Beköszöntő

Ahogy azt ígértük...

Megváltozunk. Ígérem. Na, nem úgy, ahogy Bajor Imre ígérte tíz éve egy kabaréban, hogy új életet kezd, de mivel a piát szereti, ezért az új életében is inni fog. Mi inkább azon tulajdonságainkat igyekszünk átmenteni, melyeket olvasóink is a lap értékének tartanak.

Az elmúlt időszakban folytatott piackutatások és olvasói levelek alapján három főbb változási cél rajzolódott ki előttünk. Olvasóink szerint:

- A cikkek túl tömörek, nehezen olvashatóak
- Kevés a kezdőknek, próbálkozó kedvűeknek szóló cikk
- Több hazai vonatkozású, olvasmányos cikkre van igény

A tervek megvitatása közben még egy gondolat folyamatosan felvetődött: valahogy jobban be szeretnénk vonni a hazai szakembereket a Linuxvilág szerkesztésébe. Ezzel egyrészt egy érdekesebb, színesebb anyag állhat össze, másrészt pedig lehetőséget is biztosítunk egymás megismerésére. Az új „nyílt" szerkesztés jegyében kialakítottunk tehát egy weboldalt (➲ linuxvilag.hu/szerzoknek), ahol bárki jelentkezhet, aki szívesen írna cikket, vagy elmondaná, hogy milyen cikket látna örömmel az újságban. A cikkírók itt további anyagot is találnak a leadandó cikkekkel kapcsolatban.

Reményeink szerint a szeptemberi számtól már egy teljesen új arculattal indul, mindhárom célterületen változva, olvasóink igényéhez jobban alkalmazkodva. Mint mindig, most is örömmel várunk bármilyen véleményt, ötletet, kritikát!

A „mostani generáció" utolsó két lapszámához is kellemes olvasást kívánok!

Szy György
főszerkesztő

Kütyüimádóknak

A *Nokia* újfajta, internet tábla névre keresztelt mobil eszközt mutatott be. A teljes nevén *Nokia 770 Internet Tablet*

készülék internetezésre, elektronikus levelek kezelésére, médiafájlok megtekintésére és lejátszására, *RSS* hírcikkek olvasására, internetes rádiózásra használható. Operációs rendszere a nokiás hagyományokkal szakítva *Linux* alapú, kijelzője 800x480 képpont felbontású, a hálózathoz *Wi-Fi* és *Bluetooth* összeköttetésen keresztül képes csatlakozni, az adatbevitelt pedig a képernyőn megjelenő billentyűzettel teszi lehetővé. A tábla szoftverét a *Nokia* rendszeresen frissíteni tervezi, az első megújulást jövő évre tervezik, amikor *IP* feletti hangtovábbítási és azonnali üzenetküldési képességgel ruházzák majd fel a készüléket. A sokoldalúsága ellenére inkább szórakoztató haszontalanságnak tűnő tábla *Amerikában* és *Európában* a harmadik negyedévben jelenik meg, egyelőre ismeretlen áron.
➲ www.nokia.com

Van képzeletük

Az *AMD* véglegesítette és nyilvánosan is elérhetővé tette *Pacifica* virtualizációs megoldásának leírását. A célközönséget elsősorban a programozók alkotják, akik így talán kellő időt kapnak arra, hogy alkalmazásaikat képessé tegyék az először 2006 első félévében megjelenő *AMD x86* processzorok által támogatott technológia nyújtotta lehetőségek kihasználására. A hardveres virtualizáció az *AMD* ígéretei szerint a kiszolgálókba és az ügyfélgépekbe szánt lapkák esetében egyaránt elérhető lesz, bár használatához nem lesz elég új processzort vásárolni, új alaplapra és lapkakészletre is szükség lesz. A fejlesztésnek további lendületet adhat, hogy *Vanderpool* néven, ugyancsak jövőre az *Intel* is hasonló képességekkel fogja felruházni *Itanium* és *Xeon* processzorait, és a két megoldás nagymértékben hasonlítani fog egymásra.
➲ http://www.xbitlabs.com/news/video/display/20050519225638.html

Táblájuk még nem volt

Az *IBM* PC-s üzletágát nemrég átvevő *Lenovo* bemutatta az első *ThinkPad* tábla PC-t. Az *IBM ThinkVantage* megoldásait a terméksorozat egyéb tagjaihoz hasonlóan támogató táblagép pontos jelölése *ThinkPad X41 Tablet*, alapjául az *Intel Centrino* platformjának legújabb, *Sonoma* kódnevű változata adja. A gépbe alacsony és ultraalacsony feszültségű *Intel* processzorok és *DDR2 RAM* kerül, továbbá természetesen a *802.11 a/b/g* csatoló sem maradhat el belőle. A 12”-os kijelzőjű gép súlya 1,58 kg lesz, akkumulátoros üzemideje elvileg a 8,5 órát is elérheti majd. Operációs rendszere a *Windows XP Tablet PC Edition* 2005 lesz, ára pedig 1899 dollártól indul.

Legközelebb milyen mosóport vásárol?

A *VeriFone* beágyazott *Linux* multimédiás *PIN*-táblát mutatott be.

A készülék mostantól ősinek számító példányai mindannyiunk számára ismerősek az üzletekből: kártyás vásárlásnál ezt a kisméretű billentyűzetet nyomják a kezünkbe, hogy írjuk be a *PIN*-kódunkat, majd „nyomjuk meg a zöldet”. Az *MX870* a kor színvonalának megfelelően támogatja a rádiós azonosítókat, az intelligens kártyákat és a biometrikus azonosítást, színes érintőképernyője pedig alkalmas arra, hogy a fizetéssel bíbelődő vásárlónak multimédiás hirdetéseket jelenítsen meg. Az élmény fokozását egy 5,7”-es, 240 x 320 képpontos, 65000 szín megjelenítésére képes kijelző, egy 200 MHz-es *ARM* processzor, valamint 16 MB RAM és 32-128 MB *Flash* memória biztosítja – mindezekhez kiegészítő jelleggel beépített, térhatású hangszórókat is lehet kérni.
➲ www.verifone.com

Titkos biztonság

Minden adatra kiterjedő, hardveres titkosítást végző, hordozható számítógépekbe és külső meghajtókba szánt merevlemezt mutatott be a *Seagate*.

Seagate
We turn on ideas

A szintén nemrég megjelent, 2,5”-os *Momentus* termékcsalád 5400-as tagjaira épülő meghajtó a lemez egy elkülönített területén egy titkosító kulcsot tárol, ennek alkalmazásával minden a csatolófelületen keresztülhaladó adatot automatikusan, a felhasználó további beavatkozása nélkül titkosít. A kulcs megadása nélkül a lemezen tárolt adatokhoz még a meghajtó gépből való kiszerelésével sem lehet hozzáférni, illetve az operációs rendszer elindítására sincs lehetőség. Az új meghajtóval elsősorban az üzleti felhasználókat célozzák meg, hiszen a vállalkozásoknak sokszor bizalmas adatokat kell hordozható gépeken tárolniuk, amelyek viszont erősen ki vannak téve az ellopás veszélyének. Természetesen az adatok titkosítása eddig sem volt megoldhatatlan feladat, ám azt általában szoftveresen vagy valamilyen kiegészítő eszközzel végezték; ez az első példa a titkosítási szolgáltatás merevlemezes meghajtóba építésére. Ha a megoldás sikeresnek bizonyul, akkor elképzelhető, hogy hamarosan a Seagate egyéb termékeiben is megjelenik.
➲ www.seagate.com

Csábító hangok

Kifejezetten a *Windows*-felhasználók elcsábítására állította össze a *Xandros* a főként otthoni, egyszerűbb irodai feladatok ellátására szánt *SurfSide Linuxot*. A terjesztés tartalmaz víruskeresőt és tűzfalvarázslót, lehet vele dokumentumokat szerkeszteni, *DVD*-t írni, fájlokat titkosítani és így tovább. Különlegessége, hogy nemcsak tartalmazza a *Skype* internettelefon-alkalmazást, de a dobozos változathoz mellékelnek egy *USB*-s *Plantronics* fejbeszélőt, valamint egy *Skypeout* kupont is, amivel a felhasználó 120 percnyi beszélgetést folytathat a világ bármely telefonszámára. A *Xandros SurfSide Linux* ára öt cent híján 100 dollár.

Nyíltan, szépen

Az *OASIS (Organization for the Advancement of Structured Information Standards, szervezet a strukturált információs szabványok fejlesztéséért)* bejelentette, hogy szabványként fogadta el az *Open Document Format for Office Applications*, röviden *OpenDocument* 1.0-s változatát – ezt az *XML* alapú formátumot fogja alkalmazni az *OpenOffice.org* irodai csomag 2.0-s, hamarosan megjelenő változata is. Az új formátum, bár az *OpenOffice* 1.x formátumokra épül, azokkal nem kompatibilis, ugyanakkor tökéletesen nyílt, bármilyen irodai programcsomag által használható, szövegek, táblázatok, grafikonok és grafikai anyagok tárolására egyaránt alkalmas. Érdemes megjegyezni, hogy a *Microsoft Office* következő változata is *XML* alapú formátumot fog használni, ám az ettől eltérő és részben zárt lesz, használatáért jogdíjat kell majd fizetni. Az új formátum mögött máris felsorakozott többek közt a *Sun*, az *IBM* és a *Red Hat*.

A *Siemens* bejelentette, hogy eladja – egyébként masszívan veszteséges – mobiltelefon-üzletágát a tajvani *BenQ*-nak, miközben 2,5 százalékos részesedést is szerez benne. Az üzlet révén a *Siemens* megszabadul a pénznyelő ágazattól, a viszonylag kis, mindössze három éve létező *BenQ* viszont a világ legnagyobb mobiltelefon-eladói közé nőhet fel. A *BenQ* a megegyezés szerint még 5 évig használhatja a telefonokon a Siemens nevet, illetve bizonyos sporttámogatói szerződéseket is átvesz.

Az inkább az olcsó termékek szegmensében terjeszkedő *BenQ* tempós fejlődést vár a mobiltelefonok területén, az első *BenQ-Siemens* készülékek már ez év végén megjelenhetnek az üzletekben.

Önállósodó Fedora

A *Red Hat* önálló, független alapítványi feladattá tette a *Fedora* terjesztés gondozását. Bár ez egyfajta távolodást jelent a cég részéről, szerencsére arról szó sincs, hogy a kalaposok levennék kezüket az ingyenes operációs rendszerről, a lépés sokkal inkább a kódok szerzői és használati jogainak rendezésére irányuló lépésnek tűnik. A *Red Hat* terjesztések a *Fedora* tervezet keretein belül továbbra is elérhetők lesznek, ahogy az alapítvány a cég pénzügyi és műszaki segítségére is számíthat. Ugyancsak átadásra kerül például az újonnan bemutatott címtárkiszolgáló és a hamarosan megjelenő tanúsítványkezelő rendszer is, a felhasználók mindkét programot *GPL* hatálya alatt érhetik majd el. Ide kapcsolódik a *Red Hat Software Patent Commons* kezdeményezése is, melynek mintájául a grafikákat, hangokat, szövegeket összefogó *Creative Commons* gyűjtemény szolgált, és melynek célja segítséget nyújtani a szoftverszabadalmak használati jogainak kezeléséhez, átadásához. A gyűjtemény életre hívása talán stratégiai húzás is, hiszen a *Red Hat* a jelenlegi – szerintük a nyílt forrású fejlesztéseket gátló – szabadalmi és szerzői jogi szabályok megváltoztatása mögé állt.

Oracle? Mi az nekünk!

Az *EnterpriseDB Corporation* kiadta *EnterpriseDB* 2005 adatbázis-kiszolgálójának béta változatát. A szabadon kipróbálható csomag nyílt forrású programokra épül, konkrétabban az adatbázis alapját a *PostgreSQL* adja, ennek továbbfejlesztésével a cég állítólag nagyvállalati szintű, az *Oracle* adatbázisokkal versenyképes, ám azoknál olcsóbb terméket tudott előállítani. Az *Oracle*-nek vetett kesztyű – bár a két cég viszonyát leginkább a bolha és az elefánt párharcához hasonlíthatnánk – nem merül ki az ígérgetésben, az *EDB2005* támogatja az *Oracle* alapokon fejlesztett alkalmazásokat, illetve az *Oracle*-féle *SQL*-szintaxist, adattípusokat, ravaszokat és natív tárolt eljárásokat, miközben általános körülmények között gyorsabbnak ígérkezik a hasonló nyílt forrású termékeknél. A különféle terjesztésekhez jelenleg szabadon letölthető csomagban az adatbázis-kezelő motor mellett egy grafikus felhasználói és rendszergazdai konzol, továbbá *JDBC*, *ODBC*, *.NET*, *ESQL/C++*, *PHP*, *Perl* és *Python* csatlakozók találhatók. A cég idén nyáron végleges változatban is, egyelőre ismeretlen áron megjelenő rendszerével az ingyenes, de kis tudású és a komoly tudású, de drága termékek közé szeretne beékelődni a piacon.

➲ www.enterprisedb.com

Medgyesi Zoltán
(mz@rettesoft.hu)
A Linuxvilág hírszerkesztője. Szabadidejét legszívesebben a barátnőjével tölti, szeret autózni és bográcsban főzni.

Mi újság a rendszermag fejlesztése körül?

A *iswraid* illesztőprogram jó úton halad afelé, hogy a 2.4-es fa részévé váljon, annak ellenére, hogy újabb szolgáltatásokkal bővít egy üzembiztos sorozatú rendszermagot. *Marcelo Tosatti* végül *Jeff Garzik* mellé állt e kérdésben, dacolva számos más fejlesztő ellenvéleményével. *Jeff* úgy érvelt, hogy az *iswraid* nélkül a 2.4-es használói nem tudják majd beüzemelni új hardvereszközeiket, míg az ellenzők (köztük *Arjan van de Ven*, *Bartlomiej Zolnierkiewicz* és *Christoph Hellwig*) rámutattak, hogy pontosan ugyanez történik minden más hardverrel is, aminek még nem oldódott meg a támogatása. A jelenlegi helyzet szerint *Jeff* feladata a kérdés rendezése, vagyis az *iswraid* szerepelni fog a 2.4 jövőbeli kiadásaiban.

Számos új illesztőprogram látott napvilágot. *Vojtech Pavlik* például készített egy a soros *Elo* érintőképernyők támogatására alkalmas illesztőprogramot, amely az összes soros *Elo*-t ismeri. A rendszermagnak ez a területe egyelőre még nem bontakozott ki, ugyanis, bár az érintőképernyők támogatására eddig is láthattunk példákat, azok inkább belső, vállalati tervezetek keretében valósultak meg. Új valós idejű óra illesztőprogram készült az *ST M41T00 I2C RTC* lapkához; *Mark A. Greer* szerzeménye gyakorlatilag készen áll arra, hogy a 2.6-os fa részévé váljon. *Mark* a *PPC* és *MIPS* rendszerekben található *Marvell* gazdahíd *I2C* vezérlőjéhez is készített egy illesztőprogramot.

Willy Tarreau – Marcelo Tosatti áldásával – új gyorsjavítási ágat indított a 2.4-es fán belül. A *-hf* ág ugyanazokat a javításokat fogja tartalmazni, mint a 2.4-es, ám rövidebb megjelenési időkkel készül. A *-hf* ág az új illesztőprogramokat és a meglévő illesztőprogramok frissítéseit nem fogja tartalmazni, kizárólag biztonsági és a kód letisztulását szolgáló javításokat foglal majd magába. Magától értetődő kérdés, hogy a *-hf* ág elindítása előtt maga *Marcelo* is megfontolhatta volna a kiadások felgyorsítását. A helyzet az, hogy a kialakult megoldás összhangban volt *Marcelo* azon szándékával, hogy a 2.4-es rendszermagot inkább az üzembiztosság irányába terelje, amivel viszont semmiféle kapkodás nem egyeztethető össze.

Christoph Lameter scrubd névvel készített egy lapnullázó démont, illetve összeállította a használatához szükséges rendszermagbeli környezetet is. Célja az, hogy a lehető legjobb teljesítményt lehessen kihozni a laphibakezelőből, segítségével a memórialapok nullázása még használatba vételük előtt megtörténik, nemcsak akkor, amikor egy folyamat kéri őket. Szép dolog még az ilyen aprónak tűnő fejlesztésekre is figyelmet fordítani. Nem egy új illesztőprogramról van szó, semmilyen *API* módosítását nem igényli, sőt, a külvilág számára gyakorlatilag észrevehetetlen, mégis hozzájárul ahhoz, hogy a *Linux* kifinomult, kiváló, mindannyiuk igényeinek megfelelő operációs rendszerré váljon. Pontosan ezek az optimalizálások adják a *Linux* színe-javát, és megérdemlik, hogy az új illesztőprogramok és a különleges fájlrendszerek mellett említsük őket.

Az *out-of-memory process killer (OOM Killer, „memórián kívüli folyamatok gyilkosa”)* továbbra is az egyik legkeményebb dió a *Linux* fejlesztésében. *Mauricio Lin* nemrég kiadott egy felhasználói térben futó változatot, amely állítása szerint ugyanolyan jól működik, mint a rendszermag térben futó. A kérdés azonban sajnos nem ilyen egyszerű. A felhasználói térben futó eszköz ugyanis legalább annyira esélyes a memórián kívüli állapotba kerülésre, mint az összes többi program. Ugyanakkor a rendszermag oldali *OOM Killert* sokkal nehezebb az egyes rendszerek egyedi jellemzőinek megfelelően hangolni. *Mauricio* egyfajta kompromisszumot kötött azzal, hogy az osztályozó algoritmust a felhasználói térbe helyezte, ahol könnyebben lehet módosítani a beállításait, magát a gyilkost azonban a rendszermagban hagyta, ahol védettebb az éppen általa vadászott memórián kívüliségtől. Bár az egész problémát rendkívül nehéz megfelelően kezelni, az *OOM*-kezelő eszközök ugyanis mindig bonyolultak, *Mauricio* megoldása a vezető fejlesztők között is támogatásra lelt, például *Marcelo Tosatti* részéről. *Mauricio* a témához kapcsolódó területeket is körbejárta, nemrég például olyan foltot készített, amelynek segítségével a felhasználók a */proc* könyvtárban figyelemmel követhetik a folyamatok fizikai memóriahasználatát. Mondani sem kell, ennek kapcsán is voltak viták, ám *Andrew Morton* támogatja a dolgot, és mások is felvetettek olyan alkalmazási lehetőségeket, amelyek a gyakorlatban is hasznossá teszik a szolgáltatást.

Jeff Garzik nemrég felhívást tett közzé, mely szerint több félbehagyott, hibás illesztőprogramot kell hamarosan eltávolítani a 2.6-os fából. Az *iphase* illesztőprogram például már évek óta gazdátlan, mára már a lefordítása is lehetetlenné vált. A *xircom_tulip_cb* illesztőprogramot senki sem tartja karban, ráadásul nem ismeri az összes 32 bites *Xircom* kártyát. A *xircom_cb* illesztőprogram ellenben igen, vagyis kiváló helyettesítőt jelentene. A *eepro100* illesztőprogram szintén magára maradt, szerepét az *e100* veszi át. Szerencsére azoknak a felhasználóknak sem kell izgatniuk magukat, akik az *e100*-at valamiért nem tudják használni, ugyanis a felmerült hibákat még az *eepro100* eltávolítása előtt el fogják hárítani.

Zack Brown

Linux Journal 2005. június, 134. szám

Új termékek

SUSE Linux Professional 9.3

A *Novell* kiadta a *SUSE Linux Professional 9.3*-at, mely teljes értékű operációs rendszert, több mint háromezer nyílt forrású csomagot, több száz nyílt forrású alkalmazást és termelékenységnövelő programokat foglal magába, illetve támogatja az otthoni hálózatokhoz szánt szolgáltatásokat is. Az újoncok és a régi *Linux*-felhasználók számára egyaránt jól használható *SUSE Pro 9.3* számos újdonságot tartalmaz. Az operációs rendszer a 2.6.11-es rendszermagra épül, mely köré *KDE 3.4*-et és *GNOME 2.10*-et, *Firefox 1.0*-t, *OpenOffice.org 2.0*-t, *F-Spot* fényképszervezőt, *The GIMP 2.2*-t, *Mono 1.1.4*-et, *KDevelop 3.2*-t, *Eclipse 3.0.1*-et és továbbfejlesztett *VoIP*-támogatást tartalmazó programcsomagot állítottak össze.
A *SUSE Pro 9.3* a *Wi-Fi* kapcsolatok és a *Bluetooth* készülékek magasabb szintű támogatásával, a zsebtitkárokkal és telefonokkal végzett szinkronizálás lehetővé tételével, az *iPod* készülékek támogatásával, beépített tűzfallal, levélszemétszűrővel és víruskeresővel, továbbá *Novell Evolution 2.0*-val és *Kontact 3.4*-el segíti a felhasználókat mobilitásuk megőrzésében. A 9.3-mas kiadás ezek mellett tartalmazza a *XEN* virtualizációs környezetet, a keresésekben könnyen használható motorokkal segíti a felhasználókat, illetve egyaránt támogatja az *AMD Athlon 64*-et és az *Intel Extended Memory 64 Technology*-t.
www.novell.com

OPTion

Az *OPTion* egy képzetes vékony ügyfél linuxos munkaállomásokhoz. *GNOME* és *KDE* alatt egyaránt használható, egyetlen alkalmazáson keresztül az összes ismertebb ingyenes vagy kereskedelmi terminálkiszolgáló környezet elérését lehetővé teszi. Minden ügyfélkapcsolat beállítá-

sa és felügyelete központilag történik, és minden feladat megjelenítése és futtatása egy központi keretrendszerben folyik.
A támogatott ügyfélkapcsolatok között szerepel az *XDMCP*, teljes képernyős és/vagy ablakos módban; a biztonságos és közvetlen *X*; a biztonságos *X* bejelentkezés, teljes képernyős és/vagy ablakos módban; az *RDP*, szintén teljes képernyős és/vagy ablakos módban; az *xRDP*, beépített *Ericom* modullal, mely lehetővé teszi az észrevétlen együttműködést a *WTS 2000/2003*-mal, illetve ingyenes *RemoteView* terminálkiszolgálóügynököt biztosít; az *ICA*, kiszolgáló- és alkalmazásböngészővel; az *Ericom PowerTerm Emulator* csomag; a *NoMachine NX Client*, mely az *NX Server 1.3* és 1.4 támogatására képes; illetve a natív Tarantella. A támogatott *Linux*-terjesztések körét a *MandrakeLinux*, a Fedora, a Novell/SUSE és a Xandros alkotja.
www.smartflextech.com

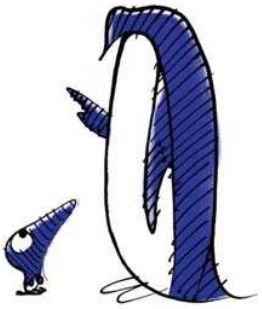
ConvertX SDK

A *Plextor Corporation* bejelentette egy szabadon hozzáférhető, a *ConvertX* videórögzítő eszközök-

höz készült linuxos szoftverfejlesztő készlet elérhetőségét.
A készlet segítségével az *USB* 2.0 felületre csatlakozó, hardveres *MPEG-1*, *MPEG-2*, *MPEG-4* és *Motion JPEG* tömörítésre képes *Plextor ConvertX* személyi videófelvevő készülékekhez lehet alkalmazásokat készíteni.
A linuxos fejlesztői készlet támogatja a *Video for Linux 2 (V4L2)* és az *Advanced Linux Sound Architecture (ALSA)* előírásait.
A ma már elavultnak számító *Open Sound System (OSS)* alapú alkalmazások támogatását az *ALSA* által biztosított *OSS* együttműködési réteg révén teszi lehetővé. Az új illesztőprogram, mely immár 2.6-os rendszermagot igényel, nyílt és zárt alkalmazásokban egyaránt felhasználható kódrészleteket tartalmaz, ezek alapján a fejlesztők remélhetőleg könnyebben el tudnak indulni munkájukkal.
www.plextor.com

ARCOS 4.0

A *Plus Three, LP* kiadta az *ARCOS 4.0*-t – az alkalmazás *Linuxra*, *Apache*-ra, *MySQL*-re és *Perl*-re épül, célközönségét az adománygyűjtő szervezetek adják. Az *ARCOS* alkalmas a támogatókkal kiépített kapcsolatok kezelésére, az elektronikus levelek és a kapcsolattartások követésére, az eseménykezelésre, a csoportos kapcsolattartásra, illetve internetes „tevékenységközpont" fenntartására. Új szolgáltatás a továbbfejlesztett valós idejű jelentéskészítés az adatbázisok alapján, a nagyvállalati szintű biztonsági mentés, illetve a nagyobb és gyorsabb felhasználó-adatbázis. Az *ARCOS* elektronikus hírlevél moduljával a felhasználók számtalan különböző, az adatbázisban tárolt szempont szerint állíthatják össze a e-mail címlistákat. A webes közzétételi modul testreszabható támogatói oldalak létrehozását és „mondd el a barátodnak" oldalak létrehozását is lehetővé teszi. Emellett az e-mail és a webes közzétételi modulokat összeépítették, így a felhasználók óránként akár kétmillió üzenet feldolgozását is elvégezhetik.
www.plusthree.com



A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tükröoldalait, a gyakori kérdéseket és az egyéb útmutatásokat a ➲www.linuxjournal.com honlapon olvashatjátok el. A rovatban közzétett válaszokat Linux-szakértők kis csapata készítette el. További kérdéseiteket szívesen fogadják (angol nyelven) a ➲www.linuxjournal.com/lj-issues/techsup.html címen, ahol csak egy kérdőívet kell kitöltenetek, de a bts@ssc.com címre levelet is írhattok. A levél tárgyában szerepeljen a „BTS” kulcsszó.

A hónap szakmai tanácsai

Nagyméretű meghajtók?

Red Hat 9.0, Fedora 1 és ***Debian 3.0r4*** rendszereket használok. Segítséget kértem az ***Inteltől*** a 160 GB-os merevlemezek használatával kapcsolatban, de ők azt válaszolták, hogy a lehetőségeket az operációs rendszer határolja be. Utána a ***Windows 2000***-re és a ***Windows XP***-re utaltak, ezért arra gondoltam, talán a ***BIOS*** is szerephez juthat. Mit gondoltok a témáról, és vajon hol találhatnék további anyagokat róla?

Georg Robertson
➲ *grobertson29@earthlink.net*

A gép ***BIOS***-a valóban korlátozza, korlátozhatja a merevlemezek méretét. A jó öreg ***DOS Int 13*** eljárása nagyjából 8 GB-ban szabta meg a meghajtók maximális kapacitását, míg a korszerű ***BIOS***-ok és merevlemezek képesek a 32 bites szektorcímzésre, amivel több mint 2 TB az elméleti határ – ez persze új kihívásokat jelent a szoftverek számára. Mindemellett az operációs rendszer lemez-illesztőprogramjai, rendszertöltője, fájlrendszere és egyéb szolgáltatásai – például szoftveres ***RAID*** – szintén befolyásolhatják a lemezmeghajtók vagy meghajtócsoportok kihasználható kapacitását.

Felipe Barousse Boué
➲ *fbarousse@piensa.com*

Én sokszor dolgozom furcsán formázott, általában windowsos meghajtókkal; szerencsére a rendszermagnak kézzel is meg lehet adni, hogy mit kezdjen velük. A témakörhöz készült egy kiváló útmutató, javaslom, hogy kezdésként ezt tanulmányozd át: ***www.tldp.org/HOWTO/Large-Disk-HOWTO.html***.

Chad Robinson
➲ *chad@lucubration.com*

Mobiltelefon használata USB kábelen keresztül?

Több ***GPRS*** mobilt is tudtam már használni, köztük ***Motorola V66***-ot és ***Timeportot***, de csak soros kábelen keresztül. A legújabb ***GPRS*** mobilok azonban már csak ***USB***-s adatkábellel rendelkeznek. Próbáltam az egyiket ***Linux*** alatt használni, de mindhiába, a számítógép nem találta meg a modemet. Tudtok valamit javasolni? Vajon hol találok megfelelő illesztőprogramokat?

➲ *kimaya@vsnl.com*

Ezek az eszközök szinte mindig sorosak, csak tartalmaznak egy az ***USB*** felületet biztosító ***USB***-soros átalakító lapkát. Az átalakítás kétféle módon történhet. Az egyik megoldásnál, ilyen például az ***FTDI*** lapkakészlet, egy virtuális soros kapu jön létre az ***USB*** felületen keresztül. Ezeket a termékeket már támogatja a ***Linux***, de ha mégse, akkor csupán idő kérdése, hogy a dolog rendeződjön.

A második csoportba a zárt fejlesztések tartoznak, ezeknél egyedi illesztőprogramok tartják a kapcsolatot a távoli lapkakészlettel. Itt már bajosabb a hordozhatóság, hiszen a gyártók jellemzően csak ***Windows*** alá készítik el az illesztőprogramokat, amelyek nélkül persze nem lehet adatkapcsolatot létesíteni az eszközzel. Szerencsére ezekből egyre kevesebb van, bár sajnos az ilyen megoldások olcsóbbak, mint a virtuális kapukat létrehozó lapkakészletek, ezért teljes eltűnésük sem várható. A legjobb az, ha a hírcsoportok, fórumok és egyéb források segítségével tájékozódsz, utánaolvasol, hogy másoknál mi működik és mi nem, és elkerülöd az ilyen termékeket.

Chad Robinson
➲ *chad@lucubration.com*

Számtalan ***GPRS*** telefon használható ***Linux*** alatt; a következő webhelyeken rengeteg hasznos információt találsz a témával kapcsolatban:

kotinetti.suomi.net/mcfrisk/linux_gprs.html, ***users.tkk.fi/~kehannin/bluetooth/bluetooth.html*** és ***markus.wernig.net/en/it/usb-serial-handy-ppp.phtml***.

Javaslom, hogy a telefon és a ***Linuxot*** futtató gép közötti kapcsolat létrehozását ***Bluetoothon*** keresztül is próbáld meg – természetesen ehhez megfelelő csatoló és ***Bluetooth***-képes telefon kell.

Felipe Barousse Boué
➲ *fbarousse@piensa.com*

A ***Tuxmobil.org*** webhelyen szerepel egy lista a működőképesnek bizonyult összeállításokról, illetve útmutatást is találsz az egyes telefonmodellek használatához.

Don Marti
➲ *dmarti@ssc.com*

Hibajelzés a MySQL ügyféltől

A ***Fedora Core 3*** grafikus ***MySQL*** ügyfélprogramját próbálom használni, de állandóan összeomlik, a következő üzenettel:

```
[anupam@localhost mysqlgui-1.7.5-1-linux-
➥static]$ ./mysqlgui
mysqlgui: dynamic-link.h:57:
➥elf_get_dynamic_info:
Assertion `! “bad dynamic tag”’ failed.
Aborted
```

Vajon mi a baj?

Anupam De
➲ *anupam@sail-steel.com*

A *mysqlguit* binárisan töltötted le, vagy szöveges vagy *ASCII* módban? Ha ugyanis szöveges vagy *ASCII* módban töltötted le, akkor valószínűleg megsérült a fájl. A félig statikus bináris fájl helyett próbáld meg a statikusan fordított változatot letölteni. Így a függőségek túlnyomó részével nem kell foglalkoznod, ugyanis a – valamivel nagyobb – futtatható fájl minden szükség elemet tartalmaz.
Felipe Barousse Boué
➲ *fbarousse@piensa.com*

Soros kapuk IRQ-inak megadása

Win4Lin-t futtatok *SUSE 9.2* alatt, és keményen küzdök a 2-es *COM* kapu *IRQ*-jának megváltoztatásával. A *Windowsra* egy energiafelügyeleti program miatt van szükségem, ami külső hívásokkal ellenőrzi több épületbéli rendszer állapotát is. A *Linux* 10-es *IRQ*-t állított be, de 4-nek kellene lennie. Meg tudjátok mondani, hogyan változtathatom meg az *IRQ*-t?
John Langston
➲ *jdl.28@cox.net*

Az *IRQ*-t a *BIOS* beállításai között kell megváltoztatnod. Ha nem sikerül, akkor használd a linuxos *setserial* segédprogramot.
Greg Kroah-Hartman
➲ *greg@kroah.com*

A segédprogram kapcsolóit a `man setserial` paranccsal tekintheted át. Ne feledd, ha a fizikai soros kapuk fix *IRQ*-val és/vagy memóriacímmel rendelkeznek, akkor a *setserial* használatakor és/vagy más eszközök beállításainak módosításakor ütközéseket okozhatsz.
Felipe Barousse Boué
➲ *fbarousse@piensa.com*

Működésképtelen GigaDrive

Nemrég vettem egy *Linksys GigaDrive*-ot az *eBayen*. A készülék látszólag működik, legalábbis bekapcsol, de az alkalmazások egyikét sem tudom elérni vagy futtatni. Arra gondoltam, talán megformázták vagy kicserélték a meghajtót, ezért újra kellene telepíteni a linuxos rendszert és az alkalmazásokat. Tudtok valamilyen tanácsot adni a művelet elvégzéséhez, azon túl, hogy küldjem vissza a *Linksysnek*? Van ugyan *A+* vizsgám, de linuxos tapasztalatom nem nagyon. Talán, ha sikerülne szereznem egy telepítő *CD*-lemezt, akkor életet tudnék lehelni a gépbe, nem? Egyelőre a *CD* beszerzését sem tudom, hol kezdjem. Van valami ötletetek vagy javaslatotok?
Randy Warner, ➲ *warn4421@bellsouth.net*

A *Linksys* webhelyén van egy oldal, ami ismerteti a *GigaDrive* belső programjának feltöltését:
www.linksys.com/support/support.asp?spid=17

Ha ezzel nem jutsz előbbre, esetleg próbáld megszerezni egy jól működő, azonos méretű lemezt tartalmazó GigaDrive merevlemezét, majd szereld be mindkét meghajtót egy linuxos gépbe. A működő meghajtó legyen a második *IDE* csatoló mester meghajtója, a nem működő pedig a szolga; majd add ki a következő parancsot:

```
dd if=/dev/hdc  of=/dev/hdd
```

Don Marti
➲ *dmarti@ssc.com*

Kettős rendszerindítású gép mentése

Jelenleg *Microsoft Windows XP Professional* operációs rendszert használok, de ha sikerült megismerkednem a használatával és felügyeletével, szeretnék áttérni *Linuxra*. Jelenleg a *System Works 2004*-ben található *Norton Ghostot* használom mentésre.

Feltelepítettem a *Fedora Core 1*-et, ugyanis ez volt az általam megvásárolt könyvhöz mellékelve. A telepítés gond nélkül lezajlott, amit kaptam, az tetszett. Csakhogy, amikor a *Ghostot* akarom használni, és ezért visszaváltok *Windowsra*, a *Ghost* a következő hibaüzenetet adja:

Biztonsági mentési hiba. Nincs hely az MBR-ben.

Gondoltam, hagyom a *Nortont*, majd *Linux* alól elvégzem a mentéseket – de vajon melyik programot használjam? Ti mit ajánlotok?
Lev Ranara
➲ *pinoy_techie@yahoo.com*

Linux alatt általában egyszerű a mentések elvégzése. A *Windowszal* ellentétben nincsenek különleges rendszeradatok (mint például a rendszerleíró adatbázis), amelyeket hagyományos eszközökkel nem lehet menteni. Egy teljes mentés sok esetben egy egyszerű fájlmásolással letudható, hacsak nem futtatsz adatbázis-kiszolgálót. Ha mégis, akkor a mentés idejére azt le kell állítanod.

Összetettebb megoldásokból is van bőven, ezekkel felügyelt, katalógusszerű mentéseket lehet végezni, illetve az egyes fájlok önálló visszaállítására is biztosítanak lehetőséget. Egy részük ingyenes (mint például az *Amanda* és a *Bacula*), másokat hagyományosan windowsos mentési programokat készítő cégek kínálnak (*VERITAS*, *CA* stb.), megint másokat pedig kifejezetten linuxos megoldásokat fejlesztő cégektől (mint például a *BRU*) vásárolhatsz meg. Ha most *Ghostot* használsz, akkor vélhetően nem

fájl alapú mentéseket készítesz. A legegyszerűbb megoldás talán egy tömörített tar archívum létrehozása lenne. A teljes rendszer visszaállításához ekkor elég lenne megadni a lemez felosztását, megformázni a meghajtót, kibontani az archívumot majd telepíteni a rendszertöltőt.

Ha helyes a gondolatmenetem, akkor talán kezdd a tar megismerésével; majd kiderül, hogy megfelel-e az igényeidnek. A…

```
tar -jlcvf /tmp/mentes.tgz /bin /boot
➥/dev /etc \
```

parancs például a legtöbb esetben megfelelő. Kiadása után a létrejövő */tmp/mentes.tgz* fájlt egyszerűen másold *CD*-lemezre, szalagra vagy egy fájlkiszolgálóra. Ha gondolod, a tar állományt közvetlenül a szalagon is létrehozhatod.
Chad Robinson, ➲ *chad@lucubration.com*

Tapasztalataim szerint a linuxos világban a legjobb megoldás mentések készítésére a jó öreg tar parancs használata. Szóba jöhetnek még más tömörítőprogramok is, mint a *zip* és a *bzip*; az egyedi igényeimre pedig írtam néhány parancsfájlt. Megbízható, hordozható, egyszerű és ingyenes – mondhatni, a szabad és anyagiak terén tudatos felhasználó választása. A *Linux* alatt végzett biztonsági mentések témakörében mindent megtalálsz a *www.linux-backup.net* webhelyen. A *Unix Backup and Recovery* című könyv szintén ezzel a témával foglalkozik, a *www.linuxjournal.com/article/3839* címen rövid áttekintést is találsz róla.

Mivel az *FC1* mára elavulttá vált, próbálj *FC3*-at telepíteni. Az *FCx* terjesztések számtalan tetszetős megoldást tartalmaznak, ezek közül például a kattints és húzd módban végezhető *CD*-írás mentési célokra is alkalmazható.
Felipe Barousse Boué
➲ *fbarousse@piensa.com*

Az ügyfél csatlakozik, az átvitel mégis sikertelen

Egy *TFTP*-kiszolgálót próbálok üzembe helyezni, de nem járok szerencsével. A helyzet röviden a következő. *Fedora Core 3*-at futtatok egy *PIII*-as gépen. Telepítettem az *rpmfind.net* oldalon talált legújabb tftpd-t, majd megadtam a – gondolom – megfelelő beállításokat az *xinetd/in.tftpd* fájlban. Egy másik linuxos gép *tftp*-ügyfelével ugyan sikerül csatlakoznom a kiszolgálóhoz, ám az olvasási kérés már megválaszolatlan marad. Az ügyfél többször is próbálkozik, majd időtúllépést jelez. A */var/log/xinetd* fájlban minden az ügyfél által elküldött olvasási kérés hatására a következő bejegyzések jelennek meg:

```
05/3/16@14:11:14: FAIL: tftp address
➥from=153.90.196.30
05/3/16@14:11:14: START: tftp pid=20184
➥from=153.90.196.30
05/3/16@14:11:14: EXIT: tftp pid=20184
➥duration=0(sec)
```

A kiszolgálón a következőket végeztem el. Létrehoztam egy *tftp* felhasználót, a kezdőkönyvtára a */tftpboot* lett, majd lefuttattam a */sbin/nologint*. A */etc/hosts.allow* fájlhoz hozzáadtam az `in.tftpd:ALL` bejegyzést. Létrehoztam a */tftpboot* könyvtárat, és megadtam hozzá a megfelelő engedélyeket és tulajdonosi beállításokat. Létrehoztam a */etc/xinetd.d/tftp* fájlt a következő tartalommal:

```
service tftp
{
  disable = no
  socket_type     = dgram
  protocol        = udp
  wait            = yes
  user            = root
  server          = /usr/sbin/in.tftpd
  server_args     = -s /tftpboot -u tftp
  per_source      = 11
  cps             = 100 2
  flags           = IPv4
  #only_from      = 153.90.196.30
}
```

Az `only_from` sort megjegyzésbe tenni és onnan kivenni is próbáltam. Ellenőriztem, hogy a tűzfal engedélyezi-e a 69-es *UDP*- és *TCP*-kapu használatát. A */etc/xinetd.conf* tartalma helyes, a `chkconfig`-gel ellenőriztem, a *tfptd* valóban fut. A `netstat` szerint a 69-es kapu elérhető. Az *in.tftpd*-t önálló kiszolgálóként is megpróbáltam futtatni (`server_args = -l`).

A dologgal már napok óta szenvedek, de nem sikerül egyről a kettőre jutnom. A *Linux* világában még viszonylag új vagyok, megkértem néhány tapasztaltabb ismerősömet, hogy nézzék meg a rendszert, de hiába. Az interneten is órákat kutattam már, de nem sikerült rájönnöm, mi a megoldás. Remélem, ti végre meg tudjátok mondani, merre induljak el.
Todd Trotter, ishamt@esus.cs.montana.edu

Úgy tűnik, majdnem mindent jól csináltál – de csak majdnem. Először is, a */etc/xinetd.d/tftp* fájlban a felhasználót írd át `nobody`-ra, egyébként az *in.tftpd* démon rootként fog futni, ami nem biztonságos. Ezután ellenőrizd, hogy a

```
tftp            69/tcp
tftp            69/udp
```

sorok nincsenek-e megjegyzésbe téve a */etc/services* fájlban. Javaslom, hogy a */etc/hosts.deny* fájlt is nézd át, nincsenek-e letiltva az *in.tftpd* démonnak intézett kérések, esetleg az összes szolgáltatás vagy adott *IP*-cím (ügyfélgép) kérései.

A tesztelés idejére – de kizárólag ekkor – ki is ürítheted a fájlt, indítsd újra az *xinetd*-t (`service xinetd reload`), majd próbálkozz újra. Szintén a tesztelés időtartamára megpróbálkozhatsz a tűzfalad (`service iptables stop`) leállításával. A próbálgatás során először a helyi működést kell elérni, vagyis a `tftp localhost` parancs használhatóságát, a távoli elérés csak ez után következhet. Remélem, segítettem.

Felipe Barousse Boué
➲ *fbarousse@piensa.com*

A szemétgyűjtés volna a válasz?

Az újságban olvastam a szemétgyűjtésről. Van egy gondom, hadd írjam le röviden. Kezdetben a tervezet 192 MB memóriát foglal. A futása folyamatos. 12 óra után a memóriahasználat 335 MB-ra nő. Mi lenne a megoldás? A szemét miatt van ez? A *BDW* szemétgyűjtővel megoldódna a gondom? A tervezet char mutatókat tartalmaz, `malloc` hívásokat viszont nem.

A *BDW* szemétgyűjtés csak akkor működik, ha `malloc`, `calloc` és `realloc` hívásokat alkalmazok? Létezik olyan program, amit a saját tervezetemmel párhuzamosan futtatok, és felszabadítja a feleslegesen lefoglalt memóriát?

Mythily J.
➲ *mattuvar@yahoo.co.in*

Utolsó kérdésedre a válasz: nem. Hacsak nem valami szövevényes, túlbonyolított rendszerről van szó, kizárólag a saját programod tudja felszabadítani a saját maga által lefoglalt memóriát.

A többi kérdésed valóban jó, csak azt tudom mondani, hogy amint kipróbálod az ötleteidet, azonnal megkapod kérdéseidre a választ. Lehet, hogy bár a `malloc` függvénycsaládot nem használod, mégis indítasz olyan könyvtári hívásokat, amelyek memóriát foglalnak le, majd elhagyod azokat, amelyek felszabadítanák ezt a területet.

A jó hír az, hogy a programodból készíthetsz olyan változatot is, amely a `malloc`-ra „ráakaszkodva” minden memóriakezelő műveletnél szemétgyűjtést használ, ide értve a könyvtári kódok által végzett memóriafoglalásokat is. Példaként lásd a következő cikkben szereplő kódrészletet:

www.linuxjournal.com/article/6679

Don Marti
➲ *dmarti@ssc.com*

A futási szint módosítása

A 2005. májusi szám szakmai tanácsaiban, a „Régi Red Hat” című részben *Timothy Hamlin* a */etc/inittab* bejegyzésének módosítását javasolja, a következőről

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

erre:

```
x:3:respawn:/etc/X11/prefdm -nodaemon
```

A módosítással elvileg kikapcsolható az X alapú, grafikus bejelentkezés. Azt hiszem, itt becsúszott egy hiba. Az általa javasolt módosítás után az X 3-as futási szinttel indul el. Helyette a következőt kellene megváltoztatni:

```
id:5:initdefault:
```

erre:

```
id:3:initdefault:
```

Ezzel valóban az alapértelmezett futási szintet változtatjuk meg.

Az *„A fájlleírók és a blokkméretek módosítása”* című részben *Don Marti* pedig arra utal, hogy a *Red Hat 9* támogatása megszűnt, ami a régebbi, 486-os gépek esetében gondot jelenthet. Nos, ennél sokkal nagyobb bajt jelent a *Red Hat* által igényelt memória mérete. Nem vagyok biztos abban, hogy 32 MB RAM-mal hajlandó települni; legalább-is 16 MB-tal biztosan nem telepíthető, ennyi volt ugyanis a jó öreg 486-os, hordozható gépemben.

Roland Roberts
➲ *roland@astrofoto.org*

Az *inittab* mindkét módosítása megfelel a célnak. A másodiknak megvan az az előnye, hogy megőrzi a *Red Hat*-féle hagyományt, mely szerint az 5-ös futási szint a grafikus bejelentkezéshez tartozik. A *Fedora* kibocsátási tájékoztatója szerint (*fedora.redhat.com/docs/release-notes/fc3/x86*) a minimális, szöveges telepítéshez legalább *Pentium* processzor és 64 MB memória szükséges. (Alternatív megoldást az utolsó levélnél találsz.)

Don Marti
➲ *dmarti@ssc.com*

Na és a Fedora Legacy?

A 2005. májusi szakmai tanácsok között ***Don Marti*** azt írja: ***„Sem a Red Hat 9-hez, sem a Red Hat 6.2-höz nincs már támogatás, vagyis nem adnak ki hozzájuk biztonsági frissítéseket."*** Bár a ***Red Hat*** valóban felhagyott a ***Red Hat 9*** támogatásával, a közösségi ***Fedora-Legacy*** tervezet (***www.fedoralegacy.org***) továbbra is készít biztonsági frissítéseket a ***Red Hat 9***-hez, ahogy a ***Red Hat 7.3***-hoz, valamint a ***Fedora Core 1***-hez és (hamarosan) a 2-höz is. ***Marti*** úr elég sokat ártott a tervezetnek azzal, hogy egyszerűen figyelmen kívül hagyta az annak keretein belül történő erőfeszítéseket.

John Dalbec
➲ *jdalbec@cboss.com*

Amikor nyomdába adtuk a lapot, a ***Fedora Legacy*** még nem kezdte meg a biztonsági frissítések tényleges közzétételét.

Don Marti
➲ *dmarti@ssc.com*

A ***Linuxvilág*** 2005. májusi számának szakmai tanácsai között szerepel néhány helytelen és hiányos állítás abban a válaszban, amelyet a 486-os gépeken ***Red Hat 9***-et futtatni kívánó felhasználó kapott. ***Don Marti*** azt írta, hogy ***„[A Red Hat-] utód, a Fedora futtatásához legalább Pentium processzor kell... Tulajdonképpen mindegy, hogy mit teszel fel, ezek a gépek egy korszerű munkakörnyezet futtatásához túlságosan lassúak."*** A ***RULE*** tervezet (***www.rule-project.org***) ezen próbál segíteni. Egy évvel ezelőtt már ***Red Hat 9***-et futtattam ***Pentium I***-es hordozható gépen, 32 MB memóriával. A tervezetnek köszönhetően a ***KOffice*** segítségével bemutatókat készítettem, ***Firefox*** alól pedig gond nélkül intéztem banki ügyeimet:
www.rule-project.org/article.php3?id_article=55

Alig egy hónapja bemutattuk telepítőnk ***Fedora Core 3***-hoz készült változatát is: ***www.rule-project.org/breve.php3?id_breve=19***

Kétségtelen, hogy egy tarkabarka ***KDE***, ***GNOME*** vagy ***OpenOffice.org*** alapú telepítés bármilyen gépen lehet lassú, még a jóval újabbakon is. Ugyanez igaz a videószerkesztésre, a 3D-s játékok futtatására is, amelyekhez a mindenkori legjobb gép szükséges. Ha viszont a korszerű munkakörnyezeten az otthoni vagy kis irodai szolgáltatások használatát értjük – ***IMAP***, digitális aláírások, ***HTML4/CSS***-támogatás, ***CUPS***, azonnali üzenetküldés, ***Bayes***-döntésekre alapuló levélszemétszűrés, csicsa nélkül –, akkor teljesen szükségtelen a pénzszórás. Az olyan tervezetek, mint a ***RULE***, illetve a például a mini-***KDE*** létrehozására irányuló munkák sokkal gazdaságosabb lehetőségeket biztosítanak. Nem igaz tehát, hogy a korszerű, a fősodorba tartozó terjesztéseket nem lehet régebbi gépeken futtatni, csak egy odafigyelésre, a probléma gondosabb kezelésére van szükség.

Marco Fioretti
➲ *mfioretti@mclink.it*

Linux Journal 2005. június, 134. szám

Ők mondták

A legnagyobb baj az, hogy újra kell írni a költségvetést, és ki kell találni, hogy mit kezdjünk azzal a pénzzel, ami eddig a Microsoftnak ment.
Boyce Williams, beszélgetés Doc Searls IT Garage-ében (garage.docsearls.com/node/550)

Ne úgy gondolkozz, mint egy költségközpont, úgyis csak megvonják a keretedet. Úgy gondolkozz, mint egy vállalkozó.
Névtelen, szintén Doc Searls IT Garage-ének egyik beszélgetéséből (garage.docsearls.com/node/550)

A környezetükben műszaki zseninek tekintett személyek és a professzionális szolgáltatásokat igénylő amatőrök között egyre kisebb a távolság.
Rael Dornfest

Buheráld a gépedet! Szerintem jó dolog.
Peggy Rogers, „Ms. Computer", The Miami Herald

Úgy vélem, minden programozónak ki kell vívnia az elismerést, függetlenül attól, hogy melyik cég adja a fizetését. Nézzük csak meg a szórakoztatóipart. Az, hogy ki hol jelenik meg a stáblistában, fontos, nagyon fontos kérdés. Közvetlenül befolyásolja a munkához való viszonyt, és kiváló képet ad arról, hogy ki mekkora részt vállalt a munkából. Szerintem ez az egyik legszebb dolog a nyílt forrás világában.
Danese Cooper, danesecooper.blogs.com/divablog/2005/03/about_attributi.html

A Linux rendszermag API-ja ismét olyan rejtélyes módon változott, hogy a rendszermagfán kívüli illesztőprogramok fejlesztőit lassan a bolondok házába lehet zárni. Ez van, amíg az illesztőprogramjuk be nem kerül a rendszermagfába.
Greg Kroah-Hartman, ww.kroah.com/log/2005/02/15

Linux Journal 2005. június, 134. szám

DRM pro és kontra

A mostanában oly sokat emlegetett DRM a digitális jogkezelő rendszerek (Digital Rights Management) összefoglaló megnevezése. A fejlesztés eredeti célja a jogvédett tartalmak illegális terjesztésének és másolásának megakadályozása volt...

Mára azonban kiderült, hogy a *DRM* sajnos más területeken is hatékony eszköz. Segítségével a nagyvállalatok megszabhatják, hogy az általunk amúgy hivatalosan megvásárolt terméket hol, hogyan és hányszor használhatjuk fel.

A *DRM* rendszerek fejlesztése eredetileg azért indult el, hogy a korábbinál egyszerűbb és biztonságosabb módon lehessen megakadályozni a kalózmásolatok készítését, illetve hogy a felhasználóknak ne kelljen „megmerülni" a végfelhasználói szerződések megértéséhez szükséges jogi ismeretekben.

Bár ez a cél önmagában kifejezetten jó, a *DRM* mostanra meglehetősen sok vihart kavart, mert sokan a „Digitális Nagy Testvért" látják benne. Az ügyben pillanatnyilag a legmeglepőbb talán az, hogy szinte minden résztvevőnek más a véleménye, de abban a legtöbben mégis egyetértenek, hogy jelen formájában a digitális jogkezelő rendszerek tervezete nem jó. Még olyanok is akadnak, akik szerint a „kalózkodást" egyáltalán nem kellene megtiltani vagy szankcionálni, a digitális termékeket pedig semmilyen módon nem kellene védeni.

A digitális jogkezelő rendszerek hátrányai

A *DRM* csak abban az esetben lenne működőképes, ha a jelenlegi tervezettel szemben egységes rendszer lenne. A jelenlegi elképzelések azonban olyan hibákat tartalmaznak, amelyek hátrányos helyzetbe hozhatják mind a digitális termékek alkotóit mind azok felhasználóit.

Talán meglepő ez a kijelentés, de egyes kutatások szerint a vásárló általában hajlandó kifizetni azt az összeget, amit a termék előállítója azért kér. A probléma az, hogy a „pénztártól való távozás után" a vásárló rádöbbenhet: nem olyan árut kapott, amit korlátozások nélkül felhasználhat.

Lássuk, mi minden érheti a – hangsúlyozzuk – jóhiszemű vásárlót!

Kompatibilitási problémák...

Mivel a különböző cégek pillanatnyilag mind más módon próbálják megvédeni saját termékeiket, csak a saját eszközükkel engedhetik megnyitni/lejátszani saját, egyedi adatformátumukat. Az ehhez szükséges eszközt azonban, ami bizonyos esetekben csak szoftver, általában szintén pénzért kínálják.

Ha tehát a gyanútlan felhasználó megvásárol egy filmet tartalmazó *DVD*-t, vagy egy elektronikus könyvet, nem biztos, hogy használni is tudja majd azt, mert megeshet, hogy a lejátszójába nem építették bele a kérdéses adatformátum dekódolásához szükséges modult. Ilyenkor – ha nem akar összeütközésbe kerülni a törvénnyel – nem tehet mást, új lejátszót kell vásárolnia.

Ez nyilvánvalóan a felhasználó szabadságának korlátozása, hiszen vannak olyan termékek, amiket kizárólag egy bizonyos formátumban lehet megvásárolni, tehát ha a felhasználó meg szeretné venni, kényszerítve van a digitális terméket lejátszó eszköz megvásárlására. Ez a *DRM* legtöbbször említett hátulütője.

Hasonló a helyzet a digitális könyvekkel. Ha valaki megvásárol egy hagyományos könyvet, akkor joga van azt elolvasni bárhol és bármikor. Ezzel szemben a *DRM* védelme alatt álló digitális könyv csak egy bizonyos programmal olvasható, ami egyes platformokon esetleg nem hozzáférhető. Ha tehát a vásárló elmegy egy idegen helyre, ott nem biztos, hogy tudja olvasni a szöveget.

És ez még csak az első probléma.

Ha valaki megvesz egy papír alapú könyvet, miután elolvasta, továbbadhatja vagy el is ajándékozhatja azt. Ugyanennek a műnek a digitális változata viszont ezek ellen a galádságok ellen is hatékonyan védve van, hiszen a következő tulajdonos minden valószínűség szerint képtelen lesz elolvasni.

Ez egyrészt meglehetősen egyoldalúan befolyásolja a felhasználót abból a szempontból, hogy milyen formátumú könyvet vegyen, másrészt pedig úgy tűnik, a digitális forradalom beköszöntével nyugdíjba vonulhat az összes antikvárius...

A *DVD*-k esetében a régiókód is a digitális jogkezelő rendszer része. Működésének lényege, hogy a lemezről a lejátszó egyértelműen meg tudja állapítani, milyen országból származik. Így például a *Magyarországon* megvásárolt *DVD* itthon tökéletesen működik, de mondjuk *New York*-ban már nem lehet lejátszani.

A dolog pikantériája, hogy jelenleg egyszerűen nincs olyan szerzői jogi törvény, amely a szerzőnek illetve forgalmazónak lehetővé tenné, hogy a vásárlót ilyen módon és mértékben korlátozza. A digitális jogkezelő rendszerek működése tehát pillanatnyilag egyszerűen jogsértő.

Biztonsági másolatok...

Az is előfordulhat, hogy egy legális termékről (például egy *DVD*-ről) a vásárló biztonsági másolatot szeretne készíteni. Ilyenkor az egyik forgatókönyv szerint már maga a *DVD*-író program nem engedi lemásolni a lemezt, hanem kiírja, hogy annak tartalmát szerzői jog védi, illegális másolat készítése pedig tilos...

A másik lehetőség, hogy a másolás ugyan sikerül, de a lejátszóban már nem működik a másolat.

Hasonló esettel kerülhetünk szembe, ha egy mobiltelefonra töltünk le alkalmazásokat. Ha a felhasználó letöltött egy programot, általában lementi azt a számítógépére is, gondolván arra az esetre, ha netán valami baleset érné a „kézi készüléket". Emberünk, mivel egyszer már kifizette az alkalmazás árát, joggal gondolhatja úgy, hogy minden további nélkül visszatöltheti azt a telefonjára. Ezzel szemben ha az alkalmazást *DRM* védi, a számítógépről való visszatöltés után hibaüzenetet fog kiírni, mondván a program egy illegális másolatát próbáltuk meg futtatni. Ezután a felhasználónak nincs más választása, mint hogy újból kifizesse a termék árát, hogy az ismét működőképes legyen.

A képzettebb – és főleg öntudatosabb – fogyasztók persze ilyenkor a megfelelő segédeszközökhöz nyúlnak majd. Nem, nem a házi jogtanácsosukról van szó...

Akárcsak eddig a szoftverkalózok tették, megpróbálják majd kikerülni a másolásvédelmi rendszert. Léteznek majd olyan programok, amiket csak le kell futtatni, és a digitális jogvédelem már el is tűnt a termékből. Ez természetesen ugyanúgy illegális, mint a szoftverek másolása. A különbség csak annyi, hogy most a jogos tulajdonos lopta el az anyagot. De pontosan kitől is?...

Ami a biztonságtechnika alapelveit illeti, a DRM megalkotói úgy tánik elkövettek egy szarvashibát: átadják a vásárlónak a kódolt terméket, a dekódoló eszközt, és a kulcsot is. Ez a felállás tulajdonképpen a teljes rendszer értelmét kérdőjelezi meg, hiszen ezek után hatékony védelemről szó sem lehet. Ha egy felhasználót eleve tisztességes szándék vezet, attól fölösleges a kérdéses anyagot megvédeni. Aki pedig jogtalanul akar egy tartalomhoz hozzáférni, az némi ügyeskedés árán a DRM ellenére is megteheti, hiszen várhatóan pillanatokon belül meg fognak születni a DRM kikerülését lehetővé tevő programok.

Minderre a digitális jogkezelő rendszereket alkalmazó cégek azt mondják, hogy a DRM nem a szervezett kalózcsoportok, nem az ügyes egyetemisták sőt még csak nem is azok ellen készült, akik képesek kreatív módon használni a Google-t vagy a Kazaa-t, hanem az ***„átlagos felhasználók"*** ellen. A DRM tehát az ő olvasatukban nem egyéb, mint a villanypásztor digitális megfelelője.

A gondolatmenetben mindössze annyi a hiba, hogy a gyártók sokak szerint messze alulbecsülik az „átlagos felhasználó" képességeit. Az említett programoknak ugyanis csak a megírása igényel különleges felkészültséget, a használata nem. Tény, hogy ismerni kell hozzájuk az egér és a billentyűzet kezelését, de aki erre sem képes, attól nem a DRM fogja megvédeni a digitális tartalmakat, hanem a saját butasága.

Valami Amerika – avagy az EUCD

Az ***EUCD*** az ***Amerikai Egyesült Államokban*** érvényes ***DMCA (Digital Millennium Copyright Act)*** európai változata. Amerikában a ***DMCA*** elsősorban azzal vívta ki a társadalom ellenszenvét, hogy a törvény megtiltotta a kutatók számára a másolásvédelmi technológiák tanulmányozását, valamint azt, hogy nyilvános ságra hozzanak olyan információkat, amelyek ezeknek a technológiáknak a hibáival, biztonsági réseivel kapcsolatosak.

Az ***EUCD*** arra kötelezi az ***Európai Unió*** országait, hogy olyan törvényeket léptessenek életbe, melyek megfelelő jogi védelmet biztosítanak a különböző titkosításoknak, másolásvédelmeknek és egyéb, a szerzői jogokat védő műszaki megoldásoknak. Az ***EUCD*** azt is kimondja, hogy az Európai Unió országaiban minden lehetséges eszközzel meg kell akadályozni az olyan elektronikai eszközök forgalomba kerülését, melyeket készítőik nyilvánvalóan egyes digitális védelmi rendszerek kiiktatására terveztek.

2004. július 29-én létrejött egy digitális jogkezelő rendszerekkel foglalkozó munkacsoport (DRM munkacsoport), amely a digitális jogkezelő rendszerek hatását és szükségességét vizsgálja jogi és technikai szempontból. A munkacsoport 2005. július 1-ére készíti el összefoglaló jelentését a kormány számára.

Lehetséges megoldások a DRM és az EUCD „kijavítására"

Oktatási célokra valamilyen mértékben mindenképpen hozzáférhetővé kellene tenni azokat a műveket, amelyek a tanuláshoz szükségesek, de egyébként a szerzői jogvédelmet élveznek. Ennek egyrészt történelmi jelentősége van, hiszen a felnövekvő nemzedék nem lehet a jelen jogi csatározásainak áldozata, másrészt hazánkban z az iskolák anyagi helyzete is indokolja.

A kompatibilitási problémák elkerülése érdekében a fejlesztők számára elérhetővé kellene tenni a különböző formátumok és digitális jogkezelő rendszerek működésének leírását. Ez a *DRM* rendszerek biztonsága érdekében is kívánatos volna, hiszen csak a technológia ismerete ad lehetőséget a fejlesztőknek arra, hogy a benne talált hibákat kijavítsák.

A vonatkozó törvényekben engedélyezni kell saját célra a biztonsági másolatok készítését, műszaki oldalról pedig lehetővé kell tenni azok felhasználását.

Az utóbbi időben egyre több az olyan dokumentum, amely eleve csak digitális formában jelenik meg. Ha mindent ilyen anyagot *DRM* védene, a digitális művek terjedése nagy mértékben lelassulna, ez pedig lassan az írók és fejlesztők hátrányára is válna. Azt tulajdonképpen ma sem vitatja senki, hogy a szerzőknek is az a jó, ha művük ismert és gyorsan terjed.

Összességében elmondható, hogy a *DRM* jelenlegi tervezetével mindenki elégedetlen, legyen akár szerző, kereskedő, vagy egyszerű vásárló. A legtöbben arra várnak, hogy megjelenjen egy egységes és fejlett jogkezelési rendszert, és erre talán van is némi esély. A jelenlegi *DRM* azonban semmiképpen sem ilyen.

Guba Norbert (nguba@monornet.hu)
Tanuló, körülbelül 3 éve használ Linuxot. Szabadidejét leginkább programozással tölti, hobbija az operációs rendszerek gyűjtése.

Gazdálkodj okosan!

Mindig értetlenül állok olyan üzleti célú szoftverek előtt, melyek csak egyetlen operációs rendszer alatt működnek. A fejlesztőcégek bele sem gondolnak abba, hogy létezhetnek más megoldásokat használó potenciális ügyfelek is?

Pontosan úgy, ahogyan egy operációs rendszernek az összes lehetséges hardver felett el kellene futnia, úgy kellene a könyvelő, útnyilvántartó, ügyfélkapcsolati szoftvereknek is működnie lehetőleg az összes operációs rendszer felett. Lehetőség éppen lenne rá...
Úgy látszik, ahogy egyfelől az innovációnak nincsenek korlátai, úgy bizonyos esetben a korlátoltságnak sem. Nézzünk meg tehát egy gyöngyszemet a sötét tenger fenekén, melynek neve ***SQL-Ledger*** és a két gyöngyhalász, aki most felszínre hozza nekünk ezt a gyöngyöt: ***Kabai József*** és ***Sásdi András***.

Először is azt szeretném, ha egy pár szóban összefoglalnád a magyar ügyviteli szoftver piac helyzetét jelenleg, mik azok amik jók, mik azok amik hiányosságok?
K.J.: Az ügyviteli szoftverek hagyományosan ***Windows*** operációs rendszeren futnak. Ez az ügyviteli szoftverek túlnyomó többségét, vagyis körülbelül 90-95%-át jelenti. Az a tapasztalatunk, hogy a ***Linux*** világában nagyon nagy hiány van az ügyviteli szoftverekből, és ezért az ***Sql-Ledger*** egy nagyon jó jelölt azoknak, akik ***Linuxot*** használnak.

Vannak-e olyan más szabad ügyviteli szoftverek, amik hasonlóak az SQL-Ledgerhez?
K.J.: Természetesen vannak, hiszen az egész világon fejlesztenek szabad szoftvereket ügyviteli célokra. Mi körülbelül két évvel ezelőtt kezdtünk ezzel komolyabban foglalkozni, és úgy ítéltük meg, az ***Sql-Ledgernek*** van a legbíztatóbb fejlesztési modellje és a legjobb fejlődési üteme. Most már elmondhatjuk, hogy megérzésünk nem okozott csalódást, hiszen azóta is nagyon dinamikusan fejlődik, és elmondhatjuk, hogy a kis- illetve középvállalatok számára a legjobb, legtöbb funkcióval rendelkező szabad szoftver.

Nem sok cég van idehaza, akik a szabad szoftverekhez nyújtanak támogatást, tehát tulajdonképpen honnan jött az ötlet, hogy ti egy szabad szoftvert karoltok fel, nem pedig egy külföldi kereskedelmi szoftverrel kezdtek foglalkozni?
K.J.: Természetesen nem az volt az első szándékunk vagy ötletünk, hogy mi ezt termékké alakítsuk át. Egy több milliárdos forgalommal rendelkező kereskedelmi cég keresett meg minket azzal, hogy szeretne egy olyan rendszert, amely az ő igényei szerint működik, és sokkal jobb megoldásnak tűnt, hogy keresünk egy szabad szoftvert, amit mi átalakíthatunk, ahelyett, hogy egy zárt forráskódú szoftvert fejlesztő céggel kezdünk egyezkedni, hogy „szabja testre" saját termékét a cég igényei szerint. A mi termékünk végül ebből a fejlesztésből alakult ki.

Tehát akkor tulajdonképpen ti is hozzáraktatok valamit az SQL-Ledgerhez. Tudsz mondani erre konkrét példát, vagyis valamit amit ti fejlesztettetek ki?
K.J.: Az ***Sql-Ledger*** egy globális könyvelési rendszer, olyan funkciókkal rendelkezik, ami a legtöbb országban működik. Magyarországon vannak olyan speciális tulajdonságai a könyvelési, ügyviteli szabályoknak, amelyek ebben a programban természetesen nem voltak benne. Magyarországon van az egyik legszigorúbb számlázási rendelet, a magyar könyvelők más szempontból tekintenek a könyvelésre, más módszer alapján könyvelnek le egy számlát, mint az angolszász országokban. Lévén az Sql-Ledger egy kanadai program, sok módosítást kellett rajta elvégezni ahhoz, hogy a magyar viszonyok között is kényelmesen működjön. Ezeket a módosításokat természetesen vissza lehetne rakni az eredeti programba, de nem kerülnek bele, hiszen itt hazánkra nézve teljesen specifikus dolgokról van szó.

Az eredeti programot ugyebár nem ti fejlesztitek, ti csak hozzáraktok bizonyos részeket. Nem okoz-e nagy problémát, ha a szoftver kanadai fejlesztők átalakítanak valamit? Könnyen hozzá tudjátok-e igazítani az új változathoz is saját fejlesztéseiteket, mondjuk egy modul formájában, vagy azért ennél egy kicsit problémásabb a helyzet?
K.J.: Igen, ez valóban probléma, hiszen figyelni kell arra, hogy az eredeti szerző mit fejleszt bele, illetve a mi változtatásainkat is figyelni kell, hogy jól működjön az új verziókkal. Ezt a problémát úgy oldottuk meg, hogy le lehet tölteni az eredeti verziót az oldalunkról valamint az eredeti verziónak a magyar fordítását, illetve a magyar sablonokat, tehát ha valakinek erre van szüksége, akkor nyugodtan ingyenesen használhatja, ha viszont már számlázni akar vele, vagy szeretne egy kicsit a magyar viszonyokhoz közelibb felületeteket kapni a munkája során, akkor minket keres meg, és a mi verziónkat adjuk oda, amibe természetesen egy kicsit

később kerülnek bele azok a magyar felhasználók számára is hasznos tulajdonságok, amelyeket a kanadai szerző folyamatosan fejleszt a programba.

Mivel bárki szabadon letöltheti, például a Ti oldalatokról is a forráskódot, ezért elvileg holnaptól bárki elkezdheti terjeszteni. Nem féltek ettől, vagy nem jelent ez valamilyen kockázatot számotokra?
S.A.: Természetesen sok energiát és munkát fektettünk a szoftver fejlesztésébe, de mi tudjuk, hogy a szolgáltatás árát kell úgy megállapítani, hogy az versenyképes legyen. A cél tehát az, hogy a hozzáadott érték, vagyis az a többlet, amit mi az SQL-Ledgerhez adunk, a legjobb legyen a piacon, és ezért potenciális ügyfeleink bennünket válasszanak. Természetesen itt is versenyhelyzet van, mint mindig és mindenütt, de ezt a versenyt mi álljuk.

Gondolom az ügyfeleitek egy része egy kicsit félt kezdetben a szabad szoftverektől. Nem tudták mondjuk, hogy elég megbízhatóak-e, illetve ha van benne valami hiba, akkor kié a felelősség. Ennek ugye jelen esetben akár komoly anyagi vonzata is lehet. Hogyan sikerült az ügyfeleiteket mégis megnyugtatni?
K.J.: A szolgáltatásunk nagyon fontos része a havi támogatás, tehát bármilyen probléma esetén minket fel lehet keresni e-mailben, szükség esetén távoli eléréssel megnézzük a rendszert, kijavítjuk a hibákat, vagyis gyorsan tudunk reagálni a problémákra. Ami a megnyugtatást illeti, jellemzően olyan emberek, olyan ügyfelek, cégek keresnek meg bennünket, akiket eleve vonz a szabad szoftverekkel kapcsolatos filozófia. Ha úgy döntenek, akkor nem a mi szolgáltatásunkat veszik igénybe, tehát megkereshetnek bárki mást, a kényszerfüggőség számukra nem létezik. Ettől függetlenül legtöbben minket választanak, hiszen mi értünk hozzá a legjobban, ami a testreszabást illeti mi tudunk a legjobb árakkal szolgálni, illetve mi tudunk a leggyorsabban reagálni a problémákra.

Akkor tehát úgy tűnik, hogy lassan kezd kialakulni a bizalom a szabad szoftverek iránt. Ti is így látjátok?
S.A.: Tény, hogy amit mi a szabad, vagy nyílt forráskódú rendszerek népszerűsítése terén teszünk, az egyfajta missziós tevékenység is. Azt próbáljuk minden potenciális ügyfelünkkel megértetni, hogy éppen akkor van biztonságban, ha a nyílt forráskódú, szabad szoftver mellett teszi le a garast, és éppen akkor futhat be egy veszélyes zsákutcába, ha valamilyen zárt forráskódú kereskedelmi szoftvert vásárol meg. Ilyenkor ugyanis a szolgáltató monopol helyzetben van és – durván fogalmazva – „elszemtelenedhet”.

Mik voltak azok az igények, amelyeket az ügyfeleitek a régi ügyviteli szoftverükkel nem tudtak megvalósítani, ugyanakkor az SQL-Ledger az erre már alkalmas volt?
K.J.: A legfontosabb, hogy Linux alatt stabilan fut, valamint, hogy szabad szoftver. Vonzó az is, hogy másképp van felépítve ez a program, tehát egy könyvelési tudással nem rendelkező ügyvezető is biztonsággal tudja használni. A program a bizonylatok rögzítésekor a háttérben automatikus lekönyveli a tételeket, ezért a kezelőnek nem kell tudnia, hogy milyen főkönyvi számokat kell beírnia az űrlapra, hogy ez rögzítésre kerüljön. A program ezt „magától” tudja.

Az SQL-Ledger mellett szól az is, hogy nagyon jól lehet használni intranet hálózatokon, illetve interneten keresztül is. Elég föltelepíteni egy szerverre, és a webes felületének köszönhetően bárhonnan elérhető a program. Az utazó eladók, otthonról netező ügyvezetők is meg tudják nézni, hogy miből mennyi van raktáron. Persze rögtön hozzá kell tenni, hogy megfelelő biztonsági eszközöket azért használni kell. Végül többen a nyitottsága miatt szeretik, azaz könnyen összeköthető más rendszerekkel, mint például Webáruház.

S.A.: Nem is az a lényeg, hogy ***Linuxon is*** fut, hanem, hogy alapvetően ***Linuxon*** fejlesztették, és így a rendszert kiszolgáló programok is szabadok és ingyenesek. Még az adatbáziskezelője (PostgreSQL) is ingyenes, sőt ismereteim szerinti ez a legfejlettebb adatbázis motor, ami Linux alatt elérhető. Nagyon sokan pontosan azért váltanak erre a szoftverre, mert egy fejlettebb felületet és egy fejlettebb rendszert szeretnének. A rendszer a cég vezetését átlátható információkkal látja el, ami nagyon fontos lehet a vállalat vezetőinek az üzleti döntések meghozatala során.

Nincs a webes felületnek valami hátránya? Mondjuk valamelyik böngésző esetleg nem tud kezelni bizonyos adatmennyiséget? A nyomtatásnál is a böngészőre vagytok utalva, vagy nem?
K.J.: Nos igen, a programnak az a legnagyobb hátránya, hogy az adatokat webes felületen kell rögzíteni, egy könyvelő pedig ahhoz van szokva, hogy könnyen és gyorsan tudjon felvinni dokumentumokat, számlákat. Már dolgozunk ennek a kiváltásán. A nyomtatás természetesen nem csak úgy működik, hogy a böngészőben megjelenítjük és onnan nyomtatjuk ki az anyagot. A program képes PostScript és PDF formátumot is előállítani, amelyet vagy megjelenítünk a képernyőn (és onnan nyomtatjuk ki), vagy közvetlenül a nyomtatóra irányítjuk.

Akkor gyakorlatilag milyen százalékban van a windowsos és linuxos kiszolgálóra telepített programok aránya?
S.A.: 99% Linux, 1% Windows.

K.J.: Ehhez annyit azért hozzátennék, hogy természetesen ***Windowsra*** is lehet telepíteni. Azok a segédprogramok, amik ***Linuxra*** ingyenesek, ugyanúgy ***Windowson*** is megvannak, és ugyanúgy ingyenesek. A ***Windowsnál*** egyedül az operációs rendszert kell megvásárolni. A ***windowsos*** változatnál annyi a probléma, hogy egyelőre nincs alaposan letesztelve, hiszen az adatbázismotor ***Windowsra*** írt natív verziója (a 8.0-ás) gyakorlatilag a közelmúltban jött a ki. Úgy érezzük ezt még alaposan le kell tesztelni ahhoz, hogy biztonsággal tudjuk támogatni. Alapos tesztelés után viszont ugyanolyan egyenértékű lesz, mint ***Linuxon***. Persze lehet keverni is a megoldásokat. Például meg lehet azt is oldani, hogy minden a ***Windowson*** fut, kivéve az adatbázismotort, amely ***Linux*** kiszolgálón van. Ha a szerveren kész a telepítés, attól kezdve a rendszer egy egyszerű böngészővel ellátott bármilyen operációs rendszert futtató munkaállomásról használható, legyen az ***Linux***, ***Windows*** vagy egy ***Macintosh***.

Az interjút Horváth Ernő készítette

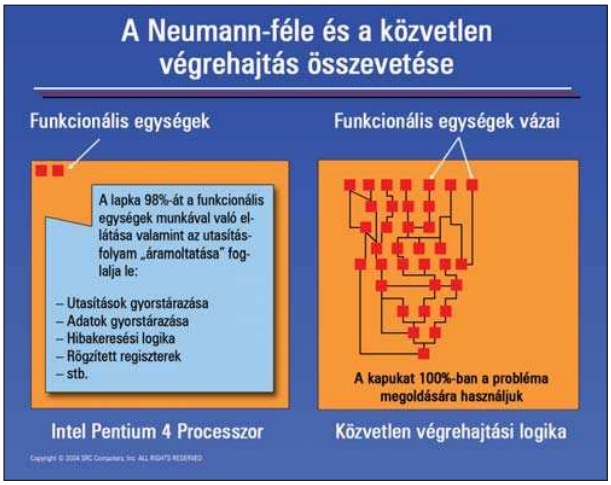
Program által meghatározott processzorok

A rendszerlogika menet közben megvalósított újraépítésével ez a módszer képessé tehet egyetlen FPGA-t tíz vagy akár több száz közönséges processzor feladatainak ellátására.

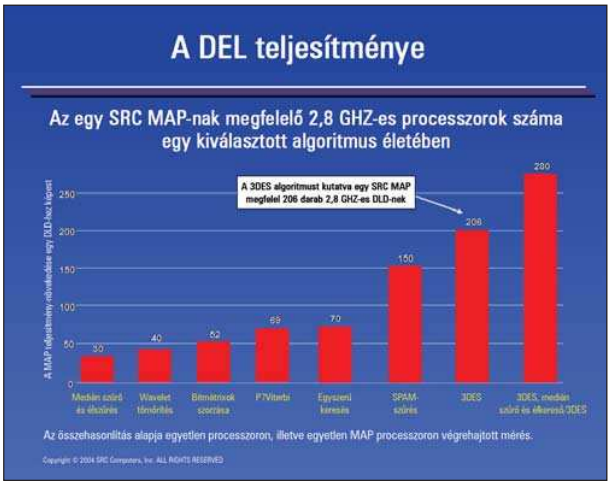
Az alkalmazás által meghatározott processzorok az ***átszerkeszthető számítási elv (reconfigurable computing, RC)*** elvén alapulnak. Az ***RC*** egy olyan számítási eljárás, amely elmossa a hardver és szoftver közti határvonalat és megteremti az alapjait annak, hogy újabb nagy lépést tegyünk a számítási hatékonyság növelése terén, csökkentett teljesítmény- és tárterület-igény mellett. Az ***RC*** gyakorlati megvalósítása az ***átszerkeszthető (reconfigurable)*** hardvereszközök alkalmazásával történik. Egy ***RC***-rendszerben lévő processzorok olyan hardvereszközök, amelyek a rajtuk futtatott programra lettek optimalizálva.
Cikkemben elmagyarázom az ***RC*** eljárás elvét, megvizsgálok néhány ***SRC***-rendszert, amelyek az ***RC*** gyakorlati megvalósítását jelentik, és megmutatom, hogy milyen teljesítménybeli előnyökkel bír az ***RC*** a hagyományos mikroprocesszorokhoz képest. Bemutatom emellett az ***RC*** programozási modelljét és az ***RC***-ben rejlő lehetőségeket az ***Open Hardware*** támogatására.

Mit jelent az átszerkeszthető számítási elv és miért fontos ez a számunkra?

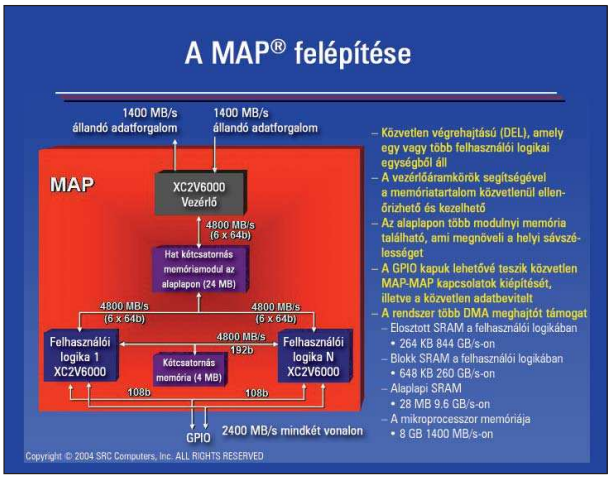
Az ***RC*** egy a hardveren alapuló számítási módszer, amely minden egyes futtatandó alkalmazás számára dinamikusan hozható létre. Az ***RC*** olyan hardverelemekből épül fel, amelyek dinamikusan megadható logikájú áramköri lapkákat tartalmaznak, vagyis az alkalmazott számítási módszer nem a gyártáskor kerül rögzítésre. Az ***RC*** már sok éve létezik, és számos hardverösszetevőben került már megvalósításra. Ilyenek az ***FPGA***-k ***(Field Programmable Gate Array, általános célú programozható áramkör)***, az ***FPOA***-k ***(Field Programmable Object Array***, az FPGA-hoz hasonló jellegű, de számos fontos tulajdonságában eltérő programozható áramkör - a ford.) és az ***összetett programozható logikájú eszközök (complex programmable logic devices, CPLD)***.
Az alkalmazásfejlesztők számára fontos tényező, hogy a modern újraszerkeszthető áramköri lapkák olyan órajellel és teljesítménnyel rendelkeznek, amelyek lehetővé teszik, hogy ***RC***-hardverekben nagyteljesítményű számításokra is alkalmazzák azokat.
Az ***RC*** megvalósítására használt legelterjedtebb lapkatípus az ***FPGA***. Az ***FPGA SRAM*** memóriacellákból felépülő lapka, amelyben ezek a memóriacellák tárolják a lapka beállításait. Az ***FPGA***-k logikai kapukat, flip-flopokat, ***RAM***-ot, aritmetikai magot, órajel-generátort és az ezek összekötésére szolgáló beállítható huzalozást tartalmaznak. Az ***FPGA***-k tetszőleges logikai funkció megvalósítására beállíthatók, így olyan egyedi processzorok hozhatók létre, melyek egy adott alkalmazásra optimalizálhatók.
Egy ***FPGA***-készlet így alkothat akár ***MIPS***, ***SPARC PowerPC*** vagy ***Xeon*** processzort, esetleg egy teljesen egyedi felépítéssel rendelkezőt is. Valójában az sem szükséges, hogy utasításfeldolgozó egységről legyen szó, ***közvetlen futtató logika (DEL, direct execution logic)*** is lehet, amely csak számítási logikát tartalmaz, és nincs szüksége az algoritmust meghatározó utasításokra.
A közvetlen futtató logikát tartalmazó (***DEL***) processzorok igen nagy teljesítmények elérését teszik lehetővé. Ezek a processzorok pontosan az adott algoritmus végrehajtásához szükséges erőforrásokkal hozhatók létre. A hagyományos utasításfeldolgozó egységek rögzített erőforrásokkal rendelkeznek, összeadókkal, szorzókkal, regiszterekkel és átmeneti tárakkal, és jelentős lapkafelület és teljesítmény szükséges az olyan többletmunka végrehajtásához, mint az utasítások visszafejtése, a végrehajtási sorrend megállapítása és az átmeneti tár kezelése.
A ***DEL***-processzorok olyan átrendezhető számítógépek, amelyekben minden alkalmazáshoz más-más felépítés tartozik szemben a rögzített felépítésű processzorokkal, amelyeknél mindent ugyanazzal az egységgel kell megoldani. A ***DEL***-processzorok a leghatékonyabb áramköri felépítést szolgáltatják egy adott alkalmazáshoz az algoritmusban található párhuzamosságok kezelésének és a funkcionális egységek pontosságának tekintetében.
Az átépíthetőségből adódóan minden program számára egyedi DEL-processzor hozható létre a másodperc törtrésze alatt.
De miért fontos számunkra, hogy a ***DEL***-processzorok dinamikusan hozatók létre egy alkalmazás számra és hogy hatékonyabban használják ki a rendelkezésükre álló áramköri lapkákat, mint a hagyományos mikroprocesszor?
A válasz egyszerű: hatékonyabb teljesítmény- és energiafelhasználás. Egy ***DEL RC*** processzor úgy építhető fel, hogy tartalmaz minden párhuzamosságot, ami az adott



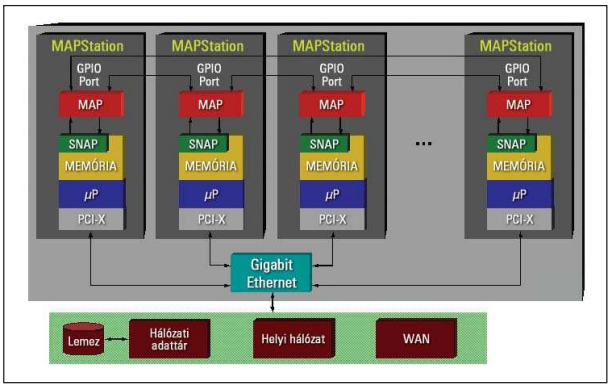
1. ábra A közvetlen futtatású logika (DEL) minden logikai kaput a valódi probléma megoldására mozgósít



2. ábra Számos 2,8 GHz-es processzor szükséges a MAP közvetlenül futtató processzor teljesítményének az eléréséhez



3. ábra A MAP szerkezeti felépítése



4. ábra A fürtözött SRC-6 rendszer felépítése

algoritmusban előfordul, a mikroprocesszorban lévő felesleges többletmunka nélkül. A cikk további részében a téma részletesebb tárgyalásának érdekében az ***RC*** processzorokat ***FPGA*** elemekkel megvalósított egységeknek feltételezzük.

Hogyan érhető el az a bizonyos nagy teljesítmény?

Az ***RC*** processzorok teljesítménye a logika párhuzamos futtatási képességéből adódik. Az ***RC*** processzorok teljes mértékben párhuzamos működésűek. Valójában egy adott algoritmusnak megfelelő logika létrehozása nem áll másból, mint a párhuzamos futások összehangolásából, vagyis hogy a részeredmények a megfelelő pillanatban jöjjenek létre, kerüljenek megosztásra illetve visszatartásra.

A ***DEL*** processzor az adatutak és vezérlőjelek által összekapcsolt funkcionális egységek hálózata. A hálózatban lévő minden számítási egység minden egyes órajelre aktívvá válik. Az *1. ábra* egy logikai részletet mutat egy kifejezés kiszámítására és a lapka kihasználtságára egy olyan Neumann-féle utasítás-processzorban, mint amilyen az ***Intel Pentium 4*** mikroprocesszor is.

Habár egy mikroprocesszor 3GHz körüli órajelen képes működni szemben az ***FPGA*** lapkák 100-300 MHz-es frekvenciájával, a párhuzamosságnak és a belső sávszélességnek köszönhetően egy ***DEL***-processzor az összteljesítményt tekintve nagyságrendekkel múlhatja felül a mikroprocesszort. A *2. ábrán* néhány összehasonlító teljesítményteszt látható, amelyben az ***SRC DEL***-processzora a ***MAP*** és egy jellegzetes Neumann-féle utasításvégrehajtó processzor, az ***Intel Xeon*** 2,8 GHz mikroprocesszor méri össze erejét. A párhuzamos futás, a pontosan a szükséges számú funkcionális egység alkalmazása, a nagy belső sávszélesség, az utasítás feldolgozásából, betöltéséből és tárolásából adódó többletmunka kiküszöbölése együttesen vezetnek a ***MAP*** és ***Intel*** processzorok közti 30-szoros órajelkülönbséghez.

Képes a DEL processzor a Linux futtatására?

Elvileg a ***DEL*** alapú processzorok képesek lennének a ***Linux*** futtatására, de szükség van-e erre egyáltalán? A ***Linux*** rendszermagjának kódszegmensei minden bizonnyal nagyobb teljesítménnyel futnának egy ***DEL*** processzoron, és a ***Linux*** rendszercsomag programjai is éreznék ennek az előnyét. Mégis, egy operációs rendszernek, és különösen a rendszermagnak az a szerepe, hogy kezelje a hardvert, és elérhetővé tegye a programok számára a megfelelő teljesítményszintet. Más szóval

az operációs rendszernek félre kell állnia az útból és hagyni, hogy az alkalmazások kihasználhassák a hardver nyújtotta szolgáltatásokat.
A programok nem csak elmélyült számítási műveletekkel foglalkoznak, hanem emellett kapcsolatot tartanak a felhasználókkal, fájlokat olvasnak és írnak, megjelenítik az eredményeket és a Világhálón keresztül információt cserélnek a világgal. Az alkalmazásoknak tehát egyaránt szükségük van számítási erőforrásokra és egy operációs rendszer szolgáltatásaira. A nehéz számítási műveletek és a magas fokú párhuzamosság ki tudja használni a *DEL* processzorok előnyeit. Bár a soros kódok is futhatnak közvetlen futtató logikán, ezeket mégis a hagyományos processzorok tudják a legjobban kiszolgálni.
A legtöbb alkalmazás számára az optimális megoldást a hagyományos és *DEL* processzorok keveréke jelenti. Ez a kombináció lehetővé teszi az alkalmazások számára, hogy nagyságrendekkel nagyobb teljesítményt érjenek el, miközben a hagyományos ***Linux*** környezetben futnak és rendelkezésükre áll az operációs rendszer összes szolgáltatása és megszokott eszköze.
Az alkalmazásnak azok a részei, amelyben túlsúlyban vannak a soros kódok, vagy amelyek az operációs rendszer szolgáltatásait igénylik, a rendszer hagyományos processzort tartalmazó részén futhat, míg azokat az alkalmazásokat, sőt az operációs rendszer bizonyos részeit, amelyek számára előnyös a *DEL* párhuzamossága, a szorosan a rendszerhez kapcsolt *DEL* processzorok szolgálják ki.

Az SRC Computers RC-rendszere

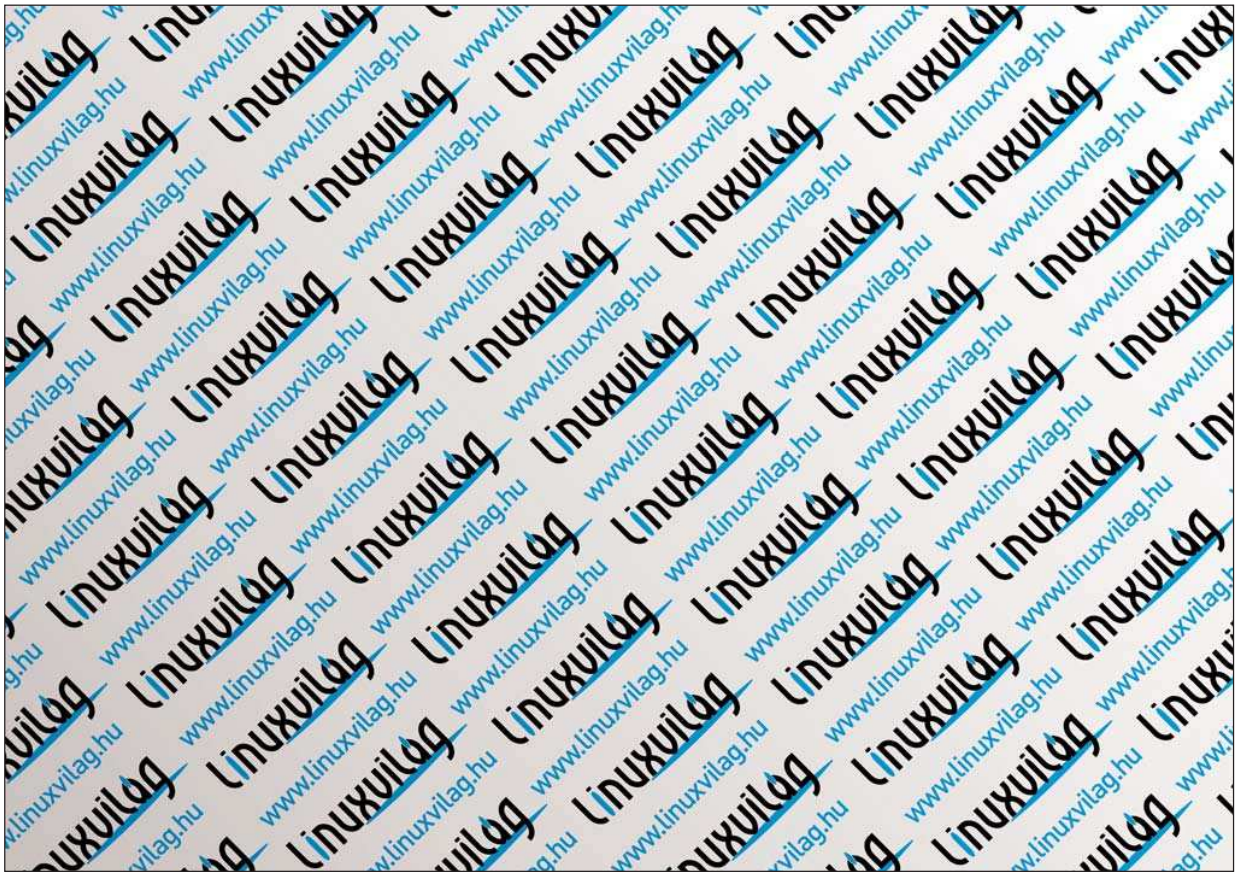
Az ***SRC*** létrehozott ***DEL***-processzorokat és mikroprocesszorokat egyaránt tartalmazó rendszereket. Ezek a rendszerek operációs rendszerként a ***Linuxot*** futtatják, emellett biztosítanak egy ***Carte*** nevű programozói környezetet, amely alkalmas a mikroprocesszoros utasításokat a ***DEL***-lel ötvöző programok írására, és egyetlen rendszeren belül támogatja a mikroprocesszorra és ***DEL***-processzorra épülő hardverhátteret.

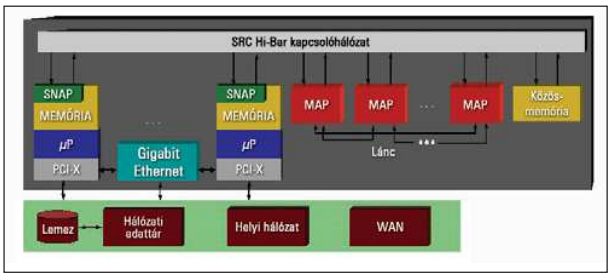
A DEL-processzor: MAP

A ***MAP*** az ***SRC*** nagyteljesítményű, szabadalmazott ***DEL***-processzora. A ***MAP*** átszerkeszthető összetevőket tartalmaz a vezérlés és felhasználó által meghatározott számítások megvalósítására, az előzetes utasításkód-lehívásra és az adatelérésre. Ez a számítási képesség igen nagy belső és külső sávszélességgel párosul. A ***MAP*** kétkapus alaplapra integrált memóriái 11,2 GB/sec helyi átviteli sebességet biztosítanak. A ***MAP*** külön bemeneti és kimeneti kapukkal van felszerelve, amelyek 1,4 GB/sec adatforgalom kezelésére képesek. Ezen felül minden ***MAP*** rendelkezik két általános I/O (***GPIO***) kapuval további 4,8 GB/sec sávszélességet biztosítva a közvetlen MAP-MAP kapcsolat vagy a forrásadatok számára. A ***MAP***-processzor szerkezeti felépítését a *3. ábrán* láthatjuk.

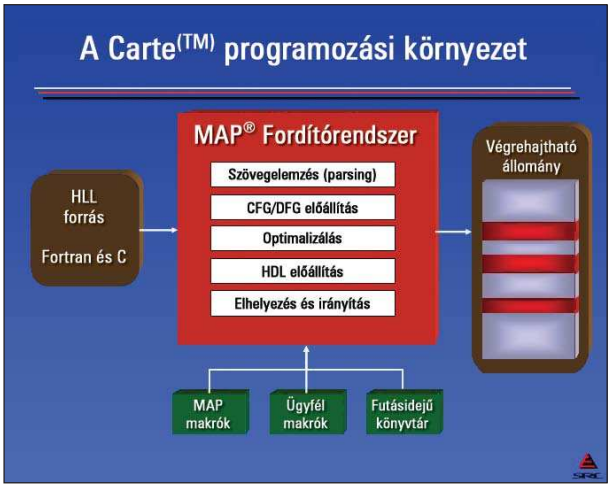
A mikroprocesszorok kapcsolódása a SNAP segítségével

Az ezekben az eszközökben használt ***DLD***-k ***(Dense Logic Device, hagyományos fix logikájú eszközök)*** kétprocesszoros ***Intel IA-32*** egységek. Ezek a külső

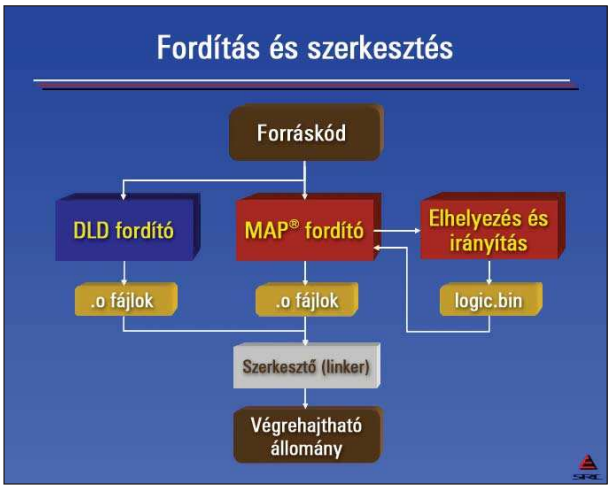




5. ábra A Hi-Bar kapcsolóval felépített SRC-6 elrendezése



6. ábra A Carte programozói környezet



7. ábra A Carte fordításának folyamata

gyártótól származó áramkörök az *SRC* által kifejlesztett *SNAP*-felülethez csatlakoznak. A *SNAP* lehetővé teszi a hagyományos alaplapok csatolását és a memóriamegosztást a *MAP* processzorokkal és a közös memóriacsomópontokkal, amelyek az SRC-rendszer további észét képezik.
A *SNAP*-felületet úgy tervezték, hogy az ne a mikroprocesszor ki/bemeneti alrendszerére, hanem közvetlenül a memória-alrendszerhez csatlakozzon, ezáltal lényegesen nagyobb csatlakozási sávszélességet biztosítva.
A *SNAP* külön kimeneti és bemeneti kapukat használ, amelyek jelenleg 1,4 GB/sec átviteli sebességet tesznek lehetővé.
A *SNAP* intelligens *DMA*-vezérlője olyan összetett utasításkód-előreolvasásra és adatelérési műveletekre képes, mint az adattömörítés, lépdelő (strided) adatelérés és a szórás/összegyűjtés, amelyek mindegyike a rendszerkapcsolatok rendelkezésére álló átviteli sávszélességének minél jobb kihasználását célozza. A kapcsolódás hatásfoka több mint tízszer jobb egy átmeneti tárat használó mikroprocesszorénál, amely az ilyen műveletekhez általánosan használt kapcsolatokat alkalmazza.
A *SNAP* csatlakozhat közvetlenül egyetlen *MAP*-hez, vagy az *SRC Hi-Bar* elosztójához több *MAP*, mikroprocesszor, vagy a közös memória eléréséhez.

Az SRC-6 rendszer szintű felépítésének megvalósítása

A rendszer-szintű beállítások vagy egy *MAP*-állomásokból álló fürtöt, vagy egy kapcsolóval megoldott kereszt-elrendezést valósítanak meg. A *4. ábrán* is látható fürt-alapú rendszerek a mikroprocesszort és a korábban ismertetett *DEL*-processzort közvetlen kapcsolatban használják. Habár ez az elrendezés a hagyományos és *DEL*-processzor szoros kapcsolatára épül, mégis ki tudja használni a szabványokra épülő fürtözési technikát a nagyon gyors rendszerek megvalósításához.
Amikor ennél nagyobb rugalmasságra van szükség, alkalmazhatók a *Hi-Bar* kapcsolókra épülő rendszerek. A *Hi-Bar* az *SRC* saját fejlesztésű skálázható, nagy sávszélességű és kis válaszidejű kapcsolója. Minden *Hi-Bar* támogatja a 64 bites címzést és 16 bemeneti valamint 16 kimeneti kapuval rendelkezik a 16 csomóponthoz való csatlakoztatáshoz. A *Hi-Bar*-hoz csatlakozhatnak mikroprocesszorok, *MAP*-ek és közös memóriák bármilyen, a 4. ábrán is látható elrendezésben. Minden be- és kimentei kapu 1,4 GB/sec átviteli sebességgel rendelkezik, amely így egy kettéosztott 22,4 GB/sec sávszélességgé adódik össze a 16 kapun. A kapu-kapu késleltetés 180 ns a kapunkét megvalósított ***egyszeres hibajavítással és kétszeres hibaészleléssel (SECDED, Single Error Correction Double Error Detection)***.
A *Hi-Bar* kapcsolók többsoros elrendezésben is összekapcsolhatók, lehetővé téve, hogy két sor 256 csomópontot kezeljen. Minden *Hi-Bar* kapcsoló egy 2U magas, 19 hüvelyk széles keretszerelésű készülékházban foglal helyet a tápegységével és a hűtőrendszerével együtt, így könnyen beépíthető a kiszolgálókeretekbe.
A *Hi-Bar* kapcsolókkal megvalósított összekapcsolást használó *SRC*-kiszolgálók egyesítik a közös memória csomópontokat a mikroprocesszorokkal és a *MAP*-ekkel. Minden egyes közös memória csomópont saját intelligens *DMA*-vezérlővel és akár 8 GB DDR SDRAM-mal rendelkezik. Az *SRC-6 MAP-*, *SNAP-* és közös memória csomópontjai (CM) 64 bites virtuális memóriacímzést használnak a rendszerben lévő összes memória címzéséhez, ami lehetővé teszi az alkalmazások számára, hogy egyetlen egybefüggő memóriaként kezeljék a teljes rendelkezésre álló tárterületet. Minden csomópont 1,4 GB/s elsőbbségi írási és olvasási átviteli sebességgel rendelkezik.
A *CM* intelligens *DMA*-vezérlője olyan összetett *DMA*-

1. táblázat *Karakter-összehasonlítási eredmények*

Megvalósítás	Szövegméret	Minta	Keresési idő	Sebesség
Brute Force (Xeon)	20MB	6	0.827 sec	1.00×
Boyer-Moore (Xeon)	20MB	6	0.597 sec	1.38×
Brute Force (MAP)	20MB	6	0.0143 sec	57.75×
Brute Force (Xeon)	20MB	10	1.398 sec	1.00×
Boyer-Moore (Xeon)	20MB	10	1.051 sec	1.33×
Brute Force (MAP)	20MB	10	0.0141 sec	98.81×

2. táblázat *Egy titkosított szövegben való keresés eredményei*

Megvalósítás	Szövegméret	Minta	Keresési idő	Sebesség
DES-Brute Force (Xeon)	20MB	6	2.77 sec	1.00×
DES-Boyer-Moor (Xeon)	20MB	6	2.63 sec	1.05×
DES- Brute Force (MAP)	20MB	6	0.0143 sec	193.09×
DES-Brute Force (Xeon)	20MB	10	3.31 sec	1.00×
DES-Boyer-Moor (Xeon)	20MB	10	3.11 sec	1.06×
DES- Brute Force (MAP)	20MB	10	0.0143 sec	231.76×

műveletek végrehajtására is képes, mint az adattömörítés, lépdelő adatelérés éss a szórás/összegyűjtés a rendszer belső sávszélességének minél hatékonyabb kihasználására. A belső kapcsolatok hatékonysága tízszer akkora, mint az átmeneti tárhelyekre épülő mikroprocesszoré, amely ugyanezt az ilyen műveletekhez általánosan használt kapcsolatot alkalmazza.
Ráadásul az *SRC* közös memória csomópontjai kizárólagos szemafor-kapcsolással rendelkeznek, amely szintén elérhető Az összes *MAP*-processzor és mikroprocesszor számára a szinkronizáláshoz.

Az átszerkeszthető számítási mód programozási modellje

Az *RC* programozási modellje a hagyományos elgondolás szerint egy hardvertervezési szempont. Mivel az *RC* alapját képező *FPGA*-technológia által megkövetelt eszközök az elektronikai tervezési iparból származnak, egy szoftverfejlesztő tényleg nem sok eszközt találhatott ismerősnek ezen a területen. Az eszközök az olyan ***hardverleíró nyelveket (HDL)*** támogatták, mint a *Verilog*, *VHDL* és a *Schematic Capture*.
A *SOC*-rendszerek ***(system-on-a-chip, egylapkás rendszer)*** technológiájának megjelenésével és az egyre összetettebbé váló hardverleíró meghatározásokkal elérhető közelségbe kezdtek kerülni a magasszintű programozási nyelvek. A *Java* és *C*-szerű nyelvek használata egyre általánosabb az *RC*-lapkák programozása terén. Ez egy nagymértékű előrelépést jelent, ami azonban az alkalmazásprogramozóktól is jelentős mértékű váltást követel.
Az *SRC*-programozási modell egy hagyományos programfejlesztő modell, amelyben a *MAP*-processzor programozására a *C* és a *Fortran* használatos, és bármely nyelv, amely összeköthető a futásidejű (C-ben írt) programkönyvtárakkal, lefordítható és futtatható a rendszer mikroprocesszoros részén.
Az ***SRC Carte*** programozói környezet azzal a tervezői feltételezéssel készült, hogy az alkalmazásfejlesztők az *RC* platformra fogják fejleszteni és átültetni a programjaikat. Emiatt az ***SRC-6*** rendszerre való fejlesztés során a hagyományos fejlesztési és tervezési elvek érvényesülnek, a ***magasszintű programnyelven (HLL)*** való kódolás, a programfordítás, a szabványos hibakeresővel való ellenőrzés, a kód javítása. Csak amikor már hiba nélkül fut az alkalmazás egy mikroprocesszoros környezetben, akkor kerül sor a program *DEL*-processzorra, a *MAP*-re történő átfordítására.
Egy *RC*-rendszerre történő programfordítás két olyan lépésből áll, amely meglehetősen idegen az utasításvégrehajtó processzorra történő programozáshoz képest. A *HLL*-fordítóprogram kimenetének egy hardverdefiníciós nyelvnek kell lennie. A *Carte*-ben ez vagy a *Verilog* vagy az ***EDIF (Electronic Design Interchange Format, elektronikus tervezés adatcsere-formátuma)***. Az *EDIF*-fájlok azok a hardverleíró objektumfájlok, amelyek az *RC*-lapkákban megvalósított áramköröket írják le. Ha a kimenet *Verilog*, akkor ezt a *HDL*-t *EDIF* formátumúra kell hozni egy olyan *Verilog*-fordítóval mint a ***Synplicity Synplify*** nevű programja.
Egy utolsó lépés, az elhelyezés és útvonalkijelölés, veszi az *EDIF*-fájlokból álló készletet és létrehozza az *RC* lapkán az áramkörök fizikai elrendezését. A folyamat bemeneti fájljai olyan konfigurációs bitfolyamok, amelyek az *FPGA*-ba tölthetők az *RC*-processzorba programozandó algoritmus hardveres kialakításának létrehozása céljából.
A *C* vagy *Fortran* nyelvből az *FPGA* által használható bitfolyammá történő fordítást a *Carte* programozói környezet végzi el anélkül, hogy a programozónak a folyamatba bele kellene avatkoznia. A program a mikroprocesszorokba szánt kódokat tovább fordítja objektummodulokká.
A *Carte* számára az utolsó lépés annak az egységesített futtatható fájlnak a létrehozása, amely egyetlen linuxos futtatható fájllá olvassza össze a mikroprocesszor objektummoduljait, a *MAP*-bitfolyamokat, és az összes szükséges futtatói programkönyvtárat. A *6.* és *7. ábrán* a *Carte* fordítófolyamatát láthatjuk.

A nyílt forráskód hardver-lehetőségei

A *Linux* élen járt és jó hasznot húzott a nyílt forráskód mozgalmából, amelynek során a programfejlesztők egy elkötelezett csoportja létrehozta, és továbbfejlesztette a *Linux* rendszermagját, mégpedig az újítások és minőség olyan szintjén, ami nem mérhető egyetlen kereskedelmi programokkal foglalkozó vállalathoz sem. Az átszerkeszthető számítási módban megvan annak a lehetősége, hogy a hardver tervezési szintjén használja ki ezeket az újításokat és technikai előnyöket. Ennek a cikknek a jelentős

része azzal foglalkozik, hogy bemutassa azt az elvet, ahogyan az alkalmazások programozói a szabványos programozói eljárásokat használva hozzák létre az alkalmazástól függő hardvert, anélkül, hogy ismerniük kellene a hardver felépítését. Az *RC*-ben ennek ellenére az alkalmazásprogramozók által létrehozott építőelemek a funkcionális egységek. Ezek az egységek olyan számítási alapműveleteket végeznek, mint az összeadás, lebegőpontos szorzás vagy a trigonometrikus függvények. A funkcionális egységek ezen felül lehetnek olyan speciális nagyteljesítményű egységek is, mint a háromszoros titkosítást (DES) megvalósító függvény vagy egy olyan, nem szabványos pontosságú aritmetikai egység, mint a 24 bites IEEE lebegőpontos operátorok.

A funkcionális egységeket a logikai tervezők hozzák létre. Az *RC*-fordítóprogramok, mint amilyen az *SRC Carte MAP* fordítója, képesek arra, hogy lehetőséget biztosítsanak a fordítóprogram által támogatott szabványos operátorok mellett felhasználó által létrehozott funkcionális egységek hozzáadására. Ha ilyen újszerű és eredeti funkcionális egységeket teszünk elérhetővé az alkalmazásprogramozók számára, még nagyobb teljesítmények válhatnak elérhetővé.

A funkcionális egységek újszerű hardverfelépítésének létrehozása az a terület, ahol a nyílt hardver mozgalom jelentős előrelépést hozhat a számítástudományba. Az újítás és termékenység, amit a nyílt forráskód területén tapasztalhattunk, most a nyílt hardver köntösében jelenhet meg újra. Az *RC* még nagyon sok kreatív tervezőnek kínál eszközt arra, hogy új és eredeti hardvert hozzon létre, amit azután az alkalmazásfejlesztők hasznosíthatnak. Az *Opensources.org*-hoz hasonló csoportokon keresztül pedig megoszthatók és továbbfejleszthetők ezek a funkcionális egységek. Az a jelentős előny, ami a nyílt forrású programoknak köszönhetően a számítástudományban megmutatkozott, könnyen jelentkezhet egy nyílt hardverre összpontosító mozgalomban is.

Kódolási példa

A *DEL*-processzorban rejlő teljesítményelőnyt egy karakterlánc-összehasonlító példán keresztül fogom megvilágítani. A példákhoz tartozó forráskód letölthető a *Linux Journal FTP*-oldaláról (lásd a kapcsolódó címeket). A példa *Christian Charras* és *Thiery Lecroq* honlapjáról származik és a *NIST Dictionary of Algorithms and Data Structures* (az algoritmusok és adatszerkezetek NIST szótára) hivatkozik rá. Összehasonlításképpen a „nyers erő” és a *Boyer-Moore* féle karakterlánc-összehasonlító algoritmust valósítjuk meg egy 2,8 GHz-es *Intel Xeon* processzoron az *Intel C++ 8.0 Linuxos* fordítóprogramjával.

A „nyers erő” algoritmust az *SRC*-rendszerhez a *Carte 1.8* verziójú programozói környezettel állítjuk elő. A „nyers erő” algoritmus egy egyszerű karakterenkénti összehasonlítást végez a karakterlánc és a minta között. A *Boyer-Moore* algoritmust tekintik a leghatékonyabb karakterláncösszehasonlító eljárásnak. A példában egy 20 MB-os véletlenszerűen előállított szövegben keresünk hat illetve tíz mintát. A fordításokat -O3 optimalizáló beállítással végeztük. Az összehasonlító eredményeket az *1. táblázat* tartalmazza. További keresési mintákat adva a feladathoz a mikroprocesszor végrehajtási ideje megnövekszik, azonban a *MAP* futási idejére nincs hatással, köszönhetően a párhuzamos végrehajtásnak. Bár a *Xeon* 2,8 GHz-en fut, a *MAP* pedig 100 MHz-en, a *DEL* párhuzamossága mégis 99-szeres teljesítményelőnyt biztosít a *MAP* számára. A példa a *MAP* egyetlen *FPGA*-jának 60%-át vette igénybe. Egy kétlapkás összeállítás több mint 200-szoros teljesítményt eredményezne.

Annak a bemutatására, hogy egy további számítás hozzáadása a csővezetékkel ellátott ciklushoz miként befolyásolja a teljesítményt és hogy szemléltetni tudjam az egyedi funkcionális egységek képességeit, egy másik összehasonlítást is elvégeztem, amelyben a keresési eljárás egy *DES*-titkosítással kódolt szöveggel teszteltem. A szöveget a keresés előtt dekódolni kell. A *MAP* megvalósításban egy *DES* csővezetékes egységet alkalmaztam. A *Verilog* nyelvű leírás az *Opencores.org* oldalról származik, ezt építettem be a keresési ciklusba. Mivel a ciklus csővezetéket alkalmaz, órajelenként egy egész eredményhalmazt szolgáltat. Emiatt a 20 MB-os szövegben való keresés és a *DES*-dekódolás becsült ideje nem változik a keresések számának növelésével. Ez vezet a megdöbbentő 232-szeres sebességnövekedéshez a mikroprocesszoros megoldással szemben. A 10 mintás keresés *MAP* megvalósítása csak az *FPGA* teljesítményének 70%-át vette igénybe, így egy kétlapkás felépítés 460-szoros többletet eredményezne.

A *Xeon* processzoron megvalósított dekódolás *Stuart Levy (Minnesota Supercomputer Center)* optimalizált kódját alkalmazta.

Összegzés

Cikkemben elmagyaráztam az átszerkeszthető számítási mód elvét, valamint példákat mutattam a módszerekre és az elérhető eredményekre. Látható, hogy nagyon jelentős teljesítménytöbblet érhető el ilyen módon. Jelenleg az *RC* nagymértékben hozzájárulhat a számítástudomány további fejlődéséhez és a jövő is sokkal többet tartogat a számunkra, mint az a mikroprocesszorok fejlődésének Moore-törvénye alapján várható lenne. Az *RC* már most elérhető a programozók számára, akik használhatják a megszokott fejlesztői modellt, és egy olyan keretrendszer is rendelkezésre áll, amellyel a hardvertervezők szélesebb köre is bekapcsolódhat a nagyteljesítményű számításokba a nyílt forrás biztosította kreativitás és alkotóerő kihasználásával. Az *RC* már hosszú ideje jelen van, a jelenlegi szoftver- és hardvertechnológia megteremtette annak a lehetőségét, hogy minden számítógépnek a részévé váljon a beépített processzoroktól a *Peta-Scale* szuperszámítógépekig.

Linux Journal 2005. január, 129. szám

Kapcsolódó címek: ➲ ***www.linuxjournal.com/article/7867***

Dan Poznanovic (poz@srccomp.com) a szoftverfejlesztés vezetője az SRC Computers-nél. A nagyteljesítményű számítások kutatásában 1987 óta vesz részt, amikor a Cray Research céghez csatlakozott.

ATA Over Ethernet: merevlemezek a helyi hálózaton

Napjainkra az ATA felületű merevlemezek olcsóbbak lettek, mint a szalagok, a címben szereplő egyszerű, új megoldással pedig archívumok, biztonsági mentések vagy éles használat céljára építhetünk tárolótömböket.

Előbb-utóbb mindenki kifogy a tárolóhelyből. Szerencsére a merevlemezek egyre olcsóbbak, miközben kapacitásuk folyamatosan növekedik. Hiába azonban a több hely, hiszen egyre többet használunk el, így újra szűkébe kerülünk.

Bizonyos adatfajták természetüknél fogva méretesek. A mozgóképek például mindig nagy mennyiségű helyet foglalnak. A vállalkozásoknak sokszor kell mozgóképeket tárolniuk, különösen, mióta elkezdődött a videómegfigyelő rendszerek térnyerése. Sőt, még otthoni gépükön is sokan szeretnek filmeket nézni vagy szerkeszteni.

A biztonsági mentés és a redundancia minden számítógépeket használó vállalkozás számára fontos fogalom. A valóság nagyjából az, hogy bármennyi tárolókapacitásunk is van, sose árt még egy kicsit több. Elég csak arra gondolni, hogy még az elektronikus leveleknek szánt tárolóhely is folyton kevés – az internetszolgáltatók bizonyára tudnának erről mesélni.

A korlátlan tárolóhely akkor válhat valósággá, ha a meghajtók kikerülnek a számítógépházból, és ezzel a tárolóeszközök függetlenekké válnak az őket használó számítógépektől. A rugalmasságnak az egymással együttműködő összetevők szétválasztásával elért növelésére több területen is láthatunk példát, nem csak az adattárolásén. A moduláris forráskód rugalmasabban használható az előre nem látott igények kielégítésére, és egy több részből álló sztereó rendszer is érdekesebb összeállításokban telepíthető, mint egy mindent egyetlen dobozban tartalmazó.

A leginkább ismert példa a készen kapható adattárolásra talán a ***tárolóhálózat (storage area network, SAN)***. Emlékszem, amikor a *SAN*-ok megjelentek, hatalmas zsongás vette őket körül, és elég nehéz volt kideríteni, hogy valójában mik is. Amikor végre sikerült, csalódottan kellett tudomásul vennem, hogy a *SAN*-ok bonyolultak, egy-egy gyártóhoz kötődnek és drágák.

A *SAN*-ok támogatása érdekében a linuxos közösség mindennek ellenére fontos módosításokat hajtott végre a rendszermagban. A 2.6-os rendszermag újdonságainak jelentős részét a 2.4-es rendszermagsorozat nagyvállalati használatra szánt tagjainak szolgáltatásai ihlették, és napjaink üzembiztos rendszermagjai számos olyan dolgot tudnak, amit néhány éve még nélkülöznünk kellett.

Például hatalmas kapacitású blokkos eszközöket használhatunk, messze túlnyúlva a régi két TB-os határon, és nagyobb számú lemez egyidejű csatlakoztatása sem okozhat gondot. Szintén fontos fejlesztés a tárolókötetek kifinomult kezelése, ahogy a fájlrendszerek is hatalmas méretűekre hízhatnak, akár becsatolt állapotban, használat közben is.

Írásomban ismertetem, hogyan aknázhatjuk ki a rendszermag új szolgáltatásaiban rejlő lehetőségeket, hogyan vehetjük ki a lemezeket a számítógépekből, és hogyan léphetünk túl a tárolók használatával és kapacitásával kapcsolatos korábbi korlátokon. Az ***ATA over Ethernetet (ATA Ethernet felett, AoE)*** talán úgy szemlélhetjük, mint megoldást arra, hogy az *IDE* kábel helyére egy *Ethernet* hálózat kerüljön. Ha az adattárolást leválasztjuk a számítógépről, és kihasználjuk a két rendszerelem közötti *Ethernet* rugalmasságát, akkor a lehetőségeket csupán képzelőerőnk és új dolgok elsajátítására irányuló hajlandóságunk végessége fogja korlátozni.

Mi az AoE?

Az ***ATA over Ethernet*** egy az *IEEE*-nél ***0x88a2*** azonosítóval bejegyzett *Ethernet* hálózati protokoll. Az *AoE* alacsony szintű, jóval egyszerűbb a *TCP*/*IP*-nél, sőt, az *IP*-nél is. A *TCP*/*IP* és az *IP* az internet esetében fontos megbízhatóságot növelő feladatokat lát el, ám a számítógépeknek komoly munkát jelent az ebből fakadó bonyolultság kezelése.

Az *iSCSI*-t használóknak nyilván ismerős ez a *TCP*/*IP*-vel kapcsolatos gond. Az *iSCSI* egy másik megoldás be- és kivitelek *TCP*/*IP* feletti továbbítására, segítségével olcsó *Ethernet* készülékekkel lehet helyettesíteni a költséges *Fibre Channel* berendezéseket. Sok *iSCSI*-használó ***TCP tehermentesítő motorokat (TCP offload engine, TOE)*** kezdett vásárolni. A *TOE* kártyák olcsók, és alkalmasak arra, hogy az *iSCSI*-t használó gépek válláról levegyék a *TCP*/*IP* kezelésének terhét.

Érdekes megfigyelés, hogy a gyakorlatban a legtöbb esetben az *iSCSI*-t nem internet felett használják. Ha a csomagoknak egyszerűen a szomszéd szekrényben található gépbe kell befutniuk, akkor a nehézsúlyú *TCP*/*IP* protokoll használata túlzásnak tűnik.

A *TCP/IP* tehermentesítés helyett tehát nem volna jobb, ha teljesen elhagynánk az internetes protokollt? Az ***ATA over Ethernet*** protokoll pontosan ezt teszi, kihasználva a napjainkban kapható olcsó kapcsolók által kínált lehetőségeket. A korszerű kapcsolók képesek a folyamvezérlésre, maximális átviteli sebességet és minimális csomagütközést biztosítva. A ***helyi hálózaton (local area network. LAN)*** a csomagok sorrendje nem változik meg, és a hálózati hardver minden csomag sértetlenségét ellenőrző összeggel védi.

Minden ***AoE***-csomag vagy egy ***ATA*** meghajtónak szóló parancsot, vagy egy ***ATA*** meghajtótól érkező választ hordoz. A ***Linux*** rendszermag ***AoE*** illesztőprogramja minden ***AoE***-műveletet elvégez, és a távoli lemezeket normál, blokkos eszközökként teszi elérhetőkké, ilyen például a ***/dev/etherd/e0.0***. Az ***IDE*** illesztőprogram hasonlóan gondoskodik az ***IDE*** kábel végén található helyi meghajtó például ***/dev/hda*** alatti elérhetőségéről. Az illesztőprogram szükség esetén a csomagok újraküldését is elvégzi, vagyis az ***AoE*** eszközök a rendszermag többi része számára tökéletesen egyenértékűek a többi lemezzel.

Az ***ATA*** parancsok továbbítása mellett az ***AoE*** lekérdező csomagokkal a rendelkezésre álló ***AoE*** eszközöket is képes egyszerű módon azonosítani. Ennyi is az egész: ***ATA*** parancscsomagok és lekérdező csomagok.

Aki már dolgozott ***SAN***-nal, vagy legalábbis tanult az ilyen rendszerekről, bizonyára megdöbbenve mered maga elé: ha minden lemez a ***LAN***-ra csatlakozik, hogyan lehet korlátozni az elérésüket? Vagyis hogyan biztosítható, hogy az A gép feltörése után a ***B*** gép lemezei biztonságban maradjanak?

A válasz onnan indul, hogy az ***AoE*** nem irányítható. Azt, hogy mely számítógépek mely lemezeket látják, egyszerűen, alkalmi (ad hoc) ***Ethernet*** hálózatok összeállításával határozhatjuk meg. Mivel az ***AoE*** készülékek nem rendelkeznek ***IP***-címmel, nem okoz nehézséget az elszigetelt ***Ethernet*** hálózatok létrehozása. Egyszerűen adjunk áramot a kapcsolónak, majd csatlakoztassuk hozzá a kívánt rendszerelemeket. Emellett sok újabb kapcsoló kapu alapú ***VLAN*** szolgáltatást is nyújt, amivel a kapcsolót hatékonyan lehet különálló, egymástól elválasztott szórási tartományokra osztani.

Az ***AoE*** protokoll annyira egyszerű, hogy olcsó vassal is használható. Jelenleg a ***Coraid*** az egyetlen gyártó, mely ***AoE*** eszközöket kínál, ám várható, hogy hamarosan további gyártók és alkalmazásfejlesztők is felfedezik majd, hogy az ***AoE*** leírása csupán nyolc oldal hosszú. Érdemes ezt az egyszerűséget szembeállítani az ***iSCSI*** több száz oldalas, többek közt titkosítási szolgáltatásokat, forgalomirányítást és felhasználó alapú hozzáférés-kezelést taglaló leírásával. A bonyolultság mindig költséggel jár, és most már választhatunk, hogy igényeljük ezt a bonyolultságot, vagy inkább megtartjuk az árát.

A legegyszerűbb eszközök is lehetnek hatásosak. A ***Linux***-felhasználók számára aligha újdonság, hogy egyszerűsége ellenére az ***AoE*** lehetőségek elképesztő választékát kínálja, ha az adattárolás egyszer átkerült a hálózatra. Nézzünk tehát egy tényleges példát, majd vizsgáljuk meg a lehetőségeket.

Stan, a levéltáros

A következő példa igaz történetre alapul. ***Stan*** egy képzelt rendszergazda, aki a helyi önkormányzatnak dolgozik. Az új rendelkezések szerint minden hivatalos dokumentumot maradandóan archiválni kell. A város bármely polgára bármikor betekintést kérhet bármelyik dokumentumba. ***Stannek*** tehát hatalmas és korlátok nélkül bővíthető tárolóhelyre van szüksége, ugyanakkor a tárolóeszköz teljesítményének nem kell jobbnak lennie, mint egy átlagos, helyi, ***ATA*** felületű lemezének. Célja az, hogy minden adatot könnyen és gyorsan elő lehessen keresni.

Stan ismeri az ***Ethernet*** hálózatokat és a linuxos rendszereket, ezért az ***ATA over Ethernet*** kipróbálása mellett dönt. Vásárol néhány eszközt; összesen kevesebb mint 6500 dollárt fizetve az alábbiakért:

- Egy darab kétkapus Ethernet kártya, a kiszolgáló régi, 100 Mbps sebességű kártyája helyett
- Egy darab 26 kapus hálózati kapcsoló kettő darab gigabites kapuval
- Egy darab Coraid EtherDrive polc és tíz darab EtherDrive fiók
- Tíz darab 400 GB-os ATA meghajtó

A tíz fiókot befogadó polc három egység magas. Minden ***EtherDrive*** fiók egy-egy kisméretű számítógép, mely az ***AoE*** protokoll kezelésével hatékony módon illeszt a hálózatra egy-egy ***ATA*** lemezt. Az adatokat csíkozással elosztva a polc fiókjai között hasonló teljesítményt kapunk, mint helyi ***ATA*** meghajtókkal, tehát a gigabites összeköttetéssel hatékonyan ki tudjuk használni a rendelkezésünkre álló átviteli kapacitást. Bár ***Stan*** megtehette volna, hogy az ***EtherDrive*** fiókokat ugyanarra a hálózatra csatlakoztatja, amire mindenki más is kapcsolódik, ám biztonsági és teljesítménybeli szempontok miatt inkább úgy döntött, hogy a tárolórendszert külön hálózatra helyezi, és a kiszolgáló második kapujához, az ***eth1*** csatolóhoz kapcsolja.

Stan átolvassa a ***Linux Software RAID HOGYAN***-t (lásd az internetes forrásokat), majd úgy határoz, hogy ***RAID 10*** csíkozást használ, tükrözött lemezpárok felett. Bár ezzel a megoldással kevesebb tárkapacitást kap, mintha ***RAID 5***-öt használna, viszont a ***RAID 10*** a lehető legjobb megbízhatóságot nyújtja, miközben a processzorra a ***RAID*** kezelése miatt háruló többletterhelés minimális marad, valamint a tömb újraépítése is rövidebb ideig tart, ha egy-egy lemez kiesik.

Az ***LVM HOGYAN*** (lásd a forrásokat) átolvasása után ***Stan*** kidolgoz egy tervet, amellyel elkerülheti, hogy valaha is elfogyjon a szabad tárhely. A ***JFS*** fájlrendszer képes dinamikusan méreteződni akár egészen nagyra is, vagyis ***Stan JFS***-t tesz egy logikai kötetre. A logikai kötet ebben az esetben csak egy fizikai kötetre terjed ki, ez pedig a ***RAID 10*** blokkos eszköz lesz. A ***RAID 10*** a ***Coraid*** polcban lévő ***EtherDrive*** fiókokból jön létre, a ***Linux*** szoftveres ***RAID*** megoldásával. Ha később újabb polcra lesz szüksége, akkor majd létrehoz egy másik ***RAID 10*** készletet, abból lesz egy fizikai kötet, amelyre kiterjeszti a ***JFS***-t tartalmazó logikai kötetet.

1. kódrészlet Egy nagyszámú AoE meghajtóból álló szoftveres RAID eszköz létrehozásának első lépése az AoE telepítése, beállítása

```
# A gazdagép beállítása AoE használatára
# az AoE illesztőprogram lefordítása
# és telepítése
tar xvfz aoe-2.6-5.tar.gz
cd aoe-2.6-5
make AOE_PARTITIONS=1 install
# Az AoE nem igényel IP-címet!  :)
ifconfig eth1 up
# A hálózati csatoló felélesztése
sleep 5
# Az ATA over Ethernet illesztőprogram betöltése
modprobe aoe
# A rendelkezésre álló AoE lemezek megtekintése
aoe-stat
```

Az *1. kódrészlet* azokat a parancsokat tartalmazza, amelyeket ***Stan*** a kiszolgáló ATA over Ethernet előkészítésére használ. Az ***AoE*** illesztőprogramot az `AOE_PARTITIONS=1` átadott értékkel fordítja le, ugyanis 2.6-os rendszermagot futtató Debian sarge rendszere van. A ***sarge*** jelenleg nem támogatja a nagyobb értékű eszköz-alazonosítókat (lásd az Alazonosítók című széljegyzetet), ezért kikapcsolja a lemezek részekre osztásának támogatását, így ugyanis több lemezt tud használni. Figyelembe véve a ***Debian*** 292070-es számú hibáját, ***Stan*** telepíti a legújabb eszközleképezőt és az ***LVM2*** felhasználói programokat.

Alazonosítók

Ha egy program használni akar egy eszközt, akkor ezt jellemzően az eszközhöz tartozó különleges fájl megnyitásával teszi. Jól ismert példa a ***/dev/hda*** fájl. Ha kiadjuk az `ls -l` parancsot, akkor a ***/dev/hda*** esetében két számot látunk, ezek a 3 és a 0. A ***/dev/hda1*** fájl alazonosítója 1, főazonosítója viszont szintén 3.

A 2.6-os rendszermag megjelenése előtt az alazonosító ábrázolása nyolc biten történt, vagyis értéke 0 és 255 között lehetett. Mivel senkinek nem volt ilyen sok eszköze, a korlát tulajdonképpen senkit nem zavart. Most, hogy a lemezeket le lehet választani a kiszolgálókról, megváltozott a helyzet, ezért a 2.6-os rendszermag már 20 biten ábrázolja az eszközök alazonosítóját.

Így az alazonosító 1048576-féle értéket vehet fel, ami nagy segítség ahhoz, hogy a rendszerekhez nagyszámú eszközt tudjunk csatlakoztatni – csakhogy a változást nem minden program követte. Ha a glibc vagy valamelyik alkalmazás azt hiszi, hogy az alazonosítók még mindig nyolcbitesek, akkor gondjaink lesznek a 255-nél nagyobb alazonosítók használatával. Az átmenetet segíti, hogy az ***AoE*** illesztőprogramot a lemezrészek támogatása nélkül is le lehet fordítani. Ilyenkor egy-egy lemezhez 16 helyett csak egyetlen egy alazonosító tartozik. Nem baj tehát, ha a rendszer nem ismeri még a 2.6-os nagyméretű eszköz-alazonosítóit, akár 256 ***AoE*** lemezt is használhatunk.

2. kódrészlet A szoftveres RAID és az LVM kötetcsoport beállítása

```
# A RAID kezdeti értékadásának felgyorsítása
for f in `find /proc | grep speed`; do
        echo 100000 > $f
done
# Tükrök létrehozása (az mdadm kezeli a forró
# tartalékokat)
mdadm -C /dev/md1 -l 1 -n 2 \
        /dev/etherd/e0.0 /dev/etherd/e0.1
mdadm -C /dev/md2 -l 1 -n 2 \
        /dev/etherd/e0.2 /dev/etherd/e0.3
mdadm -C /dev/md3 -l 1 -n 2 \
        /dev/etherd/e0.4 /dev/etherd/e0.5
mdadm -C /dev/md4 -l 1 -n 2 -x 2 \
        /dev/etherd/e0.6 /dev/etherd/e0.7 \
        /dev/etherd/e0.8 /dev/etherd/e0.9
sleep 1
# A csík létrehozása a tükrök felett
mdadm -C /dev/md0 -l 0 -n 4 \
        /dev/md1 /dev/md2 /dev/md3 /dev/md4
# A RAID 10 megfeleltetése LVM fizikai kötetnek
pvcreate /dev/md0
# Bővíthető LVM kötetcsoport létrehozása
vgcreate ben /dev/md0
# A fizikai kiterjedés vizsgálata
vgdisplay ben | grep -i 'free.*PE'
# A teljes helyre kiterjedő logikai kötet
# létrehozása
lvcreate –extents 88349 –name franklin ben
modprobe jfs
mkfs -t jfs /dev/ben/franklin
mkdir /bf
mount /dev/ben/franklin /bf
```

A fájlrendszer és a logikai kötet létrehozására használt parancsokat a *2. kódrészlet* tartalmazza. ***Stan*** a kötetcsoportnak a `ben`, a logikai kötetnek pedig a `franklin` nevet adja. Ezután módosítania kell néhány dolgot az ***LVM2*** beállításaiban. Először is, szükség lesz egy `types = [ "aoe", 16 ]` sorra, ami lehetővé teszi, hogy az ***LVM*** felismerje az ***AoE*** lemezeket. Ugyancsak szükség van az `md_component_detection = 1` sorra, amelynek hatására a gép a ***RAID 10*** készleten belüli lemezeket figyelmen kívül hagyja, miután a teljes ***RAID 10*** készlet egyetlen a fizikai kötetet alkot.

Stan rendszerét magam is összeállítottam, ***Debian sarge*** operációs rendszerrel, kettő darab 2,1 GHz-es ***Athlon MP*** processzorral, 1 GB memóriával és egy ***Intel PRO/1000 MT*** kétkapus hálózati kártyával, ma már elavulófélben lévőnek számító 40 GB-os meghajtókkal. Hálózati kapcsolóként egy ***Netgear FS526T*** készüléket használtam. Ha a ***RAID 10*** készletet nyolc darab ***EtherDrive*** fiókra terjesztettem ki, akkor 23,58 MBps folyamatos olvasási és 17,45 MBps folyamatos írási sebességet sikerült elérnem. A mérések elvégzése előtt

3. kódrészlet Leválasztása nélkül úgy tudjuk bővíteni a fájlrendszert, hogy létrehozunk egy második RAID 10 készletet, hozzáadjuk a kötetcsoporthoz, majd kiterjesztjük a fájlrendszert

```
# A második polc RAID 10 készletét
# létrehozása után
# /dev/md5-ként adjuk hozzá a kötetcsoporthoz
vgextend ben /dev/md5
vgdisplay ben | grep -i 'free.*PE'
# A logikai kötet majd a jfs megnövelése
lvextend –extents +88349 /dev/ben/franklin
mount -o remount,resize /bf
```

egy 1 GB-os fájlt a */dev/nullba* másolva kiürítettem a gyorsítótárat, az írási időbe pedig egy `sync` parancs végrehajtását is belevettem.
Ebben az esetben a ***RAID 10*** készletnek négy csíkeleme volt, mindegyik egy pár tükrözött meghajtóból állt. Általános esetben egy ***EtherDrive*** fiókokból álló készlet átviteli sebességét a csíkelemek száma alapján tudjuk megbecsülni. Egy ***RAID 10*** készletben feleannyi csíkelem van, mint lemez, hiszen minden lemez tükrözve van egy másikra.
A ***RAID 5*** esetében egy lemez tárolja a paritásadatokat, a több lemez pedig csíkelemként veendő figyelembe.
A várható olvasási sebesség a csíkelemek száma szorozva 6 MBps-mal. Ha tehát ***Stan*** a nyolc fiókból álló ***RAID 10*** készlet helyett két polcot vásárolt volna, és a készletet 18 meghajtóból állította volna össze, akkor valamivel több, mint kétszer ekkora átviteli sebességet kapott volna. ***Stan*** azonban nem igényel ekkora átviteli sebességet, és egy viszonylag kisméretű, 1,6 TB-os fájlrendszer is megfelel az igényeinek.
A *3. kódrészlet* azt szemlélteti, hogy ***Stan*** milyen könnyen ki tudja bővíteni a fájlrendszert, ha vásárol egy második polcot is. A kódrészletben ***Stan mdadm-aoe.conf*** fájlja és indító és leállító parancsfájljai nem szerepelnek. A figyelő módban futó `mdadm` folyamattal az `mdadm` beállító fájl közli a forró tartalékok kezelési módját, így ezekkel bármikor helyettesíteni lehet bármelyik meghibásodott tükör bármelyik meghajtóját. Lásd még az ***mdadm man*** oldalának ***tartalékcsoportok (spare groups)*** című részét.
Az indító és leállító parancsfájlokat könnyen össze lehet állítani. Az indító parancsfájl egyszerűen összeállítja a tükrözött ***RAID 1*** majd a ***RAID 0*** készleteket, végül elindít egy `mdadm` figyelő folyamatot. A leállító parancsfájl leállítja az `mdadm` figyelőt, melyet követően leállítja először a ***RAID 0***, majd a tükrözött készleteket.

Blokkos tárolóeszköz megosztása

Láttuk, hogyan működik az ***ATA over Ethernet***, és az olvasóban bizonyára felmerül a kérdés: vajon mi történik, ha egy másik állomás is hozzá szeretne férni a tárolóhálózathoz? Van arra lehetőség, hogy a második állomás is befűzze a ***JFS*** fájlrendszert, és hozzáférjen ugyanazokhoz az adatokhoz? Nos, biztonságosan erre nincs mód. A ***JFS***-t az ***ext3***-hoz és a többi fájlrendszer túlnyomó részéhez hasonlóan egyetlen állomással való használatra tervezték. Az ilyen „egygépes” fájlrendszerek megsérülhetnek, ha ugyanazt a blokkos tárolóeszközt több állomás is befűzi. Ennek oka a puffer gyorsítótárban keresendő, ami a 2.6-os rendszermagokban egyesítve van a lapgyorsítótárral.
A ***Linux*** meglehetősen rámenős módon, minden lehetőséget megragadva gyorsítótárazza a fájlrendszer adatait a memóriában, kerülve a lassú blokkos tárolóeszközök használatát, és ezzel javítva a teljesítményt.
A gyorsítótárazás hatékonyságáról bárki meggyőződhet, ha kétszer egymás után lefuttat egy `find` parancsot ugyanazon a könyvtáron.
Bizonyos fájlrendszereket úgy terveztek, hogy több állomás is használhassa őket. Az úgynevezett fürtfájlrendszerek például megfelelő eljárásokat tartalmaznak annak biztosítására, hogy az összes állomás gyorsítótára összhangban maradjon a fájlrendszerrel. Kiváló példa erre a nyílt forrású ***GFS***.
A ***GFS*** külön fürtkezelő programot használ annak követésére, hogy ki szerepel a fájlrendszert használó állomások csoportjában. A fájlrendszert elérő állomások együttműködését zárak segítségével biztosítja.
Fürtfájlrendszer, például ***GFS*** használatával megoldható, hogy az ***Ethernet*** hálózat több állomása is használhassa ***ATA over Ethernet*** felett ugyanazt a blokkos tárolóeszközt. Ilyenkor nincs szükség például ***NFS***-kiszolgálóra, hiszen mindegyik állomás közvetlenül, a be- és kiviteleket szépen elosztva éri el a tárolóeszközt. Van azonban egy kis bökkenő. Minél nagyobb számú lemezt használunk, annál nagyobb a valószínűsége annak, hogy az egyik meghibásodik. Ezt a problémát általában ***RAID*** használatával szokták orvosolni, némi redundanciát hozva a rendszerbe. Sajnos a ***Linux*** szoftveres ***RAID***-je nem képes a fürtök kezelésére. Nincs tehát mód arra, hogy a hálózat összes állomása ***RAID 10*** készletet és `mdadm`-et használjon.
A ***Linux*** fürtkezelése ugyanakkor elképesztő tempóban fejlődik. Biztos vagyok abban, hogy egy-két év múlva kiváló, fürtképes ***RAID*** áll majd rendelkezésünkre. Addig is azonban más megoldásokat kell keresnünk az ***AoE*** alapú fürtök megosztására. Az alapötlet az, hogy központosítsuk a ***RAID*** szolgáltatást. Vásárolhatunk például egy vagy két ***Coraid RAIDBlade*** vezérlőt, majd a fürtcsomópontokkal az ezeken keresztül exportált tárolóeszközt érhetjük el. A ***RAIDBlade*** vezérlők minden mögöttük lévő ***EtherDrive*** fiókot képesek kezelni. Ha szeretünk barkácsolni, akkor ugyanezt a feladatot egy linuxos, szoftveres ***RAID***-et futtató géppel is megoldhatjuk, mely ***ATA over Etherneten*** keresztül maga teszi elérhetővé a lemezhibáktól mentesített blokkos tárolóeszközt. Például a `vblade` program (lásd a forrásokat) képes tetszőleges tárolóeszközt elérhetővé tenni ***ATA over Etherneten*** keresztül.

Biztonsági mentés

Mivel az ***ATA over Ethernet*** olcsó merevlemezeket csatlakoztat az ***Ethernet*** hálózatra, akár biztonsági mentések készítésére is alkalmazható. A mentési stratégia sokszor másodvonalbeli adattárolást is magába foglal, vagyis olyan tárolóeszközt, amely ugyan nem olyan gyors, mint az éles rendszer, de könnyebben hozzáférhető, mint a szalagos.
Az ***ATA over Ethernet*** segítségével olcsó ***ATA*** meghajtókat használhatunk másodvonalbeli adattároló eszközökként.

Sőt! Ha ilyen olcsó merevlemezeink vannak, és ennyire üzembiztos szoftveres ***RAID***-et tudunk alkalmazni, miért is ne használhatnánk mentési adathordozóként a merevlemezeket? A szalagokkal ellentétben ezek azonnali hozzáférést biztosítanak bármely archivált fájlhoz.
Sok új biztonsági mentési alkalmazás a fájlrendszerek mentési szolgáltatásait is képes kihasználni. Közvetlen hivatkozásokkal több teljes mentést is képesek elvégezni, a növekményes mentések hatékonyságával. További tudnivalók az internetes források között szereplő ***Backup PC*** és ***rsync*** hivatkozások révén érhetők el.

Összefoglalás

Olcsó merevlemezek közvetlenül a hálózaton – jogos a kérdés, hogyhogy korábban senkinek nem jutott eszébe? Az adattárolás a kiszolgálóktól való elválasztását érdemes lehet valamilyen egyszerű hálózati protokoll segítségével végezni, drága eszközök nélkül – még ha az egyszerű protokoll használhatósága a helyi ***Ethernet*** hálózatra korlátozódik is. A helyi hálózaton ugyanakkor nincs is szükség a mindenre kiterjedő szolgáltatásokat nyújtó internetes protokollok, például a ***TCP/IP*** bonyolultságára és többletterhelésére. Ha az adattárolást a helyi hálózatra akarjuk korlátozni, és az ***Ethernet*** hálózatok kiépítése révén kapott hozzáférésvezérlés kielégíti az igényeinket, akkor az ***ATA over Ethernet*** számunkra a megfelelő választás. Ha a tárolóprotokolltól titkosítási, irányíthatósági és felhasználó alapú hozzáférés-vezérlési szolgáltatásokat várunk, akkor az ***iSCSI***-val érdemes barátkoznunk.

Az ***ATA over Ethernet*** olyan egyszerű alternatíva, amely eddig egyszerűen hiányzott a linuxos adattárolási lehetőségek közül. Az egyszerűség egyben lehetőségeket is jelent. Az ***AoE*** által biztosított alapra tetszőleges adattároló megoldást felépíthetünk. Mindenkit szeretnék arra buzdítani, hogy engedje szabadjára a képzeletét, majd írja meg nekem, hogy milyen rendszert sikerült felépítenie.

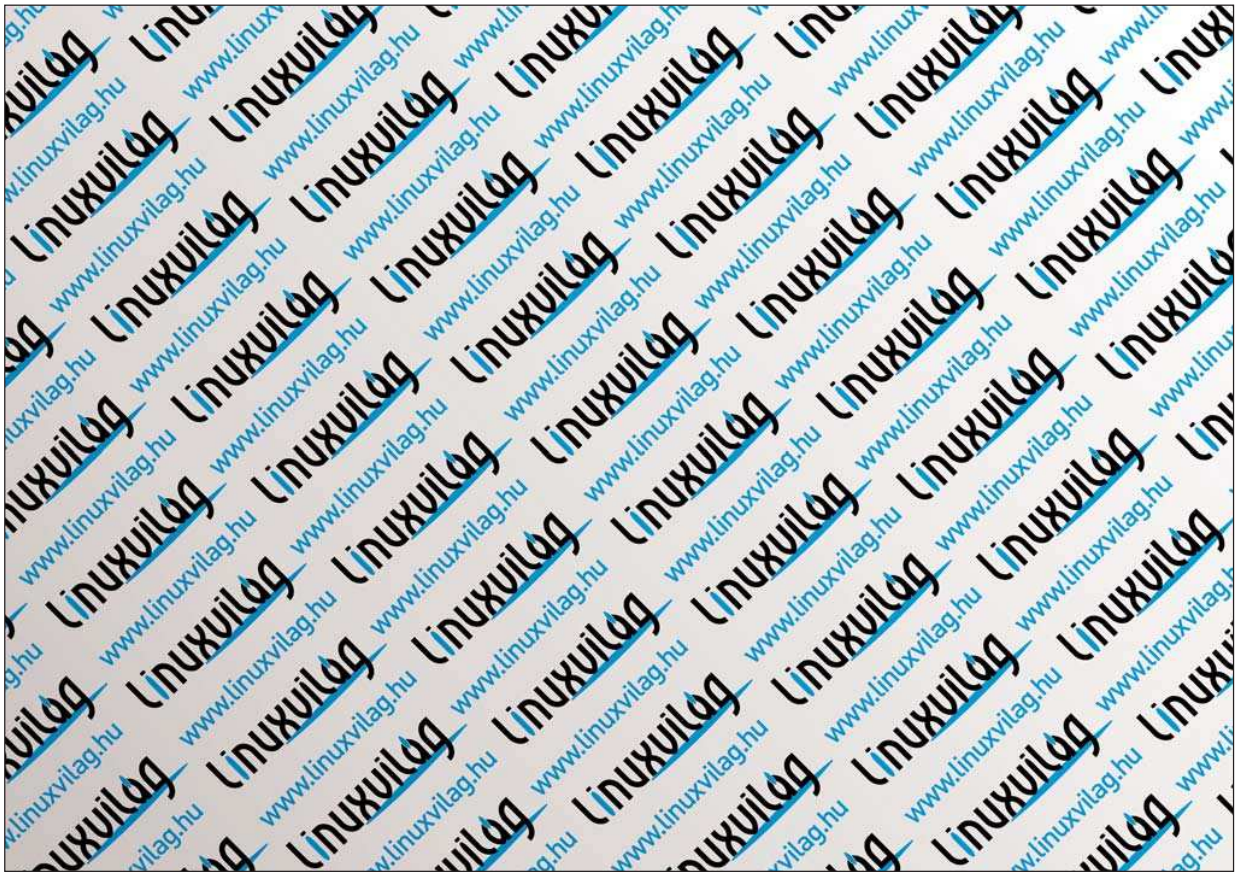
Köszönetnyilvánítás

Értékes visszajelzéseiért köszönettel tartozom ***Peter Andersonnak***, ***Brantley Coile***-nak és ***Al Dixonnak***. Szintén szeretném megköszönni ***Brantley***-nek és ***Sam Hopkinsnak***, hogy ilyen kiváló tárolóprotokollt fejlesztettek ki.

Linux Journal 2005. június, 134. szám

A cikkhez tartozó források elérhetősége:
➲ ***www.linuxjournal.com/article/8201***

Ed L. Cashin 1997 óta különféle tudományos és linuxos szakmai területekkel foglalkozott, volt már webalkalmazás-fejlesztő, rendszergazda és rendszermag-programozó egyaránt. Jelenleg a Coraid munkatársa, itt történt az ATA over Ethernet kifejlesztése. Az ecashin@coraid.com címen érhető el. Miközben harcművészeti óráira tart, legszívesebben zenét és hangos könyveket hallgat.



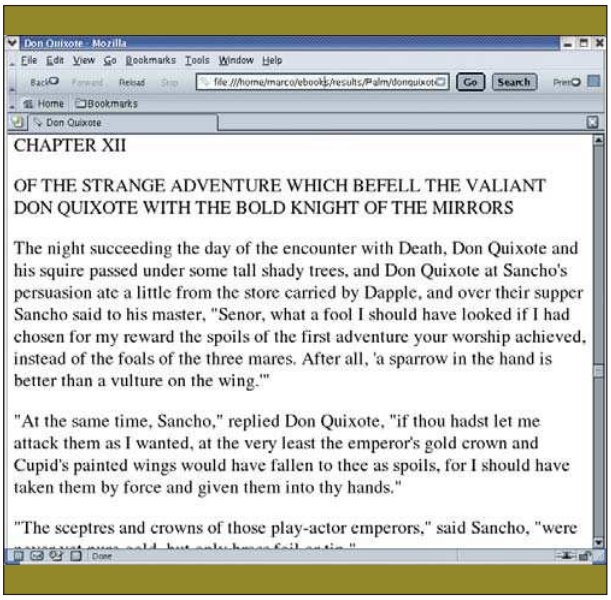
e-könyvek átalakítása nyílt formátumokra

Az e-könyvek a gyártófüggő formátumok elképesztő kavalkádját hozták magukkal. Alakítsuk hát HTML formátumúra őket, így bármilyen eszközön meg tudjuk jeleníteni tartalmukat!

A digitális formátumú könyvek, e-könyve (elektronikus könyvek, e-book) olyan készülékeken jeleníthetők meg, amelyek általában túl kevés erőforrással rendelkeznek egy normál webböngésző futtatásához. Sok kiadó, a ***Project Gutenberghez*** hasonló tervezeteket már nem is említve, több ezer új és klasszikus kiadványt jelentetett meg digitális formában. Baj egyrészt a hardverrel van – legyen szó akár általános célú zsebtitkárról, akár célkészülékről –, másrészt magával az e-könyvek kiadásával foglalkozó ágazattal, amely jóval széttöredezettebb, mint a személyi számítógépek és a webböngészők piaca. Lehet tehát, hogy megveszünk ma egy e-könyvet, és tíz év múlva már nem tudjuk elolvasni – sőt, lehet, hogy már holnap sem, ha például új hordozható gépet vagy zsebtitkárt vásárlunk. Mivel a széttagoltság elleni fellépés sokak érdeke, írásomban néhány olyan parancssori eszközt szeretnék ismertetni, amelyek segítségével a népszerűbb formátumú e-könyveket ***ASCII*** vagy ***HTML*** formátumú anyagokká tudjuk alakítani. Jelenleg gyakorlatilag nem léteznek eszközök az e-könyv formátumok ***PDF*** vagy ***OpenDocument*** formátumba (az ***OpenOffice.org*** által használt új ***OASIS*** szabvány) történő kimentésére; szerencsére ez nem jelenti azt, hogy az átalakítás ne volna megoldható. Ha egyszer a szöveg már ***ASCII*** vagy ***HTML*** formátumot kapott, szövegböngésző – például w3m – vagy egyéb program – mint a ***html2ps*** – segítségével már könnyedén átalakítható egyszerű szöveggé vagy ***PDF*** fájllá. Ha ezt az utat választjuk az átalakításra, akkor azt akár már ma is elvégezhetjük, hiszen utóbbi nyílt formátum, ahogy 20 év múlva is az lesz.

PalmDoc

PalmOS alatt az eredeti és legelterjedtebb e-könyv formátum a ***PalmDoc***, más néven ***AportisDoc*** vagy egyszerűen ***Doc***; nem keverendő össze a ***Microsoft Word*** hasonló nevű, ***.doc*** formátumával. A ***Doc*** formátumú fájlok a ***.pdb (Palm Database, Palm adatbázis)*** vagy a ***.prc (Palm Resource Code, Palm erőforráskód)*** kiterjesztésről ismerhetők fel, mindkettő lényegében összefűzött rekordokat tartalmazó ***PalmPilot*** adatbázist rejt.



1. ábra PalmDoc fájl böngészőben jobban megjeleníthető HTML formátumúra alakítva

A szabványból később több változat is kialakult, többek közt az alapszintű formátumot ***HTML*** címkékkel bővítő ***MobiPocket***.
Minden ***Palm*** e-könyv három részből áll: a fejrészből, a szövegrekordok sorozatából és a könyvjelzők sorozatából. Alapesetben a fejrész 16 bájtos, bizonyos ***Doc***-olvasók ezt futási időben, egyedi adatok tárolása céljából kibővítik. Alapesetben a fejrész adja meg többek közt a tömörítetlen szöveg teljes hosszát, a pillanatnyilag megjelenített rész pozícióját, illetve tartalmaz egy két bájtos, előjel nélküli egész számokból álló tömböt, mely az egyes szövegrekordok tömörítetlen méretét jelzi. A rekordok maximális mérete jellemzően 4096 bájt, és a rekordok tömörítése egyenként történik.
A könyvjelző rekordok egy 16 bájtos névből és egy 4 bájtos, a szöveg kezdetétől számított eltolásból állnak. Mivel a könyvjelzők elhagyhatók, sok ***Doc*** e-könyv nem is tartalmazza őket, a ***Doc***-olvasók pedig sokszor

1. kódrészlet Egyszerű Perl parancsfájl a Pyrite által kimentett szöveg HTML-lé alakítására

```
#! /usr/bin/perl
undef $/;
$TEXT  = <>;
$TEXT =~ s/\n\n/<p>/gm;
print <<END_HTML;
<html><body>
$TEXT
</body></html>
END_HTML
```

másféle, vagyis nem hordozható módszereket is támogatnak megadásukra. Az egyes olvasóprogramok saját kiterjesztései kategória, változatszám, illetve e-könyvek közötti hivatkozások megadására is alkalmasak lehetnek. Utóbbi adatok szinte mindig a *.pdb* vagy *.rc* fájlon kívül tárolódnak, vagyis ne várjuk megőrzésüket, amikor e-könyveinket átalakítjuk.

A ***Pyrite Publisher***, korábbi nevén ***Doc Toolkit*** egy tartalomátalakító eszközöket magába foglaló készlet ***Palm*** rendszerekhez. Jelenleg csak néhány szövegformátum átalakítására van lehetőség, ám szolgáltatásait ***Python*** beépülő modulok segítségével bővíteni is lehet. A ***Pyrite Publisher*** alkalmas az átalakítandó dokumentumok közvetlenül, webről történő letöltésére és a letöltött könyvjelzők közvetlenül a kimeneti adatbázisba való beillesztésére is. A csomag használatához 2.1-es vagy újabb ***Python*** szükséges, futtatása pedig parancssorból vagy a ***wxWindows*** alapú grafikus felülettel történik. A program ***Linux*** és ***Windows*** alá érhető el, forrás és bináris fájl formájában egyaránt. Ha az utóbbit választjuk, akkor ne feledjük el, hogy a legtöbb előre lefordított program a */usr* könyvtárban keresi a ***Pythont***. A linuxos változat az átalakított fájlokat a ***JPilot*** vagy a ***pilot-link*** segítségével azonnal át tudja másolni zsebtitkárra is.

A ***Pyrite*** telepítése és futtatása ***Fedora Core 2*** alatt problémátlan volt. Az egyéb, később említendő parancssori átalakítókkal ellentétben a ***Pyrite*** csak ***ASCII*** formátumba tud menteni, ***HTML***-be nem. A futtatható fájl neve ***pyrpub***. A ***.pdb*** fájlok átalakításához a következő parancsformátumot kell használni:

```
pyrpub -P TextOutput -o don_quixote.txt \
Don_Quixote.pdb
```

Ha csak gyorsan tárgymutatót akarunk készíteni egy digitális könyvtárról, akkor a ***Pyrite***-on kívül másra nem is lesz szükségünk. Ugyanakkor a kimenetet nem nehéz tovább formázni, amivel böngészőben is könnyebben olvashatóvá válik. Az alábbi, Perlben készült kódrészlet ugyan csúnya, ám tökéletesen megfelel a ***Don Quixote*** HTML változatának előállítására. (*1. kódrészlet*)

A parancsfájl betölti a teljes korábban a ***Publisher*** segítségével létrehozott ***ASCII*** szöveget, és minden alkalommal, amikor új sor karaktert talál, a helyére a bekezdés ***HTML*** címkéjét illeszti. Az alapszintű ***HTML*** formázású eredményt a szabványos kimenetre írja ki. Ha meg akarjuk változtatni az igazítást, a betűtípust vagy a színeket, akkor csak annyit kell tennünk, hogy a megfelelő stíluslapot beillesztjük a `<html><body>` sor mögé.

A várhatóan 2005 tavaszán megjelenő ***OpenOffice.org 2.0*** ***.pdb*** formátumban is képes lesz menteni a szövegeket. Ha olvasni is képes lesz az ilyen fájlokat, akkor tömeges átalakítási szolgáltatásával (***File›AutoPilot›Document Converter***) az egész átalakítási kérdés egy csapásra elegáns megoldást nyer. Kipróbáltam a szolgáltatást az 1.9.m65-ös előzetes kiadásban, de egyelőre csak egy általános ki/beviteli hibára utaló üzenetet sikerült kapnom. Remélhetőleg a későbbi változatokban már tökéletesen fog működni ez a szolgáltatás is.

A P5 Perl csomag

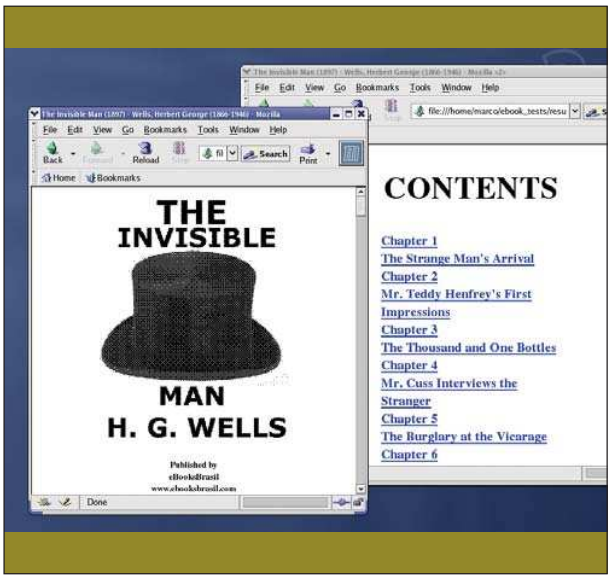
A ***Pyrite Publishert*** elsősorban normál ***HTML*** vagy szöveges fájlok ***Palm*** gépeken olvasható formátumra alakítására készítették, a több lehetőség inkább mellékhatásként fogható fel. A fent ismertetett eljárás csak nehézkesen alkalmazható, ha például nagy mennyiségű, egyedi ***HTML*** címkéket, hiperhivatkozásokat és metaadatokat tartalmazó palmos e-könyvet kell átalakítani. Ilyenkor a legjobb megoldás egy ***Perl*** parancsfájl alkalmazása, igénybe véve a nyelv szabványos ***XML*** vagy ***HTML*** és ***P5-Palm*** moduljait. A modulokat a ***Comprehensive Perl Archive Network*** (lásd az internetes forrásokat) webhelyről lehet elérni. A ***P5-Palm*** modulkészletben található osztályok segítségével a ***PalmOs*** alapú készülékek által használt ***.pdb*** és ***.prc*** adatbázisfájlok olvasása, feldolgozása és írása egyaránt megoldható.

A Rocket Ebook és a MobiPocket

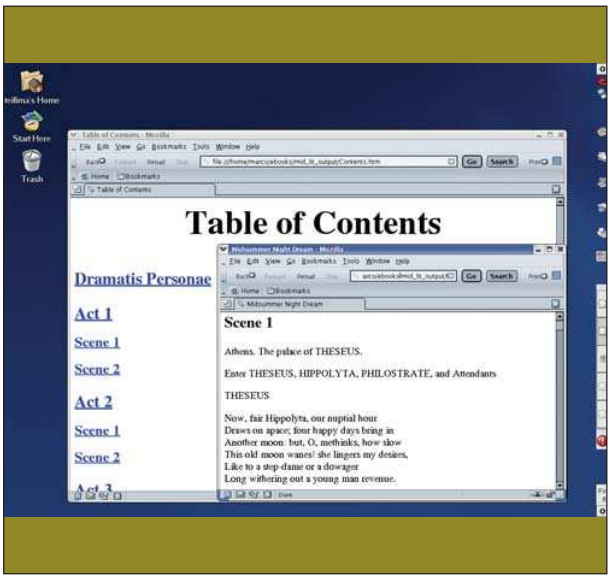
A ***RocketBook*** e-könyvek több érdekes jellegzetességgel is rendelkeznek, például támogatják a tömörített ***HTML*** fájlokat, valamint a bekezdések formázásait és a hivatkozási nevek pozícióit összegző indexeket. Ezekről és az ***.rb*** fájlok pontos felépítéséről az internetes források között található, az RB formátumot ismertető oldalon lehet bővebben olvasni. A ***Rocket Ebook*** és a ***Mobipocket*** fájlokat szétbontani az ***Rbmake*** nevű parancssori eszközkészlettel lehet. A készlet honlapján forráskódot, bináris csomagokat, levelezési listát és hibajelentések elküldésére alkalmas címet egyaránt találunk. Az `rbmake` használatához szükség van a ***libxml2 2.3.1***-es vagy újabb változatára, a ***pcre*** (***Perl-Compatible Regular Expressions***) könyvtárra, illetve a tömörítések kezelése miatt a ***zlib***-re. A forrásból történő fordítás elvégzéséhez – legalábbis ***Fedora Core 2*** alatt – külön telepíteni kell a ***pcre-devel*** csomagot is.

Az Rbmake könyvtár

Az ***Rbmake*** egyik pozitívuma modulárisan összeállított forráskódja. Ha úgy akarjuk, egy teljes objektumorientált



2. ábra Az Rbmake a RocketBook fájlok összes összetevőjét kibontja, ide értve a szöveget és a képeket is



2. ábra A Convert Lit jól olvasható HTML fájlt állít elő, hiperhivatkozásokból álló tartalomjegyzékkel

2. kódrészlet Az OPF egy XML alapú formátum könyvek jellemzőinek leírására

```
<dc:Title>A Midsummer-Night's Dream</dc:Title>
<dc:Creator role="aut"
    file-as="Shakespeare, William, 1564-1616">
        William Shakespeare, 1564-1616
</dc:Creator>
<dc:Description>fiction, poetry</dc:Description>
```

C könyvtárat le tudunk fordítani, függetlenül a csomag egyéb részeitől, így bármilyen más programból is kezelni tudjuk az .rb fájlokat. Ezzel a módszerrel saját, akár minden részletében testreszabott ***Rocket Ebook*** átalakítót is tudunk írni, esetleg az összes e-könyvünkről jegyzéket készíthetünk egy adatbázisba; minden esetben csak arra a kódrészre van szükségünk, amely az ***.rb*** formátum tényleges írásáért és olvasásáért felelős, vagyis az ***RbFile*** osztályra. Ez a kódrész megnyitja a fájlt, visszaadja a könyvet alkotó szakaszok listáját, majd futás közben csak a főprogram által ténylegesen igényelt részeket bontja ki. Ha éppen arra van szükségünk, a könyvtár keresési és cserélési eljárásokkal is rendelkezik, ezeket a ***Perl***-ben megszokott reguláris kifejezésekkel tudjuk használni.

Az Rbmake eszközöknek minden korszerű GNU/Linux terjesztésen gyorsan és gondok nélkül le kell fordulniuk. A forrás .tar állományban részletes HTML leírást is találunk. A HTML fájlok előállítására alkalmas bináris fájl az rbburst. A program az eredeti .rb konténerben lévő összetevők – szöveg, képek és leíró adatok – mindegyikét kibontja. A 2. ábrán két Mozilla ablakban látható, hogy ***H. G. Wells The Invisible Man*** című könyvén az `rbburst`-öt futtatva milyen fedlapot és tartalomjegyzéket kapunk.

Microsoft Reader

A ***Microsoft Reader*** fájlokat a ***.lit*** kiterjesztésről ismerhetjük fel. Számos a hagyományos könyvekre jellemző tulajdonságuk van, mint például oldalszámok, kiemelések és jegyzetek. A kulcsszó alapú keresést és a hiperhivatkozásokat is támogatják, ám egyetlen olvasóprogramhoz kötődnek.

Az ilyen fájlok átalakítására szolgáló eszköz neve spártai egyszerűséggel ***Convert Lit***. A programot a `-help` kapcsolóval futtatva – hűen a unixos hagyományokhoz – az összes parancssori kapcsolót megtekinthetjük.

A program háromféle üzemmódot ismer: ***robbantás (explosion)***, ***lefelé átalakítás (downconversion)*** és ***dedikálás/beleírás (inscribing)***. Meglévő ***.lit*** fájlt ***OEBPS***-megfelelő csomagba a robbantás móddal tudunk átalakítani. Az ***OEBPS***-ről (***Open eBook Publication Structure***) később még lesz szó.

A *3. ábra* ***Shakespeare A Midsummer's Night Dream*** című művének a ***Convert Lit*** programmal robbantott változatát szemlélteti. A lefelé átalakítás ezzel ellentétes eljárás, általa ***Microsoft Reader***-megfelelő eszközökön használható ***.lit*** fájlt kapunk. A beírásnál a lefelé átalakítás közben felhasználó által megadott feliratot csatolunk a ***.lit*** fájlhoz. A pontos szintaxist illetően a program honlapján találunk részletes tájékoztatást (lásd a forrásokat).

Már volt róla szó, hogy a ***Convert Lit*** egy különböző fájlokból álló ***OEBPS*** csomagot állít elő. A fenti példánál a teljes fájllista a következő: ***Contents.htm, copyright.html, ~cov0024.htm, cover.jpg, MidSummerNightDream.opf, MobMids.html, PCcover.jpg,***

PCthumb.jpg, *stylesheet.css* és *thumb.jpg*. A *HTML*, a *CSS* és a *JPG* fájlok különösebb meglepetést nem jelentenek, de vajon mi célt szolgál az *.opf* fájl? Ez egy *XML* konténer, ez írja le az eredeti könyvben található metaadatok egyes részeinek szerkezetét. Az *OPF* kiterjesztés az *electronic book package formatra*, az elektronikus könyv csomag formátumra utal. Az *OPF* az e-könyv egyéb részeire mutató hivatkozásokat is tartalmaz, illetve magába foglalja jellemzőik leírását is. Szerepét könnyebben átláthatjuk egy példán keresztül, ezért a 2. *kódrészletbe* bemásoltam a *MidSummerNightDream.opf* egy kisebb részét.

A gyakorlatban mindebből az fakad, hogy a *Convert Lit* akkor is jól használható lehet, ha teljes gyűjteményünket zárt, gyártófüggő formátumban akarjuk hagyni. Ekkor elég futtatnunk a programot az összes *.lit* formátumú e-könyvünkön, majd az *.opf* fájlok kivételével törölnünk mindent. Ezt követően egy rövidke parancsfájllal vagy akár egy nagyobb tudású *XML* feldolgozó programmal végig tudunk menni ezeken, és az általunk kívánt adatbázisba tudjuk másolni a tárgymutatót.

A *Convert Lit* a *digitális jogkezelő (digital rights management, DRM)* részeket is eltávolítja az e-könyvekből, legalábbis ami a régebbi, *DRM1* változatot illeti. Ha *Microsoft Reader* e-könyveket gyűjtünk, valószínűleg *Microsoft Windows* operációs rendszerrel és a *Microsoft Reader* egy jogtiszta példányával is rendelkezünk. A *Convert Lit* weboldala szerint a *Convert Litet Windows* alatt lefordítva, a *Windows DRM* segítségével az újabb *DRM5*-öt alkalmazó e-könyveket is át tudjuk alakítani *DRM1*-essé.

Tömeges átalakítás

Eddig inkább parancssori átalakításokról volt szó. Aki viszont nagyobb, többféle formátumból összeálló e-könyvgyűjteménnyel rendelkezik, az egyetlen héjparancsfájllal egyszerre is elvégezheti ezek átalakítását. Mint már láttuk, ha a szöveg átkerül *ASCII* vagy *HTML* formátumba, a határ a csillagos ég. A ciklust egy-két sorral bővítve a tárgymutató elkészítése a `glimpse` vagy a `ht::dig` használatával is megoldható; akár mindent beírhatunk egyetlen *PostScript* könyvbe és így tovább.

OEBPS

Jelenleg még fejlesztés alatt álló megoldás e-könyvek – legalábbis a közeljövőben beszerezhető példányok – nyílt formátumba alakítására. Pontos neve *Open eBook Publication Structure (nyílt e-könyv kiadási szerkezet, OEBPS)*. Célja egy *XML*-alapú, a meglévő nyílt szabványokra épülő szabálygyűjtemény összeállítása többféle e-könyv rendszeren történő tartalomszolgáltatáshoz. Az *OEBPS*-t, mely immár 1.2-es változatban érhető el, az *Open eBook Forum* tartja karban. A csoportban több mint 85 szervezet található meg, hardver- és szoftvergyártók, kiadók, szerzők és az elektronikus kiadványok piacán érdekelt felhasználók egyaránt. Az *OEBPS* önmagában, közvetlenül semmilyen digitális jogkezelést nem szolgál. Az *OeBF Rights and Rules Working Group (jogok és szabályok munkacsoport)* azonban szorgalmasan tanulmányozza a vonatkozó kérdéseket, munkájának célja „egységes, kölcsönösen támogatott előírásgyűjtemény összeállítása az elektronikus kiadói közösség számára". Még meglátjuk, mi sül ki belőle.

Bármi is történjék, az *OEBPS* hátteréként szolgáló nyílt szabványok szilárd alapokra épülnek. Az *XML*, az *Unicode* és az *XHTML* mellett a *CSS1* és a *CSS2* bizonyos részei is szerepelnek benne. Az *Unicode* kódolások családja, alkalmazásával több tízezer karaktert is félreértések és keveredések nélkül lehet kezelni. Az *XHTML* a *HTML 4* új, *XML* alapú megtestesülése. Az *OEBPS* röviden talán úgy jellemezhető, mint az *XHTML* e-könyvek kezelésére testreszabott változata – olyasvalami, ami akkor is fennmarad, ha a támogatását adó vállalatok némelyike esetleg el is tűnik az üzleti élet porondjáról. A képek *PNG* vagy *JPEG* formátumúak lehetnek. A metaadatok (szerző, cím, *ISBN* stb.) kezelése a *Dublin Core* szótáron keresztül fog történni.

Az *OEBPS* tehát alkalmas arra, hogy az összes e-könyvünket megőrizzük, és biztosítsuk időtállóságukat, még akkor is, ha némelyik hardver- vagy szoftvergyártó neve néhány év múlva homályos emlékké válik. Sajnos az ilyen „nyílt" e-könyvekre alkalmazott *DRM* megoldások egy-egy gyártóhoz köthetnek bennünket. Amíg az *OEBPS* e-könyveket *DRM* nélkül tudjuk beszerezni, addig az *OEBPS* a legjobb megoldás annak biztosítására, hogy gyűjteményünk még akkor is használható maradjon, ha az ezen a piacon érdekelt cégek akár mindegyike a süllyesztőbe kerül.

Linux Journal 2005. június, 134. szám

Marco Fioretti hardver-rendszermérnök, a szabad szoftverekkel mint EDA alaprendszerekkel és a hatékony asztali környezet létrehozására irányuló RULE Project vezetőjeként áll kapcsolatban. Marco Olaszországban, Rómában él a családjával.

KAPCSOLÓDÓ CÍMEK

- www.cpan.org
- www.linuxmafia.com/pub/palmos/development/doc-format and www.timwentford.uklinux.net
- www.blorf.net/~wayne/rb_format.html
- www.pyrite.org/doc_format.html
- www.pyrite.org/publisher
- freshmeat.net/projects/rbmake
- www.kyz.uklinux.net/convlit.php
- www.openebook.org/oebps/oebps1.2/index.htm
- userpage.fu-berlin.de/~mbayer/tools/html2text.html
- user.it.uu.se/~jan/html2ps.html

Készítsünk Impress és PowerPoint bemutatót LaTeX és Perl segítségével

Aki valamiért kénytelen védett formátumokat használni, a nyílt forráskód annak is megkönnyítheti a dolgát.

Kezdjük az elején.
Második könyvem, melyet ***Dr. Michael Moorhouse***-al közösen készítettünk, végre elkészült. Hat hónap plusz időt töltöttem vele, ami egyben azt is jelentette, hogy hat hónap késésben voltam. Az összes időt szedéssel, átolvasással, írással és ***Michael Microsoft Word*** állományainak ***LaTeX*** formátumúra alakításával töltöttem, elolvasva majd újra elolvasva mindent. Végül mindet még egyszer átnéztem. Amikor végre minden elkészült, teljesen kimerültem. Nem sokkal ez után megkaptam a borító végleges mintáját. És akkor kiderült, hogy a hátlapon azt ígérjük, hogy a weblapon a szöveggel együtt használható ***Microsoft PowerPoint*** bemutatók jelennek majd meg. A borító megváltoztatásához már túlságosan késő volt, következésképpen valahogy meg kellett csinálni a bemutatókat. Úgy tűnik elfelejtettem, hogy ebben egyeztünk meg a projekt indításakor, több mint 18 hónappal korábban.

A PowerPoint „szabvány"

Nyolc hónappal ezelőtt, akadémia berkekben a bemutatók de facto szabvány formátuma a ***PowerPoint*** volt. Manapság már a ***PDF*** is népszerű. Mint oly sokan a Linux közösségben, már régebben átálltam az ***OpenOffice.org***-ra és elhagytam a ***PowerPoint*** környeztet. A könyvben 20 fejezet szerepelt, úgy becsültem, hogy körülbelül 20 napba telik elkészíteni a diákat. Ráadásul mindezt a ***PowerPointtal*** megcsinálni nem igazán volt ínyemre. Természetesen dolgozhattam volna az ***OpenOffice.org Impress***-ében, majd azt exportálhattam volna ***PowerPointba***, de ez az ötlet sem tetszett különösebben. Az alapvető probléma az volt, hogy jól tudtam, minden adat megvolt ***LaTeX*** állományokban, és ha arra gondoltam, hogy ezeket mind újra létre kell hozni egy fóliakészítő alkalmazásban csak még kimerültebbnek éreztem magam. Ha valahogy ki tudnám találni, hogyan lehet programozottan kiszedni az adatokat a ***LaTeX*** állományaimból és beletenni a ***PowerPoint*** diákba, az jelentősen leegyszerűsítené a dolgokat.

Munka a bemutató fájlformátumokkal

A ***Google*** keresés nem hozott eredményt. Talán nem is meglepő, hogy a ***PowerPoint*** fájlformátum részleteihez igen nehéz ráakadni. Találtam egy ***Microsoft Windows Help*** formátumú állományt amely a ***Microsoft Office*** dokumentumok ***XML*** szabványát ismertette. Ebbe a formátumba lehet a ***PowerPoint*** dokumentumokat exportálni. Sajnos igen méretes és bonyolult írás volt. Miután láttam, hogy a ***Google***-lel nem sokra megyek, átnyergeltem a ***Comprehensive Perl Archive Network (CPAN)*** hálózatára. A ***Perl***, a választott programozási nyelvem, szinte minden fájlformátumra és számítási platformra felkészült. Ha valaki korábban már játszadozott kicsit a ***Perl*** és ***PowerPoint*** párossal, munkájára esetleg ráakadhatok a ***CPAN***-on. Sajnos ez a keresés is eredménytelen maradt.
És akkor rájöttem a megoldásra: ha a nyílt és széles körűen dokumentált ***OpenOffice.org Impress*** dokumentumformátummal tudnék dolgozni, utolsó lépésként az ***Impress*** diákat kimenthetném ***PowerPoint*** formátumba.
Az ***OpenOffice.org*** weblap gyors átolvasása után hamarosan meg is találtam a ***OpenOffice.org*** fájlformátumainak ***XML*** leírását. A szabvány a maga 600 oldalával, súlyosabb mint a saját könyvem!
Az ***XML*** dokumentum szépen meg van írva, de egy kicsit nehéz olvasmány. Visszakanyarodtam hát a ***CPAN***-ra, hátha dolgozott már valamelyik programozó az ***OpenOffice.org*** formátumaival, és volt olyan nagylelkű, hogy feltöltötte munkáját a ***CPAN***-ra. Ezúttal nem kellett csalódnom. ***Jean-Marie Gouarne*** a ***Genicorptól*** nemrégiben adta közre az ***OpenOffice::OODoc*** modult, amely a ***Perl*** csatoló felületet biztosít az ***OpenOffice.org*** formátumához. Egy már létező dokumentum esetében az ***OpenOffice::OODoc*** modullal módosíthatjuk a tartalmat, igény szerint hozzáadhatunk, törölhetünk és frissíthetünk a lemezes állományban.

A diakészítő terve

Egy egyszerű ***Perl*** szűrővel kezdtem amely ***LaTeX*** állományokat fogad bemenetként és dia adatokat készít kimenetként saját szöveges formátumban. A szöveges állomány létrehozásával biztosítani tudtam, hogy bármilyen szövegszerkesztővel könnyedén módosítani tudom a szűrő kimenetét, illetve szükség esetén finomhangolhatom a szöveges kimenetet. Amikor elértem a kívánt eredményt, egy

másik, szintén *Perlben* készült szűrő a szöveges állományból elkészíti az *Impress* bemutatót. Az *Impress* bemutatót aztán meg lehet nyitni *Impress*-ben és exportálni *PowerPoint* vagy *PDF* formátumban.

Dia tervek

A bemutatómat tudatosan próbáltam a lehető legegyszerűbb alakban elkészíteni, ezért úgy döntöttem csak három diatípust használok. A cím dia tartalmazza a fejezet címét a bemutató állomány elején. A bemutatóban a `title_slide` (cím dia) kettős szerepet tölt majd be, ez tartalmazza majd a bemutatóba illesztett képeket is, azaz képenként külön `title_slide`-ot készítünk. A `bullet_slide` (pont dia) tartalmazza a szakaszok címeit a dia fejlécében valamint az alcímeket a pontokban. Végül, a `sourcecode_slide` típusú, rögzített karakterhosszúságú, betű szerinti diákat használom fel a programforrások megjelenítéséhez.
A három diás bemutatót kézzel készítettem el *Impress*-ben és *blank.sxi*-nek néven mentettem el. Az elkészített három dia az előző bekezdésben bemutatott három diatípusnak felelt meg. Úgy terveztem, hogy a fejezetek programozott bemutatókészítése során ezeket fogom majd másolni. A másolás segítségével biztosíthatom a bemutatóm egységes külalakját.

A szöveges tartalom kigyűjtését végző szűrő

A `getcontent` olyan stílusú héjprogram, amilyeneket igen gyakran készítenek a *Perl* programozók, felhasználják, majd eldobják őket. (A hálózati források közt megtaláljuk a cikkben említett programok forrását) Soronként olvasva végiglépked a szabványos bemeneten, és mintakereséssel próbálja meg felderíteni a lényeges sorokat. Ha a sor megfelel a keresésnek, elkészíti a megfelelő kimenetet.
A `getcontent` működésének bemutatására álljon itt példaként a *LaTeX* állomány fejezetcímeit kezelő kód:

```
if ( /\\chapter\{(.*)\}/ )
{
    print "CHAPTERTITLE: $1\n";
    next;
}
```

A *LaTeX* fejezet makróját egyszerű szabályos kifejezés keresi ki; Ha volt egyezés, a fejezet címét kigyűjtjük és elkészítjük a kimenetet. A `next` hívás rövidre zárja a kört, így ha találtunk valamit, a szabványos bemenetről a következő sort olvassuk be. Ezáltal a következő *LaTeX* részlet:

```
\chapter{Working with Regular Expressions}
```

a következő szöveges tartalomra cserélődik le:

```
CHAPTERTITLE: Working with Regular Expressions
```

Azaz a *LaTeX* jelrendszerét eltávolítottuk, és egy sokkal egyszerűbb jelöléssel cseréljük le. A alfejezet és szakaszcímeket jelölő *LaTeX* formátumot ugyanilyen módszerrel kezeljük. A kód a következőképpen néz ki:

```
if ( /\\section\{(.*)\}/ )
{
    print "BULLETTITLE: $1\n";
    next;
}
if ( /\\subsection\{(.*)\}/ )
{
    print "BULLETCONTENT: $1\n";
    next;
}
```

A forráskódlistákkal sem sokkal nehezebb boldogulni, igaz itt figyelnünk kell rá, hol kezdődik és végződik a nyers szöveg bevitele. A következő kódrészlet kezeli a *LaTeX „verbatim"* blokkját:

```
if ( /\\begin\{verbatim\}/ )
{
    print "STARTCODE\n";
    $in_verbatim = TRUE;
    next;
}
```

A `verbatim` blokkból kilépést pedig a következő kóddal keressük:

```
if ( $in_verbatim )
{
    if ( /\\end\{verbatim\}/ )
    {
        print "STOPCODE\n";
        $in_verbatim = FALSE;
    }
    else
    {
        print;
    }
    next;
}
```

A `$in_verbatim` nevű egyszerű `bool` típusú változó segítségével vizsgáljuk, hogy a parancsfájlunk még a `verbatim` blokk belsejében van-e. Hasonló kód kezeli a könyvben megjelenő tételeket a képeket, azok feliratai és egyéb érdekes dolgokat pedig néhány feltétles blokk kezeli. vegyük például a következő *LaTeX* leírást:

```
\chapter{The Basics}
\textit{Getting started with Perl.}
\section{Let's Get Started!}
There is no substitute for practical experience
➥ when first
learning how to program. So, here is the first
➥ Perl program
\index{welcome@\texttt{welcome}, and the first
➥ program, called
\texttt{welcome}:
\begin{verbatim}
    print "Welcome to the World of Perl!\n";
\end{verbatim}
\noindent When executed by \texttt{perl}
\footnote{We will learn how to do this is in
```

```
just a moment.}, this small program displays
the following, perhaps rather not unexpected,
message on screen:
\begin{verbatim}
    Welcome to the World of Perl!
\end{verbatim}
```

A `getcontent` parancsfájlja a fenti ***LaTeX*** állományból a következő szöveges állományt készíti:

```
CHAPTERTITLE: The Basics
CHAPTERCONTENT: Getting started with Perl.
BULLETTITLE: Let's Get Started!
STARTCODE
    print "Welcome to the World of Perl!\n";
STOPCODE
STARTCODE
    Welcome to the World of Perl!
STOPCODE
```

Figyeljük meg hogy valamennyi ***LaTeX*** jelölés eltűnt, és helyére egy egyszerű jelölésrendszer került, amelyet majd a diák létrehozására használhatunk fel. Feltételezve, hogy a ***LaTeX*** részlet a `chapter3.tex` nevű állományban kapott helyet, a `getcontent` parancsfájlt elindítva, az átalakítás eredményét átirányítjuk a megfelelően elnevezett célállományba:

```
perl getcontent chapter3.tex > chapter3.input
```

A `chapter3.input` állományba így a szöveges tartalom kerül, amit aztán tetszőleges szövegszerkesztővel finomhangolhatunk a diák létrehozása előtt.

Az Impress bemutató készítő szűrő

A diák létrehozását az ***Impress*** dokumentumban több tényező is nehezítette. Először is az ***OpenOffice::OODoc*** modullal nem hozhatunk létre új ***OpenOffice.org*** állományt, az ugyanis csak a már létező állományokat tudja módosítani. Továbbá a modult elsősorban az ***OpenOffice.org Writer*** állományok (tehát szövegszerkesztő fájlok és nem ***Impress*** bemutatók) szerkesztésére készítették. Példaként álljon itt egy rövid, `appendpara` nevű program, amely rövid szöveget fűz a ***Writer*** dokumentumunkhoz:

```
#! /usr/bin/perl -w
use strict;
use OpenOffice::OODoc;
my $document = ooDocument( file => 'blank.sxw' );
$document->appendParagraph
(
    text    => 'Some new text',
    style   => 'Text body'
);
$document->save;
```

A rövid program az `OpenOffice::OODoc` modul segítségével készít dokumentum objektumot egy már meglévő ***Writer*** állományból. A program ezután az `appendParagraph` metódus meghívásával beszúr némi szöveget majd meghívja a `save` metódust és a lemezre menti a változtatásokat.
Az `appendParagraph` metóduson felül az `OpenOffice::OODoc` modulban találunk egy `insertElement` nevű metódust is, amellyel egy adott típusú lapot szúrhatunk a dokumentumba. A lap egy már meglévő lap másolata vagy tényleges nyers ***XML*** lehet.
Alig jutottam el a 6. oldalig a több mint 600 oldalas ***OpenOffice.org XML*** fájlformátum leírásban, máris megtaláltam, hogy az ***Impress*** a `//draw:page` ***XML*** típust használja a bemutatók diáinak megjelenítésére. Sajnos az `OpenOffice::OODoc` modul ezzel az objektumtípussal közvetlenül nem tudott dolgozni, úgyhogy valamilyen más módszert kellett kieszelnem az adatok kezeléséhez.
Egészen pontosan a ***blank.sxi*** dokumentumban lévő lapokat szerettem volna kiemelni és szükség szerint másolni az oldalait, miközben a dia tartalmát a ***getcontent*** parancsfájl által gyűjtött tartalommal helyettesítem. Ehhez persze egy kicsit jobban meg kellett ismernem az ***Impress XML*** formátumát.
Két választással volt: tovább olvasom a 600+ oldalas szabványdokumentációt, vagy belenézek egy létező állományba hátha eleget megtudok a feladat kivitelezéséhez. Az utóbbit választottam. Emlékezve egy korábbi ***Linux Journal*** cikkre mely szerint az ***OpenOffice.org*** több részből álló állományait a népszerű ***ZIP*** algoritmussal tömöríti, készítettem egy ideiglenes könyvtárat és kitömörítettem a ***blank.sxi*** állományt:

```
mkdir unzipped
cd unzipped
unzip ../blank.sxi
```

Ezáltal kaptam néhány fájlt és könyvtárat:

```
content.xml
META-INF
meta.xml
mimetype
settings.xml
styles.xml
```

a legérdekesebb a ***content.xml*** állomány, hiszen ez tartalmazza a dokumentumot alkotó tényleges szöveget. képernyőn vagy egy szövegszerkesztőben nézve rengeteg nehezen értelmezhető ***XML*** kódot látunk. Azért, hogy a részek lehető legkisebbek legyenek, az ***XML*** formázására egyáltalán nem fordítottak semmilyen figyelmet, a zippelt állomány egyik részében sem. Az ***XML*** általában tagolás és szóközök nélküli szövegfolyamként kerül mentésre. Mielőtt értelmezhettem volna, valamilyen olvasható formában kellett kinyomtatnom ezt az ***XML***-t. Pillanatnyi ihlettől vezérelve, átléptem parancssorba és begépeltem az `xml` szót két tabbal kísérve. A képernyőn megjelentek az `xml`-el kezdődő előre telepített szóközök:

```
xml2-config     xml-config      xmllint
xmlto           xml2man         xml-i18n-toolize
xmlproc_parse   xmlwf           xml2pot
xmlif           xmlproc_val     xmlcatalog
xmlizer         xmltex
```

Az `xmllint` egyből felkeltette az érdeklődésemet. A kézikönyv oldalán hamar rábukkantam a `--format` kapcsolóra, amely – ahogy azt biztosan kitalálták – formázza az eszköznek átadott *XML* adatokat. Következésképpen az `xmllint --format content.xml` paranccsal olyan kimenetet kaptam amit aztán a less-be csövezve már nehézségek nélkül el lehetett olvasni. Alább bemutatjuk a *content.xml* olvasható formában kinyomtatott rövidített részletét, ahol a *blank.sxi Impress* dokumentum `title_slide` *XML* részletét láthatjuk:

```
<draw:page draw:name="page1" draw:style- ...
  <draw:text-box presentation:style-name= ...
    <text:p text:style-name="P1">
      <text:span text:style-name="T1">
        ChapterTitleSlide
      </text:span>
    </text:p>
  </draw:text-box>
  <draw:text-box presentation:style-name= ...
    <text:p text:style-name="P3">
      <text:span text:style-name="T2">
        ChapterTitleSlideText
      </text:span>
    </text:p>
  </draw:text-box>
  <presentation:notes>
    <draw:page-thumbnail draw:style-name= ...
    <draw:text-box presentation:style-name ...
  </presentation:notes>
</draw:page>
```

Figyeljük meg a `ChapterTitleSlide` és a `ChapterTitleSlideText` tartalmat, amelyeket a *blank.sxi* létrehozásakor gépeltem be az *Impress*-be. Ha az `insertElement` metódus segítségével e részlet alapján be tudnék vinni nyers *XML* adatokat, miközben az üres tartalmat a saját szöveges adataimmal helyettesítem, lényegében már készen is volnék.
A példa kedvéért gondoljuk végig mi is történik, amikor a bemutató címét és alcímét feldolgozza a `produce_slides` rutin. Az `insertElement` a következő formában hívódik meg, létrehozva a az új diát:

```
$presentation->insertElement( '//draw:page',
  $last_slide++,
    title_slide( $title_title, $title_content ),
      position => 'after' );
```

A `title_slide` alprogram nyers *XML* adatot ad vissza, amit a dokumentumba szúrhatunk.
Ha a `getcontent` által készített szöveges tartalommal egyező formátumú bemenetei állományt használunk, a `produce_slides` parancsfájl lemásolja a *blank.sxi Impress* állományt majd tetszőleges számú diával tölti fel, így programozottan hozza létre a bemutatót. A parancsfájl szerkezetében nem különbözik sokban a `getcontent`-től mindössze annyi a különlegessége, hogy a *blank.sxi* állományban tárolt, három különféle diatípushoz tartozó nyers *XML* adatot átemeli. Az előadás elkészítéséhez a következő alakban hívjuk meg a `produce_slides` programot:

```
perl produce_slides 3 chapter3.input
```

Eredményképpen egy új *Impress* dokumentumot kapunk, amely *chapter3.sxi* néven jelenik meg a merevlemezünkön.
Miután létrehoztam az *Impress* állományt, a grafikus képek helyfoglaló elemeit le kellett cserélnem a tényleges képekre. A `getcontent` parancsfájl azonban csak a képek neveit bontotta ki, és nem magukat a képeket. A képek importálása *Impress* alá nem lenne túl nehéz feladat, eltekintve attól, hogy az általam birtokolt eredeti állományok meglehetősen gyenge minőségűek voltak azokhoz képest amelyek végül a könyvbe kerültek.
A könyvbe kerülő képeket jelentősen feljavította a kiadó végső szedőfázisa. Nekem pedig természetesen nem voltak meg ezek az állományok.
Aztán eszembe jutott, hogy a kiadó elküldte nekem a végső *PDF* változatot, amelyben ott volt valamennyi jó minőségű kép. Az *xpdf* segítségével 200%-ra nagyítottam a képeket, majd beindítva a *The GIMP*-et leolvastam az *xpdf* megjelenítő ablakát. Kivágtam a grafikus képeket és elmentettem *JPEG* formátumban. Igaz beletelt egy kis időbe, amikor azonban elkészült, gyönyörű, könyv-minőségű képekhez jutottam, amelyeket már be lehetett importálni az *Impress* bemutatóba. Miután ezzel elkészültem, elmentettem az *Impress* dokumentumot *PowerPoint* formátumban és a feladat meg volt oldva. A kezdeti 20 napos becslésem körülbelül 20 órás tényleges munkára csökkent.
Ráadásul most már, ha gyorsan kell diákat összeraknom, a szöveges tartalmat akár kézzel, *vi* segítségével is összerakhatom, majd a `produce_slides` parancsfájl lefuttatása után már készen is vagyok.

Zárszó

Ami először lehetetlen feladatnak tűnt – programozottan létrehozni egy *PowerPoint* bemutatót – kiderült, hogy a nyílt forráskódnak köszönhetően nagyon is lehetséges. A megszokott *Red Hat 9* terjesztésemben valamennyi szükséges eszköz rendelkezésemre állt: `vi`, `unzip`, *Perl*, `xmllint`, `xpdf`, *The GIMP* és az *OpenOffice.org* csomag.

Linux Journal 2005. április, 132. szám

Paul Barry (paul.barry@itcarlow.ie)
A Carlow-i Műszaki Intézetben tart előadásokat, Írországban. Weblapján előadásaival, könyveivel és cikkeivel kapcsolatos információkat találunk: glasnost.itcarlow.ie/~barryp.

KAPCSOLÓDÓ CÍMEK

- ftp.ssc.com/pub/lj/listings/issue132/7972.tgz
- search.cpan.org
- www.openoffice.org

Készítsünk Live CD-t!

Készítsük el saját, különleges live CD terjesztésünket az itt bemutatásra kerülő nem túl ismert rendszerindító CD trükkök alapján.

Biztosan mindenki hallott már a ***Knoppixról***, arról a ***Debian*** alapú terjesztésről, amely 2GB alakalmazást zsúfol egyetlen CD lemezre. Használják ***Linux*** bemutató eszközként, mentőlemezként, sőt még ***Debian*** telepítőként is. Hasonló projektek egész kis seregét ihlette, az egy-két plusz és mínusz csomagot tartalmazó ***Knoppix*** CD-ktől kezdve egészen a rendszer teljes újratervezéséig.
Jómagam mostanában fogtam neki egy termékbemutató CD készítésének. Kíváncsiságtól hajtva darabokra szedtem a ***Knoppix*** CD-t míg végül egy ***Makefile*** és néhány mellékes állományra jutottam, amelyek ugyan elég egyértelműen ***Knoppix*** ihletésűek de azért van bennük némi új kód is. Itt tartok most.

Rövid Körséta

Ha a ***Knoppix*** CD-t betesszük a meghajtóba és felcsatoljuk, hamar feltűnik, hogy nem nagyon hasonlít a hagyományos ***Linux*** telepítésekre. Van ugyan néhány grafikus fájl és néhány zenesáv, de nincs ***init***, nincs ***/dev*** és nincs ***/bin***. A mágia a ***/KNOPPIX/KNOPPIX*** nevű, igen méretes állományban van, amely a ***cloop*** eszközzel tömörített ***ISO9660*** lenyomat.
A rendszermagban található megszokott ***loop*** eszköz lehetővé teszi, hogy néhány fájlrendszeren úgy érjük el az állományokat mintha azok valódi eszközök lennének. Az eszköz blokkjainak elérési kérelmei az alatta elhelyezkedő állomány blokkjaira fordítódnak le. Mivel az eszközt fel tudjuk csatolni a fájlrendszerek másolatait lényegében ugyanúgy érhetjük el mintha valódi lemezes eszközök lennének. Amennyiben a hálózatról töltöttük le a ***Knoppixot***, akkor minden bizonnyal van egy ***ISO9660*** lenyomatunk amit a ***loop*** segítségével fel tudunk csatolni, ha kíváncsiak vagyunk a tartalmára:

```
# mkdir /tmp/knoppix-cd
# mount -o loop -r \
$HOME/KNOPPIX_V3.3-2003-09-24-EN.iso /tmp/knoppix-cd
```

A `cloop` tömörített `loop` eszköz még egy lépéssel tovább megy. Ez a `loop` eszköz úgy modellezi az alkatrészt, hogy minden blokkot a `gzip`-pel tömörít, amit eléréskor átlátszó módon kibont. A ***/KNOPPIX/KNOPPIX*** pontosan ilyen másolat, amelyet rendszerindításkor csatolunk fel – ezzel a trükkel tud a ***Knoppix*** 2GB-ot elhelyezni egy 650MB-os CD-n.
Ha csak egyszerűen körbe szeretnénk nézni a fájlrendszeren, nem muszáj a rendszermagunkba fordítanunk a ***cloop*** képességet. Elegendő a ***cloop-utils*** csomagot feltennünk és az ***extract_compressed_fs*** segédeszközt használnunk. Az állomány elhelyezéséhez körülbelül 2GB szabad helyre lesz szükségünk a ***/var/tmp*** vagy valamely egyéb könyvtárban:

```
# mkdir /tmp/knoppix-cloop
# extract_compressed_fs \
   /tmp/knoppix-cd/KNOPPIX/KNOPPIX \
    >/var/tmp/KNOPPIX-cloop
# mount -o loop /var/tmp/KNOPPIX-cloop \
   /tmp/knoppix-cloop
# find /tmp/knoppix-cloop -print
```

Mindent a szemnek, semmit a kéznek – az ***ISO9660*** fájlrendszer csak olvasható. A terjesztés módosításához először is mindkét fájlrendszer szerkezetet egy hagyományos könyvtárba kell másolnunk:

```
# mkdir $HOME/my-knoppix-tree  \
    $HOME/my-knoppix-cd-tree
# tar -C /tmp/knoppix-cloop -cf - . | \
    tar -C $HOME/my-knoppix-tree -xvpf -
# tar -C /tmp/knoppix-cd -cf - . | \
    tar -C $HOME/my-knoppix-cd-tree -xvpf -
# umount /tmp/knoppix-cd /tmp/knoppix-cloop
```

Most már nekiláthatunk megvalósítani szívünk vágyát. A legkényelmesebb módszer, ha a ***Knoppix*** belső fában a `chroot` paranccsal megváltoztatjuk a gyökeret:

```
# mount -t proc none $HOME/my-knoppix-tree/proc
# cp /etc/resolv.conf \
$HOME/my-knoppix-tree/etc/resolv.conf
# chroot $HOME/my-knoppix-tree /bin/sh
```

Ettől kezdve a szokásos ***Debian*** csomagkezelő parancsok segítségével (`dpkg`, `apt-get` és a többiek) telepíthetünk és törölhetünk tetszés szerint. Miután elkészültünk, lépjünk ki

a chroot-ból és csatoljuk le a *proc*-ot (hacsak nem akarjuk halhatatlanná tenni a fejlesztői rendszerünk folyamatlistáját a CD-n). A create_compressed_tree és mkisofs parancsokkal készítsük el a külső és belső fát:

```
# mkisofs -L -R -l -V "KNOPPIX ISO9660" -v \
-allow-multidot $HOME/my-knoppix-tree | \
create_compressed_fs - 65536 > \
$HOME/my-knoppix-cd/KNOPPIX/KNOPPIX
# mkisofs -l -r -J -V "KNOPPIX with local stuff" \
-hide-rr-moved -v -b KNOPPIX/boot-en.img \
-c KNOPPIX/boot.cat -o knoppix.iso \
$HOME/my-knoppix-cd
```

Végül, írjuk ki a *knoppix.iso*-t egy *CD-ROM*-ra és indíthatjuk a rendszert. Akár írás nélkül is kipróbálhatjuk *Bochs* vagy *VmWare* segítségével.

Következő lépés

Ez az egyszerű megközelítés azonban kezd nehézkessé válni amikor komolyabb vállalkozásba fogunk. Például, ha azt szeretnénk, hogy az X egy adott ablakkezelőt indítson de nem akarjuk, hogy *GNOME* vagy *KDE*-t használjon, kénytelenek leszünk belenyúlni a parancsfájlokba. Ez persze nem nehéz feladat, viszont azt jelenti, hogy lényegében *Knoppix* leágazást készítünk. Amikor egy új *Knoppix* verzió jön ki, mindent újra meg kell csinálnunk. Továbbá, ha üzleti céllal szeretnénk eladni a *Knoppix* alalpú CD-nket, meg kell felelnünk valamennyi terjesztésre szánt program engedélyének, ami feltételezi hogy pontosan tudjuk mi van a lemezen. Az általam vizsgált *Knoppix* verzió tartalmazott néhány olyan állományt amelyek nem Debian csomagokból származtak, sőt, némelyik még csak nem is volt szabad program.
Nem közelíthetnénk meg a problémát más oldalról? Szerencsére megtehetjük. Hála a *Progeny* kezdeményezésnek, amely telepítőjét ajándékozta a *Debian Projektnek*, *Klaus Knoppernek* a *Knoppix* szerzőjének és a *cloop* meghajtó alkotójának, és más Debian fejlesztőknek akik kódját a fő *Debian* tárhelybe felvették, ma már nulláról össze tudunk állítani egy teljes live CD rendszert kizárólag *Debian* csomagok segítségével. A cikk további részében ezzel fogunk foglalkozni.

Letöltések

A cikkünkben szereplő parancsfájlok és állományok tarlabdája a következő címről tölthető le: *ftp.linux.org.uk/~dan/livecd*. Helyszűke miatt cikkünkben a legtöbb kódot nem mutatjuk be. Többnyire *Makefile* által vezérelt, pár héjprogrammal és néhány egyszerű *Perl* kóddal megtűzdelve, úgy gondolom elég jól követhető. Kicsit nehezebb dolga van annak aki nem *Debiant* használ. Ha sikerül más terjesztés alatt is működésre bírni a dolgot, kérem küldjék el a foltot.
A *debootstrap* program az további munkához szükséges *Debian* alaprendszert biztosítja. Ha megadjuk a *Debian* kiadás nevét és a csomagtükör *URL*-t, a *debootstrap* a kiválasztott alkönyvtárba tölti le és telepíti az alaprendszer. Ez igen rugalmas megoldás. Beléphetünk *chroottal*, használhatjuk *UML* gyökérként, avagy, ha a kiválasztott alkönyvtár külön fájlrendszer volt, újraindíthatjuk a gépet és közvetlenül is használhatjuk. Akár CD-re is kiírhatjuk, és pontosan ezt akarjuk majd tenni. De addig még van egy két elvégzendő feladatunk.
A parancsfájljaink próbálgatása folyamán jó sok *debootstrap* és csomagtelepítésre számíthatunk. Mielőtt továbblépnénk, egy kis időt és sávszélességet takaríthatunk meg, ha feltesszünk valamilyen *proxy* csomagarchívumot (például az *apt-proxy*-t) egy kényelmesen elérhető gépre.

Csomagok felvétele

A *Makefile*-ban található fix_inner cél csomagokat ad az alaprendszerhez. Első lépésként lecseréljük a start-stop-daemon programot a */bin/true*-ra, nehogy az *telepítés utáni (post-installation) parancsfájlok* szolgáltatásokat indítsanak el a *chroot* környezetünkben. Ez után többször *chroot*-olunk a rendszerbe és futtatjuk az apt-get és dpkg parancsokat. Tesztelési és kísérletezési célokra a *run-chroot.pl* Perl parancsfájlt használhatjuk, amely rendszerindítást szimulál *chroot* környezetben. A legtöbb szolgáltatást nem indítja el, hiszen azok már elve futnak a gépen és ütközést okozhatnának, de az *SSH* kiszolgálót és az X indítószkripteket lefuttatja. Sokkal kényelmesebb mint mindig kiírni egy CD-t és újraindítani a gépet valahányszor ki akarjuk próbálni.

autologin

Nem sok értelme van bejelentkezésre kényszeríteni az embereket egy egyfelhasználós, bemutatóra szánt rendszeren. Úgyis meg kell mondani nekik a jelszót, és mivel a CD csak olvasható legfeljebb ideiglenesen tudják megváltoztatni azt. A *GDM* rendelkezik ugyan *autologin* képességgel, de mivel a méretet nem szeretnénk túlságosan megnövelni, jobb lenne elkerülni a rengeteg *GNOME* függőséget. Ehelyett egyszerűen a su paranccsal indítjuk az X-et nem-root felhasználóként, és lefuttatjuk az *.xsession* parancsfájlt, amely beindítja az *xterm*-et, az *Emacs*-ot és az alkalmazásainkat. Az *autologin-x* parancsfájl */etc/init.d.autologin-x* néven települ, és elkészíti a rendszerindítás folyamatába illesztéshez szükséges közvetett hivatkozásokat.
A parancsfájl a DISPLAY változó értéke alapján dönti el, hogy melyik X kiszolgálót szükséges futtatni. Ha már be van állítva, az *Xvnc*-t indítja az *XFree86* helyett. Ez a tesztelést segíti: amikor az *autologin-x*-et a *run-chroot.pl* futtatja egy *xterm* ablakban, hozzákapcsolódhatunk egy *VNC* ügyféllel és meggyőződhetünk róla, hogy valamennyi X alkalmazás megfelelően indul. Természetesen ha az CD-ROM-ról indítva szeretnénk futtatni az X-et, tudnunk kell, a felhasználó milyen videókártyával rendelkezik.

Alkatrészfelderítés

A *Linux* alkatrész-felderítési képességei igen sokat fejlődtek az utóbbi tíz évben, amit az alkatrész-technológiák fejlődése is elősegített. Egy mai *PCI* vagy *USB* alkatrészt sokkal könnyebb megbízhatóan és biztonságosan azonosítani mint egy korabeli *ISA* csatolót.
A legtöbb *Linux* terjesztésben találunk valamilyen megoldást, amely végigvizsgálja a rendszer *PCI* és *USB* eszközeit és betölti a megfelelő modulokat. A *Knoppix* a *Kudzut* használja, amit eredetileg *Red Hat*-hez írtak, az eredeti *Debian*

pedig a `discover` parancsot alkalmazza. A két megoldás alkatrész lefedettsége közel azonos; tekintve, hogy mind a kettő nyílt forrású, nyugodtan másolhatnak egymás alkatrész adatbázisából. A ***Debian*** X kiszolgáló csomagja eleve a `discover` segítségével állapítja meg az X beállítási kérdések alapértékét, így inkább ennél maradunk.

debconf

Mit kezdünk a felderített alkatrészekkel? A ***Debian*** csomagoknak kézzel szerkeszthető beállításállományuk van, de általában telepítés-utáni parancsfájlokkal interaktív módon hozzák létre azok kezdeti verzióját telepítéskor. Amikor megoldható, mint például az X és hálózati beállítások esetében, a parancsfájlok lefuttatják az alkatrész kereső eszközöket.
A gond csak az, hogy mi jelenleg ***chroot*** környezetben telepítünk a gazdagépen, és a gazda alkatrészeinek felderítése nem sokat segít a célgépen. Ezért a ***debconf*** adatbázisát valami írható helyre kell tennünk, így indításkor a `debconf-communicate` segítségével törölhetjük a csomagok beállításait, majd a ***.config*** parancsfájlok futtatásával elhitetjük velük, hogy első ízben állítjuk be őket. Ez alaposabb megközelítés mint a `dpkg-reconfigure` futtatása, ami néha olyanokat kérdez mint: „Biztos, hogy újra be kívánja állítani a csomagot?”. Ez megzavarhatja a végfelhasználót aki természetesen még semmit sem állított be.
A `debconf-communicate` kézikönyvoldalain és a tarlabda ***target/etc/init.d/configure-xserver*** állományában további részleteket találunk.

Maradandó tároló: Hotplug

A CD-ROM csak olvasható, a ramdisk megsemmisül amint a gépet lekapcsoljuk. Az emberek viszont el szeretnék menteni az állományaikat, illetve el akarják érni a meglévő merevlemezeiken és cserélhető tárolóikon (például ***USB*** kulcson vagy ***ZIP*** meghajtón) őrzött állományokat. Akárcsak az előbb, a munka legnehezebb részét már elvégezték helyettünk; most a ***hotplug*** és az ***autofs*** lesz a megmentőnk.
A ***hotplug*** az új eszközök behelyezését és eltávolítását figyeli. Amikor új ***USB*** tárolóeszközt vesz észre, betölti a megfelelő modulokat és emulált ***SCSI*** gazdát hoz létre. Továbbra is nekünk kell tudnunk azonban, hogy milyen eszközök állnak a rendelkezésünkre és nekünk kell kézzel felcsatolni őket, itt jön a képbe az ***autofs***.
Az ***autofs*** igény szerint fel- és lecsatolja a fájlrendszereket. A `map` program használatával egy ***Perl*** parancsfájlt futtatnunk valahányszor a felhasználó lekérdezi a ***/media/list*** könyvtárat; itt létrehoz egy könyvtárat, valamint elkészíti a csatlakoztatott eszközök nevének megfelelő közvetett hivatkozásokat. A hivatkozások ***autofs*** csatlakoztatási pontokra mutatnak melyeken keresztül elérhetjük a fájlrendszert.
Vizsgáljuk meg a ***target/etc/auto.master*** és ***target/usr/local/sbin/autofs-device-list*** állományokat a tarlabdában.

A rendszermag

Lényegében ugyanazt a rendszermag beállítást használjuk mint a ***Knoppix*** (vizsgáljuk meg egy futó ***Knoppix*** rendszer ***/usr/src/linux/.config*** állományát, vagy a tarlabdánk ***kernel-config*** állományát), de eltávolítunk belőle néhány számunkra nyilvánvalóan felesleges dolgot, például a ***ZISOFS***-t.

A szabványos ***Debian*** `make-kpkg` eszközcsomagok felépítik és telepítik a rendszermagot. A gazdagépen található Debian függőségről van szó (szükségünk lesz a ***cloop-src*** csomagra), és mivel valószínűleg ez az egyetlen ilyen nemegyértelmű függőség, a későbbi verziókban esetleg érdemes lehet áthozni a ***chroot***-ba.

A fájlrendszer

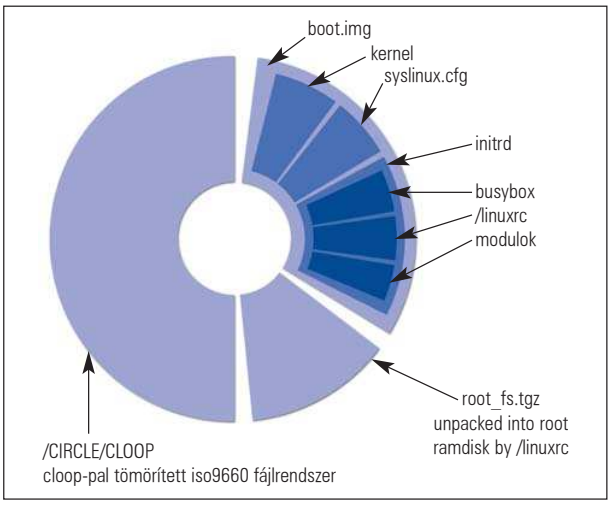
A legtöbb ***UNIX*** fájlrendszer szívesen elindul csak olvasható módban is, ám nekünk néhol szükségünk van fájlírásra. Például az X kiszolgáló beállításállományait indításkor a használt alkatrészeknek megfelelően át kell alakítanunk, a ***debconf*** adatbázist frissítenünk kellene illetve vannak különféle napló és zár-állományaink is.
A ***tmpfs*** fájlrendszer segítségével ***RAM***-alapú fájlrendszert hozunk létre. A rendszer ezt a memórialemezt fogja használni gyökérként és a /ro alkönyvtárban várja a CD lemez felcsatolását. A csak olvasható könyvtárakhoz közvetett hivatkozásokat hozunk létre, például a ***/usr***-t a ***/ro/usr***-hez rendeljük.
A csak olvasható könyvtárakról listát készítünk, és kétszeresen ellenőrizzük. Először egy tarlabdát készítünk a rendszerről amely nem tartalmazza ezeket a könyvtárakat, helyükön a megfelelő közvetett hivatkozásokkal. Ezt a tarlabdát aztán a futó rendszer gyökerébe másoljuk. Másodszor, amikor a ISO9660 képmást ***cloop***-tömörítéssel kiírjuk, ezt a könyvtárlistát fogjuk csatolni.

initrd

Mielőtt a rendszer beindulna két igen fontos dolgot kell elintéznünk. Először is fel kell csatolnunk a ***cloop*** másolatot, szükség szerint be kell töltenünk a ***CD-ROM*** modulokat, majd meg kell találnunk és fel kell csatolnunk a CD-t. Ezek után telepítjük a ***cloop*** eszközt majd felcsatoljuk rá a belső fájlrendszert. Másodszor elkészítjük a gyökér fájlrendszerhez a memórialemezt és rámásoljuk a ***root_fs.tgz*** tartalmát a CD-ről.
Az ***initrd (initial ramdisk)*** támogatás segítségével készítünk egy mini gyökér rendszert amit a rendszermag felcsatol és lefuttat mielőtt a valódi `init` elindulna. Ez gzippelt fájlrendszer lesz. Amikor ***initrd*** támogatással rendelkező rendszermagot indítunk a `initrd=filename` paraméterrel, a megadott fájlnév tartalmát fogja betölteni és memórialemezt készít belőle. Ez után memórialemezen elindítja a ***/linuxrc*** állományt.
Amikor a ***linuxrc*** befejeződött, a ***pivot_root*** hívást használja a valódi gyökér fájlrendszerbe váltáshoz (amely addig a /ramdisk volt) majd lefuttatja a valódi init-et.
A rendszermag és az ***initrd***, az indítómásolat valamennyi állományát beleszámolva, elég kicsi kell legyen, hogy beleférjen 1.44MB memóriában. Ez nem valami sok hely, hiszen a ***GNU*** önmagában 1,200K, úgyhogy kreatívnak kell lennünk.

dietlibc, Busybox

Aki soha még csak nem is vágyott ***Linuxos PDA***-ra vagy autóba építhető MP3 lejátszóra, most az is igazán hálás lehet a beágyazott ***Linux*** programozóinak.
A ***Busybox*** és ***dietlibc*** segítségével fogjuk ugyanis átjuttatni a tevét tű fokán. A ***busybox*** apró héjprogram amit fordí-



1. ábra Doboz a dobozban: a kész CD fájlrendszer másolatba ágyazott fájlrendszer másolatban megbúvó fájlrendszer másolatokat tartalmaz

táskor beállíthatunk úgy, hogy beépítettként tartalmazzon sok általános eszközt, a ***dietlibc*** pedig egy kis méretre optimalizált C könyvtár változat. Örömmel vehetjük észre, hogy a ***busybox*** mindent tud, amire az ***initrd***-ben szükségünk lehet, valamint hogy statikusan fordítva a ***dietlibc***-vel mindössze 100K helyet foglal. Összehasonlításképpen ugyanezekkel a ***Busybox*** kapcsolókkal statikusan fordítva a ***glibc***-vel 500K méretű végrehajtható állományt kapunk.
A ***Busybox*** bővítményeit a (tarlabdában található) ***Config.h*** állomány `#define` kulcsszavaival kapcsolhatjuk be. A kikapcsolt értékek talán önkényesnek tűnhetnek, de amikor könyvtárlista kérésére használhatjuk az `echo *` és `tar cvf /dev/null` parancsokat is, az `ls` igazán luxusnak tűnik.
Az ***initrd***-t a `genext2fs` programmal készítjük, így nem lesz szükségünk ***loopback*** felcsatolásra. A program egy könyvtárszerkezetből készít ***ext2*** fájlrendszert, amit ***gzip***-el tömörítünk és a rendszerindító lemezmásolatba mozgatunk (*1.ábra*).

Rendszerindítás

A CD-ROM-ról történő rendszerindítás szabványa ***El Torito*** néven vált ismertté amit eredetileg a ***Phoenix BIOS*** készítői hoztak létre. Az ***El Torito*** egy vagy több lemezképmás létrehozását teszi lehetővé a CD lemezen. Rendszerindításkor a ***BIOS*** felderíti ezeket, szimulált lemezt hoz belőlük létre és erről indítja a rendszert. A másolatok hajlékony lemez (1.44MB vagy 2.88MB) illetve merevlemez másolatok lehetnek. Ezen kívül létezik nem-emulációs mód is, ahol a ***BIOS*** egy megadott állományból tölt be szektorokat és emulált lemez kialakítása nélkül hajtja végre azokat.
Természetesen itt is van buktató: Az ***El Torito*** rendszert ***BIOS*** írók készítették. A laptoppal vagy egyéb érdekes alkatrésszel rendelkező ***Linux*** tulajdonosok jól tudják, hogy a ***BIOS***-ok nem éppen a leginkább hibamentes programok ezen a földön. Nem ok nélkül feltételezhetjük, hogy bizonyos gyártók nyugodt szívvel hagyják figyelmen kívül a hivatalos rendszerleírásokat mindaddig, míg szerkezetük az éppen aktuális ***Windows*** rendszer alatt valahogy elindul.
Így aztán akármilyen fájdalmas is a méretbeli korlát, a maximális hordozhatóság érdekében a ***Knoppix*** által kijárt utat fogjuk követni és egyetlen 1.44MB hajlékonylemezes másolatot alkalmazunk.

boot.img

Mit tegyünk ebbe az 1.44MB-ba? Indíthatunk nyers Linux rendszermagot, esetleg használhatjuk a megszokott ***LILO*** vagy ***Grub Linux*** rendszerindítót. ***H Peter Anvin SYSLINUX*** eszköze azonban használhatóság terén mindkét lehetőséget könnyedén veri. A ***SYSLINUX MS-DOS*** fájlrendszert használó indítólemezeket készít, így a hajlékonylemez másolatot a felhasználói ***mtools*** segítségével is létrehozhatjuk.
A lemeznek a rendszermag ***vmlinuz*** állományra, a ***syslinux.cfg***, csatolt segítségnyújtó állományokra és az ***initrd*** másolatra van csak szüksége. Amint készen vagyunk futtathatjuk rajta a ***SYSLINUX***-ot.
Már csak az maradt hátra, hogy elkészítsük és kiírjuk a fájlrendszerünket, amint azt korábban is tettük. A belső fájlrendszer a `$(SCRATCH)/CLOOP`-ban található. A külső fájlrendszerünk a ***boot.img*** és ***root_fs.tgz*** állományokat tartalmazza majd. Ezt kiírjuk a CD-re (egy-két CD-RW lemez sem árthat) majd elindítjuk róla a gépet. Hacsak nincs nagy balszerencsénk, működni fog.

Végszó

Számomra, régi ***Linux*** felhasználó számára, aki szokásos telepítést már évek óta nem csinált, igazán lenyűgöző, hogy milyen sokat fejlődött az automatikus alkatrész felismerés és támogatás. És ahogy telik az idő, biztos vagyok benne, hogy még ennél is csak jobb lesz.
Merre lehet továbblépni? Az ***automount*** támogatáson kell kicsit csiszolni; esetleg kipróbálhatunk valami mást, például a ***Volumatic***-ot. A többi leginkább a készülő termékünktől függ. Azonban minden parancsfájl szabad program, és érdeklődve várom a visszajelzéseket.

Linux Journal 2005. április, 132. szám

Daniel Barlow független szakértő az Angliai Oxfordban, ahol Linux és Common Lisp fordítókkal dolgozik. Szabadidejében, nagyon szeret elektromos gitáron rosszul játszani, ami nagy szerencse, ugyanis csak így tud. Megjegyzéseket a dan@metacircles.com címen várja.

KAPCSOLÓDÓ CÍMEK

debconf:
➲ www.kitenet.net/programs/debconf

El Torito:
➲ www.phoenix.com/resources/specs-cdrom.pdf

Hotplug:
➲ linux-hotplug.sf.net

SYSLINUX:
➲ syslinux.zytor.com

VNC:
➲ www.realvnc.com

InfiniBand és Linux

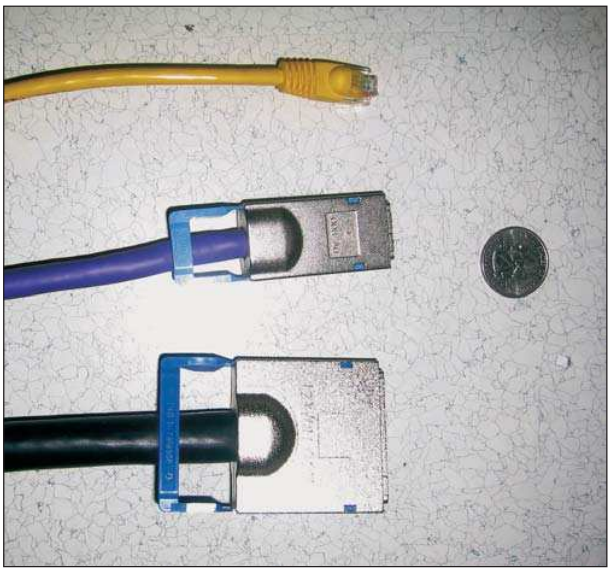
Vajon miért is jó, ha a hálózatunk távoli rendszere irkálhat a memóriánkba, hogyan küldhetnek felhasználói programok a rendszermag zaklatása nélkül adatokat, valamint néhány újabb adalék a nagy sebességű kapcsolattartásról.

Hosszú vajúdást követően végül megszületett az ***InfiniBand (IB)***, sőt a ***Linux IB*** támogatása is már fejlesztés alatt áll. Fizikai szinten az ***IB*** nagyon hasonlít a ***PCI Express***-re. Az adatokat több nagy sebességű soros csatornán továbbítja. Az ***InfiniBand*** első változatának leírása valamennyi csatornán azonos jelzésszélességet engedélyezett, mégpedig 2.5Gb/s-ot, akárcsak a ***PCI Express***.
A leírás legutóbbi változata (1.2), ugyanakkor már támogatja a csatornánkénti 5Gb/s és 10Gb/s sebességeket. Ezen kívül az IB az 1X, 4X, 8X és 12X, míg a ***PCI Express*** a X1, X2, X4, X8, X12, X16 és X32 szélességeket támogatja. A leggyakrabban használt ***IB*** sebesség manapság a 4X 2.5Gb/s/csatorna sebesség mellett, azaz összesen 10Gb/s. A 12X szélesség a 10Gb/s/csatorna sebességgel kombinálva azonban a jelenlegi ***IB*** leírás szerint akár a lenyűgöző 120Gb/s átviteli teljesítményt is támogatja.
Minthogy az ***IB*** általában hálózatépítésre használatos, a réz és optikai kábelezést egyaránt támogatja, míg a ***PCI Expressz*** kábelmeghatározásai még fejlesztés alatt állnak.
A legtöbb ***IB*** telepítés rézkábelezést alkalmaz (*1. ábra*) amely körülbelül 10 méterig használható. Az ***IB*** ezen kívül lehetővé tesz optikai kábelezést is, amely elméletileg akár 10 kilométerig is használható.

Az elmúlt években az ***IB***-t tekintették a ***PCI*** utódjának, de manapság már nem ez a helyzet. Az ***IB*** adapterek továbbra is inkább külső eszközök maradnak amelyek ***PCI***, ***PCI Express***, ***HyperTransport*** vagy hasonló periféria buszokon keresztül csatlakoznak a rendszerhez.
A rendszereket az ***IB*** hálózathoz csatoló hálózati eszközt ***Host Channel Adapternek (HCA)*** nevezik. Az eszköz igen magas sebessége mellett az ***IB HCA*** egy üzenetközvetítő felületet is biztosít, melynek segítségével az ***InfiniBand*** 10Gb/sec (vagy még nagyobb) sebességét ki tudják aknázni a rendszerek. Az ***IB*** sebességének kihasználásának kulcsa a zéró másolású hálózatkezelés támogatása; máskülönben az alkalmazások egyfolytában csak az adataikat másolnák.
A zéró másolást a ***HCA*** felület három kulcseleme teszi lehetővé: az elvont, magasszintű munkasor, a rendszermag megkerülés és a ***távoli közvetlen memória elérés (Remote DMA; RDMA)***. Az elvont munkasor annyit jelent, hogy az alkalmazások nem csomagról csomagra formálják meg és dolgozzák fel a hálózati csomagokat, hanem kérelmeket küldenek a ***HCA*** által kezelt soroknak.
Egy munkakérelmen belül küldött üzenetet legfeljebb 4GB hosszú lehet. A ***HCA*** felügyeli az üzenet csomagokra tördelését illetve várakozik a jóváhagyásokra és küldi

1. táblázat ***Összehasonlító tábla***

Technológia	Adatátviteli sebesség	Kábelezés
USB	12Mb/s	5m
Hi-Speed USB (USB 2.0)	480Mb/s	5m
IEEE 1394 (FireWire)	400Mb/s	4m
Gigagbit Ethernet	1,000Mb/s	100m (cat5 kábel)
10 Gigabit Ethernet	10,000Mb/s	10m (réz IB kábel), 1+ km (optikai)
Myrinet	2,000Mb/s	10m (réz), 200m (optikai)
1X InfiniBand	2,000Mb/s	10m (réz), 1+ km (optikai)
4X InfiniBand	8,000Mb/s	10m (réz), 1+ km (optikai)
12X InfiniBand	24,000Mb/s	10m (réz), 1+ km (optikai)



1. ábra Felülről lefelé: cat5 Ethernet kábel, 4X InfiniBand kábel és 12X InfiniBand kábel (méretarány kedvéért egy negyeddolláros érme látható)

1. lista IPoIB Meghajtó alaphelyzetbe állítása

```
struct ib_qp_init_attr init_attr = {
    .cap = {
        .max_send_wr  = IPOIB_TX_RING_SIZE,
        .max_recv_wr  = IPOIB_RX_RING_SIZE,
        .max_send_sge = 1,
        .max_recv_sge = 1
    },
    .sq_sig_type = IB_SIGNAL_ALL_WR,
    .rq_sig_type = IB_SIGNAL_ALL_WR,
    .qp_type     = IB_QPT_UD
};
priv->pd = ib_alloc_pd(priv->ca);
priv->cq = ib_create_cq(priv->ca,
            ipoib_ib_completion,
            NULL, dev,
            IPOIB_TX_RING_SIZE +
            IPOIB_RX_RING_SIZE + 1);
if (ib_req_notify_cq(priv->cq, IB_CQ_NEXT_COMP))
    goto out_free_cq;
priv->mr = ib_get_dma_mr(priv->pd,

IB_ACCESS_LOCAL_WRITE);
init_attr.send_cq = priv->cq;
init_attr.recv_cq = priv->cq,
priv->qp = ib_create_qp(priv->pd, &init_attr);
```

újra az elveszett csomagokat. Mivel a *HCA* alkatrész felügyeli a nagyméretű üzenetek kézbesítését bármiféle *CPU* felhasználás nélkül, az alkalmazások több *CPU* időt használhatnak fel a küldött és fogadott adatok elkészítéséhez és feldolgozásához.

A rendszermag megkerülése lehetővé teszi, hogy az alkalmazások közvetlenül küldjenek munkakérelmeket és fogadjanak befejezési eseményeket a *HCA* sorairól, így a rendszerhívásokkal járó felesleges környezetváltás elkerülhető. A rendszermag meghajtó beállítja a sorokat, ahol a szokásos memória védelem gondoskodik arról, hogy minden folyamat csak a saját erőforrásait érje el. Valamennyi nagysebességű irányítási (path) művelet ugyanakkor a felhasználói térben megy végbe.
Az utolsó elem, az *RDMA*, lehetővé teszi, hogy az üzenet magával vigye a célcímet ahová a memóriában majd kerülnie kell. Az adat helyének megadása például az *IB* alatti tárolók kialakításakor lehet hasznos, ahol a kiszolgáló lemezolvasásai teljesen rendszertelenek lehetnek. *RDMA* nélkül vagy a kiszolgálónak kell időt pazarolnia mikor áll elküldésre készen az adat vagy az ügyfélnek kell *CPU* erőt pazarolnia az adatok végső helyre másolásához.
Bár a távoli rendszerek memóriájába firkálással szemben vannak bizonyos fenntartások az *IB* használó alkalmazások szigorú határokat és jogosultságokat állíthatnak be az *RDMA* szolgáltatásnak. Az *IB RDMA* legalábbis biztonságosabb dolog mint a lemezvezérlő *DMA*-ját a memóriába engedni.
Nagy sebességén felül, az *IB* egyúttal a *fürtök (clusters)* készítését és kezelését is leegyszerűsíti, hiszen egyetlen eszköz szállítja a hálózati és tároló forgalmat valamint a fürt kommunikációt. Több csoport is hozott létre különféle *IB* felett futtatható felső szintű protokollokat, néhány ezek közül:

- *IP-over-InfiniBand (IPoIB)*: az *Internet Engineering Task Force (IETF)* szabványfejlesztő munkacsoportja az *IB* alatti *IP* forgalomküldés ösvényét tapossa ki. Ezek az ösvények idővel a *IpoIB RFC* szabványává válhatnak. Az *IPoIB* ugyanakkor nem használja maximálisan ki az *IB* képességeit, hiszen a forgalom továbbra is az *IP* vermen keresztül utazik és csomagról csomagra kerül elküldésre. Az *IPoIB* egyszerű lehetőséget biztosít régebbi alkalmazásaink futtatására vagy a vezérlőforgalom átküldésére *IB*-n keresztül.

- *Sockets Direct Protocol (SDP)*: az *InfiniBand Trade Association* maga adott meg egy olyan protokollt, amely a szabványos socket műveleteket helyi *IB RDMA* műveletekké alakítja. Ezáltal a socket alkalmazások változtatás nélkül futtathatók, ugyanakkor az *IB* teljesítményének szinte minden előnyét élvezhetik.

- *SCSI RDMA protokoll (SRP)*: a *SCSI* szabványokért felelős *InterNational Committee for Information Technology Standards (INCITS) T10* bizottság, kiadott egy szabványt, amely leírja hogyan kell a *SCSI* protokollt az *IB*-hoz rendelni. A második generációs *SRP-2* protokoll fejlesztése jelenleg folyik.

Sok más csoport is tanulmányozza az *IB* kihasználási lehetőségeit, például a *DAT Collaborative* és az *Open Group Interconnect Software Konzorciuma* készített *API*-kat, *RDMA* csatolásokat az *NFS*-hez valamint *IB* támogatást különféle *MPI* csomagokhoz.

2. lista IPoIB meghajtó küldési kérelem feladása

```
priv->tx_sge.lkey            = priv->mr->lkey;
priv->tx_sge.addr            = addr;
priv->tx_sge.length          = len;
priv->tx_wr.opcode           = IB_WR_SEND;
priv->tx_wr.sg_list          = &priv->tx_sge;
priv->tx_wr.num_sge          = 1;
priv->tx_wr.send_flags       = IB_SEND_SIGNALED;
priv->tx_wr.wr_id            = wr_id;
priv->tx_wr.wr.ud.remote_qpn = qpn;
priv->tx_wr.wr.ud.ah         = address;

ib_post_send(priv->qp, &priv->tx_wr, &bad_wr);
```

Természetesen nyílt forráskódú támogatás nélkül az egész csinos eszköz nem lenne különösebben érdekes a ***Linux*** világban. Szerencsére az ***OpenIB Alliance*** ipari szövetség feladata pontosan az, hogy egy teljesen nyílt forrású ***IB*** vermet hozzon létre. Az ***OpenIB*** jelenleg 15 tag-céggel rendelkezik, amelyek között egyaránt találunk ***IB*** alkatrész terjesztőket, kiszolgálócégeket, programcégeket és kutatási szervezeteket.

Az ***OpenIB*** programon 2004 februárjában kezdődtek meg a munkálatok, az első rendszermag meghajtó pedig 2004 decemberében került a rendszermag fába, közvetlen azután, hogy a 2.6.10-es kiadás után megszületett a 2.6.11-es fa. A rendszermagba illesztett első ***IB*** meghajtó kód csomag éppen csak annyit tud, hogy már azért működőképes. Tartalmaz egy középréteget, amely elfedi az alacsonyszintű alkatrészmeghajtókat a felsőszintű protokollok elől, egyetlen alacsonyszintű meghajtót a ***Mellanox HCA***-hoz, valamint egy ***IPoIB*** felsőszintű protokoll meghajtót amellyel egy alhálózat kezelőt futtathatunk a felhasználói térben.

Az ***IPoIB*** meghajtó kódjának néhány apró részlete sokat elárulhat a rendszermag ***IB*** támogatásának felhasználási lehetőségeiről. Amennyiben egészében szeretnénk látni a kódot, a teljes ***IPoIB*** meghajtót kell megnéznünk, amelyet egyébként a ***Linux*** forrásfájának ***drivers/infiniband/ulp/ipoib*** könyvtárában találjuk meg.

Az *1. lista* bemutatja, mit is csinál az ***IPoIB*** meghajtó az ***IB*** erőforrások lefoglalásához. Először meghívja az ib_alloc_pd() függvényt, amely lefoglalja a ***védelmi tartományt (protection domain vagy PD)***, ez egy átlátszó tároló amelyre valamennyi ***IB*** felhasználónak szüksége van az erőforrások tárolásához.

Itt most kihagytuk a listából a helyes hibakezelést, annak ellenére, hogy minden valódi rendszermag kód valamennyi függvény visszatérési értékét ellenőrizné. Minden erőforrást foglaló és mutatót visszaadó ***IB*** függvény a szokásos ***Linux*** módszert alkalmazza a hibák visszaadására az `ERR_PTR()` makrón keresztül, következésképpen az állapotot az `IS_ERR()` segítségével le lehet kérdezni. Például a `ib_alloc_pd()` hívás a valódi rendszermag kódban a következőképpen nézne ki:

```
priv->pd = ib_alloc_pd(priv->ca);
if (IS_ERR(priv->pd)) {
        printk(KERN_WARNING "%s: failed "
                "to allocate PD\n", ca->name);
        return -ENODEV;
}
```

Ezek után a rendszermag meghívja az `ib_create_cq()` függvényt, amely létrehozza az ***befejeződési sort (completion queue, azaz CQ)***. A meghajtó igényli, hogy az befejeződési esemény bekövetkeztekor meghívódjon az `ipoib_ib_completion()` függvény, valamint, hogy a ***CQ*** legalább `IPOIB_TX_RING_SIZE + IPOIB_RX_RING_SIZE + 1` darab munkabefejezési struktúrát tárolhasson. Erre a méretre abban a különleges esetben van szükség, amikor a meghajtó a lehető legnagyobb számú üzenetet küldi és és fogadja majd nem tud lefutni, míg valamennyi el nem készül. Elég zavaró módon, a ***CQ***-k olyan ***IB*** erőforrások, amelyek nincsenek kapcsolatban a ***PD***-kel, ezért a függvénynek nem is kell átadnunk ***PD*** adatot.

Amikor a ***CQ*** elkészült a meghajtó az `ib_req_notify_cq()` függvényhez fordul és kérelmezi, hogy amikor a következő munkabefejezés a ***CQ*** sorba kerülésére hívódjon meg a befejezési esemény függvény. Az `ipoib_ib_completion()` eseményfüggvény feldolgozza a befejezéseket amíg a ***CQ*** ki nem ürül, majd meghívja az `ib_req_notify_cq()`-t. Így újra elindul, amint újabb befejezések válnak elérhetővé.

A meghajtó ekkor meghívja az `ib_get_dma_mr()`-t amely beállítja a rendszermag ***DMA*** osztó ***API***-tól kapott ***DMA*** címmel használható memóriaterületet (***MR***). Ennek kezeléséhez az ***IB HCA***-ban to elkészülnek az átalakító táblák, és egy helyi kulcsot (`L_Key`) adunk vissza, amelyet aztán tovább lehet adni a ***HCA***-nak, és hivatkozhat az adott ***MR***-re.

Végül a meghajtó az `ib_create_qp()`-t hívja meg, amely létrehoz egy sorpárt (***QP***). Ezt az objektumot azért nevezik sorpárnak mert két darab munkasort tartalmaz. Egyet a küldési kérelmeknek és egyet a fogadási kérelmeknek. A ***QP*** létrehozásához először ki kell töltenünk a meglehetősen méretes `ib_qp_init_attr` szerkezetet. A fedőstruktúra a készítendő küldő és fogadó sorok méretét adja meg. A `sq_sig_type` és az `rq_sig_type` mezőket `IB_SIGNAL_ALL_WR` értékre állítjuk, így minden munkakérelem befejezési eseményt vált ki .

A `qp_type` mezőt `IB_QPT_UD` értékre állítjuk, létrehozva ezzel a ***megbízhatatlan adatcsomag (unreliable datagram avagy UD) QP***-t. Az ***IB QP*** esetében négyféle szállításról beszélhetünk: ***megbízhatóan összekapcsolt (RC)***, ***megbízható adatcsomag (RD)***, ***megbízhatatlanul kapcsolt (UC)*** és ***megbízhatatlan adatcsomag (UD)***.

A megbízható szállítások esetében az ***IB*** alkatrész garantálni tudja, hogy minden egyes üzenet vagy sikeresen kézbesítődik, vagy hibát jelez amennyiben a hibát egy javíthatatlan hiba okozza (például kihúzzuk a kábelt). A kapcsolt adatszállítás esetében minden üzenet egyetlen célállomásra irányul, amelyet a ***QP*** beállításakor határozunk meg, az adatcsomag szállítás esetében minden egyes csomag különböző célokra irányulhat.

Amikor az ***IPoIB*** meghajtó elkészítette a ***QP***-t, a kapott csomagokat a ***QP***-n keresztül küldi a hálózati verembe.

A *2. lista* mutatja be, hogy mit kell tennünk akkor,

ha egy kérelmet akarunk helyezni a *QP* küldési sorába. Először is a meghajtó a küldési kérelemhez felállít egy gyűjtőlistát. Az `lkey` mező annak az *MR*-nek az `L_Key` értékét veszi fel, amelyet az `ib_get_dma_mr()`-ből kaptunk. Minthogy az *IPoIB* olyan csomagokat küld, amelyek egyetlen összefüggő részben találhatóak, a gyűjtőlistában mindössze egyetlen bejegyzés található. A meghajtónak csak a címet és csomag hosszát kell megadnia. A gyűjtőlistában található cím nem a virtuális cím hanem a `dma_map_single()` függvénnyel megkapott *DMA* cím lesz. Általánosságban a programok hosszabb gyűjtőlistákat is használhatnak, ezáltal rákényszerítve a *HCA*-t, hogy egyetlen üzenetbe gyűjtsön össze több puffert és így nem kell egyetlen pufferbe másolnia az adatokat.

A meghajtó ezután kitölti a munkakérelem maradék mezőit. Az `opcode`-ot `send`-re állítja, az `sg_list` és `num_sge` értékeket az éppen kitöltött gyűjtőlista címe kerül a `send_flags` értéke pedig `signaled` lesz, így a munkakérelem befejezést vált ki amikor végez. A távoli *QP* számot és címkezelőt beállítjuk, majd a `wr_id` mezőbe a meghajtó munkakérelem azonosítója kerül.

A munkakérelem kitöltése után a meghajtó meghívja az `ib_post_send()`-et, amely végül a kérelmet felveszi a sorba. Amikor a kérelmet teljesítette az *IB* alkatrész, a munka teljesítése a meghajtó *CQ* sorába kerül, amelyet végül az `ipoib_ib_completion()` dolgoz fel.

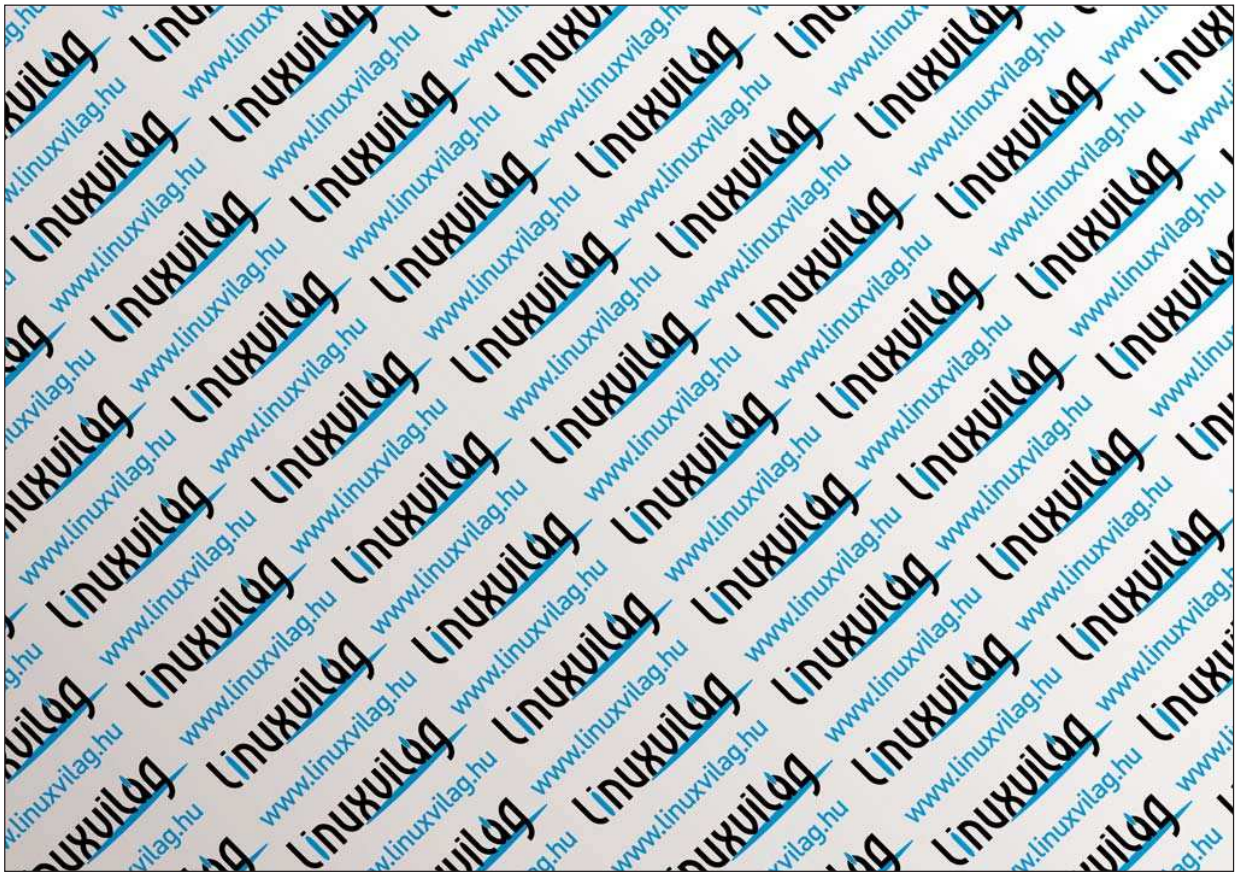
Az *InfiniBand* rengeteg mindenre képes és az *OpenIB Alliance* éppen csak elkezdte megírni a képességeit kihasználó programokat. Most, hogy már a Linux rendelkezik az alapvető *IB* támogatással, nekiláthatunk a felsőszintű protokollok elkészítésének, ideértve például az *SDP*-t és a tároló protokollokat. A másik komolyabb terület amely jelenleg foglalkoztat bennünket, a közvetlen felhasználói térbeli *IB* elérés, azaz a korábban említett rendszermag megkerülő képesség. Rengeteg érdekes dolgot lehet még csinálni az *IB*-hez és az *OpenIB Projekt* mindenki számára nyitott, kalandra fel tehát, csatlakozz a mókához.

Linux Journal 2005. május, 133. szám

Roland Dreier Linux InfiniBand meghajtók karbantartója és vezető fejlesztője az OpenIB.org projekt keretében. Roland a Kaliforniai Berkeley egyetemen szerezte matematika doktorátusát és több különféle titulusa volt már akadémiai körökben valamint a műszaki világban. 2001 óta a Topspin Communications alkalmazásában áll.

KAPCSOLÓDÓ CÍMEK

- www.openib.org
- www.infinibandta.org
- www.ietf.org
- www.t10.org
- www.datcollaborative.org



VIA PadLock – Boszorkányosan gyors titkosítás

A VIA olcsón beszerezhető processzora támogatja az Advanced Encryption Standardet, így segítségével kiemelkedő szintű titkosítást végezhetünk, méghozzá hardveres szintű sebességgel.

Aki használt már titkosítást, az valószínűleg hamar észrevette, hogy a művelethez nem csekély processzorteljesítményre van szükség. A régebbi rendszereken például a titkosított fájlrendszerek használata lassabb fájlműveleteket eredményez, az újabb rendszereken pedig legalábbis magasabb processzorterheléssel kell számolnunk. A hálózati forgalom ***IPsec*** alapú titkosítása szintén lelassítja a dolgokat, és sokszor még egy hétköznapi, 100 Mbps sebességű hálózaton is észlelhető a teljesítmény visszaesése.

- A titkosítás/teljesítmény kompromisszum kezelésére többféle út kínálkozik:
- Nem használunk titkosítást: látszólag a legolcsóbb megoldás, ám hosszú távon rendkívül drágává is válhat.
- A lassúbbodás elfogadása: a legjellemzőbb megoldás.
- Különálló titkosításgyorsító eszköz használata: például ***PCI*** buszra illeszkedő kártya, amely sajnos nem javít annyit a helyzeten, mint várnánk, ugyanis az adatoknak a feltétlenül szükségesnél többször kell megjárniuk a ***PCI*** buszt.
- ***VIA PadLock*** technológiát használó processzor használata. De vajon mi az a ***VIA PadLock***? Rögtön kiderül.

VIA PadLock

Nem is oly rég a ***VIA*** egyszerű és egyben vitatható ötlettel állt elő: válasszunk ki néhány kriptográfiai algoritmust, majd drótozzuk be őket a processzorba. Létrejött tehát egy ***i686*** osztályú processzor, amely néhány új, kifejezetten titkosítási célt szolgáló utasítást is képes megérteni. A technológia a ***VIA PadLock*** nevet kapta, míg maga a processzor teljes mértékben kompatibilis az ***AMD Athlon*** és az ***Intel Pentium*** lapkákkal.
A rendelkezésre álló ***PadLock*** szolgáltatások körét a gép processzorának változata határozza meg. A processzorváltozatokat általában család-modell-fokozat hármassal jelölik. Az ***i686*** osztályú processzorok mindegyike a 6-os családba tartozik. Ha a modellszám 9-es, akkor a processzor ***Nehemiah***, ha 10-es, akkor pedig ***Esther*** maggal rendelkezik. A fokozatok az egyes modellek megújított változatai. A processzor változatát a ***/proc/cpuinfo*** fájlban találjuk meg.
A ***Nehemiah 3***-as fokozata, illetve az újabb példányok ***elektromos zaj alapú véletlenszám-előállítóval (random number generator, RNG)*** rendelkeznek, ami meglehetősen jó véletlen értékeket képes előállítani különféle célokra. Az ***RNG*** elérésére az `xstore` parancs szolgál. Az ***Intel*** és az ***AMD*** processzorokhoz hasonlóan a ***VIA*** processzorok véletlenszám-előállítóját is a `hw_random` illesztőprogram támogatja.
A ***Nehemiah 8***-as és újabb fokozatai már két független ***RNG***-t tartalmaznak, valamint az ***Advanced Cryptography Engine***-nel ***(speciális kriptográfiai motor, ACE)*** bővültek. Az ***ACE*** az ***Advanced Encryption Standard (speciális titkosítási szabvány, AES)*** algoritmus alkalmazásával, háromféle szabványos kulcshosszal – 128, 192 és 256 bit – négyféle módban képes adatokat titkosítani és visszafejteni. Az üzemmódok a következők:

- Elektronikus kódkönyv (electronic codebook, ECB)
- Titkosítási blokkláncolás (cipher block chaining, CBC)
- Titkosítási visszacsatolás (cipher feedback, CFB)
- Kimeneti visszacsatolás (output feedback, OFB)

Az ezeknek megfelelő utasítások az `xcryptecb`, az `xcryptcbc` stb. A továbbiakban ezeket közös névvel `xcrypt`-nek fogom nevezni, az egyes módokhoz tartozó parancsokat nem fogom használni.
Az ***Esther*** mag 0-s és újabb fokozatai örökölték a ***Nehemiah*** két ***RNG*** egységét. Az ***ACE*** kibővült a ***számláló (counter, CTR)*** mód támogatásával, valamint az ***üzenethitelesítő kódok (Message Authentication Code, MAC)*** számításának lehetőségével. Itt két újabb rövidítéssel kellett megismerkednünk, ezek a ***PHE*** és a ***PMM***. A ***PadLock Hash Engine (PadLock kivonatoló motor, PHE)*** adott bemeneti blokk kriptográfiai célú kivonatának elkészítésére alkalmas; az ***SHA1*** vagy az ***SHA256*** algoritmussal dolgozik. Az ide kötődő parancs az `xsha`.
A ***PadLock Montgomery Multiplier (PMM, PadLock Montgomery szorzó)*** az aszimmetrikus, más szóval nyilvános kulcsú titkosítási eljárások során a legszélesebb körben használt parancsok egyikének felgyorsítását szolgálja; ez az A^B ***mod M***, ahol az ***A***, a ***B*** és az ***M*** hatalmas nagy, általában 1024 vagy 2048 biten ábrázolható szám. Szolgáltatásait a `montmul` utasítással vehetjük igénybe.
Mint már utaltam rá, a továbbiakban leginkább az xcrypt utasításokról lesz szó. Az ismertetésre kerülő alapelvek túlnyomó részt a többi egységre is érvényesek, és az xcrypt csupán példaként szolgál. Sőt, a titkosítás kapcsán említett kifejezések és fogalmak a visszafejtés területén is megőrzik érvényüket.

A PadLock használata

A külső, általában a ***PCI*** buszra csatlakozó kriptográfiai gyorsítókkal ellentétben a ***PadLock*** motor a processzor beépített része. Ennek köszönhetően jóval egyszerűbb a használata,

1. táblázat *Az OpenSSL teljesítményteszt eredménye 1,2 GHz-es órajelű VIA Nehemiah processzoron [kBps]*

Típus	16bájt	64 bájt	256 bájt	1024 bájt	8192 bájt
aes-128-ecb (szoftveres)	11274,53	14327,79	14608,64	14672,55	14693,72
aes-128-ecb (PadLock)	66892,82	346583,52	910704,21	1489932,59	1832151,72
aes-128-cbc (szoftveres)	8276,27	12915,75	13264,13	13313,02	13322,92
aes-128-cbc (PadLock)	48542,30	241898,79	523706,28	745157,61	846402,90

hiszen nem kell a busz elérésével vagy éppen a megszakítások és az aszinkron műveletek lekezelésével foglalkozni. Egy-egy memóriablokkot titkosítani egy `xcrypt` utasítással ugyanolyan egyszerű, mint átmásolni a `movs` utasítással. Ezen a szinten a titkosítás majdnem oszthatatlan művelet. Az utasítás végrehajtása előtt a puffer a nyílt adatokat tartalmazza, néhány órajellel később, az utasítás végrehajtásának befejezése után pedig már a titkosítottakat. Ha a kért művelet egyetlen blokk feldolgozása, ami az ***AES*** algoritmus esetében 16 bájtot jelent, akkor a művelet teljesen oszthatatlan. Vagyis, a processzor nem szakítja meg a műveletet, és semmi másba nem kezd bele, amíg a titkosítással nem végzett. De mi történik, ha a puffer több gigabájtnyi titkosítandó adatot tartalmaz? Nem volna túl jó ötlet minden más műveletet leállítani a hosszan tartó titkosítás befejezéséig. Ilyenkor a processzor minden egyes 16 bájtos blokk után jogosult megszakítani a titkosítást. Elmenti a pillanatnyi állapotot, majd elvégzi a rá várakozó feladatokat, mint a megszakítások kezelése vagy az egyéb folyamatok futtatása. A titkosítási folyamat újraindításakor az utasítás elvégzése attól a ponttól folytatódik, ahol felfüggesztésre került. Ez az, amiért csak majdnem oszthatatlan a művelet: a hívó folyamat számára annak látszik, ám nagyobb fontosságú esemény megszakíthatja.
A pillanatnyi feldolgozási állapot és a futó folyamat regiszterei elmentésre kerülnek a memóriába, vagyis egyszerre több folyamat is végezhet titkosítást, nem kell adataik összekeveredésétől tartani. Ismétlem, mindez sokban hasonlít a memóriablokkoknak a `movs` utasítással végzett másolására.

Mennyire gyors?

A ***VIA*** adatai szerint egy 1,2 GHz órajelű processzor maximális átviteli sebessége több mint 15 Gbps, vagyis majdnem 1,9 GBps. Az általam végzett teljesítménymérések igazolták, hogy valós alkalmazásoknál is láthatunk hasonló sebességeket, nemcsak a ***VIA*** marketinganyagaiban, ami rendkívül kellemes meglepetés volt.
A tényleges titkosítási sebesség két tényezőtől függ, a titkosítási módtól és az adatok igazításától. Az ***ECB*** a leggyorsabb, a legszélesebb körben használt ***CBC*** pedig nagyjából az ***ECB*** sebességének felét képes hozni. A ***PadLock*** megköveteli az adatok 16 bájtos határokra igazítását, ezért az igazítás nélküli adatokat először a megfelelő címekre kell mozgatni, ami némi időveszteséget okoz. Egyes esetekben az ***Esther*** processzorok képesek az adatok önműködő igazítására is, ám a sebességcsökkenés ekkor is fellép.
Az *1. táblázat* méréseim eredményeinek egy részét tartalmazza. Az eredményeket az ***OpenSSL*** teljesítményteszttel, 1,2 GHz-es ***VIA Nehemiah*** processzoron kaptam, a mennyiségek mértékegysége kBps.

Minél nagyobbak a blokkok, annál jobbak az eredmények, ugyanis eltűnnek az ***OpenSSL*** könyvtár által okozott többletterhelések. A 8 kB-os blokkok ***ECB*** módban végzett titkosítása körülbelül 1,7 GBps sebességgel lehetséges, míg ***CBC*** módban 800 MBps-ot kapunk.
A szoftveres titkosítással összehasonlítva a ***PadLock ECB*** módban ugyanazon a processzoron 120-szor gyorsabb, míg a ***CBC*** mód esetében 60-szoros az eltérés.
A gyorsulásnak köszönhetően az ***IPsec*** 100 Mbps sebességű hálózaton teljes értékűen, nagyjából 11 MBps sebességgel működik. A titkosított fájlrendszerek esetében hasonló sebességjavulásokat lehet tapasztalni. A ***Bonnie*** teljesítménytesztet egy ***UDMA100*** módban üzemelő ***Seagate Barracuda*** merevlemezzel futtatva, nyílt adatokkal 61543 kBps-os átviteli sebességet kapunk. ***PadLockot*** használva ez 49961 kBps-ra mérséklődik, míg tisztán szoftveres titkosítás használatakor csupán 10005 kBps-ot lehet elérni. A ***PadLock*** alkalmazásakor tehát a titkosítatlan átvitelhez képest csupán 20 százalékos teljesítménycsökkenéssel kell számolni, míg a szoftveres megoldásnál ez közel 85 százalék. A források között szerepel egy a teljesítménymérések eredményeit tartalmazó oldalamra mutató hivatkozás; ott további részletek és számok is megtalálhatók.

Linux alatti támogatás

A linuxos támogatást eddig – az alábbi csomagokhoz – csak az `xcrypt` utasítás révén használható ***AES*** algoritmushoz készítettem el, ***Esther*** magos processzorhoz ugyanis még nem jutottam hozzá. Amint megkapom az új processzor egy példányát, szükség szerint meg fogom oldani a többi algoritmus támogatását is.

Rendszermag

Ha a rendszermagnak szüksége van az ***AES*** algoritmusra, akkor alapesetben az `aes.ko` modult tölti be, amely az algoritmus szoftveres megvalósítása. Ha az ***AES***-t a ***PadLockkal*** akarjuk futtatni, akkor az `aes.ko` helyett a `padlock.ko` modult kell betöltenünk. Ezt kézzel, a `modprobe` paranccsal tehetjük meg, illetve az alábbi sort hozzáadva a */etc/modprobe.conf* fájlhoz:

```
alias aes padlock
```

Ezt követően minden alkalommal, amikor ***AES***-re lesz szüksége, a rendszermag a `padlock.ko` modult is betölti.
Foltok a rendszermag 2.6.5-ös és újabb változataihoz léteznek; lásd a források közötti linuxos ***PadLock*** oldalt. Az alapszintű illesztőprogram a 2.6.11-es rendszermag alapváltozatában, mindenféle foltozás nélkül is elérhető lesz.

OpenSSL

Azok, akik elég merészek ahhoz, hogy az ***OpenSSL*** újabb, ***CVS*** alatti változatait használják, már rendelkeznek a ***PadLock*** támogatásával. Az ***OpenSSL 0.9.7***-es változatát futtatóknak meg kell foltozniuk és újra kell fordítaniuk a könyvtárat; illetve lehetséges, hogy terjesztésük csomagjai már eleve tartalmazzák a foltot, ilyen terjesztés például a ***SuSE Linux 9.2***. Azt, hogy saját ***OpenSSL*** példányunk rendelkezik-e ***PadLock*** támogatással, az alábbi egyszerű paranccsal vizsgálhatjuk meg:

```
$ openssl engine padlock
(padlock) VIA PadLock (RNG, ACE)
```

Az `(RNG, ACE)` helyett lehet, hogy `(no-RNG, no-ACE)` jelenik meg, ez azt jelenti, hogy bár ***OpenSSL***-ünk képes a ***PadLock*** támogatására, gépünk processzora nem nyújt ilyen szolgáltatást. Lehet, hogy csúf hibaüzenet jelzi, hogy nincs ilyen motor a gépünkön, ekkor el kell végeznünk az ***OpenSSL*** könyvtár frissítését vagy foltozását. A titkosítási műveleteik végrehajtására az ***OpenSSL***-t használó programoknak a ***PadLock*** támogatás használatához az úgynevezett ***EVP_interface***-t kell használniuk, valamint valamikor futásuk elején el kell végezniük a hardveres gyorsító kezdeti értékadását:

```
#include <openssl/engine.h>
int main ()
{
    [...]
    ENGINE_load_builtin_engines();
    ENGINE_register_all_complete();
    [...]
}
```

További részletek az ***OpenSSL*** leírásának ***evp(3) man*** oldalán találhatók. Például ***SuSE Linux 9.2*** alatt az ***OpenSSH***-nak van egy hasonló foltja, amivel gyorsabb hálózati ***scp***-átviteleket lehet végezni.

Binutils

Ha a ***PadLockot*** saját programjainkban akarjuk használni, akkor a megfelelő utasítást név szerint is meghívhatjuk – mint például `xcryptcbc` –, de hexadecimális formában, közvetlenül is írhatjuk:

```
.byte 0xf3,0x0f,0xa7,0xd0
```

A régebbi fejlesztőeszközökkel való együttműködés biztosítása érdekében inkább a műveleti kódos formátumot használjuk. A ***Binutils 2.15***-ös és újabb változatai ugyanakkor, ahol kell – ilyen például a ***gas (GNU assembler)*** és az ***objdump*** –, ott ismerik a szimbolikus neveket is. A ***binutils*** többek közt az utasításszintű műveletekért felelős ***BFD*** könyvtárát használja a ***GNU gdb*** hibakereső is. Egy titkosító függvény kiíratása például így nézhet ki:

```
(gdb) x/3i $pc
0x8048392 <demo1+14>:    lea    0x80495f0,%edx
0x8048398 <demo1+20>:    repz xcryptecb
0x804839c <demo1+24>:    push   %eax
```

Mint sejteni lehetett, a ***SUSE Linux 9.2*** minden vonatkozó csomagja rendelkezik ***PadLock*** foltokkal, így a ***PadLock*** támogatás ennél a terjesztésnél már gyári állapotban is megoldott. Akinek a terjesztése nem rendelkezik a foltokkal, az látogasson el a források között szereplő ***Linux PadLock*** honlapomra, és keresse meg a számára szükségeseket.

A PadLock programozása

Az alábbiakban néhány a ***PadLock*** programozásával kapcsolatos irányelvet szeretnék ismertetni, és kicsit részletesebben is foglalkozni fogok az `xcryptcbc`-vel. Példát mutatok a ***PadLock*** alkalmazására egy adatpuffer ***AES*** algoritmussal, 128 bites kulccsal, ***CBC*** módban végzett titkosítására. Az `xcrypt` csoport összes többi utasítása is pontosan így használható, illetve az egyéb ***PadLock***-szolgáltatások is hasonlóan működnek.

xcryptcbc

Az `xcryptcbc` semmilyen explicit operandussal nem rendelkezik. Ehelyett minden regiszternek meghatározott szerepet ad:

- `ESI` – forráscím.
- `EDI` – célcím.
- `EAX` – kezdeti értékadási vektorcím.
- `EBX` – titkosító kulcs címe.
- `ECX` – a feldolgozandó blokkok száma.
- `EDX` – vezérlőszó címe.

Alapesetben minden címet 16 bájtos határra kell igazítani.

ESI/EDI – A forrás- és céladatok címei

A forrás- és a célcím akár azonos is lehet, vagyis a titkosítást helyben is végezhetjük. A célpuffernek legalább akkorának kell lennie, mint a forrásnak. Mindkettő méretének a blokkméret (16 bájt) többszörösének kell lennie. Egyes esetekben az ***Esther*** processzor az igazítás nélküli pufferek használatát is lehetővé teszi, de ilyenkor a művelet végrehajtása lassabb.

EAX – kezdeti értékadási vektorcím

A ***kezdeti értékadási vektor (initialization vector, IV)*** egyike a titkosítás eredményét meghatározó átadott értékeknek. Az ***IV*** mérete megegyezik a blokkmérettel, vagyis 16 bájt. A kezdeti értékadási vektorokról a megfelelő szakirodalomban lehet részletesebben olvasni.

EBX – titkosító kulcs címe

A titkosító kulcs mérete 128, 192 vagy 256 bit lehet. Az ***AES*** algoritmus belül úgynevezett kiterjesztett kulcsot használ, amit a titkosító kulcsból származtat. A 128 bites kulcsoknál a kiterjesztett kulcsot a ***PadLock*** számítja, a hosszabb kulcsoknál ezt nekünk kell megtennünk.

ECX – feldolgozandó blokkok száma

Az `xcrypt` utasításokat mindig a `rep` előtaggal használjuk, ami lehetővé teszi ismételt végrehajtásukat egészen addig, amíg az `ECX` regiszter értéke nulla nem lesz. Az `ECX` által tárolt érték az egyes blokkok titkosításakor vagy visszafejtésekor fokozatosan csökken.

EDX – vezérlőszó címe

A ***PadLock*** csak akkor tudja, hogy pontosan hogyan kell feldolgoznia az adatokat, ha az úgynevezett vezérlőszó adatszerkezetet feltöltjük a következő elemekkel:

- Algoritmus (`algo`) – csak az ***AES***-t választhatjuk.
- Kulcsméret (`ksize`) – a támogatott méretek egyike.
- Irány (`encdec`) – titkosítás vagy visszafejtés.
- Kulcselőállítás (`keygen`) – elkészítettük a kiterjesztett kulcsot, vagy a ***PadLocknak*** kell kiszámítania?
- Körök (`rounds`) – az algoritmus egyik belső értéke, jelentését lásd a későbbiekben vagy a ***PadLock*** leírásában.

C-ben egy `union` tökéletesen megfelel az adatszerkezet helyének lefoglalására, elemeit pedig egy bitmező segítségével könnyen meg tudjuk adni és el tudjuk érni:

```
union cword {
    uint8_t cword[16];
    struct {
        int rounds:4;
        int algo:3;
        int keygen:1;
        int interm:1;
        int encdec:1;
        int ksize:2;
    } b;
};
```

Assembler példa

Elméletből mindent tudunk, nézzünk egy valódi példát. Kezdjünk néhány tiszta assembler-sorral:

```
.comm   iv,16,16
.comm   key,16,16
.comm   data,16,16
.comm   cword,16,16
.text
cryptcbc:
    movl    $data, %esi  #; forráscím
    movl    %esi, %edi   #; cél
    movl    $iv, %eax    #; IV
    movl    $key, %ebx   #; titkosító kulcs
    movl    $cword, %edx #; vezérlőszó
    movl    $1, %ecx     #; blokkszám
    rep xcryptcbc
    ret
```

A fenti kódrészlet megadott titkosító kulccsal és kezdeti értékadási vektorral, a `cword` vezérlőszóban megadottak szerint titkosít egy adatblokkot. Megjegyzem, hogy mivel helyben titkosítunk, a forrás- és a célcím azonos. Mivel a `data` mező mérete egyetlen blokknyi, az `ECX` regiszter értékeként egyet adtunk meg.

C példa

Ha a ***PadLockot*** közvetlenül, ***C*** programból akarjuk használni, akkor megtehetjük például, hogy a ***PadLock*** eljárásokat külön assembler forrásfájlokba helyezzük, majd a fordítást különálló modulokba végezzük, végül összekapcsoljuk a bináris fájlt. Egyszerűbb azonban a ***GCC*** beépített assemblerét használni, és az utasításokat közvetlenül a ***C*** kódba illeszteni. A források között szerepel egy a beépített assemblerrel kapcsolatos oktatóanyagra mutató hivatkozás.

```
static inline void *
padlock_xcryptcbc(char *input, char *output,
    void *key, void *iv, void *control_word,
    int count)
{
    asm volatile ("xcryptcbc"
        : "+S"(input), "+D"(output), "+a"(iv)
        : "c"(count), "d"(control_word),
        ➥"b"(key));
    return iv;
}
```

A fenti kódrészlet a fordítót a megfelelő bemeneti, számláló és egyéb átadott értékek megfelelő regiszterekbe való betöltésére utasítja. Ezután sor kerül az `xcryptcbc` utasítás kiadására, majd az EAX regiszterben talált értékkel térünk vissza, mint az új kezdeti értékadási vektorra irányuló mutatóval. Ügyeljünk a vezérlőszó adatszerkezet helyes kitöltésére. A legjobb az, ha első lépésként töröljük az `union`-t, így elkerülhetjük a memóriában esetlegesen előforduló értékek használatát:

```
memset(&cword, 0, sizeof(cword));
```

Most egyenként töltsük fel a mezőket. A lista első eleme a `rounds`. Ez adja meg, hogy az ***AES*** algoritmus hányszor menjen végig a bemeneti blokkon. Az algoritmus minden körben a kiterjesztett kulcs egy egyedi részét használja. Ha a ***FIPS AES*** szabványát akarjuk követni, akkor 128 bites kulcsnál 10, 192 bitesnél 12, 256 bitesnél pedig 14 kört kérjünk. Ha a `key_size` változó a titkosító kulcs méretét bájtban adná meg, akkor a körök számát az alábbi képlettel kapnánk:

```
cword.b.rounds = 10 + (key_size - 16) / 4;
```

A következő mező az `algo`. Segítségével a későbbiekben más, az ***AES***-től eltérő algoritmusokat is választhatunk majd, ám jelenleg kizárólag az ***AES*** áll rendelkezésünkre. Értékét tehát hagyjuk nullán.
A `keygen` mezőt akkor állítsuk egyre, ha a kiterjesztett kulcsot magunk állítjuk elő. A nullás érték azt jelenti, hogy a ***PadLocknak*** magának kell azt előállítania; erre viszont csak 128 bites kulcsnál van lehetőség:

```
cword.b.keygen = (key_size > 16);
```

Az `interm` lehetővé teszi az algoritmus futtatásának egyes körei után kapott köztes értékek tárolását. Van egy olyan gyanúm, hogy a processzor tervezői ezt a mezőt a processzormag hibáinak felderítésére használták, alkalmazásokon belüli használatának nem látom értelmét.
A titkosítás és a visszafejtés megkülönböztetése az `encdec` bittel történik. A nulla titkosítást, az egy visszafejtést jelent. Végül a kulcsméretet a `ksize` két bittel adja meg:

```
cword.b.ksize = (key_size - 16) / 8;
```

Ennyi. A vezérlőszó előkészítése és a pufferek igazítása után meghívhatjuk a `padlock_xcryptcbc()`-t. Ha a gépben keringő elektronok is úgy akarják, néhány pillanat múlva megkapjuk a titkosított adatokat.

Összefoglalás

A ***PadLock*** leírása nyilvánosan elérhető a ***VIA*** webhelyén, ahol további a ***PadLock*** programozásával kapcsolatos anyagokat is találni. Saját honlapomon szerepel egy teljes, a ***PadLock*** segítségével egyetlen adatblokkot titkosító példaprogram. A források között további figyelemre érdemes hivatkozások is szerepelnek.

Linux Journal 2005. május, 133. szám

A cikk forrásai: ➲ ***www.linuxjournal.com/article/8137***

Michal Ludvig (michal@logix.cz) nemrég költözött Prágából a világ másik felére, Aucklandbe, ahol az Asterisk Ltd. vezető mérnöke. Feleségével és lányával mostanában Új-Zéland titkainak felfedezésével töltik szabadidejüket.

WLAN-ok védelme WPA és FreeRADIUS alkalmazásával – III. rész

Új, a korábbiaknál biztonságosabb vezeték nélküli hálózatunk üzembe helyezésének végső lépéseként fel kell készítenünk néhány nem linuxos ügyfelet az új szabvány kezelésére.

Előző két írásomban áttekintettem, hogy a ***WPA (Wi-Fi protected access, Wi-Fi védett elérés)*** segítségével hogyan védhetjük meg a ***vezeték nélküli (wireless)*** helyi hálózatokat, röviden ***WLAN***-okat a jogosulatlan hozzáférésektől és a lehallgatásoktól. Elkezdtem ismertetni, hogy a ***FreeRADIUS*** segítségével hogyan valósíthatjuk meg a ***WPA***-t saját ***WLAN***-unkon. Az eddigiek során a ***FreeRADIUS*** telepítéséről, a ***hitelesítő szervezet (certificate authority, CA)*** létrehozásáról, valamint a ***WPA***-val történő használatra szánt digitális tanúsítványok előállításáról és aláírásáról volt szó. Ebben a hónapban megnézzük, hogy hova kell elhelyezni ezeket a tanúsítványokat, hogyan kell beállítani a ***FreeRADIUS***-t, a vezeték nélküli hozzáférési pontot és az ügyfeleket. Mindezek alapján, úgy vélem, bárki nekiláthat saját ***WLAN***-ja biztonságának megerősítéséhez.

Rövid ismétlés

Azok számára, akik csak most kapcsolódnak be a cikksorozatba, esetleg fel kell frissíteniük az emlékezetüket, hogy pontosan mit is próbálunk elérni, röviden tekintsük át céljainkat és lehetőségeinket. A ***WPA*** erőteljes hitelesítési eljárásokkal bővíti a régebbi, kriptográfiailag megtört ***WEP*** protokollt; teszi ezt a ***802.1x*** protokoll és alprotokolljai révén, mint az ***EAP***, a ***PEAP*** és az ***EAP-TLS***. A ***WPA*** a ***TKIP*** protokoll révén képes a munkamenetkulcsok dinamikus egyeztetésére és a kulcsok önműködő megújítására is. Ha vezeték nélküli ügyfelünk támogatja a ***WPA***-t – vagyis rendelkezik ***WPA*** kérvényezővel –, továbbá a vezeték nélküli hozzáférési pontunk is rendelkezik ilyen képességgel, akkor kétharmad részben már sikerrel jártunk. Ha viszont teljes mértékben ki akarjuk használni a ***802.1x*** által kínált lehetőségeket, akkor szükségünk lesz egy ***RADIUS*** háttérkiszolgálóra is – itt lép be a képbe a ***FreeRADIUS***.
A múlt alkalommal kialakított példakörnyezetben egy ***FreeRADIUS*** kiszolgálót helyezünk üzembe, amelynek feladatául a tetszőleges ***WPA***-képes vezeték nélküli hozzáférési ponthoz csatlakozó ***Windows XP*** ügyfelek hitelesítését tűztük ki. ***802.1x*** eljárásunk az ***EAP-TLS*** lett. Az ***EAP-TLS***, mint talán néhányan még emlékeznek rá, a ***TLS*** protokollt használja a vezeték nélküli kérvényezők (ügyfelek) és a hozzáférési pontok kölcsönös hitelesítésére; illetve mindehhez ***X.509*** digitális tanúsítványokat alkalmaz.
Még hátralévő feladataink a következők:

- A múlt alkalommal létrehozott kiszolgáló és ***CA*** tanúsítványok telepítése a ***FreeRADIUS*** kiszolgálóra
- A ***FreeRADIUS*** beállítása ezeknek a tanúsítványoknak a használatára, ***EAP-TLS*** felett, a hozzáférési pont felhasználóinak hitelesítése céljából
- A hozzáférési pont beállítása a hitelesítés átirányítására a ***FreeRADIUS*** kiszolgálóra
- Az ügyfélprogram és a múlt alkalommal létrehozott ***CA*** tanúsítványok telepítése egy ***Windows XP*** alapú ügyfélre, illetve beállítása úgy, hogy a ***WLAN***-hoz történő csatlakozáskor ***WPA***-t használjon.

A FreeRADIUS kiszolgáló előkészítése

A ***WPA***-ról szóló sorozat második részében létrehoztunk három ***X.509*** digitális tanúsítványt: a ***hitelesítő szervezetét*** (***cacert.pem***), egy ***kiszolgálótanúsítványt*** (***kiszolgalo_kulcstanusitvany.pem***) és egy ***ügyféltanúsítványt*** (***ugyfel_tanusitvany.p12***). A kiszolgáló és az ügyfél fájlja a tanúsítványt és annak titkos kulcsát egyaránt tartalmazza, ezért mindkettő telepítését kellő körültekintéssel kell elvégezni. A ***CA*** tanúsítványt ugyanakkor a kulcstól elkülönítve tároljuk, vagyis a ***cacert.pem***-et szabadon terjeszthetjük.
A ***FreeRADIUS*** beállító fájljai a ***/etc/raddb/*** vagy a ***/usr/local/etc/raddb/*** könyvtárban találhatók, terjesztéstől függően. A könyvtárnak van egy ***certs/*** nevű alkönyvtára, ide kell bemásolnunk a ***CA*** tanúsítványát, valamint a kiszolgálótanúsítványt és kulcsát. Ellenőrizzük, hogy a cacert.pem tulajdonosa a root felhasználó-e, a fájlra vonatkozó engedélyek pedig a következők legyenek:

```
-r--r--r--
```

1. kódrészlet A raddb/certs könyvtárban található tanúsítványokra megadott tulajdonosok és engedélyek

```
-r--r--r--  1 root   users 1294 2005-02-10 01:05
➥cacert.pem
-r--------  1 nobody users 1894 2005-02-10 01:00
➥kiszolgalo_kulcstanusitvany.pem
```

2. kódrészlet A radiusd.conf két módosítandó beállítása

```
user = nobody
group = nobody
```

A ***kiszolgalo_kulcstanusitvany.pem*** fájlnak ugyanakkor a ***nobody*** felhasználó tulajdonába kell tartoznia, `-r--------` engedéllyel. Az *1. kódrészlet* ennek a két fájlnak a hosszú könyvtárlistáját szemlélteti.
Ha már a fájlok tulajdonosaival foglalkozunk, ellenőrizzük, hogy a */var/log/radius/radius.log* fájlra és a */var/run/radiusd/* könyvtárra van-e írási engedélye a ***nobody*** felhasználónak. Ha a ***FreeRADIUS***-t forrásból fordítottuk, akkor lehetséges, hogy az előbbiek helyett a */usr/local/var/log/radius/radius.log* fájllal és a */usr/local/var/srun/radiusd/* könyvtárral kell dolgoznunk. A ***radius.log*** és a ***radiusd/*** tulajdonosa egyaránt a ***nobody*** legyen.
Mielőtt beleásnánk magunkat a ***FreeRADIUS*** beállító fájljaiba, létre kell hoznunk további két, a ***FreeRADIUS*** által a ***TLS*** használatához igényelt fájlt. Az első a ***Diffie-Hellman (dh)*** átadott értékeket tartalmazza, ezekre a ***TLS*** munkamenet-kulcsok egyeztetésekor van szükség. A ***dh*** fájlt úgy hozhatjuk létre, hogy átváltunk a ***FreeRADIUS raddb/certs/*** könyvtárába, majd kiadjuk a következő parancsot:

```
# openssl dhparam -check -text -5 512 -out dh
```

A második fájl egy véletlenszerű bitfolyamot tartalmazó, szintén a ***TLS*** műveleteknél szükséges adatfájl, ***random*** névvel. Itt hívnám fel rá a figyelmet, hogy a véletlenszerűnek szánt tartalmak létrehozását ***nem szabad*** egyszerűen az aktuális időbélyeg vagy valamilyen hasonló, a legkevésbé sem véletlenszerű karakterlánc beírásával letudni; még akkor sem, ha a ***WPA***-val kapcsolatos, az interneten fellelhető leírások némelyikében ezt javasolják. Ehelyett a rendszermag kiváló minőségű véletlenszám-előállítóját kell használni. A ***raddb/certs*** könyvtárból tehát a következő parancsot adjuk ki:

```
# dd if=/dev/urandom of=random count=2
```

Mindkét fájlra olvasási jogot kell adni a ***nobody*** felhasználónak, az írási jogot pedig mindenkitől meg kell vonni.

A FreeRADIUS beállítása

Végre készen állunk a ***FreeRADIUS*** beállításainak megadására. Elég ijesztő lehet a */raddb* könyvtárban sorakozó fájlok sokasága, de riadalomra semmi ok. Az ***EAP-TLS*** alapú ***WPA*** használatához csak a következő három fájlt kell módosítanunk: ***radiusd.conf***, ***eap.conf*** és ***clients.conf***.
A ***radiusd.conf*** fájlban csupán azt kell megadnunk, hogy a ***radiusd*** folyamat melyik felhasználó és csoport jogaival fusson. A jogok alapesetben a démont indító felhasználótól öröklődnek. Ha a ***radiusd***-t parancsfájlból indítjuk, akkor a root jogait örökli – nyilván ezt el szeretnénk kerül. Felhasználóként és csoportként tehát a ***radiusd.conf***-ban egyaránt ***nobody***-t adjunk meg, ahogy ezt a *2. kódrészlet* is szemlélteti. Természetesen a ***nobody/nobody*** helyett más, kiemelt jogokkal nem rendelkező felhasználót és csoportot is választhatunk, ám ha így teszünk, akkor a korábban említett tanúsítványfájlok tulajdonosait és a vonatkozó jogokat is módosítanunk kell. Bárhogy is döntünk, ellenőrizzük, hogy a kiválasztott felhasználóhoz a */etc/password* fájlban tartozó bejegyzés szerint a felhasználó nem kaphat héjhozzáférést (a bejegyzés például */bin/false* vagy */bin/true* lehet); a fiókot ***SSH***, ***telnet*** és hasonló programok nem használhatják. Mondanom sem kell, azt sem árt ellenőrizni, hogy a felhasználó és a csoport egyáltalán létezik-e, és ha nem, létre kell hozni őket.
A ***radiusd.conf*** további beállításokat is tartalmaz, ám csak a fenti kettő módosítása az, ami igazán lényeges. További tudnivalókat a ***radiusd.conf(5) man*** oldalon vagy ***Jonathan Hassell RADIUS*** című könyvében találni.
A következő átírandó fájl az ***eap.conf*** – itt merülünk a sűrűjébe. A *3. kódrészlet* az ***eap.conf*** módosítandó sorait tartalmazza. A 3. kódrészletben a `private_key_password` átadott értékkel megadtam egy kiszolgáló-kulcs jelszót. Ez valójában üres, feltéve, hogy a kiszolgáló tanúsítványát és kulcsát az ***OpenSSL*** `-nodes` kapcsolójával hoztuk létre. Sajnos a múlt hónapban magam is ezt javasoltam, ám most, talán még időben, szeretném ezt visszavonni. A jelszó nélküli ***X.509*** kulcsok használata ront a biztonságon, még ha a kulcsot nyílt szöveges beállító fájlban tároljuk is, mint például az ***eap.conf***. Bizony, ha a ***FreeRADIUS*** kiszolgálón egy behatoló root jogokhoz jut, akkor – köszönhetően az ***eap.conf*** fájlnak – még egy jelszóval védett tanúsítvány bizalmassága is sérülhet. Ha viszont a tanúsítványt/kulcsot út közben – például, amikor a ***CA*** állomásról a ***FreeRADIUS*** kiszolgálóra másoljuk – hallgatják le, akkor, ha jelszóval védett, a támadó nem tud mit kezdeni vele.
Bármelyik megoldást is válasszuk, ellenőrizzük, hogy az ***eap.conf*** a root tulajdonában van-e, illetve csak ő rendelkezik-e hozzá írási joggal, és nem a ***radiusd.conf*** fájlban megadott felhasználó. Furcsa, igaz? A ***nobody***-nak nem kellene jogot adni a beállító fájlok olvasására? A válasz nem, hiszen ha a ***radiusd***-t rootként indítjuk, akkor először beolvassa a beállító fájlokat (***radiusd.conf***, ***eap.conf*** és ***clients.conf***), csak ezután vált át a ***nobody*** jogosultságaira.
Végül létre kell hoznunk egy bejegyzést a hozzáférési pont számára a clients.conf fájlban. A *4. kódrészlet* erre mutat példát.
A 4. kódrészletben a `client` (ügyfél) utasítás adja meg a hozzáférési pont IP-címét. A hozzá tartozó `secret` (titok) átadott érték adja meg azt a karakterláncot, amelyet a hozzáférési pont a ***FreeRADIUS*** kiszolgálónak küldött kérésekben titkosítási kulcsként használ. A `shortname` (rövid név) egyszerűen csak egy álnév a hozzáférési ponthoz, például a naplóbejegyzésekben találkozhatunk vele.

3. kódrészlet Az eap.conf fájl módosításai

```
eap {
  # Itt számos általános EAP átadott érték
  # megadására van lehetőség,
  # ám számunkra most csak
  # a default_eap_type fontos:
  default_eap_type = tls
  # Ezután következnek az egyes
  # EAP-típusok beállításai.
  # Mivel EAP-TLS-t akarunk használni,
  # csak a tls{} szakasszal
  # kell foglalkoznunk:
  tls {
    # Az alábbi értékek azt adják meg
    # a radiusd-nek, hogy hol
    # találja a tanúsítványokat és a kulcsokat,
    # illetve a dh és a random fájlokat:
    private_key_password =
    ➥ide_jon_a_kulcs_jelszava
    private_key_file =
    ➥${raddbdir}/certs/bt_keycert.pem
    certificate_file =
    ➥${raddbdir}/certs/bt_keycert.pem
    CA_file = ${raddbdir}/certs/cacert.pem
    dh_file = ${raddbdir}/certs/dh
    random_file = ${raddbdir}/certs/random
    }
  }
```

4. kódrészlet Hozzáférési ponthoz tartozó bejegyzés a clients.conf fájlban

```
client 10.1.2.3/32 {
      secret          = felhasznalojelszo
      shortname       = hozzaferesi_pont
}
```

A ***radiusd*** készen áll az indításra, amit az ***rc.radiusd*** parancsfájllal tehetünk meg:

```
rc.radiusd start
```

Az újraindítás az `rc.radiusd restart` paranccsal történik. Ha a ***radiusd*** hiba nélkül elindult, továbbléphetünk.

A hozzáférési pont beállítása

A következő lépés az egész folyamat legkönnyebb része: a vezeték nélküli hozzáférési pont beállítása ***WPA*** használatára és a ***FreeRADIUS*** kiszolgáló címének megadása. Mindehhez csupán kétféle adatra van szükség, a ***FreeRADIUS*** kiszolgáló ***clients.conf*** fájljában megadott ***RADIUS titokra*** (***secret*** átadott érték), valamint a kiszolgáló ***IP***-címére. Az, hogy ezeket az adatokat ténylegesen hogyan kell megadni a hozzáférési pontnak, az alkalmazott eszköztől és a rajta futó szoftvertől függ. Az én hozzáférési pontom egy ***WLAN***-hozzáférés biztosítására is képes ***Actiontec DSL*** forgalomirányító. Ennek webes felületén a ***Setup›Advanced Setup›Wireless Settings (Beállítások›Speciális beállítások›Vezeték nélküli beállítások)*** pontra kattintottam, majd a ***Security (Biztonság)*** beállításnál a ***WPA***-t választottam. Ezután előre megosztott kulcs helyett átállítottam ***802.1x*** használatára. Kellett adnom neki egy kiszolgálócímet, ez 10.1.2.3 lett, továbbá be kellett írnom a ***FreeRADIUS*** kiszolgáló ***IP***-címét, valamint a 4. kódrészletben már látott titkot (felhasznalojelszo). A kapuszámot az alapértelmezett 1812-n hagytam.
Ha már szóba került a téma: ha a hozzáférési pontot és a ***RADIUS*** kiszolgálót tűzfal választja el egymástól, akkor lehetővé kell tennünk, hogy a hozzáférési pont elérhesse a ***RADIUS*** kiszolgáló 1812-es és 1813-as kapuját. Ekkor egyben a ***RADIUS*** kiszolgáló is módot kap válaszainak ezeken a kapukon keresztül történő továbbítására.

A Windows XP alapú ügyfelek beállítása

Végre elérkeztünk oda, hogy a ***Windows XP*** alapú, vezeték nélküli ügyfeleket beállíthassuk a ***WPA*** használatára képessé tett hozzáférési ponthoz való csatlakozásra. Tudom, hogy linuxos magazinba írok, ezért nem akarok túlságosan sokat rágódni a témán, akit részletesebben is érdekel, az olvassa el ***Ken Roser HOGYAN***-jának 4.3-as szakaszát (lásd a forrásokat). Teendőink röviden:

1. ***Start›Futtatás (Start›Run)***, majd az `mmc` parancs futtatása.

2. A ***Microsoft Management Console***-ban válasszuk a ***Fájl›Beépülő modul hozzáadása/eltávolítása (File›Add/Remove Snap-in)*** parancsot, válasszuk ki a ***Tanúsítványok (Certificates)*** beépülő modult, majd válasszuk a saját fiókunkhoz tartozó tanúsítványok helyi gépre vonatkozó kezelését.

3. Másoljuk át ***CA*** tanúsítványunkat (***cacert.pem***) a windowsos rendszer merevlemezére, például ***C:\cacert.pem*** névvel.

4. Az ***MMC***-ben bontsuk ki a ***Kezelőpultgyökér (Console Root)*** és a ***Tanúsítványok - Aktuális felhasználó (Certificates - Current User)*** csomópontot, majd kattintsunk az egér jobb gombjával a ***Megbízható legfelső szintű hitelesítésszolgáltatók (Trusted Root Certification Authorities)*** elemre. A helyi menüből válasszuk az ***Összes feladat›Importálás (All Tasks›Import)*** parancsot. A varázslóban válasszuk ki a ***C:\cacert.pem*** fájlt, majd mentsük el a megbízható legfelső szintű hitelesítésszolgáltató alá.

5. Másoljuk át az ügyfél tanúsítvány/kulcs fájlját a windowsos rendszerre, például ***C:\ugyfel_tanusitvany.p12*** névvel.

6. A kezelőpulton kattintsunk az egér jobb gombjával a ***Tanúsítványok (Certificates)*** csomópont ***Személyes (Personal)*** ágára. A helyi menüből válasszuk az ***Összes feladat›Importálás (All Tasks›Import)*** parancsot. A megjelenő varázslóban importáljuk be a ***C:\ugyfel_tanusitvany.p12*** fájlt.

7. Az importáló varázsló bekéri tőlünk a tanúsítvány jelszavát, illetve ugyanezen a párbeszédpanelen felkínálja a titkos kulcs erőteljes védelmét is. Sajnos ennek az engedélyezése megakadályozza a *WPA* működését, vagyis a használatától el kell tekintenünk. A kulcs exportálását engedélyező négyzetet is hagyjuk érintetlenül, jobban járunk ugyanis, ha a jelszóval védett fájlról készítünk biztonsági mentést, mintha az importált, jelszóval nem védett változat exportálását engedélyeznénk.

8. A következő képernyőn hagyjuk, hogy a varázsló magától kiválassza a tanúsítványtárolót.

Ezzel a *Windows XP* alapú rendszer készen áll, már csak egy vezeték nélküli hálózati profilt kell létrehoznunk. Ennek módja a vezeték nélküli kártya illesztőprogramjaitól és attól függően változik, hogy a *Windows XP* melyik szervizcsomagját telepítettük. Nálam *Windows XP SP1* fut, *Centrino* lapkakészleten, és az *XP* saját *WPA* kérvényezőjével hoztam létre vezeték nélküli hálózati profilt, megadva saját *WLAN*-om *SSID*-jét. *Hálózati hitelesítésre (Network Authentication) WPA*-t, *adattitkosításra (Data encryption) TKIP*-t választottam, az *EAP típusa (EAP type)* pedig *intelligens kártya vagy más tanúsítvány (Smart Card or other Certificate)* lett. A *Windows* magától felismerte, hogy milyen ügyféltanúsítványt akarok használni – ez annak köszönhető, hogy a múlt alkalommal külön lépésekkel gondoskodtunk arról, hogy az ügyféltanúsítványunk tartalmazza a *Windows XP* kiterjesztett jellemzőit.

Miután megtörtént a vezeték nélkül hálózati profil összeállítása, a *Windowsnak* önműködően kapcsolódnia kell a hozzáférési ponthoz, és végre kell hajtania a *WPA*-kapcsolat egyeztetését. Amennyiben ez sikerrel jár, a *Hálózati kapcsolatok (Network Connections)* között olyan jelzésnek kell megjelennie, mely szerint a vezeték nélküli hálózati kapcsolat hitelesítése sikeresen megtörtént.

Összefoglalás

Messzire jutottunk, remélem, mindenki követni tudott, és a továbbiakban senki számára nem okoz gondot a *WPA* használata. Bár a *WPA* távolról sem tökéletes – valójában a jelszóval védett ügyféltanúsítványoknak a jelszavak nyílt szövegben való tárolása nélküli kezelésére is képes *WPA* kérvényezőkre volna szükség –, azért elmondhatjuk, hogy a vezeték nélküli hálózatok végre elindultak a biztonság irányába.

Linux Journal 2005. június, 134. szám

A cikkhez tartozó források elérhetősége:
➲ *www.linuxjournal.com/article/8200*

Mick Bauer (mick@visi.com)
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban található Upstream Solutions LLC Inc.-nél.

Dinamikusan előállított naptárak

Jó lenne, ha weboldalunk figyelmeztetni tudná a felhasználókat a közeljövő eseményeire, esetleg az egész cégnek közös, szinkronban tartott naptárat biztosítana? Python alatt iCalendar fájlokat előállítva mindez valósággá válhat.

A múlt alkalommal a ***Sunbirddel***, a ***Mozilla Foundation*** naptárkövetésre alkalmas önálló alkalmazásával ismerkedtünk meg. Mint láttuk, a ***Sunbird*** képes az ***iCalendar*** formátumú naptárak kezelésére. A naptárak a helyi fájlrendszerben is lehetnek, de ***HTTP***-n keresztül távoli kiszolgálóról is lekérhetők. Láttuk azt is, hogy a ***Sunbirddel*** milyen egyszerű egy távoli kiszolgálón található naptárat használni. Egyszerűen be kell írnunk a megfelelő ***URL***-t egy párbeszédpanelen, majd miután a ***Sunbird*** letöltötte az ***iCalendar*** fájlt, az új események máris megjelennek a képernyőnkön. Az interneten már jelenleg is számos különböző távoli, ***iCalendar*** formátumú naptárat találunk, megkeresésük és előfizetésük viszonylag egyszerűen letudható. Persze mindennek csak akkor van haszna, ha már meglévő, a nyilvánosság számára elérhetővé tett naptárat szeretnénk igénybe venni. De mi legyen, ha szervezetünk az ***iCalendar*** formátumra akarja alapozni a belső adatcserét? Hogyan tudunk ***iCalendar*** fájlokat létrehozni és terjeszteni, lehetővé téve, hogy mások is értesülhessenek az őket érintő eseményekről? Ebben a hónapban az ***iCalendar*** fájlokkal kapcsolatos, a kiszolgáló oldalra vonatkozó tudnivalókat tekintjük át, valamint naptáralkalmazások – mint a ***Sunbird*** – általi letöltésre, szervezeten belüli használatra szánt naptárakat fogunk készíteni.

iCalendar fájlok

Ha két számítógép között naptárakat akarunk cserélni, nyilván szükségünk lesz egy szabványra, mely meghatározza, hogy ezeket a naptárakat hogyan kell formázni. A továbbításra használt protokoll jelenleg nincs meghatározva, ám a szabványok és a napi gyakorlat egyaránt arra utal, hogy a ***HTTP*** csaknem kizárólagos szerepet nyert a hasonló átvitelek kezelésében. A naptárcsere formátuma, mely az ***RFC 2445*** dokumentumban található, tükrözi korának színvonalát. Míg egy új naptárformátum kétségtelenül ***XML***-re épülne, az 1998-ban született ***RFC*** még név-érték párokat alkalmaz, az elemeket egyszerű hierarchiába rendezve. Példaként vegyük elő a múlt hónapban, a ***Sunbird*** megismerésekor is látott ***iCalendar*** fájlt:

```
BEGIN:VCALENDAR
VERSION
 :2.0
PRODID
 :-//Mozilla.org/NONSGML Mozilla Calendar V1.0//EN
BEGIN:VEVENT
UID
 :05e55cc2-1dd2-11b2-8818-f578cbb4b77d
SUMMARY
 :Linuxvilág határidő
STATUS
 :TENTATIVE
CLASS
 :PRIVATE
X-MOZILLA-ALARM-DEFAULT-LENGTH
 :0
DTSTART
 :20050211T140000
DTEND
 :20050211T150000
DTSTAMP
 :20050209T132231Z
END:VEVENT
END:VCALENDAR
```

Mint látható, a fájl a `BEGIN:VCALENDAR` címkével kezdődik és az `END:VCALENDAR` címkével végződik. A fájl tetején néhány a teljes naptárra érvényes adat található, ilyen a `VERSION` és a `PRODID`, alattuk azonban már kezdődik is az első és egyben egyetlen esemény, melyet a `BEGIN:VEVENT` és a `END:VEVENT` címke fog közre. Nyilván nem okoz nehézséget elképzelni, hogy nagyobb számú eseményt hogyan tárolna a fájl.
Az ***iCalendar*** rendszeres időközönként megismétlődő események megadását is lehetővé teszi. Egyetlen `VEVENT` bejegyzés is elég tehát ahhoz, hogy jelezzünk egy minden hét hétfő délutánján megtartott értekezletet vagy figyelmeztessük magunkat arra, hogy kedden és pénteken ki kell tennünk a kukát. Minden eseménynek van kezdő és záró időpontja, ez a `DTSTART` és a `DTEND`, az események hossza gyakorlatilag tetszőleges.
Bár a fenti példából nem tűnik ki, az ***iCalendar*** az ismétlődő eseményeknél kivételek megadását is lehetővé teszi. Ha például a hétfő délutáni értekezletre nyaralás közben nem akarunk elmenni, akkor egy `EXDATE` bejegyzéssel jelezhet-

1. kódrészlet static-calendar.py, egyszerű, Pythonban készült CGI program iCalendar fájl megnyitására és HTTP feletti elküldésére

```
#!/usr/bin/python
# A CGI modul beemelése
import cgi
# Az esetleges hibák naplózása
import cgitb
cgitb.enable(display=0, logdir="/tmp")
# Hol van a naptárfájl?
calendar_directory =
'/usr/local/apache2/calendars/'
calendar_file = calendar_directory + 'test.ics'
# content-type fejrész küldése a felhasználó
böngészőjének
print "Content-type: text/calendar\n\n"
# A fájl tartalmának elküldése a böngészőnek
calendar_filehandle = open(calendar_file, "rb")
print calendar_filehandle.read()
calendar_filehandle.close()
```

jük távol maradásunkat. A naptárat megjelenítő alkalmazás ezt követően figyelmen kívül hagyja az adott dátumra eső ismétlődő eseményt.

iCalendar fájlok közzététele

Ha már van ***iCalendar*** fájl a rendszerünkön, ennek közzététele rendkívül egyszerű. Az ***1. kódrészlet*** egy egyszerű, ***Pythonban*** készült ***CGI*** programot tartalmaz, mely megkeres egy ***iCalendar*** fájlt a megadott könyvtárban, majd visszaadja a fájl tartalmát a kérést intéző naptáralkalmazásnak. Aki még nem írt ***CGI*** programot ***Pythonban***, az a fenti példa alapján talán már rájött, mennyire egyszerű is az egész. Az alapvető ***CGI*** szolgáltatások eléréséhez be kell tölteni a ***CGI*** modult. A következő lépés a ***cgitb***, a ***CGI*** visszakövető modul betöltése, amellyel hiba esetén hibakereső adatokat tudunk írni egy fájlba.
A következő a ***text/calendar*** tartalomtípus (***content-type***) fejrész elküldése. Feltételezhetjük, hogy a webes tartalmak túlnyomó részének tartalomtípusa ***text/html*** (HTML formázású szöveg) vagy ***text/plain*** (egyszerű szöveges fájl), amihez különféle ***image/jpeg***, ***image/png*** és ***image/gif*** elemek társulnak. Az ***iCalendar*** szabvány szerint a naptárfájlokhoz ***text/calendar*** típust kell rendelni, még akkor is, ha a ***Sunbird*** és a hozzá hasonló programok a ***text/plain*** formátumot is elfogadják. A program utolsó lépése a naptárfájl tartalmának elküldése, amelyet a helyi fájlrendszerből olvas ki.
Aki még egyáltalán nem foglalkozott webes programozással, annak alighanem tökéletesen értelmetlennek tűnik a fenti példa. Például teljesen eszement dolognak tűnik az, hogy külön programot írunk egy állandó fájl tartalmának elérésére – csakhogy így megtehetjük, hogy a külvilág felől elfedjük a naptárfájl valódi helyét. Persze vannak erre jobb megoldások is, például az ***Apache Alias*** utasításának használata. A programot továbbfejleszthetnénk úgy, hogy a naptárfájl nevét átadott értékként közöljük vele, ám ilyenkor is szükségünk lenne állandó jelleggel rendelkezésre álló, előre elkészített fájlokra.

iCalendar létrehozása

A valódi megoldás, ami persze érdekesebb is, az, hogy az ***iCalendar*** fájlt menet közben, a felhasználó kérésének hatására hozzuk létre. A ***CGI*** program tehát nem egy meglévő ***iCalendar*** fájl tartalmát adja vissza, hanem maga állítja azt elő, majd átadja a felhasználó naptár-ügyfélprogramjának. Első ránézésre ennek megvalósítása egyszerűnek látszik. Végül is, az ***iCalendar*** fájlformátum egyszerűnek tűnik, valószínűleg nem okoz gondot a program összeállítása. Ha viszont jobban megvizsgáljuk, rájövünk, hogy az ***iCalendar*** fájlt létrehozni sokkal nehezebb, mint beszélni róla, különösen, ha ismétlődő eseményeket is meg akarunk adni. Figyelembe véve az ***iCalendar*** szabvány növekvő népszerűségét és a nyílt forrású tervezetek megszámlálhatatlan sokaságát, meglepve tapasztaltam, hogy az ***iCalendar*** a legnagyobb nyílt programozói közösségek részéről is csak minimális figyelmet érdemelt ki. Meglepetésem annak is volt köszönhető, hogy az ***iCalendar*** évek óta létezik, számtalan vállalat és naptárprogram támogatja, kezdve a ***Novell Evolutiontől***, a ***Lotus Noteson*** keresztül egészen a ***Microsoft Outlookig***. Az ilyen nagymértékű támogatottság általában sokféle nyelvre kiterjedő, gazdag választékot eredményez. Először a ***Perl*** környékén tapogatóztam; a ***CPAN*** archívum számos moduljának, köztük a különféle internetes szabványokkal kapcsolatosaknak köszönhetően méltán szerzett elismerést. Furcsa, hogy bár ***iCalendar*** fájlok feldolgozásához több ***Perl*** modul közül is válogathatunk, létrehozásukhoz egyetlen naprakész modul sem létezik. A ***Net::ICal::Libical*** például egy burkoló lett volna a ***C***-ben készült ***libical*** könyvtárhoz, ám utolsó kiadása egy alfa változat előtti csomag. A ***Net::ICal*** a ***ReefKnot*** tervezet része volt, ami a jelek szerint szintén félbemaradt.
Szerencsére egy dán fejlesztő, ***Max M*** (lásd az internetes forrásokat) nemrég úgy döntött, hogy pótolja a hiányosságokat, és készített egy ***iCalendar*** fájlok egyszerű létrehozására használható ***Python*** csomagot. Én minden gond nélkül le tudtam tölteni és telepíteni tudtam a csomagot a gépemre, és jómagam is úgy találtam, hogy rendkívül könnyű vele naptárat készíteni. Korábbi egyszerű kis ***CGI*** programunkkal együtt alkalmazva kiválóan megfelel naptárak létrehozására és közzétételére.

Dinamikus naptár létrehozása

A ***maxm.dk*** oldalról letöltöttem az ***iCalendar*** csomagot, majd telepítettem. Sok korszerű ***Python*** csomaggal ellentétben telepítése nem önműködő, vagyis kézzel kell bemásolnunk rendszerünk ***site-packages*** könyvtárába, ami az én ***Fedora Core 3*** alapú gépemen a ***/usr/lib/python-2.3/site-packages*** volt. Amint a ***2. kódrészletből*** is látható, az újonnan telepített ***iCalendar*** csomaggal ***Calendar*** (naptár) és Event (esemény) típusú objektumokat hoztam létre. Az első teendőm a megfelelő csomagok beemelése volt az aktuális névtérbe:

```
from iCalendar import Calendar, Event
```

Az ***iCalendar*** csomag ***Calendar*** és ***Event*** modulja rendre a teljes ***iCalendar*** fájlhoz és azon belül egy-egy eseményhez tartozik. A ***Calendar*** objektumból tehát egy példányra van szükségünk, míg minden eseményhez egy-egy ***Event*** objektumnak kell tartoznia.

2. kódrészlet dynamic-calendar.py, iCalendar formátumú naptárat előállító program

```
#!/usr/bin/python
# A CGI modul beemelése
import cgi
from iCalendar import Calendar, Event
from datetime import datetime
from iCalendar import UTC # időzóna
# Az esetleges hibák naplózása
import cgitb
cgitb.enable(display=0, logdir="/tmp")
# content-type fejrész küldése a felhasználó
# böngészőjének
print "Content-type: text/calendar\n\n"
# Naptárobjektum létrehozása
cal = Calendar()
# Milyen termék hozta létre a naptárat?
cal.add('prodid',
        '-//Python iCalendar 0.9.3//mxm.dk//')
# Az RFC 2445-nek a 2.0-s változat felel meg
cal.add('version', '2.0')
# Esemény létrehozása
event = Event()
event.add('summary', 'ATF határidő')
event.add('dtstart',

datetime(2005,3,11,8,0,0,tzinfo=UTC()))
event.add('dtend',

datetime(2005,3,11,10,0,0,tzinfo=UTC()))
event.add('dtstamp',

datetime(2005,3,11,0,10,0,tzinfo=UTC()))
event['uid'] = 'ATF20050311A@lerner.co.il'
# Kapjon kiemelt fontosságot!
event.add('priority', 5)
# Az esemény hozzáadása a naptárhoz
cal.add_component(event)
# A naptár utasítása önmaga leképezésére
iCalendar
# fájlként, majd a fájl átadása a HTTP válaszban.
print cal.as_string()
```

Ezután megalkothatjuk a `Calendar` objektumot:

```
cal = Calendar()
cal.add('prodid',
        '-//Python iCalendar 0.9.3//mxm.dk//')
cal.add('version', '2.0')
```

A második és a harmadik sorban, ahol a `cal.add()` eljárást hívjuk meg, módunk nyílik azonosító adatok hozzáadására az ***iCalendar*** fájlhoz. Az elsővel megadhatjuk az ügyfélprogramnak, hogy milyen alkalmazás hozta létre az ***iCalendar*** fájlt. Ez főleg hibakeresésnél hasznos; ha rendszeresen hibás ***iCalendar*** fájlokat kapunk egy megadott programcsomagtól, akkor kapcsolatba tudunk lépni a készítővel vagy a terjesztővel, és jelenthetjük neki a hibát. A második sorban a változatazonosítót adtuk hozzá, amivel azt jelezzük, hogy az ***iCalendar*** előírások melyik változatát követjük. Az ***RFC*** 2445 értelmében ennek a mezőnek 2.0 értéket kell adnunk – már amennyiben követni kívánjuk a dokumentum előírásait.

Most, hogy megvan a naptárunk, adjunk hozzá egy eseményt, amelyet egészítsünk ki egy összegzés sorral; utóbbi az ***iCalendar*** fájlra előfizető személyek naptáralkalmazásában fog megjelenni:

```
event = Event()
event.add('summary', 'ATF határidő')
```

Mint a példafájlnál is láttuk, minden eseményhez három dátum/idő mező tartozik: a kezdés dátuma és időpontja (***dtstart***), a befejezés dátuma és időpontja (***dtend***) és az az időpont, amikor a bejegyzés bekerült a naptárba (***dtstamp***). Az ***iCalendar*** szabvány egy meglehetősen furcsa formátum szerint tárolja a dátumokat és az időpontokat, ám az Event objektum tudja ezeket kezelni, ha ő maga ***datetime*** objektumot kap a normál datetime Python csomagtól. Tehát:

```
event.add('dtstart',
          datetime(2005,3,11,14,0,0,tzinfo=UTC()))
event.add('dtend',
          datetime(2005,3,11,16,0,0,tzinfo=UTC()))
event.add('dtstamp',
          datetime(2005,3,11,0,10,0,tzinfo=UTC()))
```

Megjegyezném, hogy időzónaként mindhárom esetben ***UTC***-t adtam meg. Amikor az ügyfélalkalmazásban megjelenik az ***iCalendar*** fájl tartalma, akkor természetesen a felhasználó a saját időzónája szerint látja a bejegyzéseket, és nem ***UTC*** szerint.

Az eseményeknek létrehozásuk után egyedi azonosítót kell adni. Az egyediség alatt itt valódi, a világ minden számítógépének minden naptárára kiterjedő egyediséget kell érteni. Ez elég fogós dolognak hangzik, de nem kell megijedni tőle. Sokféle megoldás létezik, az egyik az, hogy a létrehozás időbélyegét, az esemény létrehozására használt számítógép ***IP***-címét és egy nagy véletlenszerű számot kombinálunk. Én egy egyszerű ***UID*** előállítása mellett döntöttem, de ha olyan alkalmazást készítünk, amelyet több számítógépen is használni fognak, akkor érdemes alaposabban átgondolni és egységesíteni az azonosítók létrehozását:

```
event['uid'] = 'ATF20050311A@lerner.co.il'
```

A végső lépés az esemény fontosságának, prioritásának megadása, mely 0 és 9 között lehet. A normál, átlagos fontossági szint az ötös, a sürgősebb elemeknek nagyobb, a kevésbé sürgőseknek kisebb fontosságot kell adni:

```
event.add('priority', 5)
```

Az eseményt létrehozása után csatolni kell a naptár objektumhoz, mely jó ideje csak arra vár, hogy végre kezdjünk vele valamit:

```
cal.add_component(event)
```

Szükség szerint további eseményekkel is bővíthetjük a naptárat, csak az ***UID*** mező egyediségére ügyeljünk. Végül ***Calendar*** objektumunkat az `as_string()` eljárás segítségével beírjuk egy ***iCalendar*** fájlba:

```
print cal.as_string()
```

Mivel a `print` alapesetben a szabványos kimenetre ír, a ***CGI*** programok szabványos kimenete viszont a ***HTTP*** ügyfél felé irányul, végeredményként az ***iCalendar*** fájlt a ***HTTP*** kérést indító ügyfélnek küldjük el. ***MIME*** típusként ***text/calendar***-t választottunk, vagyis a ***HTTP*** ügyfél tudni fogja, hogy a kapott adatokat naptárként kell értelmeznie, illetve képes lesz azok helyes megjelenítésére. Ha megvizsgáljuk a kimenetet, láthatjuk, hogy valóban ***iCalendar*** formátumú:

```
BEGIN:VCALENDAR
PRODID:-//Python iCalendar 0.9.3//mxm.dk//
VERSION:2.0
BEGIN:VEVENT
DTEND:20050311T160000Z
DTSTAMP:20050311T001000Z
DTSTART:20050311T140000Z
PRIORITY:5
SUMMARY:ATF határidő
UID:ATF20050311A@lerner.co.il
END:VEVENT
END:VCALENDAR
```

Be kell vallanom, az előzőhöz hasonlóan ez a példa is egy kicsit mesterkélt volt. Igaz ugyan, hogy a naptárat dinamikusan állítottuk elő, de az esemény bele volt drótozva a programba, így módosítása, hozzáadása vagy törlése csak a programozó számára lehetséges. Fogalmazzunk tehát úgy, hogy újabb lépést tettünk az események és dátumok programból történő előállítása felé. A következő az lesz, hogy a dátumokat fájlban vagy akár relációs adatbázisban helyezzük el, és a programmal az adatokat futás közben alakítjuk át.

Összefoglalás

Ebben a hónapban megvizsgáltuk a dinamikus naptárak a ***Python*** egy egyszerű ***CGI*** programba burkolt ***iCalendar*** moduljával végzett előállítását. Ugyanakkor láttuk azt is, hogy milyen korlátai vannak egy olyan naptárnak, amelynek bejegyzéseit a merevlemezen tároljuk. Jobb megoldás lenne, ha az események adatait egy relációs adatbázisban helyeznénk el, amely önmagában is képes a dátumok kezelésére, valamint megfelelő eljárásokat biztosít a felhasználók és a csoportok hozzáférésének szabályozására. A következő alkalommal tovább bővítjük a naptárprogramot, adatait adatbázisból fogja venni, az ***iCalendar*** fájlokat ***PostgreSQL*** táblák alapján fogja előállítani.

Linux Journal 2005. junius, 134. szám

A cikk forrásai: ➲ www.linuxjournal.com/article/8197

Reuven M. Lerner

FreeBSD – a szomszéd vár (9. rész)
A várbörtön cellái

Gyakran kerülhetünk olyan helyzetbe, amikor meg kell védenünk a rendszerünket egy szolgáltatástól, esetleg önmagától, vagy a többi programtól, a külső-belső támadásoktól, illetve a sérülékeny vagy könnyen támadható az adatokat mindezektől. A probléma megoldása egyszerű, a szolgáltatásokat bezárjuk egy börtön cellájába: ez a jail alrendszer.

Mi az a jail?

A ***Linux*** esetén már ismert `chroot` program távoli rokona, amellyel ***FreeBSD*** alatt tudunk programokat elzárni a rendszer többi részétől; egy kicsit másképp és egy kicsit jobb körülmények között. A `jail` ugyanis ***FreeBSD*** virtuális gépeket hoz létre, amelyeknek közös rendszermaggal bírnak, viszont minden mást külön kezelnek. A gépen kívülről nézve egy `jail` alrendszer olyan, mintha lenne még számítógépünk, amelyeken ugyan az a ***FreeBSD*** van, mint a mellette lévőkön (holott csak egy gépünk van). Minden virtuális gépnek van egy külön ***IP*** címe, amelyre hallgat, a rendszer többi részével csak (virtuális) hálózati felületeken tud kommunikálni, leszámítva a megadott könyvtárstruktúrát, amelyek a rendszermagon keresztül képes kezelni. Azt tudom mondani, hogy a ***FreeBSD*** `jail` megvalósítása többet tud, mint a ***Linux*** chroot (teljesen elszeparált virtuális gép); kevesebbet tud, mint az ***UML (User Mode Linux)***, mivel a rendszermag közös, az nem cserélhető.

Miképp készül?

A bebörtönzést elő kell készíteni, amely tevékenység alatt az alaprendszer könyvtárainak és állományainak a `jail` könyvtárába való másolását értjük; ezen túl néhány beállítás is szükséges lehet, hogy a bebörtönzött folyamat megfelelően tudjon működni.

Általában két módszer közül választhatunk, mind a kettő célravezető, de más-más módon éri el a biztonságos üzemeltetést. Megtehetjük, hogy a teljes telepítés után eltávolítjuk az állományok és könyvtárak egy részét, egészen addig, amíg ***még*** működőképes a bebörtönzött program (kövér modell). Esetleg egyenként másoljuk át az állományokat, és a könyvtárakat, egészen addig, amikor ***már*** működőképes a program (***sovány*** modell). Akár a kettő módszert kombinálhatjuk is, felmásolunk egy-egy könyvtárat, majd elkezdjük törölgetni a mappából az állományokat. Az interneten keresgélve nagyon valószínű, hogy megtaláljuk a kész recepteket az adott szolgáltatás működéséhez minimálisan szükséges állományok listájával együtt.

A börtön felépítését sok különféle módon tehetjük meg, ebből három módszert szoktunk használni: ***mezei*** másolás, a `make world` parancs segítségével, illetve egy külön telepített programot használva.

A mezei másolás okán egyszerűen átmásoljuk a `jail` könyvtárába azokat az állományokat, amelyek szükségesek a bezárandó programunk futásához (és magát a futtatandó programot is). Gyakorlatilag ez a legegyszerűbb megoldása a börtönünk létrehozásának, bár kétségkívül nagyobb szakértelmet kíván. Sajnálatos, hogy a börtön karbantartása is nehezebb így, mint a többi módszert használva.

A klasszikus (leginkább támogatott) módszer szerint egy új alaprendszert építünk fel a börtön egy-egy cellájában. Teljesen azonosan kell eljárnunk, mint az alaprendszer frissítésénél, egyetlen paraméterrel kell többet megadnunk: hol készüljön el az új alaprendszer, illetve néhány más parancsot is végre kell hajtanunk.

```
# export JAILDIR=/usr/local/jails/10.1.1.213
# cd /usr/src
# mkdir -p $JAILDIR
# make world DESTDIR=$JAILDIR
# cd $JAILDIR
# ln -sf dev/null kernel
```

Ezzel a tevékenységgel számítógépünk ugyan annyi időt tölt el, mint az alaprendszer frissítésével, tehát akár több órán át is tarthat. Ha gyakran kell `jail` rendszert elkészítenünk, akkor érdemes egy példányt eredeti állapotában megtartani, s így csak át kell másolnunk a teljes könyvtárszerkezetet. Ezzel a módszerrel egy börtöncella mérete azonos lesz a teljes alaprendszer méretével, vagyis közel 200MBájt helyet fog elfoglalni. Ezen a hatalmas méreten sokat tudunk csökkenteni, ha a felesleges részeket eltávolítjuk (dokumentáció, kézikönyv oldalak, stb.).

Második legnagyobb szakértelmet kívánó kihívás a `sysinstall` program használata a `jail` beállításához. Ekkor alapvetően egy programot kell felmásolni a börtönünk könyvtárába: a `/stand/sysinstall` állományt.

```
# mkdir $JAILDIR/stand
# cp /stand/sysinstall $JAILDIR/stand
```

A bebörtönzés elindításához szükséges a helyi hálózathoz hozzáadni egy IP címet (például `10.1.1.213`), ahol elérjük majd a börtönükbe bezárt programjainkat.

```
# ifconfig vr0 alias 10.1.1.213/32
```

Két fontos könyvtár van, amit nem tudunk könnyedén létrehozni, a `dev` és a `proc`, amelyek speciális fájlrendszert hordoznak: a `devfs` és a `procfs` nevűt. Ezt a két fájlrendszert fel kell csatolnunk msinden egyes jail könyvtárba.

```
# mkdir dev
# mount_devfs devfs $JAILDIR/dev
# mkdir proc
# mount_procfs proc $JAILDIR/proc
```

Az alaprendszer elkészítése nem vonja maga után a */etc* könyvtár elkészítését (amely teljesen érthető a `make world` alapvető feladatát tekintve), így ezt külön parancs segítségével kell nekünk elkészítenünk.

```
# cd /usr/src/etc
# make distribution DESTDIR=$JAILDIR
```

Nincs más hátra, mint beállítani a börtön kényelmi szolgáltatásait, ehhez már be kell térnünk a cellába is. A `jail` parancs szolgál a bebörtönzés elindítására, amelynek az első paramétere az elkészített könyvtárszerkezet belépési pontja, a második paramétere a börtön neve, harmadik paramétere a kiválasztott IP cím, majd ezeket követi a végrehajtandó parancs elérési úttal együtt.

```
# jail /usr/local/jails/10.1.1.213/ private213
➥10.1.1.213 /rescue/sh
```

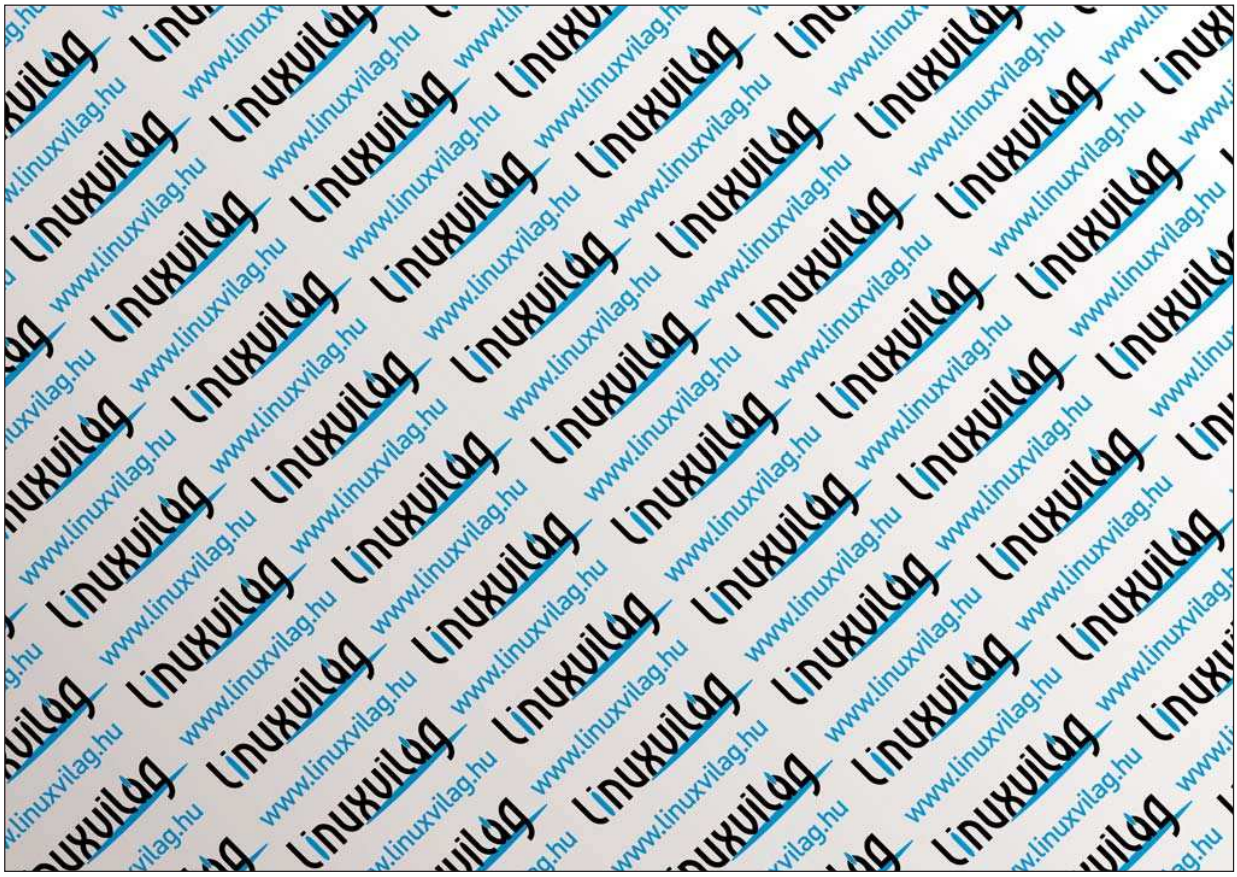
A kapott üres parancssorba kezdésképp a `sysinstall` programot írhatjuk.

```
$ /stand/sysinstall
```

Nem kell meglepődni, megkapjuk a ***FreeBSD*** telepítésénél is kapott programot, amellyel most újra feltelepíthetjük a ***FreeBSD*** rendszerünket: a `jail` alrendszer ugyanis olyan, mint egy teljesen új telepítés. A rendszermag már fut, az alaprendszer rendelkezésre áll, de semmi egyéb nincs beállítva, engedélyezve: mindent újra be kell állítanunk és engedélyeznünk. Az első használatbavétel előtt – amikor már a bebörtönzött programunkat szeretnénk futtatni – érdemes a `/sbin/init` programot elindítani a `jail` első beállításához (***SSH*** kulcsok elkészítése, stb.). Nem kell megijedni, a képernyőkép teljesen azonos lesz, mint amikor a ***FreeBSD*** rendszerünk elindul, hiszen a virtuális gépünkben is egy ***FreeBSD*** indul el. Programok telepítése és karbantartása azonos módon történik, mint azt a börtönünkön kívül tettük, így az alaprendszer frissítése és a `ports` adatbázis is járható út.

Segédprogramok

Érdemes néhány segédprogramot telepíteni, amelyek sokat segítenek a börtönünk felügyeletében. Néhány



programot a börtöncellába kell telepíteni, némelyiket a börtönön kívülre. A legfontosabb a ***jailutils*** csomag, amely a börtönön belül futó programok kezelésében segít (`jps`, `jtop`, `jkill`), mivel a `jail` belső mechanizmusai ezen programok normál futását jelentősen befolyásolják. A `jailadmin` vagy a `jailctl` csomag a börtönök elindításában, leállításában; illetve elkészítésében is segítséget nyújtanak.

A börtöncella beállítása

Fontos tulajdonsága a `jail` rendszernek, hogy nem hallgatózik a számítógépünk összes elérhető hálózati felületén, csak a parancssorban megadott IP címen. Ha megnézzük a börtönünkön kívül a `vr0` eszközhöz rendelt címeket, akkor láthatjuk, hogy több címe is van, azonos fizikai címmel.

```
# ifconfig vr0
vr0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
➥mtu 1500
        inet 10.1.1.211 netmask 0xffffff00
        ➥broadcast 10.1.1.255
        inet6 fe80::211:5bff:fe09:782a%vr0
        ➥prefixlen 64 scopeid 0x2
        inet 10.1.1.213 netmask 0xffffffff
        ➥broadcast 10.1.1.213
        ether 00:11:5b:09:78:2a
```

A `jail` rendszeren belül szintén megtaláljuk ezt az eszközt, de már csak a saját ***IP*** címével, a többi cím nem is látszik.

```
$ ifconfig vr0
vr0:
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
➥mtu 1500
        inet 10.1.1.213 netmask 0xffffffff
        ➥broadcast 10.1.1.213
        ether 00:11:5b:09:78:2a
```

Ha ki szeretnénk próbálni a jail hálózatának működőképességét, nem várt problémába ütközhetünk.

```
$ ping 10.1.1.212
ping: socket: Operation not permitted
```

A `ping` működéséhez igazzá kell állítani a `security.jail.allow_raw_socket` rendszerváltozót még a `jail` elindítása előtt.

```
# sysctl -a security.jail.allow_raw_sockets=1
security.jail.allow_raw_sockets: 0 -> 1
# jail /usr/local/jails/10.1.1.213/ private213
➥10.1.1.213 /rescue/sh
$ ping 10.1.1.212
PING 10.1.1.212 (10.1.1.212): 56 data bytes
64 bytes from 10.1.1.212: icmp_seq=0 ttl=64
➥time=7.635 ms
```

E nélkül csak a ***TCP*** és az ***UDP*** csomagokat jutnak ki a börtönből, ha például ***ICMP*** csomag is szükséges, akkor ezt a beállítást tegyük meg. Ez az engedélyezés viszont biztonsági kockázatot is jelent, így csak végszükség esetén használjuk; célravezetőbb egy olyan helyettesítő eszköz használata, amely ***TCP*** porton át is képes a hálózat működőképességét ellenőrizni (például az ***echoping*** csomag).
A hálózat többi beállítása (gazdagép neve, útválasztás, névfeloldás, stb.) azonos módon történik az alaprendszer telepítéskori beállításaival, használjuk nyugodtan erre a célra a `sysinstall` programot.

Lemezkezelés

A börtönben futó program nem látja a külsőleg felcsatol állományrendszereket, csak a saját főkönyvtárát tartalmazó fájlrendszert tudjuk lekérdezni.

```
$ df
Filesystem  1K-blocks      Used      Avail Capacity
➥Mounted on
/dev/ad0s1f 150307738 16267994 134039744     11%
➥/usr
```

Néhány tanács és pár megjegyzés

A `jail` nem óvja meg a gépünket a feltöréstől, mindössze – helyes tervezés esetén – az érzékeny adatokat védi meg a támadótól. Minél több `jail` készül el a gépünkön, annál nehezebb lesz azok karbantartása, hiszen egy-egy alaprendszert érintő biztonsági hiba kijavítását az alaprendszerünkön és az összes `jail` rendszerünkön is végre kell hajtanunk. Érdemes a `jail` belső felhasználóit leredukálni minimálisra, hiszen a legtöbb inaktív lesz. A `jail` alkalmas arra, hogy a teljes ***FreeBSD*** rendszer rendszergazda (root) jogú adminisztrálását kiadjuk egy másik személynek. Ő képes belépni a megadott IP címen, minden frissítést, telepítést és beállítást képes elvégezni, viszont nem képes az alaprendszer is érintő beállítások elvégzésére (particionálás, fájlrendszer elkészítése, más fájlrendszer felcsatolása, hálózati eszközök átállítására, stb.). Igazából egy olyan virtuális gépet tudunk készíteni a kiszolgáló gépünkön, amelyet egy ***FreeBSD*** rendszert ismerő személy képes kezelni, s úgy tekinthet rá, mint egy önálló számítógépre.

Auth Gábor (auth.gabor@enaplo.hu)
Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

A FreeBSD projekt honlapja: ➲ http://www.freebsd.org

A magyar FreeBSD honlap: ➲ http://www.freebsd.hu

A magyar BSD honlap: ➲ http://www.bsd.hu

A kézikönyv magyar fordítása
➲ http://www.enaplo.hu/FreeBSD/handbook/

KimDaBa

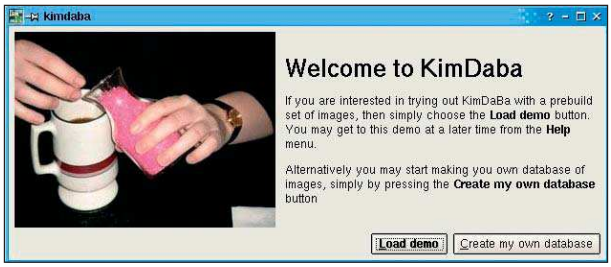
Rendet a képek között.

Ha leszámítom a gyerekkori fotóimat és fényképezőgépemet, akkor azt kell mondjam, hogy éppen egy évtizede foglalkozom egy amatőr fotós szintjén fényképezéssel, többnyire hobbi céljából. Ennek az időnek a felét egy – a profi kategória küszöbét éppen átlépő – tükörreflexes fényképezőgéppel fényképeztem el. Többnyire diafilmre dolgoztam, amit felhasználás előtt hűtőben kellett tárolni. Azt hiszem, részben azért tanultam meg jól fényképezni, mert egy-egy elrontott kép okozta veszteség igencsak forintosítható volt. Digitális gépet egészen sokáig nem voltam hajlandó venni, mivel a beszerezhető típusok egyszerűen nem feleltek meg a fotózásról alkotott elképzeléseimnek. Aztán egyszer csak fordult a kocka, és számomra már megfelelő minőséget kaptam elérhető áron. Minőségi etalonnak természetesen továbbra is a tükörreflexes gépemet tekintettem. Annyi történt mindössze, hogy egyszerre annyi pénzért tudtam megvenni egy hasonló tudású digitális gépet, mint amennyiért annak idején a közönséges filmre dolgozó gépemet.

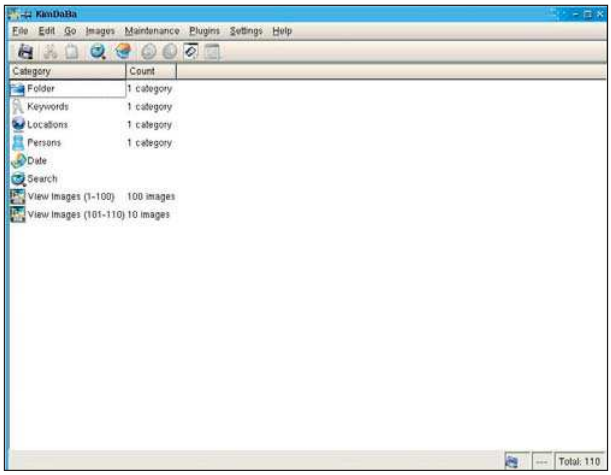
Az évtized első öt évének fényképei egy átlagos család fotóalbumával vannak egy „súlycsoportban”. A kartondobozon túl nem fordítottam különösebb figyelmet ezeknek a csoportosítására vagy kategorizálására. A tükörreflexes géppel készültekre már több gondot fordítottam, hiszen ezek többségében sok munka fekszik. A diakockákat tekercsenként egy-egy légmentesen lezárható dobozkába táraztam be diakeretekbe, amelyekre ráírtam a tekercs témáját, majd a diakocka száma szerint az adott képpel kapcsolatos félsoros összefoglaló szöveget is feljegyeztem.

Igazából mindig sajnáltam, hogy nem tudtam magamat arra rávenni, hogy minden egyes kockához leírjam a helyszínen a készítés körülményeit, így ezeket nem jegyeztem fel a kockák mellé. Havonta-kéthavonta egy tekercsnyi (36 diakocka) kép tárolása és leírása nem volt nehéz, sajnos a visszakereshetőség nem volt adott; elég jól ismertem a „gyűjteményem”, de mindig akadt néhány elfeledett fotó, amely pont ***akkor*** nem került a kezeim közé, amikor szükség lett volna rá.

A digitális fényképezés egy kicsit felborította ezt az idilli állapotot, mivel csak a fényképezőgép tudásának megismerésére való tekintettel 1200 képet készítettem az első két hét alatt (amelyek nagy részét aztán könnyű szívvel töröltem). A kattintások száma idővel csökkent, bár nem nagyon jutottam még a heti 50 képet jelentő álomhatár alá. Pedig nagyon igyekszem, csak egyszerűen minden apróságra



Nyitókép

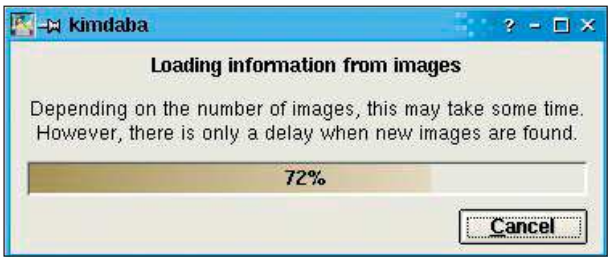


A főablak

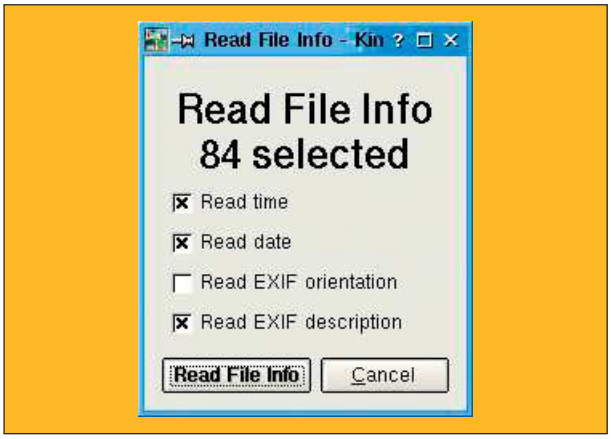
bizseregni kezd a mutatóujjam, és valahogy nagyon könnyen lenyomódik az exponáló gomb...

Amikor a digitális fényképezőgép már a második hónapját kezdte eltölteni a kezeim között, rájöttem: ha így folytatom, akkor bele fogok fulladni a képekbe. Első felindulásomban arra a következtetésre jutottam, hogy keresek egy programot, amely a fényképek kategorizálásában segít, visszakereshetővé teszi az egyes eseményeket és helyszíneket. Sajnos az első felindulás nem oldott meg semmit, a letöltött és kipróbált tucatnyi program nem váltotta be a hozzá fűzött reményeimet, így elkezdtem játszadozni a gondolattal, hogy írok egy ilyen programot.

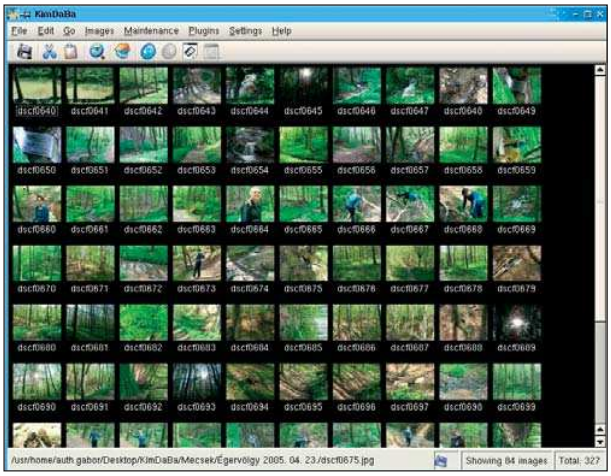
Nem, nem a saját programomról írom a cikket, mivel a munkát idő hiányában el se kezdtem. Viszont egy puszta véletlennek köszönhetően megtaláltam a ***KimDaBa*** csomagot.



Képek újratöltése



EXIF információk



Előképek

Egyszerűen a ***Beep Media Player*** médialejátszó programot szerettem volna megkeresni a ***FreeBSD*** rendszerem csomagadatbázisában és erre a „bmp” szöveget találtam a legkifejezőbbnek. Sajnos a ***BMP*** programot nem találtam a kapott listában, viszont az „***Image database for KDE***” szöveget annál inkább. Ez mindössze annyi kapcsolatban volt a „bmp” keresési feltétellel, hogy történetesen ***BMP*** formátumú képeket is kezel. Kaptam az alkalmon, és feltelepítettem a ***KimDaBa*** programot, a médialejátszó program telepítése pedig teljes feledésbe került. Igazából csak most jutott eszembe amint ezt a cikket írom. Tehát: ezennel megragadom az alkalmat, és minden kedves olvasóm szíves figyelmébe ajánlom a ***Beep Media Player*** programot, amiről a fenti malőrnek köszönhetően a nevén kívül egyelőre nem sokat tudok.

És most vissza a ***KimDaBa***-hoz. Bizonyosság híján csak remélni merem, hogy a főbb ***Linux*** terjesztések tartalmazzák ezt a remek programot. Ha mégsem, akkor a ➲ ***http://ktown.kde.org/kimdaba*** címről is letölthető. Verziószám szerint már a 2.1-es változatnál tart a fejlesztőcsapat, így bizton remélhetjük, hogy kiforrott és könnyen kezelhető a program. Én egy ideje a 2.0 verziót használom 3.4-es ***KDE*** rendszer alatt. Sajnos magyarítás nincs hozzá, de az általános használatához elegendő néhány alapvető angol szó ismerete. (Amúgy pedig várom azokat a jelentkezőket, akik segítenének a program magyar változatának elkészítésében.)

Mindenképpen javaslom a programmal érkező demó megnyitását, mivel ezzel egy már jól beüzemelt fotóalbumot tudunk megtekinteni (átlagos családi fotóalbummal). Ezek után készítsük el a saját adatbázisunkat, amelynek egy külön mappában adjunk helyet (ennek a neve találóan lehet mondjuk ***KimDaBa***).

Érdemes megjegyezni, hogy a megadott könyvtárba kezdésképp nem tanácsos túl sok fényképet tenni, mert a program az összeset betölti, átnyálazza és begyűjti a hozzájuk tartozó információkat. Mármost ha az első futtatás során minél hamarabb meg szeretnénk ismerni a program szolgáltatásait, nem szerencsés, ha induláskor ki kell várni azt a tizenhét percet, amíg a rendszer az említett könyvtárban található 2553 képet feldolgozza. (Igen, tapasztalatból írom ezt...)

A program fő ablaka puritán, vagy ha így jobban tetszik kényelmes és átlátható. Mellőz mindennemű céltalan díszítést. A képeinket mappa, kulcsszó, helyszín vagy személy szerint tudjuk csoportokba rendezni. Alapesetben egyetlen nagy csoportunk van, hiszen nem adtunk meg kategóriákat.

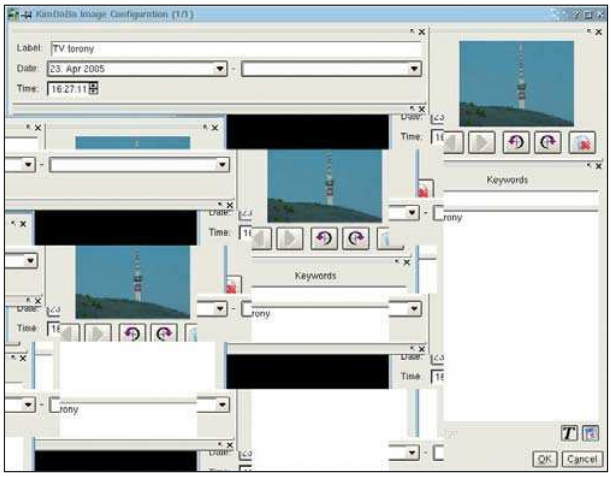
Az első és legfontosabb feladat a képek mappákba rendezése, mivel így később sokkal hatékonyabban tudunk szortírozni. Érdemes egy „ömlesztett” könyvtárat is fenntartani, ahova az újabb képeket tudjuk bemásolni a fényképezőgépről. (Erre a célra én az induló könyvtárat tartom fenn.)

A kezdeti néhány képen kívül másoljunk bele a ***KimDaBa*** könyvtárába néhány képet, amelyet már előzőleg mappákba szortíroztunk, s szólítsuk fel a képek újraolvasására, amelyet a ***Maintenance/Rescan for Images*** menüpontban tehetünk meg. A bemásolt képek mennyiségétől függően a program rövidebb-hosszabb ideig keresi a változásokat.

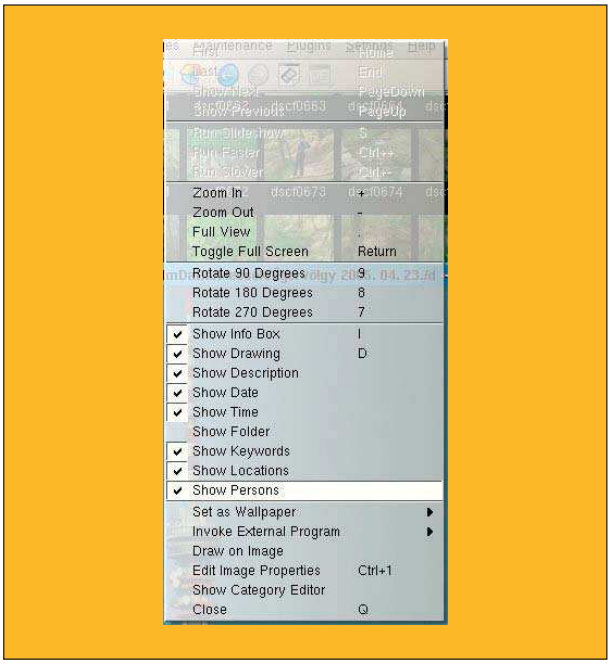
Válasszunk ki egy szimpatikus mappát – esetemben ez a ***Mecsek/Égervölgy 2005. 04. 23.*** nevű lesz – s a ***Maintenance/Read EXIF info from files...*** menüpont segítségével dolgoztassuk fel a programmal a képekben található ***EXIF*** információkat (blende, expozíció ideje, gyújtótávolság, stb.). Sajnos ez a lépés egyben törli a képekhez rendelt saját információinkat, így mindenképpen ezzel a művelettel kell kezdjük a képek kezelését.

Alapesetben csak a képhez rendelt idő és a dátum kerülne beolvasásra, ezért mindenképpen jelöljük be a képhez tartozó leírást, illetve akár az orientációt is. A beolvasás gyorsan megtörténik, s ha rákattintunk a „***View images***” szövegre, akkor azonnal láthatjuk is a képeket kicsiben. Sok kép esetén ez a művelet eltarthat egy ideig, mivel a program alapértelmezésként röptében gyártja le

Kép előnézete



A kép jellemzőinek szerkesztése



Helyi menü

TV torony

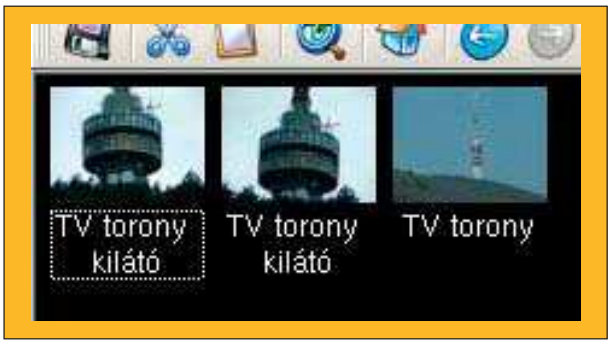
az előképeket. Ezt a ***Maintenance/Build thumbnails*** menüponttal változtathatjuk meg, mikor is a program elkészíti a mappa összes tagjának az előképét.

Ha rákattintunk egy képre, akkor a rendszer azt megmutatja valamivel nagyobb méretben (600x450 pixeles felbontásban). Jól nézzük meg ezt a kis előnézeti ablakot, mert ez a program lelke, noha ez elsőre nem igazán látszik rajta. De kattintsunk csak a jobb egérgombbal az ablak területén, s rögtön előbukkan az egyik legnagyobb méretű helyi menü.

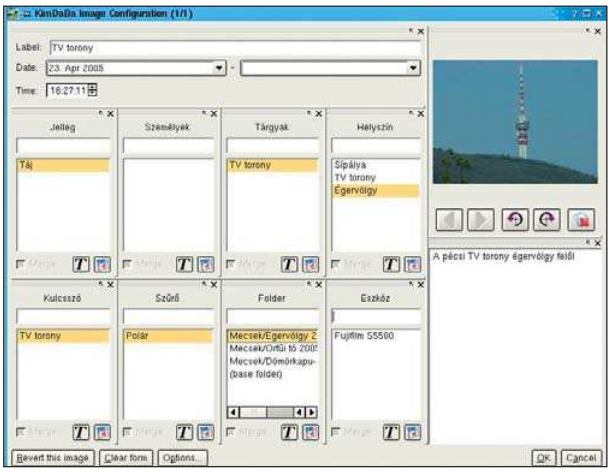
Ennek a legfontosabb és leggyakrabban használt pontja az ***Edit Image Properties*** lesz, amely egy hatalmas ablakot nyit meg. Ebben szépen sorban megadhatjuk a kép kiegészítő információit: adunk neki címkét és leírást; hozzáadjuk a személyek, a helyszínek, illetve kulcsszavak listájához a képen található látnivalókat, majd bezárjuk az ablakot. Ha mindent jól csináltunk, akkor a kép előnézetén már láthatjuk is a változásokat.

Szembetűnő változás, hogy a kulcsszavak, illetve a helyszínek és a személyek neve kék színnel jelenik meg, ráadásul aláhúzva: ezek bizony kapcsok a hasonló képekhez. Gyorsan keresünk emlékezetből néhány azonos tartalmú képet, s már láthatjuk is működés közben a képek közötti kapcsokat.

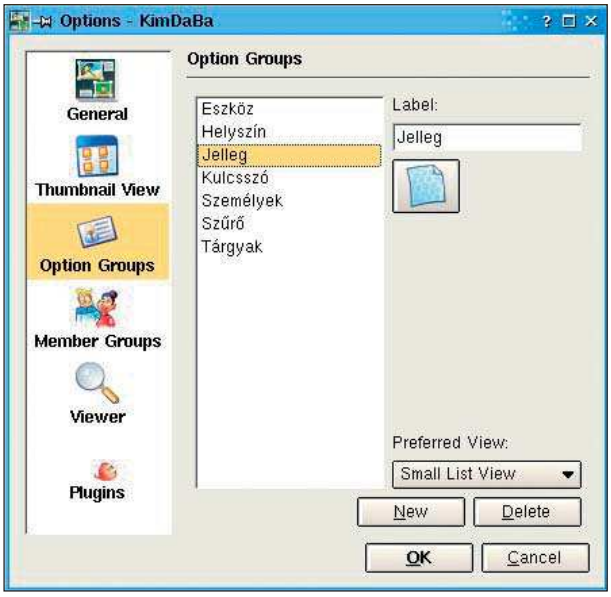
Ha például egy kulcsszóra kattintunk rá, akkor a program kigyűjti azokat a képeket, amelyeken a kulcsszó szerepel, így került egymás mellé a három kép a pécsi TV toronyról. Ha egy képen találunk valami fontos részletet, amelyet mindenképpen szeretnénk megmutatni az ismerősöknek, akkor akár rajzolhatunk is rá a ***Draw on image*** menüponttal. Nem büszkélkedhet a program túl összetett rajzeszközökkel (vajon miért nem integrálták a ***KolourPaint*** és a ***Karbon14*** programot erre a célra?), de azért kellemesen el lehet vele szórakozni. Egyetlen előnye ennek a rajzoló modulnak, hogy utólag el lehet tüntetni a képre rárajzolt dolgokat, azok ugyanis nem kerülnek bele magába a képállományban. Az egyszerű geometriai formák csak hozzárendelődnek a kép nevéhez. Ellenben a képhez meg tudunk hívni külső programot, így szükség esetén professzionális eszközökkel is tudjuk szerkeszteni vagy retusálni. Ha pedig valamelyik fotó nagyon megtetszik, egyetlen kattintással a munkaasztalunk hátterévé tudjuk tenni.



TV torony kulcsszó alapján



A kép jellemzőinek szerkesztése



Kategóriák szerkesztése

HTML oldal

A program egyik lényeges tulajdonsága, hogy át tudjuk szabni a kategóriákat a ***Settings/Configure KimDaBa...*** menüpont hatására megnyíló ablak ***Option Groups*** részén. A magam részéről elkezdtem a magyarítást azzal, hogy itt a csoportok megnevezéseit átírtam.

Érdemes hasonló módon kibővíteni a program által felkínált kategóriákat, így ugyanis sokkal több jellemző szerint tudjuk csoportba rendezni a képeket. Egy kép jellemzőinek szerkesztésekor azzal szembesülhetünk, hogy nem látjuk a módosított, illetve az újként hozzáadott saját kategóriákat. Ennek valószínűsíthető oka, hogy a program név szerint tárolja ennek az ablaknak az alkotórészeit. A megoldás kézenfekvő: az ***Options*** nyomógomb nyomán felbukkanó menüben tegyük láthatóvá a saját kategóriáinkat is. A külön nyíló ablakokat rá tudjuk húzni a fő dialógusablakra, így ez már minden kívánalomnak megfelel.

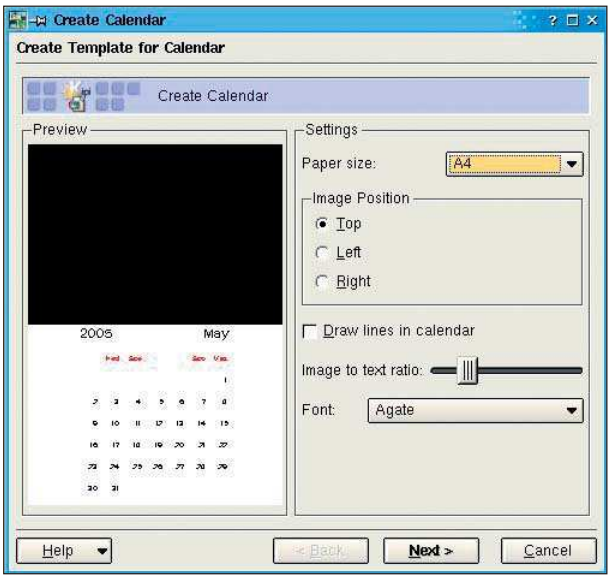
Ha már sikerül néhány tucat képünket a megfelelő kategóriákba besorolni, akkor tovább próbálkozhatunk a program egyéb képességeivel.

Három igencsak hasznos szolgáltatást találunk a ***File*** menüben (a mentést és a kilépést leszámítva): a képeink exportálását, importálását illetve egy ***HTML*** oldalba foglalt galéria elkészítését.

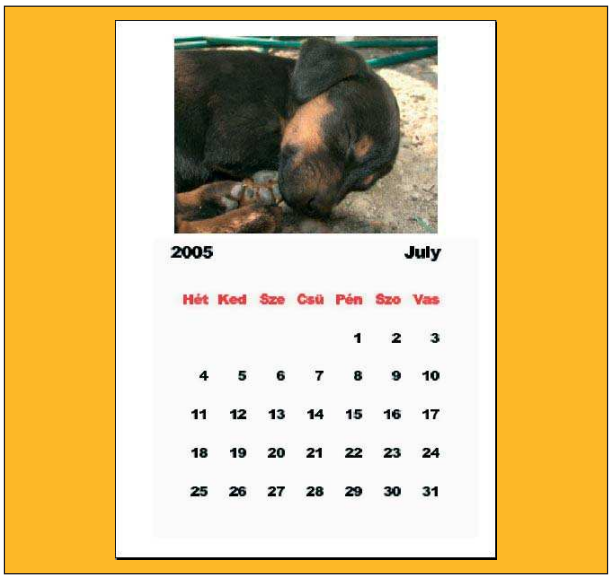
Az exportálás során a kiválasztott album (amelyet éppen nyitva tartunk) kerül ki a megadott könyvtárba. Alapesetben csak a kis előképek kerülnek bele egy ***.kim*** kiterjesztésű állományba a szükséges ***index.xml*** állománnyal együtt. Természetesen hozzámásoltathatjuk a képeket is, amelyek azonban sok helyet foglalnak el, így érdemes megadni az átméretezést is. Készüljünk fel arra, hogy az átméretezés több másodpercet is elvehet képenként, így egy 100-200 képet tartalmazó album exportálása több perc is lehet.

Az exportált állományt aztán oda tudjuk adni másoknak, akik képesek ezt importálni a saját képadatbázisukba. Egyszerűen csak ki kell választanunk a kapott állomány helyét, majd betölteni a képeket a megjelenő varázsló segítségével, amely lépésről-lépésre vezet minket kézenfogva.

Nagyon érdekes a ***HTML*** album készítése, amellyel a weboldalak készítését spórolhatják meg a fotográfusok. (Őszintén szólva én nagyon szeretek weboldalakat készíteni, így engem a KimDaBa-nak ez a szolgáltatása nem annyira vonz.) Számos jellemzőjét adhatjuk meg az elkészülő ***HTML*** oldalnak: címét, leírását; grafikai megjelenését, megadhatjuk a megosztani kívánt képek legnagyobb felbontását, s végül azt a könyvtárat, ahova menteni szeretnénk (alapbeállításként a saját ***public_html*** könyvtárunk lesz ez).



Naptár készítése



A kész naptár egy lapja

A program legérdekesebb része vitathatatlanul a ***Plugins*** menüpont, ahol több tucat eszköz, program és módszer várja, hogy valamelyiket munkára fogjuk. Én ezek közül hármat tudok nyugodt szívvel ajánlani: a naptár készítését (***Tools/Create Calendar...***), a videóállományba mentett bemutatót (***Tools/Create MPEG SlideShow...***), és a ***CD/DVD*** lemezre történő archiválást (***Export/Archive to CD/DVD...***).

A képes naptár készítése ugyan nagyon hálás feladat, csak a program egyes szövegeket a ***KDE*** nemzeti beállításaiból vesz, másokat pedig maga ír meg angol/amerikai formátumban. Az eredmény így aztán kissé felemás lesz. A naptárvarázsló első lapján az olyan feltételeket tudjuk megadni, mint a lap mérete, a kép elhelyezkedése vagy a szöveg stílusa. Ezt követi a képek kiválasztása, ahol a hónapokat látjuk egymás mellett. Ha rákattintunk egy hónapra, akkor a kapott listából ki tudjuk választani a kívánt képet. (Itt az aktuális mappában szereplő képek közül választhatunk, tehát még a naptárkészítés előtt válasszunk ki olyan kulcsszót vagy jellemzőt, amely alapján a képek leválogathatók.) A képek kiválasztása után következhet a nyomtatás. A program csak azokat a hónapokat nyomtatja, amelyhez választottunk képet.

Családi képeket nézegetve bennem gyakran támad olyan megmagyarázhatatlan inger, hogy legszívesebben mindet megmutatnám a család aprajának-nagyjának. Igen ám, de ehhez vagy vinnem kell magammal a számítógépet, vagy minden érintett állományról papírképet kell készíttetnem. Esetleg ki is nyomtathatom őket... Az első lehetőség nehézkes, a többi meg drága.

A problémának ugyanakkor van egy olyan megoldása is, ami egyenértékű a gordiuszi csomó átvágásával: készítsünk olyan bemutatót, amelyet le lehet játszani a legtöbb otthonban már megtalálható ***CD/DVD*** lejátszó készülékekkel. Ehhez egyszerűen a megfelelő menüpontot kell kiválasztanunk, majd néhány opciót beállítani (képek közötti szünet, a képek közötti átmenet ideje, háttérszín, hangállomány helye), illetve megadni a kész ***MPEG*** állomány végleges helyét és kiválasztani a kívánt képeket. A többit bízzuk nyugodtan a ***KimDaBa*** programra, illetve a ***K3b***-re amely az elkészült ***MPEG*** állományt felírja egy ***CD*** lemezre.

A digitális képeink archiválása nagyon fontos feladat, mivel csak és kizárólag digitális formában tudjuk veszteség nélkül tárolni őket. Minél fontosabb egy-egy képünk, annál inkább igyekezzünk minél több helyre és formában menteni.

Az egyik legjobb módszer a ***CD*** vagy ***DVD*** lemezre történő írás, amely azonban csalóka lehet, mivel a lemezek nagyon sérülékenyek. Itt egy karcolás eredménye nem egyetlen karc lesz egyetlen képen, mint azt a filmes világban esetleg megszoktuk...

A lényeg, hogy egyszerűen és csoportosítva mentsük a képeket, és erre a ***KimDaBa*** tökéletesen megfelelő. A lemezekre felkerül egy ***HTML*** oldal is, amely segít a képek közötti eligazodásban, illetve kapunk egy listát is az adott mappában található állományokról. Ha valami baj történne a gépünkkel, akkor a mentésből szépen vissza tudjuk állítani a képeinket.

A ***KimDaBa*** itt is a ***K3b*** íróprogramot használja a mentés elkészítésére, illetve a már ismert ***HTML*** galériát a navigációhoz. Javaslom e menüpont gyakori alkalmazását!

Zárszó

Lassan két hete használom teljes megelégedéssel a ***KimDaBa*** programot. Bár egyelőre vannak hiányosságai, világosan látszik, hogy a fejlesztők munkája ígéretes, és határozott a fejlődés. Szinte alig tudok olyan szabad programot mondani, amelyik felvenné a versenyt a ***KimDaBa*** könnyed használatával és átlátható felületével; a program tudása pedig folyamatosan bővül. Így nem kell aggódnunk: amit ma nem tud, azt holnapra talán már kezelni fogja.

Auth Gábor (auth.gabor@enaplo.hu)
Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

Hordozd a tenyereden – Linuxos PDA-k

Korábban egy rövid cikksorozatban már bemutattuk, hogyan tudunk egyszerűen és gyorsan egy hordozható számítógépre Linuxot telepíteni, illetve milyen beállítások szükségesek ahhoz, hogy kedvenc operációs rendszerünk a laptopunkon is működjön. Most megnézzük, hogyan telepíthetünk Linuxot egy tenyérszámítógépre, avagy közismertebb nevén egy PDA-ra.

Milyen rendszer fut és milyen rendszer futhat egy PDA-n

A palmtopokon gyári kivitelben két nagy gyártó rendszere fut, a ***PalmOS*** vagy a ***Microsoft PocketPC***. A két rendszer közül ugyan az előbbi nyúlik vissza nagyobb múltra, de mára inkább a ***Microsoft PocketPC*** terjed el, úgy vélem az ***Európában*** értékesített gépek többsége ezzel a rendszerrel van szerelve. Emellett még léteznek egyéb más operációs rendszerek tenyérméretű eszközökre, a legismertebb talán a ***SymbianOS***, de a tenyérszámítógépek operációs rendszerének uralkodója a ***PocketPC***.

A ***PDA***-kra készült rendszerek mindegyike különös, az asztali számítógépekétől kicsit eltérő rendszerrel van felszerelve, ezek úgynevezett ***beágyazott (embedded) rendszerek***. Ilyen beágyazott rendszereket használnak a mobiltelefonokban, az úgynevezett ***smartphone***-okban, illetve a ***tablet pc***-k is egyfajta beágyazott rendszert futtatnak. Természetesen ezen rendszerek között óriási eltérések lehetnek és vannak is.

Miután a ***Linux*** az új évezredben erőteljesebb hódításra indult, mint valaha, a fejlődés irányvonalából nem eshettek ki a palmtopok sem, így megjelentek az első ***Linuxok PDA***-ra. Ezek eleinte sokkal jobban hasonlítottak egy nagy-nagy hegesztésre, mintsem kiforrott rendszerre, de valahol mindent el kell kezdeni. A 2000-es évek elején egy nagy, ma már önmagában nem létező gyártó piacra dobott egy jópofa termékcsaládot, az ***iPAQ***-et. A ***Compaq*** annak idején ezzel a termékkel azt tűzte ki céljául, hogy meghódítsa azokat az embereket, akik egy notebook-ot túl nagynak, használatát pedig túl körülményesnek ítélték, viszont egy eszközt, amely könnyedén elfér egy zsebben is praktikusnak találtak, így megszületett az eszköz, amely leváltotta a jegyzettömböket és a határidőnaplókat. Természetesen nem csak a vállalati vezetők és az elfoglalt beosztottaik szívét sikerült megdobogtatni, hanem azokat a technikai újítások irányt érzékeny felhasználókat is, akik aztán először hobbiból, majd később komoly projektek keretében kezdtek ***PDA***-ra ***Linuxot*** fejleszteni.

Mára eljutottunk oda, hogy palmtopra is egész komoly, többé-kevésbé jól használható linuxos rendszerek állnak rendelkezésre. Ilyen rendszer az ***Intimate*** projektből

GPE – A beállítások ablak

kifejlődött ***Familiar Linux***, amely egy kifejezetten PDA-kra készült Debian ősből kifejlesztett terjesztés (***http://familiar.handhelds.org***).

Ahogy az asztali gépekhez készített ***Linux*** terjesztések is rendelkeznek különböző grafikus környezetekkel, úgy természetesen a ***PDA***-ra készült ***Familiar Linux*** is rögtön két környezetet is tartalmaz, az egyik egy ***Qt*** alapú ***KDE***-vel rokon rendszer az ***Opie***, a másik egy ***GTK***-alapú, ***Gnome*** rokon a ***GPE***.

Az első lépések

Mielőtt elkezdenénk a ***PDA***-nkra ***Linuxot*** telepíteni ne felejtsük el lementeni minden adatunkat, mert a telepítés megkezdése után ezeknek búcsút mondhatunk!

Ha megtörtént a biztonsági mentés, akkor kezdhetjük is telepítést. Töltsük le a ***Familiar*** weblapjáról a legfrissebb telepítőcsomagot és csomagoljuk ki a ***PC***-nk merevlemezére. Ez jelenleg a 0.8.2-es verzió, amit egy ***tar*** állomány

tartalmaz. Jelenleg hat különböző ***HP-Compaq IPAQ*** készülékhez érhető el telepítőcsomag, ettől eltérő eszközökre való telepítés előtt érdemes a fórumokban és a fejlesztői oldalakon körülnézni.
Amennyiben letöltöttük a telepítőcsomagot, akkor neki is foghatunk telepíteni a rendszert. Ennek rögtön több különböző módjával is próbálkozhatunk.

A bootloader telepítése

Mint egy hagyományos asztali ***PC*** esetén, itt is telepíteni kell valamiféle bootloadert (továbbiakban nevezzük ***rendszertöltőnek***), ami az indításkor a rendszermagot betölti.
Amennyiben olyan ***PDA***-ra telepítjük a rendszert, amelyen ***PocketPC*** operációsrendszer fut, akkor mindenképpen új ***rendszertöltőt*** kell telepíteni, ha viszont egy korábbi Linuxról frissítünk, akkor a megfelelő bootloader megléte esetén ettől a lépéstől eltekinthetünk.
Amennyiben telepíteni kell a rendszertöltőt a tenyérgépre, akkor többféle adathordozót is választhatunk telepítési forrásként. Én ezek közül most a gép saját memóriájából,

GPE – A naptár alkalmazás

GPE – A gombok beállítása

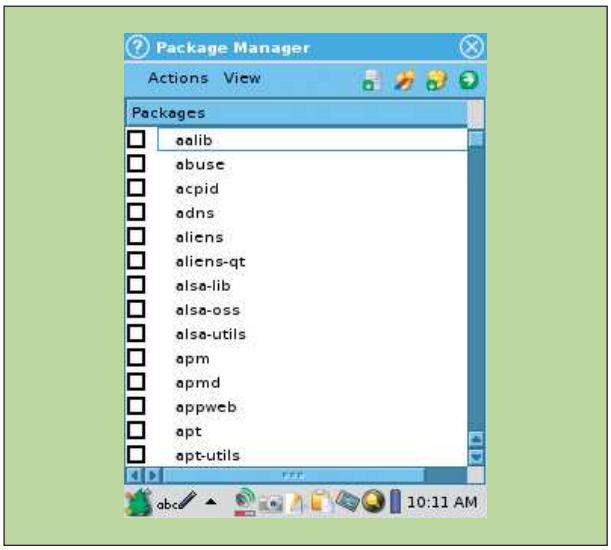
illetve egy külső CF/SD/MMC kártyáról való telepítést fogom röviden leírni. A többi telepítési mód megtalálható a terjesztés hivatalos weboldalán.

Rendszertelepítés a gép saját memóriájából

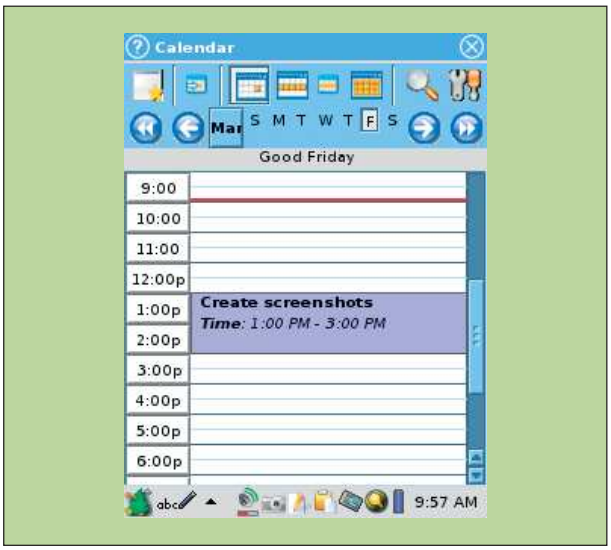
Ehhez először a telepítőkészletből két állományt, a ***BootBlaster*.exe***-t és a ***bootldr*.bin*** állományt másoljuk az ***ActiveSync***, vagy a ***SyncCE*** programok segítésével a ***PDA***-ra, majd a ***Pocket PC*** beépített állománykezelőjének segítésével futtassuk a fenti ***exe*** fájlt. Ezzel a segédprogrammal lementhetjük a rendszertöltő jelenlegi állapotát későbbi visszaállítás céljából, illetve felülírhatjuk azt a letöltött csomaggal.
Nagyon fontos, hogy ezt a műveletet csak hálózati áramforrásról végezzük, mert ha véletlenül áramkimaradás lép fel, a ***PDA*** könnyen érvénytelen memóriaállapotban maradhat és akkor csak a szervíz segít rajtunk.
Ha lementettük a jelenlegi rendszertöltőt, akkor töltsük is át a ***PC***-nkre, vagy mentsük valamilyen memóriakártyára.
Ezután már elkezdhetjük a rendszeröltő felülírását, amelyet a ***Flash*** menü ***Program*** menüpontjával indíthatunk el. Itt meg kell adni azt a bináris állományt, amely a rendszertöltőt tartalmazza és indíthatjuk is a memória felülírását.
Ez a művelet úgy fél percet vehet igénybe. Ha végeztünk, akkor az új rendszertöltővel a ***PDA***-t újraindítva a ***Pocket PC*** operációs rendszerének újra kell indulnia.

Rendszertelepítés memóriakártyáról

Ez utóbbi módszer ha lehet, még az előzőnél is sokkal egyszerűbb. Mindössze a memóriakártyára kell másolni a telepítőcsomag két fenti állományát, majd az előbb leírtakhoz teljesen hasonlóan kicserélni a rendszertöltőt.
Ha kicseréltük a rendszertöltőt, akkor nagyon egyszerűen telepíthető az új operációs rendszer. Másoljuk a memóriakártya gyökérkönyvtárába a ***reflash.ctl***, ***md5sums*** és ***jffs2*** állományokat, majd a ***Reset*** gombbal indítsuk újra a gépet úgy, hogy közben a ***joypad***-et lenyomva tartjuk. Újraindulás után, amint a boot képernyő megjelenik engedjük fel a ***joypad***-et és a hangfelvétel gombjával válasszuk ki a menüből a memória „újraflashelésével" való rendszertelepítést. Ez a folyamat néhány perc alatt lefut, majd a gép újraindul és feláll a kiválasztott grafikus felület.
Mind az ***Opie***, mind pedig a ***GPE*** programcsomag esetén néhány alapvető beállítást el kell végezni, de pár perc alatt üzemkész állapotba hozható a PDA. Ha ezzel megvagyunk, akkor készen állunk arra, hogy birtokba vegyük az új rendszert.
Mint már említettem a ***Familiar Linux*** egy ***Debian*** alapokra építkező rendszer, így azok a felhasználók, akik ismerik a ***Debian*** belső felépítését nagyon könnyen megbarátkoznak az új környezettel. A ***Familiar Linux*** csomagkezelőjét ***ipkg***-nak hívják, amelynek használata nem meglepő módon nagyon hasonló a ***Debian*** alatt megismert ***dpkg*** és ***apt*** csomagkezelőhöz. Az ***ipkg***-n keresztül telepíthetünk tehát új csomagokat a rendszerre a megfelelő tükörszerver kiválasztása után.
Amennyiben rendelkezünk a ***PDA***-hoz valamilyen hálózati kártyával, akkor a ***Debian Linuxnál*** megszokott ***/etc/networking/interfaces*** állományban beállíthatjuk az eszközt és innentől kezdve teljes hálózati támogatása van a ***PDA***-nak. Támogatott szinte az összes ***CF*** kártya alapú ***LAN*** és ***WLAN*** kártya, így ennek kiválasztása minden bizonnyal nem jelenthet különösebb problémát.
Amennyiben nincs ilyen hálózati eszközünk, akkor a ***Familiar*** weblapján találunk leírást, hogy miként lehet a soros, vagy ***USB*** csatlakozással az asztali gépünkön keresztül hálózatot biztosítani a ***PDA***-nak.
Miután én először telepítettem és beállítottam a rendszert, első dolgom volt a hálózat belövése, majd pedig ***SSH*** telepítése a ***PDA***-ra, mert így ***SSH***-n keresztül lényegesen kényel-



Opie – A csomagkezelő



Opie – A naptár

mesebben lehet a szöveges parancsokat végrehajtani. Ugyan kifogástalanul működik a *PDA* tapogatós billentyűzete, illetve a kézírás felismerő is, de ezek használata azért kicsit körülményes. Főleg parancssori eléréshez.

A felhasználói programok áttekintése

A következőkben átnézzük, hogy milyen csomagok is állnak rendelkezésünkre a ***Familiar Linuxhoz***. Ez alkalommal – bár a *Gnome* közelebb áll a szívemhez – az ***Opie*** felülettel ismerkedünk meg, mert meglátásom szerint ez kiforrottabb, mint a *GPE*, bár idővel ez is biztosan hozni fogja a tőle elvárható minőséget.

Az ***Opie*** felület egy négy lapból álló alkalmazáskezelővel és egy ***tálcával (taskbar)*** rendelkezik, minden funkció ezen keresztül érhető el. Az alkalmazáskezelő első füle a ***PIM***, a ***Personal Information Manager***. Ezen a fülön található meg a naptár, a jegyzettömb, névjegyzék és egy teljes értékű e-mail kliens. Utóbbi annyira teljes értékű, hogy támogatja a *POP3* és *IMAP* protokollokat, méghozzá azon titkosított változatát is, kezeli a mappaszerkezeteket és támogatja a ***HTML***-levelek megjelenítését is. Őszintén szólva ennek a kliensnek a használhatósága engem is meglepett, bőven túlszárnyalta minden várakozásomat. A naptár egy teljesen szabványos naptárprogram, hasonlóan a névjegyalbumhoz. Jegyzeteket két programmal is készíthetünk, az egyik az úgynevezett ***DrawPad***, a másik pedig a hagyományos ***Text Editor***. A ***DrawPad*** mint neve is mutatja inkább egy firkaprogram, ide kézzel, illetve a tollal írott szöveget rögzíthetünk, míg a ***Text Editor*** egy hagyományos szövegszerkesztő program, ide az érintőbillentyűzettel, vagy a karakterfelismerővel vihetünk fel szöveget.

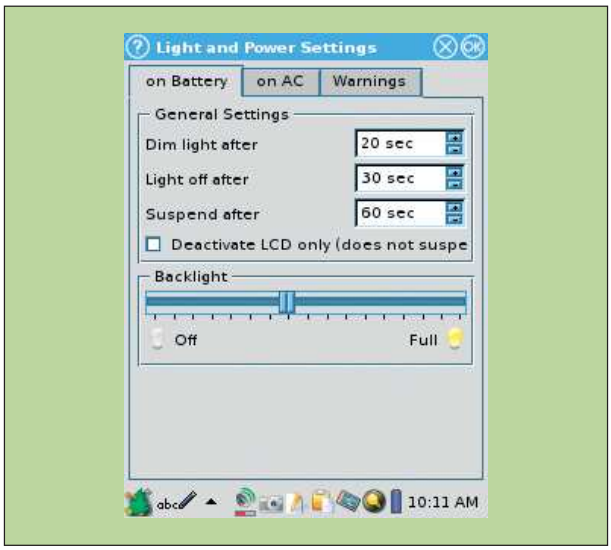
Az ***Alkalmazások*** fül alatt találhatóak meg azok a programok, amelyeket mi telepítettünk a csomagkezelő programon keresztül. Én olyan programokat néztem ki magamnak, amelyeket a mindennapokban az asztali *PC*-men is használok, így telepítésre került az ***Opie Sheet***, ami egy ***Excel*** vagy ha úgy tetszik ***OpenOffice.org Calc*** klón, valamint az ***Opie Writer***, ami egy ***OpenOffice Writer*** klón. Telepítettem a *PDA*-ra egy *PDF* olvasó csomagot, amely a *PDF* fájlok 99%-át gond nélkül kezelte. A maradék 1%-ért pedig a linuxos ***PDF***-olvasóm sem volt oda túlságosan...

Az ***Opie*** tartalmaz egy ***Konqueror*** böngészőt, amellyel egy tökéletesen használható, teljes értékű webböngészőt kapunk a rendszerhez. Mind az ***Opie***, mind a *GPE* támogatja azt a megoldást, hogy 90 fokkal elforgathatjuk a képernyőt, így pedig már egész kényelmesen lehet weblapokat nézegetni. Eddig azok a programok, amelyeket vártam a *PDA*-tól, most pedig nézzünk pár olyat, amely kisebb meglepetésként ért. Található a csomagok között egy ***Familiar Linuxra*** optimalizált ***XMMS***, amelyet annyira eltaláltak a fejlesztők, hogy szinte tökéletesre sikerült. Használati értéke megegyezik az asztali ***Linuxokra*** telepíthető ***XMMS***-el. Telepítettem továbbá egy ***Familiar Linuxos rdesktop***-ot, amely programmal távoli kliensekkel létesíthetünk asztalkapcsolatot, így akár a *PDA*-ról vezérelhetjük az asztali gépünk grafikus felületét. Természetesen a 320x240-es felbontás azért némi korlátot szab az örömöknek, de szükséghelyzetben ez egy szintén teljesen használható megoldás, nekem nagyon tetszett. Találtam egy ***Gaim*** csomagot is, úgy tűnik, hogy eltekintve pár gyerekbetegségétől ez a program is teljesen használható a mindennapokban, szinte megegyezik az asztali változattal. Az összes program közül azonban legjobban a csomagból telepíthető ***mplayer*** tetszett. Szinte bármilyen videót próbáltunk lejátszani vele, simán működött – persze a nagyon nagy felbontású videók némelyike szaggatott picit, de ezt tudjuk be a 200MHz körüli processzornak. Egy kicsit erősebb vassal és egy méretes memóriakártyával kész a nyílt forrású mobil médialejátszó. Ennél többet és jobbat kívánni sem lehet. Emellett még nagyon jópofa programnak találtam a ***GPSDrive Familiar*** linuxos változatát, amit azonban sajnos hely hiányában már nem volt lehetőségem komolyabban kipróbálni. Amennyiben működése megközelíti az asztali változatét, akkor egy megfelelő GPS eszközzel már navigációs rendszerré is alakíthatjuk a *PDA*-t.

A beállítások

A beállítások fül alatt grafikus felületen keresztül bekonfigurálhatjuk a teljes tenyérgépes környezetet, így akik

Opie – Az asztal



Opie – Fényerő és energiagazdálkodás

Opie – Az asztal elforgatva

nincsenek kibékülve a parancssorral, vagy nem merülnének el annak rejtelmeiben, szintén nem kell félniük, hogy ne tudnák használni a rendszert.
Ahogy azt a *KDE*-ből megszokhattuk, a rendszer megjelenését teljes egészében a saját igényeinkre szabhatjuk, kezdve a színektől az ablakkereteken át egészen a felületelemek megjelenéséig. Nincs az a *Pocket PC*-s *PDA*, amelyik ilyen tág keretek között szabható testre – bár ezen már nem lepődöm meg.
Szintén testreszabható a *PDA*-n található nyomógombok funkciója, így minden gombhoz egyedi parancsot tudunk rendelni. Szintén a beállítások fül alatt találjuk az olyan rendszerparaméterek állítását, mint az idő, nyelv és régió beállítás. A fül alatt találjuk a rendszer energiagazdálkodási beállításait is. Itt külön szabályokat állíthatunk be tápellátás alatt álló módra és akkumulátoros üzemre is. Testre szabhatjuk a fényerő erősségét, a háttérvilágítás lekapcsolásának és a *PDA* kikapcsolásának időpontjait. Mindezt a két profilhoz teljesen szétválasztva. A fényerő beállítását bízhatjuk az automatikára is, ilyenkor az uralkodó fényviszonyoknak megfelelően változtatja a *PDA* a megvilágítás erősségét.
Szintén a beállítások között találjuk a ***Familiar Linux*** grafikus csomagkezelőjét, ahol csomagok ki- és bepiálásával válogathatunk a telepített és telepítendő csomagok között. Ezt a csomagkezelőt bármelyik felhasználó könnyedén használhatja. Hasonlóan egyszerűen állíthatóak be a hálózati beállítások. A rendszer érzékeli az összes elérhető hálózati eszközt és felajánlja ezek beállítását. A rendszer támogatja a legújabb *WLAN* eszközöket is, sőt a *WEP* titkosítást. Sajnos azonban *WPA* titkosításra még nincs lehetőség. Van helyette viszont *VPN* kezelés, méghozzá több protokollon is, így akár *IPSec* kapcsolatot is kiépíthetünk bármely elérhető kiszolgálóval, így a vezetéknélküli hálózati forgalmat is tudjuk titkosítani.

Összegzés

Az elmúlt másfél hónapban, mióta a linuxos asszisztensemet használom nagyon megszerettem. Használata, filozófiája teljesen megegyezik egy hagyományos asztali ***Linuxéval***, ha gondba kerültem bármi miatt továbbra is hagyatkozhattam a naplókra, ha valamit kézzel kellett beállítani, akkor parancssorból ezt is pillanatok alatt meg lehetett oldani, de még ha véletlen otthon felejtettem a gépet sem kellett kétségbe esni, mert *SSH*-n elérhető volt a gép.
Amellett pedig, hogy digitális asszisztensnek tökéletes, találtam egy más – kissé talán rendhagyó – felhasználási módot is. Két *CF* hálózati kártya beszerzésével a gép bármikor egy határozottan kis fogyasztású, de teljes értékű linuxos tűzfallá és/vagy útválasztóvá alakítható... Ezen a téren is mindent tud, amit elvárhatunk tőle.
Ha viszont éppen úgy döntök, hogy útra kelek, fogok egy nagy *CF* kártyát és ráteszek egy-két filmet, számtalan mp3-at és kész a mobil szórakoztató-központ.
Természetesen a rendszerben még vannak gyenge pontok, vannak gyerekbetegségek, de aki tud kompromisszumot kötni, az egy windowsos *PDA*-val egyenértékű eszközt kaphat. Idővel aztán sokkal többre is képessé válhat ez a rendszer. Akinek van kedve és megfelelő eszköze, az próbálja ki, ennyit szerintem megér.

Illés Viktor (viktor@illesviktor.hu) 25 éves,
az ***Assixo KFT*** munkatársa

Kávéfőzés lépésről lépésre (2. rész)

Az öröklődés, avagy mindig legyen nálad egy angol szótár ha programozol.

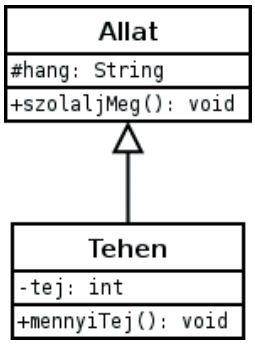
Múlt hónapban feltelepítettük a *Java* fejlesztői környezet 5.0-ás változatát, megírtuk a kötelezőnek számító „Helló világ!” alkalmazást, végül egy tehén, egy ló és egy kutya képében megismertük az objektum fogalmát. Megtudtuk, hogy az objektum nem más, mint egy osztály példánya. Az osztály általános leírást jelent, az objektum ennek az általánosításnak egy megvalósítása. Szó volt még a tagváltozókról, vagy tulajdonságokról, és az ezeken műveletet végző tagfüggvényekről, azaz metódusokról. Ezek igen közel állnak a hagyományos programozási nyelvek változó- és függvényfogalmához. Elevenítsük fel az állatok hangját utánzó program `main(String[])` tagfüggvényt:

```
...
public static void main(String[] args) {
   Allat tehen = new Allat(“múúú”);
   Allat lo = new Allat(“nyihaha”);
   Allat kutya = new Allat(“vauvau”);
   tehen.szolaljMeg();
   lo.szolaljMeg();
   kutya.szolaljMeg();
}
...
```

A metódus nyilvános (`public`), ami azt jelenti, hogy az osztályon kívülről elérhető. Továbbá statikus (`static`), ami azt jelenti, hogy az osztály minden példányára közös, és ezért akár példányosítás nélkül is használható. Mivel ez jelenti az alkalmazás belépési pontját, ennek szükségképpen így kell lennie. Nem rendelkezik visszatérési értékkel (`void`), paraméterként pedig szövegfüzérek tömbjét kapja (`String[]`), mely a parancssori paramétereket tartalmazza. A C-hez szokott szemnek talán furcsa, de ennek mindig így kell lennie, nem hozható létre belépési pont például `int` típusú visszatérési értékkel. A tagfüggvényben először létrehozunk három változót, melyek `Allat` típusúak, és rögtön kezdőértéket is kapnak egy-egy példányosítás révén. A `tehen`, a `lo` és a `kutya` úgynevezett referenciák, segítségükkel objektumokra hivatkozhatunk. Az objektumok létrehozása az osztály konstruktorának meghívásával történik, és ezen keresztül állítjuk be a hangjukat. Végül az objektumoknak egyesével meghívjuk a `szolaljMeg()` metódusát, így a képernyőn megjelennek az állatok hangjai. Teljhatalmú programozóként állatokat hoztunk létre, és láttuk, hogy ez jó. Mégis, a *Java* mögött álló szemlélet azt sugallja, hogy a megoldás még nem tökéletes. Bármilyen objektumközpontú alkalmazás fejlesztéséhez modellalkotáson keresztül vezet az út. Az általunk használt modellben minden állat ugyanazokkal a tulajdonságokkal rendelkezik. Így, ha a megbízónk tehenek esetében meg szeretné jeleníteni a naponta adott tej mennyiségét, már gondban lennénk. Az `Allat` osztályt nem bővíthetjük ilyen tulajdonsággal, mert azzal azt állítanánk, hogy lovaink és kutyáink is tejtermelő jószágok. Kézenfekvő megoldásnak tűnik, hogy képviseljen a tehén önálló osztályt. Kérdés viszont, hogy mi legyen a meglévő tagváltozókkal és tagfüggvényekkel. Egyszerű, másolás-beillesztés módszerrel kaphatunk egy megoldást, de érezhető, hogy ez nem csak csúnya, de rossz hatékonyságú kódot eredményező módszert is jelentene. Az objektumközpontúság egyik nagy előnye pedig a nagy mértékű kódújrafelhasználás, aminek itt valahogy érvényt kell szerezni. Ekkor lép képbe az ***öröklődés (inheritance)***.

A tehén egy állat, de...

Öröklődésről akkor beszélünk, amikor egy meglévő osztályból kiterjesztéssel egy új osztályt hozunk létre. Az eredeti, vagyis az ősosztály megfelelő láthatóságú elemeit megörökli az új, mintha csak saját tulajdonságokról és metódusokról lenne szó. Az egyetlen kérdés a láthatóság, de mielőtt elvesznénk ennek részleteiben, fogadjuk meg azt a bölcsességet, mely szerint egy kép többet mond ezer szónál, és rajzoljuk le a modellt, mielőtt megvalósítanánk.
Az ***UML*** a ***Unified Modeling Language*** rövidítése, és pont az, amit a neve is mutat: egységesített modellező nyelv. Objektumközpontú problémák leírására szolgál, programozási nyelvtől és környezettől függetlenül. A szabvány által bevezetett diagrammok segítségével könnyen emészthetően fogalmazhatók meg objektumközpontú állítások, melyek megvalósítása egy-egy programozási nyelvben már kevés önálló ötletet igénylő kódolási feladatot jelent. Nem célom az ***UML*** részletes bemutatása, csak annyira használom, amennyire segítheti az objektumközpontúság alapfogalmainak megértését. Lássunk egy ***UML*** diagrammot az öröklődésre.
Az ábrán két osztály látható, ezeket három rekeszre osztott dobozok jelképezik. Az első rekesz az osztály nevét, a második a tulajdonságokat, a harmadik a műveleteket mutatja. Az üres hegyű, folytonos vonalú nyíl mutatja az öröklődést. A nyíl irányára több magyarázat is létezik. Szerintem a legfontosabb annak a ténynek a hangsúlyozása, hogy ennél a kapcsolatnál a `Tehen` osztály hivatkozik az `Allat`-ra, és az ősosztály semmilyen formában nem értesül az öröklődésről, pusztán elszenvedi azt. Az osztályok elemei mögött, egy ket-



tőspont után azok típusa látható. Az elemek nevét pedig egy most minket jobban foglalkoztató jelölés előzi meg. Egyetlen különleges karakter, ami a láthatóságot mutatja. A + a nyilvános, a – a személyes, a # pedig a védett (`protected`) tagváltozót vagy -függvényt jelenti.
Ez utóbbiról még nem volt szó. Nézzük meg, hogyan alakul a láthatóság öröklődés esetén. Egy nyilvános elem mindenki számára szabadon hozzáférhető, és ezen az öröklődés nem változtat. Egy személyes elem ezzel szemben a teljes elrejtés megvalósítása. Ha származtatunk egy osztályt, abban nem lesz látható egy személyesre állított elem. Érezhető, hogy szükség van egy köztes megoldásra a láthatóság tekintetében. Ez a védett, ami idegen osztályoknak továbbra sem elérhető, öröklődés esetén pedig megtartja védett tulajdonságát, ezért tetszőleges számú öröklés után is látható osztályon belülről.
Ezért, noha a múlt hónapban személyesre állítottuk az `Allat` osztály `hang` tagváltozóját, ennek láthatóságát védettre kell állítani. Nagyon fontos, hogy ezt nem a `szolaljMeg()` metódus érdekében tesszük! A `Tehen` örökli ezt a metódust, és az meghívható egy `Tehen` típusú objektum esetén. A metódus használja a `hang` változót, de a láthatóság kérdésének eldöntésekor az számít, hogy a művelet hol helyezkedik el, és nem az, hogy melyik osztály példányáról van szó.

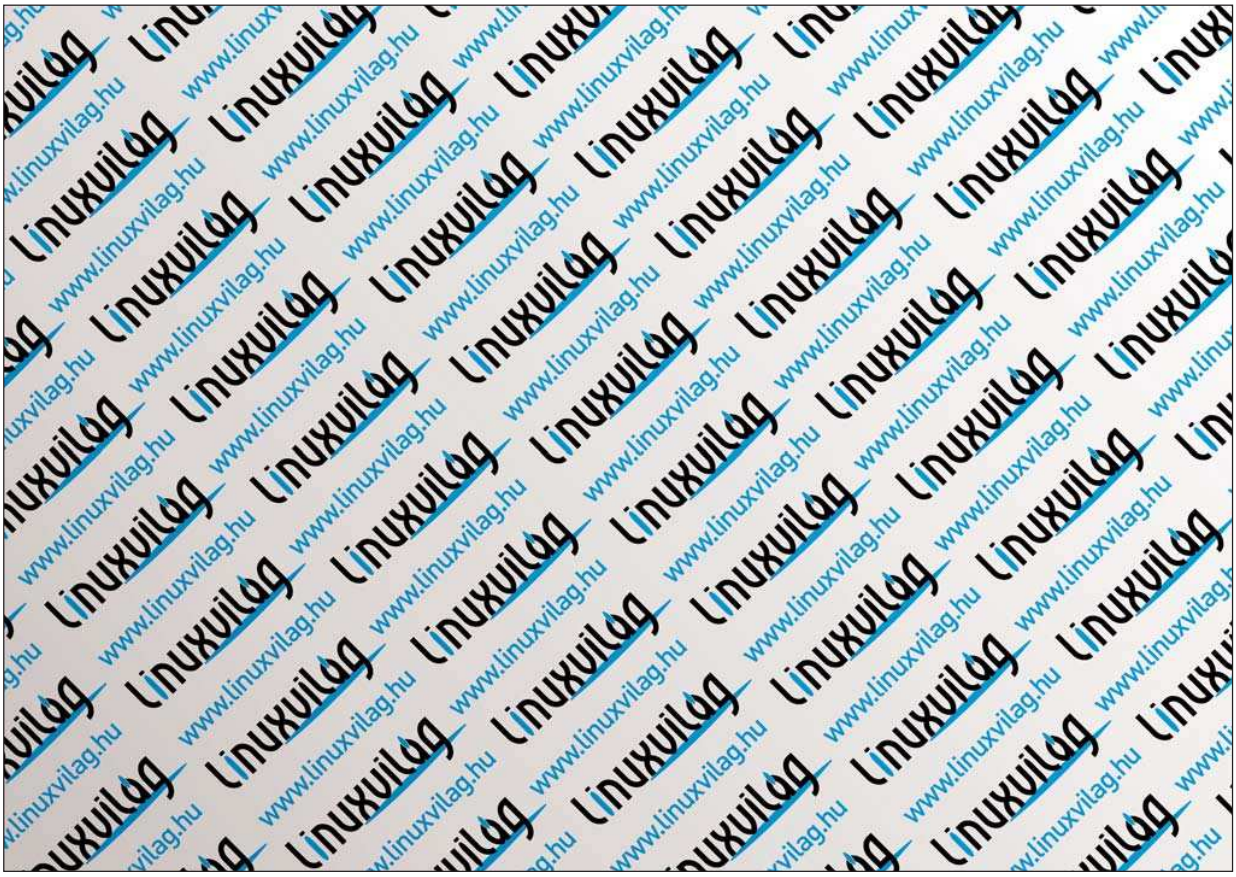
Ha nem lenne olyan tagfüggvény a `Tehen` osztályban, amely felhasználja a hang tulajdonságot, nem lenne szükség a védett láthatóságra. Viszont ha egy tehéntől megkérdezzük, hogy hány liter tejet ad, hozzáteszi, hogy „múúú”, ezért kell a `protected` módosító.
Lássuk a megvalósítást. Két osztályunk, így két forrásállományunk lesz. Az első a már múlt hónapban is látott `Allat.java`, azzal a módosítással, hogy a `hang protected`:

```
/**
* Ez az osztaly egy allatot ir le.
* Tulajdonsaga a hangja.
* Meg lehet szolaltatni.
*/
public class Allat {

    /**
    * Az allat hangja.
    */
    protected String hang;

    /**
    * Letrehoz egy uj allatot
    * a megadott hanggal.
    */
    public Allat(String h) {
       hang = h;
    }

    /**
```

```
    * Kiirja a kepernyore az
    * allat hangjat.
    */
    public void szolaljMeg() {
        System.out.println(hang);
    }

    /**
    * A program belepesi pontja.
    * Letrehoz harom allatot, es
    * megszolaltatja oket.
    */
    public static void main(String[] args) {
        Tehen tehen = new Tehen("múúú", 20);
        Allat lo = new Allat("nyihaha");
        Allat kutya = new Allat("vauvau");
        tehen.szolaljMeg();
        tehen.mennyiTej();
        lo.szolaljMeg();
        kutya.szolaljMeg();
    }

}
```

Következzen a ***Tehen.java*** állomány:

```
/**
* Ez az osztaly egy tehenet ir le
* Tulajdonsaga a tej mennyisege, amit egy nap ad.
* Ezt ki lehet iratni a kepernyore.
*/
public class Tehen extends Allat {

        /**
        * A naponta adott tej mennyisege.
        */
        private int tej;

        /**
        * Letrehoz egy uj tehenet a megadott
        * hanggal, és napi tej mennyiseggel
        */
        public Tehen(String h, int t) {
                super(h);
                tej = t;
        }

        /**
        * Kiirja a kepernyore a napi
        * tej mennyiseget
        */
        public void mennyiTej() {
                System.out.println(tej + " liter,
                ➥" + hang);
        }

}
```

A fordítás és futtatás a megszokott módon történik:

```
$ javac Allat.java Tehen.java
$ java Allat
```

Az új osztály bevezetése akkor nyer értelmet, ha használatba is vesszük, így a program belépési pontja is változott. Tehenet immár a `Tehen` osztály példányosításával hozunk létre. Az öröklés miatt meghívhatjuk az ősosztályban szereplő `szolaljMeg()` mellett a kibővítés révén nyert `mennyiTej()` metódust is. Így a képernyőn az alábbi kimenet látható:

```
múúú
20 liter, múúú
nyihaha
vauvau
```

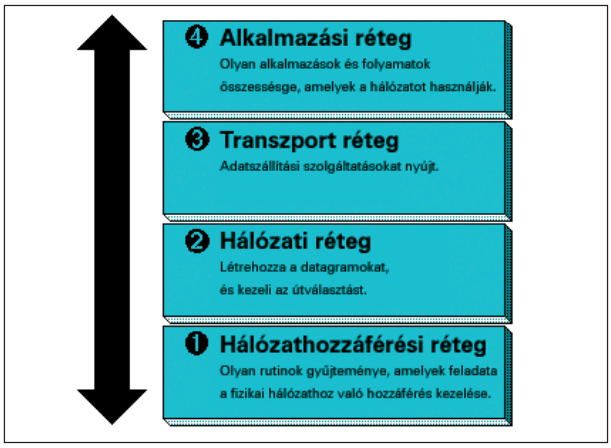
Időzzünk el még egy keveset az öröklődés példaképét jelentő `Tehen` osztály elemzésével. Magának a kapcsolatnak a kifejezése nagyon egyszerű, csupán az osztály fejében kell ezt jeleznünk az `extends` (kiterjeszt) kulcsszó segítségével. Az egyetlen kérdés az, hogy pontosan mi is történik a példányosításkor meghívódó konstruktorokkal. Nyilvánvalóan akkor hasznos az öröklődés, ha a meglévő metódusokat nem kell újraírni, így a tagváltozókat értékkel ellátó konstruktort sem. Lehet valahogy hivatkozni az ős kosntruktorára?
A válasz igen, de nem magától értetődő, hogy hogyan. A konstruktor különleges függvény abban a tekintetben, hogy minden más metódussal ellentétben nem jelezzük a visszatérési értékét, hiszen meghívásakor objektum készül, amit viszont nem a függvény ad. Pontosan ezért különleges bánásmódot igényel. Meghívása a `super` hivatkozással történik, melyet az ős konstruktorának megfelelően kell paraméterezni.
Két dolgot hangsúlyoznék ezzel kapcsolatban. Ha az ősosztály tartalmazna paraméter nélküli konstruktort, és a származtatott osztály konstruktorában nem lenne `super` hívás, akkor a származtatott osztály példányosításakor először önműködően meghívódna az ősosztály kosntruktora. Továbbá nagyon fontos, hogy ha így hivatkozunk egy ősosztálybeli konstruktorra, a `super` hívásnak a legelső műveletnek kell lennie, egyébként a fordító hibát is jelez.
A `mennyiTej()` metódusban a kiíratásnál a + operátor segítségével fűzzük eggyé a szövegfüzéreket. Ez egy igen kényelmes, és könnyen olvasható megoldást jelent. Ugyanakkor ez azt is jelenti, hogy a + több értelemmel is bír, hiszen számok aritmetikai összeadása mellett `String` típusú objektumok összefűzését is lehetővé teszi. Ez egy beépített megoldás, amit az objektumközpontúság szakzsargonjával élve operátor túlterhelésnek hívunk (`operator overload`). Jó tudni, hogy egyes nyelvek, például a *C++* ezt a programozó számára is lehetővé teszik. Jelen helyzetre lefordítva, ha azt mondanánk, hogy két tehén egyenlősége akkor teljesül, ha ugyanannyi tejet adnak, túlterhelhetnénk az == operátort, hogy kényelmesen hasonlítsunk össze két `Tehen` típusú objektumot. Ugyanez *Java*-ban hatékonysági okokból nem megvalósítható.
Mindezek alapján már bátran belegondolhatunk, mi is az ami miatt hasznos az objektumközpontú tervezés. Egy kellően átgondolt modell segítségével olyan alkalmazást építhetünk, ami nem csak könnyebben átlátható, de egyszerűen bővíthető is. A sorozattal kapcsolatban várom az észrevételeket, javaslatokat. Következő hónapban az interfészek lesznek terítéken.

Fülöp Balázs (bigwig42@gmail.com)

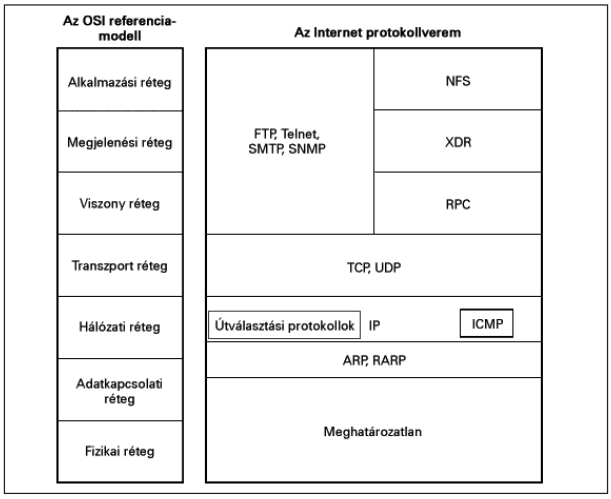
Számítógép-hálózatok (19. rész)
Az Internet hálózati rétege, az IP protokoll, címzések és alhálózatok

A sorozat elmúlt néhány részében áttekintettük, mi is a feladata, illetve miként is működik a hálózati réteg. Most egy konkrét megvalósítást veszünk szemügyre, amelyen megnézzük, miként is mennek a dolgok a gyakorlatban. Ebben a hónapban tehát témánk tehát az Internet hálózati rétege és annak protokolljai.

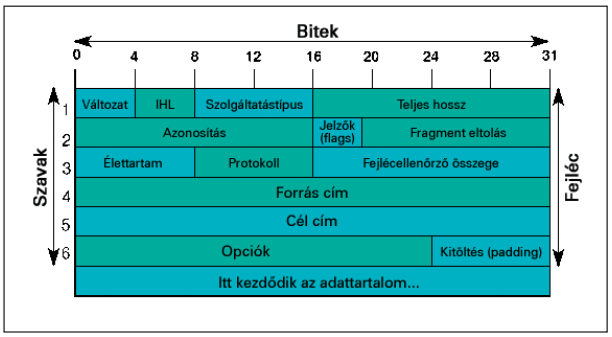
A hálózati réteg feladata nem más, mint a szállítási rétegtől kapott adatokat eljuttassa a címzettig. Mindezt úgy kell megvalósítania, hogy a szállítási réteg szempontjából teljesen mindegy legyen, hogy a cél a forrással egy hálózatban van-e, vagy sem. Magyarul a szállítási réteg csak az elküldendő adatot és a cél címét adja meg, minden más a hálózati rétegre van bízva.
Az Interneten zajló kommunikáció tehát a következőképp zajlik (*1. ábra*): a ***szállítási réteg (transport layer)*** kap egy adatfolyamot az ***alkalmazási rétegtől (application layer)***. Ezt az adatfolyamot blokkokra, úgynevezett datagramokra bontja, majd átadja a ***hálózati rétegnek (network layer)***, amely gondoskodik azok célbajuttatásáról. Látható, hogy az Internet felépítése nem követi az ***OSI*** modellt, amelyre sorozatunkat is építettük. A fizikai és az adatkapcsolati réteg ugyanis össze van vonva, továbbá hiányzik a viszont és a megjelenítési réteg (ezek egyes feladatai a szállítási, mások pedig az alkalmazási rétegben kerülnek megvalósításra).
A *2. ábrán* láthatjuk az Internet protokolljainak egymáshoz való viszonyát. Ezek közül a mezei felhasználók csak az alkalmazás rétegbeli protokollokkal kerülnek közelebbi kapcsolatba, mint például az ***FTP***-vel vagy az ábrán ugyan fel nem tüntetett, ám manapság leggyakrabban használt ***HTTP***-vel, amely a weboldalak lehívásában játszik szerepet. A ***TCP (Transmission Control Protocol)*** két gép közötti megbízható bájtfolyam alapú kommunikáció kialakítására alkalmas protokoll, az ***UDP (User Datagram Protocol)*** pedig ennek megbízhatatlan „párja”. Ezek a szállítási réteg protokolljai, a későbbiekben részletesen is foglalkozunk velük. Mi most azonban még maradunk a hálózati rétegnél, és azok protokolljainál. Közülük a legjelentősebb az ***IP (Internet Protocol)***, amely a gépek közötti adatátvitelt végzi. Fontosak még a vezérlőprotokollok, mint például az ***ICMP (Internet Control Message Protocol)***, vagy a logikailag az adatkapcsolati és a hálózati réteg határán elhelyezkedő ***ARP*** és ***RARP***.



1. ábra Az Internet rétegei. Ez a felépítés némiképp eltér az OSI modelltől, hiszen hiányzik az adatkapcsolati, a viszony és a megjelenési réteg.



2. ábra Az Internet protokolljai



3. ábra Az IP csomagok fejlécének felépítése

Az ennél mélyebb szinten lévő dolgokról az Internet szabványa már semmit sem mond. Itt az adott ***LAN***-ra van bízva, hogy miként valósítja meg a benne lévő gépek közötti adatátvitelt. Az ***Internetbe*** tehát ugyanúgy beköthetünk például egy vezérjeles gyűrűt, mint egy ***Ethernet*** hálózatot. Most pedig vegyük kicsit részletesebben is szemügyre az ***Internet*** hálózati rétegének protokolljait!

Az IP (Internet Protocol)

A szállítási réteg által előállított ***TCP*** és ***UDP*** datagramok ***IP*** csomagok belsejében utaznak a világhálón. Egy ***IP*** csomag két részből áll: magából a szállított adatból és egy fejlécből. A fejléc szerkezetét láthatjuk a *3. ábrán*.

Amikor két különböző számítógép, például egy ***SPARC*** és egy ***Pentium*** processzorral felszerelt masina bitsorozatokat szeretne egymással cserélni, akkor rögtön félreértések adódnak. A ***Pentium*** ugyanis alsó vég szerint tárolja a biteket, azaz mindig a legkisebb helyiértékű bitet küldi elsőnek. A ***SPARC***-nál ez pont fordítva történik. Ha a ***Pentium*** azt akarja mondani partnerének, hogy „41”, akkor ő az 100101 bitsorozatot fogja küldeni, amit a ***SPARC*** 37-nek fog értelmezni. Ezért először meg kell egyezni abban, hogy a biteket alsó- vagy felső vég szerint továbbítjuk. Az ***IP*** protokoll felső vég szerint tárolja az adatokat a fejlécében. (Ezért ha ***Pentium*** processzorra írunk hálózati alkalmazást, figyelnünk kell arra, hogy a cél címét először átalakítsuk felsővégűre, majd csak utána adjuk át a szállítási rétegnek).

Most nézzük sorra, mi mit is jelent a fejlécben. A verzió értelemszerűen az ***IP*** protokoll verzióját adja meg, pontosabban azt, hogy az adott csomag az ***IP*** protokoll mely változatához tartozik. Az ***IHL*** mező a fejléc utolsó részének, az opciók méretét adja meg. Erre azért van szükség, mert ennek a mezőnek nincs kötött mérete, hossza csomagonként változó lehet. Mindenesetre 60 bájtnál nagyobb nem lehet.

A ***szolgáltatás típusa (type of service)*** eredetileg arra hivatott, hogy az alhálózat számára hordozzon információkat a továbbításánál felmerülő kérdésekkel kapcsolatban. Ilyen lehet például a csomag prioritása (erre az ***IP*** 8 szintet biztosít), vagy például az, hogy gyorsan, vagy inkább biztonságosan kívánjuk a csomagot céljához eljuttatni. Példaként most is felhozhatjuk az előző részben leírtakat: egy hálózaton keresztüli videónézésnél fontosabb, hogy a csomagok sebessége ne lassuljon le, ugyanakkor egy fájl átvitelekor inkább az a jobb, ha a csomagok sértetlenül érkeznek meg. Az igazsághoz azonban az is hozzátartozik, hogy az útválasztók többsége nem foglalkozik az itt megadott paraméterekkel. Ez gyakorlatilag azt jelenti, hogy bármit is írunk erre a mezőre, nem lesz kihatással a csomagunk célbajutásának módjára.

A ***teljes hossz (total length)*** az egész csomag méretét adja meg, beleértve a fejlécet és a szállított adatokat is. Mivel ez egy két bájtos mező, ebből következően egy csomag nem haladhatja meg a 65535 bájtot (egy bájt híján 64 kilobájt). Ez első hallásra soknak tűnhet, de a sávszélesség növekedésével előbb-utóbb kevésnek számítanak majd, és nem lesz gazdaságos ilyen „kis” csomagmérettel dolgozni.

Az útválasztók időnként feldarabolják a datagramokat. Az ***azonosítás (identification)*** mező adja meg, hogy az adott csomag melyik datagram része. A ***jelzők (flags)*** három bitből áll, amelyek közül az első kihasználatlan, a második az úgynevezett ***DF (Don’t Fragment – ne darabold!)*** bit. Ez megtiltja az útválasztók számára, hogy az adott csomagot több részre szabdalják. Ezt a bitet akkor szokás beállítani, amikor a célállomás valami oknál fogva nem képes több darabból összeállítani az eredeti datagramot. A darabolás megtiltásának azonban ára van: elképzelhető, hogy a csomag egy darabban csak kerülőúton keresztül érhet célba. A jelzők mező harmadik bitje az ***MF (More Fragments – még több darab)***. Ez azt jelzi a célállomás számára, hogy a datagram még nem ért át teljesen, újabb darabokra kell számítani.

Ahhoz, hogy a gép rekonstruálhassa a számára küldött datagramot, a darabok sorrendjét is ismernie kell (hiszen semmi sem garantálja, hogy a csomagok útnak indulásuk sorrendjében érkeznek meg). Erre szolgál a ***darabeltolás (fragment offset)*** mező, amely megmondja, az adott darab mely részét képezi a datagramnak. Az utolsó darabot leszámítva ennek a mezőnek az értéke mindig a 8 valamely egész számú többszöröse. Mivel az egész mező összesen 13 bájt méretű, ezért egy datagramot 8192 darabnál többre nem oszthatunk.

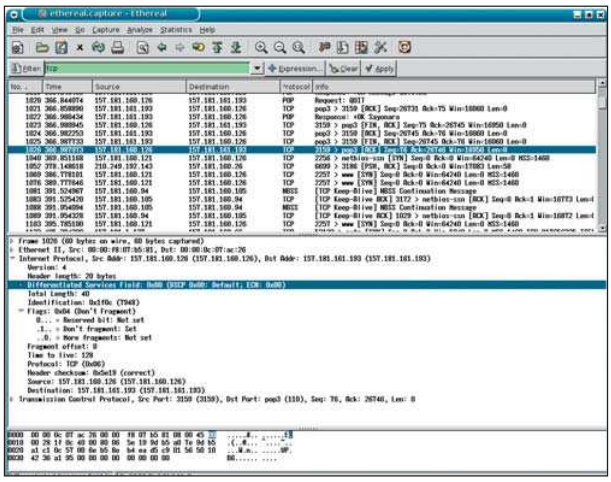
Az alhálózaton örökre bolyongásra ítélt csomagok elkerülésére szolgál az ***élettartam (time to live)*** mező, amely megmondja, még mennyi ideig élhet az adott csomag. Ha ez az érték nullára csökken, az útválasztók a csomagot eldobják. A szabvány szerint az élettartamot másodpercben kell megadni, de a gyakorlatban inkább ugrás-számban határozzák meg, azaz minden útbaeső útválasztó eggyel csökkenti a mező értékét. Ha egy csomag ideje lejár, az útválasztó, amelyik a csomagok eldobta, egy értesítőt küld a feladónak a történtekről.

A protokoll rész arról árulkodik, hogy a csomag milyen típusú datagramot hordoz, például ***TCP***, ***UDP***, stb. Ez a célállomás számára fontos, hiszen annak hálózati rétege innen tudja, hogy mely szállítási réteghez tartozó folyamatnak kell a beérkező adatokat átadnia.

A ***fejlécellenőrző összeg (header checksum)*** értelemszerűen a fejléc sértetlenségének megállapítására szolgál. Mivel a fejléc élettartam mezője minden ugrásnál változik, ezért az útválasztóknak a csomag továbbítása előtt az ellenőrző összeges újra ki kell számolniuk.

A ***forrás és a cél címe (source/destination address)*** megmondja, ki a címzett és ki a feladó. Az ***Interneten*** a gépek úgynevezett ***IP*** címekkel vannak azonosítva, erre mindjárt részletesen is kitérünk.

4. ábra A traceroute parancs segítségével feltérképezhetjük, hogy milyen útválasztókon keresztül érhetjük el a megadott célállomást



5. ábra Egy csomag belső felépítése

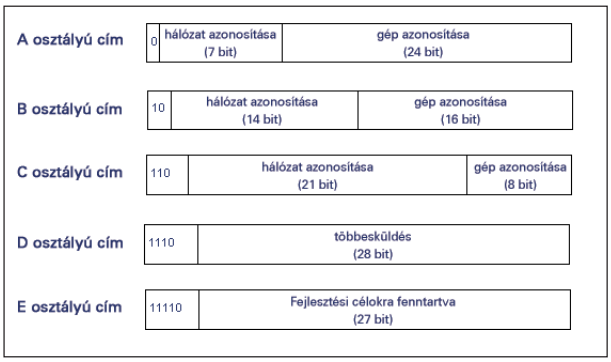
Az ***opciók (options)*** rész eredetileg arra szolgált, hogy később jelentősebb változtatások nélkül bővíthessék az *IP* protokoll által nyújtott szolgáltatásokat. Mint már említettük, ez a rész változó méretű, lehet teljesen üres is (de 40 bájtál semmiképp sem több), így csak akkor kell több helyet lefoglalni a fejrész számára, ha az opció mező valóban felhasználásra is kerül.
Az *IP* tervezése óta öt dologgal bővítették ki az *IP* fejlécét, illetve pontosabban az opció mezőt. Ezek közül az első a biztonság, amellyel meghatározhatjuk a csomagban szállított datagram titkosságát. Az eredeti célkitűzés az volt, hogy bizonyos titkosnak számító adatokat csak meghatározott útválasztókon keresztül lehessen továbbítani. Ezzel elkerülhető például az, hogy katonai titkokat hordozó csomagjaink áthaladjanak ellenséges (vagy legalábbis nem túl barátságos) országok útválasztóin. A ma használatban lévő útválasztók nem támogatják ezt a szolgáltatást.
Egy másik opció a szigorú forrás általi forgalomirányítás, amikor a datagramot küldő gép határozza meg, hogy a csomagnak milyen útvonalon kell eljutnia a célállomásáig. Ilyenkor az opciórészben meg kell adni a pontos útvonalat (útválasztók IP címeinek sorozatát), amelyen a csomagnak haladnia kell. Ennek egy „enyhébb" változata a laza forrás általi forgalomirányítás, amikor csak azoknak az útválasztóknak a címeit mondjuk meg, amelyeken a csomagnak biztosan át kell haladnia. Ezzel az opcióval lehetőség nyílik arra, hogy csomagjainkkal elkerüljünk bizonyos térségeket.
A negyedik opció az útvonal feljegyzése. Ennek hatására az útválasztók az átmenő csomagok opció mezejébe beírják saját címüket, így a célállomás rekonstruálhatja, milyen útvonalon érkeztek meg hozzá a csomagok. A ***traceroute*** parancs (*4. ábra*) is hasonló szolgáltatást nyújt, segítségével kilistázhatjuk, milyen útválasztókon kell keresztülmennünk ahhoz, hogy elérjünk egy megadott gépet. A különbség csak az, hogy a ***traceroute*** nem használja az útvonal feljegyzése opciót, teljesen más elven működik. Az az igazság, hogy nem is lehet ezt a módszert erre a célra használni, ugyanis ezt még akkor találták ki, amikor az Internet mérete még jóval kisebb volt, és 9 ugráson belül bárhova el lehetett jutni. Ma már más a helyzet. A 4. ábrán látható, hogy például egy népszerű amerikai keresőportál eléréséhez a csomagoknak legalább 17 útválasztón kell keresztülverekedniük magukat. Ilyen hosszú útvonal pedig már aligha fér el 40 bájton.
A ***traceroute*** inkább azt használja ki, hogy az útválasztók visszajeleznek, ha egy általunk küldött csomagnak lejár az életideje, és el kell dobniuk. Így kiküld különböző ám kis élettartamú csomagokat (amelyek annyi idő alatt biztosan nem jutnak el céljukig), és amelyik útválasztó visszajelzi a csomag halálhírét, az biztos része a csomag útvonalának.
Végezetül megemlítjük még az utolsó opciót is, az időbélyeget, amely szinte teljesen megegyezik az előzővel, csak azzal a különbséggel, hogy az útválasztók nem csak az *IP* címüket, hanem a csomag beérkezésének idejét is feljegyzi. Ezáltal pontosabb képet kaphatunk a csomagok alhálózatbeli terjedéséről.

Az ICMP (Internet Control Message Protocol)

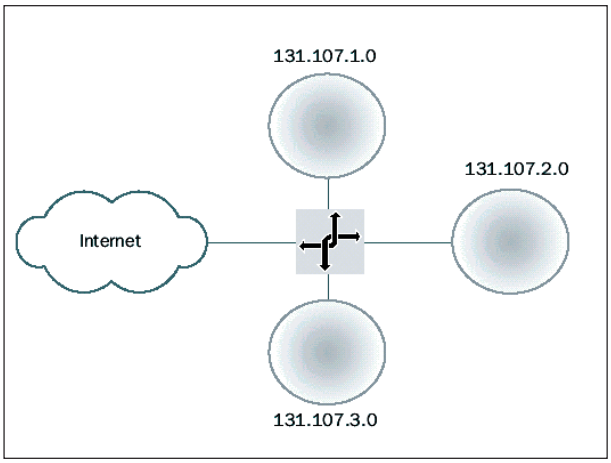
Az *IP*-n kívül más protokollokat is használ az ***Internet*** hálózati rétege. Ezek közül az egyik legfontosabb az *ICMP*, amelynek két funkciója van. Először is lehetővé teszi hogy mérhessük a hálózat bizonyos tulajdonságait. Másodszor lehetőséget biztosít az útválasztók számára, hogy jelezzék a csomag feladójának, ha valami váratlan esemény következik be.
Ilyen váratlan esemény lehet például az, ha az útválasztó nem találja magát a célt, vagy hibás fejlécű *IP* csomagot kapott. Az *ICMP* üzenetek másik gyakori felhasználása az, amikor egy gép életjeleit kívánjuk megállapítani (például a ***ping*** parancs segítségével). Ilyenkor egy úgynevezett *ECHO_REQUEST* üzenetet küldünk, amelyre a gép – amennyiben életben van – egy *ECHO_REPLY ICMP* üzenettel válaszol.

Címzés az Interneten

Ahhoz, hogy a hálózati réteg megtalálja a csomagok célállomását, minden gépnek (és útválasztók) rendelkeznie kell egy vagy több egyedi azonosítóval. Az ***Interneten*** ez az azonosító az *IP* cím, amely egy 32 bites összeg.



6. ábra Az IP címosztályok



7. ábra Alhálózatok egy lehetséges felosztása

Ezt a számot általában bájtonként decimálisan, pontonként elválasztva szokás megadni, például: 195.38.96.96. A valamivel több mint 4 milliárd lehetséges ***IP*** címet öt csoportra oszthatjuk (*6. ábra*). Ezek közül az első három osztály (A, B és C) ***IP*** címei azonosítanak gépeket. Az egy hálózatban lévő hosztok hálózati címeinek tehát meg kell egyezniük. A D osztályba tartozó címek a ***többesküldésre (multicasting)***, az E-beli címek pedig későbbi felhasználásra vannak fenntartva.

Hogy egy ***IP*** cím melyik osztályba tartozik, azt az első pár bitje adja meg. Az A osztályú címek például mindig 0-val kezdődnek, ez azt jelenti, hogy az 1.0.0.0 – 127.255.255.255 tartományon belül minden ***IP*** cím A típusú. Az A, B és C osztályok közötti különbség az, hogy mennyi bitet használnak magának a hálózat, és mennyi bitet a hálózatban lévő egyes gépek azonosítására. Az A osztály esetében összesen 126 hálózatot lehet megkülönböztetni, viszont ezek a hálózatok akár 16 millió gépből is állhatnak. (Persze ez azt is jelenti, hogy összesen csak 126 darab A osztályú címcsoport osztható ki). Ezzel szemben a C típusú címek már 21 bitet használnak a hálózaton azonosítására, viszont csak 256 (gyakorlatban 254, lásd később) darab gépet tartalmazhatnak.

Vannak speciális célra fenntartott ***IP*** címek is, amelyek nem oszthatók ki a gépek számára. Ilyen például a csupa egyes bitből álló (255.255.255.255), amely az adatszórást teszi lehetővé a helyi hálózaton. Azok az ***IP*** címek, amelyeknek a hálózati része csupa nullából áll, mindig az aktuális hálózatra vonatkoznak. A 127.*.*.* tartomány szintén nem osztható ki, ez az úgynevezett ***visszacsatolás (loopback)***, amelyeken keresztül a gép saját magát „érheti el”. Próbáljuk csak megpingelni bármelyik 127-el kezdődő ***IP*** címet, mindig a saját gépünk fog válaszolni. A visszacsatolásnak például akkor vehetjük hasznát, ha ki szeretnénk próbálni egy hálózati alkalmazást anélkül, hogy egy második gépet is igénybe vennénk.

Alhálózatok

Már említettük, hogy az egy hálózatba tartozó gépek hálózati címeiknek meg kell egyezniük. Egy B osztályú címtartományt birtokló szervezet esetében azonban nincs mind a 65 ezer gép egy fizikai hálózaton. Egy egyetem esetében például külön hálózatot képezhetnek az egyes géptermek, illetve a tanszékek számítógépei, és ezek a különálló ***LAN***-ok hidak segítségével vannak összekapcsolva. Felmerül a kérdés, hogy akkor miként osszuk ki az ***IP*** címeket?

Kézenfekvő megoldás lehet az, hogy visszaadjuk a B címtartományunkat, és kérünk annyi C osztályút, ahány hálózatunk van. Ezzel azonban több problémánk adódna, például jelentősen megnehezülne a hálózat menedzselése. A megoldás az ***alhálózatok (subnets)*** létrehozásában rejlik. (Az elnevezés sajnos megtévesztő. Ezeknek az alhálózatoknak semmi köze az útválasztókból álló kommunikációs alhálózathoz).

Az alhálózatok kialakításához az ***IP*** címek gépek azonosítására szolgáló 16 bitjéből leválasztunk valamennyit, amelyek az egyes alhálózatokat fogják megadni. A *7. ábrán* láthatunk is erre egy példát. Tegyük fel, hogy a 131.107.*.* B típusú ***IP*** címtartománnyal rendelkezünk. Itt az utolsó két bájt azonosítja a gépeket. Az ábrán ebből 8 bitet az egyes alhálózatok azonosítására áldozunk. Ekkor összesen 254 alhálózatunk lehet, és mindegyikben 254 gép helyezhető el.

Fontos, hogy ez a felosztás kívülről (egy másik hálózatból) nem látszik. Az ottani szemlélődő úgy látja, hogy nekünk csak egyetlen nagy hálózatunk van. A belső útválasztóink azonban ismerik a hálózatunk belső felépítését, és tudják, melyik gép merre található. Pontosabban csak azt kell tudniuk, hogy az egyes alhálózatok merre találhatóak. Az úgynevezett alhálózati maszk segítségével ugyanis az útválasztók az ***IP*** címből ki tudják számolni, hogy melyik alhálózat felé is kell terelni a csomagot. Az alhálózati maszk egy olyan 32 bites szám, amellyel ha össze ÉS-eljük az ***IP*** címmel, akkor kinullázódik annak a gépet azonosító része.

Az előző példánknál a ***netmask*** minden bizonnyal a 27 darab egyesből, és 8 darab nullából (255.255.255.0) lenne. Ha beérkezik egy csomag, amely a 131.107.12.2-es című gépnek szól, akkor az útválasztó az alhálózati maszkkal „összeéselve” megkapja, hogy a kérdéses gép melyik alhálózatban is található (jelen esetben a végeredmény a 131.107.12.0 lesz).

A következő részben megismerkedünk az ***Internet*** más fontos, szintén a hálózati rétegbe tartozó protokolljaival, például az ARP-al és a RARP-al.

Garzó András
garzoand@interware.hu

L'Intranet Originale

Azt hiszed, nem tudsz egy teljes közösségnek helyt adni nagyjából 64 k-ban? Próbálj ki egy elektronikus hirdetőtábla rendszert!

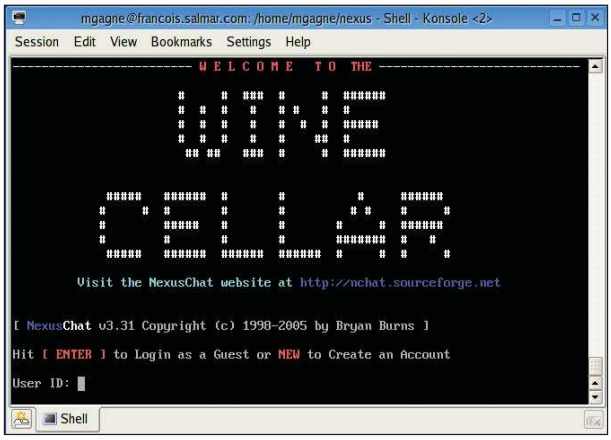
Még szép, hogy nem grafikus, de színek azért vannak, ***mon ami***. Miért? Ugyan már, ***François***, hadd nosztalgiázzak egy kicsit! Amikor láttam, hogy ennek a lapszámnak az intranetek adják a témáját, elgondolkoztam ezen az egész intranet-dolgon, ami tulajdonképpen egy belső hálózat, egy apró magán-univerzum, amihez, ha úgy akarjuk, csak néhányan férhetnek hozzá. Általában azt gondoljuk, hogy ilyesmit inkább vállalatok és hasonló szervezetek használnak, de például az azonos hobbi iránt érdeklődők számára is hasznos lehet. Amikor intranetekről beszélünk, mostanában inkább webes kapcsolatkezelő rendszerekre és portálokra gondolunk.

Quoi? Szöveges a képernyő? Tény. Az intranetek sokkal korábban kialakultak, mint hogy internetezni kezdtünk, ***mon ami***, és a kapcsolattartás bizony szöveges felületen történt. ***Mon Dieu***, elbeszélgetjük az időt! Már meg is érkeztek a vendégeink! Üdvözöllek benneteket, ***mes amis***, helyezzétek magatokat kényelembe, amíg ***François*** megtölti poharatokat! A borospincébe, ***François***! Hozd fel azt a 2003-as ***Coastal Sauvignon Blanc***-t. ***Vite!***

Éppen az eredeti intranetekről meséltem ***François***-nak, ***mes amis***. Éppen csak kiléptem a tinédzserkorból, amikor üzembe helyeztem egy ilyen intranetet a ***Commodore 64***-esemen. Úgy hívták őket, hogy elektronikus hirdetőtáblák, röviden ***BBS***-ek ***(bulletin board system)***. Lényegében annak idején saját ***BBS***-t írtam és tartottam fenn. Egyetlen telefonvonal tartozott hozzá, vagyis egyszerre csak egy felhasználó használhatta. Nem tartozott semmilyen hálózatba, de azért intranet volt, és fennállása csúcsán 40-50-en is használták. A régi szép idők emlékének adóztam, amikor ***BBS*** programokból állítottam össze mai menünket.

Sokan gondolhatják, hogy ma már minden grafikus, és senki nem dolgozik szöveges megjelenésű ***BBS*** programokkal. A valóság azonban az, hogy számos ***BBS*** van még jelenleg is üzemben, és a szükséges programok fejlesztése jelenleg is folyik.

Első fogásunk ***Bryan Burns NexusChatje***. A ***NexusChat***, más néven ***NChat*** egy kiváló, ***BBS***-es stílusú program, különféle felhasználói szintekkel, több szobával, magán és csoportos csevegési lehetőséggel, e-mail küldéssel, üzem közben módosítható beállításokkal, súgóval



1. ábra Telnettel csatlakozva a NexusChat kapujára ezt a bejelentkezési képernyőt kapjuk

és számtalan további szolgáltatással. A ***NexusChat*** futtatásához nincs szükség root jogosultságokra, sőt, telepítése is elvégezhető ezek nélkül. Első lépésként hozzuk létre azt a könyvtárat, ahova telepíteni szeretnénk a csevegési kiszolgálót. Én például létrehoztam egy ***nexuschat*** könyvtárat a saját kezdőkönyvtáramban. A következő lépés a forráscsomag kibontása:

```
tar -xzvf nchat-3.31.tar.gz
cd nchat-3.31
./setup.sh
```

A megválaszolandó kérdések meglehetősen egyszerűek, és néhány kivételtől eltekintve az alapértékeket is nyugodtan elfogadhatjuk. Amikor a parancsfájl rákérdez, hogy hová szeretnénk telepíteni a bináris fájlokat, adjuk meg a korábban létrehozott könyvtár nevét. Az alap adatkönyvtár (alapértéke ***/home/nchat/etc***) ezután a telepítési könyvtár alkönyvtára legyen. A következő kérdés a kapuk száma. Ez határozza meg, hogy egyszerre hányan csatlakozhatnak a csevegési kiszolgálóhoz, alapértéke 15. Miután erre az utolsó kérdésre is válaszoltunk, adjuk ki a `make` parancsot. A fordítás mindössze néhány másodperc, melyet követően létre kell hoznunk a felhasználói adatbázist. Alapesetben 999 felhasználót adhatunk meg.

Ennyi, az `install` parancs elmarad. A végső lépés az ***etc*** könyvtár kézi átmozgatása végső helyére, illetve ezt kell tennünk az ***nchat*** és az ***userdb*** bináris fájlokkal is. Mivel nálam a kiszolgáló a ***/home/marcel/nexuschat*** könyvtárból fut, a következő parancsokat adtam ki:

```
mv etc /home/marcel/nexuschat
mv nchat /home/marcel/nexuschat
mv userdb /home/marcel/nexuschat
```

Lépjünk át a ***NexusChat*** könyvtárába, majd a `userdb -z -s 999` paranccsal készítsük elő a felhasználói adatbázist. Létre kell hoznunk a 000 felhasználót is, akinek a root jelszót kell adnunk. Az alapesetben a 4000-es kapun futó kiszolgáló elindítása a `/elérési_út/nchat` paranccsal történik. Most egy másik terminálról a 000 felhasználóval jelentkezzünk be a csevegési kiszolgálóra:

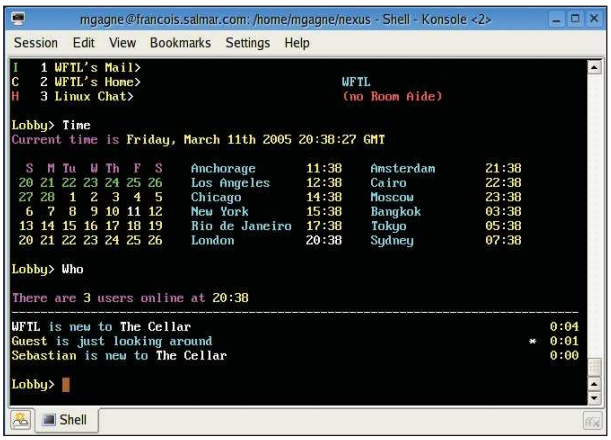
```
telnet kiszolgáló 4000
```

A bejelentkezés után első teendőnk a jelszavunk megváltoztatása. Ezt a `/passwd titok` paranccsal tehetjük meg, ahol a titok az új jelszó. Csatlakozás után, csevegés közben számos további parancs áll rendelkezésünkre, ezek a jelszóváltoztatás parancsához hasonlóan egy perjellel kezdődnek. A parancsok listáját a `/?` utasítással kérhetjük le. Ha valami miatt nem látjuk, amit begépelünk, a `/echo` parancsot kell használnunk.
Ettől a ponttól kezdve vendégek bejelentkezésére is van mód; a vendégként való belépéshez elég lenyomni az ENTERT. A vendégek a `NEW` paranccsal kezdeményezhetik normál felhasználóként való bejegyzésüket, ám bejelentkezni csak azt követően tudnak, hogy a rendszer gazdája jóváhagyta a bejegyzést. Ezen a ponton módjuk nyílik adataik megváltoztatására, illetve csevegésre, bár csak egy korlátozott parancskészlettel. A rendszergazda, vagyis az `nchat` programot futtató személy az állandó felhasználók hozzáadására és az önállóan bejegyzett felhasználók engedélyezésére a felhasználószerkesztőt használhatja, mely a `/ue username` paranccsal indítható el. A művelet a parancssorból is elvégezhető, mégpedig a másik korábban telepített bináris fájllal, a `userdb`-vel. Ha tehát a ***NexusChat*** könyvtárából szeretnénk felhasználót hozzáadni, a következő parancsot kell használnunk:

```
./userdb -a user -u -l 003 -h Francois -p 123
➥-t 3600
```

A kapcsolók jelentése a következő:
`-a` – felhasználói fiókot adunk hozzá (létezik rendszergazdai is)
`-u` – frissítjük az adatbázist
`-l` – a felhasználó sorszáma 003
`-h` – a felhasználó neve ***Francois***
`-p` – a hozzárendelt jelszó 123
`-t` – a kapcsolatok időtúllépési ideje 3600 másodperc

Ha kapcsolók nélkül csupán a `userdb` parancsot adjuk ki, akkor a rendelkezésünkre álló kapcsolók listája jelenik meg.



2. ábra A bbs100 elektronikus hirdetőtábla rendszer csevegőszobáinak és naptárainak fenntartásához mindössze néhány kB memóriára van szükség

Már említettem, hogy az alapértelmezett kapuszám a 4000-es. Ezt és néhány további beállítást a ***etc/nchatrc*** fájl átírásával lehet módosítani. Érdemes például a `chat_name`-et az általunk kívántra, például ***BBS***-ünk nevére módosítani. Bizonyos beállítások, mint például az `ask_ansi = true` megjegyzésbe vannak téve. Bár a legtöbb terminál gond nélkül kezeli az ***ANSI*** színeket, ha gondoljuk, akkor bejelentkezéskor kínáljuk fel a felhasználóknak a választás lehetőségét.
Az ***etc*** könyvtárban további érdekes fájlokat is találunk. Az ***nc_login*** fájl például azt határozza meg, hogy a felhasználók mit látnak bejelentkezéskor. Hasonló célt szolgál az ***nc_ansi_login*** és az ***nc_motd*** (napi üzenet) is.
A ***NexusChat*** szórakoztató kiegészítő, könnyű használni, felügyelete csak kevés tennivalóval jár. Rendkívül rugalmas, a felhasználók és a csevegőszobák létrehozása egyaránt egyszerű. Mivel alapszintű e-mail szolgáltatásokat is biztosít, azoknak a felhasználóknak is hagyhatunk magánüzeneteket, akik éppen nincsenek bejelentkezve. Aki a ***NexusChat*** kipróbálása mellett dönt, feltétlenül látogasson el a program weboldalára is, ahol megtalálja szolgáltatásainak teljes listáját (lásd az internetes forrásokat).
Míg ***François*** újratölti a poharakat, nézzünk egy másik ***BBS***-es példát. Vannak programok, melyek jóval több lehetőséget kínálnak, mint a ***NexusChat***, például teljes értékű üzenetkezelést, összetettebb szobalétrehozást – külön szobákat az üzenetek továbbítására és a csevegésre –, statisztikák készítését, világórát, naptárat stb. Ilyen például a ***Walter de Jong*** által fejlesztett ***bbs100***.
A ***bbs100***-at forrásból kell lefordítanunk, amit viszont a ***bbs100*** webhelyéről tölthetünk le (lásd a forrásokat). A fordítás és a telepítés menete egyszerű, bár a lépések kissé furcsának tűnhetnek:

```
tar -xzvf bbs100-2.1.tar.gz
cd bbs100-2.1/src
./configure –prefix=/home/bbs100
make dep
make
make install
```

A fenti `prefix` kapcsolóra külön fel szeretném hívni a figyelmet. Fontos, hogy ne az alapértelmezett */usr/local* könyvtárat használjuk, mert a ***BBS***-nek írási jogra van szüksége a megadott könyvtárhoz, ami a */usr/local* esetében erősen kérdéses. A `make install` parancsot nem rootként futtattam, ugyanis erre semmi szükség. Természetesen ilyenkor ellenőriznünk kell, hogy van-e írási jogunk arra a könyvtárra, ahová a telepítést el szeretnénk végezni. Nálam ez a ***BBS*** a */home/bbs100* könyvtárba került.
Ha túljutottunk a telepítésen, váltsunk át a telepítési könyvtárba (esetemben ez a */home/bbs100*), majd kedvenc szövegszerkesztőnkben nyissuk meg az *etc/param* fájlt. Néhány beállítást azonnal írjunk is át, ilyen például a ***BBS*** neve, a csatlakozások fogadására használt kapu száma és a telepítési könyvtár:

```
bbs_name                A pince
port_number             12345
basedir                 /home/bbs100
```

Mielőtt továbblépnénk, röviden ismerkedjünk meg az *etc* könyvtárban található egyéb fájlokkal. Találunk köztük üdvözlőképernyőt, napi üzenetet, súgófájlokat, rendszerszabályokat, amelyek az első bejelentkezésnél jelennek meg, valamint számos további érdekességet.
Már majdnem végeztünk! Mivel ***Francois*** lett a rendszergazda, valamilyen jelszót kell adnunk neki.
A ***BBS*** telepítési könyvtárában adjuk ki a `bin/mkpasswd rendszergazda_neve` parancsot; ezt követően meg kell adnunk a kívánt jelszót:

```
bin/mkpasswd Francois
bbs100 2.1 mkpasswd by Walter de Jong
<walter@heiho.net> (C) 2004
Enter password:
Enter it again (for verification):
OIGxutxGpuTowzw2AgMXZRkCNk
```

Az utolsó sor tartalmazza a rendszergazda titkosított jelszavát. Ez közölnünk kell a ***BBS***-sel, tehát nyissuk meg az *etc/su_passwd* fájlt, írjuk be a rendszergazda nevét, majd egy vesszővel elválasztva a fenti titkosított jelszót:

```
Francois:OIGxutxGpuTowzw2AgMXZRkCNk
```

A ***BBS*** elindítása a `/home/bbs100/bin/bbs start` paranccsal történik. Ha már fut a démon, `telnet`-tel, a megadott kapun keresztül csatlakozhatunk hozzá:

```
telnet kiszolgáló 12345
```

A rendszergazda (superuser, root, ha úgy tetszik) ***BBS***-es megfelelőjére, vagyis a ***SysOp***-ra (system operator, rendszeroperátor) a $ gyorsbillentyűvel lehet átváltani. A gyorsbillentyűt csak azok a felhasználók érhetik el, akiknek a neve szerepel a *etc/su_passwd* fájlban, egyébként egy csinos, a Föld különféle pontjain érvényes időt mutató naptár jelenik meg. Miután átváltottunk a ***SysOpra***, jó néhány további parancsot is el tudunk érni. A ***SysOP*** menübe a CTRL+S billentyűkombinációval léphetünk be. A ***SysOpnak*** joga van a rendszerbeállítások módosítására, üzenettovábbító és csevegőszobák létrehozására, valamint szükség esetén le tudja rendezni a kellemetlenkedő felhasználókat is.
Bár kell egy kis idő, mire megszokjuk őket, a ***BBS***-ek mégis sokoldalúak, könnyen megszerethetők. Nézzünk egy másik szempontot is: hat aktív felhasználóval nálam a memóriahasználat – a ***bbs100*** program memóriaigényét is figyelembe véve – 66917 bájt volt. Mint láthatjátok, ***mes amis***, a kis méretnek és az egyszerűségnek is megvan a maga előnye.
Miközben csodálattal adózunk az azonnali és a mobiltelefonos üzenetküldés népszerűsége előtt, egy pillanatra jusson eszünkbe, hogy milyen régre nyúlnak ezeknek a szolgáltatásoknak a gyökerei. Példaként egy réges-régi szolgáltatást szeretnék felidézni: annak idején sokat játszottunk a `write` és a `mesg` nevű paranccsal. A `mesg` segítségével lehet bekapcsolni az üzenetkezelő rendszert:

```
mesg y
```

Ezzel gyakorlatilag engedélyeztük másoknak, hogy üzeneteket küldjenek nekünk. Most jelentkezzünk be a másik terminálon is, és szintén engedélyezzük az üzenetküldést. Tegyük fel, hogy `marcel` névvel bejelentkeztem az egyik terminálon, míg ***François*** egy másikon dolgozik. Ha csevegni akar velem, a következő parancsot kell kiadnia:

```
write marcel /dev/pts/16
```

Ezután begépelheti, amit közölni akar velem. A kapcsolat lezárására a CTRL-D billentyűkombináció szolgál. Nálam a következő szöveg jelent meg:

```
[marcel@francois marcel]$
Message from francois@francois.salmar.com on
pts/14 at 19:30 ...
Helló, főnök!
Eldöntötted már, hogy ma este milyen bort
➥szolgáljunk fel?
```

Ahogy a mondás tartja: ***Plus ça change, plus c'est la même chose***.
Úgy tűnik, ***mes amis***, ismét ránk esteledett. Lassan fejezzétek be a beszélgetést. Persze csak kényelmesen, ebben a világban oly könnyű és jó hátradőlni és kapkodás nélkül kortyolgatni egy pohárka bort. Azt mondom tehát, igyunk egymás egészségére, ***mes amis***! A ***votre santé! Bon appétit!***

Linux Journal 2005. június, 134. szám

A cikkhez tartozó források elérhetősége:
➲ ***www.linuxjournal.com/article/8198***

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.

Betörések, betörési kísérletek II. avagy ki nevet a végén?

A sorozat előző tagjából megtudhattuk, pontosan milyen következményekkel jár az, ha valaki egy számítógépét vagy számítástechnikai rendszerét megpróbál feltörni. Most azokról a (jogi)eszközökről lesz szó, melyekkel egy ilyen csibészt (ckrackert) nyakon lehet csípni.

Mindenek előtt nem árt definiálni, hogy mit is ért a jog számítástechnikai rendszer alatt:

„Számítástechnikai rendszer minden olyan berendezés, amely közvetlen emberi beavatkozás nélkül (automatikusan) végez adatfeldolgozást, azaz adatok bevitelét, kezelését, tárolását, továbbítását látja el. A számítástechnikai rendszerek körébe tartoznak a számítástechnikai adatfeldolgozásra épülő, memóriával rendelkező olyan egységek is, amelyek megjelenésükben nem hagyományos számítógépet jelentenek.
A számítástechnikai rendszer fogalma azonban nemcsak az egyes berendezésekre terjed ki, hanem felöleli az azok összekapcsolása révén létrejött hálózatot, valamint az adattovábbítást, a kapcsolatfelvételt biztosító műszaki berendezéseket is." [1]

Ennek értelmében, ha mondjuk pár év múlva valaki betörve az intelligens hűtőszekrényembe rendel nekem 15 karton tejet, pontosan ugyanezt a bűncselekményt fogja elkövetni – feltéve persze, hogy addig nem változtatják meg a tényállás elemeket.

A betörés felfedezése

Függetlenül attól, hogy az elkövető a rendszerünkbe belépve milyen mértékű kárt okoz, azzal, hogy a rendszert feltörte, már megvalósította a számítástechnikai rendszer elleni bűncselekmény tényállásának valamennyi elemét. Feltétel persze, hogy a rendszerünknek legyen legalább valamilyen minimális szintű védelme (hardver- vagy szoftver-alapú tűzfal), hogy azt jogosulatlan belépéssel, kijátszással vagy más módon meg lehessen sérteni.

Alapszint

Rendszergazdaként belépve azt tapasztaljuk, hogy valaki az éj vagy a bitek leple alatt bejutott a rendszerünkbe, és tallózta a könyvtárainkat.[2] Annak nincs nyoma, hogy bármiféle változtatást eszközölt volna, vagy adatokat másolt volna le. Ilyenkor az eljárás, (azt követően, hogy a rendszerünket újra biztonságossá tettük például másik típusú tűzfal telepítése, trójai kereső program lefuttatása... stb.) hogy a korábban már begyűjtött naplóbejegyzéseket, illetve a visszakereshető ip címet lementjük (lehetőleg egy olyan gépre, ami nincs hálózatra kötve, hogy esetleg a visszatérő garázda ne leljen rá).
Az e módon kigyűjtött információkat – egy ismeretlen tettes elleni feljelentés keretében – eljuttathatjuk az illetékes ügyészhez vagy nyomozóhatósághoz.

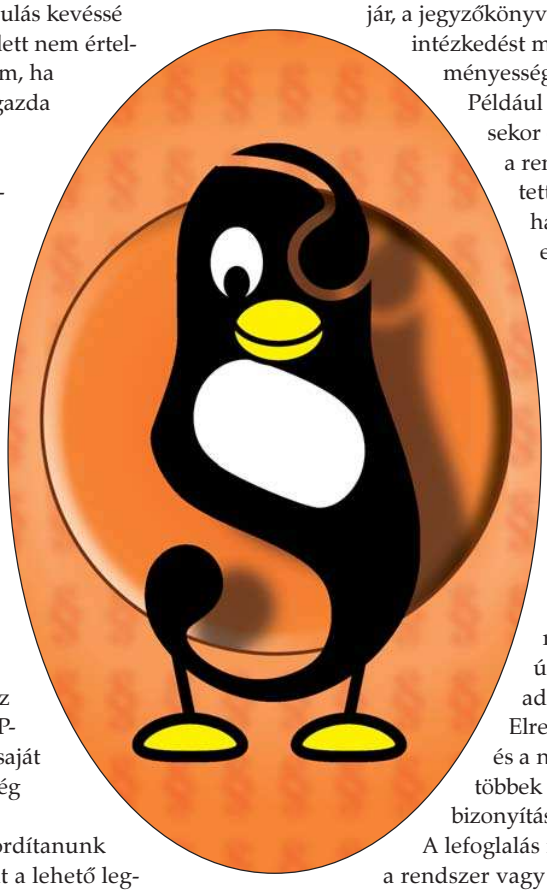
Középszint

A betörő jogosulatlanul belépett, jelenleg is bent tartózkodik, és éppen a ***szupertitkos*** feliratú mappánk tartalmát próbálja meg letölteni. A probléma leggyorsabb megoldásaként válasszuk le a gépet a netről. Ez esetben az elkövető észre fogja venni, hogy jelenlétére felfigyeltek, és talán soha többé nem próbálkozik meg a rendszerünkkel.
Ha a szupertitkos felirat azonban csak mindenféle kacatot rejtett, melyeket épp csalinak helyeztünk el ezen a gépen, akkor a helyzet persze merőben más. Amennyiben a rendelkezésünkre álló adatok alapján megtudtuk az elkövető ip címét, megkereshetjük a cím szolgáltatóját, és megkérdezhetjük, az előfizető nevét. Ezzel akár a későbbiekben eljáró nyomozó hatóság munkáját is segíthetjük, hiszen ebben az esetben már nem ismeretlen tettes ellen kell nyomozást folytatniuk.
Fontos tudni, hogy a párbajok ideje a XIX. századdal lejárt, nem megoldás az, ha a betörő kilétének felderítése érdekében mi is megpróbáljuk feltörni a minket ostromló rendszerét, hiszen akkor ugyanúgy jogosulatlan belépésre teszünk kísérletet vagy adott esetben, ha az akció célravezető, mi is elkövetővé válunk. Figyelemmel arra, hogy a Büntető

[1] Complex CDJogtár Btk. 300/F§ Kommentár részlet.
[2] Linuxos rendszerben ennek is marad nyoma, windows használata esetén pedig tételezzük fel, hogy így tett.

törvénykönyv nem nevesíti a „hirtelen felindulásból elkövetett számítástechnikai rendszer védelmét biztosító technikai rendszer kijátszását," mint tényállást – így e bűncselekmény elkövetésénél az erős felindulás kevéssé beszámítható. Jelen feltételek mellett nem értelmezhető jogos védelemként az sem, ha a büszkeségében sértett rendszergazda bosszút áll.

A nyomozóhatóságnak azonban – szemben a haragos rendszergazdával – lehetősége van rá, hogy külön engedély keretében titkos adatszerzést végezzen, ennek érdekében meghatározott számítástechnikai rendszer védelmét biztosító technikai intézkedéseket kijátsszon.



„Hajjaj" szint

Az elkövető belépve a gépünkre – azon kívül, hogy garázdálkodott az adataink között, – megnyitott pár portot, melyeken át a mi gépünk ip címét használva ostromol már hálózatokat.

Ez az eset mind közül a legsúlyosabb, itt mindenképpen javasolt az internetszolgáltatók (a kiderített IP-cím szolgáltatója csakúgy, mint a saját szolgáltatónk) valamint a rendőrség értesítése.

Itt különösen nagy gondot kell fordítanunk arra, hogy a betörés körülményeit a lehető legprecízebben dokumentáljuk (Adott esetben tanúk jelenlétében készített jegyzőkönyvekkel, melyeket a betöréssel kapcsolatosan tudtunkra jutott információhoz mellékelünk.)

Miért van mindez?

1. Értesíteni kell a betörő ip címét adó szolgáltatót a betörés körülményeiről is, hogy adott esetben saját hálózatán belül megpróbálhassa visszakeresni az elkövetőt.

2. Értesíteni kell a saját szolgáltatónkat, illetve azokat a szolgáltatókat akik felé a gépünkről kísérletet indítottak, hogy felhívhassák ügyfeleik figyelmét a veszélyre.

3. Értesíteni kell a rendőrséget, mert jó eséllyel előfordulhat, hogy a tőlünk indított kísérletet valaki más bejelenti az ügyészségen vagy a nyomozóhatóságnál, elkövetőként a mi ip címünket jelölve meg.

A feljelentés sorsa

A feljelentést elsődlegesen nyilvántartásba veszik. ***Megvizsgálják, hogy a*** történeti tényállás bűncselekmény gyanújának megállapítására alkalmas-e, ***a*** büntetőeljárás megindításának van-e akadálya, szükséges-e halaszthatatlan nyomozási cselekmény vagy más intézkedés foganatosítása, a bűncselekmény nyomozására van-e hatásköre, illetékessége.

A nyomozó szerv a Büntető eljárásról szóló törvény szerint köteles már a feljelentés vételekor, illetőleg a nyomozó szerv tagjának észlelésekor – ha a késedelem veszéllyel jár, a jegyzőkönyv felvétele előtt is – minden olyan intézkedést megtenni, amely a nyomozás eredményességét, gyors teljesítését elősegíti. Például amennyiben a betörés észlelésekor és bejelentésekor az elkövető a rendszerben tartózkodik, és tettenérhető, helye lehet azonnali házkutatásnak. Indokolt lehet ezen „gyorsított eljárás" azért is, mert ennek későbbi elvégzése az eljárás eredményességét károsan befolyásolhatja.

Lefoglalás

A lefoglalás jogászi nyelven: a bizonyítás érdekében a dolog birtokának elvonása a birtokos rendelkezése alól. Külön érdekes részlet, hogy a lefoglalás kapcsán a törvény már külön nevesíti a számítástechnikai rendszernek vagy ilyen rendszer útján rögzített adatokat tartalmazó adathordozónak a lefoglalását. Elrendelésére a bíróság, az ügyész és a nyomozóhatóság jogosult, többek között akkor, ha az adott tárgy bizonyítási eszköz.

A lefoglalás menete a következő: először is a rendszer vagy adathordozó birtokosát illetve az adat kezelőjét fel kell szólítani, hogy a keresett dolgot adja át, illetőleg a számítástechnikai rendszer útján rögzített adatot tegye hozzáférhetővé. (Tehát a nyomozóhatóság az internetszolgáltatót is kötelezheti valamennyi általa ismert információ kiadására). Amennyiben ezt önként nem teszi meg, rendbírsággal (általában 1000-200.000 forint közötti összeg) sújtható, kivéve a terheltet és azt, aki a tanúvallomást megtagadhatja, illetőleg aki tanúként nem hallgatható ki. (Tehát ha az elkövető takarítónőjét találják otthon a házkutatásra érkező rendőrök, aki makacs módon nem akarja átadni a gépét, rá kiszabhatnak rendbírságot, de ha maga az cracker van jelen, akkor ő, mint terheltet nem lehet megbírságolni). Fontos azonban tudni, hogy az átadás megtagadása egyik esetben sem lesz akadálya annak, hogy a keresett dolgot, illetőleg a számítástechnikai rendszer útján rögzített adatot házkutatással, illetve motozással megszerezzék.

A nyomozást általában annak megindulását követő 2 hónapon belül befejezik. Ez alatt az idő alatt vagy lehull a lepel vagy a tettes örökre ismeretlen marad.

Dr. Dudás Ágnes (dudas.agnes@abend.hu) ügyvédjelölt, az FSF egyik aktivistája. 2004-ben végzett az ELTE Jogtudományi Karán. Szakdolgozatát a szoftverek szerzői jogi védelméről írta, a 2003-as évet pedig e terület kutatásával a berlini Humboldt Egyetemen töltötte.