért, hogy a felhasználóknak ne kelljen annyi kapcsolót megjegyezniük.

Összegzés

Az *nmap* további, olykor megdöbbentő képességeinek felfedezését már az Olvasóra bízom. Remélem a fentiekből remekül kitűnt az az összetettség és rugalmasság, amit az *nmap* kapupásztázó program magában hordoz, és amivel elősegíti számítógépeink hálózati biztonságának megerősítését. Forgassuk nagy haszonnal és megelégedéssel, de ne feledjük, csak jó célra!

Leszkoven Csaba
(leszkovencsaba@gmail.com)

Mahjongg, a türelemjáték

Miután meggyőztük kedvenc Titkárnőnket, miszerint a Linux és az Openoffice sokkal jobb – sőt fel is telepítettük neki –, nem árt ajánlani valamit a Passziánsz helyett, amivel a kávészünetet eltöltheti.

Noha a játék távolkeleti eredetű, a számítógépes *Mahjongg* szabályok nem egyeznek teljesen az eredetivel. (Az eredeti játékszabályokról a *Wikipedia*ban olvashat a kedves Olvasó.) Maga a *Mahjongg* legalább annyira izgalmas játék, mint *Windows* alatt a *Passziánsz*. Sőt még izgalmasabb, ugyanis sokkal több döntési lehetőség van, és előfordul, hogy több döntési lehetőséggel is megnyerhetjük a játékot, vagy a játék közepén hozott, akkor jónak tűnő, de rossz döntés okoz a legvégén bukást.

Cél, szabályok

A játék célja: levenni az összes kockát a játéktábláról. Ez pár egyszerű szabály észben tartásával hajtható végre, ha szerencsénk és jó taktikai érzékünk van. Egy kocka csak akkor vehető le, ha balról vagy jobbról nincs mellette másik kocka. A kockák levételéhez két egyforma és levehető kockát kell kijelölnünk az egérrel. Ahol több kocka van egymáson, ott mindig csak a felső kockát vehetjük le. Minden kockából 4-4 egyforma darab van, kivéve az évszakokat jelző kockákból, illetve az *orchid*, *mum*, *bamboo*, *plum* feliratúakból.

A játék telepítése egyszerű, hiszen a legtöbb disztribúcióban elérhetőek a *Mahjongg*-ot (is) tartalmazó csomagok. Általában *xmahjongg*, *gnome-games* (*Gnome* verzió), illetve *kmahjongg* (*KDE*) név alatt találjuk meg őket. Noha mindegyik felkínálja a klasszikus elrendezésű *Mahjongg*-ot, a *kmahjongg* felajánl táblaszerkesztőt, illetve a *Gnome mahjongg*-ja pár előre definiált alakzatot.

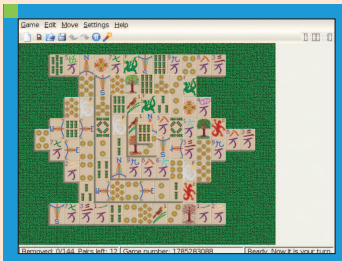
Az *xmahjongg* kivételével mindegyik támogatja a kocka kinézetének (*skin*ek) váltását.

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.

1. ábra Mahjongg X alatt

2. ábra Mahjongg Gnome alatt



3. ábra Mahjongg KDE alatt

Videótömörítés Linuxon, még hatékonyabban

Linux alatt is készíthetünk jó minőségű, kis helyet foglaló H.264-es tömörítésű videót. A digitális videózásban egyre nagyobb szerepet kap a H.264. Apple és a hozzá hasonló cégek is szívesen használják. Az MPEG-4-es szabvány részeként (part 10) a H.264 bekerült a nagyfelbontású HD-DVD és a Blu-ray szabványokba is. Ez annak köszönhető, hogy a H.264 kis sávszélesség mellett is hihetetlenül jó minőséget képes nyújtani.

■ Az alacsony sávszélesség igény teszi számunkra érdekessé ezt a kódolási eljárást. Szerencsére ***Linux***on megjelent az ***x264***, amely sikeres és hatékony nyílt forrású megvalósítása a ***H.264***-nek, mely ***Advanced Video Codec (AVC)*** néven is ismeretes. Valójában az ***x264*** projekt megnyerte a ***Doom9*** 2005-ös kodek tesztjét is. Az ***x264*** egy jelenleg is aktív projekt, melyet folyamatosan fejlesztenek. A kiemelkedő minőségű ***AVC kodek*** előnyeit számtalan helyen élvezhetjük kezdve a házi videógyűjteményünk biztonsági másolataitól, a webes videótovábbításig, de érdekes lehet csupán kísérletezgetni a legújabb technológia nyújtotta lehetőségekkel.
Jelen cikk célja bemutatni, hogyan készíthetünk egyszerű lépésekkel ***.mp4*** állományt, mely ***H.264***-es videóval és ***AAC*** (***Advanced Audio Codec***, szintén ***MPEG*** szabvány része) hanggal rendelkezik. Azonban egy összetett és átfogó cikk messze meghaladná a rendelkezésre álló terjedelmet, azonban ez remélhetőleg nem riasztja el az Olvasót a téma mélyebb tanulmányozásától.
Minthogy az ***AVC*** és az ***AAC*** is az ***MPEG*** szabvány része, így jó pár segédeszköz (legyen az kereskedelmi vagy másmilyen) támogatja ezeket. Például az ***Apple QuickTime*** programjával is készíthetünk ilyen videókat, illetve a népszerű nyílt forrású ***Mplayer*** is használható ***.mp4*** fájlok lejátszására.

Vágjunk bele

A célként kitűzött videófájl létrehozásához három egyszerű lépést kell végrehajtanunk: a tömörített videó létrehozása, a tömörített hang létrehozása és végül, de nem utolsósorban a két fájlt egyesítenünk kell megfelelő módon. Ehhez az alábbi programok szükségesek:

- ***MPlayer*** (tartalmazza az ***mencoder***-t, ***CVS*** verzióból ***060109*** vagy frissebb szükséges)
- ***faac*** 1.24 vagy frissebb
- ***MP4Box*** (***gpac*** része, 0.4.0 vagy frissebb szükséges)
- ***x264*** (***gpac*** támogatással fordítva)

A célunk: létrehozni egy kis sávszélességű videót, mely a weben könnyen továbbítható. Kis méretű lesz, de reményeink szerint hasonló minőségű, mint egy nagyobb sávszélességű ***XviD***-el tömörített videó. Forrásként egy kilenc másodperces ***raw*** formátumú házi videót (`max.dv`) használunk, melyet a digitális videókameráról mentettünk le.
Kezdjük a hanggal, mely átalakítása igencsak egyszerű művelet. Az ötlet: nyers videóból kinyerhetjük ***Mplayer***rel a hangot:

```
mplayer -ao pcm -vc null -vo
➥null max.dv
```

A művelet végén kapunk egy állományt, melynek neve `audiodump.wav` lesz. A videóval egyelőre nem foglalkozunk. Alakítsuk át ***AAC*** formátumra:

```
faac --mpeg-vers 4
➥audiodump.wav
```

Az `--mpeg-vers` kapcsolóval jelezhetjük az ***MPEG*** verziót. A végeredményt le is ellenőrizhetjük: játsszuk le az `audiodump.aac` fájlt ***Mplayerrel***.
A videó tömörítésekor több megoldás közül választhatunk. A legjobb minőség érdekében többmenetes tömörítést kell alkalmaznunk. A videót legalább kétszer kell feldolgozni ahhoz, hogy a rendelkezésre álló sávszélesség optimálisan legyen elosztva a videón. A többmenetes tömörítéssel hajszálpontosan beállítható a sávszélesség és a fájlméret. Sajnos azonban az ***AVC*** tömörítőprogramok, mint amilyen az ***x264*** is, meglehetősen igénybe veszik a processzort és emiatt hosszú a feldolgozás is. Éppen ezért nem várható el, hogy a többmenetes tömörítésre várjon az Olvasó. Ehelyett lehetőség van egymenetes tömörítésre is. Noha ez is kiváló minőséget ad, sose lesz olyan jó, mint a többmenetes. Az egymenetes tömörítéssel a fájlméret és a sávszélesség pontos beállításának lehetőségét is elvesztettük. Azonban lehetőség van mérlegelni, melyik a fontosabb: a gyorsaság vagy a minőség. Szerencsére az ***x264*** kínál megfelelő középutat is. Egy opcióval beállítható

a fix sávszélesség (vagy fix minőség), így az ***x264*** számításba veszi a mozgalmas és a kevésbé mozgalmas részek közötti különbséget. Minthogy az emberi szem a gyors mozgásoknál nem képes a részletekre figyelni, így a tömörítés során a mozgalmas részeken nyert bitek átcsoportosíthatóak máshova. Így összességében élvezetesebb lesz a végeredmény. Az opció alkalmazásával érhető el a legjobb minőség anélkül, hogy az időrabló többmenetes módot kellene használnunk.
Ennek a megoldásnak azonban az ára, hogy nem tudjuk előre megmondani a végleges fájl méretét és sávszélességét. Természetesen kétmenetes módban ez továbbra is lehetséges, de ezzel megduplázzuk a feldolgozáshoz szükséges időt. A példánkban tehát maradjunk az egymenetes feldolgozásnál, azonban állítsuk be a ***Constant Rate Factor*** (`--crf`) lehetőséget a megfelelő minőségért. ***18*** és ***26*** közötti ***Constant Rate Factor*** érték már általában megfelelő (a kisebb érték jobb minőséget, de nagyobb fájlt eredményez).
A megfelelő érték természetesen függ a kívánt mérettől, minőségtől és a rendelkezésre álló időtől. Többmenetes tömörítés esetén azonban nagyobb szabadságot kapunk.
Az ***x264*** csak nyers ***YUV 4:2:0*** bemenetet fogad, ehhez egyszerűen irányítsuk az ***mencoder*** kimenetét az ***x264*** bemenetére egy egyszerű csővezetékkel (pipe):

```
mkfifo tmp.fifo.yuv
mencoder -vf format=i420
➥-nosound -ovc raw -of
➥rawvideo \
    -ofps 23.976 -o
    ➥tmp.fifo.yuv max.dv
    ➥2>&1 > /dev/null &
x264 -o max-video.mp4 --fps
➥23.976 --crf 26 --progress \
    tmp.fifo.yuv 720x480
rm tmp.fifo.yuv
```

Amint látható, meg kell adnunk, hogy hány képkockát szeretnénk egy másodperc alatt lejátszani (`--fps`), ugyanis az ***x264*** nem fogja magától kitalálni. Ugyanígy meg kell adnunk a videó felbontását is. Jelen példánkban az ***x264*** alapértelmezett beállításaival tömörítettünk, ami elég jó minőséget ad, azonban tovább finomíthatunk rajta. Jelen esetben kicsit módosítva a tömörítési stratégiát, javíthatunk a dolgon anélkül, hogy ez lényegesen több időt venne igénybe. Az ***x264*** rengeteg paraméterrel bír, mellyel tovább javíthatunk a minőségen ilyen vagy olyan módon. Természetesen némely opció alkalmazása költségesebb, időigényesebb, mint másoké. Végül pedig pár opció alkalmazásával elfordulhat az is, hogy bizonyos lejátszók nem fogják tudni lejátszani a videót, nevezetes például a ***QuickTime***.
A kompatibilitás érdekében ezeket tartsuk szem előtt.

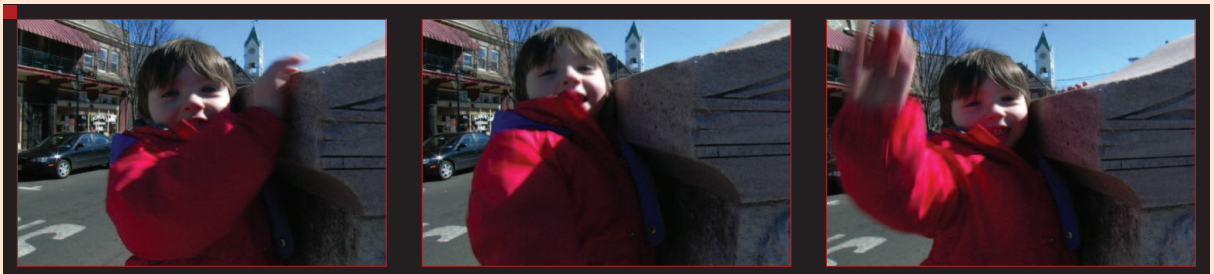
QuickTime és a H.264

A ***QuickTime 7***-es verziója már támogatja a ***H.264***-es videókat. Az ***Apple*** az interneten elérhető mozielőzeteseket például már ***H.264*** szerint tömöríti. Noha ez jó, és elősegíti a kodek terjedését, pár korlát is került a ***QuickTime*** megvalósításába, melyek közül legjelentősebb a ***B-képkockákkal (B-Frames)*** és a ***Profil*** támogatással kapcsolatos. Teszünk egy rövid kitérőt, hogy lássa az Olvasó, mit is jelent ez számunkra.
A ***H.264*** többféle profillal rendelkezik, ilyen a ***Baseline***, a ***Main***, az ***Extended*** és a ***High***. Ezek a profilok leírják, hogy az adott videó lejátszásához milyen képességek szükségesek a lejátszó részéről. Ahogy az sejthető, a ***Baseline*** a legegyszerűbb profil, míg a ***Main***, az ***Extended*** és a ***High*** több számítási kapacitást igényel és technológiailag is bonyolultabb lejátszani. A ***QuickTime 7*** csak a ***Baseline***-t támogatja teljesen, illetve a ***Main***-t részben, az ***Extended***-del és a ***High***-al nem foglalkozik.
A ***B képkocka (B-Frames)*** egy megoldás a videók tárolására. Az ilyen képkockák dekódolásánál létfontosságú a korábbi képkockák hibátlan dekódolása. ***B képkockákat (B-Frames)*** más képkockákkal fésülnek össze (a ***B képkockák*** csupán a változásokat tárolják), ilyenek például az ***I (I-Frames)*** és a ***P képkockák (P-Frames)***. Technikai részlet, de a ***QuickTime 7***-ben alkalmazott ***H.264*** megvalósítás csupán 2 darab ***B képkockát*** enged. Ez nem túl jó, hiszen több ***B képkocka*** alkalmazásával bizonyos körülmények között javítható a képminőség.
Ezt a korlátozást tartsuk észben, hogy ***QuickTime*** kompatibilis maradjon a videónk. Ez a korlátozás természetesen nincs jelentős kihatással a végeredményre. Pár opcióval tovább javíthatunk a dolgon. Ilyen például a pixel elmozdulás becslési pontossága (`--subme`), mellyel beállítható, mennyire legyen pontos pixel elmozdulásának becslése ***x264***-es tömörítés közben. Ezt a 6-os maximumra állítva lényegesen szebb végeredményt kapunk, és noha hosszabb lesz a tömörítés, megéri a többlet idő. A minőség javítása érdekében azt is megadhatjuk, hogy az ***x264*** hogyan elemezze a képkockákat (`--analyse`). Jegyezzük meg: van olyan mód, amely csak ***High*** profil esetén érhető el – ilyen például a ***8x8 DCT*** – , így erről le kell mondanunk a ***QuickTime*** kompatibilitás érdekében. A ***PSNR*** kikapcsolásával gyorsíthatunk a tömörítésen (`--no-psnr`) anélkül, hogy bármi különbség látszana a videónkon.
Mindent összevetve most már mindent tudunk ahhoz, hogy létrehozzunk jó minőségű, kis sávszélességű ***Quicktime*** kompatibilis ***H.264*** kódolású videót:

```
mkfifo tmp.fifo.yuv
mencoder -vf format=i420
➥-nosound -ovc raw -of \
    rawvideo -ofps 23.976 -o
    ➥tmp.fifo.yuv \
    max.dv 2>&1 > /dev/null &
x264 -o max-video.mp4 --fps
➥23.976 --bframes 2 \
    --progress --crf 26
    ➥--subme 6 --analyse \
    p8x8,b8x8,i4x4,p4x4
    ➥--no-psnr tmp.fifo.yuv
    ➥720x480
rm tmp.fifo.yuv
```

Tovább finomíthatjuk a végeredményt. Minthogy a webre szánjuk a videónkat, így például nem hátrány ha kisebbre vesszük a képkockákat és levágjuk a nem kívánt részeket. A képkocka átméretezése például az alábbi paranccsal valósítható meg:

```
mkfifo tmp.fifo.yuv
mencoder -vf scale=480:320,
➥format=i420 -nosound -ovc \
    raw -of rawvideo -ofps
    ➥23.976 -o tmp.fifo.yuv \
    max.dv 2>&1 > /dev/null &
```



1. ábra DV

2. ábra XviD

3. ábra x264

1. táblázat *Eredmények*

Fájl	Fájl méret	Sávszélesség
max.dv	*32Mbyte*	*3MByte/s*
max-xvid.avi	*623Kbyte*	*418kbit/s*
max-x264.mp4	*522Kbyte*	*392kbit/s*

```
x264 -o max-video.mp4 --fps
➥23.976 --bframes 2 \
    --progress --crf 26
    ➥--subme 6 --analyse \
    p8x8,b8x8,i4x4,p4x4
➥--no-psnr tmp.fifo.yuv
➥480x320
rm tmp.fifo.yuv
```

Az *mencoder* 480x320 képpont felbontású videót készít, így az *x264*-nek is meg kell adnunk a helyes méretet. Ez a művelet csökkentette a videó méretét, mely a gyorsabb letöltés miatt előny.

Utolsó lépések

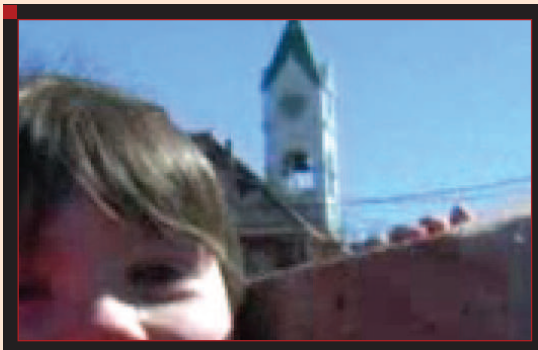
Az *.mp4* formátum (melyhez alapul a *QuickTime* szolgált) többféle médiaformátumot (hang, videó) tárolhat, így ideális számunkra a *H.264*-es videó és *AAC* hang tárolására is. Az *MP4Box* segítségével – mely a *gpac* projekt része – a korábbiakban elkészített két állományt egyesíthetjük:

```
MP4Box -add max-video.mp4 -add
➥audiodump.aac \
   -fps 23.976 max-x264.mp4
```

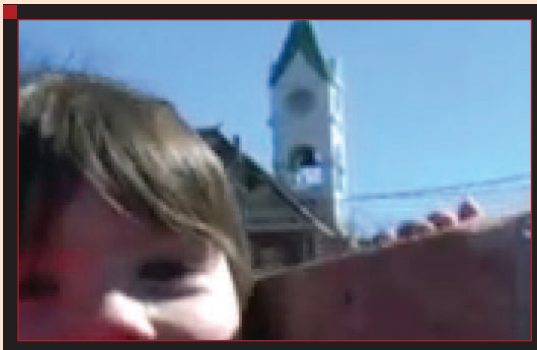
Létrehoztuk a végleges `max-x264.mp4` állományt, melyet le is játszhatunk *Mplayer*rel, vagy *nem linuxos* gépeken az *Apple QuickTime* lejátszójával. A videónk weboldalba is beilleszthető. Ezt például *Linux* alatt *Firefox*ból a telepített *mplayer-plugin* segítségével tekinthetjük meg. Összehasonlítás végett az állományok méretei és sávszélességei nyers *DV*, a tárgyalt *H.264* és az alábbi parancssor generálta *XviD* szerint:

```
mencoder max.dv -vf
➥scale=480:320 -ovc xvid
➥-xvidencopts \
   fixed_quant=7:qpel:nopacked
   ➥-oac mp3lame \
   -ofps 24000/1001 -o max-
   ➥xvid.avi
```

És minden videóból egy-egy képkocka: *1-3. ábra.*



4. ábra XviD részletek



5. ábra x264 részletek

Ahogy látható, a ***H.264*** éppolyan jó minőséget nyújt, mint az ***XviD*** (talán jobbat is), de mindezt kisebb sávszélesség igény és méret mellett. Belátható, hogy ugyanolyan minőséghez kisebb tárhely is elég, vagy ugyanakkora tárhelyen jobb minőséget érhetünk el. Emellett a munkamenet és a beállítások hasonlóak az ***XviD***-hez, viszont jobb minőséget ad a ***H.264***. Tehát ha már tömörített az Olvasó ***XviD***-del, az ***x264*** se fog meglepetést okozni.
Minél többet kísérletezik az ember az ***x264***-el, annál jobban elcsodálkozik, mekkora megtakarítás érhető el a minőség megtartása mellett. A videó tömörítés kétségkívül egyfajta varázslat, hiszen több száz opció sokféle értéke befolyásolja a végeredményt. A videótömörítéshez nincs általánosan használható eljárás vagy ajánlott paraméterek, minden videó más és más. A ***H.264*** technológiai fölénye az ***XviD***-el vagy ***MPEG2***-vel szemben elég jelentős ahhoz, hogy figyelmen kívül hagyjuk. Már ma élvezhetjük az előnyeit. Tekintve, hogy a ***H.264*** kodek ***MPEG*** szabvány, így a befektetett munka végeredménye időtálló és jó minőségű lesz. Jelen cikk talán segített rávilágítani arra, miért lesz a jövő tömörítési formátuma a ***H.264***.

Linux Journal 2006., 150. szám

Dave Berton profi programozó, a mosey@freeshell.org címen érhető el.

KAPCSOLÓDÓ CÍMEK

A cikk forrása:
➲ www.linuxjournal.com/article/9197

Neverball – Ahol golyó még sosem járt

Ritkán ejtek szót hamisítatlan ügyességi játékokról. Most pótolom a hiányosságot: rögtön egy olyan darab következik, mely egyszerre klasszikus és különc.

A „8 bites korszak” derekán találkoztam először olyan ügyességi játékokkal, ahol a „főszereplő” egy labda volt, melyet adott időn belül el kellett juttatni a (bonyolult) pálya kijelölt pozíciójába. Ebben az időben – a nyolcvanas évek végén – két elképzelésen alapult e kategória: akadt olyan játékprogram, ahol a gép (jellemzően a felhasználó felé) görgette a szabálytalanul kilyuggatott talajt, a játékos pedig a golyót mozgatta jobbra-balra. A másik ötlet szerint a terepet mozgatta az érdeklődő, ennek megfelelően gurultak az azon lévő labdák. Személyesen az első megoldást jobban szerettem, viszont ez a stílus az idők folyamán sajnos teljesen elhalt. A másik elképzelés viszont él és virul, egyik jeles képviselője *GPL* licencű forráskóddal rendelkezik. A *Neverball* névre keresztelt Szabad projekt naprakész, kihívó és szórakoztató.

1. ábra A Neverball, KDE asztalon

2. ábra A feladat igen összetett

Miről is van szó?

Dióhéjban arról, hogy a terepet minden irányban megdönthetjük, az arra helyezett golyó pedig a dőlésnek megfelelően kezd gurulni. A feladat a következő: a labda mozgását úgy kell koordinálni, hogy ne essen le a terepről, eközben megszabott mennyiségű zsetont kell begyűjteni a pályáról – úgy, hogy adott időn belül elérjük a kijáratot. Elsőre talán nem hangzik túl bonyolultnak a kombináció, a gyakorlati tapasztalat azonban teljesen ellentmond minden lenéző legyintésnek... A *Neverball* komoly kihívást jelent, a nyomasztó nehézségét pedig a naprakész *3D* grafikai leképezése, valamint az élethűnek mondható fizikája próbálja ellensúlyozni.

A telepítés

A forráskódot az *Icculus* csapatának honlapjáról indulva lehet elérni: látogassunk el a ➲ *http://icculus.org/neverball* címre! Töltsük le *neverball_verzió.tar.gz* csomagot, majd a kicsomagolt archívban adjuk ki *root* jogkörrel a `make` parancsot.

3. ábra Akad pálya, bőven!

Ha a fő függésekkel rendelkezünk (*libSDL*, *SDLImage*, *SDL_mixer*, *SDL_ttf*, *libGL*, *libPNG*), akkor rövidesen felépül az *ELF* bináris. A teljes forrás mappát másoljuk egy mindenki által elérhető területre, a létrejött indítható fájlt pedig linkeljük ki egy elérési útra *neverball* néven. Ha mindennel végeztünk, felhasználóként kiadott

```
neverball
```

parancsra lehet a játékba lépni. A teljesített pályák szerencsére meg vannak jelölve a */home/$/.neverball* úton (egy speciális fájlban), így adott felhasználónak nem kell mindig elölről kezdeni a próbákat. Teszem azt, bárki nyithat a tizedik pályán – feltéve, hogy valamikor már teljesítette az előző kilencet. Szükség is van erre, hiszen a három nehézségi fokozatra osztott szintek 75 terepet biztosítanak: „fél délelőtt" sem elég a teljes végigjátszáshoz.

Használható trükkök

Először is a kamera mozgatása érdemel néhány szót. Az aktív nézőpontváltoztatás kissé nehezen szokható meg, segítségképpen az automatika az *„F3"* billentyűvel kézi rögzítéssel leállítható. Ehhez viszont nagy rutinra van szükség, mivel a talajt mozgatni és a kameraállást egy időben kezelni (az egérgombokkal) nem könnyű feladat. Tehát marad az automatika megszokása...
A továbbjutáshoz szükséges pontok száma a jobb alsó sarokban van kijelezve. A pályán található zsetonok közül a sárgák egyet, a pirosak négyet, a kékek pedig tízet érnek: érdemes megbarátkozni ezekkel a számokkal, és kezdetben csak az értékesebbeket összegyűjteni. A szükséges mennyiség elérését egy csilingelő hang jelzi, ekkor indulhatunk a kijárat felé. Amennyiben képesek vagyunk összegyűjteni száz zsetont, úgy ajándék élethez juthatunk! A terepen gyakran „kapcsolók" is találhatóak: ezek az érintési pontok felelnek a mozgó pályaelemek aktiválásáért (például liftek, csúszó felületek és teleportok).
Ha valaki úgy érezné, hogy neki már nem kihívás a *Neverball*, akkor javaslok egy házi bajnokságot rendezni a projekt köré. Amennyiben nincs erre lehetőség, akkor sem nagy a baj, hiszen bármely teljesített feladatot rögzíteni lehet a *„Replay"* menüt használva – így a kételkedők is megbizonyosodhatnak a büszke játékos teljesítményéről.
Jó „gurigázást"!

Kovács Zsolt (kovi@linuxforum.hu)

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

KSudoku

A Sudoku ősét Euler találta ki, de mint minden más azóta ez is modernizálódott. A mai formáját Maki Kaji alkotta meg az 1980-as években. Az első Sudoku programokat Wayne Gould-nak köszönhetjük...

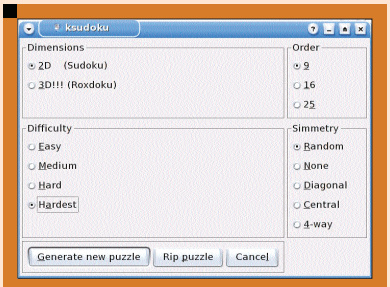
A The Times jelentette meg elsőként 2004. november 12-én újságban a rejtvényrovatban. Annyira megtetszett az olvasóknak, hogy 2005 nyarára a legjelentősebb amerikai napilapok is megjelentették a saját bűvös négyzeteiket. 2005. októberében pedig az első bajnokságot is megrendezték Angliában.

Szabályok

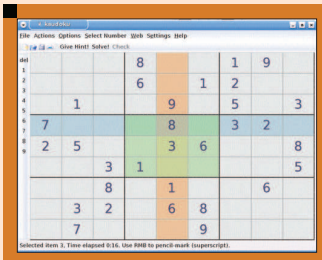
Töltsünk fel egy 9x9-es négyzethálót az 1-9 számjegyekkel úgy, hogy minden sorban és oszlopban illetve minden 3x3-as zónában a számok csak egyszer szerepeljenek.

KSudoku

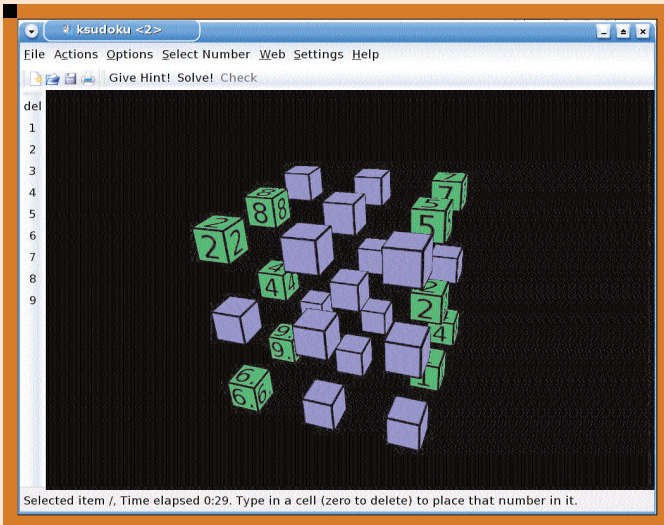
Mivel rendszeresen megfejtem a ***Metroban*** megjelenő ***Sudokukat***, nagyon megörültem annak, hogy a ***KDE*** oldalán ráleltem erre a gyöngyszemre. A program indításánál kell eldönteni, hogy milyen bűvös négyzetet szeretnénk kitölteni.
Kiválaszthatjuk a nehézségi szintet, a szimmetriát, és a négyzet nagyságát, ami 3x3 - 5x5 lehet. Megjelentethetjük 2D-s vagy 3D-s formában. Miután beállítottunk mindent, el kell készíttetni az új táblát. Kezdőknek ajánlatos 3x3-as táblát választaniuk. Haladók már próbálkozhatnak a 4x4 vagy 5x5 táblák kitöltésével is.
Egy 5x5 tábla elkészítése 3D-s megjelenítéssel eltarthat egy darabig, de megéri elkészíteni legalább egyszer, még ha nem is tudunk vele játszani, mert a látvány elképesztő.
Színösszeállítása teszi tetszetőssé, viszont a pasztellszíneinek köszönhetően a szemünk sem fárad el olyan könnyen. Könnyű kezelhetősége csak ráadás. Automatikusan kijelölődik a sor, az oszlop, és a 3x3-as zóna. Bevitelre számos lehetőségünk van. Vagy a sor és oszlop metszéspontjába írjuk be a számjegyet, vagy a bal oldalon lévő illetve a már beírt számok közül kiválasztunk egyet és egérrel visszük be a táblába a számokat. Ha egy számot rossz helyre írjunk be, akkor a színe pirosra vált, ami nagyban megkönnyíti a tábla kitöltését.



1. ábra Új Sudoku készítése



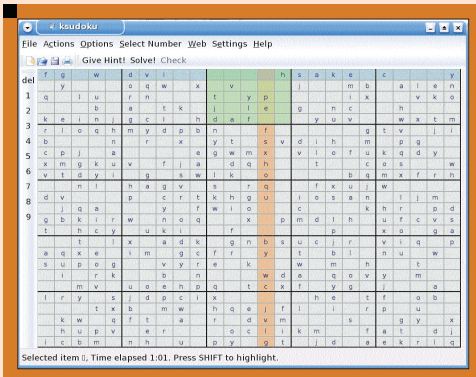
2. ábra Egyszerű 3x3 tábla 2D-s megjelenéssel



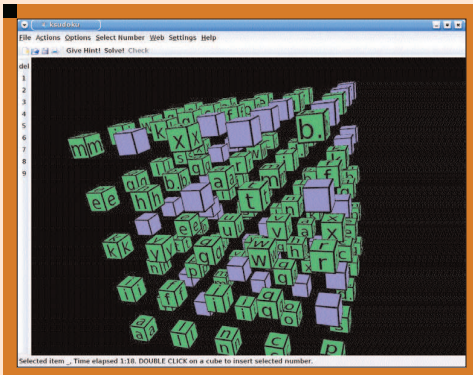
3. ábra Egyszerű 3x3 tábla 3D-s megjelenéssel

Egyéb lehetőségek

A *KDE* oldalán találunk még egy ***Sudoku*** programot, (***Su-per-Doku***) igaz ***superkaramba*** témaként.



4. ábra 5x5 tábla



5. ábra 5x5 tábla 3D-s megjelenítéssel – igazán extrém

Szerencsére még a superkarambát sem kell letöltenünk és feltelepítenünk abban az esetben, ha a rendszerünkön már legalább ***KDE 3.5.0***-van, ugyanis ettől a verziótól része a ***superkaramba*** a ***KDE***-nek.
A másik lehetőség az on-line játék (lásd a kapcsolódó címeket). Szintén érdekes lehetőség a nyomtatott forma. Léteznek ***Sudoku*** kiadványok, illetve napilapokban kereshetjük őket. Ha agyilag le vagyok strapálva, nagyon jó kikapcsolódást tud nyújtani.

Szurmai László (laszlo.szurmai@hu.bosch.com)

4 éve foglalkozom Linuxszal. Leginkább az adatbázis-kezelés érdekel, ezzel foglalkozom már 7 éve hol mellék hol főállásban. Asztaliteniszezem a Kispest III csapatában.
Játszani a Diablo II -vel a Battle.net-en szoktam.

KAPCSOLÓDÓ CÍMEK

➲ www.websudoku.com, ➲ www.jigsawdoku.com, ➲ www.su-do-ku.org, ➲ www.chevrolet.hu