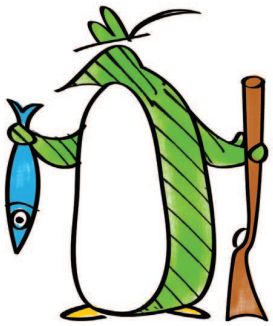
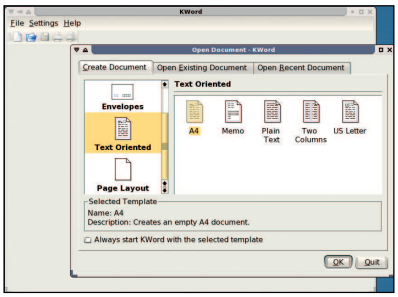
Programvadászat



KOffice

Linux alatt már megszokhattuk az alkalmazások bő választékát, legyen szó bármilyen feladatról. Nem kivétel ez alól az irodai munka területe sem.



Akitől megkérdezik, hogy milyen irodai programcsomagot használ, az nagy valószínűséggel az OpenOffice.org-ot fogja említeni. Ez egy igen népszerű, nagytudású program, amivel szinte mindent meg tudunk oldani a szövegszerkesztéstől, a táblázatkezelésen keresztül a bemutatók készítése. Hátránya viszont az igen nagy mérete, ami a mai gépeken nem jelenthet gondot egy régebbi konfigurációnak azonban már meggyűlhet vele a baja. Ugyanilyen feladatokra készített csomag a KDE-vel szervesen együttélő (azonban anélkül is használható) KOffice . Ebben is megtalálható minden alkótóelem, kisebb erőforrásigényének köszönhetően azonban kevesebb memóriával szerelt gépeken is futtatható. Bár ha KDE alatt szeretnénk használni akkor azt hiszem az igénye megegyezik az OpenOffice.org memóriaigényével.

A programok:
KWord – szövegszerkesztő
KSpread – táblázatkezelő
KPresenter – bemutatót készítő
Kivio – folyamatábra/grafikon készítő
Karbon 14 – vektoros rajzolóprogram
Krita – képfeldolgozó
Kugar – jelentéskészítő

KFormula - képletszerkesztő

A KOffice hátrányaként említhető az import/export szűrők hiányosságai. A http://www.koffice.org oldalon sok kiegészítő alkalmazást találunk, amelyekkel teljesebbé tudjuk tenni irodai környezetünket.

Abiword

Ez egy igazán remek szövegszerkesztő. Kis erőforrásigényének köszönhetően nagyon hatékonyan használható régebbi gépeken, ahol az OpenOffice.org már számításba sem jöhet. A korongra a Fedora Core1, Fedora Core2, RedHat 9 csomagok kerültek fel, telepítésük az `rpm-i csomagnév` paranccsal könnyedén elvégezhető.

Rendszermag

Természetesen felkerültek korongunkra a jelenlegi legfrissebb rendszermagok is. (2.4.27, és a 2.6.8.1). A 2.6-os rendszermagsorozat lassan elér abba az állapotba amikor mindenféle környezetben bátran használhatjuk. Hasonlóan a 2.2-ről a 2.4-re történő váltáshoz, sokan most is halogatják ezt a lépést. Ez főként az asztali rendszereknél jelenthet gondot az újabb hardverelemek támogatása miatt (ilyen lehet például a Serial ATA vezérlők meghajtója). Sok meghajtónak létezik úgynevezett backportja, ami a 2.6-os rendszermagból átkerült a 2.4-es sorozatba.

Mozilla, Firefox és Thinderbird

Ez a hármas méltán népszerű webes körökben. A Mozilla egy minden igényt kielégítő böngésző/levelező/üzenő/html-szerkesztő alkalmazás, a Firefox a legjobb, leggyorsabb és a leginkább testreszabható böngésző, a Thunderbird pedig ugyanez a levelző programok között.

Mindhárom program jelenleg elérhető legfrissebb változata került fel a mellékletre.

Magazin

ALSA

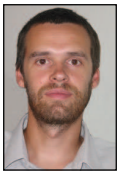
Az ALSA hangrendszer programozása cikkhez tartozó példakódok, dokumentáció, és a legfrissebb ALSA programok a ***Magazin/ALSA*** könyvtárban kaptak helyet.

Fogadó

Marcel által írt cikkben szereplő játékok telepíthető formában és forráskódban érhetőek el a ***Magazin/Fogado*** könyvtárban.

Moodle

Ezzel a hatékony e-learning rendszerrel mindenki könnyen elérhetővé tudja tenni az elektronikusan oktatható tananyagokat, lehetőség van a tudás ellenőrzésére is. A legnagyobb előnye, hogy a HTML ismerete nélkül is könnyedén lészíthetjük el a kívánt tanfolyamot.



Csontos Gyula
(Csontos.Gyula@linuxvilag.hu)
A Linuxvilág szakmai és CD-szerkesztője. Szabadidejében szívesen mászik hegyet és kerékpározik.

A teljesítmény bére

A *Sun* újra feltalálta a gépidő számlázását – új, elosztott számítógép-hálózati szolgáltatásáért ugyanis az igénybevétel időtartama alapján fizethetnek az ügyfelek. A szolgáltatás alapvetően egy órás csomagokban vásárolható meg, az alapár pedig 1 dollár/processzor/óra. A gépidő bérlése elsősorban az elosztott rendszereken hatékonyan futtatható feladatok megoldásánál előnyös. Ilyenek például a különböző szimulációk, modellezések és leképezések, de a rendszer szabványos elosztott hálózatokra írt programok tesztelésére is alkalmas. A Sun elképzelései szerint a számítási teljesítmény vásárlása hamarosan ugyanolyan természetes lesz, mint az egyéb közművek igénybevétele, sőt akár arra is lehetőségünk lesz, hogy világszerte számításiidő-nagykereskedőktől vásároljunk erőforrásokat.
➲ www.sun.com/tech-center

Beépített védelem

A *National Semiconductor* új biztonsági lapkákat mutatott be. A *SafeKeeper Trusted Input/Output* lapkákat asztali és hordozható számítógépek alaplapjaira fogják építeni, feladatuk pedig az, hogy egy úgynevezett *Trusted Platform Module (TPM)*, egy be/kiviteli alrendszer és egy beágyazott belső program egyesítésével megvalósítsák a *Trusted Computing Group* által előírt biztonsági szolgáltatásokat. (Ezek gyakorlatilag a vírusok és a betörések ellen védik a rendszert.) Az *IBM* már meg is kezdte a lapkák használatát. A *TPM*-ek lényegében olyan mikrokontrollerek, amelyek feladata a számítógépeken használt jelszavak, digitális tanúsítványok, titkosítási kulcsok biztonságos tárolása, továbbá a gépek BIOS-ának, operációs rendszerének és alkalmazásainak védelme a támadásoktól és a jogosulatlan hozzáférésektől. Hasonló célokra szoftveres megoldások is léteznek ugyan, ám ezeket – a cég képviselőinek állítása szerint – könnyebb támadni, mint a hardveres megvalósítást. A *National Semiconductor* lapkái nagy tételben mindössze 5 dollárba kerülnek, így komoly költségnövelő tényezőnek sem nevezhetjük őket.
➲ www.national.com/appinfo/advancedio

Videokamera merevlemezzel

A *JVC* bejelentette, hogy *Everio* sorozatú digitális videokameráinak egy részében a jövőben 1”-os, 4 GB kapa-

citású *Microdrive* merevlemezeket fog alkalmazni a felvételek tárolására. A most bejelentett, digitális fényképezőgépként is használható két kameramodell 2,12 MP felbontással dolgozik, és 10x zoommal rendelkezik. A gépek súlya akkumulátorral és merevlemezzel együtt 315 illetve 355 gramm.
A *JVC* kameráiba emellett egy rendkívül kis méretű, mégis nagy teljesítményű *MPEG-2* tömörítő lapka is kerül, amely a változtatható tömörítési aránynak köszönhetően egy órányi *DVD* minőségű, vagy 2-5 órányi gyengébb minőségű felvétel rögzítését teszi lehetővé. A merevlemez használata kapcsán érdekesség a cég által az üzembiztos használat elősegítésére kidolgozott megoldás: az apró merevlemezt zselészerű anyaggal veszik körül, amely működés közben ebben úszik, pontosabban lebeg. A meghajtó egyébként megfelel a *CompactFlash* szabványnak is, így hordozható gépbe, egyéb készülékbe gond nélkül áthelyezhető, ha a felvételeket például át szeretnénk másolni vagy szerkeszteni.
➲ www.jvc.com/presentations/everio

Java 1.5

A *Sun* elérhetővé tette a *Java 2 Platform Standard Edition* legújabb, 5.0-s változatát. A belső változatszámát tekintve 1.5-ös kiadás igazán forradalmi változásokat nem hozott, ellenben számos helyen finomították, fejlesztették képességeit. Új típusok bevezetésével próbálták megkönnyíteni a fejlesztők munkáját, javítottak a felügyeleti lehetőségeken és a teljesítményen, valamint felfrissítették a felhasználói felületet is. A *J2SE 5.0* teljes értékűen támogatja a nemzetközi használatot és az *OpenGL* alapú hardveres gyorsítást. Letöltése a *Sun* oldaláról természetesen továbbra is ingyenes.
➲ *java.sun.com*

LSB 2.0

A *Free Standards Group* közzétette a *Linux Standard Base (LSB)* második változatát. A linuxos megoldások együttműködését segítő szabvány újdonsága a kódok együttműködését segítő *C++ ABI* (Alkalmazási Bináris Felület), valamint az *IMP PowerPC 64* és az *Intel* illetve az *AMD* 64 bites processzorainak támogatása, továbbá számos kisebb alkotóeleme, alszabványa is megújult. Az egymással együttműködni képtelen fejlesztési ágak sokak véleménye szerint a legkiválóbb kezdeményezést is képesek megölni – kiváló példa erre a *UNIX* –, tehát nem véletlen, hogy a *Linuxot* ettől a sorstól óvni kívánó *LSB* mögött olyan nagy cégek is felsorakoztak, mint például az *IBM*, a *HP*, az *Intel* és az *AMD*, illetve gyakorlatilag az összes nagy terjesztés készítőjét is megtaláljuk a támogatók sorában.

➲ www.freestandards.org

Asztali nagyokos

A korábbi hasonló megoldásokhoz képes határozottan olcsó, mégis hatalmas tudású kommunikációs eszközt állított össze az *Amstrad*. Az *E3* jelölésű készülék talán a videotelefonokhoz áll a legközelebb, ám színes kijelzője és billentyűzete révén szöveges és multimédiás üzenete küldésére, webezésre, videókonferenciára és elektronikus levelek és faxok továbbítására, játékok futtatására és üzenetrögzítőként való használatra egyaránt alkalmas. A *páronként* mindössze 100 fontos, vagyis 37000 forintos áron megvásárolható készülékek csak *Nagy-Britanniában* kaphatók, ugyanis az általuk elérhető szolgáltatások helyi szolgáltató vállalathoz kötöttek.

A készülék kijelzője 480x320 képpont felbontású, belsejében kétmagos, 150 MHz órajelű, 32 MB *Flash* és 32 MB RAM memóriával gazdálkodó *Texas Instruments* processzort találunk. Az eszköz későbbi bővítését többek közt *USB* kapu teszi lehetővé, a szolgáltatóhoz alapesetben 56 kb/s sebességű modemmel kapcsolódik. Operációs rendszere a 2.4.18-as Linux rendszermagra épül, alapját a *MontaVista* összeállítása adja.

➲ www.amstrad.co.uk

Felfedezőút digitális kísérővel

A *Node* nevű cég *Explorer v2* megerősített, elsősorban kültéri használatra szánt, *GPS*-vevővel is ellátott kézigépet jelentett be. A gép 400 MHz órajelű *Intel XScale* processzorra épül, és a hírek szerint *Linux* és *Microsoft Pocket PC* operációs rendszerrel egyaránt használható.

A géphez a *Node* adatgyűjtő és újratöltő dokkoló állomásokat is kínál, a központi felületet és követést pedig az *Engine* nevű megoldás teszi lehetővé. Az *Explorer v2* 320x240 képpont felbontású érintőképernyővel, multimédiás képességekkel és vezeték nélküli hálózati csatolóval rendelkezik, 256 különböző színben rendelhető háza vízálló, akkumulátorról elvileg akár egy teljes napig is üzemeltethető.

A készülék nemcsak ipari területeken állhatja meg helyét, de arra is alkalmas lehet például, hogy egy szabadtéri múzeum látogatói önállóan, akár személyre szabott multimédiás bemutatókkal segítve ismerkedhessenek meg az őket érdeklő látnivalókkal. Eközben az üzemeltetők pontosan figyelemmel tudják kísérni a látogatók mozgását, megelőzhetik a tiltott területekre való belépést, felismerhetik, hogy melyek a jellemző mozgásirányok, majd az elemzések alapján javítani tudják szolgáltatásaikat.

➲ www.nodeexplore.com

A Windows felé vezető út Linuxszal van kikövezve?

Bár a számok azt sugallják, hogy – különösen Kelet-Európában, Latin-Amerikában és Ázsiában – dinamikusan nő a linuxos számítógépek aránya, a *Gartner* kutatásai szerint a valóság közel sem fest ennyire jól: a *Linux* operációs rendszerrel értékesített számítógépek 80 százalékán ugyanis a *Linuxot* előbb-utóbb *Windows* váltja fel - sok esetben korántsem jogtiszta forrásból. (Hozzátenném: a folyamat hazánkban a *Sulinet program* keretében eladott gépek esetében figyelhető meg a legjobban.) A *Linux* látszólagos terjedését az indokolja, hogy alkalmazásával az eladó megtakaríthatja az operációs rendszer árát – amelynek költsége különösen az alsó kategóriás gépeknél lehet komoly tényező. Ha *Linuxot* telepít, akkor kedvezőbb árat tud kínálni a vevőnek, mintha jogtiszta *Windows* rendszerrel kínálná portékáját. Érdekes kérdés, hogy a Távol-Keleten, Indiában és Oroszországban hamarosan megjelenő *Windows XP Starter Edition*, a *Windows XP* lebutított, olcsó változata, amelyet a *Microsoft* a *Linux* ellenfelének szán, mennyire fogja átalakítani a vásárlási és szoftverhasználati szokásokat. A *Microsoftnak* nyilván az az érdeke, hogy az új gépekre bármilyen változatban is, de jogtiszta *Windows* kerüljön. Az viszont kétséges, hogy a felhasználók egy teljes értékű *Linux* terjesztés ellenében hajlandóak lesznek-e pénzt adni egy lebutított rendszerért, vagy maradnak a jelenlegi, nem jogtiszta megoldásnál.

Marad a szemét

Az *IETF* úgy döntött, hogy feloszlatja a levélszemét témakörével foglalkozó munkacsoportját, mert annak munkája gyakorlatilag leállt, lehetetlenné vált a *Microsoft Sender ID* javaslatával kapcsolatosan kialakult viták miatt. A *Microsoft Caller ID* megoldásból és a *Sender Policy Frameworkből* összeálló *Sender ID* célja az lett volna, hogy a küldő fél hitelesítésével megakadályozza a kéretlen levelek továbbítását. Csakhogy a *Caller ID*-vel kapcsolatosan a *Microsoft* rendelkezik bizonyos szabadalmakkal, amelyek miatt a *MADRID (MTA Authorization Records In DNS)* munkacsoport működése – a részvevők jogi kérdésekben való egyetértésének hiánya miatt – gyakorlatilag ellehetetlenült. A munkacsoport feloszlatása – részben talán a túlzott remények miatt is – világszerte csalódottságot váltott ki. Ugyanakkor az eset nem különösebben váratlan, hiszen a *Microsoft* által szabott feltételek miatt korábban már többen is elutasították a *Sender ID* használatát. Azt még nem tudni, hogy az iparág szereplői képesek lesznek-e egyetlen általános megoldást kidolgozni a levélszemét problémájára, vagy többféle, egymással együttműködni képtelen rendszer fog létrejönni. Annyi bizonyos csak, hogy a szemét egyelőre marad a postaládákban.

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

Mi újság a rendszermag fejlesztése körül?

Wichert Akkerman jó ideje próbál *Debian* csomagkészítőt adni a magforrásokhoz, az *RPM*-készítő mellé. A 2003. októberi próbálkozása meglehetősen rosszul volt időzítve, hiszen a 2.6-os tesztsorozat közepére esett. *Wichert* abban az időben fejezett be némi munkát a folton, de a jelek szerint titokban tartotta 2004. júniusáig, amikor újra megpróbálta közzétenni. Most, hogy a a 2.6.6-os változat körül még mindig van némi mozgolódás, *Wichert* foltja sokkal jobb fogadtatásra lelt, és úgy néz ki, hogy felveszik azt a rendszermagba.

Hosszú ideig *Mel Gorman* tartotta fenn a 2.4-es és a 2.6-os *Virtual Memory Subsystem* (*Virtuális memória alrendszer*) hálózati dokumentációját. *Linus Torvalds* meglepő lépése után, amikor a korai 2.4-es kiadások után megpróbálta teljes egészében lecserélni az alrendszert, a dokumentáció hiányos lett, vagy el is tűnt. *Mel* erején felül dolgozott, hogy leírja a memóriakezelő belső működését, majd ingyenesen hozzáférhetővé tegye. Végül a *Prentince Hall* megszerezte a munkáját, és *Bruce Perence Open Source Series* (*Nyílt forráskódú sorozat*) részeként ki fogja adni papíron is. Az *Understanding the Linux Virtual Memory Manager* (*A Linux virtuálismemória-kezelő*) című könyv a *Linux* rendszermag egyik legösszetettebb elemének átfogó vizsgálata.

A rendszermag és a *C* fordító kapcsolata állandó vitákra ad okot. Amikor a rendszermag egy változata nem hajlandó lefordulni, gyakran nem teljesen világos, hogy a *C* kóddal van baj, vagy azzal, ahogy a fordító megpróbálja azt gépi kóddá alakítani. Mostanában *Linus Torvalds* és mások elkezdtek azon tanakodni, hogy a *Linux* jövőbeni változataiban, talán a 2.8-as vagy a 3.0-s sorozatban 3.3-masnál nagyobb *GCC* változatra lesz szükség. *Linus* a korábbi változatok rossz álnévkezelésére hivatkozott, de a fordítóprogramon dolgozóknak valószínűleg megvannak a saját elképzeléseik azzal kapcsolatban, hogy pontosan mi is a rossz a kódjukban és mi nem. Nem szokatlan, hogy a rendszermag alkalmazásfrissítést igényel, de minden alkalommal, amikor ilyen történik, az összeférhetőségben történt szakadás más alkalmazások frissítését is kikényszeríti. Az optimális nyilván az lenne, ha egyáltalán nem volna szükség alkalmazásfrissítésre.

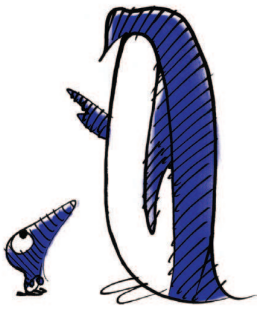
Greg Kroah-Hartman hivatalosan is vállalta a *sysfs* karbantartójának szerepét. *Christian Gmeiner* kényszerítette rá, hogy a nyilvánosság elé lépjen, amikor megpróbálta megtalálni a *sysfs* karbantartóját, hogy feleljen egynéhány a kódolással kapcsolatos kérdésre. Korábban senki nem szerepelt a *MAINTAINERS* (*Karbantartók*) fájlban, de *Greg* azt mondta, hogy valamikor közzétesz egy foltot, amely a saját elérhetőségeivel frissíti. A *sysfs* projekt egyébként egészen jól áll a 2.5-ös és 2.6-os időkeretben, ami azért is jó hír, mert mostanra határozottan kedveltté vált a felhasználók körében. Bár sokan még mindig ragaszkodnak a */proc* rendszerhez, a program finoman kiigazítja őket.

Az *NTFS* fájlrendszerek írásának támogatása nem éppen a számítástechnika Szent Grálja, de a felhasználók körében folyamatos igény van a támogatására. *Anton Altaparmakov*, nemrég jelentett be egy újabb fejlesztési lépést, nevezetesen a fájlrendszerben már meglévő fájlok felülírásának képességét. Ezzel együtt az *NTFS* meghajtó némi házimunkát is ellát, amit korábban kézzel kellett végeznünk, hogy elkerüljük a sérüléseket. Az *NTFS* még nincs kész igazán, s így inkább csak azok használják, akiknek a régi lemezeikről adatokat kell másolniuk, de úgy tűnik, eljön az idő, amikor teljes *NTFS* támogatottság lesz *Linux* alatt is.

A *PC9800* alrendszert kiveszik a 2.6-os fából. A hardver elavult, ami nem feltétlenül ok arra, hogy eltávolítsanak a rendszermagból egy szolgáltatást, de ha hozzátesszük azt is, hogy a *PC9800* kód karbantartása megszűnt, akkor rögtön világosabbak a döntés mögött meghúzód indokok. *Linus Torvalds* világosan kijelentette, hogy nem számít, ha vannak olyan felhasználók, akik méltatlankodnak egy szolgáltatás megszüntetése miatt. Ha egy kód hibás, és senki nincs, aki kijavítsa, akkor mennie kell. Természetesen a szabadalmazott *BitKeeper* kódfelügyeleti rendszernek köszönhetően viszonylag könnyű újra visszatenni egy kódot a rendszermagba, így ha előkerülnek a karbantartásra vállalkozó emberek, nincs akadálya a *PC9800* „reinkarnációjának". Az elárvult kódok eme könyörtelen kezelésének egyébként jótékony hatása is lehet, hiszen a „haláleset" felkeltheti olyan emberek figyelmét, akik esetleg elvállalnák a karbantartást.

A jó öreg *UMSDOS* szolgáltatás szintén leeshet a 2.6-os fáról. A régi, szép időkben az *UMSDOS* arra szolgált, hogy segítségével a *Linuxot* egy meglévő *MS-DOS* partícióra telepíthessük anélkül, hogy az ott működő operációs rendszer működésképtelenné válna. Így minden kockázat nélkül kísérletezhettek a kezdő felhasználók. Ma már nem nagyon használják, ezt a megoldást, a történelmi értéke azonban olyannyira jelentős, hogy *Linus Torvalds*, *Andrew Morton* és a többi rendszermag-karbantartó igazán bent hagyhatná az utókor kedvéért. Sajnos az *UMSDOS* a 2.6-os változatban nem működik, ami az előzőek ismeretében erősen csökkenti ennek az esélyét. Ha nem akad, aki meggyógyítja, ez az eszköz, amely olyan sok embernek segített megismerkedni a *Linux*-szal, csupán emlék marad.

Zack Brown (Linux Journal 126. szám)



A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tüköroldalait, 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

Békén hagynak a levélférgek?

A kérdésem: miért kellene ***Linuxot*** használnom? Jelenleg ***Microsoft Windows XP***-t használok. A gépet leginkább a világháló böngészésére, néha levelek küldésére, zeneletöltésre és néha a ***Microsoft Worddel*** való szövegszerkesztésre használom. Éppenséggel nem vagyok egy számítógépes zseni. Érdemes ***Linuxot*** használnom? Vagy tartsak ki a most használt rendszernél és ne is foglalkozzak vele többet? A vírusok, férgek és kémprogramok miatt nemrég fontolgatni kezdtem, hogy valami mást tanuljak meg. Köszönöm a segítséget.

Henry Stone,
➲ henrystone39@hotmail.com

A fenyegető biztonsági kérdésekkel kapcsolatban sok mindent megtudhatsz, ha ellátogatsz a ***mozilla.org*** oldalra a ***Firefox*** ingyenes böngészőprogramért vagy a ***Thunderbird*** levelezőprogramért, amelyek a mostani operációs rendszereden is használhatóak. A legkönnyebben akkor válthatsz ***Linuxra***, ha épp egy olyan projekten dolgozol, ami a ***Linux*** megtanulására ösztönöz, úgy tűnik, most nem vagy ilyen helyzetben.

Ha a ***Linuxra*** való átállást fontolgatod, légy óvatos a zeneletöltésekkel, kerüld a másolásvédett formátumokat. A zenefájlokhoz alkalmazható másolásvédő rendszerek két legnagyobb gyártója jelen van az operációs rendszerek piacán is és nem támogatja a ***Linuxot***. Tarts ki az ***MP3***, ***Ogg Vorbis*** és ***FLC*** formátumoknál, amelyek nem másolásvédettek, és így könnyen áthelyezhetőek lesznek az új gépre egy fejlesztés esetén.

Don Marti,
➲ dmarti@ssc.com

A kérdés kitűnő, olyasvalamire kérdez rá, amit mindenkinek meg kell fontolnia, mielőtt bármilyen operációs rendszert a gépére telepítene. Egy operációs rendszer csak annyira lehet jó, mint a rajta futtatható programok. Ha a jelenlegi platform kielégíti az igényeidet, semmi okod a váltásra. Gondolkozz el a következő kérdéseken:

- Szükséged van a hordozhatóságra?
  A ***Linux*** számos különféle felépítésű számítógépen futtatható, így elméletileg egy Apple hordozható gépen is használhatjuk ugyanazt a környezetet.

- Mennyire fontos számodra a biztonság?
  A ***Linux*** nem igényel gyakori foltozást és a legtöbb folt a rendszer újraindítása nélkül telepíthető. A ***Linux*** rendszerek ritkán válnak az önműködő hálózati támadások célpontjává. Ez nem azt jelenti, hogy biztonságosak, azt viszont igen, hogy olcsóbb lehet a rendszer és kevesebb időt igényel a rendszer biztonságának fenntartása.

- Milyen tapasztalatokkal rendelkezel?
  Ha a ***Linux*** még nem elég kényelmes számodra, hajlandó vagy erre egy kis energiát fordítani? A jól kidolgozott rendszercsomagok és kézre álló felhasználói felületek sokkal könnyebbé tették a ***Linux*** kezelésének elsajátítását, de ha sem időd, sem kedved nincs a tanuláshoz, nem éri meg az erőfeszítést.

- Elvárás számodra a licencelés rugalmassága?
  A ***Linux*** megvásárolható kereskedelmi terjesztőtől, de ingyen is letölthető. Ha úgy döntesz, hogy a jelenlegi terjesztőd nem tetszik, más forrást választhatsz, és biztos lehetsz benne, hogy a korábban használt programjaid ugyanúgy működni fognak.

Majdnem minden operációs rendszer ki tudja elégíteni az általad felsorolt kívánalmakat. Ha a jelenleg használt környezet megfelel az elvárásaidnak és nincsenek további biztonsági vagy rugalmassági igényeid, nincs ok a lecserélésére. Viszont ha meg szeretnél ismerni egy biztonságosabb vagy sokoldalúbb operációs rendszert, a ***Linux*** kitűnő választást jelent a kipróbálásra.

Minden új ***Linux***-felhasználónak azt javaslom, hogy vessen egy pillantást a ***Knoppix*** rendszercsomagra (a ***www.knoppix.net*** címen érhető el). A csomag nem igényel telepítést, egyszerűen a CD-ről futtatható. Emiatt kicsit lassabban fut, mint egy merevlemezre telepített rendszer, de érintetlenül hagyja az eredeti operációs rendszert. Így alkalmunk van a ***Linux***, ***OpenOffice.org***, ***Mozilla*** és a többi alkalmazás kipróbálására, mielőtt az átállás mellett döntenénk.

Chad Robinson, ➲ chad@lucubration.com

Szabványos blade-kiszolgálók

Létezik valamilyen szabvány a blade-kiszolgálók felépítésének formátumára vonatkozóan? Jelenleg többféle toronnyal és kerettel rendelkezem, amelyek elsődlegesen az ***ATX*** formátumot használják. Helytakarékosság céljából fontolóra venném egy blade-kiszolgáló alkalmazását, de nem szeretnék emiatt egy gyártóhoz kötődni, különös tekintettel a gyártók közt manapság tapasztalható gyors termékváltásokra. Összeilleszthetők a különböző gyártóktól származó blade-keretek? Úgy tűnik, erre a kérdésre a gyártók sem igazán tudják a megfelelő választ.

Michael Barnes, ➲ mbarnes@hcjb.org

A kérdésed a blade-kiszolgálókkal kapcsolatban rávilágít az iparág mai helyzetére és a felhasználók egyik fő kifogására. A blade-kiszolgálók viszonylag újak, és eddig még nem született szabvány a csatlakozó tartópanelekre vagy magára a blade-formátumra. Minden gyártónak megvan a saját, zárt formátuma. A következő években ez változhat, de gyors eredményekre nem lehet számítani.
A blade-kiszolgálók gyártói a csatlakozó tartópanelek és szerelőkeretek költségeit elenyészőnek tekintik a kiszolgáló-berendezések egészét tekintve, ezért nem is nagyon törekednek egy közös formátum kialakítására. Jellemzően minden csatlakozó tartóelem 3-8 egység magas, amelybe 8-20 lap bárhova beilleszthető.

A különböző gyártóktól származó szerelőkeretek együtt tudnak működni egy fiókban, így ha gyártók akarnál váltani, általában telepíthetnél új kereteket. Valójában a blade-kiszolgálók gyártói nem támogatnak sokáig egy bizonyos tartópanel-formátumot, így ha egy gyártót választunk is ki, előfordulhat, hogy különböző szerelőkereteink lesznek ugyanabban a fiókban.
Chad Robinson,
➲ chad@lucubration.com

Elakadt Fedora telepítés

Azért írok, mert gondjaim vannak a *Fedora 2* telepítésével. A rendszer újraindulása miatt nem jutottam túl messzire. 2004 május 28-án a letöltöttem a CD-ROM ISO képfájlokat Red Hat tükörkiszolgálójáról a

download.fedora.redhat.com/pub/fedora/linux/core/2/i386/iso

címről és lemezre is írtam azokat. A képfájlok május 13-i időpecséttel vannak ellátva, ha jól emlékszem.

A lemezek jónak tűnnek. A CD1-ről indítva a rendszert rendesen megjelenik a *Fedora 2* üdvözlőképernyője és emblémája, az *<F1>*-től *<F5>*-ig terjedő választási lehetőségek, az utasítás, hogy üssek Entert a grafikus telepítéshez, vagy gépeljem be a linux text szöveget a szöveges telepítés elindításához. Ennél a pontnál ütköztem a problémába. A következőkkel próbálkoztam sikertelenül:

- Egyszerű ENTER-leütés a grafikus telepítéshez.
- `linux text`
- `linux noprobe`
- `linux askmethod`
- `linux skipddc`
- `linux nofb`
- `linux resolution 1024x768`
- `linux mediacheck`

Bármelyiket is választottam, a rendszer újraindult, miközben a következőket láttam a képernyőn:

```
loading vmlinuz
loading initrd.img
```

Az egyetlen sikeres próbálkozásom a *memtest86* lehetőség kiválasztása volt, ami elindította a memória ellenőrzését és órákig futott bármilyen esemény nélkül.

A *VMware*-t is használom, ezért úgy gondoltam, adok egy esélyt a *Fedora 2*-nek a *VMware* virtuális gépként való futtatására. Ez működött is, de nem így szeretném használni, telepíteni szeretném egy külön merevlemezre. A rendszerem és a *VMware* közti látható különbség az, hogy a *VMware BIOS* csak egy *IDE* lemezt lát, egy virtuális lemezt, amire szerintem a Linux /dev/sda egységként hivatkozik. A számítógépem felépítése a következő:

- BIOS: *American Megatrends v2.51*
- Alaplap: *Asus P4P800 Deluxe* alaplap, *Intel P865PE* lapkakészlet, *Socket 478*
- Processzor: *Intel Pentium 4 2.4GHz*
- Memória: 256MB PC3200 DDR RAM (egyetlen egység, így 400MHz-en fut)
- Merevlemez: Primary Master – *EIDE* 20GB, Secondary Master – *Samsung DVD/CDRW* combo meghajtó és Third Master – *SATA* 80GB merevlemez
- CD-ROM meghajtó: *Samsung DVD/CD-RW* combo meghajtó
- Videokártya: *ATI RADEON* 9200se 128MB AGP 8x

Annak ellenére, hogy a *BIOS* támogatja az órajel növelést, ki van kapcsolva ez a lehetőség. Azt gondolván, hogy valamilyen *BIOS*-beállítás lehet a háttérben, visszatöltöttem az alapértelmezett beállításokat, de semmilyen változást nem tapasztaltam.

Amikor a fenti módon próbáltam telepíteni a *Fedora 2*-t, a 20 GB-os merevlemezt a primary master *IDE*-csatolóra kötöttem. Mivel ez a próbálkozásom nem járt sikerrel, megpróbálkoztam a 80 GB-os *SATA* lemezzel is a third master *IDE*-csatolón. Mindkét eset-

ben fizikailag leválasztottam a másik merevlemezt a számítógépről és a *BIOS*-ból is eltávolítottam a bejegyzést, vagyis ütközésről szó sem lehet.

Minden próbálkozásom kudarcot vallott. Arra a következtetésre jutottam, hogy vagy *BIOS*-probléma lehet az ok, vagy az alaplapom nem megfelelő. Tovább már nem tudok nyomozni az ügyben, mivel elértem a tárgyra vonatkozó számítástechnikai ismereteim határát.

A *BIOS*-frissítéssel várnék inkább addig, amíg biztos nem lesz, hogy ez a hiba oka. Korábban már többször frissítettem *BIOS*-t, de a szívem minden alkalommal ki akart ugrani a helyéből. Az alaplap leírása szerint a BIOS frissítését nem lehet elhibázni, de én még mindig vonakodom.

A 20 GB-os merevlemez egy idősebb *Cobalt Cube* gépből származik, amin egy darabig *netBSD* működött biztonsági háttérkiszolgálóként. Miután a merevlemezt áttettem a fent leírt gépbe, több másféle Linuxot, így *Debian Woody*-t és a *LindowsOS 4.5* változatát is feltelepítettem rá és semmilyen hibát nem tapasztaltam.

Végül annyit, hogy nem próbáltam ki a CD1 lemezen lévő *README-en.html* fájl által leírt pen-boot lehetőséget, mivel nem rendelkezem USB pen drive eszközzel. A gépemet alkatrészenként állítottam össze és hajlékonylemezes meghajtóval sem rendelkezem, úgyhogy a hajlékonylemezről történő rendszerindítás sem jöhet szóba.

Megkérdeztem a *Fedora* támogatói fórumokat is, akik rögtön azt ajánlották, hogy ellenőrizzem a lemezeimet. Mivel a rendszer rögtön az *initrd.img* betöltése után újraindul, a lemezeket nem tudtam a *Linuxszal* ellenőrizni. Ezután jött az ötlet, hogy a *VMware* virtuális gépen ellenőrizzem, ami megtörtént és semmilyen hibát nem találtam.

Kezdek arra a következtetésre jutni, hogy valami programhibáról van szó, vagy a gépem *BIOS*-ában vagy a *Fedora 2* magjának *initrd.img* fájljában. Nem tudom, mi lehetne a következő lépés, nem szeretném a fejlesztőket zaklatni, akik általában amúgy is valamelyik támogatói fórumra hivatkoznak. Teljesen tanácstalan vagyok, segítsen valaki!

Johnny,
➲ johnny@skydiveflorida.net

Nagyszerű munkát végeztél a hiba felderítésével kapcsolatban. Ahogy korábban te is említetted, nekem is az a véleményem, hogy valamilyen inkompatibilitásnak kell lennie az *FC2* és a géped között. Ugyanez a gondom adódott nekem is, amikor egy *Epia 800*-as gépen szerettem volna elindítani az *FC2*-t.

Esetleg érdemes lehet egy olyan *FC2* indítólemezt keresni, ami a gépednek jobban megfelel. Nekem sikerült ilyet találnom a *fedora.redhat.com* címen lévő *Bugzilla* felületen végrehajtott keresés során.

Christopher Wingert,
➲ cwingert@qualcomm.com

Próbáld meg a `linux noacpi`, `linux disableapic` és `linux noacpi disableapic` indítási lehetőségeket. Néhány alaplap és *BIOS* kombináció összeütközésbe kerül a *Linux APIC* és *ACPI* szolgáltatásaival, ezek tiltása mindig kipróbálásra érdemes lehetőség.

Chad Robinson,
➲ chad@lucubration.com

Használok egy *Linux* proxy-kiszolgálót. Kétféle módszerrel tudok a világhálóhoz kapcsolódni, az egyik vezeték nélküli módszerrel, a másik pedig egy DSL-modemen keresztül történik.

Milyen beállítással vegyem rá a proxy-kiszolgálót arra, hogy mind a két módszert használja a kapcsolat létrehozásakor, vagyis ha az egyik éppen nem működik, akkor is elérjem a világhálót anélkül, hogy a beállításokat megváltoztatnám? Jelenleg két proxy-kiszolgálót használok, ha az egyik kapcsolat nem működik, a másikat indítom el.

John Van De Veer,
➲ jvdvjav@hotmail.com

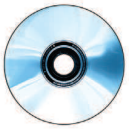
Használd egyszerűen az alábbi parancsot:

```
/sbin/ip route add default equalize
nexthop
➥via A.B.C.D dev eth0 weight 1
➥nexthop via E.F.G.H dev eth1 weight 1
```

Az `A.B.C.D` helyére írd be a kapcsolatod IP-címét, az eth0 és eth1 értékeket pedig a ténylegesen használt eszközökkel helyettesítsd. Ezzel tájékoztatjuk a *Linuxot*, hogy ugyanolyan súllyal használhatja a két útvonalat és így egyensúlyban tartható a forgalom.

Mario Bittencourt,
➲ mneto@argo.com.br

Linux Journal 2004. október, 126. szám



Bevezetés az ALSA alapú hangprogramozásba

A 2.6-os rendszermag hangkezelő alrendszerének minden szolgáltatását egyetlen teljes értékű API-n keresztül vehetjük igénybe.

Az *ALSA* az *Advanced Linux Sound Architecture* (fejlett linuxos hangkezelő alrendszer) rövidítése. Rendszermag-illesztőprogramokból, egy alkalmazásprogramozási felület (application programming interface, *API*) könyvtárból és *Linux* alatti hangkezelést támogató segédprogramokból áll. Írásomban szeretném röviden áttekinteni az *ALSA Projectet* és szoftveres összetevőit. Elsősorban az *ALSA PCM* felületének programozásával fogok foglalkozni, a próbálgatást, az ismerkedést pedig kódrészletek mellékelésével szeretném elősegíteni.
Az *ALSA* megismerése nemcsak azért lehet érdekes, mert újdonság, hanem mert nem ez az egyetlen hangkezelő *API*. Az *ALSA* kiváló választás, ha alacsony szintű hangkezelő eljárásokat szeretnénk megvalósítani, vagy különleges, a többi hangkezelő *API* által nem támogatott szolgáltatásokat szeretnénk használni. Ha meglévő hangkezelő alkalmazással rendelkezünk, dönthetünk úgy, hogy az *ALSA* illesztőprogramok natív támogatásával bővítjük. Ha a hangkezelés kevésbé érdekel bennünket, és például csak hangfájlokat szeretnénk lejátszani, akkor a magas szintű hangkezelő eszközkészletek valamelyike, mint az *SDL*, az *OpenAL* vagy a saját terjesztésünkben szereplő egyéb megoldás számunkra jobb választás lehet. Az *ALSA* használatakor gyakorlatilag az *ALSA* támogatására képes rendszermaggal ellátott rendszerekre korlátozzuk önmagunkat.

Az ALSA története

Az *ALSA Project* azért indult el, mert a *Linux* rendszermag hangkezelő illesztőprogramjai (az *OSS/Free* illesztőprogramok) aktív gondozás nélkül maradtak, így az újabb hangkezelő megoldásokat sem voltak képesek támogatni. A tervezetet *Jaroslav Kysela* indította, aki korábban egy hangkártya-illesztőprogramot készített. Idővel egyre több fejlesztő csatlakozott hozzá, egyre több hangkártya támogatása valósult meg, az *API* felépítése pedig fokról fokra finomodott.
A 2.5-os sorozat fejlesztése során az *ALSA* a *Linux* rendszermag hivatalos része lett. A 2.6-os rendszermag megjelenésével az *ALSA* az üzembiztos ág részévé vált, és várhatóan széles körben el fog terjedni.

A digitális hangkezelés alapjai

A hangok lényegében változó légnyomáshullámok, ezeket valamilyen átalakítóval, például mikrofonnal formáljuk elektromos jelekké. Az analóg-digitális átalakítók (*analog-to-digital converter*, *ADC*) az analóg feszültségeket diszkrét értékeknek feleltetik meg, ezeket mintáknak nevezzük. A műveletet rendszeres időközönként végzik el, amit mintavételi időnek hívunk. Ha a mintákat egy digitális-analóg átalakítóra küldjük, majd kimeneti átalakítóra, például egy hangszóróra továbbítjuk, akkor helyre tudjuk állítani az eredeti hangot.
A minták méretét bitekben szokták megadni, ez az egyik olyan tényező, amely meghatározza, hogy a digitális formátummal milyen pontosan tudjuk ábrázolni a hangot.
A hangminőség másik fontos befolyásoló eleme a mintavételi gyakoriság. A *Nyquist*-tétel kimondja, hogy a legmagasabb visszaadható frekvencia egyenlő a mintavételi frekvencia felével.

ALSA alapismeretek

Az *ALSA* különféle hangkártyákhoz készült illesztőprogramok gyűjteménye, valamint tartalmaz egy *API* könyvtárat is, ez a *libasound*. Az alkalmazásfejlesztők számára az *API* használata javasolt, és nem a rendszermag felületé.
A könyvtár magasabb szintű és a fejlesztők számára barátságosabb felületet biztosít, továbbá lehetővé teszi az eszközök logikai elnevezését, így a fejlesztőknek nem kell elveszniük az alacsony szintű részletekben, mint például az eszközfájlok kezelése.
Az *OSS/Free* illesztőprogramok programozása ezzel szemben rendszermag szintű hívásokkal történik, amihez a fejlesztőknek eszközfájlneveket kell megadniuk, illetve számos műveletet *ioctl* hívásokkal kell elvégezniük.
A visszirányú együttműködés lehetősége érdekében az *ALSA* olyan rendszermagmodulokat is tartalmaz, amelyek emulálják az *OSS/Free* hangkezelő illesztőprogramokat, így a legtöbb hangkezelő alkalmazás módosítások nélkül is tovább használható. Az *OSS/Free API*-t rendszermagmodulok nélkül a *libaoss* burkolókönyvtárral lehet emulálni.
Az *ALSA* a beépülő modulokat is támogatja, ezekkel új eszközök támogatása is megvalósítható, akár teljes egészében szoftveresen megvalósított képzetes eszközök létrehozására is mód nyílik. Az *ALSA* számos parancssori segédprogramot tartalmaz, egyaránt található köztük keverő, lejátszóprogram, valamint az egyes hangkártyák különleges képességeinek vezérlésére szolgáló kiegészítők.

Az ALSA felépítése

Az *ALSA API* az általa támogatott fontosabb felületek alapján osztható fel:

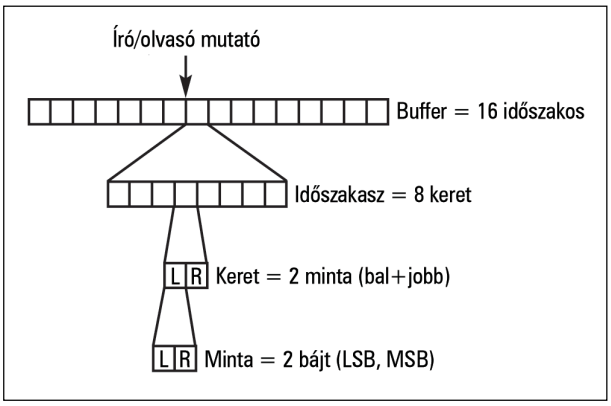
- *Vezérlőfelület*: Általános célú felület a hangkártyák regisztereinek kezelésére és az elérhető eszközök lekérdezésére.
- *PCM felület*: A digitális hangfelvételek és lejátszások kezelésére szolgáló felület. A továbbiakban erről a felületről lesz szó, ugyanis a digitális hangkezelő alkalmazásokban ez jut a legfontosabb szerephez.
- *Nyers MIDI felület*: Az elektronikus hangszerek terén elterjedt szabvány, a *MIDI* (*Musical Instrument Digital Interface*, digitális hangszer felület) támogatását valósítja meg. Ez az *API* a hangkártyán található *MIDI* busz elérését teszi lehetővé. A nyers interfész a *MIDI* eseményekkel közvetlenül dolgozik, a protokoll és az időzítések kezeléséért a programozó felelős.
- *Időzítő felület*: A hangesemények szinkronizálása céljából biztosítja a hangkártyákon található időzítő eszközök elérését.
- *Sorrendvezérlő felület*: A nyers *MIDI* felületnél magasabb szintű *MIDI* programozó és hangkeltő felület. A *MIDI* protokoll és az időzítések túlnyomó részét képes kezelni.
- *Keverő interfész*: A hangkártyákon lévő, a jeleket irányító és a hangerőszinteket szabályozó eszközöket vezérli. A vezérlő felületre épül.

Eszköznevek

A könyvtári *API* eszközfájlok helyett logikai eszköznevekkel dolgozik. Az eszköznevek valós, fizikai eszközök és beépülő modulok egyaránt lehetnek. A hardvereszközök a `hw:i,j` formátumot használják, ahol az `i` a kártya számát, a `j` pedig az adott kártyán lévő eszközt adja meg. Az első hangeszköz a `hw:0,0`. Az írásomhoz tartozó példákban szereplő `default` (alapértelmezett) álnév az első hangkezelő eszközre hivatkozik. A beépülő modulok egyéb egyedi neveket használnak, a `plughw:` például olyan beépülő modul, amely hardvereszközhöz biztosít hozzáférést, de egyéb, a hardvereszköz által alapesetben nem támogatott szolgáltatásokat is biztosít, például a mintavételi gyakoriság megváltoztatását. A `dmix` és a `dshare` beépülő modul segítségével több adatfolyamot tudunk egyesíteni, illetve egyetlen folyamot dinamikusan tudunk megosztani különböző alkalmazások közt.

Hangpufferek és adatátvitel

A hangkártyák rendelkeznek egy pufferrel, amely tárolja a rögzített mintákat. Ha a pufferben elegendő minta gyűlt fel, megszakítást hoz létre. A rendszermag hangkezelő illesztőprogramja közvetlen memória-hozzáférést (*direct memory access*, *DMA*) használ a minták továbbítására a megfelelő alkalmazás memóriabeli pufferébe. A lejátszás hasonlóan folyik, *DMA* használatával az alkalmazás memóriabeli pufferének tartalma a hangkártya hardveres pufferébe továbbítódik.
A hardverpufferek körkörös elven működnek, vagyis végük elérésekor az újabb adatok beírása az elejükre történik. A hardver- és az alkalmazáspufferben egyaránt mutató jelzi az aktuális írási-olvasási helyzetet. A rendszermagon kívüli összetevők közül csak az alkalmazáspuffer érdemes figyelemre, ezért a továbbiakban csak ezzel foglalkozunk.



1. ábra Az alkalmazáspuffer

A puffer méretét *ALSA* könyvtárhívásokkal lehet beállítani. A puffer akár egészen nagy méretű is lehet, ám ekkor tartalmának egyetlen műveletben való továbbítása elfogadhatatlan mértékű késleltetéshez, lappangási időhöz vezet. Ennek elkerülése érdekében az *ALSA* a puffert időszak sorozatokra osztja (ezeket *OSS/Free* alatt töredékeknek hívják), és az adatokat időszak egységekben továbbítja.
Egy időszak kereteket tartalmaz, ezek mindegyike adott időpontban rögzített mintákat foglal magába. Egy sztereó eszköznél egy-egy keretnek két csatornához kell mintákat tartalmaznia. Az *1. ábrán* – elméleti értékeket használva – a pufferek időszakokra, keretekre és mintákra osztását szemléltettem. Ebben az esetben a bal és a jobb oldali csatorna adatai felváltva szerepelnek a keretekben; ezt átlapolt módnak nevezzük. A nem átlapolt mód támogatása – ennél az egyik csatorna adatai a másik csatorna adatai után szerepelnek – szintén megoldott.

Alul- és túlcsordulás

Ha egy hangeszköz aktív, akkor folyamatos adattovábbítás folyik a hardver- és az alkalmazáspuffer között. Adatrögzítéskor (felvételkor), ha az alkalmazás nem olvassa ki elég gyorsan a puffert, akkor annak tartalma a körkörös működés miatt felülírásra kerül az új adatokkal. Az ilyen típusú adatvesztést túlcsordulásnak nevezzük. Ha lejátszáskor az alkalmazás képtelen kellő gyorsasággal biztosítani a szükséges adatokat, a puffer kiürül, ami alulcsordulási hibát eredményez. Az *ALSA* leírásában előfordul, hogy mindkét hibára az `XRUN` kifejezéssel hivatkoznak. A helyesen tervezett alkalmazások az `XRUN` hibák számát a lehető legkisebbre szorítják le, és fellépésük után is képesek folytatni működésüket.

Egy átlagos hangkezelő alkalmazás

A *PCM* felületet használó programok általában az alábbi lépéseket követik:

- a felület megnyitása rögzítés vagy lejátszás céljából
- a hardver beállításainak megadása
- (hozzáférési mód, adatformátum, csatornák, gyakoriság stb.)

Az adatok feldolgozása közben:

- PCM adatok olvasása (felvétel)
- vagy PCM adatok írása (lejátszás)
- interfész lezárása

1. kódrészlet Néhány PCM típus és formátum megjelenítése

```
#include <alsa/asoundlib.h>

int main() {
  int val;

  printf("Az ALSA könyvtár változatszáma: %s\n",
         SND_LIB_VERSION_STR);

  printf("\nPCM adatfolyam típusok:\n");
  for (val = 0; val <= SND_PCM_STREAM_LAST; val++)
    printf("  %s\n",
      snd_pcm_stream_name((snd_pcm_stream_t)val));

  printf("\nPCM hozzáférési típusok:\n");
  for (val = 0; val <= SND_PCM_ACCESS_LAST; val++)
    printf("  %s\n",
      snd_pcm_access_name((snd_pcm_access_t)val));

  printf("\nPCM formátumok:\n");
  for (val = 0; val <= SND_PCM_FORMAT_LAST; val++)
    if (snd_pcm_format_name((snd_pcm_format_t)val)
      != NULL)
      printf("  %s (%s)\n",
        snd_pcm_format_name((snd_pcm_format_t)val),
        snd_pcm_format_description(
                          (snd_pcm_format_t)val));

  printf("\nPCM alformátumok:\n");
  for (val = 0; val <= SND_PCM_SUBFORMAT_LAST;
       val++)
    printf("  %s (%s)\n",
      snd_pcm_subformat_name((
        snd_pcm_subformat_t)val),
      snd_pcm_subformat_description((
        snd_pcm_subformat_t)val));

  printf("\nPCM állapotok:\n");
  for (val = 0; val <= SND_PCM_STATE_LAST; val++)
    printf("  %s\n",
      ➥snd_pcm_state_name((snd_pcm_state_t)val));

  return 0;
}
```

Az alábbiakban ténylegesen működő kódrészleteket is látni fogunk. Mindenkinek javasolom, hogy fordítsa és futtassa le ezeket saját ***Linux*** rendszerén, vizsgálja meg a kimenetet, és próbálkozzon meg a javasolt módosításokkal. A cikkhez tartozó példaprogramok teljes változata a 65. CD Magazin/ALSA könyvtárban megtalálható.

Az *1. kódrészlet* az ***ALSA*** által használt ***PCM*** adattípusok és beállítási értékek egy részét jeleníti meg. Az első követelmény a fejrészfájl beillesztése, ez tartalmazza az ***ALSA*** könyvtári függvényeinek megadásait. A megjelenő megadások egyike az ***ALSA*** változatszáma.

A program fennmaradó része különféle ***PCM*** adattípusokon lépeget végig, kezdve az adatfolyam típusokkal. Az ***ALSA*** az utoljára számba vett értékhez szimbolikus neveket is szolgáltat, illetve egy segédfüggvénnyel egy karakterláncban rövid leírást is képes mellékelni az egyes értékekhez. Amint a kimenetből is látható, az ***ALSA*** rengeteg, a saját rendszeremen például 38 különféle formátum támogatására képes.

A programot fordításkor előbb össze kell kapcsolni az ***ALSA libasound*** könyvtárával, ezt az esetek többségében a `-lasound` kapcsolóval tehetjük meg. Az ***ALSA*** könyvtári függvények egy része használja a `dlopen` függvényt, illetve lebegőpontos műveleteket végez, ezért a `-ldl` és a `-lm` kapcsolóra úgyszintén szükség lesz.

A *2. kódrészlet* az alapértelmezett ***PCM*** eszközt nyitja meg, megad néhány beállítást, majd a megjeleníti a hardveres beállítások túlnyomó részének értékét. Hangrögzítést vagy -lejátszást nem végez. Az `snd_pcm_open` hívással megtörténik az alapértelmezett ***PCM*** eszköz megnyitása, illetve a hozzáférési mód beállítása `PLAYBACK`-re, vagyis lejátszásra. A függvény első átadott értékében egy kezelőt (***handle***) ad vissza, a későbbi hívásokban ezzel történik a ***PCM*** adatfolyam manipulálása. A legtöbb ***ALSA*** könyvtárhíváshoz hasonlóan a függvény egy egész állapotértéket ad vissza, ha ez negatív, az hibára utal. Ebben az esetben ellenőrizzük a visszatérési értéket, az `snd_strerror` függvénnyel megjelenítjük a hibaüzenetet, végül kilépünk. A könnyebb áttekinthetőség érdekében a hibaellenőrzések túlnyomó részét kihagytam a példaprogramokból. Valódi alkalmazásban természetesen minden ***API*** hívás visszatérési értékét ellenőrizni kell, és gondoskodni kell a megfelelő hibakezelésről.

Ahhoz, hogy meg tudjuk adni az adatfolyam hardveres beállításait, kell foglalnunk egy `snd_pcm_hw_params_t` típusú változót; ezt az `snd_pcm_hw_params_alloca` makróval tehetjük meg. Következő lépésként az `snd_pcm_hw_params_any` függvénnyel elvégezzük a változó kezdeti értékadását, ekkor az imént megnyitott ***PCM*** adatfolyamot is átadjuk.

Most ***API*** hívásokkal megadjuk a kívánt hardverbeállításokat, ehhez felhasználjuk a ***PCM*** folyamkezelőt és a hardverbeállításokat tartalmazó adatszerkezetet. Az adatfolyamot átlapolt módra állítjuk, továbbá 16 bites mintaméretet, kétcsatornás működést és 44100 b/s mintavételi gyakoriságot választunk. A mintavételi gyakoriságnál ügyeljünk arra, hogy a hangeszköz nem biztos, hogy minden mintavételi gyakoriságot megfelelően támogat. A kívánt értékhez legközelebb álló és támogatott mintavételi gyakoriságot az `snd_pcm_hw_params_set_rate_near` függvénnyel tudjuk kiválasztani. A hardverbeállítások addig nem lépnek érvénybe, amíg az `snd_pcm_hw_params` függvényt meg nem hívjuk.

A program fennmaradó része különféle a ***PCM*** adatfolyamra vonatkozó beállításokat jelenít meg, köztük az időszak hosszát és a puffermétereket. A megjelenő eredmények a hangeszköztől nagymértékben függenek.

Miután lefuttattuk a programot saját gépünkön, próbáljunk kísérletezni egy kicsit, és végezzünk el néhány módosítást. Változtassuk az eszköznevet `default`-ról `hw:0,0`-ra vagy `plughw:`-re, és vizsgáljuk meg a módosítás eredményét. Módosítsuk a hardverbeállításokat is, és vizsgáljuk meg, hogy a megjelenő értékek miben térnek el.

A *3. kódrészlet* az előbbi példa kibővítése, ebben hangmintákat is írunk a hangkártyára, vagyis hangokat szólaltatunk

2. kódrészlet PCM eszköz megnyitása és a beállítások megadása

```
/*

A példaprogram az alapértelmezett PCM eszközt
nyitja meg, megad néhány beállítást, majd a meg-
jeleníti a hardveres beállítások túlnyomó része-
nek értékét. Hangrögzítést vagy -lejátszást nem
végez.

*/

/* Az újabb ALSA API használata*/

#define ALSA_PCM_NEW_HW_PARAMS_API

/* Az ALSA könyvtári API megadása teljes
egészében ebben a fejrészben található
*/

#include <alsa/asoundlib.h>

int main() {
  int rc;
  snd_pcm_t *handle;
  snd_pcm_hw_params_t *params;
  unsigned int val, val2;
  int dir;
  snd_pcm_uframes_t frames;

  /* PCM eszköz megnyitása lejátszásra */

  rc = snd_pcm_open(&handle, “default”,
                    SND_PCM_STREAM_PLAYBACK, 0);
  if (rc < 0) {
    fprintf(stderr,
            “A PCM eszközt nem lehet megnyitni:
             ➥%s\n”, snd_strerror(rc));
    exit(1);
  }

  /* A hardverbeállításokat tároló objektum
     lefoglalása */

  snd_pcm_hw_params_alloca(&params);

  /* Feltöltése alapértékekkel */

  snd_pcm_hw_params_any(handle, params);

  /* A kívánt hardverbeállítások megadása */

  /* Átlapolt mód */

  snd_pcm_hw_params_set_access(handle, params,
    ➥SND_PCM_ACCESS_RW_INTERLEAVED);

  /* Előjeles, 16 bites, kis indián formátum */

  snd_pcm_hw_params_set_format(handle, params,
    ➥SND_PCM_FORMAT_S16_LE);

/* Két csatorna (sztereó) */

  snd_pcm_hw_params_set_channels(handle, params,
  ➥2);

  /* 44100 bit/másodperc mintavételi gyakoriság
     (CD minőség) */

  val = 44100;
  snd_pcm_hw_params_set_rate_near(handle,
    ➥params, &val, &dir);

  /* A beállítások kiírása az illesztőprogram
     felé */

  rc = snd_pcm_hw_params(handle, params);
  if (rc < 0) {
    fprintf(stderr, “A hardverbeállításokat
                  ➥nem sikerült megadni: %s\n”,
                  ➥snd_strerror(rc));
    exit(1);
  }

  /* A PCM felület adatainak megjelenítése */

  printf(“PCM kezelő neve = ´%s´\n”,
         ➥snd_pcm_name(handle));

  printf(“PCM állapot = %s\n”,
    ➥snd_pcm_state_name(snd_pcm_state(handle)));
  snd_pcm_hw_params_get_access(params,
    ➥(snd_pcm_access_t *) &val);

  printf(“Hozzáférés típusa = %s\n”,
   ➥snd_pcm_access_name((snd_pcm_access_t)val));

  snd_pcm_hw_params_get_format(params, &val);
  printf(“Formátum = ´%s´ (%s)\n”,
   ➥snd_pcm_format_name((snd_pcm_format_t)val),
   ➥snd_pcm_format_description(
   ➥(snd_pcm_format_t)val));

  snd_pcm_hw_params_get_subformat(params,
  ➥(snd_pcm_subformat_t *)&val);
  printf(“Alformátum = ´%s´ (%s)\n”,
   ➥snd_pcm_subformat_name
   ➥((snd_pcm_subformat_t)val),
   ➥snd_pcm_subformat_description(
   ➥(snd_pcm_subformat_t)val));

  snd_pcm_hw_params_get_channels(params, &val);
```

```
printf("Csatornák = %d\n", val);
snd_pcm_hw_params_get_rate(params, &val, &dir);
printf("Gyakoriság = %d bps\n", val);

snd_pcm_hw_params_get_period_time(params,
➥&val, &dir);
printf("Időszakhossz = %d us\n", val);

snd_pcm_hw_params_get_period_size(params,
➥&frames, &dir);
printf("Időszakméret = %d frames\n",
        ➥(int)frames);

snd_pcm_hw_params_get_buffer_time(params,
➥&val, &dir);
printf("Pufferidő = %d us\n", val);

snd_pcm_hw_params_get_buffer_size(params,
➥(snd_pcm_uframes_t *) &val);
printf("Pufferméret = %d frames\n", val);

snd_pcm_hw_params_get_periods(params, &val,
➥&dir);
printf("Időszak/puffer = %d frames\n", val);

snd_pcm_hw_params_get_rate_numden(params,
➥&val, &val2);
printf("Pontos gyakoriság = %d/%d bps\n", val,
➥val2);

val = snd_pcm_hw_params_get_sbits(params);
printf("Számottevő bitek = %d\n", val);

snd_pcm_hw_params_get_tick_time(params,
➥&val, &dir);
printf("Ütésidő = %d us\n", val);

val = snd_pcm_hw_params_is_batch(params);
printf("Kötegelés = %d\n", val);

  val = snd_pcm_hw_params_is_block_transfer
  ➥(params);
  printf("Blokkos átvitel = %d\n", val);

  val = snd_pcm_hw_params_is_double(params);
  printf("Double érték = %d\n", val);

  val = snd_pcm_hw_params_is_half_duplex(params);
  printf("Váltakozó kétirányú = %d\n", val);

  val = snd_pcm_hw_params_is_joint_duplex
  ➥(params);
  printf("Csatolt kétirányú = %d\n", val);

  val = snd_pcm_hw_params_can_overrange(params);
  printf("Tartomány túllépési lehetőség = %d\n",
  ➥val);

  val =
  ➥snd_pcm_hw_params_can_mmap_sample_resolution
  ➥(params);
  printf("mmap lehetőség = %d\n", val);

  val = snd_pcm_hw_params_can_pause(params);
  printf("Szüneteltetési lehetőség = %d\n", val);

  val = snd_pcm_hw_params_can_resume(params);
  printf("Folytatási lehetőség = %d\n", val);

  val = snd_pcm_hw_params_can_sync_start(params);
  printf("Szinkron indítási lehetőség = %d\n",
  ➥val);

  snd_pcm_close(handle);

  return 0;
}
```

meg. A szabványos bemenetről összegyűjtünk annyi bájtot, amennyi egy időszakra elegendő, majd öt másodpercnyi adatot írunk ki a kártya felé.

A program eleje az előzővel megegyező – először megnyitjuk a *PCM* eszközt, majd megadjuk a hardverbeállításokat. Az *ALSA* által kiválasztott időszakhosszt használjuk, és ennek megfelelően állítjuk be a mintákat tároló puffer méretét is. Ezután meghatározzuk az időszakhosszt, amiből tudjuk, hogy a programnak hány időszakot kell feldolgoznia ahhoz, hogy öt másodpercnyi hanganyag álljon elő.

Az adatokat kezelő hurokban a szabványos bemenetről olvasunk, a puffert pedig egy időszaknyi mintával töltjük fel. Ha fájl végéről olvasunk, esetleg a kapott bájtok száma a várttól eltérő, akkor hibakezelést végzünk.

A *PCM* eszközre az adatokat az `snd_pcm_writei` hívással írjuk ki. Ennek működése a rendszermag `write` rendszerhívásához nagyon hasonló, kivéve azt, hogy a méret keretekben van megadva. Ellenőrizzük a visszatérési értéket, ebből különféle hibákra tudunk következtetni. Az `EPIPE` visszatérési érték alulcsordulásra utal, amelynek hatására a *PCM* adatfolyam `XRUN` állapotba kerül, az adatfeldolgozás pedig leáll. Ebből az állapotból normál esetben az `snd_pcm_prepare` függvény meghívásával billenthetjük ki a folyamot, amely ilyenkor `PREPARED` (előkészített) állapotba kerül. Ezután a következő alkalommal újra írhatunk adatokat a folyamba. Ha egyéb hibajelzést kapunk, akkor kijelezzük a kódját, majd továbblépünk. Végül, ha a kiírt keretek száma nem egyezik meg az elvárttal, szintén hibaüzenetet jelenítünk meg.

A program addig ismétli a műveletet, amíg öt másodpercnyi keret össze nem gyűlik, el nem éri a fájl végét vagy olvasási hiba nem történik a bemeneten. Ekkor kerül sor az `snd_pcm_drain` meghívására, amivel engedélyezzük a függőben lévő hangminták továbbítását, illetve lezárjuk a folyamot. Felszabadítjuk a dinamikusan foglalt puffert, majd kilépünk.

Bizonyára sokaknak feltűnt, hogy a program nagyjából működésképtelen, amíg a bemenetet át nem irányítjuk a kon-

3. kódrészlet Egyszerű hanglejátszás

```
/*

A példaprogram a szabványos bemenetről olvas,
majd öt másodpercnyi adatot ír az alapértelmezett
PCM eszközre.

*/

/* Az újabb ALSA API használata*/
#define ALSA_PCM_NEW_HW_PARAMS_API

#include <alsa/asoundlib.h>

int main() {
  long loops;
  int rc;
  int size;
  snd_pcm_t *handle;
  snd_pcm_hw_params_t *params;
  unsigned int val;
  int dir;
  snd_pcm_uframes_t frames;
  char *buffer;

  /* PCM eszköz megnyitása lejátszásra */
  rc = snd_pcm_open(&handle, "default",
  ➥SND_PCM_STREAM_PLAYBACK, 0);
  if (rc < 0) {
    fprintf(stderr,
           ➥"A PCM eszközt nem lehet megnyitni:
           ➥%s\n", snd_strerror(rc));
    exit(1);
  }

  /* A hardverbeállításokat tároló objektum
     lefoglalása */

  snd_pcm_hw_params_alloca(&params);

  /* Feltöltése alapértékekkel */
  snd_pcm_hw_params_any(handle, params);

  /* A kívánt hardverbeállítások megadása */

  /* Átlapolt mód */
  snd_pcm_hw_params_set_access(handle, params,
  ➥SND_PCM_ACCESS_RW_INTERLEAVED);

  /* Előjeles, 16 bites, kis indián formátum */
  snd_pcm_hw_params_set_format(handle, params,
  ➥SND_PCM_FORMAT_S16_LE);

  /* Két csatorna (sztereó) */
  snd_pcm_hw_params_set_channels(handle, params,
  ➥2);

  /* 44100 bit/másodperc mintavételi gyakoriság
     (CD minőség) */

  val = 44100;
  snd_pcm_hw_params_set_rate_near(handle, params,
  ➥&val, &dir);

  /* Az időszakméret beállítása 32 keretre */
  frames = 32;
  snd_pcm_hw_params_set_period_size_near(handle,
  ➥params, &frames, &dir);

  /* A beállítások kiírása az illesztőprogram
     felé */

  rc = snd_pcm_hw_params(handle, params);
  if (rc < 0) {
    fprintf(stderr,
           ➥"A hardverbeállításokat nem sikerült
           ➥megadni: %s\n",
           ➥snd_strerror(rc));
    exit(1);
  }

  /* Kellően nagy méretű, egy időszakot tárolni
     képes puffert használunk. */

  snd_pcm_hw_params_get_period_size(params,
  ➥&frames, &dir);

  /* 2 bájt/minta, 2 csatorna */
  size = frames * 4;
  buffer = (char *) malloc(size);

  /* A hurkot öt másodpercig futtatjuk */
  snd_pcm_hw_params_get_period_time(params,
  ➥&val, &dir);

  /* 5 másodperc mikroszekundumokban mérve,
     osztva az időszakkal */
  loops = 5000000 / val;

  while (loops > 0) {
    loops--;
    rc = read(0, buffer, size);
    if (rc == 0) {
      fprintf(stderr, "Bemeneti fájl vége\n");
      break;
    } else if (rc != size) {
      fprintf(stderr,
             ➥"rövid bemenet: %d bájtot sikerült
             ➥olvasni\n", rc);
    }
    rc = snd_pcm_writei(handle, buffer, frames);
    if (rc == -EPIPE) {
      /* Az EPIPE alulcsordulást jelez. */
      fprintf(stderr, "Alulcsordulás történt\n");
      snd_pcm_prepare(handle);
```

```
    } else if (rc < 0) {
      fprintf(stderr,
              "writei hiba: %s\n",
              snd_strerror(rc));
    }  else if (rc != (int)frames) {
      fprintf(stderr,
              ➥"Rövid írás, %d keretet sikerült
              ➥kiírni\n", rc);
    }
  }

  snd_pcm_drain(handle);
  snd_pcm_close(handle);
  free(buffer);

  return 0;
}
```

4. kódrészlet Hangfelvétel

```
/*

A példaprogram az alapértelmezett PCM eszközről
olvas, majd öt másodpercnyi adatot ír a szabvá-
nyos kimenetre.

*/

/* Az újabb ALSA API használata*/
#define ALSA_PCM_NEW_HW_PARAMS_API

#include <alsa/asoundlib.h>

int main() {
  long loops;
  int rc;
  int size;
  snd_pcm_t *handle;
  snd_pcm_hw_params_t *params;
  unsigned int val;
  int dir;
  snd_pcm_uframes_t frames;
  char *buffer;

  /* PCM eszköz megnyitása felvételre */
  rc = snd_pcm_open(&handle, "default",
  ➥SND_PCM_STREAM_CAPTURE, 0);
  if (rc < 0) {
    fprintf(stderr,
            ➥"A PCM eszközt nem lehet megnyitni:
            ➥%s\n",  snd_strerror(rc));
    exit(1);
  }

  /* A hardverbeállításokat tároló objektum
    lefoglalása */
  snd_pcm_hw_params_alloca(&params);

  /* Feltöltése alapértékekkel */
  snd_pcm_hw_params_any(handle, params);

  /* A kívánt hardverbeállítások megadása */

  /* Átlapolt mód */
  snd_pcm_hw_params_set_access(handle, params,
  ➥SND_PCM_ACCESS_RW_INTERLEAVED);

  /* Előjeles, 16 bites, kis indián formátum */
  snd_pcm_hw_params_set_format(handle, params,
  ➥SND_PCM_FORMAT_S16_LE);

  /* Két csatorna (sztereó) */
  snd_pcm_hw_params_set_channels(handle, params,
  ➥2);

  /* 44100 bit/másodperc mintavételi gyakoriság
     (CD minőség) */
  val = 44100;
  snd_pcm_hw_params_set_rate_near(handle, params,
  ➥&val, &dir);

  /* Az időszakméret beállítása 32 keretre */
  frames = 32;
  snd_pcm_hw_params_set_period_size_near(handle,
  ➥params, &frames, &dir);

  /* A beállítások kiírása az illesztőprogram
     felé */
  rc = snd_pcm_hw_params(handle, params);
  if (rc < 0) {
    fprintf(stderr,
            ➥"A hardverbeállításokat nem sikerült
            ➥megadni: %s\n",
            ➥snd_strerror(rc));
    exit(1);
  }

  /* Kellően nagy méretű, egy időszakot tárolni
     képes puffert használunk. */
  snd_pcm_hw_params_get_period_size(params,
  ➥&frames, &dir);

  /* 2 bájt/minta, 2 csatorna */
  size = frames * 4;
  buffer = (char *) malloc(size);

  /* A hurkot öt másodpercig futtatjuk */
  snd_pcm_hw_params_get_period_time(params,
  ➥&val, &dir);
  loops = 5000000 / val;
```

```
while (loops > 0) {
  loops--;
  rc = snd_pcm_readi(handle, buffer, frames);
  if (rc == -EPIPE) {
    /* Az EPIPE túlcsordulást jelez. */
    fprintf(stderr, "Túlcsordulás történt\n");
    snd_pcm_prepare(handle);
  } else if (rc < 0) {
    fprintf(stderr,
            "read hiba: %s\n",
            snd_strerror(rc));
  }  else if (rc != (int)frames) {
    fprintf(stderr, "Rövid olvasás, %d keretet
                    sikerült beolvasni\n", rc);
  }
  rc = write(1, buffer, size);
  if (rc != size)
    fprintf(stderr,
            "Rövid írás: %d bájtot sikerült
            kiírni\n", rc);
}

snd_pcm_drain(handle);
snd_pcm_close(handle);
free(buffer);

return 0;
}
```

zolról. Megpróbálkozhatunk például a véletlenszerű adatokat előállító */dev/urandom* eszközzel:

```
./pelda3 < /dev/urandom
```

A véletlenszerű adatokkal öt másodpercnyi fehérzajt tudunk előállítani. Következő lépésként megpróbálhatjuk a */dev/null*-t vagy */dev/zero*-t használni bemenetként, majd összehasonlíthatjuk az eredményt. Változtassunk meg néhány beállítást, például a mintavételi időt vagy az adatformátumot, és figyeljük a hatást.

A *4. kódrészlet* sokban hasonlít a harmadikhoz, ám itt ***PCM*** alapú felvételt végzünk. A ***PCM*** adatfolyam megnyitásakor `SND_PCM_STREAM_CAPTURE` üzemmódot állítunk be. A fő adatfeldolgozó hurokban az `snd_pcm_readi` függvénnyel olvassuk be a mintákat a hangeszközről, majd `write` hívásokkal juttatjuk a szabványos kimenetre. Eközben ellenőrizzük, hogy nincs-e alulcsordulás, amit szükség esetén a 3. kódrészletben látott módszerrel kezelünk.

A 4. kódrészletet futtatva nagyjából öt másodpercnyi adatot tudunk összegyűjteni, illetve a szabványos kimenetre küldeni, amit természetesen fájlba is írhatunk. Ha hangkártyánkhoz mikrofon is csatlakozik, akkor keverőprogram segítségével kiválaszthatjuk azt a felvétel forrásaként, illetve beállíthatjuk a felvételi szintet. Ha gondoljuk, elindíthatunk valamilyen CD-lejátszó programot is, és a felvételi forrást a CD-re állíthatjuk. Próbáljuk meg futtatni a 4. kódrészletet, a kimenetet pedig fájlba irányítani. Ezután a 3. kódrészlettel visszajátszhatjuk, amit rögzítettünk:

```
./pelda4 > hang.raw
./pelda3 < hang.raw
```

Ha hangkártyánk támogatja a teljes kétirányú működést, akkor a programokat csővezetékkel egymáshoz is csatolhatjuk, és a hangkártyáról érkező anyagot a következő paranccsal hallgatjuk meg: `./pelda4 | ./pelda3`. A ***PCM*** beállítások megváltoztatásával kipróbálhatjuk, hogy a mintavételi gyakoriság és a formátum módosítása milyen hatással jár.

Különleges lehetőségek

A példaprogramokban a ***PCM*** folyamokat blokkoló módban nyitottuk meg, vagyis a hívások addig nem tértek vissza, amíg az adatátvitel le nem játszódott. Interaktív, eseményvezérelt alkalmazásnál ezzel az alkalmazás futásának elfogadhatatlanul hosszú időkre való megakadását okozhatjuk. Az ***ALSA*** lehetővé teszi, hogy az adatfolyamokat nem blokkoló módban nyissuk meg, ilyenkor az olvasási és írási hívások azonnal visszatérnek. Ha az adatátvitelek függőben maradnak, és a hívások befejezésére nincs lehetőség, az ***ALSA*** az `EBUSY` foglaltságot jelző hibakódot adja vissza. Sok grafikus alkalmazás visszahívókkal kezeli az eseményeket. Az ***ALSA*** támogatja a ***PCM*** adatfolyamok aszinkron módban való megnyitását is, így mód nyílik arra, hogy adott mennyiségű mintaadat összegyűjtése után egy visszahívót futtassunk.

Az `snd_pcm_readi` és az `snd_pcm_writei` hívások használata a linuxos `read` és `write` rendszerhívásához hasonló. Az `i` betű arra utal, hogy a keretek átlapoltak (***interleaved***). Hasonló függvények nem átlapolt módhoz is léteznek. Linux alatt sok eszköz támogatja az `mmap` rendszerhívást, amelynek segítségével memóriabeli területeknek lehet megfeleltetni és mutatókkal lehet kezelni őket. Végül megemlíteném, hogy az ***ALSA*** támogatja a ***PCM*** csatornák `mmap` módban való megnyitását is, amivel hatékony, másolásmentes hozzáférést biztosít a hangadatokhoz.

Összefoglalás

Remélem, hogy írásommal sokaknak sikerült kedvet szereznem az ***ALSA*** kipróbálásához. Ahogy a terjesztések egyre inkább a 2.6-os rendszermag használatára térnek át, az ***ALSA*** egyre szélesebb körben fog elterjedni, fejlett szolgáltatásainak köszönhetően egyre jobb linuxos hangkezelő megoldások jelenhetnek majd meg.

Szeretnék köszönetet mondani ***Jaroslav Kyselának*** és ***Takashi Iwainak***, akik írásom vázlatának áttekintésével és tanácsaikkal rengeteget segítettek nekem.

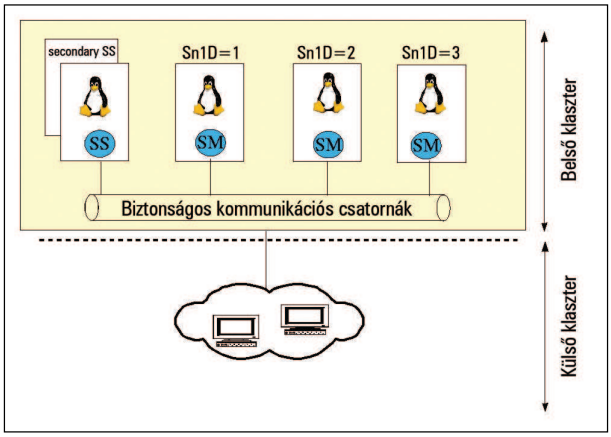
Linux Journal 2004. október, 126. szám

Jeff Tranter 1992 óta használja, fejleszti és tárgyalja írásaiban a Linuxot. A kanadai Ottawában található Xandros Corporation munkatársa.

Virtuális biztonsági zónák létrehozása egy Linux fürtön belül

Az osztott biztonsági háttér lehetővé teszi, hogy egy Linux fürtön belül különálló virtuális biztonsági zónákat alakítsunk ki. Ez az írás ezek megvalósításáról és használatáról ad áttekintést.

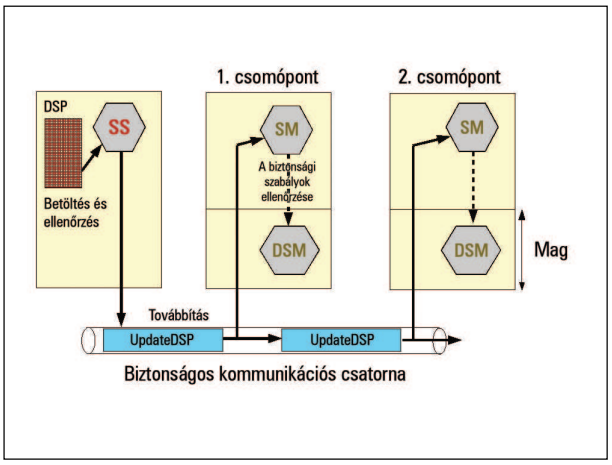
Egyre több projektben használják a ***Linuxot*** és más nyílt forrású programokat fürtözött számítógéprendszerek alapvető építőelemeként. A példák sora a mozifilmek gyártása során használt vizuális effektusoka előállító, óriási tömegű számítást végző fürtöktől a következő generációs telekommunikációs kiszolgálótelepekig terjed.
A technika fejlődésével és a méretek növekedésével párhuzamosan egyre gyakrabban fordul elő az is, hogy különféle szempontok – például a gazdaságosság, a kezelhetőség vagy a rugalmasság fokozása végett – célszerű ugyanazon a fürtön több különböző alkalmazást is futtatni. Jó példa erre a távközlés világában alkalmazott úgynevezett ***carrier-grade*** osztályú kiszolgálótelepek, amelyek erőforrásait eleve több különböző operátor között osztják meg. A közös használat ilyenkor azt jelenti, hogy az operátorok osztoznak a fürt teljes infrastruktúráján, ezeket használva esetleg egészen eltérő szolgáltatásokat nyújtanak az ügyfeleik számára, ugyanakkor a programjaikat és adataikat – nyilvánvaló gazdasági és biztonsági megfontolások miatt – nem szeretnék mások számára is elérhetővé tenni. Ezekben az esetekben a fürtök rendszergazdái sem férnek hozzá a programok forráskódjaihoz, és a biztonsági eljárásokat sem lehet a forráskód szintjén kötelezővé tenni. Szükség van tehát egy olyan biztonsági háttérre, amely biztosítja, hogy egy adott alkalmazás erőforrásait ne használhassák mások, illetve az adott szoftver működését ne zavarják meg a fürt más elemei.
Az ilyen helyzetekre az úgynevezett osztott biztonsági infrastruktúra (***DSI***, ***Distributed Security Infrastructure***) megoldást jelenthet. Ez a ***carrier-grade*** osztályú ***Linux*** telepeknél a fürt virtuális alfürtökre osztásával kísérli meg egy összefüggő biztonsági keretrendszer kialakítását. Biztosítja az alegységek közti kapcsolatok ellenőrzését, illetve a rendszergazdák által megadott szabályok alapján történő korlátozását. Annak ellenére, hogy a fejlesztések még csak két éve kezdődtek el, úgy gondoljuk, a ***DSI*** rendkívül hasznos eszköz lesz a fürtök rendszergazdáinak kezében. Ebben a cikkben bemutatjuk, hogy milyen módon használhatjuk a ***DSI***-t a ***Linux*** telepeken belüli virtuális biztonsági zónák kialakítására.



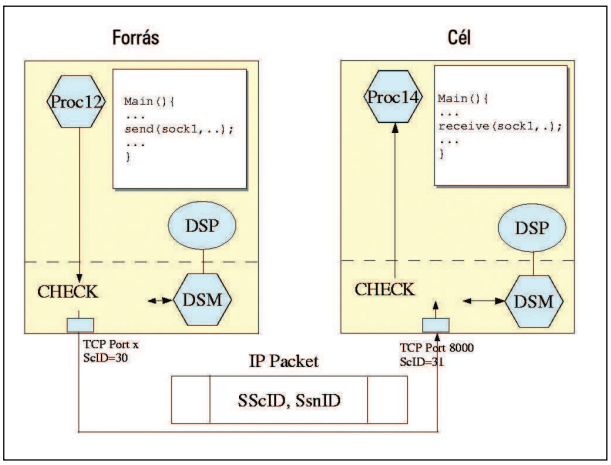
1. ábra A DSI felépítése

A DSI felépítése és eszközei

Először vizsgáljuk meg röviden a ***DSI*** felépítését. A ***DSI*** egy ***biztonsági kiszolgálóból*** (***SS***, ***Security Server***) és csomópontonként egy-egy ***biztonság-vezérlőből*** (***SM***, ***Security Manager***) épül fel (*1. ábra*). A biztonsági kiszolgálóban összpontosul a fürt ellenőrzése. Ez gyűjti össze a biztonsági vezérlők felől érkező figyelmeztetéseket és riasztásokat, és ez propagálja a fürtön belül az egyedi biztonsági intézkedéseket. A másik oldalon a biztonsági-vezérlők felelősek a biztonsági intézkedések betartatásáért, de kizárólag a saját csomópontjukon belül. A biztonsági kiszolgáló és a biztonsági vezérlők közt titkosított és hitelesített csatornákon zajlanak az üzenetváltások, amelyek a ***CORBA*** eseménycsatorna felett működő ***SSL/TLS*** használatával kerülnek továbbításra.
A ***DSI*** biztonsági elvei a folyamatok szintjén kerültek megvalósításra, vagyis a rendszer egy adott folyamat bizonyos erőforráshoz kötődő hozzáférési jogosultságait ellenőrzi. A folyamatok azonosítására a ***biztonsági kontextus azonosító*** (***ScID***) és annak a csomópontnak az azonosítója (***SnID***) szolgál, amelyen a folyamat fut. Ez a két adat egyértelműen meghatározza a rendszer számára a kérdéses folyamatot.



2. ábra A DSP elterjesztése a fürtön belül



3. ábra Biztonságos távoli elérés ellenőrzés

Az ***SnID***-ket a ***DSI SetNodeID*** nevű eszköze osztja ki. Minden azonos biztonsági kontextusba tartozó folyamat ugyanazzal az ***ScID***-vel rendelkezik. Az ***ScID***-k hozzárendelése történhet önműködően. Ilyenkor a rendszer osztja ki ezeket a megadott ***DSP***-szabályok szerint (lásd lejjebb). Ugyanakkor oszthatunk ki azonosítókat a ***DSI SetSID*** eszközével is, amikor egy konkrétan megnevezett futtatható állományhoz rendelünk hozzá kontextust. Ez utóbbi lehetővé teszi maguknak a futtatható állományoknak a biztonsági kontextus szerinti csoportosítását, vagyis a szabályrendszer nem csak a már futó folyamatokra alkalmazható.

A DSP beállítófájlja

Egy biztonsági szabály meghatározása a ***DSI***-ben valamilyen engedély megadását vagy megtagadását jelenti egy ***SnID-ScID*** párra vonatkozóan. Ezek a szabályok az egész fürtre érvényesek. A könnyű kezelhetőség érdekében minden szabály egy a biztonsági kiszolgálón lévő központi ***XML*** fájlban tárolódik.

A ***DSI*** lehetőséget biztosít az egész fürtözött telep egyedi és egységes kezelésére, valamint a szabályok frissítésének és alkalmazásának önműködő végrehajtására. Amint az adminisztrátor módosít egy már létező szabályt vagy újat hoz létre az osztott biztonsági házirendben (***DSP***, ***Distributed Security Policy***), a ***DSP***-t a ***dsiUpdatePolicy*** eszköz segítségével be kell tölteni a biztonsági kiszolgálóra. Ezután a ***dsiUpdatePolicy*** egyezteti a ***DSP***-t és a ***DSP XML*** sémáját (szintaktikai ellenőrzés). Amennyiben a ***DSP*** megerősítést nyer, a biztonsági kiszolgáló a biztonsági csatornákon keresztül a fürt minden csomópontjához eljuttatja az új szabályokat. Végül minden biztonsági vezérlő a ***DSM*** hívásával rendszermag szinten érvényesíti ezeket a szabályokat (lásd *2. ábra*). A ***DSM*** az ***LSM*** rendszermag-foltra épül. Ennek részletes ismertetésére nincs ebben a cikkben lehetőség, de a hálózaton elérhető (➲ ***www.linuxjournal/article/7688***) részben találhatunk hivatkozásokat erre vonatkozólag.

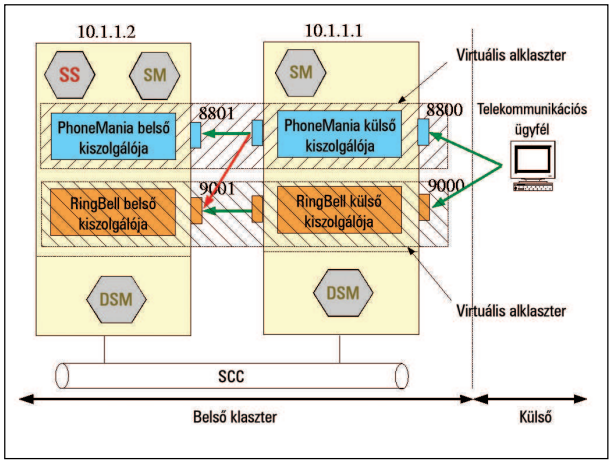
Osztott hozzáférés-ellenőrzés

A helyi erőforrásokhoz való hozzáférés ellenőrzése viszonylag egyszerű. A ***DSM***-modul megkapja a kérést kibocsátó folyamathoz tartozó ***ScID***-t és ***SnID***-t, majd ellenőrzi az érvényben lévő biztonsági szabályok között, hogy az rendelkezik-e a a megfelelő engedélyekkel. A ***DSI***-ben tulajdonképpen az az igazán eredeti, hogy megosztott módon végzi a hozzáférések ellenőrzését. Egyelőre még csak a foglalatok közti párbeszédet valósították meg. Ennek bemutatására egy olyan hozzáférés-ellenőrzési helyzetet mutatunk be, amikor a folyamat egy másik csomóponton lévő erőforrást próbál meg elérni (*3. ábra*):

- A hozzáférési kérést a helyi ***DSM*** fogadja, majd ellenőrzi, hogy a folyamatnak egyáltalán van-e joga a helyi foglalatokkal kapcsolatos rendszerhívások kibocsátására.
- Ezután a ***DSM*** a távoli csomópontnak elküldött minden IP-csomaghoz hozzáfűzi a kérő folyamathoz tartozó ***ScID*** és ***SnID*** azonosítókat.
- A fogadó csomóponton a távoli ***DSM*** a kérést indító folyamat az IP-csomagokból visszanyeri a ***ScID*** és ***SnID*** azonosítókat, majd ezeket használva ellenőrzi, hogy a kérő folyamat rendelkezik-e engedéllyel a célkerettel és a célkeretet birtokló folyamattal való adatcseréhez.
- Végül a távoli ***DSM*** helyben ellenőrzi, hogy a folyamat, amelyhez a megcélzott keret tartozik, fogadhat-e információt a kérést kezdeményező folyamattól.

A probléma

Ebben a részben részletesebben megvizsgálunk egy viszonylag egyszerű esetet, amelyen keresztül közérthetően bemutatható, hogy egy adott problémát hogyan lehet a ***DSI*** rendszer segítségével megoldani. Tegyük fel, hogy egy két csomópontból álló egységet szeretnénk megosztani két távközlési operátor között, akiket most ***PhoneMania*** és ***RingBell*** néven fogunk emlegetni, s akik a saját alkalmazásaikat futtatják a fürt csomópontjain. Mindketten telefonos reklám- és információszolgáltatást nyújtanak, amely abban merül ki, hogy a végfelhasználók felhívhatják a (***TelecomClient***-et használó) belépési pontokon lévő kiszolgálókat és lekérhetik onnan a megadott vállalatok árajánlatait. A belépési pontokon lévő kiszolgálók (***PhoneManiaEP*** és ***RingBellEP***) továbbítják a kérést a háttérben működő kiszolgálóknak (***PhoneManiaBE*** és ***RingBellBE***), amelyektől aztán megkapják az ajánlatokat és visszaküldik azokat a végfelhasználónak.



4. ábra Egy egyszerű távközlési eset

Ha valaki hozzászokott a fürtözött rendszerben való gondolkodáshoz, ezen a ponton a következő kérdés merül fel: hogyan védhetnénk meg egy ***PhoneMania*** alkalmazást attól, hogy a kéréseit tévedésből a ***RingBell*** háttérkiszolgálóinak továbbítsa? Ha semmilyen erre vonatkozó biztonsági rendszabályt nem léptetünk életbe, akkor a ***PhoneMania*** megtehetné ezt olyan esetekben, amikor a háttérkiszolgálója túlterhelt, vagy egyszerűen csak nem rendelkezik a kért információval. És akkor még nem is említettük az olyan súlyosabb eseteket, amikor mondjuk az előfizetők adatait akarják megszerezni illetéktelenek, vagy a versenytársnak akarnak szándékosan kárt okozni. Egy ilyen szituáció bemutatására minden szereplőt egyszerű ***UDP***-ügyfélként és -kiszolgálóként valósítottunk meg (*4. ábra*).
Az „eltévelyedés” forgatókönyve lépésenként a következő:

- `PhoneMania` és `RingBell` egy `munster` nevű csomóponton futtatják a háttérkiszolgálóikat:

```
[munster demo]$ ./RingBellBE -h 10.1.1.2 -p 9001
RINGBELL: bind on 10.1.1.2:9001
..
[munster demo]$ ./PhoneManiaBE -h 10.1.1.2
➥-p 8801
PHONEMANIA: bind on 10.1.1.2:8801
```

- Ahogy `PhoneMania` túlterheltté válik, úgy dönt, hogy `RingBell` erőforrásait fogja használni, így a `colby` csomóponton lévő `PhoneMania` belépési pont kiszolgálója (8800-as kapu) az ügyfeleitől érkező összes kérést a `RingBell` háttérkiszolgálóira irányítja (9001-es kapu):

```
[colby demo]$ ./PhoneManiaEP -h 10.1.1.1
➥-p 8800 -b 10.1.1.2 -r 9001
PHONEMANIA: bind on 10.1.1.1:8800
PHONEMANIA: connect on 10.1.1.2:9001
..
```

- Amikor egy ügyfél a `PhoneMania` belépési pontján (8800-as kapu) egy ajánlatot kér, `PhoneMania` valójában `RingBell` háttérkiszolgálóját használja a válasz megadására (9001-es kapu). Ennek pedig az lesz a – nem túl mulatságos – következménye, hogy `PhoneMania` kapja a pénzt `RingBell` erőforrásainak használatáért.

```
[colby demo]$ ./TelecomClient -h 10.1.1.1
➥-p 8800
Connecting to : 10.1.1.1:8800

Requesting quotation for Ericsson
Quote Ericsson
..
[munster demo]$
..
RINGBELL backend : processing quotation request
for Ericsson

RINGBELL backend : quotation for Ericsson is 83
Quote Ericsson
```

Ennek megelőzésére az egyik lehetséges megoldás az, ha a ***DSI*** segítségével a megosztott fürtöt biztonsági szempontból különálló egységekre, alfürtökre osztjuk fel. A következőkben lépésenként bemutatjuk, hogyan használható a ***DSI*** ennek a gyakorlati megvalósítására.

A DSI telepítése és beállítása

Először is a fürt minden csomópontjára telepítenünk kel a ***DSI***-t. Miután a ***SourceForge*** honlapjáról letöltöttük a ***DSI tar***-csomagját, le kell fordítanunk a saját gépünkön, mivel a program a szabványos fordítóeljárást alkalmazza.
A ***SourceForge*** oldalán található ***DSI***-leírásban részletesen ismertetjük a ***DSI*** fordításának és telepítésének lépéseit.
Minden csomóponton futtatni kell a ***Security Managert***, ami a kétcsomópontos fürtünk esetében ez azt jelenti, hogy a program a `colby` és `munster` csomópontokon fog futni:

```
[colby]$ cd ~/dsi
[colby]$ source dsi_setup.sh
[colby]$ ~/dsi/bin/dsiSecManager
```

Az egyszerűség kedvéért a `colby` a biztonsági kiszolgáló szerepét is betölti:

```
[colby]$ cd ~/dsi
[colby]$ source dsi_setup.sh
[colby]$ ~/dsi/bin/dsiSecServer
```

A biztonsági kiszolgáló és a biztonság-vezérlők egymással a ***CORBA*** eseménycsatornák használatával tartják a kapcsolatot.
Minden csomóponton betöltjük a ***DSI*** redszermagmodulját, a ***DSM***-et, amellyel a rendszermag szintjén biztosítható a biztonsági intézkedések betartása:

```
$ cd ~/dsi/lsm
$ su root
Password:
# ./load
# /sbin/lsmod
```

```
Module    Size  Used by    Not tainted
dsm      36332   0 (unused)
...
```

Ezután beállítjuk a ***DSI***-t külön IP-cím megadásával az egyes csomópontok számára a biztonságos és nem biztonságos adatkapcsolat számára. Ehhez egy `DciInit` nevű eszközt írtunk, amelynek használatáról és a ***dci_policy.conf*** fájl felépítéséről a ***SourceForge*** honlapján található ***DCI***-leírásban olvashatunk részletesebben:

```
$ cd ~/dsi/user/tools
$ ./DciInit ~/dsi/etc/dci_policy.conf
```

A megoldás: virtuális alfürtök létrehozása

Ahhoz, hogy különálló virtuális alfürtöket hozhassunk létre, lényegében külön ***ScID*** azonosítókat kell létrehoznunk a ***PhoneMania*** (a példánkban `ScID=10`) és a ***RingBell*** (`ScID=20`) erőforrásai számára. Ezután új szabályt léptetünk érvénybe a ***DSP***-ben, amely alapján a rendszer visszautasít bármilyen kapcsolatkérést, ami az `ScID=10` érték által meghatározott zónából az `ScID=20` övezetbe érkezik, illetve fordítva. Az egyes operátorokhoz tartozó erőforrások különálló csoportokba szervezésével és mindennemű köztük létrejövő kapcsolat tiltásával valójában a fürt egy virtuális felosztását értük el. Ehhez rendszerfelügyeleti céllal az adminisztrátor létrehozhatna még egy újabb zónát az `ScID=30` értékkel, amely mindkét övezethez rendelkezne hozzáféréssel.
Először is, rendeljük hozzá minden csomópont minden bináris állományához a megfelelő ***ScID*** azonosítót (ehhez a `SetSID` eszközt használjuk):

```
$ ~/dsi/user/tools/SetSID PhoneManiaEP 10
➥Changing from SID 0 to SID 10
$ ~/dsi/user/tools/SetSID PhoneManiaBE 10
➥Changing from SID 0 to SID 10
$ ~/dsi/user/tools/SetSID RingBellEP 20
➥Changing from SID 0 to SID 20
$ ~/dsi/user/tools/SetSID RingBellBE 20
➥Changing from SID 0 to SID 20
$ ~/dsi/user/tools/ls_dsi .
PERMISSION      USER     GROUP    BSID   FILE
-rwxr-xr-x      lmcaxpr install 10     PhoneManiaBE
-rwxr-xr-x      lmcaxpr install 20     RingBellBE
-rwxr-xr-x      lmcaxpr install 10     PhoneManiaEP
-rwxr-xr-x      lmcaxpr install 20     RingBellEP
```

Amikor a ***DSM*** betöltődik, foganatosítja az alapértelmezésként engedélyezett biztonsági szabályokat. A fürt felosztásának megvalósításához szerkesztenünk kell a ***DSP*** fájlját (***~/dsi/etc/SampleDSP.xml***) és minden már létező biztonsági szabályt a saját szabályainkkal kell helyettesítenünk.
A `PhoneMania` foglalatai az `ScID=10` azonosítót kapják, a `RingBell` pedig az `ScID=20` értéket használja. A következő szabály az `ScID=10` értéket rendeli a `PhoneMania` belépési pontjának ***UDP*** foglalatához (8800-as kapu):

```
<class_SOCKET_INIT_rule>
  <protocol>UDP</protocol>
  <port>8800</port>
  <SnID>ALL</SnID>
  <ScID>10</ScID>
</class_SOCKET_INIT_rule>
```

Három hasonló szabályra van még szükségünk: egy a `PhoneMania` háttérkiszolgálójához, két másik pedig az `ScID=20` érték `RingBell`-hez való rendeléséhez.
Ezután a `PhoneMania` folyamatainak (forrás `ScID=10`) engedélyezzük, hogy üzeneteket hozzanak létre, küldjenek vagy fogadjanak a saját foglalataikon (vagyis az `ScID=10` azonosítójú célokon):

```
<class_SOCKET_rule>
  <sScID>10</sScID>
  <sSnID>ALL</sSnID>
  <tScID>10</tScID>
  <tSnID>ALL</tSnID>
  <allow>CREATE CONNECT LISTEN RECEIVE
      ➥SEND</allow>
</class_SOCKET_rule>
```

A `RingBell` számára hasonló szabályt hozunk létre. Természetesen az `ScID=10` és 20 közti adatcserét le kell tiltanunk. Ezt egyszerűen úgy tehetjük meg, hogy nem adunk engedélyeket ezen ScID azonosítók közti adatcseréhez:

```
<class_SOCKET_rule>
  <sScID>10</sScID>
  <sSnID>ALL</sSnID>
  <tScID>20</tScID>
  <tSnID>ALL</tSnID>
  <allow></allow>
</class_SOCKET_rule>
```

Hasonló szabályt hozunk létre az `ScID=20` forrású és `ScID=10` célazonosító között.
Bár a háttérkiszolgálók és egy adott operátorhoz tartozó belépési pontok kiszolgálói fizikailag a fürt különböző csomópontjain helyezkedhet el, ne feledjük, hogy megosztott fürtről van szó, tehát nem rendeljük az egyes csomópontokat egyértelműen, elválaszthatatlanul a `RingBell`-hez vagy a `PhoneMania`-hoz. Így a `PhoneMania` folyamatainak (forrás `ScID=10`) képesnek kell lennie egy másik `PhoneMania`-folyamattal (cél `ScID=10`) történő kapcsolatfelvételre is a hálózaton keresztül. Ugyanez természetesen a `RingBell`-re is igaz.

```
<class_NETWORK_rule>
  <sScID>10</sScID>
  <sSnID>ALL</sSnID>
  <tScID>10</tScID>
  <tSnID>ALL</tSnID>
  <deny>NETWORK_RECEIVE</deny>
</class_NETWORK_rule>
```

Végül a `PhoneMania` (`ScID=10`) és `RingBell` (`ScID=20`) folyamatokat rendszerint valamilyen parancsértelmezőből futtatjuk (az alapértelmezett `ScID=2`). Ez nyilván csak úgy lehet-

séges, ha engedélyezzük a kérdéses héj számára új folyamatok létrehozását. Ezt egy `transition` szabály segítségével tehetjük meg:

```
<class_TRANSITION_rule>
  <parent_ScID> 2 </parent_ScID>
  <SnID>ALL</SnID>
  <binary_ScID>10</binary_ScID>
  <new_ScID>10</new_ScID>
</class_TRANSITION_rule>
```

A `binary_ScID` a bináris állományhoz explicite hozzárendelt ***ScID*** azonosító. Emlékezzünk rá, hogy a `PhoneManiaBE` és `PhoneManiaEP` programokhoz a `SetSID` segítségével rendeltünk ***ScID*** azonosítót. A `new_ScID` a létrejövő új folyamathoz rendelt ***ScID***. Mivel a 8800-as és 8801-es foglalatok elérése csak az `ScID=10` azonosítójú folyamatok számára (vagyis a `PhoneMania` számára) engedélyezett, az új folyamathoz is az `ScID=10` azonosítót kell rendelnünk. Hasonló szabályt kell a `RingBell` számára is létrehoznunk. Mindössze ennyire van szükségünk a ***DSP*** létrehozásához, vagyis 12 egyszerű biztonsági szabályra, amelyek érvényesítése tökéletesen kizárja a fent vázolt problémát. A szabályrendszer leírása után természetesen az egész fürtre kiterjedően frissítenünk kell a biztonsági rendszabályokat. Ehhez valamennyi biztonsági kiszolgálónak egy `update` (frissítés) üzenetet kell elküldenünk:

```
[colby]$ cd ~/dsi/SS/test/demoSecOM
[colby]$ ./dsiUpdatePolicy ~/dsi/etc/
➥DSP.xml
```

A biztonsági kiszolgáló ennek hatására beolvassa a módosított ***DSP*** fájlt (***~/dsi/etc/DSP.xml***) és figyelmeztetést küld, ha valamilyen formai hibát észlel. Ha mindent rendben talált, önműködően elküldi a frissítéseket minden biztonsági vezérlő számára, tehát nincs szükség arra, hogy a csomópontokra egyenként bejelentkezve minden gépen kézzel frissítsük a biztonsági intézkedéseket leíró adatbázist, vagy hogy saját Perl alapú rendszerkezelő programot fejlesszünk erre a célra, amely legalább részben automatizálja a folyamatot. Minden magától működik, mint az álom. Ez különösen akkor jelent nagy előnyt, ha több ezer csomópontból álló teleppel dolgozunk, amelyben egyes csomópontok földrajzilag esetleg a világ másik végén is lehetnek. Gondoljunk például a hálószámítási (grid computing) módszerekre. Ott nemhogy gyakori, egyenesen természetes ez a helyzet.
Most, hogy rendszerünket bebiztosítottuk a forgalomirányítási eltévelyedések ellen, itt az ideje, hogy újra kipróbáljuk azt az esetet, amikor a `PhoneMania` kérést továbbít a `RingBell` háttérkiszolgálója felé:

```
[colby demo]$ ./TelecomClient  -h 10.1.1.1
➥-p 8800
Requesting quotation for Ericsson
Quote Ericsson
...
[colby demo]$ ./PhoneManiaEP -h 10.1.1.1
➥-p 8800 -b 10.1.1.2 -r 9001

PHONEMANIA: bind/connect on 10.1.1.1:8800 = 0
PHONEMANIA: bind/connect on 10.1.1.2:9001 = 0
Quote Ericsson

Quotation request received
...
[munster demo]$ ./RingBellBE -h 10.1.1.2 -p 9001
RINGBELL: bind on 10.1.1.2:9001
...
```

A másik csomóponton (`munster`) észleljük, hogy a `RingBell` háttérkiszolgálója többé már nem kezeli a `PhoneMania` kéréseit, pedig a `PhoneMania` engedély nélkül továbbra is a `RingBell`-re irányítja azokat. A */var/log/messages* fájlban a nem engedélyezett kérések nyomon követésének céljából lehetőség van a ***DSI*** által előállított naplók rögzítésére:

```
May  6 07:47:31 munster kernel: DSI-LSM MODULE -
➥dsi_sock_rcv_skb check permission sscid 10
➥ssnid 1 tscid 20
May  6 07:47:31 munster kernel: DSI-LSM MODULE
➥Error - dsi_sock_rcv_skb - No Permission
```

Összegzés

Egy gyakorlati megoldást mutattunk be a fürtök különböző felhasználók által használt alkalmazások közti biztonságos megosztására. A ***DSI*** projekt lehetővé teszi a felhasználók számára, hogy egyszerűen hozzanak létre elkülönülő biztonsági zónákat a telepen futó alkalmazások számára. Miután a ***DSI***-t feltelepítettük a telepre, az új alkalmazások számára létrehozandó új biztonsági zónák kialakítása a programokhoz tartozó megfelelő ***ScID*** azonosítók beállítására és a kapcsolódó szabályok ***DSP*** fájlban történő rögzítésére egyszerűsödik. A forráskód módosítására nincs szükség és valószínűleg lehetőség sincs rá.

Linux Journal 2004. október, 126. szám

Makan Pourzandi (makan.pourzandi@ericsson.ca) az Ericsson kanadai kutatóközpontjának nyílt rendszerekkel foglalkozó osztályán dolgozik. A kutatási területei közé a biztonság, a fürtözött számítások és az osztott programozás komponens alapú módszerei tartoznak. Doktori fokozatát 1995-ben Franciaországban, a lioni egyetemen szerezte meg a párhuzamos számítások tárgykörében írt dolgozatával.

Axelle Apvrille (axelle.apvrille@ericsson.ca) jelenleg az Ericsson kanadai kutatóközpontjának nyílt rendszerekkel foglalkozó osztályán dolgozik. Kutatási területei a kriptográfia, biztonsági protokollok és az osztott biztonság. Számítástechnikából 1996-ban szerzett diplomát a franciaországi Bordeaux-ban, az ENSEIRB egyetemen.

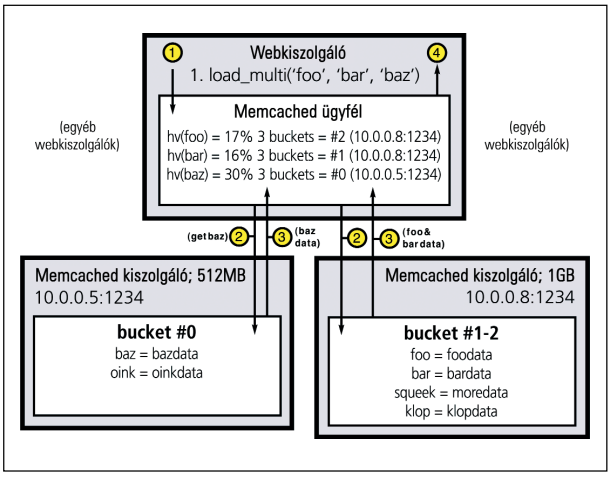
Elosztott gyorstárazás a Memcached használatával

Webkiszolgálónk adatbázisának terhelését jelentősen csökkenthetjük, ha alkalmazásainkat egy méretezhető, objektumokat gyorstárazó réteggel bővítjük.

A ***Memcached*** nagy teljesítményű, elosztott gyorstárrendszer. Bár a konkrét alkalmazásoktól tökéletesen független, leginkább a webes alkalmazások gyorsítására használják úgy, hogy segítségével az adatbázisok terhelését csökkentik. A ***Memcachedet*** többek közt a ***LiveJournal***, a ***Slashdot*** és a ***Wikipedia*** is használja.

Miért gyorstárazunk?

Az elmúlt nyolc év során nagy méretű, interaktív, adatbázisra épülő, több kiszolgálóra is kiterjedő weboldalak készítésével foglalkoztam. A ➲ ***LiveJournal.com*** blogrendszeréhez és nyilvános hálózati szolgáltatásaihoz közel 70 gép biztosítja a hátteret, összesen 2,5 millió fiókot tartva fenn. A szokásos blogolási lehetőségeken és a barátok/érdeklődés/személyes adatok megadásának lehetőségén túl a ***LiveJournal*** oldalán fórumokat, kérdőíveket, a felhasználók egyedi igényeire szabott híroldalakat, telefonokról áttöltött hangfelvételeket és egyéb változatos, az embereket egymáshoz közelebb hozó dolgokat és szolgáltatásokat lehet találni. A dinamikus weboldalak előállítási sebességének növelése mindig nehéz feladat, és ez alól a ***LiveJournal*** sem volt kivétel. A dolgot tovább nehezítette, hogy a rendszernek szinte minden tartalmi eleméhez tartozhat valamilyen biztonsági szint, illetve az elemek számos különböző nézetbe csoportosíthatók. Mivel korábban már foglalkoztam dinamikus, környezetfüggő tartalmakkal, tudtam, hogy statikus oldalak előzetes előállításával nem sokra jutunk. Az oldalakat alkotó különféle objektumok gyorstárazási jellemzői és élettartamuk annyira eltérő lehet, hogy az oldalakat többször kellene előállítani, mint ahányszor lekérik őket. Így ezzel a megoldással csak az erőforrásokat pazaroltuk volna.
Persze a gyorstárazás ettől még nem rossz dolog. A másik oldalról viszont egy számítógép sebességét jelentős mértékben befolyásolja memóriarendszerének sebessége, mérete és szintjeinek száma. A gyorstárazás valóban hasznos, de csak akkor, ha megfelelő tárolóeszközzel és felbontással tudjuk végezni. Én úgy gondolom, a legjobb az, ha az egyes oldalak objektumait külön gyorstárazzuk, és nem magukat a weboldalakat tároljuk. Ilyenkor nem növeljük feleslegesen a szükséges tárterület méretét azzal, hogy az objektumokat és a több oldalon is feltűnő sablonelemeket minden oldalhoz külön tároljuk. A végső döntés persze mindig különféle szempontok között egyensúlyozva születik meg. A processzorok egyre gyorsabbak, tehát szerintem inkább a processzor erőforrásait érdemes használni, és nem a lemezeket. Igaz ugyan, hogy az újabb lemezek egyre nagyobbak és olcsóbbak, a sebességük azonban korántsem nő ilyen mértékben. Figyelembe véve, hogy mennyire hajlamosak a meghibásodásra, és mennyire tohonyák, az a legjobb, ha nem is használjuk őket, ha nem muszáj.
A ***LiveJournal*** webes csomópontjaiban egyáltalán nincsenek merevlemezek. A rendszerindítást közös, redundáns módon tárolt NFS képről hajtják végre. Ez a megoldás nemcsak olcsóbb, de kevesebb karbantartást is igényel.
Az adatbázis-kiszolgálóból természetesen nem spórolhatjuk ki a lemezeket, itt viszont gyors, RAID-be kötött meghajtókat használunk. Jelenleg tíz különböző adatbázisfürtöt tartunk fenn, melyek mindegyike kettő vagy több gépből áll. A tíz fürtből kilenc felhasználói, vagyis ezek kizárólag a felhasználók adatait tárolják. A tizedik egy közös fürt, amely a nem a felhasználókkal kapcsolatos adatokat hordozza. Itt találhatók a felhasználó-fürt hozzárendelések is. A fürtöket az írási műveletek elosztása végett függetlenítettük egymástól. Megtehettük volna azt is, hogy egyetlen nagy méretű fürtöt állítunk össze, több száz szolgagéppel, ez azonban csak az olvasásokat tudta volna szétosztani. A válaszidő



1. ábra A Memcached ügyfélkönyvtár feladata a kérések megfelelő kiszolgálókhoz való továbbítása

minden újabb szolga hozzáadásával növekedett volna, a kiírt adatok elkerülhetetlen frissítése miatt egyre rosszabb teljesítményt kaptunk volna.
Az elmondottakból talán már látszik is, hogy a ***LiveJournal*** háttérrendszerét milyen elgondolás szerint építettük meg:

1. Minél kevesebb lemezt használunk, mert csak a baj van velük. Ha elkerülhetetlen a használatuk, akkor csak gyors és redundáns be- és kiviteli alrendszert veszünk igénybe.
2. Szélességében méretezzük a rendszert és nem felfelé, vagyis inkább több kis gépet üzemeltetünk, mint kevesebb nagyot.

A kis gép persze nem az olcsó, inkább a sokáig használható gépet jelenti. Én szeretek addig megtartani egy gépet, ameddig az általa elfoglalt hely és a megtermelt hő szempontjából érdemes. Méretezésnek nevezik azt is, amikor félévente kidobják a gépet, és vesznek egy még nagyobbat, de ez a megoldás nekem nem nagyon tetszik.

Hol gyorstárazzunk?

A Memcached bevezetése előtt a webes csomópontok kedvük és igényük szerint érhették el az adatbázisokat. A rendszer működött, de nem volt az igazi. Hamar rájöttem, hogy hiába van 4 vagy 8 GB memória az adatbázis-kiszolgálóban, gyorstárai korlátozottak, mind magának a memóriának a méretét, mind a 32 bites gépeken futó folyamatok által megcímezhető terület nagyságát tekintve. Persze vehettem volna 64 bites gépeket, amelyek több memóriát tudnak megcímezni, de mit tegyek, makacs és fösvény vagyok. Úgy döntöttem tehát, hogy a webes csomópontokon kell többet gyorstárazni. Mivel ***mod_perl 1.x***-et használunk, a gyorstárazás egyszerűen kínszenvedés. Minden folyamat, vagyis minden webes kérés saját címtérrel rendelkezik, a többivel nem tud adatokat megosztani. Meg lehetne csinálni, hogy a 30-50 folyamat mindegyike saját gyorstárat tartson fenn, de megint csak pazarolnánk az erőforrásokat. A System V megosztott memória megoldása ma már meglehetősen szokatlan korlátokkal küzd, nem hordozható, és csak egyetlen gépen működik. Több mint 40 csomóponton egyszerűen nem is használható. Azt hiszem, érzékelhető, hogy mennyire nem voltam elégedett a már meglévő gyorstárazó megoldásokkal. Még ha alkalmazásrendszerünk többszálú is lett volna, és az adatokat elvileg könnyen meg lehetett volna osztani a folyamatok között, ez akkor is csak egy-egy gépre terjedt volna ki. El akartuk kerülni, hogy a több mint 40 gép önállóan gyorstárazzon, és ki tudja, hány példányt tartson fenn az adatokból.

A Memcached születése

Egy napon, amikor már kellően tele volt a puttonyom a ***mod_perl*** alkalmazások gyenge gyorstárazásával, álmodozni kezdtem. Rájöttem, hogy egy csomó szabad memória található elszórva a hálózaton, és valahogy ki kellene használni. Aki ***Perl*** programozóként turkál egy kicsit a ***CPAN***-on, ***Cache::**** modulok ijesztő sokaságát találja. Legtöbbjük felülete valamilyen szótár. Akinek ez a kifejezés nem mond túl sokat, annak elárulom, hogy a szótár olyan elvont adattípus, amely kulcsokat és értékeket rendel össze. A Perlben járatos emberek az ilyesmit asszociatív tömbnek vagy kivonattáblának, esetleg röviden kivonatnak nevezik. A kivonattábla különleges típusú adatszerkezet, amely egy szótárfelületet bocsát rendelkezésünkre.
Olyan átfogó kivonattáblát akartam létrehozni, amelyet az összes gép összes webes folyamata akár egyszerre is el tud érni, és azonnal látja a többiek módosításait. Ez lett volna a gyorstáram. Mivel a memória drága, a hálózatok gyorsak, a kiszolgálók üzembiztossága pedig kérdéses, a gyorstárat az összes gépre el akartam osztani. Gyors keresést végeztem, és mivel ilyet nem találtam, nekiláttam megvalósítani az elképzelésemet.
A ***Memcached*** kiszolgáló példányok mindegyike egy felhasználó által megadott IP-n és kapun várja a kéréseket. Az alapötlet az, hogy a hálózaton mindenfelé, ahol csak szabad memóriát találunk, ***Memcached*** példányokat futtatunk, és ezek szolgálják ki az alkalmazásokat. Egy-egy gépen akár több példányt is lehet futtatni, ha az adott gép 32 bites, és több memóriával rendelkezik, mint amit a rendszermag egy-egy folyamat számára biztosítani tud. Például, mielőtt még rájöttünk volna, hogy inkább szélességében, mint felfelé érdemes méretezni a rendszert, vettünk néhány elképesztően drága, egyenként 12 GB memóriával felszerelt gépet. Mostanában mindenféle feladatokra használjuk őket, többek közt öt-öt 2 GB-os ***Memcached*** példányt futtatnak. Így egyetlen gép is 10 GB memóriával képes hozzájárulni az átfogó gyorstárhoz. Mindezt annak ellenére, hogy 32 bites Linux alatt egy-egy folyamat általában csak 3 GB memóriát tud megcímezni.
A ***Memcached*** trükkje az, hogy adott kulcs kezelésére mindig ugyanazt a csomópontot kell felkérni, miközben a tároló kulcsok egyenletesen oszlanak el a csomópontok között. Arra nincs lehetőség, hogy a „példa” kulcsot az 1-es gépen tároljuk, majd egy másik folyamat később a 2-es gépről próbálja betölteni. Persze ez nem is okoz különösebb gondot. A ***Memcached*** csomópontokat úgy érdemes kezelni, mint a kivonattáblák vödreit.

A Memcached működése

1. lépés: Az alkalmazás a példa1, a példa2 és a példa3 iránt bocsát ki kérést az ügyfélkönyvtáron keresztül, ez kiszámítja a kivonatok értékét, amiből ki tudja választani, hogy melyik ***Memcached*** kiszolgálóhoz kell fordulnia.
2. lépés: A ***Memcached*** ügyfél párhuzamos kéréseket küld a megfelelő ***Memcached-*** kiszolgálóknak.
3. lépés: ***Memcached*** kiszolgálók válaszolnak az ügyfélkönyvtárnak.
4. lépés: A ***Memcached*** ügyfélkönyvtár összegyűjti az alkalmazásnak szóló válaszokat.

Aki ismeri a kivonattáblák működését, nyugodtan ugorjon. Aki nem, annak azonnal adok egy gyors áttekintést. A kivonattábla vödrök tömbje. Minden vödör (azaz tömbelem) csomópontok egy listáját tartalmazza, minden csomópont egy kulcs-érték párosból áll. A megfelelő kulcsot tartalmazó csomópont után szükség esetén ebben a listában végzünk keresést. A legtöbb kivonattábla először kisméretű, majd idővel dinamikusan változik a mérete, ahogy a vödrökben tárolt listák túl hosszúakra nyúlnak.
Ha adott kulcshoz tartozó értékre vonatkozóan olvasási vagy írási kérést indítunk, akkor a kulcs alapján először egy

kivonatot kell számolni. A kivonatoló függvények egyirányú leképező függvények, amelyek egy kulcshoz (ez szám vagy karakterlánc egyaránt lehet) valamilyen számot rendelnek – ez lesz a vödör száma. A vödör számának meghatározása után keresést kell végezni a vödör listájában az adott kulcsú csomópont után. Ha nincs ilyen, akkor új csomóponttal kell bővíteni a listát.

Mi köze van mindennek a ***Memcached*** működéséhez? A ***Memcached*** a felhasználók számára egy szótárfelületet (kulcs -> érték) biztosít, de belül két rétegű kivonatolást végez. Az első réteg az ügyfélkönyvtárban található. A kulcsot képzetes vödrök listájába kivonatolva ez dönti el, hogy melyik ***Memcached*** kiszolgálónak kell elküldeni a kérést. Amikor a kérés eljutott hozzá, a kiválasztott ***Memcached*** kiszolgáló normál kivonattáblát alkalmaz.

Minden ***Memcached*** példány tökéletesen független, a többivel semmilyen adatcserét nem folytat. Mindegyik példány a legrégebben használt elemeket dobja el, amikor helyet teremt az újaknak. A kiszolgálók rengeteg statisztikát készítenek, amelyek alapján összesíthető a teljes ***Memcached*** telep lekérdezés/találat/hiba mutatója. Ha valamelyik kiszolgáló meghibásodna, az ügyfelek képesek a halott gép vagy gépek megkerülésére, a továbbra is aktív kiszolgálókat pedig tovább használják. Ez a szolgáltatás elhagyható, ugyanis használatához az alkalmazásokat is fel kell készíteni arra, hogy esetleg elavult adatokat kapnak valamelyik betegeskedő csomóponttól. Ha a szolgáltatást kikapcsoljuk, akkor a halott kiszolgálókhoz intézett kérések egyszerűen egy gyorstárhibát eredményeznek az alkalmazás oldalán. Ha kellően nagy, elegendő egyedi állomásból álló ***Memcached*** telepet tartunk fenn, akkor egy-egy gép leállása nem fogja számottevő mértékben rontani az átfogó találati arányt.

Saját rendszerünk felépítése

A ***LiveJournal.com*** jelenleg 28 ***Memcached*** példányt futtat összesen tíz gépen, ezek a leggyakrabban igényelt 30 GB-nyi adatot gyorstárazzák. Találati arányunk 92 % körül mozog, vagyis az alkalmazásoknak a korábbinál jóval kevesebbszer kell az adatbázisokhoz fordulniuk.

Webes csomópontjaink mindegyike 4 GB memóriával rendelkezik, ezeken három-három 1 GB-os ***Memcached*** példányt futtatunk, a mod_perl 500 MB-tal gazdálkodhat, 500 MB pedig tartalékként szolgál. A ***Memcached*** és a ***mod_perl*** azonos gépen való futtatása jó megoldás, a ***mod_perl*** kódunk ugyanis erősen terheli a processzort, a ***Memcached*** viszont csak alig. Természetesen megtehetnénk, hogy a ***Memcached*** futtatására külön gépeket vásárlunk, de gazdaságosabb megoldás a ***Memcached*** példányokat oda szétszórni, ahol amúgy is rendelkezünk felhasználható memóriával, esetleg memóriabővítést végezni egy régebbi gépen, ami így meg fog felelni a feladatra.

Memcached telepet különféle méretű gyorstárakkal is lehet futtatni, mi például 512 MB, 1 GB és 2 GB méretű példányokat keverünk. A példányokat és méretüket az ügyfélbeállítások között kell megadni, a ***Memcached*** kapcsolatobjektumok ennek megfelelően súlyoznak.

Sebesség

A gyorstárazás elsődleges célja nyilván a folyamatok gyorsítása, tehát a ***Memcached*** rendszert a lehető leggyorsabbra készítettük. A program első, kísérleti példányát Perlben írtuk. Szeretem a Perlt, de a kísérleti kód nevetségesen lassú és nagyméretű volt. A ***Perl*** meglehetősen sok memóriát használ, és nagyobb számú hálózati kapcsolatot sem képes egyszerre kezelni.

A jelenlegi változat C nyelven készült, egy folyamatból álló, egyszálú, aszinkron be- és kivitelt végző, esemény alapú démon. A hordozhatóság és a megfelelő sebesség érdekében az eseményekről való értesítéseket a ***libevent*** segítségével oldottuk meg. A ***libevent*** használata azért előnyös, mert futási időben a lehető legjobb módszert választja a fájlleírók kezelésére. BSD alatt például a ***kqueue***, Linux 2.6 alatt pedig az ***epoll*** szolgáltatásaira épül, amelyek több ezer párhuzamos kapcsolat hatékony kezelésére is megfelelők. Más rendszereken a ***libevent*** a hagyományos ***poll*** és ***select*** eljárásokat használja.

A ***Memcached*** belsejében minden algoritmus ***O(1)*** futási idejű. Az algoritmusok végrehajtási ideje és processzorigénye tehát az egyszerre csatlakozó ügyfelek számától – legalábbis a ***kqueue*** vagy az ***epoll*** használatakor – független, ahogy az adatok méretétől és az egyéb tényezőktől is.

Megjegyezném, a ***Memcached*** tömbfoglalót (***slab allocator***) használ a memóriafoglalásokhoz. A ***Memcached*** korai változatai a ***glibc***-féle ***mallocot*** használták, ám ez nagyjából egy heti működés után fura dolgokat kezdtek produkálni, és a címtér széttöredezése miatt csak a processzor erőforrásait fogyasztotta. A tömbfoglalók csak nagy méretű memóriaterületeket foglalnak le, ezeket a különféle elemosztályok számára kisebb részekre osztják, majd egy részek foglaltságát nyilvántartó lista vezetésével figyelik, hogy az egyes osztályok objektumai mikor szabadulnak fel. Erről a témáról további részleteket a források között szereplő Bonwick-féle leírásban lehet találni. A ***Memcached*** pillanatnyilag 64 bájt és 1 MB között képes területet foglalni úgy, hogy a memória nagysága kettő hatványa legyen. Minden objektumot a befogadására alkalmas legkisebb egységben helyez el. A tömbfoglaló használatának köszönhetően a megfelelő teljesítményt tetszőleges időtartamra biztosítani tudjuk. Az éles ***Memcached*** kiszolgálóink például 4-5 hónapon keresztül üzemeltek, átlagosan 7000 lekérdezést kezeltek másodpercenként, de semmilyen gond nem volt velük, és processzorterhelésük is alacsony maradt.

A ***Memcacheddel*** szemben további elvárás volt, hogy képes legyen zárolások nélkül működni. Minden objektummal kapcsolatban belső változatkövetést és hivatkozásszámlálást végez, így egyik ügyfél sem akadályozhatja egy másik műveleteit. Előfordulhat például, hogy egy a csomagok felét eldobáló rossz hálózati kapcsolattal rendelkező ügyfél éppen frissít egy olyan a objektumot, amit közben több tucat másik ügyfél tölt le. Még ekkor sem kell senkinek sem bárki másra várakoznia.

A működésről még annyit megjegyeznék, hogy a protokoll egyszerre több kulcs lekérdezését is lehetővé teszi, ami főként akkor hasznos, ha egy alkalmazás előre tudja, hogy több száz kulcsot is elő kell keresnie. Ilyenkor nem soros lekérdezést indít, ami a hálózati válaszidők miatt viszonylag sokáig tartana, hanem egyetlen kéréssel elintézi az összeset. Ha szükséges, az ügyfélkönyvtárak önműködően több, a ***Memcached*** példányokhoz egymással párhuzamosan indított többkulcsos lekérdezésre osztják az alkalmazásoktól ér-

kező többkulcsos kéréseket. Arra is van lehetőség, hogy az alkalmazások előre meghatározott kivonatértékeket csatoljanak az adatokhoz, így egy-egy adatcsoportot azonos példányon tudnak tartani. Ezzel a megoldással az ügyfélkönyvtár némi processzoridőt is megtakaríthat, hiszen a kivonatokat nem kell kezelnie.

Az ügyfélkönyvtárak

A ***Memcached*** ügyfél/kiszolgáló felülete egyszerű, könnyen áttekinthető. Ügyfélkönyvtár Perl, PHP, Python és Java alá egyaránt létezik. Úgy hallom, egyik munkatársam egy Ruby ügyfélen is dolgozik, ami hamarosan meg fog jelenni.
Az ügyfelek mindegyike támogatja az objektumok sorrendezését, erre saját, natív sorrendező eljárásukat használják: a ***Perl*** a ***Storable***, a ***PHP*** a ***serialize***, a ***Python*** a ***Pickle***, a ***Java*** pedig a ***Serializable*** felületet veszi igénybe. A legtöbb ügyfél képes az átlátszó tömörítés támogatására, amelyet beállítástól függően csak adott adatméret felett használ. A sorrendezés és a tömörítés használata azért vált lehetségessé, mert a ***Memcached*** megengedi az ügyfeleknek, hogy átlátszatlan jelzőket – ezek a bejövő adatok kezelésének módját határozzák meg – tároljanak az egyes tételek mellett.

A Memcached használata

A ***Memcached*** telepítése önmagában nem bonyolult, használata viszont már komolyabb körültekintést igényel. Azt, hogy a legtöbb idő milyen műveletekkel telik el, csak saját alkalmazásaink vizsgálatával tudjuk eldönteni – ezeken a helyeken kell aztán gyorstárazni. A gyorstár frissítéséről és tisztán tartásáról szintén gondoskodnunk kell, a gyorstár tartalmának egységessége ugyanis a legtöbb alkalmazás számára fontos. Ha alkalmazásunk belső API-ja már letisztult, és az adatbázist nem ide-oda kapkodva érjük el, akkor a ***Memcached*** támogatásának hozzáadásával nem lesz gond. A lekérdező függvényekben egyszerűen csak ***Memcached***-hez kell elsőként fordulni. Hiba esetén lépjünk tovább az adatbázishoz, majd töltsük fel a ***Memcached*** tárait is. Az adatírásokat végző függvényekben az adatbázis és a ***Memcached*** frissítését egyaránt végezzük el. Minden bizonnyal lesznek olyan versenyhelyzetek és adategységességi gondok, amelyek megoldásra várnak, ám a ***Memcached*** API-ja biztosítja a kezelésükhöz szükséges eszközöket.
A ***Memcached*** olyan adatok tárolására is használható, amelyeket lemezre nem akarunk kiírni. A ***LiveJournal*** például úgy előzi meg az adatküldések véletlen kettőződését, hogy a tranzakciók kódjait a ***Memcached*** segítségével tárolja, kulcsként a tranzakciók aláírását használva. A ***Memcached*** elsődleges adattárként való használatára egy másik példa az egyszerű és/vagy rosszindulatú robotok és szemétküldők kihajigálása. Ha figyelemmel követjük, hogy az egyes IP-címekről származó kapcsolatok révén milyen műveleteket és mikor indítottak, akkor bizonyos minták alapján már korán fel tudjuk ismerni a támadásokat, és meg tudjuk kezdeni a szükséges intézkedéseket. Ezeket az adatokat felesleges volna adatbázisban tárolni, csak a lemezmeghajtókat terhelnénk velük. Bár a memóriában való tárolás elegáns megoldás, ha valamelyik ***Memcached*** csomópont meghibásodik, akkor az adatok is elvesznek.
A levelezőlistán megkérdeztem, hogy mások mire használják a ***Memcached***-et, és a következő válaszokat kaptam:

- Sokan alkalmazzák ugyanarra a célra, mint a ***LiveJournal***, vagyis kisméretű webes objektumok gyorstárazására.
- Az egyik oldalon az éppen lejátszás alatt lévő zeneszámot adják át vele Java alapú adatfolyam-kiszolgálójuktól PHP alapú webhelyüknek. Erre a célra korábban egy adatbázist használtak, de a ***Memcached*** szerintük jobban működik.
- Sokan hitelesítési adatokat és munkamenetkulcsokat gyorstáraznak vele.
- Az egyik helyen az ismert jó és rossz állomások és hitelesítési adataik gyorstárazásával a levélkiszolgálók működését gyorsítják.

Mindig újabb és újabb érdekes elektronikus leveleket és javaslatokat kapok – örömmel látom, hogy sokaknak tudtunk segíteni.

Egyéb megoldások

Ha valaki egy többszálú alkalmazást egyetlen gépen futtatva meg tudja oldani feladatait, illetve nincs szüksége átfogó gyorstárra, annak nincs mit kezdeni a ***Memcached***-del. Aki csak egy gépet használ, annak a ***SysV*** megosztott memória szolgáltatása is megfelelő lehet.
Néhányan felvetették, hogy a ***MySQL 4.x*** lekérdezés-gyorstárának alkalmazásával a ***Memcached*** kiiktatható. A ***MySQL*** lekérdezés-gyorstáblája minden alkalommal kiürítésre kerül, amikor valamelyik érintett tábla – bármilyen módon is – frissítésre kerül, ezért inkább olyan helyeken használható, ahol a tartalmat csak olvassák. A ***LiveJournal***-on rengeteg írás történik, ahogy a nagy forgalmú webhelyeken általában lenni szokott. Más adatbázisokhoz hasonlóan a MySQL gyorstárainak mérete összesen nem haladhatja meg a rendszermag által biztosított címteret, ami a 32 bites gépeken 3 GB – manapság ez már kevésnek számít.
Sokaknak megfelelő megoldás a ***MySQL*** memóriabeli táblakezelője is. Nekem azért nem tetszett, mert csak szabott méretű rekordokkal tud dolgozni, ***BLOB*** és ***TEXT*** oszlopok használatát nem engedi. Az általa tárolható adatok összesített mérete is korlátozott, vagyis belőle is nagy számú példányt kellett volna futtatni, és ugyanúgy gondoskodni kellett volna a kulcsok szétosztásáról.

Köszönetnyilvánítás

Szeretnék köszönetet mondani ***Anatoly Vorobey*** fejlesztőnek, ***Lisa Phillipsnek***, amiért hajlandó volt a korai, folyton összeomló változatokkal küzdeni, továbbá a levelezőlista tagjainak, akik rengeteg foltot, kérdést és javaslatot küldtek nekünk.

Linux Journal 2004. augusztus, 124. szám

Brad Fitzpatrick nyolc évig adatbázis alapú webhelyek építésével foglalkozott. Brad szeret kerékpározni, illetve különféle, egyébként csak pénzköltéssel megoldható gondokra alternatív megoldásokat kitalálni. Aki nem kék tablettákat forgalmaz, és nem halott kiszolgálókról akar értesítést küldeni, annak levelét örömmel fogadja a brad@danga.com címen.

A Linux fájlrendszer biztonsága (1. rész)

A jogosultságok beállításának hibája számos gyakori Linux probléma – nem egyszer idegesítő – forrása, ezért kell elsajátítani a jogosultságok alapjait, és megtenni az első lépéseket a Linux biztonságának megértése felé.

A ***Linux Journal Paranoid pingvin*** rovatában az elmúlt négy évben elsősorban eszköz-központú cikkek jelentek meg. Megtudtuk, hogyan kell biztonságossá tenni a ***Sendmailt***, hogyan lehet a ***Stunnel*** segítségével ***SSL*** titkosítást alkalmazni, valamint egy egész sereg hatékony biztonsági program beállításának és futtatásának módját. A következő néhány oldalt ezzel szemben a a fájlrendszer jogosultságainak szenteljük, amely egyfelől a Linux biztonságának egyik legalapvetőbb, legfontosabb tényezője, másrészt – és ez az igazán meglepő – viszonylag elhanyagolt téma. Ha bölcsen használjuk, a felhasználók és a behatolók nehezebben tudnak visszaélni a jogosultságokkal. Ha azonban nemtörődöm módon állítjuk be, a kisebb gyenge pontok nagymértékben veszélyeztethetik a rendszert. Ezek a cikkek különösen hasznosak lehetnek a Linux újoncok számára, akik nem értik, mit jelenthet az a sok `drwxr-xr-x` halandzsa fájllistában. De ezek a cikkek, különösen a 2. rész, még a középhaladók számára is – esetleg akik még nem értik a ***setuid*** és a ***setgid*** pontos következményei – tartogathat újdonságokat.

Bevezetés: Minden fájl

Nem sokan tudják, hogy a ***UNIX*** és a hozzá hasonló rendszerekben alapvetően mindent fájlok képviselnek. A dokumentumokat, a képeket és akár a futtatható programokat is könnyű a merevlemezen elhelyezkedő fájlokként felfogni. Ugyan a könyvtárakat fájltárolóként képzeljük, pedig valójában olyan fájl, amely – ahogy sejtettük – más fájlokat tárol. Hasonlóképpen, annak ellenére, hogy a rendszerhez kapcsolt CD-ROM elég kézzelfoghatónak tűnik, de a Linux rendszermag számár az is csak egy fájl: egy különleges eszközfájl, a */dev/cdrom*. A rendszermag tulajdonképpen ebbe a különleges fájlba ír és ebből olvas ahhoz, hogy adatot küldjön róla, vagy hogy a CD-ROM meghajtóra írja. Valójában a legtöbb rendszerben a */dev/cdrom* jelképes hivatkozás a */dev/hdb*-re, vagy valamelyik más különleges fájlra. Ki sem találnánk, hogy a jelképes hivatkozás viszont nem más, mint egy olyan fájl, amely egy másik fájl elérési útvonalát tartalmazza.
Más fájlok, mint például a nevesített csővezetékek, bemeneti/kimeneti csatornaként viselkednek, lehetővé téve, hogy a programok adatokat adjanak át egymásnak.

Nem az a lényeg, hogy leírjunk minden egyes fájltípust, ami a ***Linuxban*** vagy a ***UNIX***-ban létezik, hanem, hogy szemléltessük, hogy szinte mindent fájlok képviselnek. Amint ezt felfogjuk, sokkal könnyebb lesz megérteni, hogy miért olyan fontos a fájlrendszer biztonsága, és hogyan működik.

Felhasználók, csoportok, jogosultságok

A ***Linuxban*** tulajdonképpen két olyan dolog van, amit nem fájlok képviselnek: a felhasználói fiókok és a csoportfiókok. Többféle fájl is tartalmaz adatokat a rendszer felhasználóiról és csoportjairól, de valójában egyik sem képviseli azokat.
A felhasználói fiók olyan valakit vagy valamit jelent, aki/ami képes a fájlok használatára. Ez azt jelenti, hogy emberek is és rendszerfolyamatok is használhatnak felhasználói fiókokat. Ha egy felhasználói fiók neve például webmester, az általában egy olyan embert jelent, aki webhelyeket tart fent, de szabványos ***lp*** Linux felhasználói fiókot a soros nyomtató démon (***lpd***) használja – az ***lpd*** program ***lp*** felhasználónéven fut. Később elmagyarázzuk, hogy mit jelent a programok számára, ha egymás elleni felhasználóként futnak.
A csoportfiók egyszerűen felhasználói fiók egy listája. Minden egyes felhasználói fiók egy fő csoporttagsággal van meghatározva, de valójában annyi csoporthoz tartozhat, amennyihez szükséges. A ***maestro*** (mester) felhasználónak például lehet a ***conductors*** (karmesterek) csoportban a fő tagsága, és tartozhat a ***pianists***-hoz (zongoristák) is.
A felhasználók fő csoporttagsága az */etc/password* fájlban található fiókbejegyzésben van meghatározva. A felhasználót további csoportokba is sorolhatjuk, az */etc/group* szerkesztésével, a felhasználónevet minden olyan csoportbejegyzés végéhez írva, amelyben a felhasználónak szerepelnie kell. Másik lehetőségként használhatjuk a `usermod` parancsot – további információkért lásd a `usermod(8)` súgóoldalt.
Az alábbi lista első sora a */etc/password* fájl `maestro`-ra vonatkozó bejegyzését mutatja, a következő kettő pedig a */etc/group* fájl megfelelő tartalmát.

```
maestro:x:200:100:Maestro Edward
Hizzersands:/home/maestro:/bin/bash

conductors:x:100:
pianists:x:102:maestro,volodyia
```

A *passwd* fájlban az első mező a felhasználói fiók nevét, a *maestro* nevet tartalmazza. A második (x) *maestro* jelszavának mutatója, ami arra utal, hogy a jelszó valójában az */etc/shadow* fájlban tárolódik. A harmadik mező a `maestro` számazonosítója (*uid*-ja), amely ebben az esetben 200. A negyedikben a felhasználó elsődleges csoportjának számazonosítója (*gid*) látható, amely ezúttal 100. A többi mező egy megjegyzést, `maestro` saját könyvtárát és alapértelmezett bejelentkezési héját tartalmazza.
A */etc/group* fájlban az egyes sorok csupán a csoportnevet, a csoportjelszót (általában nem használatos – az `x` a mutató), a csoport számazonosítóját (*gid*) és vesszővel elválasztva, a csoportba másodrendű tagként felvett felhasználók listáját tartalmazzák. Így láthatjuk, hogy a `conductors` csoport *gid*-je 100, ami megfelel `maestro` főcsoportként meghatározott *gid*-jének. Azt is látjuk, hogy a `pianist` csoportnak tagja a maestro felhasználó, és másodrendű tagként egy `volodyia` nevű felhasználó.
Az */etc/password* és az */etc/group* fájl felhasználói fiókok létrehozása, módosítása és törlése céljából való megváltoztatásának legegyszerűbb módja a `useradd`, `usermod` illetve `userdel` parancsok használata. A parancs írásmód helyett most inkább az elvre összpontosítunk, tehát elég annyit mondani, hogy mind a három parancs a csoporttagságok beállítására és módosítására szolgál, és a megfelelő súgóoldalakon jól dokumentáltak. Ahhoz, hogy egy rövid összefoglalót lássunk a használatról, gépeljük be a parancs után a `--help` paramétert, például `useradd --help`.

Egyszerű fájl-jogosultságok

Minden fájlnak két tulajdonosa van: egy felhasználó és egy csoport, mindegyik saját jogosultság-beállításokkal rendelkezik, amelyek meghatározzák, hogy a felhasználó mit csinálhat az adott fájllal – olvashatja, írhat bele és végrehajthatja. Egy harmadik beállításkészlet arra vonatkozik, hogy mások – olyan felhasználói fiókok, amelyek nem a fájl tulajdonosai, illetve nem tartoznak olyan csoporthoz, amely a fájl tulajdonosa – mit csinálhatnak a fájllal. Az *1. listán* a */home/maestro/baton_dealers.txt* hosszú fájllistája látható.
A jogosultságok a felhasználói engedélyek, a csoportengedélyek és a többi engedély sorrendje alapján vannak felsorolva. Az 1. listában bemutatott fájl esetében a tulajdonosa (maestro) olvashatja és írhatja a fájlt (`rw-`), a tulajdonos csoport (`conductors`) szintén olvashatja és írhatja a fájlt (`rw-`), de a többi felhasználó csak olvashatja. A jogosultságok azért kicsit bonyolultabbak. Az egyéb kategóriába sorolt felhasználók egy adott fájlra vonatkozó jogosultságok alapján bármilyen fájlt törölhetnek egy olyan könyvtárban, amelyre van írási jogosultságuk. Más szóval, ha egy felhasználónak csak olvasási jogosultsága van egy fájlra, akkor nem szerkesztheti, de ha a fájl könyvtárára írási jogosultsága van, akkor törölheti a fájlt.
Az olvasáson és az íráson kívül van egy harmadik jogosultság: a végrehajtás, amelyet `x` jelöl, amikor be van állítva. Ha a `maestro` ír egy héjprogramot, *punish_bassonists.sh* néven, és a jogosultságait a `-rwxrw-r--` értékre állítja, akkor úgy hajthatja végre ezt a héjprogramot, hogy beírja a nevét a parancssorba. Ha viszont elfelejti beállítani a végrehajtási engedélyt, nem tudja majd futtatni a programot, annak ellenére, hogy ő a tulajdonosa.

Parancsok és súgóoldalak

Ebben a cikkben a pontos írásmód és az adott parancsok használata helyett a fájlrendszer alapelveire összpontosítunk. De a kezdők biztosan szeretnék tudni, hogy egyáltalán hogyan kell futtatni ezeket a parancsokat, és hol találhatunk segítséget az írásmódhoz és az alkalmazáshoz.
Először is, minden példában és példahelyzetben a terminálablakban dolgozunk. A Microsoft *Windows* felhasználók úgy képzelhetik el a terminálablakot, mint egy *DOS* készenléti jelet, vagy parancsablakot. A terminálablak biztosítja a Linuxszal való párbeszéd legközvetlenebb formáját, vagyis minden parancsot kézzel írhatunk be, ahelyett, hogy egérkattintásokkal indítanánk el.
Ahhoz, hogy saját héjmunkamenetet indítsunk a *GNOME*-ból, kattintsunk a *Main Menu* (*Főmenü*) gombra, és válasszuk a *System Tools®Ternóminal* lehetőséget. A *KDE*-ben a terminálparancsot *konsole*-nak hívják, és saját ikonja van a tálcán: egy kagylóhéj egy számítógép-monitor előtt. Másik lehetőségként elindíthatjuk a *Run Program* (*Program futtatása*) párbeszédablakot, és a készenléti jelnél begépelhetjük, hogy *konsole*. Ahhoz, hogy gyakorlatilag bármelyik *Linux* parancshoz gyors segítséget kapjunk egy terminál/héj munkameneten belül, begépelhetjük az adott parancsot, utána pedig a `--help` lehetőséget. Ha például nem emlékszünk az `ls` parancs összes parancssor lehetőségére, amely a fájlokat és a könyvtárakat listázza, csak beírjuk az `ls --help` parancsot.
A `--help` lehetőség gyors, de nem mindegyik parancsnál működik. Amikor működik is, az eredmény néha túlságosan tömör. A parancsokra vonatkozó segítségkérés legjobb módja a `man` parancs használata. A súgóoldalak (man pages) teljes útbaigazítást biztosítanak a legtöbb Linux parancs használatával kapcsolatban, és gyakorlatilag az összes UNIX-hoz hasonló rendszerben jelen vannak. Ahhoz, hogy megtekintsük például az ls parancs súgóoldalát, gépeljük be a `man ls` parancsot. A súgóoldal felsorolásban a szóköz billentyű segítségével léphetünk tovább egy oldalt, a B gombbal pedig egy oldalt visszamehetünk, ha pedig begépeljük, hogy `/valami`, akkor megkeresi súgóoldalon a „`valami`" karaktersort.
Mi van azonban olyankor, ha nem tudjuk annak a parancsnak a nevét, amelyre szükségünk van? Erre való az `apropos` parancs. Gépeljük be például, hogy `apropos list`, hogy olyan különféle parancsokat láthassunk, amelyek valamit listáznak, majd annak a parancsnak a súgóoldalára ugorjunk, amelyre valószínűleg szükségünk van.

A jogosultságokat általában a `chmod` paranccsal lehet beállítani, ami a *„change mode"* (üzemmód-változtatás) rövidítése. A példát folytatva, tegyük fel, hogy a `maestro` meggondolja magát, és nem akarja, hogy a `conductors` (karmesterek) csoportja láthassa a pálcakereskedők (`baton dealers`) listáját. A *2. listán* látható parancsok segítségével eltávolíthatja a csoport olvasási/írási jogát.
A 2. lista példa `chmod` parancsában (`chmod go-rw`) a `go` arra utasítja a `chmod`-ot, hogy változtassa meg a csoport jogosultságait, és az egyéb jogosultságokat, az `-rw` pedig megszabja,

1. lista Jogosultságokat bemutató fájllisták

```
-rw-rw-r--  1  maestro  conductors  35414
➥Mar 25 01:38 baton_dealers.txt
```

2. lista Egy fájl jogosultságainak megváltoztatása a chmod paranccsal

```
bash-$ ls -l baton_dealers.txt

-rw-rw-r--  1  maestro  conductors  35414
➥Mar 25 01:38 baton_dealers.txt

bash-$ chmod go-rw baton_dealers.txt
bash-$ ls -l baton_dealers.txt

-rw-------  1  maestro  conductors  35414
➥Mar 25 01:38 baton_dealers.txt
```

3. lista Egy csoport által olvasható könyvtár

```
bash-$ chmod g+rx extreme_casseroles
bash-$ ls -l extreme_casseroles

drwxr-x---  8  biff  drummers  288  Mar 25
➥01:38 extreme_casseroles
```

hogy el kell távolítani a szóban forgó két kategória (a csoport és az egyéb) jogosultságait. Tehát a `chmod` parancs három részből áll: egy jogosultságkategóriából, amely az `u`, a `g` és az `o` valamilyen kombinációja. Itt állhat egyszerűen az a betű is (all = valamennyi), illetve szerepelhetnek benne műveleti jelek: a `-` az eltávolításnak, a `+` a hozzáadásnak felel meg. Ezeket mindig a hozzáadandó vagy eltávolítandó jogosultságok listája (általában `r`, `w` vagy `x`) követi.

Könyvtárjogosultságok

Már tudjuk, hogyan kell megváltoztatni a hagyományos fájlok alapvető jogosultságait, de mi a helyzet a könyvtárakkal? A könyvtárjogosultságok kicsit másképp működnek, mint a hagyományos fájlok jogosultságai. Az olvasás és az írás hasonló – a könyvtáraknál ezek a jogok a könyvtártartalom listázásának illetve a fájlok könyvtáron belüli létrehozásának és törlésének felelnek meg.

A könyvtáraknál viszont a végrehajtás kicsit kevésbé logikus. Itt a végrehajtás annak felel meg, ha bármit megváltoztatunk a könyvtárban, vagy a munkakönyvtárat az adott könyvtárra változtatjuk. Azaz, ha a felhasználónak vagy egy csoportnak végrehajtási jogosultsága van egy bizonyos könyvtárra, akkor a felhasználó/csoport listázhatja az adott könyvtár tartalmát, olvashatja a könyvtár fájljait (feltéve, hogy az egyes fájlok saját jogosultságai beletartoznak), és a munkakönyvtárat az adott könyvtárra változtathatjuk a `cd` paranccsal. Ha egy felhasználónak vagy egy csoportnak nincs végrehajtási jogosultsága az adott könyvtárra, akkor semmit nem listázhat vagy olvashat a könyvtáron belül, az ott lévő elemeken beállított jogosultságoktól függetlenül. Ha egy könyvtárra nincs végrehajtási jogunk, de olvasási jogunk van, és megpróbálunk az `ls` paranccsal listázni, olyan hibaüzenetet kapunk, amely tulajdonképpen felsorolja a könyvtár tartalmát. De ez nem működik olyankor, amikor nem rendelkezünk sem olvasási, sem végrehajtási jogosultsággal a könyvtárra vonatkozóan.

Tegyük fel, hogy a példarendszernek van egy `biff` nevű felhasználója, aki a `drummers` (dobosok) csoportba tartozik. Feltételezzük továbbá, hogy az ő könyvtára tartalmaz egy `extreme_casseroles` nevű alkönyvtárat, amit szeretne megosztani az ütőstársaival. Az *3. listán* látható, hogy `biff` hogyan állítaná be a könyvtár jogosultságait.

Az 3. listánál csak ***biffnek*** van lehetősége fájlokat létrehozni, megváltoztatni és törölni az `extreme_cassaroles` könyvtárban. A `drummers` csoport többi tagja listázhatja a tartalmát, és a `cd` paranccsal beléphet. A rendszerben viszont mindenki más számára tiltott lesz a listázás, azs olvasás, a `cd` parancs használata vagy bármilyen más, a könyvtárral kapcsolatos tevékenység.

A jogosultságok és a rendszergazda

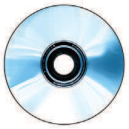
Gyakorlatias kifejezéssel élve, a jogosultságok egyszerűen nem vonatkoznak a ***root*** felhasználóra, a rendszergazda bármelyik fájllal bármit, bármikor csinálhat. Ezért olyan fontos, hogy soha ne rendszergazdaként jelentkezzünk be, és ne használjuk a `su` parancsot, hogy rendszergazdává váljunk, kivéve, ha feltétlenül szükséges. Rendszergazdaként a fájl-jogosultságok nem védenek meg a saját hibáinktól. Ez nem azt jelenti, hogy ha rendszergazdaként lépünk be, az összes program teljesen figyelmen kívül hagyja a fájl-jogosultságokat. Ha a rendszergazda például a `vim` szerkesztő segítségével megpróbál megváltoztatni egy írásvédett fájlt, akkor a `:w!` parancsot (írás kényszerítése) kell használnia. A hagyományos `ZZ` vagy `:w` parancs ilyenkor hibaüzenetet ad vissza. Nagyon sok más parancsnak viszont nincs olyan szolgáltatása, amely ellenőrizné az ésszerűséget.

Összefoglalás

Ezek a Linux fájlrendszer biztonságának legalapvetőbb elvei és gyakorlati felhasználási módjai. A 2. részben tovább megyünk, és megbeszéljük (többek között) a ***setuid***-et, a ***setgid***-et, és a numerikus jogosultsági üzemmódokat. Addig is, csak óvatosan!

Linux Journal 2004. október, 126. szám

Mick Bauer – Hitelesített informatikai rendszerek biztonsági szakértője, a Linux Journal biztonsági szerkesztője, és biztonsági szakember a Minneapolisban, Minnesotában. A Building Secure Servers With Linux (O'Reilly, 2002) című könyv szerzője.



A WordPress

Akiknek olyan „mindentudó” weblog alkalmazásra van szükségük, amelynek beállítása a lehető legegyszerűbben oldható meg a parancssorból, azoknak készült a WordPress.

Weblogkezelő programokról szóló sorozatunk méltó befejezése a ***WordPress*** rövid bemutatása, amely számos szolgáltatásának és kifinomultságának köszönhetően egyre népszerűbbé válik. A ***WordPress*** egy tökéletesen letisztult, könnyen használható felülettel rendelkező, MySQL-re alapozó, PHP-ben megírt weblogkészítő és karbantartó program.

A WordPress telepítése

Azoknak, akik már rendelkeznek némi tapasztalattal az ***Apache*** és a ***MySQL*** használatával kapcsolatban, a ***WordPress*** telepítése teljesen magától értetődő lesz. A ***WordPress*** az adatokat ***MySQL***-táblákban tárolja, vagyis telepítése előtt létre kell hoznunk legalább egy adatbázist. A ***WordPress*** lehetővé teszi, hogy egyetlen adatbázison belül több weblog adatait is tároljuk, ami különösen jól jön akkor, ha az általunk karbantartott oldal társszolgáltatója csak egyetlen ***MySQL*** adatbázist bocsát a rendelkezésünkre. Új ***MySQL*** adatbázis létrehozásához a ***mysqladmin*** program használatára van szükségünk, ami az én rendszeremen a ***/usr/local/mysql/bin*** elérési útvonalon található.

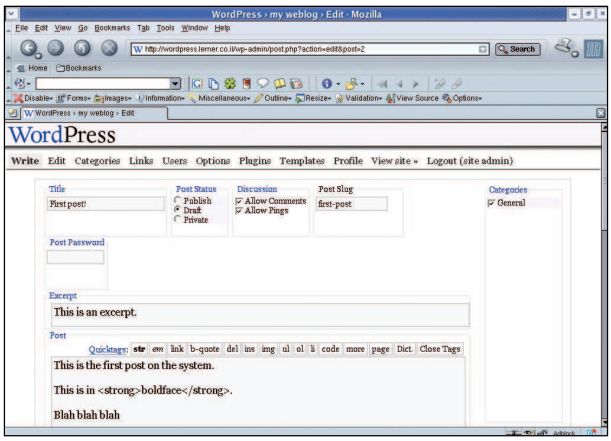
```
# /usr/local/mysql/bin/mysqladmin -p create
➥wordpress
```

A fenti parancs feltételezi, hogy rendszergazdai jogosultsággal rendelkezünk, vagyis a ***root*** felhasználó nevében futtatjuk. Ha létrehoztuk az adatbázist, a következőkben jogosultságot kell biztosítanunk a ***WordPress*** felhasználónak az adatbázisban lévő táblákhoz. Ezt az adatbázisba történő bejelentkezéssel tehetjük meg:

```
# /usr/local/mysql/bin/mysql -p -u root
```

Miután bejelentkeztünk, a következő parancsokkal jogosultságot is adhatunk az általam ***wpusernek*** elkeresztelt ***WordPress*** felhasználónak:

```
GRANT ALL PRIVILEGES ON wordpress.*
    TO wpuser@localhost IDENTIFIED BY ´wppass´;
GRANT ALL PRIVILEGES ON wordpress.*
    TO wpuser IDENTIFIED BY ´wppass´;
```



1. kép Egy weblog-bejegyzés bevitele a WordPress-ben

Most nézzük a ***WordPress*** forráskódját (65. CD Magazin/WordPress könyvtára), és bontsuk ki ezt a ***.tar.gz*** fájlt az ***Apache*** programunk dokumentumkönyvtárának gyökerében. Előfordulhat, hogy a ***WordPress*** fájljait külön könyvtárba vagy esetleg egy virtuális gép alá szeretnénk rakni, de ahhoz, hogy használni tudjuk ezeket a fájlokat, mindnek a dokumentumkönyvtár gyökerében kell lennie.
Most ugorjunk böngészőprogramunkkal a ***wp-admin/install*** címre és már helyben is vagyunk. A telepítőképernyő oldalai ellenőrzik, hogy minden hibátlanul telepítésre került-e, és néhány kérdést intéznek hozzánk. Rendszerint a képernyő alján lévő valamelyik hivatkozásra kell kattintanunk a folytatáshoz, majd be kell fejeznünk a telepítést.
Ha elmulasztottuk volna az adatbázis előzetes létrehozását, a ***WordPress*** figyelmeztet erre, tudtunkra adva, hogy a táblák létrehozásához szükség van az adatbázisra.
A ***WordPress*** telepítőjét csak egy alkalommal használhatjuk. Amennyiben egy már korábban telepített rendszeren kíséreljük meg a telepítő ismételt futtatását, a program a telepítés előtt az előző rendszer eltávolítását kéri.
Jó volt látni ezeket a kis figyelmességeket egy olyan programban, amely nyilvánvalóan egy műszakilag kevésbé avatott közönséget céloz.

A WordPress használata

A *WordPress* telepítése után az *admin* felhasználónév és egy véletlenszerűen előállított jelszó segítségével léphetünk be a programba. Rendszergazdaként lehetőségünk nyílik a bejegyzések létrehozására, szerkesztésére és törlésére, valamint a *WordPress* további felhasználóinak beállítására. A programban korlátlan számú felhasználó hozható létre, s ezek mindegyike egy 1-től 10-ig terjedő jogosultsági szintet kap. A 10-es jogosultsági szinttel rendelkező rendszergazda bármit megtehet a rendszerben, akárcsak a 10-es szinttel rendelkező egyéb felhasználók.

Természetesen nem akarjuk, hogy minden felhasználónak a rendszergazdáéval azonos jogai legyenek. A *WordPress* alapértelmezett telepítése számos szolgáltatás használatát korlátozza, így lehetőségünk van a különböző felhasználókhoz más-más jogosultsági szintet rendelni. Például az 5-ös és e feletti jogosultsági szinttel rendelkező felhasználók szerkeszthetik a képernyő jobb oldalán megjelenő hivatkozásokat, és képeket is feltölthetnek. A rendszergazdai felületen keresztül ezek a szintek megváltoztathatóak.

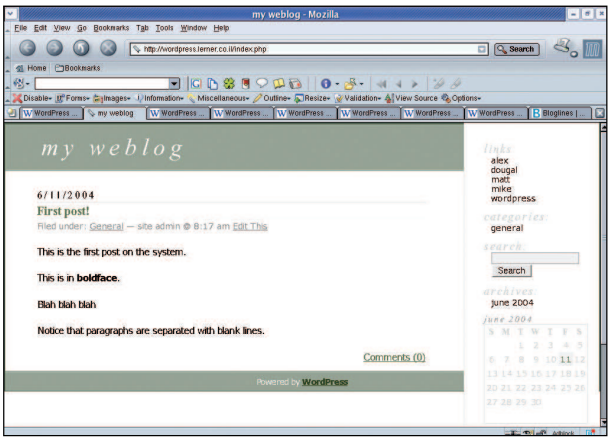
A *WordPress* rendszerfelügyeletét biztosító felület teljes egészében a Web-re épül, az adatbázis tábláinak a létrehozása után a teljes rendszer irányítása egy böngészőprogramon keresztül megoldható. A felügyeleti részben számos menüt találhatunk, amelyek némelyike további almenüket is kínál. Habár eleinte egy kicsit zavarban voltam az egyes lehetőségek elhelyezkedését illetően, de hamar érteni kezdtem az elrendezés logikáját és módosítani tudtam számos beállítást.

Kitaláltam például, hogy milyen jó lenne feltüntetni a blogtáramat, vagyis az általam rendszeresen látogatott blogok listáját. Ehhez a ➲ *Bloglines.com* web alapú számlálóját használtam, és az *OPML* használatával – amely az ilyen adatok kezelésének szabványos *XML*-alkalmazása – sikerült is a weblog listát létrehoznom. A lista *WordPress-be* történő importálása a *links* (*hivatkozások*) menü *blogroll* (*bloglista*) menüpontjával már csak egy kattintás volt. A felsorolt lehetőségek egyike az *OPML* fájl importálása volt, s amint kiválasztottam, a bloglistám láthatóvá vált a világ számára.

A felhasználói felület

A nyílt forrású programokat gyakran éri bírálat a barátságos felhasználói felület hiánya miatt. Ez főleg abból adódik, hogy az ilyen programokat író programozók saját maguk és munkatársaik számára készítik ezeket a szoftvereket, ami azt is jelenti, hogy nemigen foglalkoznak a program működéséhez nem közvetlenül kapcsolódó vonatkozásaival.

A *WordPress* alkotóinak becsületére válik, hogy a felhasználói felületre is rengeteg gondot fordítottak, megpróbálták elérni, hogy a műszakilag járatlanabb felhasználók számára is minden egyértelmű és kellemetlen meglepetésektől mentes legyen. Számos weblog rendszer feltételezi például, hogy a felhasználó ismeri az alapvető *HTML*-címkéket, és tudja, hogyan kell ezeket a dokumentumhoz adni vagy eltávolítani. A *WordPress* szerkesztőablaka lehetővé teszi a gyakorlottabb felhasználók számára, hogy ezeket kézzel írják be, ugyanakkor egy *JavaScript* gombokból álló készletet is biztosít a kevésbé járatos felhasználók számára. Ezek a gombok nem csupán jelzik a létező lehetőségeket, hanem a formájukkal mutatják azt is, hogy az adott formátum használatban van-e éppen. A sor végén lévő *Close* (*Bezár*) gomb egy különösen elmés ötlet volt: ezzel minden nyitott állapotban lévő *HTML* címkét lezárhatunk. A jó felhasználói felülettel rendelkező programokhoz hasonlóan a *WordPress* is sok olyan apró tulajdonsággal bír, amelyek hozzájárulnak a kellemes összkép kialakításához. Ezek közé tartozik az is, hogy két vagy három egymás utáni kötőjelet automatikusan cserél rövid vagy hosszú gondolatjelre. Lehetővé teszi továbbá, hogy a bejegyzéseket osztályokba soroljuk, ami alapján lehetnek piszkozatok (ez az alapértelmezett), magán vagy nyilvános dokumentumok. Így lehetőségünk van arra, hogy elkezdjünk dolgozni egy weblog bejegyzésen, közben elmenjünk ebédelni, és visszatérve ugyanonnan folytathassunk a félbehagyott munkát. A *WordPress* tisztában van vele, hogy az adattörlés olyan kényes művelet, amelyet a felhasználónak minden esetben meg kell erősítenie. Emiatt a törlés gombot úgy alakították ki, hogy amikor az egérmutatót elmozgatjuk felette, a háttere pirosra változik. További figyelemfelhívásként egy *JavaScript* ablak is megjelenik a felhasználó előtt a törlési művelet jóváhagyását kérve. A törlés kivételével könnyedén kijavíthatunk bármilyen a *WordPressben* esetleg elkövetett hibát egyszerűen visszatérve a kérdéses menüponthoz és egy más értéket kiválaszthatunk. Minden bejegyzés – beleértve a piszkozatokat és megjegyzéseket – ismételten szerkeszthető egészen addig, amíg készen nem állnak a nyilvánossá tételre.



2. kép Ugyanennek a bejegyzésnek a megjelenítése a WordPress alapértelmezett sablonjával.

Végül a *CSS* módosításával lehetőség van a *WordPress* megjelenésének és viselkedésének megváltoztatására. A *CSS* kezeli a betűtípusokat, a színeket, a méreteket és az elhelyezést; a sablonokat, amelyek nagyrészt szabványos *PHP*-ben íródtak, sőt a bővítményeket is, amelyekkel szinte bármit megváltoztathatunk. A sablon megváltoztatására lehetőséget ad maga a *WordPress* is, de úgy gondolom, ennek a folyóiratnak az olvasói inkább a fájl *Emacs*-szal vagy *vi*-vel történő közvetlen szerkesztését fogják választani.

Egy bővítmény telepítéséhez először le kell töltenünk, majd a megfelelő bővítmény-könyvtárba kell másolnunk az állományokat. A bővítmény elindítása már teljes egészében a Webes felületen keresztül történhet. Mindez azt is jelenti, hogy a rendszergazdák számos bővítményt telepíthetnek a felhasználóik számára, és rájuk bízhatják, hogy melyek

azok a bővítmények, amelyeket aktiválni szeretnének. A *WordPress* terjesztési csomagja számos mintabővítményt is tartalmaz, továbbiak letöltésére pedig a program honlapján találunk lehetőséget.

Összegzés

Az elmúlt néhány hónapban egy sereg különféle weblog programot vizsgáltunk meg. A *COREBlog* kivételével – amely egy *Zope* termék és a *Zope* kiszolgálónkra történő telepítése gyorsan és könnyen zajlott – a *WordPress* telepítése volt messze a legegyszerűbb és leggyorsabb. Teljes szolgáltatáslistával rendelkezik, s ezek nagy része tiszta és könnyen használható felhasználói felületen keresztül működtethető. Még a kezdő számítógépfelhasználók is használhatják rendszeres munkájukhoz ezt a programot. Habár a háttérben működő kód és technológia – a *PHP* és a *MySQL* – nem tartoznak a kedvenceim közé, szolgáltatásainak sokrétűsége valamint a különböző felületeken dolgozó felhasználók növekvő közössége a *WordPresst* helyezi az első helyre.

Linux Journal 2004. szeptember, 125. szám

Reuven M. Lerner régóta dolgozik web és adatbázis tanácsadóként, illetve fejlesztőként. Végzett hallgatóként elsőéves kutatásait végzi a Northwestern University Learning Sciences programjában. Weblogja az altneuland.lerner.co.il, ő maga pedig a reuven@lerner.co.il címen érhető el.

Értékeld a Linuxvilág cikkeit!

Honlapunkon mostantól pontszámmal értékelheted a Linuxvilág cikkeit. Minden szám tartalomjegyzékében, az adott cikk dobozában megjelölheted, hogy milyen osztályzatot adsz rá 1-től 5-ig.
A cikkek összesítő oldalán is értékelhetsz, akár egyszerre többet is. Jelöld meg, hogy melyiknek milyen osztályzatot adsz és kattints az oldal tetején vagy alján található „Pontozás" gombra.
Reméljük sokan fognak élni a lehetőséggel és ezáltal hasznos visszajelzést kapunk arról, hogy mely cikkek és témák a legnépszerűbbek.
Az osztályzatok alapján hamarosan megjelentetünk egy folyamatosan frissülő toplistát is.

Segítséged előre is köszönjük!

A Linuxvilág csapata

© Kiskapu Kft. Minden jog fenntartva

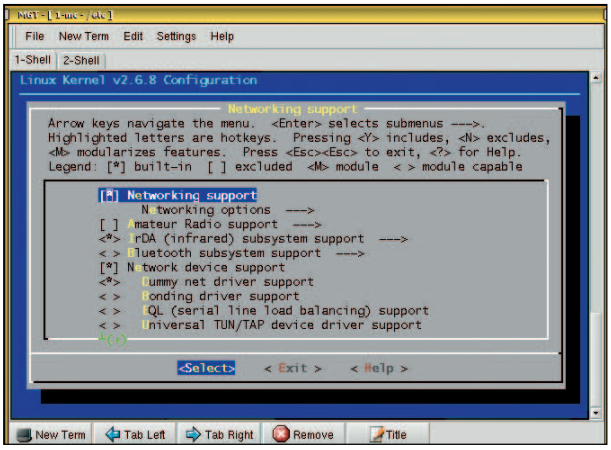
A pingvin igenis vándormadár (2. rész)

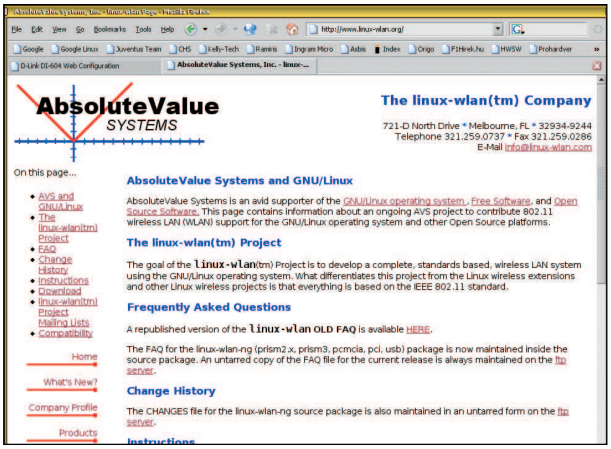
Avagy használjunk Linuxot a mobil eszközökön.

A cikksorozat bevezető részében telepítettünk a gépünkre egy ***Debian Linuxot***, valamint elkezdtük beállítani a rendszert. Első lépésben megvizsgáltuk milyen kernel és felhasználói program szintű megoldások léteznek a mobil eszközünk energiagazdálkodási paramétereinek beállítására.
Ha már megteremtettük a lehetőséget, hogy a gépünkkel eltávolodjunk a hálózati tápellátástól – a konnektortól, akkor jó volna ha a hálózati kábelt is ki tudnánk húzni a gépből és vezeték nélkül tudnánk Internetezni. Nézzünk is gyorsan pár megoldási lehetőséget.

A WLAN – a vezeték nélküli hálózati csatlakozás

Napjainkban egyre népszerűbb a technológia. Mostanra minőségben és sebességben is elérte azt a szintet, hogy bizonyos felhasználási környezetekben alternatívát nyújtson a vezetékes hálózati kapcsolattal szemben. Arról nem is beszélve, hogy bizonyos környezetekben pedig egyenesen forradalmi változásokat idézhet elő. Ilyen például a szállodákban, konferencia központokban, éttermekben, oktatási intézményekben vagy éppen a reptereken gomba módjára szaporodó úgynevezett a forró pontok (hot-spot), ahol egy vezeték nélküli hálózati csatlakozással ellátott eszközzel szabadon (esetleg díjfizetés ellenében) tudunk az Internetre csatlakozni. Ezzel az említett intézmények olyan értéknövelt szolgáltatást tudnak a vendégeik, látogatóik számára nyújtani, amely minden bizonnyal nagyon gyorsan nagyon népszerű lesz.
Mindehhez semmi másra nincs szükségünk, mint beszerezni egy megfelelő vezeték nélküli hálózati kártyát, betenni a gépünkbe és beállítani. Előbbi kettőben ugyan nem, az utóbbiban viszont megpróbálok ezen hasábokon segítséget nyújtani.
Mind a 2.4-es, mind a 2.6-os rendszermag támogatja a vezeték nélküli hálózati kártyákat, így minden bizonnyal nem lesz problémánk ezen kártyák telepítésével. A régebbi ilyen kártyák esetleg jelenthetnek kihívást, de a ***Linux*** egyik legnagyobb áldása, hogy ezekre a problémákra és megoldásukra, vagy az oda vezető útra percek alatt rábukkanhatunk az Interneten. Erre a legjobb kiindulási pont vagy a ➲ ***www.tuxmobile.org***, vagy egyszerűen a ➲ ***www.google.com/linux*** oldal.
A most a kiskereskedelmi forgalomban kapható, otthoni vagy kisvállalati (***SOHO – small office, home office***) környezeti felhasználásra szánt kártyák többsége a 2.6-os rendszermagban a ***Prism*** meghajtót használja. Természetesen van





olyan kártya is, amely nem, de ezt egy pillanat alatt kideríthetjük. 2.6-os rendszermagot használva futtassuk a ***root*** felhasználó jogosultságaival az `lspci` parancsot, amely kilistázza nekünk az összes a rendszerben jelen lévő PCI eszközt. Amennyiben a gépünkben beépített, vagy ***PCI*** kártyás vezeték nélküli hálózati csatlakozás van, akkor ezek között meg kell, hogy találjuk az eszközünket. Amennyiben ***USB***-n keresztül csatlakozó hálózati kártyánk van, akkor az `lsusb` parancs által listázott eszközök között kell, hogy szerepeljen a hálózati kártya. Amennyiben pedig PC kártya (***PCMCIA*** kártya) formátumú ***WLAN*** kártyánk van, akkor a ***pcmcia-cs*** csomag `cardctl` programjának futtatásával kaphatjuk meg a jelenleg dokkolt PC kártyák adatait.

Egy példa a waproamd-re

Nézzünk egy példát arra, hogyan is hozhatunk létre egy ismert elérési ponthoz egy olyan szkriptet, amely az elérési pont felismerése után automatikusan csatlakozik a felismert elérési ponthoz.
Hozzuk tehát létre az */etc/waproamd/scripts* könyvtárban az elérési pont *MAC* címével, vagy *ESSID* nevével ellátott állományt. Fontos, hogy a létrehozott állomány rendelkezzen futtatási joggal, különben nem fog elindulni!
A waproamd szkriptjeinek futtatási sorrendjéről jó tudni, hogy először a *MAC* címmel ellátott állományokat próbálja futtatni, azok közül is először a kisbetűseket, majd a nagybetűseket. Ha nem talált illeszthető mintát, akkor megpróbálja az *ESSID* alapján megkeresni a megfelelő futtatható állományt. Ha ez sem járt sikerrel, akkor jön a *default* nevű állomány futtatása, amely megpróbálja kitalálni, hogy mi az elérési pont *MAC* címe, vagy *ESSID* azonosítója.
Mennyiben *MAC* címmel ellátott állományt szeretnénk létrehozni, akkor adjuk névnek például a 00-01-02-03-a0-a1 nevet és adjunk rá futtatási jogot. Amennyiben a HomeWLAN *ESSID*-hez szeretnénk szkriptet írni, akkor adjuk az állománynak az essid:HomeWLAN nevet. Amennyiben szóközt, vagy egyéb speciális karaktert is tartalmazna az elérési pont neve, akkor használjuk a szabványos *http URL escape* szintaktikát, vagyis tegyük a karakter ASCII kódját egy % után.
A szóköz például a %32 kóddal írható le.
Ha ezzel megvagyunk, akkor még a *WEP* titkosításhoz tartozó kulcs beállítása lehet hátra. Ezt a */etc/waproamd/keys* könyvtárban az előbbi névhasználattal megegyező módon létrehozott állományban tehetjük meg. Egyszerűen írjuk bele azt a 10, vagy 26 számjegyet, ami kell.

Nézzünk egy példát egy *WLAN* indító programra. Ha létrehoztuk a megfelelő állományt, akkor helyezzük el benne az alábbi tartalmat:

```
#!/bin/bash
    case “$1” in
        start)
            iwconfig eth1 essid “HomeWLAN”
            iwconfig eth1 mode Managed
            iwconfig eth1 key 1122334455
        ;;
        stop)
            iwconfig eth1 key off 2> /dev/null
        ;;
    esac

[ “$1” = “start” ] && exec
/etc/waproamd/waproamd.action eth1 up
[ “$1” = “stop” ] && exec
/etc/waproamd/waproamd.action eth1 down
```

Ezzel csináltunk egy szkriptet, amely csatlakozik a HomeWLAN nevű elérési ponthoz az eth1 hálózati csatolóval, amely az 1122334455 kulcsot használja a *WEP* titkosításhoz.
Amennyiben a hordozható gépünkön van olyan gomb, amely a vezeték nélküli hálózati csatolót aktiválja, akkor ezt a gombot hozzátársítva a waproamd programhoz megtehetjük azt, hogy nem állandóan fut a démon, hanem csak akkor amikor mi ezt szeretnénk. Ez nagyon hasznos lehet abból a szempontból, hogy így feleslegesen nem meríti az állandó keresés az akkut.

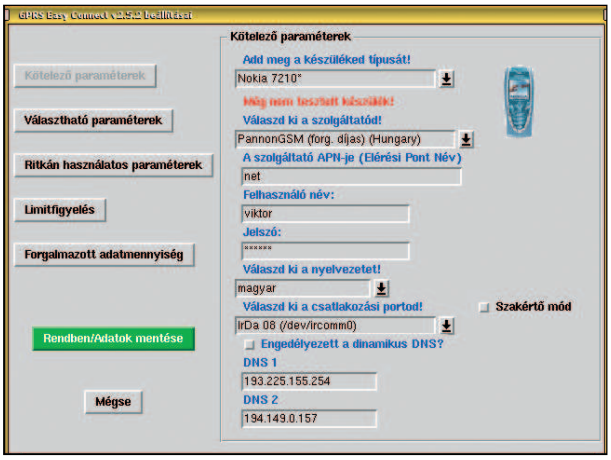
Amennyiben már telepítettünk grafikus felületet a gépünkre, akkor a `cardinfo` parancs futtatására egy jópofa grafikus PC kártya felügyeleti programot tudunk indítani.
Ha megtaláltuk, hogy a kártyánkat milyen lapkakészlettel (chipset) látták el, akkor a mag beállításai között a ***Device Drivers/Networking Support/Wireless LAN*** menüpont alatt megtaláljuk a rendszermag által támogatott modulokat. Mint említettem a beépített, a ***PCI*** és a PC kártyák jelentős része a ***Prism*** meghajtót használja. Amennyiben ***802.11b*** szabványú kártyánk van, akkor a ***Hermes*** lapkakészlet és a ***Prism 2.5*** modulokat fordítsuk a rendszermagba, amennyiben viszont ***802.11g*** szabvány (22/54/108Mbit/s sebességű kártyák) is támogató kártyánk van, akkor válasszuk az ***Intersil Prism*** modult és fordítsuk bele a magba.
Ha újrafordítottuk a rendszermagot és újraindítottuk a gépet ezzel a rendszermaggal, akkor a betöltés folyamán már meg kellett, hogy jelenjen a kártya. Ezt egyébként ellenőrizhetjük a `dmesg | more` parancs kiadásával. Ha a kártya fizikailag rendelkezésre áll, akkor telepítsük fel a ***WLAN*** kártyák kezeléséhez szükséges programokat tartalmazó ***linux-wlan-ng*** csomagot. Ebben a csomagban található többek között az `iwconfig` állomány is, amellyel a kártya kapcsolódását tudjuk beállítani a hozzáférési pontokhoz (access point). Ha ez is megtörtént, akkor nincs más dolgunk hátra, mint beállítani a hálózati beállításokat. Kezdjük azzal, hogy megkeressük a megfelelő elérési pontot és megtudjuk a paramétereit. A legfontosabb paraméterek, amelyekre szükségünk lesz az elérési pont ***ESSID***-je, ami gyakorlatilag egy szöveges azonosító, az elérési pont MAC címe, a használt titkosítás és a kulcs kiosztásának módja. Ezeket az információkat vagy elkérjük a rendszergazdától, vagy megpróbáljuk felderíteni. Utóbbiban segítségünkre lehet például a ***waproamd*** (***Wireless Access Point Roaming Daemon***) nevű csomag. Ez a program alapértelmezésben a rendszer indításakor elindul és folyamatosan keresi az elérhető csatlakozási pontokat. Amint talál egy elérési pontot, akkor megnézni, hogy az */etc/waproamd/scripts* könyvtár tartalmaz-e az elérési pontra jellemző információt – vagy a ***MAC*** címet, vagy az ***ESSID*** azonosító nevet – és amennyiben igen, úgy lefuttatja az adott szkriptet. Ha nem talál, akkor az alapértelmezett (default) szkript fut le és próbálja meg létrehozni a csatlakozást. Ez a nyilvános hotspot-ok esetén egy jó választás lehet, azonban ha az elérési pont tartalmaz valamilyen speciális beállítást, akkor már kérdéses, hogy a felderítés mennyire jár sikerrel. Amennyiben olyan elérési ponthoz szeretnénk csatlakozni, amelyik ***WEP*** titkosítást használ, akkor az */etc/waproamd/keys* könyvtárba helyezzünk el egy ***<access_point_MAC_címe>.wep*** nevű állományt, amely a ***WEP*** kulcsot tartalmazza.
Az előbb bemutatott `waproamd` egy nagyon jópofa szolgáltatás, de az élet azért nem fenékig tejfel. Sajnos vannak olyan

kártyák – főleg a régebbi típusok -, amelyek nem, vagy csak részben támogatják a felderítési funkciót. Arról, hogy kártyánk rendelkezik-e ilyen képességekkel meggyőződhetünk az `iwlist <wlan_csatoló_neve> scan` parancs beírásával, ahol a ***WLAN*** csatoló neve az az interfész, amelyik az `iwconfig` parancs által listázott eszközök között ***WLAN*** támogatást mutat. Amennyiben az `iwlist` azt adná vissza, hogy az eszközünk nem támogatja a ***WLAN*** felderítő opció használatát, még ne keseredjünk el. Van ugyanis lehetőség arra, hogy kézzel beállítva a kapcsolati paramétereket használni tudjuk a vezeték nélküli hálózatot. Sőt, annak ellenére, hogy az `iwlist` nem ad vissza értékelhető eredményt, a `waproamd` még talán működni is fog. Az én notebookom például egy olyan beépített ***WLAN*** kártyával van megáldva, amelyik az `iwlist` parancsra meg sem mukkan, de remekül be tudtam állítani a `waproamd`-t és a bekapcsoló gombra tökéletesen fut. Amennyiben valamilyen oknál fogva mégis a kézi beállítás mellett döntünk, úgy ehhez az `iwconfig` parancsot kell használni. Használata teljesen megegyezik az `ifconfig` parancs használatával, gyakorlatilag hasonló írásmódot kell használni. Nagyon jó súgóoldal tartozik az `iwconfig` és tulajdonképpen az összes vezeték nélküli hálózati csomagban szereplő eszközhöz, úgyhogy nézegessük szorgalmasan a súgót!
További hasznos információkhoz juthatunk a ➲ http://www.linux-wlan.org/ oldalon, valamint a már említett oldalak bármelyikén.
Most tehát beállítottuk a vezeték nélküli hálózati kártyánkat, ezzel már boldogan tudunk netezni. Amíg van a közelben egy elérési pont. De mi történik, ha nincs? Akkor bizony marad a régi jól bevált megoldás, a mobiltelefonos Internet. A mobiltelefonos Internet elérésnek két fajtája van, a kapcsolt vonali elérés (ma már általában minden szolgáltató támogatja a ***HSCSD***-t, amely egy nagy sebességű kapcsolt vonali elérést biztosít), illetve a csomagkapcsolt megoldás, a jól ismert ***GPRS***. Mi most inkább a ***GPRS*** kapcsolattal foglalkozunk, mivel ennek használata a jövőben talán lényegesen elterjedtebbé válik, mint a kapcsolt vonali netezés.
Ahhoz azonban, hogy ***GPRS***-en át tudjuk a hálózatot elérni a notebookunkról, ahhoz bizony először össze kell kapcsolni a gépet és a mobiltelefont. Ezt megtehetjük vezetékkel, vagy infrán, vagy ***bluetooth***-on keresztül. Én most először az infrás kapcsolatot mutatom be, mert az infra mára gyakorlatilag minden notebookban és telefonban megtalálható, így ezt minden bizonnyal minden Kedves Olvasó ki fogja tudni próbálni.

Az IrDA

Ahhoz, hogy a notebookunk infra portját a hálózat elérésére tudjuk használni, először is elérhetővé kell tenni. Nem meglepő módon a dolgunk az, hogy először a megfelelő rendszermag modulokat belefordítsuk a kernelbe. Indítsuk el ismét a kernel beállítási felületét és menjünk a ***Device Drivers/Networking Support/IrDA (infrared) subsystem support*** menüponthoz. Ott találunk egy pár modult, ezeket én az alábbi összeállításban fordítottam a rendszermagba:

```
---   IrDA protocols
<*>   IrLAN protocol
<M>   IrNET protocol
<*>   IrCOMM protocol
```




```
[*]   Ultra (connectionless) protocol
---   IrDA options
[*]   Cache last LSAP
[*]   Fast RRs (low latency)
```

Ha ez megvan, akkor lépjünk be az ***Infrared-port device drivers*** menübe és állítsuk be az alábbi modulokat is:

```
--- SIR device drivers
<M> IrTTY (uses Linux serial driver)
--- Dongle support
[*] Serial dongle support
--- Old SIR device drivers
<M> IrPORT (IrDA serial driver)
```

Ennek a három modulnak a beállításával nekem eddig majdnem minden notebookon sikerült beállítani az infra kapcsolatot, de ettől függetlenül lehetnek olyan eszközök is, amelyekhez más modulokat is be kell fordítani. Erről bővebbet olvashatunk a rendszermag forrásának a dokumentációjában.
Ha idáig eljutottunk, akkor természetesen fordítsuk újra a kernelt és indítsuk újra a gépet. Miután betöltődött a friss rendszermag, a `dmesg`-gel megnézhetjük, hogy a gép az induláskor felismerte-e az infra portot a gépben. Ha nem, akkor próbálkozhatunk más modulokkal a rendszermagban, ha igen, akkor folytassuk a beállításokat.
Következő dolog, amit el kell érnünk az az, hogy megszólaltassuk a gépünk infra portját. Ehhez fel kell tenni az ***irda-utils*** nevű csomagot. A telepítés folyamán a telepítő több kérdést is feltesz. Az első, hogy rendszerindításkor az infra engedélyezve legyen-e. Válasszuk az igent. Második kérdés, hogy engedélyezzük-e a felderítési módot, amellyel a gépünk a közelben lévő eszközökhöz akar majd csatlakozni. Én itt is az igent választottam.
Megjegyzés: a notebook akkujának kímélése szempontjából esetleg indíthatjuk kézzel is az infrát, ha úgy érezzük erre szükségünk van.
Következő kérdés, az IrDA eszköz típusa. Itt notebookok esetén 95%-ban a `serial` választ kell megjelölni, hacsak nem valamilyen natív – általában külső – eszközt csatlakoztattunk. Következő lépésben meg kell adnunk a soros port helyét. Nálam ez a */dev/ttyS1*. Az utolsó kérdés az infra vezérlőnkhöz tartozó `dongle` típusa. Notebookok esetén itt

99%-ban a none választ kell kijelölni.
Ha ezzel megvagyunk, akkor jelentem, bekonfiguráltuk az infra portot. Most tegyünk vele szembe egy megfelelő telefont és indítsuk el az irdadump nevű programot. Ez a folyamat gyakorlatilag egy pingelésnek fog megfelelni. Az infra port összegyűjti a látható eszközöket, amelyeknek meg kell jelenniük a kidobott listában. Nálam egy Nokia 7210-es mobiltelefon a tesztalany és a következő üzenetet kapom:

```
19:52:14.297525 xid:rsp db94b91e < 0000edd6 S=6
s=0 Nokia 7210 hint=b125 [ PnP Modem Fax Telephony
➥IrCOMM IrOBEX ] (27)
19:52:14.303534 xid:cmd db94b91e > ffffffff S=6
➥s=1 (14)
19:52:14.393525 xid:cmd db94b91e > ffffffff S=6
➥s=2 (14)
19:52:14.483510 xid:cmd db94b91e > ffffffff S=6
➥s=3 (14)
19:52:14.573497 xid:cmd db94b91e > ffffffff S=6
➥s=4 (14)
19:52:14.663481 xid:cmd db94b91e > ffffffff S=6
➥s=5 (14)
19:52:14.753466 xid:cmd db94b91e > ffffffff S=6
➥s=* trikko hint=4400 [ Computer LAN Access ] (22)
```

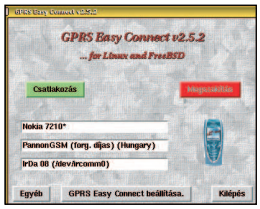
Ez teljesen jó, az első sorban ott látszik a telefon, meg az, hogy ez egy modem és egy faxmodem is egyben. Ezzel gyakorlatilag működik is az infra és a telefon közötti kommunikáció. Nincs más hátra, mint az Internetes kapcsolat beállítása.

GPRS Internet mobiltelefonos beállítása

Ahhoz, hogy beállítsuk a telefonos kapcsolatot, szükségünk lesz a ppp csomag telepítésére. Ennek a ***Debian Sid*** gyűjteményben a cikk írásakor a 2.4.2-es verziója volt feltöltve. Célszerű ebből a csomagból mindig a legfrissebbet használni, mert olykor ez sok fejtöréstől kímélhet meg minket.
Ha feltettük a *ppp* csomagot, akkor két út áll előttünk.
Az első, hogy szépen kézzel elkészítjük a *ppp* kapcsolódási szkripteket, a másik, hogy egy segédprogramhoz fordulunk, amely ezt megteszi helyettünk. Bár az előbbi sem egy különösen bonyolult feladat, ráadásul az Interneten tucatszám találhatunk tippeket, tanácsokat és példa beállításokat is, akár szolgáltató specifikusan is, mi most mégis nézzünk egy programot, amellyel nem csak könnyen beállíthatjuk és vezérelhetjük a *GPRS* kapcsolatot, de mindennek fejében mindezt azzal a tudattal tehetjük, hogy egy olyan segédeszközt használunk, amelyet honfitársaink készítettek. A program neve: *GPRS Easy Connect*.

Amit a GPRS Easy Connectről tudni kell

Személy szerint a nyáron, amikor a fejembe vettem, hogy végre beállítom a *GPRS* kapcsolatot a gépemen, hogy ne csak kábel mellől tudjak nezetni, rengeteget kutakodtam a hálózaton, hogy találjak valami használható, felhasználóbarát megoldást a kapcsolat beállítására. Olyat, amelyet nem csak én, de adott esetben egy mindennapi felhasználó is könnyen és gyorsan be tud állítani. Nem kis örömömre



rábukkantam a *GPRS Easy Connect*-re, amelyre a mai napig is úgy tekintek, mint egy áldása. A program egy teljesen jól használható grafikus felülettel rendelkezik és a beállítása tényleg nem igényel különösebb tudást. Amire pedig szükség lehet, az le van dokumentálva.
Aki kedvet kapott a program kipróbálásához, az letöltheti a legfrissebb verziót a ➲ http://easyconnect.linuxuser.hu/modules.php?name=index webcímről.
Egy másfél megabájtos ***tar.gz*** csomagot kell letöltenünk, amelyet kicsomagolunk, majd a telepítő könyvtárában található INSTALL állomány futtatásával egy nagyszerű grafikus telepítőt indíthatunk. A telepítés menete nagyjából követi a „next-next-finish" módszert, tehát senkinek nem okozhat különösebb problémát.
Ha befejeződött a telepítés akkor az /usr/bin/gprsec parancsot futtatva indíthatjuk a programot. A program a telepítőhöz hasonlóan egy nagyon baráti grafikus felületet ad mind a beállítások elvégzésére, mind pedig a kapcsolat nyomonkövetésére.
Kezdjük a beállításokat a *„GPRS Easy Connect beállítása"* feliratú gombra kattintva. Itt kiválaszthatjuk a készülékünk típusát, a szolgáltatót, a szolgáltató *APN*-jének nevét megadhatjuk, akárcsak a felhasználó nevünket, jelszavunkat.
Ki kell választanunk továbbá a csatlakozási kaput, ez valószínűleg a */dev/ircomm0* lesz. Engedélyezhetjük a dinamikus *DNS*-t, vagy akár megadhatjuk saját magunk is két DNS kiszolgáló címét, ha nekünk éppen úgy tetszik.
A *Választható paraméterek* gombra kattintva beállíthatjuk a kapcsolat sebességét, az újracsatlakozások számát és időtartamát.
A *Ritkán használatos paraméterek* között találunk egy jelölő négyzetet, amelynek a beállítása sok fejtöréstől kímélhet meg minket. Nálam a telepített alapbeállításon nem volt beállítva az *Önhitelesítés tiltása*. Ez olyan kellemetlen problémát okozott, hogy bár a gépem felcsatlakozott a hálózatra, de mivel a vissza irányú hitelesítés nem működött, ezért azonnal le is bomlott a kapcsolat. Erre figyeljünk oda, ha ilyet tapasztalnánk, ez lehet a megoldás.
A *Limitfigyelés* menüpont alatt beállíthatunk Mb-ban mért adatforgalmi korlátot. Hasznos lehet azoknak a felhasználóknak, akik egy bizonyos mennyiségű ingyenes, vagy előre kifizetett forgalommal rendelkeznek. A *Forgalmazott adatmennyiség* menüben statisztikai adatokat találunk.
Amennyiben befejeztük a beállításokat, akkor a *Rendben/Adatok mentése* gombra klikkelve elmenthetjük a beállításainkat. Ezután nincs más dolgunk, mint a nagy zöld *Csatlakozás* gombra kattintani és internetezni.
Miután elindult a kapcsolat, az ifconfig paranccsal listázott csatlakozók között meg kell jelennie egy *pppX* csatolónak, ez lesz a *GPRS* kapcsolat által használt eszköz.
Ezzel a cikk végére is értünk. A következő részben további hasznos tippeket és beállítási tanácsokat osztok meg a Kedves Olvasóval. Aki azonban türelmetlen lenne, azt csak arra tudom bíztatni, hogy használja a gépét, próbálkozzon önállóan a beállításokkal. Sok sikert!

Illés Viktor

A felhasználói viselkedés vizsgálata (1. rész)

Bevezetés, avagy mire jók a Nagy Testvér módszerei?

Kémkedni ugyebár nem szép dolog, tehát egyáltalán miért figyeljük meg a felhasználókat? Napjainkban a világháló (World Wide Web) egyik legjellemzőbb problémája az információbőség. A felhasználók ugyanakkor a lehető legegyszerűbben szeretnének hozzájutni az általuk igényelt tudáshoz. A legtöbb honlapot a megrendelő, vagyis egy cég, egy intézmény, vagy egy kormányzati szerv igénye szerint készítenek el. Gyakori ugyanakkor, hogy a megrendelő által megfogalmazott elvárások nem igazodnak a végfelhasználók, külső szemlélők elvárásaihoz. Általában nem ezeket felmérve tervezik meg a portálokat. Ez pedig azt jelenti, hogy bár a honlap híven tükrözi az adott szervezet belső struktúráját, egy új látogató számára ez a rendszer tökéletesen ismeretlen, és így az információ elérése számára igen nehézkes lesz.
Persze a látogatók igényei is folyamatosan változnak, s ilyenkor egy jó honlapnak ezt követnie kell. Ehhez pedig szükség van a látogatók viselkedésének, szokásainak folyamatos figyelésére, az adatok elemzésére, valamint a kapott eredmények „visszaforgatása", viselkedési modellek alkotására.

Nézzünk egy példát!

Képzeljük el, hogy van egy a cégünk termékeit bemutató portálunk. A termékszerkezetünk olyan, hogy az egyes árucikkek önállóan is „megállják a helyüket", de együtt is jól használhatóak. Tegyük fel mondjuk, hogy asztalt és széket gyártunk.
Az első kérdés, ami eszünkbe jut nyilván az lesz, hogy melyek a leggyakrabban megtekintett termékleírások, vagyis mit keresnek leginkább a látogatóink. Az ilyen kérdésekre egészen egyszerűen választ kaphatunk, ha a webkiszolgáló eseménynaplóján lefuttatunk egy keresőprogramot. Rögtön látszik, hogy az asztal terméklapját százszor töltötték le, míg a székét ezerszer. Ez azonban csak az egyik lehetséges kérdés és az egyik lehetséges válasz. Az így kapott számszerű adatokból ugyanis nem lehet például olyan következtetéseket levonni, amelyek megkönnyítenék annak eldöntését, hogy egy adott termék oldalán milyen más áruinkra utaló szöveget célszerű elhelyezni. Ehhez „rejtett összefüggéseket" kell felfedezni. Ha tudjuk, hogy a zöld asztalt kereső látogatóink nagy része megnézi a zöld székek kínálatát is, akkor az asztal terméklapján egy közvetlen utalást elhelyezve a zöld székeinkre várhatóan jelentős mértékben megnő azok terméklapjainak forgalma. Ráadásul ezzel a könnyítéssel a vásárlóknak is örömet szereztünk, hiszen nem kellett keresgélniük.
Természetesen ennél sokkal összetettebb összefüggésekre is kíváncsiak lehetünk. Érdekes, a teljesítmény optimalizálása végett vagy marketingstratégiai célokra felhasználható adatokat nyerhetünk például az egyes aloldalak közötti „átkattintási arányokból", az egyes lapokon eltöltött időkből, vagy a leggyakoribb útvonalakból.
Az összefüggések és számszerű adatok folyamatos figyelésével idejében felfigyelhetünk és felkészülhetünk a látogatók viselkedésének változására, vagy lemérhetjük egy-egy reklámkampány, vagy a marketingmunka hatékonyságát is.

Honnan és hogyan lesz adatunk?

A felhasználói viselkedés modellezése alapvetően két részből áll: az adatgyűjtésből és az elemzésből. A gyűjtés során tetszőleges eljárással olyan adatokat akarunk előállítani, amelyek kellően részletesek ahhoz, hogy az elemzést végző programok használható modelleket alkothassanak belőlük. Mivel csak az „átlagos látogató" paramétereire vagyunk kíváncsiak, azaz nincs külön-külön szükségünk minden egyes látogató viselkedésének a leírására, a megszemélyesítést lehetővé tévő adatokat csak az egyes felhasználók elválasztására használjuk. (Ez tipikusan hálózati címet és felhasználói nevet jelenti.) A kiszolgáló bizonyos beállításaival megoldható a személyes adatok teljes kizárása is.
Az adatgyűjtést végző kódot beépíthetjük a honlap programjába, futtathatjuk a kiszolgálón vagy az ügyfélen. Utóbbi esetben lehet ez egy böngésző-kiterjesztés vagy különálló program is. A honlap programjának módosítása munka- és költségigényes, az ügyfél oldalán végzett megfigyelés viszont adatvédelmi problémákat vet fel. A kiszolgálón végzett adatgyűjtés ezzel szemben a honlaptól és az ügyféltől is függetleníthető, de ennek a módszernek is megvannak a maga problémái. Az egyes módszerek előnyeit és hátrányait a sorozat következő tagjában fogom részletesen elemezni.
Az összegyűjtött adatok sok felesleges információt is tartalmazhatnak, ezért az adatsorokat minden esetben az adott portálra jellemző módon meg kell tisztítani. Mivel a valóságos információt néha több bejegyzés összefüggése (egyidejű megjelenése) jelenti, szükség lehet a honlapra jellemző kapcsolatok és összefüggések alapján egy további előfeldolgozási lépésre is. Előfordulhat aztán, hogy több, látszólag eltérő bejegyzés valójában azonos oldalt jelent. Az ilyen esetek kezelésére is be kell építeni a feldolgozóba egy szűrő és átalakító alrendszert.

Elemzés

A modellalkotásnak igazodnia kell a felhasználók viselkedésére vonatkozó adatok begyűjtésének módszeréhez. Az elkészülő modelleket alapvetően befolyásolja, hogy az egyes felhasználókat milyen módon sikerült megkülönböztetni, ez pedig erősen függ az adatgyűjtési módszertől. Akár az ügyfél oldalán gyűjtjük az adatokat, akár a a portál motorjába építettük be a szükséges kódot, a látogatók elkülönítése tulajdonképpen egész egyszerűen megoldható, hiszen csak arra kell figyelni, hogy az adott programban nyilvántartsuk, hogy most egy másik felhasználó használja a portál. A portálba beépített módszer esetén ez egy beléptetési (regisztrációs) folyamattal könnyedén megoldható, ügyféloldali gyűjtés esetén pedig szintén a programba épített azonosítási folyamat biztosítja a megfelelő elkülönítést.

A kiszolgálón begyűjtött adatok alapján általában jóval nehezebb a felhasználókat megkülönböztetni. Ennek az az oka, hogy a webkiszolgáló alapesetben nem tárol semmiféle, az egyes felhasználók elkülönítésére alkalmas adatot. További nehézség, hogy az esetleges megkülönböztetés alapjául szolgáló adatok (IP cím, felhasználói név, dátum) felhasználhatóságát az átmeneti tárak és tűzfalak is jelentésen rontják, hiszen ezek egy adott alhálózat valamennyi tagjának kéréseit azonos adatokkal küldik el a webkiszolgálóhoz.

A látogatók megkülönböztetését a kiszolgáló oldalán jelentősen megkönnyíti, ha egy megfelelően megírt modul segítségével minden egyes látogatóhoz egyedi munkamenetet és egyedi azonosítót rendelünk. Erről részletesen a következő cikkben, az adatgyűjtési módszerek ismertetése kapcsán fogok írni.

Röviden tehát, ha csak a legegyszerűbb, a kiszolgáló oldalán az eseménynaplóból kigyűjtött adatokat tudjuk használni, az a legfontosabb, hogy valamilyen módon az egyes látogatókat el tudjuk különíteni. Pontosabban fogalmazva azt kell megoldanunk, hogy a független oldallekéréseket valamilyen módon egyedi látogatói szakaszokba rendezzük. A módszer lényege, hogy az eseménynapló bejegyzéseiben szereplő IP cím, időbélyeg és minden egyéb, a felhasználóra jellemző adat (ilyen például a böngésző típusa) segítségével az egy meghatározott időtartamba eső, azonos helyről érkező kéréseket egy felhasználó látogatásának tekintjük. A megfelelő időtartam megválasztása nem könnyű. Érdemes több különböző intervallumhosszal elkészíteni a modelleket, hogy a látogatók viselkedési mintázatának megfelelőt kiválaszthassuk. Általánosságban elmondható, hogy a harminc perces intervallum a legjellemzőbb, tehát ezt tekinthetjük kiindulási alapnak. Egy rövid híreket közlő oldal ettől természetesen lefelé fog eltérni (gyorsabb az információ átadása), míg egy hosszabb publikációkat ismertető honlap felfelé növelheti az időkeretet, hiszen egy cikk végigolvasása hosszadalmas is lehet.

Ha sikeresen előállítottuk a felhasználókhoz rendelhető egyedi látogatási szakaszokat, máris készen állunk arra, hogy elkészítsük a modelleket. A modellkészítés során – amint azt már említettem – átlagosnak tekinthető látogatói viselkedésmintákat keressük. Ilyenek például a jellemző, sokszor előforduló oldalletöltési szekvenciák, vagy az oldalak bizonyos sorrendiségének gyakorisága. Ezek fontos jellemzők lehetnek egy portál üzemeltetése során.

Modellek

Alapvetően 5 modellt fogunk használni, de ezek természetesen az igényeknek megfelelően bővíthetőek, illetve átalakíthatók. A legfontosabb modell, amelyet a többi modell elkészítésénél is felhasználunk a ***felhasználói útvonalakat*** adja meg. Ennél összegyűjtjük minden egyes látogatók útvonalait, és azok gyakoriságát vizsgáljuk. A kapott adatok fontos információval szolgálnak a portálon belüli mozgásokról. Megtudhatjuk például, hogy mely információk keresettek, kedveltek, illetve hogy egy adott helyre honnan érkezett a látogató, és merre távozott onnan.

A *„**gyakori látogatási útvonalak**”* modell a felhasználói útvonalakat aszerint vizsgálja, hogy az összes különböző bejárás hány felhasználónál található meg. Ennek a vizsgálatnak az eredményéből tehát azt tudhatjuk meg, hogy hány felhasználó járta végig weboldalaknak pontosan ugyanazt a sorozatát. Az „azonos weboldalak” modell a „gyakori látogatási útvonalak” egy egyszerűsített változata. A különbség mindössze annyi, hogy ennél a modellnél nem számít az egyes meglátogatott honlapok egymáshoz képesti sorrendje, vagyis az eredmény az azonos oldalakat meglátogató felhasználók számát adja meg.

A későbbi felhasználás szempontjából fontos információ lehet az is, hogy az egyes weboldalakra honnan érkeztek a látogatók és merre mentek tovább. Ez a modell a *„**be- és kilépő oldalak**”* nevet viseli. Ezekből az adatokból következtetni lehet az egyes weblapok szerepére. Ha egy adott honlapra a látogatók többsége egy vagy néhány másik weblapról érkezett (ezt mutatja a belépő statisztika), akkor az a két weboldal összevonható, vagy akár helyettesíthető is a harmadikkal, hiszen a többség végül úgyis oda érkezett meg. A kilépő statisztika ennek a másik irányból történő megközelítése: ha egy weblapról meg lehet mondani, hogy melyik másik weboldalra léptek tovább róla a látogatók, akkor tisztázható az adott honlap szerepe. A két módszer nem feltétlenül ad azonos eredményt, hiszen a statisztikáknak mindig van hibája, illetve a megközelítés irány is más ebben az esetben. Az *„**átkattintási, átlépési idők**”* modellje a felhasználói útvonalakban összegyűjtött adatokra támaszkodik. Segítségével előállíthatók az egyes felhasználók útvonalaira jellemző karakterisztikus váltási idők, vagyis azok az időtartamok, amelyek az egyes weboldalak lekérése között elteltek. Kiszámítható a felhasználók által összesen a portálon eltöltött idő, illetve az egyes oldalak közötti jellemző átkattintási idők. Az adott tartalom, illetve az azt birtokló vállalkozás ismeretében mindezek az adatok egészen érdekes információkat tudnak szolgáltatni mind műszaki, mind marketing szempontból.

A következő rész tartalmából

A következő cikkben részletesen ismertetem az egyes adatgyűjtési módszerek előnyeit és hátrányait, majd egy saját fejlesztésű szkripten keresztül bemutatom a modellek elkészítésének menetét és lehetőségeit.

Beszédes Balázs (beszedes@ei.hu)
24 éves, az e-Média Informatikánál mérnök-informatikus. Hobbija a kerékpározás és a kirándulás.

Kiadja: Kiskapu

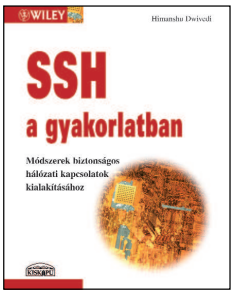
Felh. szint: haladó

Témakör: programozás, elmélet

ISBN: 9639301779

Ár: 5980,-

448 old.



Kiadja: Kiskapu

Felh. szint: kezdő-haladó

Témakör: hálózatok, biztonság

ISBN: 9639301817

Ár: 3850,-

384 old.

Erich Gamma, Ralph Johnson, Richard Helm, John Vlissides: Programtervezési minták

Ebben a könyvben egy régi és mégis új témáról van szó. Régi, hiszen néhány ötlet már évtizedekkel ezelőtt is felvetődött. Mégis újszerű, mivel egyrészt mostanában nyertek nagy teret azok a technológiák, ahol ezek a megoldások előnyösen alkalmazhatók, másrészt az utóbbi időben vált egyre divatosabbá az újrahasznosítás, a már kitalált ötletek, előre elkészített elemek felhasználásával történő programozás.

Na de mit is jelent az, hogy ***tervezési minta***? Megoldást egy olyan problémára, amely újra és újra felbukkan a mindennapi tervezési folyamatok között – például a programtervezésben. Hasonló, mint a megoldóképletek a matematikában, vagy sokat használt szerkezeti elemek az építőiparban. Egy ilyen mintát az alábbi módon mutatnak be a szerzők: Leírják a minta célját, egy a problémát bemutató forgatókönyvet, a lehetséges alkalmazási területeket, a megoldás szerkezetét egy objektummodell segítségével, az ebben résztvevő elemeket, ezek együttműködését, a megoldás alkalmazásának következményeit, s a megvalósítást kritikus pontjait – mindezt természetesen programnyelvtől függetlenül. Igen hasznos, hogy a szerzők a problémát és a megoldást egyaránt hétköznapi szoftverelemeken keresztül szemléltetik. Így az olvasó a lehető legrövidebb időn belül átláthatja, miért is lehet szükség az adott mintára. Ezek után konkrét programkódban láthatjuk minden minta használatát, illetve az is kiderül, a gyakorlatban az milyen jellegű szoftverekben szokott feltűnni.

A gyűjtemény 23 tervezési mintát tartalmaz a klasszikus besorolás, vagyis a minta célja és hatóköre szerint szerint csoportosítva. Így egy jól átgondolt, körültekintően felépített eszköztárhoz jutunk, melynek időtálló voltát számos minta „életkora” jelzi.

Az első fejezet részletesen bevezeti az olvasót a tervezési minták alapkérdéseibe, lehetőségeibe, használatába. Megtudhatjuk, hogy miért, miként és hogyan érdemes őket használni, mikor mit érdemes választani. A fenti elvek gyakorlati alkalmazását egy részletes esettanulmányon keresztül sajátíthatjuk el azután. A szerzők is jól tudták, hogy gyakorlat nélkül az elmélet nem sokat ér, ezért bőven kapunk belőle az olvasás folyamán.

Ez a könyv az első és egyetlen olyan magyar nyelvű írás, amely összegyűjtve tartalmazza a témában fellelhető összes anyagot.

Himanshu Dwivedi: SSH a gyakorlatban

„A Seucre Shell (SSH) olyan program, amely két fél között zajló kommunikációt tesz biztonságossá” - írja a szerző az első sorban. Ez egy meglehetősen tág meghatározás, a könyvet elolvasva azonban részletesen is kiderül, mit is takar pontosan ez az ipari szabvány.

Mint a címe is mutatja, a könyv elsősorban nem az elméleti alapokról szól. Azokról számtalan más írás született már, ráadásul pontos megértésükhöz helyenként egészen mély matematikai ismeretekre van szükségük. A szerző ehelyett kézenfogva vezet végig bennünket a konkrét megvalósítások telepítésétől a részletes beállításokon át az egyes felhasználási területekig. Bár a könyv minden egyes betűje konkrét példákhoz, testreszabási, beállítási lehetőségekhez kötődik, mégsem sorolható egyértelműen a HOGYAN-ok, vagy „szakácskönyvek” kategóriájába. Mindenütt részletesen ismerteti ugyanis a tárgyalt példa hatásait, következményeit, magyarázatát, átfogó gyakorlati ismeretekhez juttatva ezzel az olvasót.

A könyv három részre oszlik. Az első egyfajta alapozás, az SSH általános áttekintése. Olvashatunk itt az ügyfél és kiszolgáló megvalósításáról, a beállításokról, sőt azt is megtudhatjuk, hogyan érhetünk el Cisco útválasztókat SSH-val. A második részben megismerhetjük SSH sokoldalúságát biztosító kapuátirányítást, mégpedig minden lehetséges oldaláról. Megtudhatjuk hogyan használhatók biztonságosan az egyébként kódolatlan adatcsomagokkal dolgozó szolgáltatások (telnet,rsh,ftp,smtp,pop3,irc, stb), vagy hogyan építhetünk virtuális magánhálózatot (VPN) az SSH segítségével. A harmadik rész bemutatja, hogyan helyettesíthetők az egyes hálózati protokollok SSH-val, és hogyan tehetjük biztonságossá a proxy-kat. Végül három, a mindennapi életből vett esettanulmány segíti az elmondottak minél pontosabb megértését.

A könyv bővelkedik az olyan megoldásokban, melyeket magam is számtalanszor alkalmaztam már. A könyv grafikai kivitelezését és a szerkesztést tekintve is rendkívül szemléletes. Számtalan képernyőképet, parancssorozatot, magyarázó táblázatot találunk benne Windows és Linux/UNIX rendszerekhez egyaránt.

Ahhoz, hogy haszonnal forgathassuk ezt a könyv, szükségünk van bizonyos hálózati alapismeretekre, de a tárgyalás módja olyan, hogy az SSH-ban eddig teljesen járatlanok is nyugodtan megvehetik. Nem fognak csalódni.

A GIMP és a mintázatok

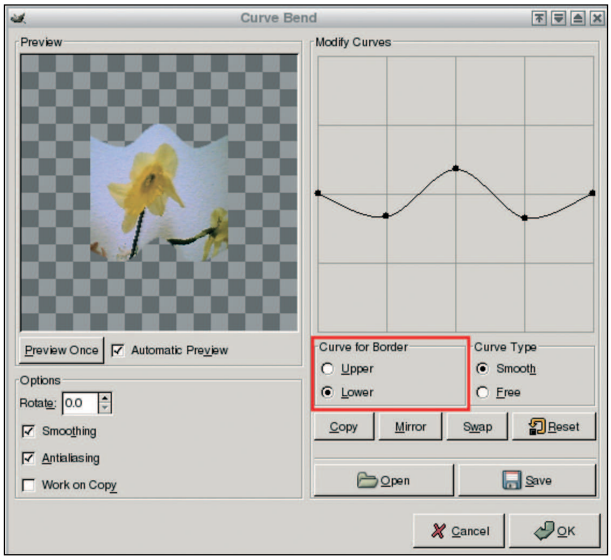
Sorozatunk utolsó előtti részében bemutatom a különféle képátalakító szűrőket, továbbá azokat, melyekkel a szerkesztett képtől független mintázatokat hozhatunk létre.

Most tehát kezdjük az ismerkedést azokkal a hatásokkal, melyeket a képablakban található menü ***Filters*** pontjának ***Distorts*** nevű almenüjében szerepelnek. Itt jegyzem meg, hogy van két olyan billentyűkombináció, amelyeket érdemes megjegyezni a szűrők alkalmazása során. Az egyik a CTRL+F, mellyel megismételhetjük az előző szűrési műveletet, a másik pedig a SHIFT+CTRL+F. Ez utóbbival a menüben való barangolást mellőzve újra kiválaszthatjuk az előző szűrőt és annak beállítóablakában változtathatjuk a paramétereket. Az új paraméterek megadása után alkalmazhatjuk ismét a szűrőt vagy elvethetjük a változtatásokat.

A ***Distorts*** menüben olyan hatásokat találhatunk, melyekkel nem egyszerűen a képpontokat manipuláljuk, hanem ezt a tevékenységet előre meghatározott céllal tesszük. Ilyen szűrő például az is, amellyel hullámossá tehetjük a kép síkját vagy mozaikokból kirakott képet készíthetünk a szerkesztett képből.

Válasszuk ki tehát a kép menüjében a ***Distorts*** menüpontot, majd a ***Blinds*** hatást. A ***Blinds*** alkalmazásával a képet elforgatható, hosszúkás síkdarabokra vághatjuk szét (mintha egy reluxára festenénk fel). A darabolás mértékét megadhatjuk a ***Num Segments*** csúszka segítségével, ennek legmagasabb értéke 10 lehet. A hatás beállító ablakában megváltoztathatjuk a síklapok elrendezését vagyis, hogy vízszintesen vagy függőlegesen vágjuk a képet és azt is, hogy milyen mértékben legyenek elforgatva az egyes síklapok. Az elforgatás mértékét a ***Displacement*** értékkel határozhatjuk meg. Amikor ez utóbbi értéket változtatjuk, könnyedén megfigyelhetjük a változásokat az előnézeti ablakban.

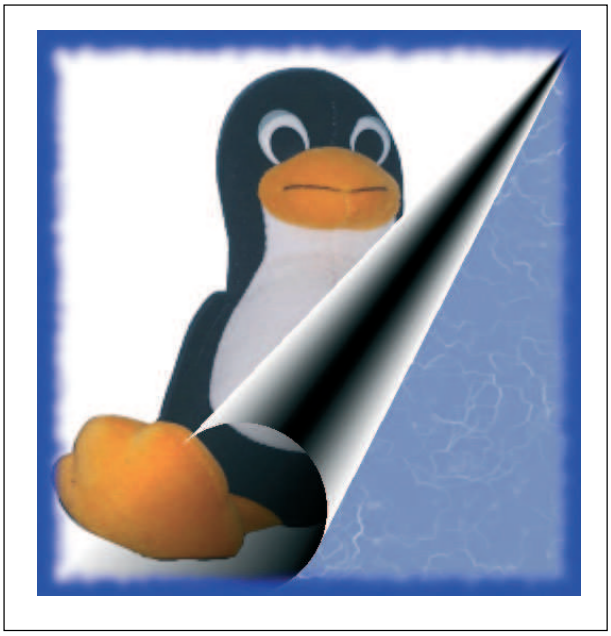
A ***Curve Bend*** hatással a kép széleit nyújthatjuk egy görbe segítségével. A hatás beállítását szolgáló ablakban az *1-es képen* pirossal keretezett területen állíthatjuk be, hogy a megadott görbe a kép alsó vagy a felső széléhez tartozik. A keretezett terület alatti gombok segítségével készíthetünk másolatot vagy tükrözött másolatot a megadott görbéről, amely így a kép másik széléhez tartozik majd. Az előnézeti kép alatti területen adhatjuk meg, hogy a két görbét milyen mértékben forgatjuk el. Értelemszerűen ha megadjuk az alsó- és a felső szélekhez tartozó görbéket, majd elforgatjuk



1. kép A 'Curve Bend' hatás

őket kilencven fokkal, akkor az eredmény a jobb- és bal szélekhez tartozik. A hatás megértéséhez tekintsük meg az 1-es képet, melyen látható, hogy a képet egy hullámos felületre feszítettük fel, így például szélben lebegő zászlót hozhatunk létre. A beállításainkat tárolhatjuk későbbi felhasználás céljából a ***Save*** gombra kattintva, majd később vissza is tölthetjük az ***Open*** gomb használatával.

Az ***Emboss*** hatás segítségével létrehozhatunk dombormű veket a megnyitott kép alapján. A dombormű lehet egyszerű szürke kőből kifaragva vagy lehet úgynevezett érdességi térkép. Ezt a beállítóablak jobb felső részén található ***Emboss*** és ***Bumpmap*** választómezők kiválasztásával adhatjuk meg. Mivel a domborműveket általában fényben látjuk, megadható a fényforrás helyzete is. Az ***Azimuth*** a fényforrás vízszintes helyzetét adja meg, a nulla fok a balról érkező fényt jelenti. A másik beállítási lehetőség a függőleges szög meghatározása. Értelemszerűen a 90 fokos függőleges szög jelenti a síkra merőlegesen érkező fénysugarakat. Az előnézeti ablakban ebben az esetben is nyomon követhetjük a beállítások változtatását, így mindenki eldöntheti, hogy



2. kép Az oldalszél felhajtása alatti réteg

3. kép Vastagítás és vékonyítás

számára milyen fényforrás a leginkább megfelelő. A ***Depth*** csúszkával állíthatjuk be, hogy milyen mértékben emelkedik ki a felület a síkból. A hatás alkalmazásakor a ***GIMP*** élkeresést végez, tehát a kiemelkedő területeket az élek mentén határozza meg.

A ***Mosaic*** hatás segítségével a képből mozaikszerű összeállítást készíthetünk. A beállító ablakban meghatározhatjuk az egyes darabkák méretét (***Tile Size***), a magasságukat (***Tile Height***) és a közöttük lévő hely nagyságát (***Tile Spacing***). Az egyes darabkák szabályosságát a ***Tile Neatness*** változó segítségével határozhatjuk meg, minél közelebb esik ez az érték a maximális 1-es értékhez, annál szabályosabbak lesznek az egyes mozaikdarabok. A mozaikot kirakhatjuk négyzetekből, hatszögekből vagy nyolcszögek és négyzetek váltakozásából.

A ***Pagecurl*** hatással a kép egyik sarkát felhajthatjuk, a ***GIMP*** pedig elvégzi a megfelelő árnyékolást a felhajtott oldalszél domborulatán és ha kiválasztjuk a ***Shade under Curl*** lehetőséget, akkor a felhajtott szél által árnyékolt területen is. Beállíthatjuk, hogy melyik oldal mentén hozzuk létre a felhajtást és azt is, hogy az milyen irányú legyen. Sajnos a hatás segítségével nem adható meg, hogy csak a sarkon legyen a hajtás, de más módszerekkel ezt is elérhetjük. A ***GIMP*** készítői gondoltak arra is, hogy a felhajtott réteg alatt mások lehetnek, tehát a hatás alkalmazása után az üres területek átlátszóak lesznek. Így láthatóvá válik, amint az a *2-es képen* is megfigyelhető.

A ***Newsprint*** hatással a többszínnyomással nyomtatott újságok színes képein látható tintafoltokat utánozhatjuk. A ***Cell Size*** értékkel határozhatjuk meg az egyes tintafoltok méretét, míg a különböző színösszetevőket tartalmazó rétegek egymáshoz viszonyított elforgatását a beállítóablak középső részén található értékekkel adhatjuk meg.

A ***Polar Coords*** hatással a képpontok koordinátái alapján polárkoordináta rendszerre képezhetjük le a szerkesztett képet.

A ***Ripple*** hatás egy egyenes vonalban terjedő hullám segítségével átalakítja a kép síkját a hullámos felületté. A beállítások között megadhatjuk a hullámhosszt és az amplitúdót, és a fázis változtatásával animációs képkockákat is készíthetünk.

A ***Waves*** hatás az előzőhöz hasonlóan hullámos felületté alakítja a síkot, de itt koncentrikus körökként jelennek meg a hullámok. Ennél a hatásnál megadhatjuk, hogy a felület tükröződő legyen, vagyis az egyes hullámokon tükröződjön a környezetük.

A ***Shift*** hatás használatával kép sorait eltolhatjuk a megadott mértékben jobbra és balra vagy felfelé és lefelé. Így létrehozhatunk például zajos képet, mint amikor nem tartjuk megfelelő stabilitással a fényképezőgépet és a kép kissé elmosódott lesz.

A ***Value Propagate*** hatás egy egyszerű matematikai műveleten alapul, melyet a digitális képfeldolgozásban is gyakran használunk. A képpontok környezetéből kiválasztjuk a maximális értéket (***Propagate Mode – More Black***) vagy a legkisebb értéket (***Propagate Mode – More White***). Így elérhetjük, hogy a képen szereplő kis felületű részletek nagyobbak legyenek vagy még kisebbekké váljanak. A digitális képfeldolgozásban ezt a műveletet vastagításnak és vékonyításnak nevezzük. A beállítások ablakában a jobb alsó részen választhatjuk ki azokat a szomszédos képpontokat, melyeket a szűrő alkalmazásakor a ***GIMP*** figyelembe vesz. Amennyiben csak egy irányhoz tartozó szomszédokat adunk meg (például fent és lent) akkor a hatás csak ebben a meghatározott irányban érvényesül.

A ***Video*** hatás alkalmas arra, hogy például a számítógéppel számított képeinket valósághűvé tegyük. Köztudott, hogy az analóg videólejátszó rendszerek zajos működésűek. Hiába számítjuk ki digitális módon, tökéletes pontossággal az animáció képkockáit, egy idő után az analóg tárolóeszközökön (pl. videokazetta) zajossá válnak a felvételek. Ezt a hatást utánozhatjuk az említett szűrővel, mégpedig többféle módon is. Meghatározhatunk függőleges csíkozást, ferde

4. kép A kész filmszalag

csíkozást és még számtalan más zajhatást is készíthetünk a ***Video*** szűrő segítségével. A ***Whirl & Pinch*** szűrővel a képet középpontja körül, a síkjára merőleges tengely mentén forgathatjuk, csavarhatjuk. Animációs lehetőségként használva ezt a szűrőt, létrehozhatunk például jobbra-balra dőlő, csavarodó (táncoló) alakokat.

A ***Wind*** szűrővel, mint a neve is mutatja szélfúvás hatását utánozhatjuk. A beállítóablak bal szélén meghatározható, hogy szélről vagy valami erőteljesebb hatásról szeretnénk képet alkotni. A szél-hatás tulajdonképpen egy elmosást jelent a megadott irányba, míg az erőteljesebb hatás esetén a képsorok eltolását láthatjuk.

Tehát ez a néhány hatás alkalmas különféle látványos képek létrehozására, de természetesen a ***GIMP*** maga is képes különféle képek és mintázatok előállítására a megfelelő beépülő modulok telepítése után. Szerencsére nem kell sokat bajlódni a modulok telepítésével, hiszen napjainkban a ***GIMP*** csomag részét képezik a következő bekezdésekben bemutatásra kerülő hatások és képalkotásra alkalmas programrészek. Tekintsük át a fontosabb modulokat a szerkesztett kép menüjében a ***Filters – Render*** menüpontokat kiválasztva.

A ***Clouds*** almenüben kétféle felhőzet készítésére alkalmas menüpontot találunk. Az első a plazma, melyről az előző számban bővebb információk található, a második pedig a ***Solid Noise*** változat, amely egy egyszerűbb algoritmus használatával éri el a felhőzet vagy domborzat készítésére alkalmas kép előállítását.

A ***Pattern*** almenü szintén többféle kép elkészítésére ad lehetőséget. A ***Checkerboard*** segítségével létrehozhatunk egyszerű négyzetrácsos mintázatot, amely alapállapotban fekete-fehér négyzetek váltakozásából áll. A kép elkészítése után a már ismertetett módszerekkel színezetet adhatunk a mintázatnak. Ilyen módszer lehet például a szín szerinti kiválasztás után alkalmazott festővödör, amivel kifestjük az egyik színt, majd a kiválasztás megfordításával és másik színnel történő kitöltéssel a másik színt is átalakítjuk.

A ***Grid*** hatás segítségével szabályos rács-vonalat rajzolhatunk a képre vagy a kiválasztott rétegre.

A ***Jigsaw*** szűrő alkalmas például arra, hogy gyermekünknek meglepetést szerezzünk, hiszen akár kedvenc képét is kirakósjátékká alakíthatjuk. A szűrő beállítására szolgáló párbeszédablakban megadhatjuk, hogy vízszintesen és függőlegesen mennyi részre daraboljuk a képet, megválaszthatjuk, hogy az egyes darabok szabályos négyszögből alakuljanak-e ki vagy lekerekített négyszögletes alakzatokból és meghatározhatjuk azt is, hogy milyen mértékű legyen a lekerekítést. Ez utóbbi két hatás alkalmazása előtt érdemes új réteget létrehozni a szerkesztés alatt álló képen, mert a szűrők az aktív rétegen dolgoznak és így felülírják annak tartalmát. Szerencsére a ***GIMP*** alkotói itt is gondoltak az átlátszóságra, tehát a rács-vonalak és a kirakósjáték darabok nem takarják el a képet.

A ***Maze*** hatás segítségével labirintusokat készíthetünk. Amikor készül legújabb játékunk még nem feltétlenül kell napokat tölteni a tényleges labirintus megtervezésével és elkészítésével, alkalmazhatjuk akár a ***GIMP*** említett funkcióját és a gyorsan elkészített labirintuson tesztelhetjük a programunk grafikai megjelenítését vagy irányíthatóságát.

A ***Qbist*** különféle kubista stílusban készült képek önműködő alkotására használható. Ismét felmerül ugyan a kérdés, hogy mi a megfeleltetési lehetőség az alkotás és a gombokra kattintgatás között, és ismét nem tudom megfogalmazni, mindenesetre a ***GIMP*** programozói gondoltak a könnyű 'alkotás' lehetőségére. Talán gyakorlati haszna olyan esetekben lehet ennek a hatásnak, amikor valamilyen mintázatot készítünk és kiindulásul a kubista alkotásokat választjuk.

A ***Sinus*** szűrővel különféle szinusz függvények alapján előállítható mintázatokat készíthetünk. Látványos hatásokat érhetünk el, például létrehozhatunk egyszerű domborzati térképet vagy vízfelülethez a három dimenziós modellező-programhoz tükröződési térképet.

Igen érdekes lehetőséget találunk a szűrők között, a ***Combine*** almenüben. Töltsünk be tetszőleges számú, azonos méretű képet és válasszuk ki a ***Film...*** menüpontot. A menüpont kiválasztásakor megjelenik egy párbeszédablak, melyben kiválaszthatjuk a film elkészítéséhez felhasználandó képkockákat, meghatározhatjuk, milyen betűkészlettel kívánjuk a feliratokat elkészíteni a filmszalagon és az ***Advanced*** fülre kattintva megadhatjuk a filmszalag szélén elhelyezkedő lyukak méretét és egymás közötti távolságát is. Amikor minden beállítást megadtunk létrehozhatjuk első „digitális filmünket". Gyakorlati haszna akkor lehet ennek a hatásnak, amikor például egy animáció képkockáit szeretnénk egyszerűen és pontosan egymás mellé elhelyezni.

Korábbi ismereteink alapján eddig ha a ***GIMP***-pel dolgoztunk, létrehoztunk egy megfelelő méretű üres képet és a másolás-beillesztés műveletekkel szépen elhelyeztük az egyes képeket. Most azonban elegendő megadni a megfelelő képeket, felvenni őket a listánkba majd a ***GIMP*** elkészíti a filmszalagot. A felesleges részek (például a lyukak és a szalag széle) levágása után tároljuk a szalagot és könnyedén felhasználhatjuk játékunkban vagy más területen.

A sorozat utolsó előtti részében tehát befejeztük a fontosabb szűrők tárgyalását, idáig pedig aki nyomon követte a sorozat egyes darabjait, képet kaphatott a ***GIMP*** kiterjedt képességeiről és reményeim szerint munkája vagy otthoni elfoglaltsága során sikerrel alkalmazta az ismereteket.

Az utolsó rész egy összetett feladat teljes megoldása lesz, nevezetesen könyvborítót készítünk a ***GIMP*** segítségével. Addig azonban kellemes alkotást kívánva búcsúzom és elektronikus levélben továbbra is legjobb tudásom szerint szívesen válaszolok a felmerülő kérdésekre.

Fábián Zoltán (dzooli@freemail.hu)
26 éves, jelenleg oktatóként dolgozik, szabadidejében szívesen foglalkozik Blenderrel, programozással és elektronikai tervezéssel. Szereti a természetet, túrázást, úszást és a kellemes baráti társaságot.

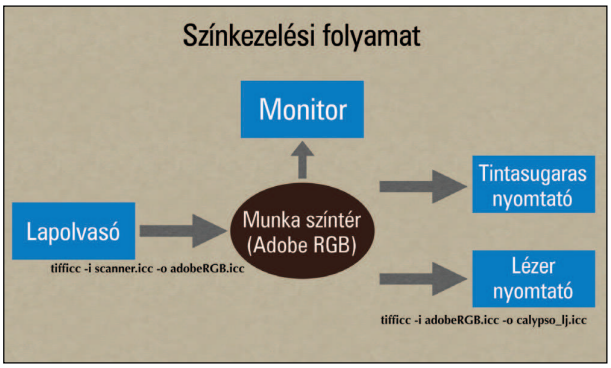
Linux eszközök a hivatásos fotográfiában

Ma már léteznek olyan eszközök, amelyekkel hivatásos fotográfiai munkákat végezhetünk el közvetlenül a Linux gépünkön.

A jó minőségű, olcsó termékek megjelenése forradalmasította a fotóipart, a ***Linux*** eszközeivel pedig ki is használhatjuk ezeket a termékeket. Az 1990-es években, amikor első digitális képemet szerkesztettem egy ***Mac SE*** gépen, titkon azt reméltem, hogy egy napon a felső kategóriás képkezelő munkaállomások valamennyi képessége az én digitális stúdiómban is elérhető lesz. Néhány évvel később már megvolt a ***Mac*** stúdióm, amiről álmodtam de lapolvasás és nyomtatás tekintetében még mindig külső cégektől függtem.
1996 körül megláttam a ***GIMP*** egy korai változatát és felvillanyozott a gondolat, hogy a stúdiómat ***Linux*** alá is átvihetem. Gyors ugrás a jelenbe: ma már kaphatóak olyan olcsó lapolvasók és fotónyomtatók amelyek elérhető áron profi minőséget nyújtanak, és léteznek az ezeket kezelő Linux meghajtók és eszközök is. A színkezelés munkafolyamata is kezd lassan alakot ölteni ***Linux*** alatt és szerencsére a ***GIMP*** is egyenletes iramban fejlődik.
Ma már a beszkennelt diától a digitális módosításokon keresztül eljuthatok a végtermékig úgy, hogy végig a ***Linux*** alapú stúdiómat használom. Ebben a cikkben egy kép útját végigkövetve bemutatom, hogyan készíthetünk professzionális minőségű digitális képeket ***Linux*** alatt. Ahelyett, hogy elmagyaráznám az egyes csomagok fordításának és telepítésének részleteit, inkább madártávlatból szeretném bemutatni, hogyan illeszkednek egymáshoz a különböző alkotóelemek és csak hivatkozásokat adok a részletesebb leírásokhoz.

Alkatrészigény

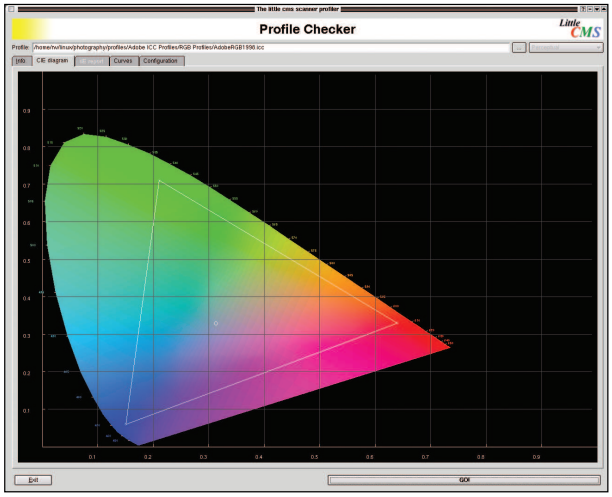
Nem sok különbség van egy jó linuxos asztali gép és az általam fotószerkesztésre használt munkaállomás között. Természetesen minél több memóriánk van, annál jobb. Legalább 512MB memória beszerzése mindenképpen javasolt és egy gyors merevlemezzel is sok időt takaríthatunk meg. A fotós munkaállomásom videókártyájához két monitor csatlakoztatható, 1.7GHz P4 processzor van benne, valamint 1GB RAM és három 36GB-os, 10000 RPM SCSI merevlemez. Ami igazán megkülönböztet egy fotószerkesztő összeállítást egy szokásos asztali géptől az a perifériák csatlakoztathatósága. Ma már kiváló minőségű lapolvasókat és nyomtatókat lehet vásárolni 1,000 dollár alatt. Legtöbbjük ***Linux*** alatt is támogatott, de mindig ellenőrizzük a ***Linux*** kompatibilitási lapokat, mielőtt megvásárolnánk valamelyik speciális eszközt. Ha lapolvasót keresünk, érdemes a hálózati fotó fórumokra, például a Photo.net-re is bekukkantani, valamint megnézni a ***VueScan*** és ***SANE*** weblapjain található információkat. Mivel általában nagy formátumú, 4″ × 5″ filmre dolgozom, nálam kizárólag a filmadapterrel kiegészített síkágyas lapolvasók jöhetnek szóba. A 35mm-es filmek olvasásához a kifejezetten erre a célra készített lapolvasók jobb minőséget biztosítanak mint a síkágyasok.
A fotónyomtatók terén nehéz megelőzni az ***Epson*** tintasugaras típusok új generációját. A ***LinuxPrinting.org*** lapon találnunk egy nagyszerű eszközt, melynek segítségével megállapíthatjuk, hogy az adott nyomtató rendelkezik-e ***Linux*** támogatással, sőt még a legjobb meghajtóra is javaslatot tesz nekünk.



1. ábra Linux alatti színkezelés munkafolyamat ábrája

Színkommunikáció

A kommunikációhoz nem kell más mint egy közös szótár, amelyet mindkét fél megért. A színkommunikációban ezt a közös nyelvet az ***International Color Consortium*** (***ICC***) határozza meg. Minden színes eszköz saját színtérrel rendelkezik, ami gyakorlatilag azt jelenti, hogy mindegyik csak egy bizonyos tartományból képes a színeket előállítani. Ezeket hívják eszközfüggő színeknek. A különféle színterek közötti átalakításokhoz, eszközfüggetlen színtérre van szükségünk. Az ***ICC*** által használt színleírás neve ***CIE 1931 standard colorimetric observer*** amely bármely színes eszköz színterét pontosan meghatározza. Az ***ICC profil*** állomá-



2. ábra Színteret megjelenítő CIE diagram.
A háromszög az AdobeRGB 1998 munkatér színterét jelzi.

3. ábra Monitor beállító képek: bal oldalon Norman Koren monitor gamma diagrammja, jobb oldalon a Photodisc színkalibrációs képe látható. Ezeket a nyomtatott példákat ne használjuk fel, hiszen színük megváltozhatott a sajtó nyomtatási egyenetlenségei miatt.

nyokban találjuk azokat az információkat, amelyek segítségével az adott eszköz színterét ebbe az eszközfüggetlen színtérbe vihetjük át. A kép szerkesztése során egy különleges, szintén eszközfüggetlen színteret, az úgynevezett munkaszínteret használunk. Így van egy közös nyelvünk és minden ettől különböző nyelvhez egy tolmácsunk, s végső soron bármilyen kommunikáció megoldható. Bár nem árt tudni a színelméletet, mindez nem szükséges a jó színnyomatok készítéséhez. Mindössze annyit kell tudnunk, hogy ha az egyik eszközről a másikra szeretnénk színhelyesen áttérni, akkor az ***ICC profil*** állományokat kell használnuk.

Monitorbeállítás

A pontos monitor rendkívül fontos, ha színes fotókkal dolgozunk. A hivatásosok egy koloriméternek nevezett eszközzel mérik a számítógép tényleges képét, majd elkészítik a megfelelő ***ICC profil*** állományt, amely alapján aztán az operációs rendszer helyesen elő tudja állítani a megjeleníteni kívánt színeket. Sajnos azt tapasztaltam, hogy egyetlen ***Linux*** meghajtó sem támogatja az ilyen eszközöket. Még ha sikerül is hozzájutnunk a monitor profil állományához, a ***GIMP*** megjelenítéskor nem támogatja a használatát. Természetesen ez az egyik olyan szolgáltatás, amit a ***GIMP***-ből nagyon hiányolok. A ***GIMP bug #78265*** számú hiba le is írja a problémát; így remélhetőleg ez a funkció is hamarosan elkészül. Addig is a legjobb, amit tehetünk az, hogy a legtöbb minőségi monitorba beépített vezérlők segítségével kézzel figyelmesen módosítjuk a monitorbeállításainkat:

1. Állítsuk a monitor fehér pontját 6,500K értékre. Ezzel több sárgát kapunk mint a legtöbb monitor alapbeállításaként szereplő 9,300K értéknél, viszont szemünknek sokkal könnyebb meglátni a helyes színt. Pontosan emiatt ez az ipari szabvány javaslata is.

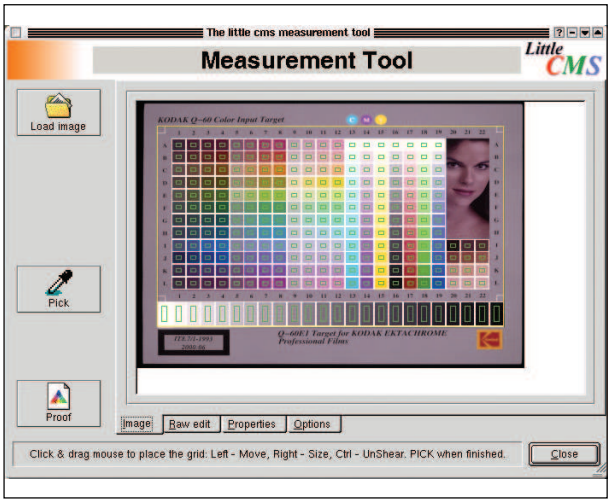
2. Emeljük meg a monitor kontraszt szintjét 100%-ra és addig tekerjük a fényerőt, amíg ***Norman Koren*** *3. ábrán* látható kiváló színbeállító tábláján a legtöbb ***B Black Level*** csík fel nem tűnik.

3. Módosítsuk a megjelenítő gamma szintjét a 3. ábra gamma csíkjával. A rajzot úgy nézzük, hogy szemünkkel addig bandzsítunk, míg már nem tudjuk megkülönböztetni a vízszintes vonalakat majd keressük meg azt a helyet, ahol a szürke tónus a teljes csíkon azonos. Ez a jelenlegi gamma szintünk. Az ***xgamma*** segítségéve módosítsuk a gamma szintet 2.2-re, azaz az ipari szabványra.

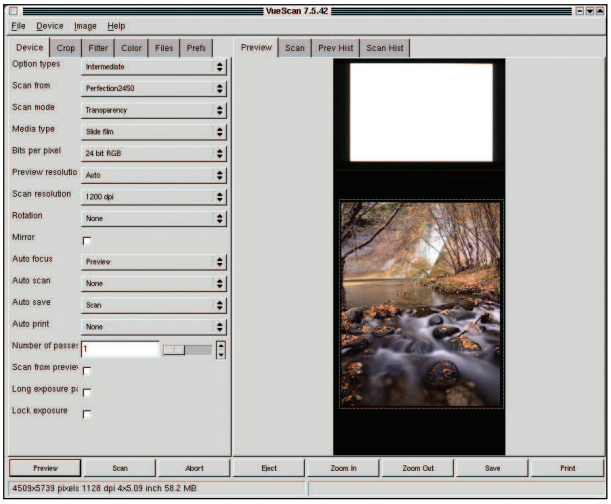
Most az R, G és B vezérlőkkel állítsuk a cél szürke színét olyan semlegesre, amennyire csak lehetséges. A monitor színegyensúlyának beállításához használjuk a 3. ábrán látható színbeállító kártyát. A bőrszínekre és természetes szürke tónusokra koncentrálva az összes komolyabb színegyensúlyi hibát kiküszöbölhetjük.
A jól beállított monitoron kívül, az egyenletesen megvilágított környezet is feltétlenül szükséges a megfelelően pontos színlátáshoz. A legjobb a fakó fény, este pedig az, ha szobánkat 5,000 K nappalifény-egyensúlyozott kompakt fénycsővel világítjuk meg. Jó megvilágításnál egy papírdarab fehér színe egészen közel kell álljon a monitorunk fehér színéhez. Az ***ISO 12646*** szabvány igen részletesen taglalja a megfelelő környezeti megvilágítást. A környezetünkben lévő ragyogó színek megváltoztatják a színérzetünket, ezért tartózkodjunk a közeli erősen megvilágított csillogó falaktól, a fényes hátterektől és a ***Linux*** környezet színes ablakaitól. Én egyszerű szürke hátteret használok és fekete-fehér ablakdíszítést.

A lapolvasó beállítása

Most már van egy viszonylag jól beállított monitorunk és helyesen megvilágított környezetünk, így figyelmünket a lapolvasó felé fordíthatjuk. A ***SANE*** és a ***VueScan*** egyaránt kiváló ***Linuxos*** megoldás a lapolvasáshoz. Én főleg a ***VueScan*** rendszert használom mivel ugyanez létezik a Mac-emre is. Bármelyik lapolvasó programot használjuk,



4. ábra Lapolvasó profil készítése a beolvasott Kodak cél segítségével



5. ábra Kezdeti kép beolvasása a VueScannel

nagyon fontos, hogy állandó programbeállításokat alkalmazzunk. Ez a helyes színérték szempontjából lényeges, hiszen az ***ICC profil*** amit készíteni akarunk csak a készítésekor használt színbeállítások esetén ad helyes eredményt.
A lapolvasó profil elkészítéséhez egy ismert minőségű anyagra van szükségünk, amit ***színbeállítási célnak*** nevezünk. Én a ***Kodak*** cég ***Q-60***-as ***Color Input Target*** termékét használom, amit a legtöbb felsőkategóriás fotóboltban megvásárolhatunk. Olvassuk be a képet a kedvenc programunkkal; ne feledjük el lementeni a beállításainkat és később mindig ugyanazokat a beállításokat használjuk a lapolvasáshoz.
Az olvasott anyagból az ***LProf*** profilkészítő eszközzel készíthetünk ***ICC*** állományt. A programot a ***Little CMS*** weblapról szerezhetjük be (lásd a hálózati forrásokat). Futtassuk a ***kmeasurementool*** programot és és nyissuk meg a beolvasott anyagot. Lépjünk az ***Options*** fülre és válasszuk ki a megfelelő színsablont, jelen esetben a ***Kodak column picker***-t. Lépjünk vissza a kép fülre és méretezzük újra a zöld cél körvonalat. Amikor a kis kockák a kép színfoltjai köré illeszkednek, válasszuk a ***Pick***-et majd mentsük az ***IT8*** lapot. Ezt követően a ***kscannerprofiler*** programmal készítsük el saját ***ICC*** profilunkat a lapolvasónkhoz megadva a megfelelő forgalmazót és köteg azonosítót. Minden cél a köteget azonosító dátummal rendelkezik. Nyissuk meg a korábban létrehozott ***IT8*** lapot és adjuk meg a kimeneti nevet az ***ICC*** profil állományunkhoz és bökjünk a ***GO***-ra. Ezzel az ***ICC*** profillal most már fel értelmezni tudjuk a lapolvasónk színterét.
Ha a lapolvasónkat beállítottuk, végre nekiláthatunk a filmek beolvasásának. Amikor képeket viszünk be, fontos, hogy ismerjük a lapolvasó valódi optikai felbontását. A legnagyobb optikai felbontást használva a lehető legtöbb részletet olvassuk be anélkül, hogy a a fájlt túl nagyra hizlalnánk. A fájlok mentéséhez legjobb ha a ***TIFF*** veszteségmentes formátumot használjuk.
A film beolvasása után az eredményül kapott ***TIFF*** állomány színtere a lapolvasó színterével egyezik meg. Ezt a színteret a munkaszíntérré kell átalakítani ahol a szerkesztéseket végezzük. Népszerű munkaszíntér az ***Adobe 1998***, amelyet az ***Adobe*** weblapjáról szerezhetünk be. Alkalmazzuk a lapolvasó profilt a ***Little CMS*** csomag részeként kapott ***tifficc***-vel, és adjuk ki a

```
tifficc -i scanner.icc -o Adobe1998.icc scan.tiff
➥ scan_munka.tiff
```

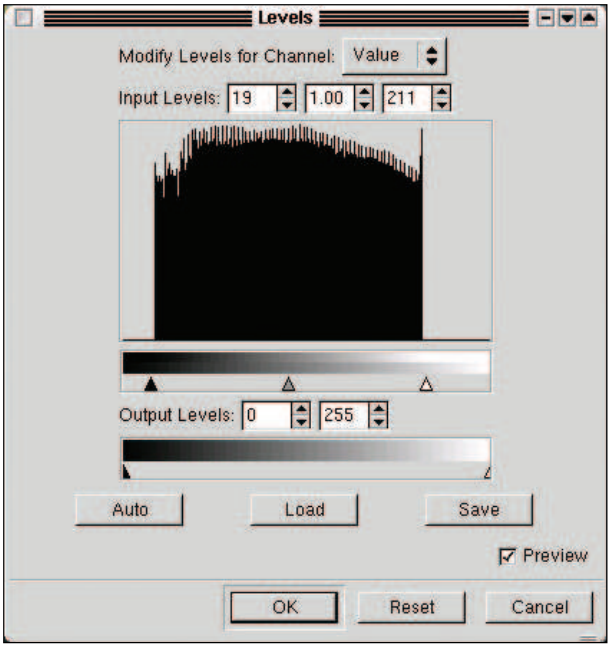
parancsot, amely ***TIFF*** képünket a munkaszínterünkbe alakítja át.

Fotóeszközök a GIMP-ben

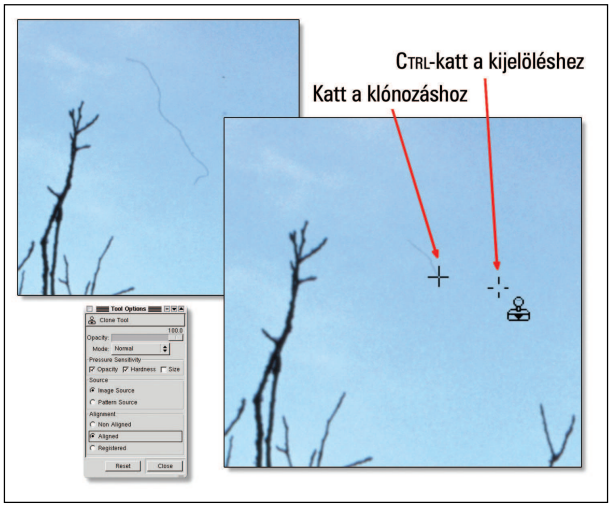
A ***GIMP***-ben, akárcsak más képszerkesztő programokban igen sok eszköz található, amelyre nincs szükség fotográfiai munkák során. A legtöbb időt a ***Szintek és Színgörbék*** eszközökkel szoktam eltölteni. Az első módosítás amit elvégzek a teljes expozíció beállítása, amire a legjobb eszköz a ***Szintek***. Soha ne használjuk a ***Fényerő/kontraszt*** eszközt, mert úgy tűnik, sokszor levágja az értékek végét. Nyissuk meg a ***Szintek*** eszközt és húzzuk a sötét- és fehérpontot addig ahol már adatot (függőleges tengely) látunk. A középső csúszkával világosítsuk vagy sötétítsük a képet.
A por és karcolások eltávolítása a következő fontos lépés, erre a feladatra a ***Másoló*** eszköz a legjobb. Nagyítsunk a képbe egészen addig, míg a porfolt könnyedén látható a képernyőn és figyelmesen nézzük végig a teljes képet. Ha meglátunk egy porfoltot, válasszuk ki a ***Másoló*** eszközt valamilyen lágy szélű ecsettel amely valamivel nagyobb mint a porfolt. Állítsuk a Másoló eszköz tulajdonságait 100% átlátszatlanságra és válasszuk a regisztrált igazítást. Ctrl+kattintással bökjünk egy hasonló szomszédos területre a másoláshoz, engedjük fel a Ctrl billentyűt és egyenesen kattintsunk a folt tetejére. Ha átlátszatlan ecsetet használtunk, a folt teljes egészében el kell tűnjön. A karcolások eltávolítása hasonlóképpen történik, de ismétlődő mintát kell keresnünk. Egy kis gyakorlással nagy területeket másolhatunk át árulkodó nyomok nélkül.

A művészi nyomat

Készen állunk a nyomtatásra. Részemről a nyomtatvány lesz a művészi alkotás végső fázisa, és igen sok időt töltök



6. ábra Expozíció módosítása a Szintek segítségével a GIMP-ben



7. ábra Por és karcolások eltávolítása a Másoló eszközzel

vele, hogy a kép tényleg mondjon valamit a nézőnek. Gyakran 10–20 alkalommal is kinyomtatom a képeimet, apró módosításokat végezve rajtuk, míg végül elégedett vagyok az eredménnyel. Ezt a tintasugaras nyomtatványt nevezem művészi nyomatnak. A képet nyomtatáshoz előkészítve végre kell hajtanunk a nyomtatási mérettel kapcsolatos néhány módosítást. Ezek a módosítások állagrontóak, ezért mielőtt nekikezdünk, képünkből mindenképpen készítsünk egy mesterállományt. Először is a ***GIMP Kép*** menüjének ***Kép átméretezése*** pontjára kattintva méretezzük át a képet. Tintasugarasokhoz a 200 PPI felbontást javaslom a kívánt nyomtatási méretben. Ezután élesítsük a képet a ***Maszk életlenítése*** eszköz segítségével. Tekintsük az esz-

közre úgy mint a fotó éleinek kiemelésére. Amennyiben valaki csodálkozna rajta, miért nevezik életlenítésnek, a név a hagyományos fotográfiai eljárásból származik ahol egy elmosott (vagy életlenített) negatív másolatot használtak az élek kiemelésére és kaptak élesebb képet. Válasszuk a ***Szűrők->Kiemelés->Maszk életlenítése*** menüpontot ahol a dialógusban három csúszka jelenik meg. A ***Sugár*** érték szabja meg, hogy az élpixelektől milyen messze dolgozunk, a ***Mérték*** adja meg, mennyire változhatnak meg a pixelek, végül a ***Küszöbszint*** dönti el, mely éleket módosítjuk. Amikor azt döntjük el, mekkora élesítést alkalmazunk, nézzük a képet 100%-on és próbáljunk ki néhány beállítást. Sajnos a ***Maszk életelenítése*** eszköznek egyelőre nincs előnézete, így kénytelenek vagyunk érvényesíteni a beállításokat, visszavonni és ismételni míg végül a kívánt eredményhez nem jutunk. Általában azt a beállítást keresem, ahol az élesítés éppen érzékelhető a képernyőmön. Legtöbbször a ***Sugár***: 1.0, ***Küszöbszint***: 3 és ***Mérték***: 0,75 értékekkel kezdem a próbálkozást.

Az ***Epson*** tintasugaras nyomtatóra nyomtatáshoz a ***CUPS*** és ***Gimp-Print*** párost használom. Útmutatásért keressük fel a ***Gimp-Print*** weblapot melynek alapján beindíthatjuk a rendszert. A ***Gimp-Print*** használatával az összes ***Epson*** jellemző elérhető a ***Print*** menüből. Ide tartozik az ***Epson escputil*** nyomtatóeszköz, amely lehetővé teszi a fejtisztítást és nyomtatási pozicionálást. Ideális esetben most a nyomtatónkhoz is elkészíthetnénk az ***ICC*** profil állományt; Sajnos azonban egyelőre nem létezik ***Linux*** alatt olyan program, amely képes lenne ilyesmire. Ehelyett azt javaslom végezzük el az ilyenkor szokásos módosításokat a ***Gimp-Print*** színmódosítás képességével. A módosításhoz használjunk olyan színteszt képet mint amilyeneket a képernyő beállításunknál is alkalmaztunk. Nyomtassuk ki a képet, és figyelmesen nézzük át az eredményt. Látnunk kell az árnyékokban a részleteket és a lépcsőket a szürke-skála csíkban; a semleges és bőrtónusoknál nem szabad színeltérést tapasztalnunk. Amennyiben nem ez a helyzet, nyissuk meg a ***Nyomtatás*** űrlapot, válasszuk a ***Kép/kivitel beállításai->Kimenet Beállítása*** pontot és végezzük el a szükséges módosításokat. Ismételjük a műveletet míg a kimenet megfelelő nem lesz. Ha sikerült találnunk egy jó beállítást, írjuk le őket, hogy későbbi nyomtatásokhoz jó kiindulási pontként szolgáljanak.

Végül eljött az igazság pillanata – vajon a végleges anyag megegyezik azzal amit képernyőn látunk? Nyomtassuk ki a képet és hasonlítsuk össze a képernyőn lévővel. Nagyon hasonlónak kell lenniük, bár azért alapvető különbség van a kép és a képernyő között. Tanulmányozzuk a képet és nézzük meg hol végezhetnénk módosításokat, majd végezzük is el őket a ***GIMP***-ben, míg végül elégedettek nem leszünk az eredménnyel.

Együttműködés egy külső szolgáltatóval

Eladható nyomatok készítésekor a ***Lightjet*** nyomatokat kedvelem, amelyeket néhány szolgáltató készít ***Cymbolic Sciences Lightjet*** nyomtatókon. Ez a nyomtató lézer segítségével vetíti a képet hagyományos fotópapírra, legtöbbször ***Fuji Crystal Archive*** anyagra. A ***Wilhelm Imaging Research*** tesztek (fotónyomat élettartam hitelesítő szervezet) szerint

8. ábra A Gimp-Print színmódosító ablaka

ezek a nyomatok 60 évig megőrzik kinézetüket egészen kis színváltozással és fakulással, ami lényegesen hosszabb idő mire a legtöbb tintasugaras nyomtató képes. Ezek a képek megkülönböztethetetlenek a hagyományos fotóktól és akár 4 × 5 láb (122x152 cm) méretűek is lehetnek.

Az állomány előkészítése a ***Lightjet***-hez hasonló a tintasugaras nyomtatáshoz. Kezdjük az eredetileg elmentett képpel, ne a tintasugarashoz átalakított változatot használjuk. Méretezzük át a képet a kíván fizikai méretnek megfelelően, ez esetben 200 PPI helyett használjunk 300 PPI értéket, élesítsük a képet a korábban leírt módszerrel és mentsük el egy másik ***TIFF*** állományba. Én a ***Calypso Imaging*** szolgáltatót kedvelem, amely a fotográfusoknak az USA egész területén nyújt ***Lightjet*** nyomtatási szolgáltatást. Ráadásul kedvezményt is adnak ha a képet megfelelően átméretezzük és a honlapjukról letölthető helyes ***ICC*** profilt alkalmazzuk a képen. Töltsük le a legfrissebb ***ICC*** profilt és alkalmazzuk a képre a ***tifficc*** segítségével. A kép most már készen áll rá, hogy CD-ROM-ra írjuk, vagy feltöltsük a ***Calypso*** FTP kiszolgálójára.

Néhány nappal azután, hogy megkaptam a végső nyomatot, visszagondoltam az egész folyamatra amellyel elkészítettem és azon gondolkodtam, hogy bár az eszközeim megváltoztak a cél ugyanaz maradt – olyan nyomatot készíteni amely az érzéseimet tükrözik egy kiválasztott témával kapcsolatban. Végre a teljes folyamatot végig tudtam vinni a ***Linux*** és elérhető árú asztali termékek segítségével, saját digitális stúdiómban. Igaz ugyan, hogy néhány területen még szükség van fejlesztésekre de a ***Linux*** már készen áll a feladatra. Tehát gyerünk, készítsünk néhány képet és alkossunk nagyszerű fotónyomatokat a linuxos eszközök kipróbálásával.

Linux Journal 2004. október, 126. szám

Amikor ***R. W. Hawkins*** éppen nem cégeknek segít Linux kiszolgálókkal boldogulni, délnyugat kanyonjaiban sátorozik és túrázik 4 × 5-ös kamerájával. A Szilíciumvölgyben él feleségével aki a tábori kajával szemben a pizzát részesíti előnyben. Weblapján, a rwhawkins.com-on, művészi nyomatok képtárát és digitális képkészítéssel kapcsolatos dokumentációkat találunk.

FreeBSD – A szomszéd vár (1. rész)

A FreeBSD rövid története, alapvető szerkezete és telepítése.

Sokan tekintenek idegenkedve a különféle ***BSD*** rendszerekre. Én is így voltam ezzel hosszú éveken át. Aztán egyszer nagy levegőt vettem és belevetettem magam a ***FreeBSD*** megismerésébe. Akkoriban csukott szemmel percek alatt bárhova fel tudtam telepíteni ***Linux*** alapú rendszereket. A ***FreeBSD*** többszöri – végül már valóban működőképes – telepítése viszont több hétig eltartott. Persze ebben nagy szerepe volt a hosszú évek alatt „idegpályáimba égett" ***Linux***-reflexeknek, és a dokumentáció figyelmen kívül hagyásának is. Azóta is rendszeresen azt kérdezem magamtól: *„**Miért nem tettem meg jóval korábban?!**"*

Egy kis történelem

A ***FreeBSD*** történetét – ha nem is a világ – de a számítógépek teremtésével kell kezdeni, jobban mondva ezek operációs rendszereivel. Az operációs rendszerek mintegy negyven évre visszanyúló történelmének kezdetén az első igazán használható, egyszerre több programot futtatni képes operációs rendszer a ***Multics*** volt. Ezt pár évig aktívan használták, de aztán inkább kistestvérét, a ***UNICS*** (később ***UNIX***) operációs rendszert kezdték el fejleszteni és széles körben használni helyette. Fejlesztői között akadnak olyan nagy nevek, mint ***Ken Thompson***, és ***Dennis Ritchie***. Meg kell azonban említeni, hogy mellőzése ellenére a ***Multics*** fejlesztése nem állt meg, egészen az ezredfordulóig karbantartották.
A ***UNIX*** rendszert ***Ken Thompson*** mutatta be a ***Berkeley*** egyetemen, ahol meghívott professzorként előadásokat tartott. Az egyetem lelkes diákjai és tanárai jelentős fejlesztéseket hajtottak végre ezen az operációs rendszeren, majd ezt önálló operációs rendszerként ***BSD*** (***Berkeley Software Distribution***) néven közreadták. Pár év múlva az ***AT&T*** (a ***UNIX*** jogainak birtokosa) kereskedelmi forgalomba hozta a ***UNIX*** rendszert. A ***BSD*** ekkor még mintegy egynegyed részben ***AT&T*** forráskódot tartalmazott, az egyetem viszont nem állt úgy anyagilag, hogy ezek jogát megvásárolhassa. Gondoltak azonban egy merészet, és a közzétették a ***BSD*** rendszer forráskódját az ***AT&T*** forráskódjai nélkül. Ez a változat természetesen működésképtelen volt, de meg lehetett hozzá írni a hiányzó részeket. Ezt ***William F. Jolitz*** tette meg, aki 1992-ben ***386BSD*** néven közreadta az új változatot. Ez sajnos soha nem lett stabil operációs rendszer. Egy év múlva csaknem egyszerre két projekt is indult a ***386BSD*** alapjain: a ***FreeBSD*** és a ***NetBSD***. Egy évvel később ***NetBSD*** két részre szakad, és létrejött az ***OpenBSD*** is, amely a legfiatalabb ***BSD*** rendszer.

Fejlesztőinek alapvető célkitűzése az volt, hogy elkészítsék a lehető legbiztonságosabb, kiszolgálókon futtatható operációs rendszert. A ***NetBSD*** csapat ugyanakkor egy gépfüggetlen, alaposan átgondolt, jól megtervezett operációs rendszer létrehozását tűzte ki célul, amely sok géptípuson működhet. Végül a ***FreeBSD*** projekt résztvevői az egyszerű kezelhetőséget és a gép teljesítményének legjobb kihasználását tartották a legfontosabbnak. A három ***BSD*** változat közül a ***FreeBSD*** a leginkább ismert és elterjedt. Előszeretettel alkalmazzák kiszolgálói feladatokra, megbízhatósága és stabilitása legendás.

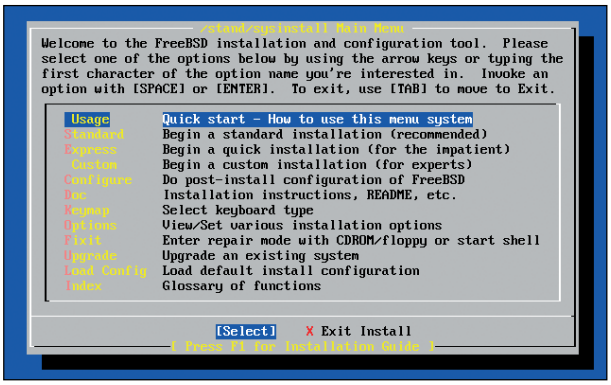
FreeBSD-s alapismeretek

Tekintettel a helyre célszerűnek tartom a ***FreeBSD*** ismertetését a ***Linux*** rendszerekkel való összehasonlítással kezdeni. Tapasztalataim alapján bizton állíthatom, hogy a 90% azonosság mellett a maradék 10% eltérés elegendő arra, hogy az első hetekben megkeseríte az ismerkedni vágyó életét. Ez természetesen nem vonatkozik azokra, akik még nem ismerik túl alaposan a ***Linux*** rendszert.
A legelső különbség az alaprendszer felépítésében van. A ***Linux*** szűken véve egy mag, amely köré a terjesztések készítői építenek működőképes rendszert. A ***Linux*** rendszermag minden különösebb probléma nélkül lecserélhető anélkül, hogy a telepített csomagokat is változtatnunk kellene. (Természetesen vannak esetek, amikor néhány csomag frissítése szükséges). ***FreeBSD***-nél egy újabb rendszermag fordítása ***feltétlenül*** maga után vonja ***a*** teljes alaprendszer „hozzáfordítását". (Ez persze nem olyan rémes és bonyolult, mint amilyennek így elsőre hangzik). Ha ezt elmulasztjuk, az rejtélyes hibákat okozhat a működésben. A számos kudarcnak talán pont ez lehet az egyik oka: ezeket az érthetetlen hibákat látva a legtöbben sarkon fordulnak és soha többé nem fognak a kezükbe ***FreeBSD*** telepítőlemezt.
Újabb nagy különbség, hogy ***Linux*** esetén (túlnyomó többségében) minden alkalmazás, programegység egy-egy csomag formájában érkezik, s ezek pontos függőségi viszonyok szerint telepíthetőek, frissíthetőek. A ***FreeBSD*** alaprendszert vehetjük úgy, mint egyetlen 200 MBájt méretű csomagot, amely tartalmazza a rendszermagot és a működéshez feltétlenül szükséges programokat: parancsértelmezőt (héjat), a szokásos ***UNIX*** programok gyűjteményét, az alapvető hálózati eszközöket, valamint sok más egyéb összetevőt. Az is lényeges, hogy a ***FreeBSD*** fejlesztői viszonylag ritkán hozakodnak elő újabb bináris terjesztéssel. Az ezek között eltelt időt sem kell azonban tétlenül tölte-

nünk, mivel a rendszer forrása óráról-órása folyamatosan fejlődik, és a megfelelő ***CVS*** kiszolgálón bárki számára elérhető. Ezt az utat választva – rendszeresen frissítve telepített rendszerünket – kikövezett úton tudunk haladni, mindig a legstabilabb és legfrissebb alaprendszer birtokosaként.
Linux rendszereknél is ismert a stabil és a fejlesztői változat fogalma. ***Linuxos*** körökben a különbséget a verziószám páros vagy páratlan volta jelzi. Ez ***FreeBSD*** esetén sincs másként, de nem a verziószámból tudjuk egyértelműen eldönteni, hogy az adott változat fejlesztői vagy stabil, hanem a mögé írt szóból. A stabil alaprendszer verziószáma mögött mindig a ***-STABLE*** kiegészítést olvashatjuk. Ezeket bátran használhatjuk élesben is. Időnként megjelenik egy-egy bináris terjesztés is, amelyet a verziószám után írt ***-RELEASE*** szó jelöl. Ez egyfajta pillanatképe a ***-STABLE*** ágnak. A ***-RELEASE*** változat néhány hónapos használata után érdemes áttanulmányozni a feltárt hibák listáját, mielőtt éles helyzetben is üzembe helyeznénk a rendszert. (A legújabb -RELEASE a 4.10, amely lassan fél éves lesz). A fejlesztői ágat a ***-CURRENT*** utótag jelzi. Erre lehetőleg ne bízzunk olyan feladatot, amely megbízhatóságot és stabilitást igényel, viszont ismerkedésre, illetve munkaállomások üzemeltetésére tökéletesen használható szokott lenni.
A ***FreeBSD*** az eszközök elnevezése terén alapvetően azonos elvet követ, a megnevezések azonban részleteikben mégis különböznek egy ***Linux*** rendszer esetén megszokottaktól. ***Linux*** alatt az eszközök nevei elsősorban azok rendeltetésétől függnek. A második ***Ethernet*** hálózati kártya például büszkén viseli az ***eth1*** nevet. ***FreeBSD*** alatt ne is keressünk ilyet. Itt ugyanennek a kártyának az elnevezése lehet ***rl1***, ***ed1***, de lehet akár ***xl1*** is, a meghajtómodultól függően.
A merevlemezek neve ***hda*** vagy ***sda*** helyett ***ad0*** vagy ***da0*** lesz, de a RAID használata ezt is felboríthatja.
Aki megszokta az a módot, ahogy a ***Linux*** a merevlemezeket kezeli, annak elsőre hajmeresztő lesz a ***FreeBSD***-nek a partíciók kezelése terén tapasztalható „fapadossága”. A ***FreeBSD*** ugyanis kizárólag az elsődleges lemezrészeket képes könnyedén kezelni (nehézkesen bár, de a kiterjesztett és a logikai típust is használhatjuk), illetve csak elsődleges partícióról képes elindulni. Egy partíciót a ***FreeBSD*** terminológia ***szeletnek*** (***slice***) hív, amely tovább osztható ***címkékre*** (***label***). Minden szelet 8 címkét tartalmazhat, egy merevlemezen tehát összesen 32, különböző célt szolgáló címkét tudunk létrehozni.
Egy olyan IDE merevlemezen például, amelyen Linux alatt a ***/dev/hda7*** a kiterjesztett partíció harmadik logikai partícióját jelentette, ***FreeBSD*** alatt ugyanez a terület a negyedik szelet harmadik címkéje lesz, és a ***/dev/ad0s3c*** nevet fogja viselni. Néhány csatolási pont esetén fontos, hogy melyik címkéhez rendeljük. A gyökérkönyvtár minden esetben a legelső (***a*** jelű), a cserepartíció pedig a második (***b*** jelű) címkén lehet. A teljes lemezterület a harmadik (***c*** jelű) címke. A többi csatolási pontnak nincs ilyen megkülönböztet szerepe, így szabály sem vonatkozik rájuk. Természetesen egy másik szeleten más célra is felhasználhatjuk ugyanezeket a címkéket.

Előkészületek a telepítéshez

Első dolgunk egy telepítőlemez beszerzése lesz. Akinek szélessávú kapcsolata vannak, annak elég lesz egy bootonly (20 MBájt) vagy egy minimal (200 MBájt) telepítőkészlet, amely csak a rendszermagot, illetve az alaprendszert tartalmazza.

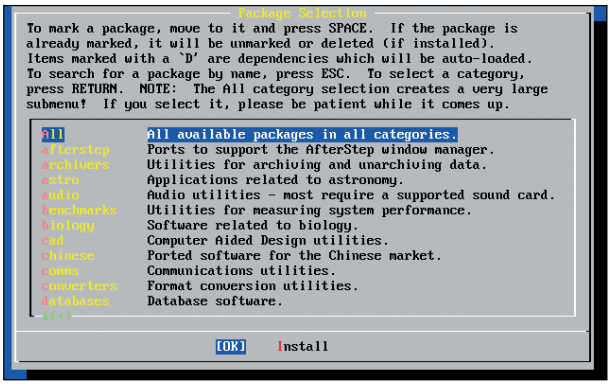


1. ábra

Gyors hálózati kapcsolat hiányában két CD lemezt kell megszereznünk. Ismerkedés céljából ajánlom a legújabb 5.3-BETA6 változatot, amely nagyon közel van a stabil ághoz, bár még nem stabil (ezt jelenti a BETA6 tag). A megfelelő telepítőkészlet letölthető az ➲ ***ftp://ftp.freebsd.org/pub/FreeBSD/ISO-IMAGES-i386/5.3/*** címről, majd tetszőleges módon adathordozóra írható.
Érdemes egy gyors leltárt készíteni a gépünkben található hardverelemekről, persze elég csak azokról, amelyeket előreláthatólag a ***FreeBSD*** alatt is használni kívánunk majd. Ez sokat segít majd a telepítés folyamán, bár a telepítőprogram elég sok eszközt magától fel fog ismerni és helyesen be is állítja azokat.
Célszerű a merevlemezen egy egész partíciónyi helyet kihagyni a ***FreeBSD*** számára, mert a telepítőprogram nem képes átméretezni illetve átmozgatni más partíciókat. Ha nincs szabad hely, akkor használhatjuk a ***fips*** vagy a ***PartitionMagic*** programokat a „helycsináláshoz”. Mint minden hasonló műveletnél, most sem árt mentést készíteni adatainkról.
Át kell gondolnunk a merevlemez új szerkezetét is, hiszen mint említettem a ***FreeBSD*** csak elsődleges partícióról képes elindulni. Csak akkor telepítsük a ***FreeBSD*** meglehetősen puritán ***BootMgr*** programját, ha a többi operációs rendszerünk is elsődleges partícióról indul. Ha van már valamilyen bootmanager program telepítve, akkor jobban járunk, ha azt használjuk.

A telepítés

A telepítés szokásos módon kezdődik, vagyis elindítjuk a telepítőprogramot. Ezt megtehetjük úgy, hogy közvetlenül CD lemezről indítjuk a gépet. Ha ez valamiért nem megy, akkor készíteni kell három indítólemezt, és ezekről bootolni. Akárhogy is kezdünk hozzá a dologhoz, a végén el kell indulnia a ***FreeBSD*** rendszermagjának.
Ez rengeteg eszközhöz tartalmaz meghajtóprogramot, viszont normális esetben ezeknek csak a töredéke található meg ténylegesen a gépünkben. Éppen ezért először a rendszermag beállításaihoz kerülünk, ahol az automatikusan felderített eszközök listájában nyugtázzuk a helyesen működőket, kitöröljük a helytelenül felismerteket. Ugyanitt tudjuk letiltani azokat a meghajtóprogramokat, amelyeket nem akarunk használni, illetve itt adhatjuk meg a működésükhöz szükséges paramétereket. (Előfordulhat, hogy bizonyos dolgokat a rendszermag nem tudott megfelelően beállítani.)



2. ábra

Az első telepítés során ezt a lépést tulajdonképpen nyugodt lélekkel kihagyhatjuk, éles rendszernél azonban érdemes az útmutató szerint beállítani a rendszermagot.
A rendszermag beállításának átugrása (vagy elvégzése) után elindul a ***FreeBSD*** karakter alapú telepítőprogramja a ***sysinstall***. Ezzel a programmal még rendszeresen fogunk találkozni, mivel ez a ***FreeBSD*** „mindenese". Szinte minden az alaprendszerrel kapcsolatos beállítást el tudunk vele végezni. Elsőre lehetőleg a ***Standard*** telepítési módot válasszuk, mivel tapasztalataim szerint ez adja a kompromisszumokkal teli, de kezdők számára leginkább működő végeredményt. Türelmetlenek számára az ***Express*** is használható, de ekkor sok olyan kérdésre a telepítő program „válaszol", amire esetleg nem kellene. Lehetőleg csak akkor használjuk, ha nincs máris operációs rendszer a merevlemezen. A ***Custom*** telepítési mód szinte mindent megkérdez, így ezt csak tapasztalt – legalább négy-öt ***sikeres*** telepítést megélt – veteránoknak ajánlom.
Az első fázisban feladatunk a szeletek kiválasztása lesz, mégpedig az ***Fdisk*** nevű program segítségével. Az egyetlen dolgunk az, hogy az üresen hagyott helyen egy szeletet létrehozzunk. Ezt a betűparancsok közül a ***c*** lenyomásával tudjuk végrehajtani. A létrejött szelet típusa ***freebsd.*** Ezt később majd tovább tudjuk osztani címkékre. Ha készen vagyunk a szelet létrehozásával, akkor az ***f*** billentyűvel véglegesíthetjük a változtatást. Ezt követően a telepítőprogram megkérdezi, hogy telepíts-e a ***BootMgr*** programot. Itt több lehetőség közül választhatunk. Próbáljuk megtalálni az igényeinkhez leginkább igazodó megoldást.
Ezek után a ***freebsd*** típusú szeleteken létre kell hoznunk a címkéket. Célszerű először a telepítőre hagyni ezek elkészítését, vagyis egyszerűen megnyomni az a billentyűt, ugyanis könnyedén el tudjuk hibázni a címkék elnevezését. A ***w*** gomb kiírja a lemezre a változásokat és létrehozza a megfelelő fájlrendszereket is. A programból a ***q*** billentyűvel léphetünk ki, mire folytatódik a telepítés. A következő lépés az alaprendszer kiegészítő összetevőinek kiválasztása. Választhatjuk a ***minimum*** telepítést, de akár az összes komponenst is feltelepíthetjük. A ***ports*** komponenst érdemes telepítésre kijelölni, mivel fontos lesz a későbbiekben.
A komponenslista után ki kell választanunk a telepítési forrást, vagyis ahonnan a telepítő be tudja tölteni az egyes részek állományait. Ez lehet CD vagy DVD, de sok hálózaton át történő letöltés közül is választhatunk. Elérkeztünk a legfontosabb pillanatig, a telepítés tényleges megkezdéséig. Ha továbbmegyünk, akkor felkerül a gépünkre az áhított ***FreeBSD*** operációs rendszer. A kiválasztott komponensektől, a telepítési forrástól no és gépünk sebességétől függően pár perctől több óráig is eltarthat a telepítésnek ez a fázisa.

Telepítés után

Az állományok felmásolása után további kérdéseket tesz fel a telepítőprogram. Ezekre akár később is válaszolhatunk, ha elindítjuk a */stand/sysinstall* programot.
Elsőként a hálózati kapcsolat beállítását ejthetjük meg. (Ha korábban hálózati telepítési forrást adtunk meg, akkor ezek a paraméterek értelemszerűen már be vannak állítva). Kiválaszthatjuk a hálózati kártyát, majd a szokásos adatok megadása után akár azonnal használatba is vehetjük azt.
Ezek után sok-sok a beállításokkal kapcsolatos kérdés következik. Nyilatkoznunk kell az ***inetd***, az ***FTP*** és az ***NFS*** kiszolgáló „ügyeiről", meg kell adnunk a konzol betűtípusát és a billentyűzetkiosztást; az időzónát; az egér típusát; valamint ki kell választanunk a rendszer által automatikusan elindítandó szolgáltatásokat. Ha feltettük az ***XFree86*** rendszert, akkor ennek a beállítását is elvégezhetjük, de jobb, ha ezzel később foglalkozunk.
Ezek után feltehetünk olyan kiegészítő csomagokat is, amelyek karbantartását nem vállalta fel a ***FreeBSD*** csapat. Nagyjából tizenkétezer ilyen csomag áll rendelkezésre, amelyek közül ki tudjuk választani a számunkra szükséges vagy megszokott programokat. Ezek kategóriákba vannak sorolva, a minden kategóriában programok százai közül választhatunk. Ezek telepítése során előfordulhat, hogy sokkal több csomag kerül fel, mint amennyit kiválasztottunk. Ennek oka, hogy a csomagoknak szüksége lehet más csomagra a pontos és hibátlan működéshez, ezért az igényelt csomagokat is telepíteni kell.
A csomagok feltelepedése után érdemes egy saját felhasználót készíteni, hogy ne ***root*** jogokkal essünk neki a használatnak. A ***root*** jelszót is meg kell adni, amelyet jól gondoljunk át, főleg akkor, ha szélessávú kapcsolatot használunk majd.
A telepítőprogramból való kilépés után a gép újra fog indulni, és sikeres beállítások esetén a ***FreeBSD*** használatba vehető. Ha valahol hibával találkozunk, érdemes megnézni a ***FreeBSD*** kézikönyvet, amely sokrétű és bő leírást ad a telepítésről (egyelőre sajnos csak angol nyelven).

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.

HIVATKOZÁSOK

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

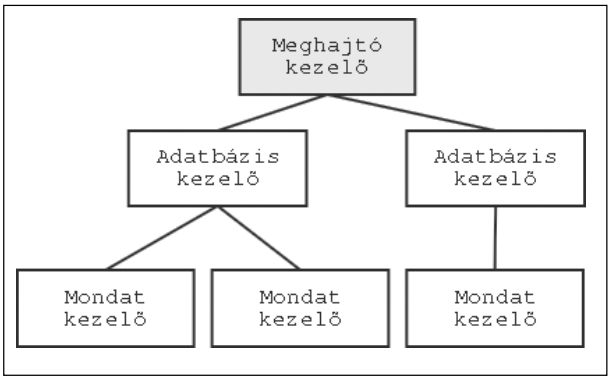
A Perl és az adatbázisok (3. rész)

Hajtsuk uralmunk alá a világ összes adatbáziskezelőjét SQL parancsokkal! A Perl DBI modulja segít nekünk ebben.

Az előző részekben megismerkedtünk az adattárolás alapjait jelentő szöveges, és a ***Berkeley*** alrendszerre támaszkodó adatbázisokkal. Ezek programozása elég egyszerű és gyors, ám nagyobb rendszerek esetén komoly nehézségek adódhatnak ha ezeket használjuk. A sokféle kódlap miatt az adatállományok nehezen hordozhatóak, a mezők típusai csak körülményesen adhatók meg, a hálózati elérésről pedig ne is beszéljünk.
Ezek a gondok abból adódnak, hogy az első részben hangoztatott adatfüggetlenítési elvet eddig egyáltalán nem vettük figyelembe. A fent említett feladatokat ugyanis egy elkülönített adatkezelő rétegnek kellene ellátnia. Az eddig bemutatásra került módszerekkel ezt nem lehetett megvalósítani, most azonban lépünk egy nagyot, és megnézzük, hogyan lehet egyszerre akár több relációs adatbáziskezelő rendszert (***RDBMS***) is használni.

Közös nyelv – közös felület

Feltételezem, hogy az Olvasó találkozott már olyan táblákból, és az azok közötti kapcsolatokból építkező adatbáziskezelővel, amelynek a kezelőnyelve az ***SQL*** volt. Ebben a hónapban azt vizsgálom meg, hogyan használhatunk egy ilyen rendszert a ***Perl*** segítségével. Persze nem tudhatom, hogy az Olvasó ***MySQL***, vagy ***PostgreSQL*** párti. Ha azonban egységesített felületen keresztül érjük el kedvenc adatbázisunkat, és a háttérben dolgozó meghajtó-'programra bízzuk az egyedi sajátságok kezelését, akkor ez nem is lényeges.
A közös felületet a ***DBI*** jelenti. Ez egy ***Perl*** modul formájában megjelenő programozói felület, amely mögött számos elterjedt adatbáziskezelő rendszerhez találunk meghajtóprogramot. Röviden szólva a ***DBI*** használatával egy újabb réteg kerül az adatbáziskezelő és saját programunk közé, ehetővé téve ezáltal hogy úgy álljunk át például ***MySQL***-ről ***Oracle***-re, hogy ehhez a program akárcsak egyetlen sorát át kellene írnunk. Azt hiszem ezt hívják programozói álomnak.
Minden támogatott rendszert egy-egy ***DBD*** modul képvisel, ezek jelentik a ***DBI*** mögötti meghajtókat. A rövidítések a belső logikai kapcsolatrendszerre utalnak: ***DBD*** = „***DataBase Dependent***" (adatbázisfüggő) és ***DBI*** = „***DataBase Independent***" (adatbázistól független). Jelenleg az alábbi rendszerekhez van megbízható támogatás: ***Oracle***, ***Informix***, ***mSQL***, ***MySQL***, ***Ingres***, ***Sybase***, ***DB2***, ***Empress***,



SearchServer, ***PostgreSQL***, és ***XBase***. A régi idők szerelmeseinek pedig van egy ***CSV*** modul is a közönséges szöveges állományokhoz.
Az egyes adatbáziskezelők meghajtóit tartalmazó modulok `DBD::rdbms` néven érhetők el, ahol az `rdbms` a szóban forgó rendszer neve. Így az ***Oracle*** meghajtót a `DBD::Oracle` tartalmazza. A ***DBI*** elég okos ahhoz, hogy a csatlakozáskor megadott adatforrás alapján önműködően betöltse a megfelelő meghajtót, így erről nekünk már nem kell gondoskodnunk. Nem csak hogy nem kell, valójában nem is szabad, így az egyetlen modul, amit nekünk, a programozónak be kell töltenünk, maga a ***DBI***.

A DBI felépítése

A ***DBI*** teljesen objektumközpontú. Némi túlzással fogalmazhatunk úgy, hogy a ***DBI*** felépítése az egységbezárás mintapéldája. Ennek megvan az a szépsége, hogy lehet akár párhuzamosan két kapcsolatunk két különböző adatbáziskezelővel, az ezeken elvégezhető műveletek biztosan nem fognak keveredni. Kevésbé pongyolán fogalmazva, tiszta és átlátható programot kapunk az elemfüggvények használatával.
Három fő objektumtípus áll rendelkezésünkre a kedvenc adatbáziskezelőnkkel történő kapcsolattartásra. Ezeket ***kezelőknek*** (***handle***) hívják, és egy jól meghatározott hierarchiát alkotnak. Az első a ***meghajtó kezelő*** (***driver handle***), amely egy belső használatra fenntartott objektum, és egy betöltött meghajtót jelképez. Nekünk ezzel nem kell foglalkoznunk, a `connect()` elemfüggvényt is az ősosztályon keresztül fogjuk meghívni.

A második az ***adatbázis kezelő*** (***database handle***), amely egy pillanatnyi munkamenetet jelképez egy adott adatbázissal. Ez a hozzá tartozó meghajtó kezelő gyermeke és mi rendszerint a `DBI -> connect()` függvénytől kapjuk vissza. Ez utóbbi a paraméterként átadott adatforrásból fejti meg, hogy mely meghajtóból kell származtatni az adatbázis kezelőt. A ***DBI*** leírása és a példa programok alapján `$dbh` néven szokás hivatkozni rá.
Végül, de nem utolsó sorban a ***mondatkezelő*** (***statement handle***) egy ***SQL*** mondatot zár egységbe. A mondat kezelők az adott adatbáziskezelő gyermekei. Mint ilyenek, egy mondat kezelő adatai védettek a többi mondat kezelő módosításaitól. Rendszerint a szülő `prepare()` elemfüggvényétől kapjuk vissza ezt az objektumot. A fentebb bevezetett elnevezési módot követve, `$sth` névvel fogunk hivatkozni rá.

Az adatforrások

Amikor egy adatbázishoz csatlakozunk a ***DBI*** segítségével, meg kell határoznunk egy úgynevezett adatforrást. Ez egy kettőspontokkal elválasztott lista. Első tagja kötött, és mindig a `DBI` betűk alkotják, akárcsak a honlapok elérésénél használt `http`. Második tagja a használni kívánt meghajtó neve, pl. `Xbase`. A további paraméterek mind átadásra kerülnek a meghajtó kezelő saját `connect()` függvényének. Ezek már meghajtónként eltérhetnek, itt szokás megadni pl. azt TCP/IP kaput, amin az adatbáziskezelő figyel.
Első körben próbáljuk meg feltérképezni saját rendszerünk adta lehetőségeket. A ***DBI*** osztály `available_drivers()` elemfüggvénye egy tömböt ad vissza, amely a használatra kész meghajtókat tartalmazza. Ezek bármelyikét átadva a `data_sources()` elemfüggvénynek, megkapjuk az adott meghajtón keresztül igénybe vehető adatforrások listáját, szintén egy tömbként. Ezek segítségével megjeleníthetjük az összes meghajtó összes adatforrását az adott rendszeren:

```
#!/usr/bin/perl -w

use strict;
use DBI;

my @meghajtok = DBI -> available_drivers ();
die "Nem találtam meghajtókat!\n" unless
     ➥@meghajtok;

foreach my $meghajto ( @meghajtok ) {

    print "Meghajtó: " . $meghajto . "\n";
    my @adatforrasok = DBI -> data_sources
       ➥( $meghajto );

    foreach my $adatforras ( @adatforrasok ) {
       print "\tAdatforrás: " . $adatforras .
             ➥"\n";
    }

    print "\n";

}
```

Úgy gondolom, a forrás önmagáért beszél. Én a következő kimenetet kaptam:

```
Meghajtó: ExampleP
       Adatforrás: dbi:ExampleP:dir=.

Meghajtó: Proxy

Meghajtó: mysql
       Adatforrás: DBI:mysql:mysql
       Adatforrás: DBI:mysql:mythconverg
       Adatforrás: DBI:mysql:proba_adatbazis
       Adatforrás: DBI:mysql:test
```

Elég szegényes telepítés, mindössze három meghajtóm van! Ezek sorát a megfelelő `DBD::meghajto` modul kézi telepítésével, vagy ***Debian*** alatt ***libdbd-meghajto-perl*** csomag beszerzésével lehet bővíteni. Azt azonban érdemes megfigyelni, hogy a ***MySQL*** adatforrások között az összes ***MySQL*** adatbázisomat látom. Ez azt is jelenti, hogy az adatbázisok neveire mindennemű hitelesítés nélkül, egyszerűen szert lehet tenni.

Lányok lekérdezése

Következő lépésként lássunk egy kis szkriptet, amely egy, a már a korábbi részekből megszokott lekérdezést valósít meg egy ***MySQL*** adatbázison. Látni fogod, hogy az adatbázishoz szorosan kötődő kód mindössze egy sor, így a program könnyen átültethető egy másik környezetbe.

```
#!/usr/bin/perl -w

use strict;
use DBI;

die "Használat: " . $0 . " <~lány neve>\n" unless
➥@ARGV;

my ($dbh, $sth, $minta, @sor);

$dbh = DBI -> connect (
    "DBI:mysql:proba_adatbazis",
    "testuser",
    "jelszo",
    {
       PrintError => 0,
       RaiseError => 1
    }
);

$minta = $dbh -> quote ($ARGV [0]);
$sth = $dbh -> prepare ("SELECT * FROM lanyok
➥WHERE nev LIKE " . $minta);

$sth -> execute ();

while (@sor = $sth -> fetchrow_array ()) {
    my ($nev, $telszam, $szereti) = @sor;
    print " Adatlap: " . $nev . "\n";
```

```
    print "==========" . "=" x length ( $nev ) .
          ➥"\n";
    print "  Telefonszám                 : " .
          ➥$telszam . "\n";
    print "  Ezzel veheted le a lábáról : " .
          ➥$szereti . "\n\n";
}

$dbh -> disconnect ();
```

Nézzük sorról-sorra! Az első sorban feltüntetjük a parancsértelmező elérési útját, és jelezzük a `-w` kapcsolóval, hogy a figyelmeztető üzeneteket is szeretnénk látni. Bekapcsoljuk a szigorú ellenőrzést, majd használatba vesszük a ***DBI*** modult. A negyedik sorban egy, a program helyes használatára vonatkozó üzenettel kilépünk, amennyiben nem volt átadott paraméter. Remélem, az eddig látottak nem okoztak meglepetést.
Az ötödik sorban jelezzük a használni kívánt változókat. A `$dbh` az adatbázis kezelő, a `$sth` pedig a mondat kezelő lesz. Láthatólag nem lesz szükségünk meghajtó kezelő objektumra, hiszen azt kizárólag belső használatra szánták. A $minta a parancssorban átadott mintát fogja tartalmazni, de ezen még finomítunk. Továbbá amikor egy ciklusban végigmegyünk az ***SQL*** lekérdezés eredménylistáján, egy `@sor` tömbben lesz a pillanatnyi rekord.
Ezután meghívjuk a `DBI` osztály `connect()` függvényét, és a visszaadott adatbázis kezelő objektumot a `$dbh` változóban tároljuk. Az elemfüggvény első paramétere az adatforrás, mely kettőspontokkal elválasztva tartalmazza a kötelező `DBI` szót, a meghajtó nevét, és a használni kívánt adatbázist. Ez utóbbit már a meghajtó saját `connect()` függvénye kapja meg. Az adatforrást követő két paraméter tartalmazza a felhasználónevet és jelszót.
A negyedik paraméter egy asszociatív tömb. Az adatbáziskezelővel történő kapcsolattartás egészére vonatkozó attribútumok adhatók meg itt, de az alapértelmezések elfogadása esetén el is hagyható a paraméter. Ha a `PrintError` `1`, bármilyen hiba fellépése esetén a ***Perl*** `warn()` függvényén keresztül a felhasználó értesül a hibáról. Alapértelmezésben 1. Ha a `RaiseError` `1`, bármilyen hiba fellépése esetén a ***Perl*** `die()` függvényén keresztül hibaüzenet után azonnal kilép. Alapértelmezésben 0.
A program az adatbáziskezelőre bízza a mintaillesztést. Ennek előnye, hogy a kiszolgálónak kell szűkítenie a kiválasztás eredményét, viszont hátránya, hogy sajnos nem lehet szabványos kifejezéseket használni az ***SQL***-ben. Mindenesetre bármilyen szöveget adunk is át az adatbáziskezelőnek, érdemes levédeni. A `quote()` elemfüggvény nem csak két idézőjel közé teszi a teljes szöveget, de a közbeeső különleges karakterekre is figyel.
A következő sorban meghívjuk az adatbázis kezelő `prepare()` elemfüggvényét. Ez előkészít egy ***SQL*** mondatot a végrehajtásra. Az előkészítés azt jelenti, hogy létrehoz egy mondat kezelő objektumot. Utóbbinak az `execute()` elemfüggvényét kell meghívni a lekérdezés végrehajtásához. Ez azért hasznos, mert a mondat összeállítása és a lekérdezés nem feltétlenül kell, hogy egy helyen legyen. Továbbá, egy lekérdezés többszöri végrehajtása adhat különböző eredményeket.

A fő ciklusban rekordonként dolgozzuk fel a kapott eredményeket. Ehhez a mondat kezelő `fetchrow_array()` elemfüggvényét használjuk. Ennek segítségével minden iterációban egy újabb rekord mezőit tartalmazza a `@sor` tömb. A ciklus magja az első rész óta nem sokat változott. Egy szép adatlapot ad, ahol a név a megfelelő hosszúságban alá van húzva, ezt követi a másik két mező. A program legvégén megszakítjuk a kapcsolatot az adatbáziskezelővel.

A hárem bővítése

A most következő gyöngyszem a tárolás műveletet valósítja meg. Némileg módosult a hibakezelés rész, továbbá van egy trükk az ***SQL*** utasítás előkészítésében. Mindezek a fortélyok még nagy segítségedre lehetnek, ha adatbázis-programozásra adnád a fejed. Lássuk!

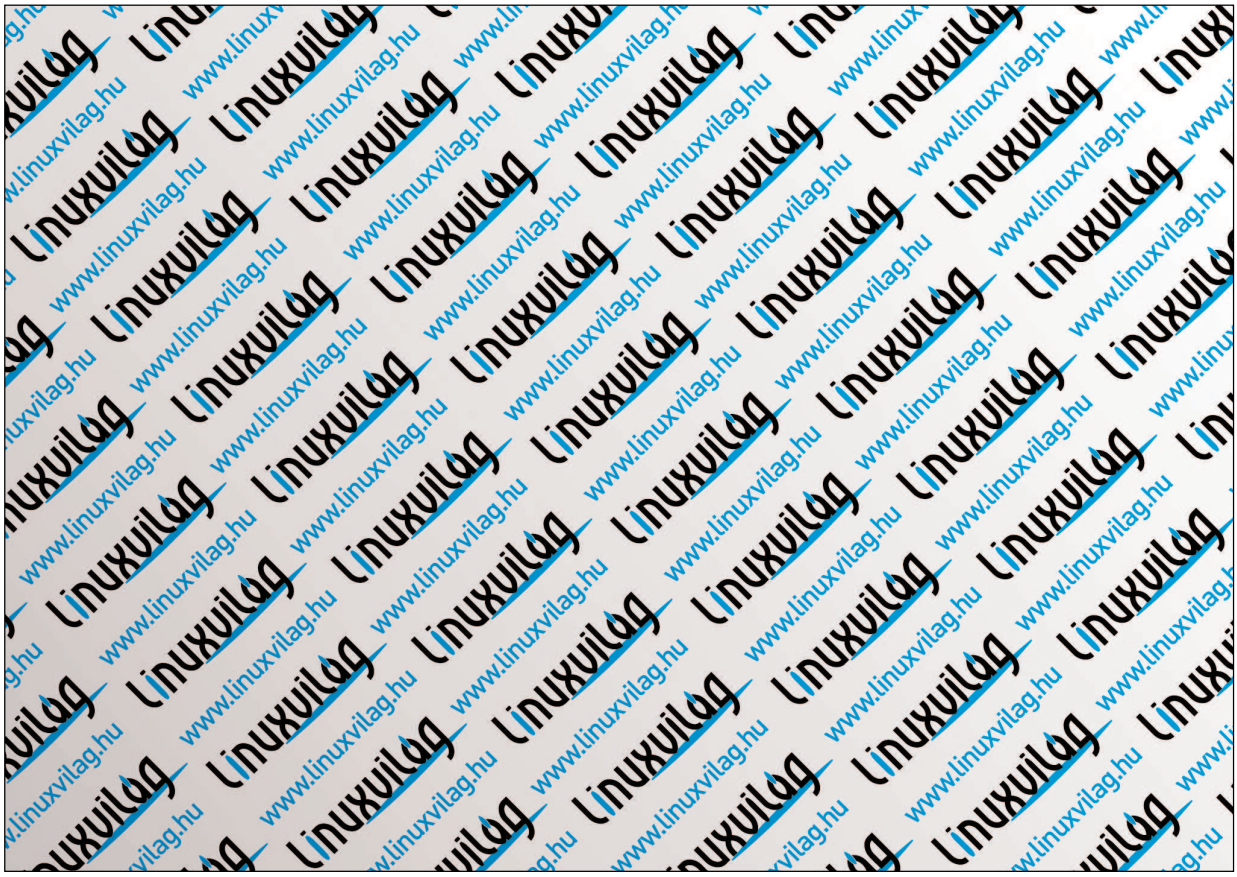
```
#!/usr/bin/perl -w

use strict;
use DBI;

die "Használat: " . $0 . " <adatbazis_nev>\n"
➥unless @ARGV;

my ($dbh, $sth);

$dbh = DBI -> connect (
    "DBI:mysql:" . $ARGV [0],
    "testuser",
    "jelszo",
    { PrintError => 0, RaiseError => 0 } ) or
     ➥die "Nem tudok csatlakozni az adatbázishoz:
     ➥" . $DBI::errstr;

eval {
    $dbh -> {RaiseError} = 1;
    $sth = $dbh -> prepare ("INSERT INTO lanyok
         ➥VALUES (?, ?, ?)");

    print "Neve                          : ";
    my $nev = <STDIN>; chop $nev;
    print "Telefonszám                  : ";
    my $telszam = <STDIN>; chop $telszam;
    print "Ezzel veheted le a lábáról : ";
    my $szereti = <STDIN>; chop $szereti;

    $sth -> execute ($nev, $telszam,
    ➥$szereti);
};
warn "Hiba történt a művelet végrehajtása során: "
➥. $@ if $@;
s
$dbh -> disconnect ();
```

Az első újdonság, hogy parancssorban át lehet adni a használni kívánt adatbázis nevét. Ezzel még nem oldjuk meg a világegyenletet, de megkönnyítheti a szkriptet fekete dobozként kezelő felhasználóink életét. Ha még ennél is figyelmesebbek szeretnénk lenni a program használóival,

az első példában bemutatott módszerrel kikereshetjük az elérhető adatbázisok neveit, és felajánlhatjuk őket választási lehetőségként.
Az ötödik sorban ez esetben már csak két változót határoztunk meg előre. Ez valójában ízlés dolga. Egy *C*-hez szokott programozó jobban szereti a program legelején látni a használni kívánt szimbólumokat, egy *C++*-on felnőtt billentyűzet-nyűvő inkább a használat helyére tenné őket. Ebben a példában a további három változó meghatározása közvetlenül a használatba vétel helyén történik. Döntsük el, melyik tetszik jobban, viszont maradj következetes.
A `connect()` függvény kevéssé változott. Természetesen az adatforráshoz hozzá kell fűzni a parancssori paramétert. Viszont figyeld meg, hogy a `RaiseError` értéke most `0`, vagyis nem lép ki önműködően a program hibaüzenettel, ha a művelet nem volt sikeres. Helyette egy kézi hibaellenőrzést végzünk a már megszokott `fv() or die;` szerkezettel. A saját hibaüzenetben felhasználtuk az ősosztály `errstr` elemváltozóját a hiba típusának leírásához.
Ezután egy `eval {}` blokk következik, melynek első utasításában visszaállítjuk a `RaiseError` értékét 1-re. Így amennyiben a blokkon belül felmerül egy hiba, az önműködően meghívott `die()` miatt nem fejezi be a futását a program, hanem csak a blokkot hagyja el. A különleges `$@` változó tartalmazni fogja a hiba leírását, így azt egy önálló hibaüzenet megjelenítéséhez nyugodtan felhasználhatjuk.
Ez a megoldás kis jóindulattal a Java-s `try {} - catch {}` kistestvérének nevezhető.

Az ezt követő előkészítés tartalmaz három kérdőjelet. Ezekre a kérdőjelekre úgy lehet gondolni, mint az ***SQL*** mondat belső változóira. Előkészítjük ugyan az utasítást, de bizonyos részeket üresen hagyunk. Ezeket a lyukakat később a végrehajtás során betömhetjük. Itt ennek nincs sok jelentősége, hiszen a beolvasás után is előkészíthettük volna az utasítást. Egy ciklusba ágyazott `execute()` esetén viszont nagyon hasznos lehet ez a módszer.
Az adatok bekérése ugyanúgy zajlik, mint előző hónapban a ***Berkeley*** alrendszer használatakor, ezért ezt most nem részletezném. Az `eval {}` blokk utolsó utasításaként végrehajtjuk az ***SQL*** mondatot a megfelelő változóbehelyettesítéssel. Nagyon fontos a blokkot lezáró pontosvessző! Ezután következhet az esetleges hibák kezelése. Ha történt valamilyen gond, a `$@` változó biztosan nem üres. Ez esetben mindössze egy figyelmeztető üzenetet jelenítünk meg.
Sorozatunk utolsó állomásához érkeztünk. Remélem, sikerült bemutatnom az adatbáziskezelés alapvető fogalmait, és hogy mennyire egyszerű ezeket a gyakorlatban alkalmazni. Sok sikert kívánok a további kísérletezgetéshez, akinek pedig kérdése van, írjon bátran.

Fülöp Balázs (admin@guardware.com)
18 éves, imádja a Túró Rudit, a Debian Linuxot és a teheneket. Kedvenc írója Slawomir Mrožek. Leginkább a számítógépes hálózatok biztonsága érdekli. A BME VIK műszaki informatikus szak hallgatója.



MOODLE: Egy ingyenes e-learning keretrendszer

Építsünk könnyen, és gyorsan oktatóportált a mindenki számára elérhető, és ingyenes Moodle program segítségével.

Amikor először kerültem testközelbe a *Moodle*-lel, még nem gondoltam volna, hogy hónapokkal később cikk írására fogom adni a fejemet. A *Moodle* felhasználóknak viszonylag szerény tábora van kis hazánkban, ezzel szemben külföldön elég népszerű, és világviszonylatban jelenleg 1843 regisztrált *Moodle* alapokon nyugvó weboldal van. Szeretném tehát bemutatni, hogy mire is képes a *Moodle*, hogyan könnyíti meg a program az oktató tartalom felvitelét, szerkesztését, módosítását, a tananyagok rendszerezését, illetve újabb tananyagok beszúrását a már meglévők közé. Mindezt egyszerűen, és gyorsan el lehet végezni, anélkül, hogy bármiféle programozói tudással kellene rendelkeznünk, sőt még a *HTML* nyelvet sem kell ismerni ahhoz, hogy hivatkozásokat, képek, ábrákat szúrjunk be az egyes tananyagokba. Megpróbálom bemutatni annak az útnak az első lépéseit, amit végig kell járnunk ahhoz az *1. képen* láthatóhoz hasonló oldalt tudjunk készíteni.

Az első lépések

A *Moodle*-t a *Moodle* honlapjáról tudjuk – ingyenesen – letölteni. Ajánlott a legutolsó stabil verzió letöltése, ez jelenleg az 1.4+, mert ekkor nem érhetnek nagy meglepetések a működés folyamán. A telepítést és a program bemutatását is ezzel a verzióval végzem, így az esetleg későbbi verziók telepítése, konfigurálása kissé eltérhet a cikkben leírtaktól. A *Moodle* szinte az összes operációs rendszer alatt képes működni, ha az *Apache*, a *PHP* és a *MySQL* programok rendelkezésre állnak. A fejlesztők is ezekkel a programokkal ajánlják a használatot, de gyakorlatilag bármelyik másik webszerver, megfelel, amelyik támogatja a *PHP*-t, illetve a *MySQL* helyett választhatjuk a *PostgreSQL*-t is, folynak a fejlesztések további adatbázis szerverekkel való kapcsolat kiépítésére lehetőségére is. A *Moodle 1.4+* működéséhez legalább a 4.1.0-ás *PHP*-ra lesz szükségünk. Ezeknek a telepítését és beállítását nem részletezem, mert célom a *Moodle* képességeinek megismertetése. A telepítést UHU-Linux 1.1.1 alatt végeztem.
Miután letöltöttük a *Moodle*-t csomagoljuk ki és másoljuk a be a Apache dokumentum könyvtárába. Ha nem tudjuk, hogy éppen melyik az aktuális Apache dokumentum könyvtár, akkor a keressük meg a */etc/apache/main.conf* fájlban a `DocumentRoot` bejegyzést, ami valami ilyesmi lesz

1. kép Nagy segítség lehet az elektronikus oktatásban az ingyenesen használható Moodle

`DocumentRoot` "`/var/www/html`" ha már itt vagyunk, akkor ajánlatos megnézni a `User` és a `Group` kezdetű sorokat is, ha nem vagyunk pontosan tisztában azzal, hogy az *Apache* a fájlok hozzáféréséhez milyen csoport és felhasználó nevet használ. A következő parancsokkal tömörítsük ki, és mozgassuk a helyére a *Moodle*-t.

```
tar -zxvf  moodle-latest-1.4.tgz
mv moodle /var/www/html
```

Ezek után a *Moolde* a */var/www/html/moodle* könyvtárba kerül. A *Moodle*-nek szüksége van még egy adat könyvtárra, ahol az egyes tananyagokhoz kapcsolódó képeket, a felhasználók által feltöltött fájlokat tárolja. Erre a célra hozzuk létre a */var/moodledata* könyvtárat az `mkdir` parancs segítségével, ajánlatos a *moodledata* könyvtárat nem a *Moodle* főkönyvtárában elhelyezni, mert biztonsági mentések készítésénél külön tudjuk menteni a két könyvtárat.
Most be kell állítani a *Moodle* egyes változóit a *config.php* fájlban, a fájl nem létezik, ne keressük, viszont a program főkönyvtárában található *config-dist.php* egy példa arra, hogy a *config.php* hogyan is nézzen ki, másoljuk le tehát ezt a fájlt *cp config-dist.php config.php* és végezzük el rajta a szükséges beállításokat, az *1. kódrészlet* alapján.

Az adatbázis beállítása

A *MySQL* adatbázis telepítése után az adatbázishoz a *root* felhasználó jelszó nélkül hozzáférhet, ezért ajánlott

1. kódrészlet

```
#az adatbázis típusa mysql vagy postgres7
$CFG->dbtype = ‘mysql’;
#az adatbázis szerver címe
$CFG->dbhost = ‘localhost’;
#a Moodle által használt adatbázis a szerveren
$CFG->dbname = ‘moodle’ ;
#a Moodle ezzel a név-jelszó párral kapcsolódik
#az adatbázishoz
$CFG->dbuser = ‘moodleuser’ ;
$CFG->dbpass = ‘password’ ;
#minden egyes Moodle által használt tábla ezzel
#a karaktersorozattal fog kezdődni
$CFG->prefix= ‘mdl_’ ;
#ezen az URL-en keresztül fogjuk elérni
#a Moodle-t
$CFG->wwwroot = ‘http://localhost/moodle’;
#a Moodle könyvtár helye
$CFG->dirroot = ‘/var/www/html/moodle’;
#a Moodle adatkönyvtárának helye
$CFG->dataroot = ‘/var/moodledata ;’
```

a `mysqladmin -u root password “uj_jelszo”` paranccsal megadni egy új jelszót. Az adatbázist a `mysqld_safe` paranccsal tudjuk elindítani. Miután elindult az adatbázis próbáljunk meg csatlakozni hozzá a `mysql -u root -p` paranccsal, majd gépeljük be a ***root*** jelszavát. Biztonsági szempontból ajánlott az alábbi utasítások kiadása, mert ezek után csak a ***root***, férhet hozzá a ***MySQL***-hez. Figyelem! Ezeket az utasításokat csak teljesen új ***MySQL*** esetén alkalmazzuk, különben megrongálhatjuk a már létező adatbázisunkat.

```
USE mysq;
DELETE FROM user WHERE host!=”localhost” or
➥user!=”root” or password=””;
DELETE FROM db;
FLUSH PRIVILEGES;
```

Most készítsük el a ***Moodle***-nek szánt adatbázist a `CREATE DATABASE moodle;` parancs segítségével, majd adjuk ki a `USE moodle;` parancsot hogy a `moodle` adatbázisban dolgozhassunk. Most pedig adjuk hozzá a ***Moodle*** felhasználót a ***MySQL*** kiszolgálóhoz, és rendeljük hozzá a szükséges jogokat is.

```
GRANT SELECT, INSERT, CREATE, UPDATE, DELETE
-> ON moodle.*
-> TO moodleuser@localhost
-> IDENTIFIED BY ‘password’;
```

Ha minden sikeresen lezajlott, akkor az `exit` paranccsal lépjünk ki a ***MySQL***-ből. Már csak a */var/www/html/moodle*, és a */var/moodledata* könyvtárat jogát kell átállítani ahhoz, hogy megkezdhessük a ***Moodle*** használatát, ezért adjuk ki a következő két parancsot, természetesen ha nekünk a */etc/apache/main.conf* fájl `User` és `Group` változói más értéket tartalmaztak, mondjuk `apache`, `apache`, akkor azt adjuk ki.

2. kódrészlet

```
DirectoryIndex index.php index.html index.htm

<IfDefine APACHE2>
     AcceptPathInfo on
</IfDefine>

php_flag magic_quotes_gpc 1
php_flag magic_quotes_runtime 0
php_flag file_uploads 1
php_flag session.auto_start 0
php_flag session.bug_compat_warn 0
php_flag memory_limit 16M
```

```
chown -R www:www /var/moodledata
chmod -R 777 /var/moodledata
chown -R www:www /var/www/html/moodle
chmod -R 777 /var/www/html/moodle
```

Kisebb módosítások szükségesek még az Apache konfigurációs fájlban is, tehát nyissuk meg szerkesztésre a */etc/apache/main.conf* fájlt, és keressük meg benne a `DirectoryIndex` kezdetű sort, és ha nem szerepel benne a ***index.php*** bejegyzés, akkor célszerű az első helyre beszúrni, hogy az Apache mindig minden könyvtárban először az ***index.php*** nevű fájlokat próbálja meg megnyitni, és ha nem sikerül, csak akkor próbálja meg mondjuk az ***index.html*** fájlt megnyitni. Nálam ez a következőképpen néz ki:

```
DirectoryIndex index.php index.html index.htm
```

Ha az ***Apache*** 2-es változatát használjuk, akkor be kell be kell állítanunk az `AcceptPathInfo on` paramétert. Ha szeretnénk, hogy a magyar karakterek is jól jelenjenek meg az oldalunkon, akkor ellenőrizhetjük, hogy ugyanebben a fájlban az `AddDefaultCharset ISO-8859-2` hasonlóképpen legyen beállítva. Ha nincs jogunk módosítani ezeket a fájlokat, akkor is van lehetőségünk a szükséges paramétereket beállítani, de előtte nézzük meg, hogy hogyan kell módosítanunk a ***PHP***-t. A ***PHP*** konfigurációs fájlt általában a */etc/php.ini* név alatt találjuk meg, ellenőrizzük le, hogy az alábbi beállítások szerepelnek-e nálunk is (általában ezek az alapbeállítások, de nem árt a biztonság):

```
memory_limit = 16M
magic_quotes_gpc = 1
magic_quotes_runtime = 0
file_uploads = 1
session.auto_start = 0
session.bug_compat_warn = 0
```

A `memory_limit = 16M` beállítását a ***Moodle*** telepítési segédlete nem említi, azonban tapasztalatom szerint az alapbeállításként használt ***8M*** kevés lehet a ***Moodle*** számára, valószínű, hogy a legtöbb esetben a probléma nem fordul elő, de úgy érzem, hogy manapság, amikor már az otthoni számítógépekben is kezd elterjedni az 512 MB-os RAM használata,

a 16M használata nem jelenthet különösebb problémát. Most térjünk vissza, hogy mi a helyzet akkor, amikor nincs jogosultságunk sem a */etc/apache/main.conf* sem a */etc/php.ini* módosítására, szerencsére ezeket a beállításokat akkor is érvényesíteni tudjuk, ha készítünk egy *.htaccess* nevű fájlt a *Moodle* főkönyvtárában a *2. kódrészlet* tartalmával. Ha az *Apache* futott a beállítások változtatása alatt, akkor indítsuk újra, hogy az új beállítások érvényesek legyenek, amennyiben még nem fut az Apache, akkor most indítsuk el.

Ismerkedés a Moodle-lel

Szerencsésen túljutottunk a *Moodle* futtatásához szükséges beállításokon, és végre megkezdhetjük használni a programot. Az első elindulás előtt viszont a *Moodle*-nek el kell készítenie a szükséges adatbázis táblákat, és nekünk is meg kell adnunk még néhány futtatás során használni kívánt fontos paramétert, ezért egy böngésző segítségével hívjuk meg a */var/www/html/moodle/install.php* fájlt, ami a beállításoktól függően valahogy így fog kinézni ➲ *http://localhost/moodle/install.php*. Sajnos itt rögtön fel is fedezhetjük a programban az első hiányosságot, hiszen a bejelentkező képernyő, és szöveg angolul van, és bár a *Moodle* nagyon sok nyelvre le lett fordítva a konfigurációhoz szükséges első lépések csak angol nyelvűek. Itt kattintsunk tehát a *YES* linkre a folytatáshoz, ekkor a Moodle elkészíti a szükséges adatbázis táblákat, majd ha minden sikerült, akkor az oldal alján található *Continue*-ra kattintva folytathatjuk a program beállítást. A következő oldalon a *lang* legördülő menüből válasszuk ki a Magyar nyelvet, és ha nem tudunk angolul, akkor az oldal alján található *Save Changes* gombra kattintva a következő oldal már magyarul szól hozzánk. Az imént átugrott beállításokat pedig később is el tudjuk végezni, amikor a *Moodle* már magyarul „szól hozzánk". A konfiguráció további lépéseit követve könnyen észrevehetjük azt is, hogy a magyar fordítás még nem teljes, és bizony néhány mondat, félmondat angolul van. Remélem ez a felhasználók többsége számára nem okoz gondot. Az eddig elkészült magyar fordítás viszont az átlagos tanulók által használt felületen majdnem teljes, nagyobb fordítási hiányosságok főleg csak a program adminisztrációs, konfigurációs részeinél tapasztalhatóak. Végül megkérdezi tőlünk a *Moodle*, hogy mik legyenek az oldal főbb beállításai. Ha ezeket is megtettük, akkor be kell állítanunk a *Admin* felhasználó tulajdonságait, ajánlatos az *Admin* felhasználó jelszavát nagy körültekintéssel megválasztani. Az *Admin* felhasználó profiljának elmentése után a *2. képhez* hasonló oldal fog fogadni minket.

Alakítsuk ki saját főoldalunkat

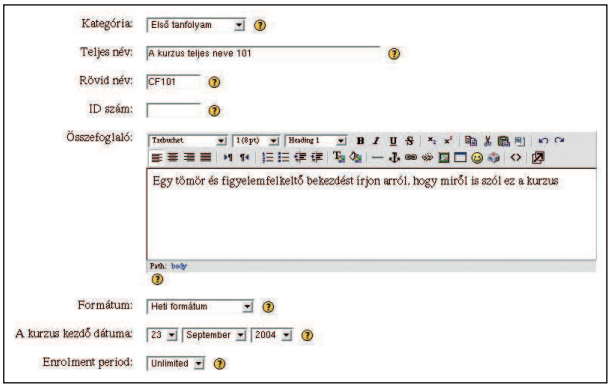
Ha már eddig a pontig eljutottunk, akkor innentől már csak a böngészőnkre lesz szükségünk ahhoz, hogy módosítsuk, bővítsük a tananyagokat, változtassuk az oldal kinézetét, felhasználókat adjunk hozzá az oldalhoz, illetve biztonsági mentéseket készítsünk az oldalról. Ahhoz, hogy az oldal minden funkcióját tudjuk használni IE 5.5-ös vagy annál későbbi verziójú böngészőre lesz szükségünk. Sajnos a programba beépített WYSIWYG szerkesztő a készítők szerint egyéb böngészőkkel nem használható, és saját tapasztalataim is ezt támasztják alá. Amikor *Admin*-ként vagyunk

2. kép Már elérhető a saját Moodle oldalunk

bejelentkezve az oldalra, akkor két lehetséges megjelenítési mód között választhatunk, a normál nézet, a másik a szerkesztési nézet. A normál nézetben ugyanazt az oldalstruktúrát látjuk, amit az egyszerű felhasználók is, amikor bejelentkeznek a rendszerbe, kivéve persze, hogy nekünk van egy adminisztrációs menünk, ahonnan tudjuk menedzselni a program működését. A szerkesztési módot a majdnem minden oldalon található *„Szerkesztés bekapcsolása"* gombbal érhetjük el. Ebben a módban az oldal megjelenése jóval bonyolultabb, ekkor van lehetőségünk arra, hogy az oldal kinézetét megváltoztassuk.
Lehetőségünk van naptárat tenni az oldalra, az aktuálisan az oldalon tartózkodó személyek nevét kiírni, megjeleníteni automatikusan az oldalon történt legfrissebb módosításokat. Akár magunk is helyezhetünk el folyamatosan az oldalon hírek, amelyekből néhányat a *Moodle* mindig automatikusan megjelenít. Ezeket a kis dobozkákat a *„Blocks"* menüpontnál tudjuk hozzáadni a főoldalhoz, ugyanakkor nem csak elhelyezni tudjuk őket az oldalon, hanem az egyes blokkokban található nyilak segítségével meg is tudjuk mondani, hogy hol helyezkedjenek el. Ha egy blokk feleslegessé válik, akkor természetesen minden gond nélkül törölhetjük is, amennyiben nem akarjuk törölni, hanem csak szeretnénk, hogy a felhasználók ne láthassák az adott blokkot, akkor arra is van lehetőségünk.
Miután a főoldal beállításaival megismerkedtünk, most válasszunk az oldalunknak egy egyedi stílust (témát, bőrt), az *Adminisztráció/Beállítások/Stílus* útvonalat végigjárva több, mint 10 különféle színvilág közül választhatunk. Amennyiben nem felel meg nekünk egyik sem, akkor *Moodle* könyvtáron belül a */themes* alkönyvtárban találjuk az összes témát, ebből bármelyikből kiindulva némi *CSS* ismerettel saját témákat tudunk létrehozni.
Sajnos a témák gyártásánál számolnunk kell azzal, hogy vannak még hiányosságok, például a legtöbb linkhez nincs stílus rendelve, így az alapértelmezett stílust használják, aminek következtében hiába szeretnénk, hogy a fejlécben más színű linkek legyenek, mint amilyenek az egyes tanfolyamokra mutató linkek, ez jelen pillanatban nem megvalósítható (holott a fejlécnek szinte mindig más a háttere, mint a tanfolyamoknak).

Tanfolyamok, tananyagok felvitele

Ha már kialakítottuk, hogy hogyan fog kinézni az oldalunk, akkor nekikezdhetünk a tartalom feltöltésének is. A *Moodle* rendszerben a felvitt anyagok tanfolyam kategóriákba,



3. kép Új tanfolyam létrehozása

azon belül tanfolyamokba és a tanfolyamokon belül tananyagokba szerveződnek. Új tanfolyam kategóriát az ***Adminisztráció/Tanfolyamok*** menüpontban tudunk hozzáadni a rendszerhez. Ne ijedjünk meg bár még nem vittünk fel tanfolyam kategóriát, mégis szerepel a tanfolyam kategóriák között egy *„**Egyéb**”* amit sajnos nem tudunk törölni sem, de szerencsére ez a tanfolyam kategória a felhasználók számára nem látható. Adjuk tehát hozzá a tanfolyam kategóriáinkat, ha esetleg más sorrendben szeretnénk, hogy legyenek, akkor a sorrend megváltoztatását a tanfolyam kategóriák neve melletti le és fel nyilakkal könnyedén elvégezhetjük.
A tanfolyam kategóriára kattintva lehetőségünk nyílik tanfolyamokat adni az egyes kategóriákhoz. Most készítsünk egy új tanfolyamot, ekkor a *3. képen* látható oldalt kell kitöltenünk. A teljes név a főoldalon megjelenő neve lesz a tanfolyamnak, a rövid név pedig az oldal fejlécében megjelenő a navigációt segítő link neve lesz. A tanfolyamnak három lehetséges formátum van, az egyik a fórum formátum, amikor a tanfolyam gyakorlatilag egy fórum lap, ahol új fórumokat lehet nyitni, és a régebbiekhez hozzászólni. A második lehetőségünk a tematikus formátum, amikor a tananyagokat fejezetekre bontva tudjuk felvinni az oldalra, és minden fejezet elejére kisebb bevezetőt is el tudunk helyezni, a fejezetek számát a *„**hetek/témák száma**”* legördülő menüből tudjuk egyszerűen megtenni. Ha később az oldalt át szeretnénk szervezni, akkor lehetőségünk van az egyes fejezetek sorrendjének megváltoztatására, sőt tananyagokat is át tudunk vinni egyik fejezetből a másikba, ez hasznos lehet például, amikor bővítjük az oldalunkat, és egy fejezetet kettő, vagy több részre bontunk. A tanfolyam harmadik formátuma lehetővé teszi számunkra, hogy az iskolai, egyetemi oktatás menetét követve hetekre bontva helyezzük el az oldalunkon a tananyagokat.

Tananyag tartalmának felvitele

Miután kialakítottuk az oldalunkon a tanfolyamok megfelelő struktúráját, elkezdhetjük a tananyagok felvitelét.

Az „add a resource” legördülő menüből lehetőségünk van ***HTML*** lapok, és nem formázott egyszerű szöveg felvitelére is. A szöveg felvitelét segíti a beágyazott szövegszerkesztő, bár egy már előre formázott szöveg felvitele során az oldal eléggé szétesheti, például ***OpenOffice***-ból történő másolás következtében. Ezért azt ajánlom, hogy ha lehetséges, akkor a szöveg formázását csak a felvitel után végezzük el, a szöveget ***HTML*** szerkesztővel készítsük el előre, vagy pedig úgy szúrjuk be a html kódot, hogy előtte átváltunk ***HTML*** módba, majd visszaválthatunk WYSIWYG módba, és ekkor már csak néhány kisebb módosítást kell elvégezni. Ha felvittük a tananyagot, akkor változás mentése gombbal elmenthetjük. A tananyagok nem csak egyszerű képekből és szövegből állhatnak, lehetőségünk van matematikai képletek korrekt megjelenítésére, ***Flash*** animációk beszúrására, vagy akár videó, és hang fájlok beszúrására is az tananyagba. Ehhez csak aktiválnunk kell, hogy a ***Moodle*** az oldalon elhelyezett tartalmat felismerje, és megfelelően kezelje.
Az aktiválást az ***Adminisztráció/Beállítások/Filters*** menüpont alatt érjük el. A matematikai képletek beszúrásához a ***TeX notation*** filter szükséges, a multimédia tartalmak lejátszásához pedig a ***Multimedia plugin*** aktiválására van szükség. A ***TeX notation*** gyakorlatilag a ***Latex*** egy egyszerűsített képlet megjelenítője, mely a megadott ***Latex*** kódból képet generál, ezáltal akár bonyolult mátrixokat is lehetőségünk van megjeleníteni az oldalon. A ***TeX notation***-öket két dupla dollárjel közé kell zárni, hogy a ***Moodle*** értelmezze, én ajánlom a következő alakot ***$$ \normalsize{ ide jön a képlet} $$*** mert ellenkező esetben nekem nagyon nagy képletek keletkeztek, persze mindez csak viszonyítás kérdése, mindemellett nem árt ha a képletek méretét az egyéb karakterek mértéhez állítjuk ***Latex*** parancsok segítségével. A kapcsolódó linkek között található egy ***TeX notation*** szerkesztő, mely az első lépéseken segíti át a felhasználót azzal, hogy könnyedén kipróbálhatjuk a beírt kód eredményét. Sajnos a ***Moodle*** nem teszi lehetővé a tananyagokba a ***PHP*** kód beszúrását, ezzel ellentétben ***JavaScript*** beszúrása lehetséges.

Tesztkérdések felvitele

Talán a tananyagok mellett az egyik legtöbbet használt szolgáltatása a ***Moodle***-nek a ***Quiz***, avagy tesztsorok készítése. Lehetőségünk van ugyanis az egyes tananyagok után tesztsorokat elhelyezni az oldalon, ezzel pedig le tudjuk mérni a tanulók tudását, illetve magunk is visszajelzést kapunk arról, hogy a feltett tananyag mennyire érthető, mennyire sikerült rávezetni a tanulókat a tananyag lényegére.
A ***Moodle*** minden egyes kitöltött feladatsort eltárol, és az arra jogosult személyek meg is nézhetik őket, illetve a tanulónak is lehetősége van visszanézni saját kitöltött feladatsorait, tehát az egyszer felvitt megoldások nem vesznek el.
Most nézzük is meg gyorsan, hogy hogyan lehet tesztkérdéseket készíteni. Először is nézzük meg, hogy milyen kérdéseket lehet készíteni a ***Moodle*** rendszer alatt:

3. kódrészlet

```
Melyek az igaz állítások? {
        ~%-50%A bolha nagyobb az elefántnál
        ~%50%A menyét nagyobb a bolhánál
        ~%50%A tigris nagyobb a menyétnél
        ~%-50%A rinocérosz kisebb a menyétnél
}
```

4. kép Lehetőségünk van a kitöltött kérdéssorokat bármikor visszanézni

- igaz-hamis kérdések,
- többszörös válasz, egy kérdésre a felsoroltak közül több helyes válasz is lehet, a cél az összeset megtalálni,
- párosítások, két halmazból kell megkeresni az összetartozó párokat
- rövid kérdések, egyszerű válaszokkal

Láthatjuk, hogy sokféle kérdés típus közül választhatunk, a kérdések formátuma sokféle lehet, ajánlom a ***GIFT*** formátumot. A kapcsolódó linkek között találhatjuk meg ennek a kicsit részletesebb leírását, de adok egy példát is arra, hogy hogyan készítsünk ***GIFT*** formátumú kérdéseket. Nézzük is az egyik legnehezebb többszörös súlyozott válasz készítését. A kérdést először szerkesszük meg egy szövegfájlban, majd pedig töltsük fel az oldalunkra, szerintem ez egy kicsit egyszerűbb, mint a ***Moodle*** beágyazott szerkesztője, illetve előnye, hogy szerkesztése off-line, bármikor elvégezhető. Nézzük meg példaképpen a *3. kódrészletet*, ahol az első válasz nyilván hibás, a második és a harmadik nyilván helyes, a negyedik szintén hibás. A negatív százalékokkal kiküszöbölhetjük azt az esetet, amikor a tanulók mindegyik választ megjelölik, hiszen ekkor a kérdésre adott válasz egyenértékű azzal, amikor a tanuló nem válaszol a kérdésre. Egyvalamire viszont figyeljünk, a ***Moodle*** hibával tér vissza, ha a kérdésre adott válaszok összértéke nagyobb, mint 100%, tehát érdemes a hibás válaszokat negatív százalékponttal „jutalmazni”. Miután elkészítettük a kérdéseinket válasszuk ki a kérdéses tanfolyamot, amelyikbe szeretnénk felvinni, és a kérdéses fejezetnél válasszuk ki az ***Add an activity*** legördülő menüből a ***Quiz*** menüpontot. Töltsük ki a kérdéssor tulajdonságait, név, rövid bevezető, mikortól elérhető, meddig elérhető, de kiválaszthatjuk azt is, hogy a kérdések sorrendjét mindig változtassa, illetve, hogy a kérdésen belül a válaszok sorrendjét is változtassa a ***Moodle***. A folytatás gombra kattintás után válasszuk ki a kérdés kategóriáját, majd megjelenik 3 gomb, válasszuk az importálást, adjuk meg, hogy ***GIFT*** formátumú kérdéseket fogunk feltölteni, majd töltsük fel a kérdéseinket, sikeres feltöltés esetén a ***Moodle*** visszajelzi, hogy hány kérdést sikerült importálni. Miután sikeresen importáltuk a kérdéseinket, hozzáadhatjuk őket a megfelelő kérdéssorhoz. Majd mentsük el az elkészített kérdéssorunkat. Végezzük el próbaképpen a kérdéssor kitöltését, majd a kitöltött kérdéssort visszanézve láthatjuk (*4. kép*), hogy a melyik válaszokat jelöltük meg, és melyek lettek volna a helyes válaszok.

4. kódrészlet

```
#!/bin/bash
#Annak a lemeznek/könyvtárnak a helye, ahova
#szeretnénk elhelyezni a mentéseket
BACKUP_PATH=/mnt/backup_disc;
#A Moodle főkönyvtárának és az adatkönyvtárának
#a helye
MOODLE=/var/www/html/moodle;
MOODLE_DATA=/var/moodledata;
#Az adatbázis hozzáféréshez szükséges
#felhasználó név/jelszó
DBUSER=moodleuser;
DBPASS=password;
#A menteni kívánt adatbázis neve
DBNAME=moodle;
#Ezekbe a fájlokba történik majd a mentés
BACKUPNAME_DB=backup.gz;
BACKUPNAME_MOODLE=moodle_backup.tgz;
BACKUPNAME_MOODLE_DATA=moodledata_backup.tgz;
#Mai dátum a mentés nevéhez
TODAY=`'date +%Y_%m_%d`;

##A Moodle fájlainak elmentése és tömörítése
tar czvf $BACKUP_PATH/$TODAY_$BACKUPNAME_MOODLE
➥ $MOODLE/*;
##A Moodle adatfájlainak elmentése és tömörítése
tar -czvf
➥ $BACKUP_PATH/$TODAY_$BACKUPNAME_MOODLE_DATA
➥ $MOODLE_DATA/*;
##Az adatbázis tartalmának elmentése és
##tömörítése
mysqldump -u$DBUSER -p$DBPASS $DBNAME | gzip - >
➥ $BACKUP_PATH/$TODAY_$BACKUPNAME_DB
```

Biztonsági mentések

A ***Moodle*** rendszer felkínálja arra a lehetőséget, hogy minden egyes tanfolyamot külön paraméterekkel bizonyos időközönként elmentsünk. Azonban ezeket a fájlokat csak a ***Moodle*** rendszeren keresztül tudjuk visszatölteni, például ha valamelyik arra jogosult felhasználó törölt egy tananyagot, tanfolyamot akkor ez a mentés megfelelő. Ugyanakkor ha ***Moodle*** oldalunk már elég nagy, és sok kényes adatot tartalmaz, akkor célszerű az egész adatbázist és a hozzátartozó fájlokat egy-egy ***tar.gz*** fájlba elmenteni. Ehhez jó elindulás lehet a *4. kódrészlet*, amelyet mondjuk ***cron***-ból meghívva minden nap elkészíti a biztonsági mentést a teljes oldalról, a fájlok neve elé a szkript az aktuális dátumot beszúrja, ezért bármikor könnyedén visszaállíthatjuk az oldalt pontosan arra a napra, amelyikre szeretnénk.

Összegzés

Sajnos a rendelkezésre álló hely nem elég arra, hogy részletekbe menően bemutassam a *Moodle* rendszer képességeit, nagyon sok képességéről a *Moodle*-nek még csak említést sem tettem (pl. *scrom*, *wiki*) melyek tovább segítik a munkánkat az e-learning rendszerek létrehozása során. Ellenben egy olyan cikket szerettem volna írni, amely egyáltalán bemutatja, hogy mi is a *Moodle*, hogyan tudunk magunknak akár otthon is, a fenti lépéseket követve, összeállítani egy e-learning keretrendszert, hogyan lépjünk át az első nehézségeken, amik a telepítése során várnak ránk.

Remélem, hogy amikor itthon is felismerik az e-learning rendszerek fontosságát, akkor a *Moodle* felhasználók száma ugrásszerűen fog emelkedni, hiszen ideális non-profit szervezetek, iskolák, főiskolák, egyetemek számára ahol a program ingyenessége, továbbfejleszthetősége nagy előny, de profitorientált oktatási portál is létrehozható, hiszen az 1.4-es verzióban már benne van a *Pay-Pal* modul, mely segítségével akár pénzt is kérhetünk a távoktatásért.

Horváth Ernő (he305@hszk.bme.hu).
23 éves, jelenleg a BME-VIK utolsó éves hallgatója. Két évvel ezelőtt ismerkedett meg komolyabban a Linux rendszerekkel és emellett érdeklődik még a robotika és a biztonság technika iránt is. Ha lenne szabadideje sokat kirándulna, és filmeket nézne.

KAPCSOLÓDÓ CÍMEK

A Moodle honlapja: ➲ http://www.moodle.org

Néhány magyarországi Moodle weboldal:
➲ http://www.biztostu.hu

➲ http://edutech.elte.hu/moktato/

➲ http://www.ollejanos.hu/moodle/

Magyarországi Moodle felhasználók fóruma:
➲ http://edutech.elte.hu/moktato/course/view.php?id=21

Angliai Moodle oktatóportál integrált felolvasóprogrammal vakok számára:
➲ http://kconline.kilmarnock.ac.uk/moodle/

TeX Notation szerkesztő:
➲ http://www.matheboard.de/formeleditor.php

Segítség a GIFT formátumú kérdések szerkesztéséhez:
➲ http://learn.uci.edu/cms/help.php?module=quiz&file=formatgift.html

Hálózatok (11. rész)

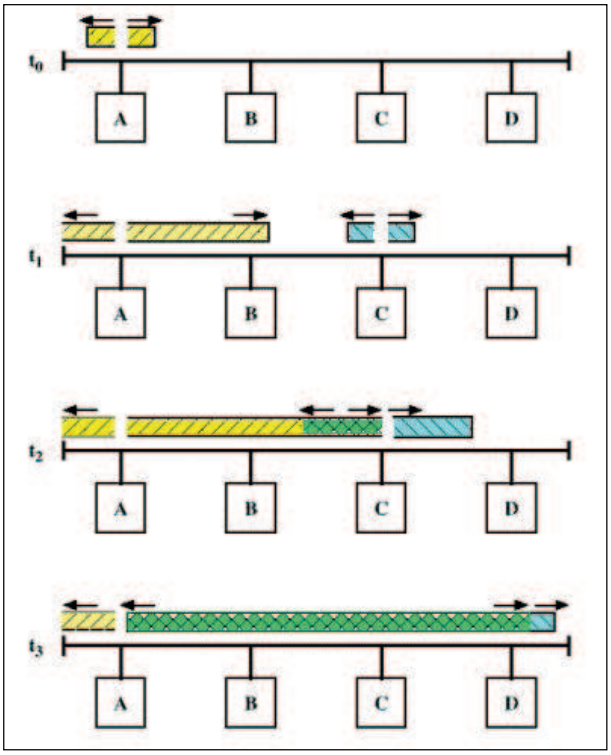
Kettes exponenciális visszalépéses algoritmus, kapcsolók, egyéb szabványok

Legutóbb az Ethernet szabvánnyal foglalkoztam, de adós maradtam néhány dologgal. Ebben a cikkben egyrészt ezeket pótolom, másrészt egy újabb, izgalmas területre vezetem be az olvasót: a kapcsolók világába.

Legutóbb az ***Ethernettel***, illetve az ***IEEE 802.3***-as elnevezésű szabvánnyal foglalkoztunk. Amikor egyszerre több állomás osztozik egy csatornán, akkor a legfontosabb megoldandó probléma a keretek ütközésének elkerülése. Meg kell akadályozni, hogy egyszerre két állomás próbáljon adni. Az előző két részben erre több megoldást is láthattunk, amelyek közül a gyakorlatban az egyik leghatékonyabbnak a ***CSMA/CD*** bizonyult. Segítségével az állomások nem csak a csatornán lévő forgalmat képesek érzékelni (azaz meg tudják állapítani, hogy az adott pillanatban éppen beszél-e valaki), hanem a keretütközést is (vagyis azt, ha egyszerre két állomás forgalmaz). Ha az utóbbi két eset közül valamelyik fennáll, akkor az állomás véletlenszerű ideig várakozik, majd ismét megpróbálkozik a keret elküldésével.

Adós maradtam azzal, hogy miként is érdemes meghatározni azt a véletlenszerű értéket, amennyit az állomásoknak várakozniuk kell. Vessünk egy pillantást az *1. ábrára*, amely már az előző részben is szerepelt. Ezen egy keretütközést örökítettünk meg, ahol az A állomás a ***t0***, a ***C*** pedig a ***t1*** időpillanatban kezdi meg egy keret adását. A csatornán a keretek a ***t3*** időpontban „találkoznak”. Könnyen beláthatjuk, hogy a keret forgalmazásának elkezdése és az ütközés pillanata között ***t3 – t1*** idő telt el. Az A csak akkor érzékeli az ütközést, mikor az ütközés helyétől visszaérkezik hozzá a jel, amely dupla ennyi időt vesz igénybe. Ha ezeket összeadjuk, megkapjuk, hogy az adás megkezdése és az ütközés érzékelése között ***2 * (t3 – t1)*** idő telt el. A ***t3 – t1*** értéke az ütközés helyétől függ, de mivel a ***802.3*** szabványban meg van határozva a hálózat maximális mérete, ezért kiszámolhatjuk, hogy mekkora lehet a ***t3 – t1*** legnagyobb lehetséges értéke. Ez pontosan ***51,2 µs***. Érdemes tehát az ütközés pillanata után az időt diszkrét egységekre felbontani, és ezt az értéket megválasztani időrésnek. A véletlenszám-generátor csak azt döntse el, hogy hány ilyen időrésnyit várakozzunk. Az igazi kérdés az, hogy mekkora legyen az az intervallum, amelyből véletlenszerűen kiválaszthatunk egy értéket.

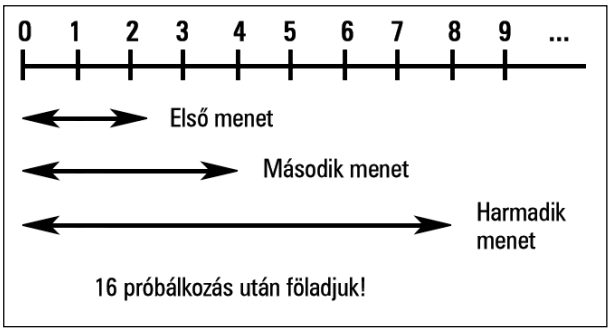
Ha ez az intervallum kicsi, akkor nagy valószínűséggel



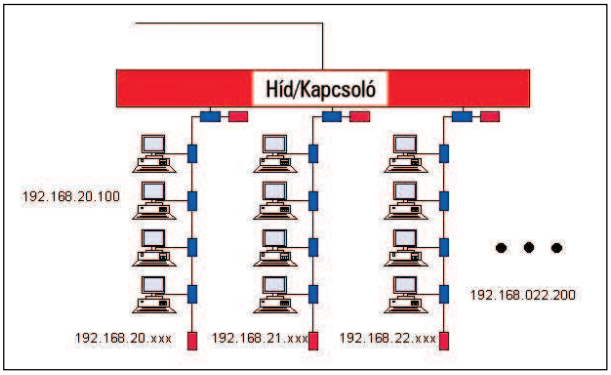
1. ábra Keretütközés

bekövetkezhet egy újabb ütközés (mivel nagyobb az esélye annak, hogy a konkurens állomás is ugyanazt az értéket választotta). Ha csak 0 és 1 lenne a lehetséges választás, akkor mondjuk 50 egyszerre forgalmazni kívánó állomás közül csak akkor adhatna valaki, ha 49-en egyszerre választották a 0-t, és csak ő egyedül az 1-et (vagy fordítva). Nem sokan szeretnének egy olyan hálózatban dolgozni, ahol akár hónapokat is várni kellene egy-egy keret elküldésére.

Az intervallum növelésével azonban az ütközések valószínűsége csökkenthető. Ha például az állomások 0 és 1023



2. ábra Az intervallum mérete minden ütközés alkalmával exponenciális ütemben változik



3. ábra A kapcsolók csak annyit követelnek meg, hogy szabványos keretek érkezzenek a bemeneteire. Így akár egy hub-ot is rákötheténk, és akár több LAN-t is összeköthetünk a segítségükkel. Ilyenkor a kapcsolót hídnak nevezzük.

közötti időrésnyi időt várakozhatnak, akkor a lottót is megszégyenítő eséllyel fordulhatna csak elő egy újboli keretütközés. Itt más probléma merül fel: túl nagy a késleltetés. Szerencsétlen esetben egy állomásnak 1023 időrésnyi időt kéne végigunatkoznia, mikor már 1 időrés elteltével vígan forgalmazhatna. Még átlagos esetben is több száz időrés a késleltetés, ami nagyon nagy hatékonyságbeli romlás. Kell tehát egy olyan algoritmus a várakozás időtartamának meghatározására, amely kis valószínűséggel okoz keretütközéseket, mégsem nő szakállunk, amíg a keret elküldésére várunk. Ilyen módszer például a ***kettes exponenciális visszalépés (binary exponential backoff)***. A működése a következő: az állomás először 0 vagy 1 időrésnyit vár, majd ismét megpróbálkozik az adással. Ha mindkét vetélkedő állomás ugyanazt az értéket választja (amelynek a valószínűsége 0,5), akkor ismét sorsolnak, de most már 0 és 3 közötti értékek közül. Ekkor az ütközés valószínűsége már csak 0,25. Ha azonban még sincs szerencsénk, akkor a választás 0 és 7 közül történik. Láthatjuk, hogy minden ütközés után a sorsolandó értékek száma exponenciálisan nő. Az ***i***-edik ütközés után 0 és 2^i - 1 közötti időrésnyi időt kell várakozni. Fontos azonban, hogy ez az érték nem nő 1023 fölé és a 16. ütközés után a hálózati kártya vezérlője általában hibajelzést küld, mivel valószínűsíthető, hogy valami probléma van a hálózattal *(2. ábra)*.

Ennek az algoritmusnak az az előnye, hogy tekintettel van a konkurens állomások számára. Ezt azzal érjük el, hogy az ütközések hatására az intervallum méretét exponenciálisan növeljük. Ezzel biztosítjuk, hogy egyrészt nagyszámú vetélkedő állomás esetén se kelljen sokáig várnunk a forgalmazásra, másrészt fordított helyzetben (kevés állomás esetén) sem kell számolnunk nagy késleltetéssel.

Kapcsolt LAN-ok

Ahogy nő a hálózatba kapcsolt állomások száma, a csatorna telítődni fog. Ebben az esetben meg kell fontolnunk, hogy beszerzünk egy úgynevezett ***kapcsoló*** vagy ***switch*** nevű mágikus dobozt *(4. ábra)*.

Míg a hub-ok csak arra voltak képesek, hogy a beérkező jelet az összes kimenetre elküldjék, addig a kapcsolók már képesek betekinteni a keretek tartalmába. Ennek köszönhetően az egyik porton beérkező keretet nem küldik tovább az összes többire, csak arra, amelyikre a célpont is csatlakozik.

A kapcsoló lelke egy belső áramkör, amelyre úgynevezett vonali kártyák csatlakoznak, amelynek száma 4 – 32 között változhat. Egy-egy kártyán 1-8 csatlakozó található. Ezektől a csatlakozóktól rendszerint csavart érpár vezet az állomásokhoz.

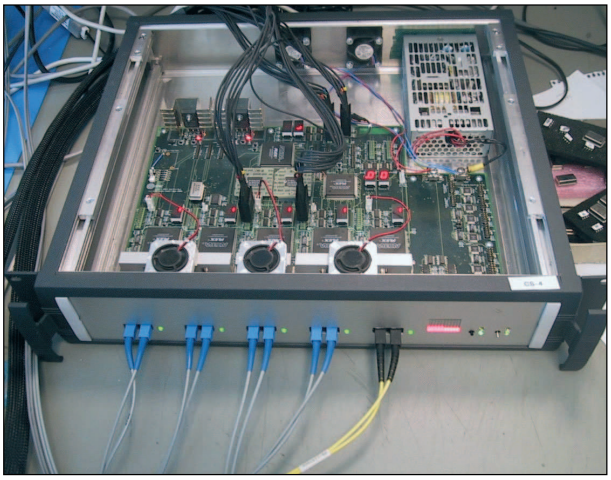
Nézzük, hogyan is zajlik a kommunikáció a kapcsoló segítségével. Tegyük fel, hogy beérkezik egy keret az egyik portra. Ekkor a vonali kártya belenéz a keretbe, hogy vajon ki a címzett. Szerencsés esetben a címzett is ugyanarra a vonali kártyára van kapcsolva, így egyszerűen a keretet csak a megfelelő portra kell másolni. Ha a címzett egy másik kártyán van, akkor előtte át kell oda vinni. Ezt a kapcsoló hátlapján lévő átviteli panel végzi, amely körülbelül 1 Gb/s-os sebességgel dolgozik, és egy belső protokollt használ. A hálózat állomásainak nem kell foglalkozniuk azzal, hogy melyik állomás melyik vonali kártyára kapcsolódik.

Érdekes kérdés, hogy mi történik akkor, amikor két vagy több, ugyanarra a vonali kártyára kapcsolódó állomás egyszerre kezd forgalmazni. A megoldás ilyenkor a vonali kártyák kialakításától függ. Az első megközelítés az, hogy a kártyán kialakított portok egymással össze vannak kötve, így egy közös csatorna alakul ki az ugyanahhoz a kártyához csatlakozó állomások között. Ez azt jelenti, hogy egy adott kártyához kapcsolódó állomások közül mindig csak az egyik adhat, különben ütközés lép fel, ugyanakkor a kapcsolóban lévő többi vonali kártya egymástól függetlenül működhet. Ezt úgy szokás mondani, hogy minden kártya saját ***ütköztetési tartománnyal*** rendelkezik.

A másik (valamivel drágább) megközelítés, hogy nem csak a kártyák, hanem a portok is függetlenek egymástól. Minden port rendelkezik egy saját pufferrel, ahova a beérkező keretek kerülnek. Ezután a kapcsoló belső programja ***(mert ilyen is van neki)*** dönti el, hogy melyik keretet mikor és hova továbbítja. Láthatjuk, hogy itt már a portok is külön ütköztetési tartománnyal rendelkeznek. Ha minden porthoz csak egy állomás csatlakozik, akkor a hálózaton ismeretlen lesz a keretütközés fogalma, és ezáltal nagyságrendekkel gyorsabb átvitelt érhetünk el, mint például egy koaxiális kábelt használó ***10Base2*** hálózaton.

Fontos megjegyeznünk, hogy a portokra nem csak egy állomást csatlakoztathatunk, hanem például egy hubot is, amely több állomást köt össze. Ilyenkor azonban már a kap-

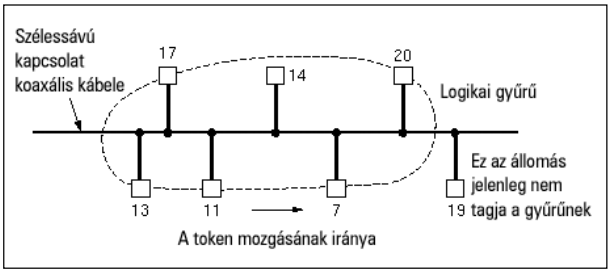
csolót két ugyanolyan szabványú ***(802.3)*** hálózatot összekötő hídnak is nevezhetjük *(3. ábra)*. A hidakkal még ebben a részben részletesen is foglalkozunk.
Az olvasó most valószínűleg arra gondol, hogy a kapcsoló nem csak valami felfejlesztett hub, hanem valójában egy számítógép. És valóban! A kapcsolónak van processzora, memóriája, perifériája ***(például a portok)***, sőt még operációs rendszere is. Alaptalan tehát az a népszerű feltételezés, hogy a kapcsoló valójában egy hub, csak sokkal gyorsabb. Igazából semmi közük egymáshoz. Míg a hub csak a beérkező jeleket ismétli minden irányban, addig a kapcsoló a beérkező kereteket, és csak arrafelé, amerre a keret címzettje van.
Hogy egy kapcsoló valóban hatékonyan működjön, szükség van egy fejlett belső szoftverre. Hogy ez a szoftver miként is végezze a feladatát ***(a keretek továbbítását)***, az nem egy egyszerű kérdés. Vannak kapcsolók, amelyek például a keretnek csak az első 16 bájtját olvassák be, amelyből már kiderül, hogy ki is a címzett. Amint kinyerték ezt az információt, egyből továbbítják is a megfelelő portra. Másfajta megközelítés a ***tárolás és továbbküldés (store and forward)*** módszer, amely az egész keretet beolvassa, és ellenőrzi annak sértetlenségét. Ez ugyan erőforrás igényes feladat, viszont előre kirostálhatóak a hibás keretek, így ezzel is csökken a hálózat terheltsége. Ilyenkor azonban a kapcsolónak meg kell kérnie az adót, hogy ismételje meg az eldobott keretet.
A kapcsolók a hálózatok gerincét képezik, ezért a szoftvernek illik másról is gondoskodnia. Például arról, hogy gonosz emberek ne terhelhessék túl ***(flood)*** a hálózatot. A legegyszerűbb ilyen eljárás az ***üzenetszórási vihar (broadcast storm).*** Ilyenkor a támadó rövid idő alatt rengeteg üzenetszórási (broadcast) csomagot indít útnak. Mivel ezek mindenkihez szólnak, a kapcsoló egy ilyen kerettel szemben úgy viselkedik, mintha egy közönséges hub lenne, vagyis minden portján továbbítja azt. Ezért a jobb kapcsolókban megtalálható a ***Broadcast Storm Control*** nevű szolgáltatás, amely minden porton egy időegység alatt csak meghatározott százalékban enged át ilyen jellegű üzeneteket. (Sajnos ez a védelem is kijátszható úgy, hogy broadcast üzenetek helyett olyan rendes kereteket küldünk, amelyek egy nem létező géphez szólnak és ezt a cél címet folyamatosan változtatjuk).
Persze a kapcsolónak nemcsak a hálózatot, hanem saját magát is védenie kell. Mivel ez is egy számítógép, ezért betelhet a memóriája, túlterhelődhet a processzora, stb., tehát a támadónak maga a kapcsoló is ideális célpontja lehet.
A processzort leterhelni például sok keret rövid idő alatt történő elküldésével lehet. A jobbfajta szoftverek ilyenkor szólnak a forgalmazó hardvernek, hogy lassabban küldjön. Ez megfelelő védelem, mivel csak nagyon régi hardverekkel lehetne kikerülni, amelyek még régebbi szabvány szerint készültek, ahol nem volt lehetőség a forgalom szabályozására. Persze vannak olyan kapcsolók is, amelyek egyszerűen csak lefagynak, esetleg eldobnak minden feldolgozhatatlan csomagot, vagy feladják, és úgy kezdenek működni, mint egy közönséges hub.
Ha már szóba került a kapcsolt hálózatok biztonsága, érdemes eloszlatnunk még egy félreértést, mely szerint a kapcsoló teljes védelmet nyújt a hálózaton folyó kommunikáció lehallgatása ***(sniffelés)*** ellen. Később látni fogjuk, hogy ez mennyire nem igaz. Léteznek olyan technikák, amellyel átejthetjük a kapcsolót, és elhitethetjük vele, hogy mi nem azok vagyunk, akik. Egy kis ügyeskedéssel rávehetjük, hogy a másnak szóló üzeneteket nekünk továbbítsa. Ezek a technikák az úgynevezett ***ARP Cache Poisoning*** nevű eljáráson alapulnak, amely nem jelent mást, mint hogy kilétünkről össze-vissza hazudunk mindenkinek. Annyit elöljáróban elárulunk, hogy az ***ARP*** egy olyan protokoll, amely segít az adatkapcsolati rétegnek megtudni, hogy melyik MAC cím felel meg a kérdéses IP címnek (mivel az adatkapcsolati réteg nem tudja az IP cím segítségével azonosítani az állomásokat). Mivel az ***ARP*** nem követel meg semmiféle hitelesítést, ezért nem lehet ellenőrizni, hogy mindenki becsületesen válaszolt-e. Ezt használják ki a ***Men in the Middle*** elnevezésű támadásoknál is. Később, az IP protokoll ismertetése után egy teljes részen keresztül foglalkozunk majd a támadásokkal, és azok ellenszereivel.



4. ábra Egy kapcsoló belülről...

Más szabványos LAN-ok

Eddig a ***802.3***, illetve az ***Ethernet*** nevű szabvánnyal foglalkoztunk. Ez a világ legelterjedtebben használt hálózattípusa. Méltán tett szert erre a népszerűségre, ugyanis itt az állomásokat kedvünkre csatlakoztathatjuk, illetve távolíthatunk el anélkül, hogy le kéne állítanunk a hálózatot. Másik szimpatikus tulajdonsága, hogy ha nincs leterhelve a hálózat, akkor a keretek küldésének késleltetése nem számottevő.
Az éremnek azonban két oldallal van, így az ***Ethernet*** is rendelkezik hátrányokkal. Az első, hogy az alapvető komponensek analógok. Fel kell fedeznünk a keretütközéseket, ezért nem küldhetünk 64 bájtnál kisebb keretet, hiába akarunk csak 1-1 bájtot átküldeni ***(például terminálok esetén)***.
A legnagyobb hiányosság mégis az, hogy az ***Ethernet*** hálózatok nem determinisztikusak. Ez azt jelenti, hogy gyakran csak a szerencsén múlik, ki mikor forgalmazhat. A dolgok például úgy is alakulhatnak, hogy egy állomás egész hosszú ideig nem képes a keretét útnak indítani. Ilyen viszonyok között szinte lehetetlen bármiféle valós idejű feladatot elvégezni.
Speciális esetekben – például egy gyártósor automatizálásakor – az ***Ethernet*** hálózat tehát nem jelenthet megoldást.



5. ábra A vezérlőjeles sín hálózat

Ezért született meg egy másik, az ***Ethernettel*** megegyező technológiát használó, ám azzal egyáltalán nem kompatibilis szabvány, a 802.4, vagy más néven a ***vezérlőjeles sín (bus token)***.

A legfontosabb eltérés a ***802.3*** és a ***802.4*** között, hogy az utóbbiban kiszámolható egy legrosszabb esetbeli viselkedés. Az ***Ethernet*** esetében nincs felső korlátja annak, hogy egy állomás mikor küldheti el a keretét. Elvileg az is megtörténhet, hogy soha sem kerül erre sor. Egy gyűrű topológia esetében már kiszámolható egy legrosszabb érték. Ha ***N*** darab állomás van, és egy keret elküldéséhez ***T*** idő szükséges, akkor minden állomás legfeljebb ***N * T*** idő múlva útnak indíthatja keretét.

A gyűrű alakú topológiával azonban rengeteg baj van. Egyrészt, ha valahol megsérül, az kihat az egész hálózatra. Másrészt, amikor egy gyártósort elképzelünk, valahogy nem a kör alak jut az embernek eszébe. Ésszerű ötlet volt tehát, hogy a ***802.4*** fizikailag egy ***Ethernet*** hálózatra emlékeztessen, viszont logikailag egy gyűrűt alkosson. Így megmaradt a legrosszabb eset, azaz, hogy N * T idő alatt minden állomás biztosan el tudja küldeni az üzenetét.

A könnyebb megértés végett vessünk egy pillantást az *5. ábrára*. A vezérjeles sín egy koaxiális kábel, amelyre az ***Ethernet*** hálózatok jól ismert módján kapcsolódnak az állomások. Ez a hálózat fizikai elrendezése. Logikailag azonban egy gyűrűt alkotnak: minden gép ismeri a közvetlen jobb, illetve baloldali szomszédjának MAC címét. Amikor a kommunikáció elindul, akkor először a gyűrűben lévő első gép küldhet. Miután elküldte a keretet, egy vezérlőjelet, úgynevezett tokent küld a szomszédjának. A hálózatban mindig csak az küldhet, akinél éppen a vezérlőjel van. Ennek köszönhetően nem léphet fel keretütközés. A vezérlőjel körbejár az egész gyűrű mentén.

Fontos, hogy a gépek gyűrűbeli sorrendje teljesen független a fizikai elhelyezkedéstől. Itt is adatszórásos közeget használunk a kommunikációra, tehát az üzenet mindenkihez el fog jutni. A vezérlőjel pedig nem más, mint egy olyan keret, amelyet az adott állomás a gyűrűbeli közvetlen szomszédjának címez. Erre értelemszerűen csak a címzett fog figyelni. Az állomásokat ezért nem érdekli, hogy fizikailag ki hol helyezkedik el. Sőt, lehet olyan állomásunk is, amely nem részese a logikai gyűrűnek. Ilyenek például a hálózatra frissen felkapcsolt állomások. Ezek nem kerülnek bele automatikusan a gyűrűbe. A logikai gyűrű karbantartása (például állomások felvétele, kitörlése, illetve a sorrend meghatározása) a hálózat közegelérési ***(MAC)*** protokolljának feladata. Ez egy rendkívül összetett protokoll, részletes ismertetése meghaladná e sorozat kereteit.

Meg kell említenünk, hogy természetesen olyan hálózatok is léteznek, amelyek fizikailag is gyűrűhálózatok. Ezek közül a legjelentősebb az 1970-es években, az IBM által kifejlesztett ***vezérlőjeles gyűrű (token ring)***. Valójában ez sokkal szélesebb körben elterjedt LAN, mint az előző, az ***Ethernet*** után a második legnépszerűbb. Az ***IEEE*** is felvette saját szabványai közé, és a ***802.5*** „kódnevet" ragasztotta rá.

Mit is mondhatunk el általánosan ezekről a hálózatokról? Az állomások értelemszerűen egy gyűrűre vannak felfűzve, egymással kétpontos csatornával vannak összekötve. A hálózat szemszögéből a csatorna típusa lényegtelen, lehet akár optikai vagy csavart érpár is. Amikor egy állomás keretet küld, az az egész gyűrűn körbehalad. Mivel a jelek gyorsan terjednek, ezért a keret eleje hamarabb visszaérkezik a feladóhoz, mint ahogy ő befejezné annak küldését. A célállomás a keret utolsó két bitjét változtatja meg: az egyiket akkor, amikor szembesült azzal, hogy a kérdéses keretet neki küldték, a másodikat pedig akkor, amikor sikeresen vette is azt. Amint a keret vége visszaért a feladóhoz, az értesülhet arról, hogy üzenete sikeresen célba ért-e. Ha igen, akkor leveheti a keretet a gyűrűről.

Az állomások itt is csak a megfelelő sorrendben adhatnak, és erről is a vezérlőjel gondoskodik, amely egy speciális keret. Minden állomás csak egy meghatározott ideig birtokolhatja a tokent, utána kötelezően tovább kell adnia. Persze ha éppen nincs mit küldenie, akkor a vezérjelet előbb is átadhatja. Lehetőség van arra is, hogy az állomások között egyfajta prioritási sorrendet állítsunk fel. Ennek a legegyszerűbb módja az, hogy bizonyos állomásoknak megengedjük a token hosszabb ideig történő megtartását.

Amikor nincs forgalom, akkor elég unalmas az élet a gyűrűhálózatokban: csak a vezérlőjel szaladgál körbe-körbe. Ha azonban nagy a terhelés, minden állomásnál már sorban állnak az elküldésre váró keretek, akkor amint elengedi valaki a vezérlő jelet, a következő állomás máris lecsap rá. Ily formán a hálózat kihasználtsága közel van a 100%-hoz.

Persze az élet nem ilyen egyszerű. A legnagyobb problémát egy-egy állomás meghibásodása jelentheti, amely kihathat az egész hálózat működésére. A problémát megoldhatjuk az úgynevezett huzalközpont segítségével. Ilyenkor a fizikai gyűrű megvalósítás helyett az állomásokat két csavart érpár segítségével a huzalközponthoz kapcsoljuk. Az egyik kábelen az állomásoktól érkező adatok haladnak, a másikon pedig azok, amiket mi küldünk feléjük. Amikor meghibásodás történik, a központ rövidre zárja a két kábelt, így a hálózat továbbra is működőképes marad.

Ezt általában úgy szokás megoldani, hogy a reléket, amelyek a rövidre zárásért felelnek, az állomások árammal látják el. Amikor egy állomás meghibásodik, illetve a gyűrű megszakad, akkor a relé nem kap több áramot, és ennek következtében zár. Lehetőség kínálkozik arra is, hogy ezeket a reléket szoftveresen vezéreljük.

A következő részben a hidakkal folytatjuk, amelyek segítségével különböző típusú LAN-okat kapcsolhatunk össze.

Garzó András
garzo@interware.hu

Biztonsági játszmák

Miután megküzdöttünk a hagyományos operációs rendszerek miatt a hálózatunkat elözönlő férgekkel és vírusokkal, kellemesen lazíthatunk egy kis játékkal.

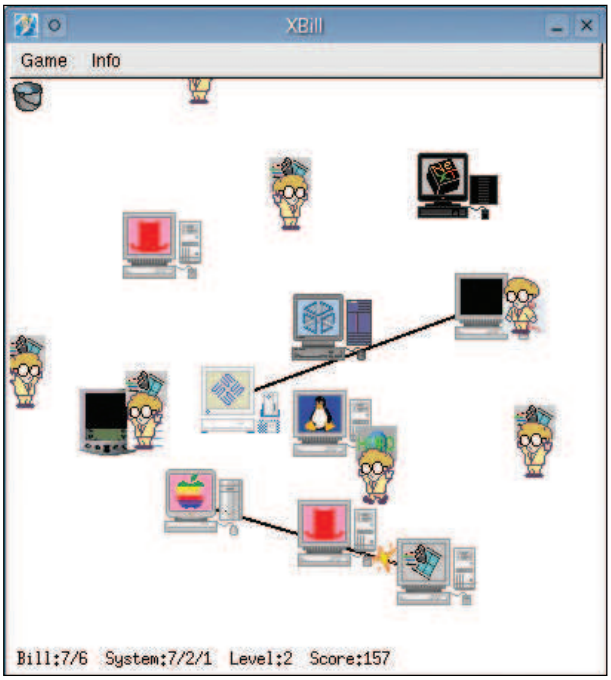
Kiváló, kedves ***François***, úgy látom, nálad máris működésbe léptek azok a vírusok. Sose bízd el magad, barátom. Néhány pillanat, és a csorda máris kiüríti a receptes könyvtáradat. Tüzelj, ***François***, tüzelj! Tudom, úgy tűnik, sosem fogynak el, de ki mondta, hogy a jó rendszerbiztonság azt jelenti, hogy hátradőlhetsz, és nyugodtan nézelődhetsz? Ez háború, ***François***, ezek a dögök egyetlen perc nyugalmat nem fognak hagyni neked! Figyelj…! Jobbra, rajta! Áhh, túl késő. Szép próbálkozás, de most én jövök.

Hogy mit gondolok? A vendégeink bármelyik pillanatban megérkezhetnek. Készítsd elő az asztalokat, és ellenőrizd, hogy a számítógépek is készen állnak-e. Munkára, ***François***! Már itt is vannak. Üdvözöl benneteket, ***mes amis***, a ***Chez Marcel***, a finom étkek, a jó borok és a ***Linux*** háza. Foglaljatok helyet, helyezzétek magatok kényelembe. Van néhány érdekes programunk, szeretnénk megmutatni és kipróbálni őket… ***François***, indíts a pincébe! A déli szárnyban találsz egy láda 1992-es olasz ***Toscana Vin Santot***. Hozd fel a vendégeknek, azonnal!

Talán már sejtitek, ***mes amis***, ebben a hónapban a biztonságra összpontosítunk. Mindannyian tudjuk, hogy a ***Linux*** tervezésénél és természeténél fogva messze biztonságosabb, mint a megszokott asztali operációs rendszerek. Nyilván tudjátok, hogy elsősorban melyikre gondolok. A különbség szembeszökő, ha láttátok már, ne adj' isten használtátok is. Figyelembe véve annak a másik operációs rendszernek a biztonsági problémáit (valamint a mögötte álló terjeszkedjünk, fojtsuk meg és foglaljuk el szemléletet), nem meglepő, hogy a té ma számos játékban is megjelenik. Mai menünk ilyen fogásokból áll – nevezhetjük őket a rendszerbiztonság könnyedebb oldalának.

Mai kínálatunk első játékát a ***Chez Marcelben*** már felszolgáltuk egyszer, mégis érdemes újra megemlíteni. ***Brian Wellington*** és ***Matias Duarte XBill*** nevű programjáról van szó. Azt hiszem, kiváló kezdő fogás lesz mára. Az ***XBill*** sokak számítógépére már eleve telepítve van. Indításához csupán az `xbill` parancsot kell kiadni. Ha nincs telepítve, nézzük át terjesztésünk CD-lemezét, esetleg töltsük le.

Alapötlete roppant egyszerű. Feladatunk a világ számítógépes hálózatainak megmentése a gonosz, ***Bill*** nevű vírusírótól, aki – okos módon operációs rendszernek álcázva iromá-

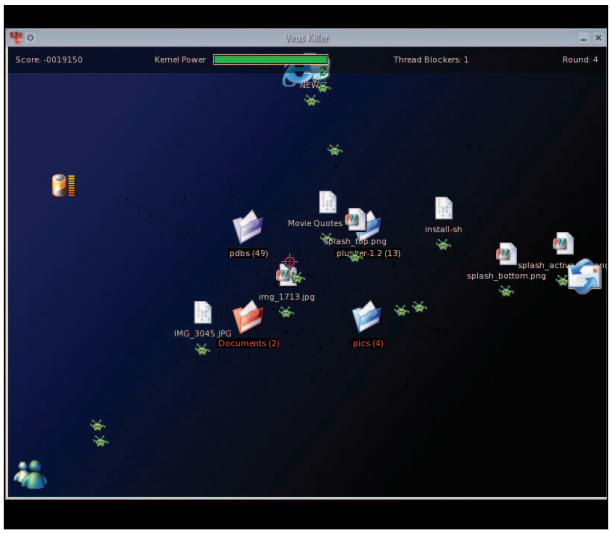


1. ábra XBill: Vajon meg tudjuk állítani Billt?

nyát – megpróbál vírust tölteni az amúgy egészséges számítógépekre. Az egész játéktéren a gonosz klónjai tűnnek fel, és ellopják az operációs rendszereket. (Lásd az *1. ábrát*.) Csak úgy akadályozhatjuk meg őket ebben, hogy rájuk kattintunk az egérrel. Gyors kézre lesz szükségünk!

Egyszer régen létezett egy roppant népszerű, ***Defender*** nevű játék, amelynek idővel jó néhány utánzata – vagy tisztelője, szemlélet kérdése – jelent meg. Az egyik ilyen ***Bill Kendrick Defendguin*** programja, amelyben a hősi ***Tux*** gonosz földönkívüliek ellen harcol, akik egy békés, pingvinekkel benépesült bolygót lakóit le akarják igázni, és szörnyű pingvinoidokká akarják változtatni. Ezek a földönkívüliek, akik amúgy nagyon hasonlítanak az ***XBill Billjére***, a közösség tagjait repülő gyilkológépekké alakítják. Ha egy pingvin egyszer átalakult pingvinoiddá, akkor számára többé nincs remény, le kell lőni. Még jobb, ha a betolakodókat pusztítjuk el, értük pont is több jár (*2. ábra*).

2. ábra Marcel újabb hajót veszít el Defendguinezés közben



3. ábra Virus Killer alatt minden vírust el kell pusztítanunk, mielőtt megsemmisítenék a fájljainkat

A ***Defendguin*** oldalon számos terjesztéshez találunk előfordított binárisokat, valamint a forráskódot is letölthetjük. A ***Defendguin*** lefordításához szükség van az ***SDL*** fejlesztői könyvtárakra. Ha a forrás mellett döntünk, akkor sem kell semmi bonyolultra számítanunk, csak ki kell bontanunk a csomagot, majd ki kell adnunk a `make` és a `sudo make install` parancsot.
A játék indításához a `defendguin` parancsot kell kiadni – előtte fárasztó, pörgős menetre készüljünk fel. Ha teljes képernyős módban akarunk küzdeni, használjuk a `-f` kapcsolót.
A legtöbb korszerű, azt a bizonyos másik operációs rendszert használó irodában soha véget nem érő harcot vívnak a vírusok ellen. Az összes könyvtár és fájl biztonságban tartása a veszélyes betolakodóktól bizony kevéssé tűnik játéknak azoknak, akik ezt a harcot minden egyes nap kénytelenek felvenni. Ha egy pillanatra megállunk, azonnal rájövünk, hogy mindez nagyszerű alapötlet egy játékhoz – így gondolta ezt ***Stephen Sweeney***, a ***Virus Killer*** készítője is. A háttértörténet annyi, hogy vírusok támadnak azokon a biztonsági réseken keresztül, amelyek ennek a másik operációs rendszernek a natív alkalmazásaiban találhatók. Fájljaink veszélyben forognak, és mi vagyunk az egyetlenek, akik útját állhatják a rombolásnak.
A ***Virus Killert*** a program weboldaláról, bináris és forrás RPM-ek formájában tölthetjük le (lásd a forrásokat).
A ***Virus Killer*** szintén ***SDL*** alapú, lefordításához szükségünk lesz az ***SDL_image***, az ***SDL_mixer*** és az ***SDL_ttf*** fejlesztői könyvtárakra. Ellenőrizzük, hogy gépünkön a ***zziplib*** és a hozzá tartozó fejlesztői könyvtárak is megtalálhatók-e. Ha minden könyvtár a helyére került, a telepítés már gyerekjáték:

```
tar -xzvf viruskiller-0.9-1.tar.gz
cd viruskiller-0.9-1
make
sudo make install
```

Ha játszani szeretnénk, a `viruskiller` parancsot kell kiadnunk. A program kezdőkönyvtárunk tartalma alapján létrehoz egy fájl- és könyvtárlistát, a játék során ezek lesznek a vírusok célpontjai. Feladatunk az, hogy kilőjük az apró szörnyeket, mielőtt bármi kárt tehetnének. Fontos, és a weboldalon lévő GYK is kiemeli, hogy a program semmiben nem tesz semmilyen kárt, hiszen csak egy játék. Ha nem bírjuk annak látványát, hogy pici zöld szörnyecskék elvonszolják magukkal a sok órai munkánkat képviselő dokumentumokat, akkor a következő paranccsal indítsuk a programot:

```
viruskiller +safemode
```

Ilyenkor a fájl- és könyvtárlista a rendszer ***/tmp*** könyvtárának tartalma alapján jön létre, és nem fogunk éjjel felriadni attól, hogy a zöld dögök fontos dokumentumokat hurcolnak el.
Mai kínálatunk utolsó fogása a ***Freedroid RPG***.
A ***Johannes Prix*** és ***Reinhard Prix*** gondozásában készülő ***Freedroid RPG*** tetszetős, 3D-s szerepjáték kiváló grafikával, nagyszerű zenével és hanghatásokkal, jól felépített képzeletbeli világgal. A háttértörténet: valamikor a nem túl közeli jövőben egy hatalmas, ***Megasoft*** nevű vállalat gyakorlatilag átvette az uralmat a galaxis felett. A vállalat ezt azzal érte el, hogy erejét kihasználva a bolygó minden számítógépes eszközére trójai vírusokat telepített, még a kormányzati és a rendőrségi gépekre is. Az emberiség ezzel rabszolgává vált. Egy szörnyű programozási hiba folytán aztán a gépek fellázadtak és átvették az uralmat, így a dolgok még rosszabbra fordultak.
Az emberiség egyetlen reménye ***Tux*** egyfajta robot változata. Csúcstechnológiájú páncélzattal, ősi mágiával és lézerkarddal felszerelkezett hősünk készen áll a gépekkel vívott harcra és a galaxis felszabadítására. Utunk során a legkülönfélébb gonosztevőkkel és szörnyetegekkel kell megküzdenünk, tárgyakat, pénzt és

4. ábra Minden készen áll, hogy megkezdjük Freedroid RPG kalandjainkat

fegyvereket kell gyűjtenünk, valamint sokféle érdekes karakterrel találkozhatunk. Én például összefutottam egy ***Michelangelo*** nevű szakáccsal, akinek tönkrement a kedvenc sütője, és ezért dilítium kristályokra van szüksége.

Vajon ***Tux*** tud neki segíteni? Csak úgy tudhatjuk meg, hogy beszerezzük a ***Freedroid RPG***-t, és végigjátsszuk a történetet (lásd a forrásokat). Letöltéskor viszonylag nagy, 60 MB-os ***.tar*** állományra számítsunk. Végül is, erősen grafikus a játék, ami indokolja a méretet. A weboldalról forrásokat is letölthetünk, ám kár ezekkel vesződni, a legtöbb terjesztéshez bináris csomagot is találunk; én például ***Mandrake*** alatt próbáltam ki a játékot. Ha mégis inkább a forrást töltjük le, a kibontás és a fordítás a szokásostól picit eltérően, öt lépésben történik:

```
tar -xjvf freedroidrpg-0.9.12.tar.bz2
cd freedroidrpg-0.9.12
./configure
make
su -c “make install”
```

Ahogy az előző esetekben is, az ***SDL*** fejlesztői könyvtárakra most is szükségünk lesz. Ha a programot ***OpenGL*** támogatással szeretnénk lefordítani, akkor az ehhez szükséges csomagokról is gondoskodjunk. Ha a fordítás és a telepítés véget ért, elkezdhetjük a kalandozást.

A ***Freedroid RPG*** kétféle módon indítható, függően attól, hogy ***OpenGL*** gyorsítóval rendelkező rendszerünk van, vagy a jó öreg 2D gyorsításhoz ragaszkodunk. Ha ***OpenGL*** módot szeretnénk, a `freedroidRPG` parancsot kell kiadnunk (ügyeljünk a kis- és nagybetűk megkülönböztetésére), míg az ***OpenGL*** nélküli indításhoz a `freedroidRPG -n` parancsot kell begépelnünk.

A játék indításakor választhatunk, hogy egy- vagy többszereplős játékot szeretnénk, illetve dönthetünk, hogy meglévő hőst töltünk be vagy újat hozunk létre. Ha először indítjuk a játékot, nyilván új hőst kell létrehoznunk. Írjuk be az általunk kiválasztott nevet, mint például ***François***. Nyomjuk le az ***Entert***, és a kaland elkezdődik. Az első alkalommal olvassuk el a képernyőn gördülő bevezetőt, amely megismertet bennünket a játék háttértörténetével, a billentyűkombinációkkal és az egyéb tudnivalókkal. Ez egész csak néhány másodperc; amikor végeztünk, a kezdéshez kattintsunk.

A 3D világban az egér segítségével mozoghatunk. Kattintsunk hősünk elé, ő pedig követni fogja parancsainkat. Az egyes tárgyakat rájuk kattintva vehetjük fel, tudásunkat és eszközeinket – például a lézerkardot – pedig a jobb oldali gombbal kattintva vehetjük használatba. A képernyő jobb alsó sarkában három kör látható. Az egyikkel karakterünk statisztikáit jeleníthetjük meg, a másikkal a felvett tárgyak leltárát érhetjük el, a harmadikkal pedig varázslatokat és eszközöket választhatunk ki (ezek bal oldalon, alul jelennek meg). A játék során bármikor lenyomhatjuk az Esc gombot, ekkor megjelenik a menü, amelyből a program például a grafikára, a hangokra vagy a teljesítményre vonatkozó beállításait tudjuk módosítani. Alapesetben a játék teljes képernyős módban fut, ám a megfelelő beállítással ezen is változtathatunk.

Játék közben fogadjátok meg egy tanácsomat, ***mes amis***. A ***Freedroid*** – ahogy a többi mai játék is – rendkívül könnyen függővé tesz. Hirtelen rádöbbenhetünk, hogy az idő féktelenül rohan, miközben játszunk. Éppen ezért hívom fel a figyelmetek a mentési lehetőségre. Ha játék közben úgy gondoljuk, hogy aludni vagy enni sem ártana, nyomjuk le az ***Esc*** gombot, majd a menüből válasszuk a ***Save Game*** (Játék mentése) parancsot. Most már biztonságosan kiléphetünk, a legközelebbi alkalommal a játék indításakor válasszuk a ***Load Existing Hero*** (Meglévő hős betöltése) parancsot.

Mon Dieu, hogy szalad az idő! Éppen csak elkezdtük, és máris záróra van. Látom, túlságosan elfáradtatok a biztonsági küzdelmek hevében, inkább mennétek végre haza. Semmi gond. Készséges felszolgálóm, ***François*** még kinyit nektek egy utolsó üveg bort. Maradék időtökben talán újabb alkalmat kaptok arra, hogy megvédjétek a világot a vírusoktól, a férgektől, a velejéig romlott nagyvállalatoktól és az egyéb ocsmány fenyegetésektől.

A következő alkalomig igyunk egymás egészségére, ***mes amis***! A ***votre santé***!

Bon appétit!

Linux Journal 2004. október, 126. szám

Marcel Gagné (mggagne@salmar.com) Ontarioban, Mississaugaban él. Legújabb, immár harmadik könyve a Moving to the Linux Business Desktop (ISBN 0-131-42192-1, Addison-Wesley). A rendszerintegrálással és hálózati tanácsadással foglalkozó Salmar Consulting, Inc. igazgatója. Hobbipilóta, tudományos-fantasztikus írások szerzője, jelenleg egy kisebb origami T-Rexen dolgozik.

Játékkiszolgáló üzemeltetés Linux alatt

Hazánkban is egyre jobban terjed az otthonokban a széles sávú internet elérés, mely – többek között – elősegíti a hálózaton játszható játékok rohamos elterjedését is. Ebben a cikkben a játékkiszolgálók tervezésének és üzemeltetésének főbb kérdéseit tekintjük át.

Egy ilyen játék analóg modemmel valószínűleg nem lenne nagy élmény. Bár a 33.6/56 Kbit/sec-os átviteli sebesség még megfelelő volna, de a késleltetés élvezhetetlenné tenné a játékot. (A hálózati csomagok a kelleténél lassabban érnének célba.)
Cikkünkben azokat a szempontokat szeretnénk bemutatni, amelyek alapján az ***Interware*** saját szolgáltatásait kialakította. A cél az volt, hogy egy nagyobb létszámú játékos közösséget kiszolgáló, megfelelő minőségű szervert bocsássunk ügyfeleink és barátaink rendelkezésére.
A piacon rengeteg hálózati játék létezik. Ezek közül itt és most csupán egyet fogunk kiválasztani, hogy rajta keresztül bemutathassuk a szolgáltatói gyakorlatban alkalmazott módszereket. A „kiválasztott" ezúttal a ***Medal Of Honor Allied Assault*** nevű, már régebben kiadott és mára viszonylag olcsón beszerezhető program lesz.
A környezet kialakítása előtt – mint bármely más rendszer megtervezésekor – tisztáznunk kell bizonyos részteleket.

- Mekkora célközönséget szándékozunk kiszolgálni;
- Mennyi szabad erőforrásunk van a célra;
- Ellenőrzött szolgáltatást kívánunk e nyújtani, vagy nem törődünk azzal, hogy a játékosok mit művelnek a szerveren játék címén.

Bár a józan ész ezt diktálná az összefüggés a hardverigény és kapcsolatok száma között nem lineáris. A játékszerverek terhelhetőségét általában a szabad férőhelyek (***slotok***) szabályozásával, illetve az egy felhasználóra jutó sávszélesség meghatározásával adják meg.
Ez az a pillanat, amikor elérkeztünk két olyan, meglehetősen sarkalatos problémához, amelyektől a borúlátóbb rendszergazdák rendszerint visszariadnak.
Az egyik súlyos, de egyáltalán nem leküzdhetetlen akadály az, hogy a szoftvereket eredetileg nem ***Linuxra*** írták.
A linuxos változatok általában nem kellőképpen optimalizáltak, nem használják ki az operációs rendszer hasznos tulajdonságait, inkább kis önálló világként léteznek benne.
Egy ilyen program „el se tudja képzelni", hogy az általa lefoglalt memóriaterületen tárolt információt akár több hozzá hasonló folyamat is használhatná. Sajnos a legtöbb – általában már évekkel ezelőtt kiadott, de ma is kedvelt és nagy felhasználótábornak örvendő – játék sokkal jobban fut azon az operációs rendszeren, amire eredetileg fejlesztették.
A második komoly probléma tulajdonképpen kapcsolódik az előzőhöz: nagyon nehezen mérhető az ilyen jellegű folyamatok erőforrásigénye. A ***top*** parancsot használva például fel se tűnik, ha egy ilyen alkalmazásnak nem áll rendelkezésére elegendő mennyiségű rendszererőforrás. Igen jó példa erre, hogy egy átlagos kiszolgálón (***Athlon XP 2000+***, 512MB memória, 7200-as fordulatszámú ***UDMA/100***-as merevlemezek) 10 db játékkiszolgálót is el tudunk helyezni, és játékosokat behívni rá anélkül, hogy a gép mérhető terhelése (***load***) jelentősen megnőne. Ennek ellenére a játék élvezhetetlen lesz, mivel a játékosok tevékenységével összefüggésben keletkező terhelésugrások apró késleltetéseket képeznek. Bár ezek például egy weblap letöltésénél észre sem vehetők még 1000 egyidejű felhasználó esetén sem, egy játéknál a hatásuk „végzetes".
Felvetődik a kérdés: mit tehetünk egy épkézláb módszerekkel gyakorlatilag nem mérhető folyamathalmaz erőforrásigényének kielégítésére.
Először is van néhány íratlan szabály:

- Ha több kiszolgálót üzemeltetünk egy gépen, akkor érdemes nulláról telepített, optimalizált (erről később még lesz szó) operációs rendszert használni.
- Ne futtassunk más, nagy sávszélesség vagy processzorigényű alkalmazást a gépen! Egy egyszerű, néhány oldalt kiszolgáló webszerver még nem jelent problémát, egy adatbázis-lekérdezésekkel tarkított, sok felhasználós oldaltömeg viszont nem lesz ínyére a játékszerver(ek) folyamatainak.
- Nekünk a ***Debian/GNU Linux*** vált be legjobban, de ez természetesen nem jelenti azt, hogy egyéb terjesztésekkel bármilyen komolyabb probléma lenne.
- Ne próbáljunk otthonról, kábelmodemes vagy ***DSL*** kapcsolattal ilyen szolgáltatást megvalósítani. Biztosan állíthatjuk, nem fog sikerülni. Tegyük a kiszolgálót egy szerverhotelbe ahol sávszélességgel kapcsolatos probléma gyakorlatilag kizárt, hiszen például egy 20 főt kiszolgáló szerver 2 megabitnyi adattömegnél nem továbbít többet másodpercenként. (Az újabb játékok hálózati kódja még ennél is hatékonyabb, akár kétszer ennyi játékost is kényelmesen elbírnak ugyanekkora sávszélességen).

- Védjük gépünket! Hasonlóan az egyéb hálózati fórumokhoz, ennél a stílusnál is elég gyakori, hogy a közösség devianciára hajlamos tagjai bizonyos konfliktusokat a szerverek kiiktatásával próbálnak rendezni. Mindenképp használjunk tűzfalszoftvert – például ***Netfilter/iptables*** -, de kerüljük az integrált, nagy tudású, vírusszűrős képességekkel ellátott tűzfalakat, ezek ugyanis nagyban lassíthatják a kiszolgálót.
- Ne indítsunk be egyszerre több kiszolgálót. Helyesebb, ha mindegyik után megvárjuk a felhasználók/játékosok visszajelzéseit. (Fontos, hogy ne felejtsük el tőlük ezt kifejezetten kérni, mert maguktól esetleg nem nyilatkoznak majd.) Újabb kiszolgálófolyamatot csak tökéletes működés esetén indítsunk be.
- Ami a hardvert illeti, az annyi legyen „amennyit elbír a ház". Memóriával és processzorral ne spóroljunk. Egy ***MOH:AA*** szerver például akár 100 MB – néha több – memóriát is hajlamos magáévá tenni, ha pedig nem kap, akkor kíméletlenül elkezd a csereterülethez nyúlkálni. Darál a lemez, áll a játék. Ami a processzort illeti eddigi tapasztalataink alapján legalább 2 Ghz-es konfigurációval érdemes nekivágni. Ez körülbelül 5 kiszolgáló-folyamatot képes futtatni, de a terheléstől függően többet is beindíthatunk rajta. Ügyeljünk rá, hogy az igények játéktól függően erősen változhatnak. Ugyanazon a hardveren, amin egy ***Counter Strike***-ból csak 2 képes működni, ***MOH:AA***-ból elfut 8, egy ***BattleField 1942***-ből pedig akár 12 is! Ezek természetesen csak irányszámok.

Lássunk egy konkrét példát!

A hardver legyen egy 3 Ghz-es ***Pentium 4*** processzort tartalmazó gép. (***Intel*** processzorral valamiért jobb eredményeket értünk el, mint ***AMD***-vel.) Ehhez társul 1 GB memória, valamint 2 db 80 GB-os (7200 RPM) merevlemez, amelyeket szoftveresen ***RAID-1***-be szervezünk. Ez egyrészt a biztonságot szolgálja, másrészt gyors olvasást tesz lehetővé. Szintén jó megoldás lehetne egy vagy két 36 GB-os ***UW-3 SCSI*** merevlemez egy megfelelően gyors vezérlővel, bár itt már takarékosabban kell bánnunk a hellyel.

Telepítsünk egy ***Debian Sarge*** terjesztést úgy, hogy kizárólag az alaprendszert tegyük fel. A ***ppp***, ***pppoe*** csomagokat, valamint minden egyebet ami nem szükséges a rendszer működéséhez, távolítsunk el.

Hozzuk létre a szoftveres ***RAID*** lemezrészeket. A felhasználói könyvtárakat (***/home***) külön lemezrészre tegyük és használjunk ***XFS*** fájlrendszert. Ez például az ext3-hoz képest nagy előrelépés jelent a sebességben.

Szinte magától értetődik, hogy a legújabb, optimalizált kernelt kell használnunk, amit mi magunk fordítunk le. Mindenképpen a 2.6-os sorozatot használjuk! Számos dolog, ami a 2.6-osban alapértelmezett a 2.4-be még nem került be hivatalosan, vagyis a 2.4-eseket alakítgatni kellene. Szintén ajánlott 2.6.3 feletti változatot fordítani, mert ezekben kevesebb biztonsági rést találtak eddig. Használjunk ***grsecurity*** foltot is (➲ ***http://www.grsecurity.net***), amely szintén növeli a biztonságot.

Hozzuk létre azt a felhasználót, akinek a nevében az alkalmazás futni fog. Esetünkben legyen ez mondjuk ***mohaa***.

Most egy egyedi telepítési folyamat következik, mivel a ***mohaa*** kiszolgálónak nincs linuxos telepítője. Töltsük le a ***mohaa*** szerver linuxos bináris állományait a ***http://mohaa.hu*** címről. Telepítsünk egy windowsos gépen egy ***mohaa*** ügyfelet, az egészet csomagoljuk össze és másoljuk be az imént létrehozott felhasználó (***mohaa***) könyvtárába. A linuxos változatot ugyanoda kell tenni, ahol a windowsos állományok vannak. Ha ez megtörtént, a kiszolgáló elvileg már futtatható is a megfelelő parancs kiadásával.

Mitől lesz könnyen adminisztrálható és 24/7 üzemmódban futtatható a kiszolgálónk?

Ehhez először is egy könnyen kezelhető adminisztrációs segédprogramra lesz szükségünk. Ilyen például a ***Screen*** amelynek telepítése Debian alatt csupán az `apt-get install screen` parancs kiadását jelenti. Persze ne feledjük, hogy ezek az alkalmazások nem mindig futnak stabilan, vagyis nem árt megírni hozzájuk egy héjprogramot, ami figyeli, hogy fut e még a kérdéses folyamat, és ha nem, akkor automatikusan újraindítja azt.

Természetesen az is megoldható, hogy a rendszer újraindítása után automatikusan elinduljanak játékszervereink. Ilyenkor ügyeljünk rá, hogy a ***Screent*** ilyenkor automatikus leválásra (***detach***) utasítsuk, és ne felejtsünk el nevet adni a ***Screen*** által kezelt folyamatoknak.

Érdemes alaposan áttanulmányozni a cikk végén felsorolt dokumentumokat, hiszen ezekben rengeteg hasznos információt találunk. Vannak itt az adminisztrációt segítő programok, és számos ötlet olyan műveletekre, amelyeknek egyelőre nincs „szoftveres megfelelője".

A ***MOH:AA*** például nincs felkészítve arra, hogy „***ban***" listát tartson fent, vagyis ha kiutasítunk valakit a szerverünkről, bármikor visszajöhet ami lássuk be, nem túl szerencsés. Ilyenkor a rendszer ***route*** parancsával elirányíthatjuk a nem kívánt IP-t valamilyen nem használt, vagy nem is létező tartományba. Így már nem valószínű, hogy az illető vissza tud térni. Ügyeljünk arra, hogy a többség dinamikus címmel érkezik játszani, vagyis akit ma letiltunk, az holnap nem biztos, hogy ugyanaz a személy lesz. Érdemes ezt a problémát is egy héjprogramból kezelni, ha nem akarjuk szép lassan letiltani az összes szolgáltató dinamikus IP címtartományát.

Jó szerencsét és sok kitartást mindenkinek, aki játékszolgáltatásra adja a fejét. Sok fejfájással járó küldetés ez, no de embertársaink öröme minden energiát megér.

Vincze Gábor (vincze.gabor@interware.co.hu)
Vidám természetű, fanatikus rendszermérnök, az Interware Rt. munkatársa. Csak zene mellett képes dolgozni, éjszaka hálózati játékokkal tölti idejét, szereti a társaságot és a tengerentúlt.

KAPCSOLÓDÓ CÍMEK

- ➲ http://www.grsecurity.net
- ➲ http://mohaa.hu
- ➲ http://www.callofduty.hu
- ➲ http://www.counter-strike.hu
- ➲ http://www.mohadmin.com

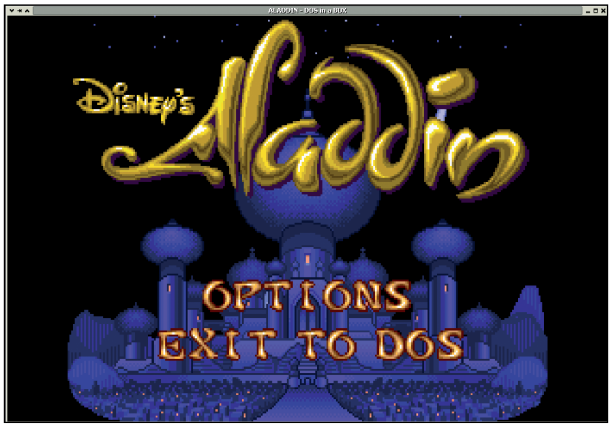
Régi idők játékai – a DOSEMU

A Linux univerzális DOS emulátora, avagy időutazás régi kedves játékaink korába.

Ki ne ismerné a jó öreg *DOS*-os játékok utánozhatatlan hangulatát? Minden régi motorosról elmondható, hogy – legyen immár akármilyen komoly ember is – egyszer, valamikor évekkel ezelőtt szabadidejében szívesen butította elméjét ezekkel az egyébként nagyszerű programocskákkal. Az ember hajlamos arra, hogy ezeken a szép dolgokon nosztalgiázzon kissé. Persze jelen esetben nem is nagyon tehet mást, hiszen a szép időknek is megvan az a rossz szokásuk, hogy elmúlnak egyszer. No nem mindörökre, de abban a tekintetben mindenképpen, hogy a mai operációs rendszerek jelentős része (***Linux***, ***WinXP***, ***Win2000***, stb.) nem képes ezeket a régi játékokat futtatni. Ha mégis játszani szeretnénk, akkor mindenféle egyéb trükkökhöz kell folyamodnunk: tarthatunk például egy *DOS* változatot valamelyik partíciónkon, vagy lehet hogy találunk a padláson egy régi gépet, amit már csak erre használunk. Az első módszer legfőbb hátulütője, hogy a *DOS* nem ismeri az új hardvereket, így legtöbbször még hangot sem csalogathatunk elő a játékokból. A második módszer már működhet, no de muszáj ehhez egy egész gépet fenntartani?

Természetesen nem, ugyanis létezik egy alternatív megoldás: a virtuális számítógépek nyújtotta lehetőség. Ezek olyan programok, amelyeket a befogadó operációs rendszer futtat, de fő feladatuk az, hogy elindítsanak más programokat, és elhitessék velük, hogy egy külön számítógépen futnak. Ha ez megvolna, nincs más dolgunk, mint az operációs rendszert, mint programot elindítani ezen a virtuális masinán és készen is van az új gép. Azaz készen ***volna***, ugyanis a helyzet ennyire azért nem rózsás. Ezek az általános virtuális gépek szinte képtelenek a grafikát szépen, jól, és kiváltképp kellő gyorsasággal megjeleníteni (tisztelet a ritka kivételnek), ezen kívül pedig jóval bonyolultabbak annál, ami a játékokhoz feltétlenül szükséges volna. Ráadásul ugyanúgy megmarad a hardver felismerésének problémája... Mindezen felül még megvan az a feladatunk is, hogy keressünk egy működő *DOS*-t, telepítsük, s megéljünk minden ezzel járó macerát.

Jól látszik a fentiekből, hogy nekünk valami olyan dologra lenne szükségünk, ami kifejezetten egy *DOS*-os környezetet tesz elénk, úgy, hogy egyrészt lássuk a hardvereszközöket a hagyományos formában, de ne kelljen azzal szenvednünk, hogy beállítsuk az új operációs rendszerünket. Emellett a rendszer legyen gyors is, ha egy mód van rá.



1. kép Aladdin, amit már DOS alatt is nehéz volt elindítani

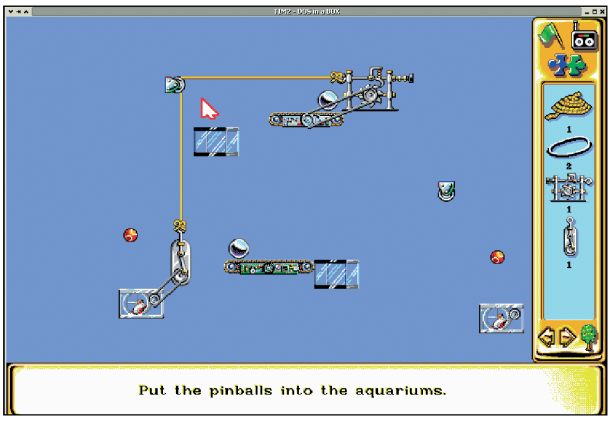
2. kép Egy másik nagy klasszikus: Duke Nukem 3D

Nos, van! Pontosan erre találták ki a ***Dosemu*** nevű programot, amely az első számú linuxos *DOS* emulátor. Felépítéséből adódóan leginkább a régi játékok kiszolgálására használják.

Mi is ez a Dosemu?

A ***Dosemu*** egy a maga nemében páratlan virtuális gép, amely kifejezetten a *DOS*-os programok számára szükséges erőforrásokat teszi elérhetővé. Fontos kiemelni, hogy nem

3. kép Oroszlánkirály – a zenéje sem marad el



4. kép The Incredible Machine – működés közben

csak a *DOS*-t, mint operációs rendszert bocsátja rendelkezésünkre, hanem a teljes hozzá tartozó környezetet is emulálja. Ez egyaránt vonatkozik a hardverelemekre és a szoftverekre. Van hangkártyánk (igazi ***Sound Blaster***!), CD-ROM, egér, ***VESA VGA***, hálózat, nyomtató, ***COM*** portok, numerikus társprocesszor, és minden egyéb, amire csak szükségünk lehet. Az egészben az a legszebb, hogy ezek (a CD-meghajtót leszámítva) alapértelmezetten be vannak állítva, vagyis nem szükséges például egérmeghajtót betölteni. Az csak úgy magától van, és minden program látja is.

Hogyan juthatok hozzá?

Mi sem egyszerűbb ennél: telepítsük rendszerünk csomagkezelőjével! Minden jelentős terjesztés (***Debian***, ***RedHat***, ***SuSE***) tartalmazza csomagok formájában, így a telepítés maga eltarthat akár két percig is! S bármennyire hihetetlen, az így előállt emulátor azonnal teljes értékű, nem szükséges további beállításokat végezni a helyes működéshez. Ahhoz, hogy ***X*** ablakban is jól működjön, szükségünk lesz egy betűtípus-csomagra is, amit ***dosemu-vga*** néven találunk a csomagok között.

Próbaképp indítsuk el nyugodtan, adjuk ki `dosemu` parancsot a héjprogramból, és nézzük meg, hogy mi történik. A rendszer betöltődik a terminál ablakban, ahonnan indítottuk, s maga az ablak egy külön gép, amelyben *DOS*-os parancsokat adhatunk ki. A *DOS*-t, mint programot a telepített ***FreeDOS*** nevű szabad *DOS* változat szolgáltatja. Ez a ***Dosemu lib*** könyvtárából nyíló ***freedos*** nevű mappában található, az újabb változatoknál egyenesen a ***UNIX***-os fájlrendszeren, nem pedig lenyomatban (image). Az említett könyvtár tartalmaz minden egyes *DOS* parancsot, súgót, és mindent, amit ettől az operációs rendszertől megszokhattunk. Természetesen van ***autoexec.bat*** és ***config.sys*** fájl is, s a rendszer betöltődésekor ezek lefutnak. Ez a könyvtár alkotja tehát a ***boot*** lemezrészt (partíció), amelyet az emulátor úgy lát, mintha az egy meghajtó (***C:***) lenne. Nem ez az egyetlen könyvtár, amely külön meghajtóként látszik, s ezáltal elérhető az emulátorból. Minden becsatlakoztatott könyvtárat elérhetünk az `lredir` parancs segítségével, sőt, különleges esetekben egész eszközleírókat is átadhatunk, mint lemezmeghajtót. (Hasonlóan használja a ***Dosemu*** a hajlékonylemezes meghajtót.) Ami a beállításokat illeti: Létezik globális és lokális beállítási fájl, de sok mindent megoldhatunk az ***autoexec.bat*** illetve a ***config.sys*** fájlok ***Linuxból*** történő szerkesztésével is. Az emulátorból az `exitemu` parancs kiadásával, illetve a CTRL+ALT+PGDN billentyűkombináció megnyomásával léphetünk ki.

Milyen eszközök állnak rendelkezésemre az emuláció során?

Alapszabály: a hardver csak akkor fog működni, ha a ***Linuxban*** helyesen be van állítva – ekkor viszont garantáltan helyesen fog működni.

Merevlemez

Kezdjük mindjárt a legfontosabb helyen, vagyis a merevlemezeknél. Mint már írtam, van egy ***C:*** meghajtónk, amely a *DOS*-t magát tartalmazza. Ez a ***boot*** lemezrész, amely csak olvasható. Ezen felül alapértelmezetten olvashatjuk ill. írhatjuk a saját könyvtárunkat (***home***) is. A ***Dosemu*** egyébként minden közvetett hivatkozást (szimbolikus linket) követ, így ha belinkeljük ide bármilyen befűzött (mount) fájlrendszer könyvtárát, bármit láthatunk a programból, mindössze egy-két linuxos héjparancs kiadásával.

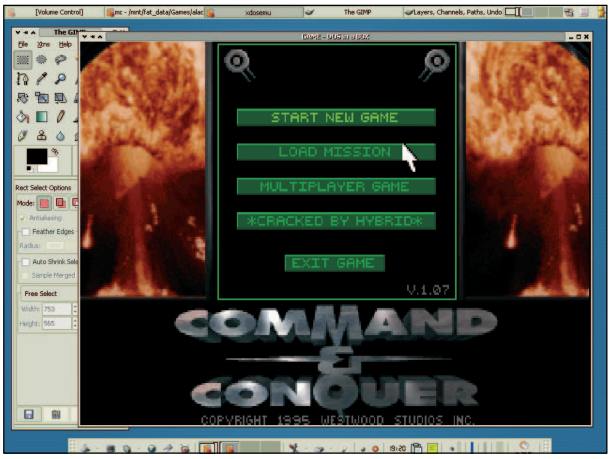
Ezen kívül arra is lehetőségünk van, hogy működés közben adjunk meg új meghajtókat. Erre szolgál a már említett `lredir` parancs, melynek segítségével az emulátorból egy meghajtó betűjelhez köthetünk a linuxos fájlrendszeren elhelyezkedő könyvtárat, illetve azok tartalmát. Íme egy példa:

```
lredir T: \linux\fs\mnt\drive_d
```

A `\linux\fs` előtag jelen helyzetben a linuxos fájlrendszer gyökerét jelöli, ez után kell tehát a könyvtár abszolút elérési útját megadni. Ha mindezt szeretnénk automatizálni, akkor nyissuk meg a ***/usr/lib/dosemu/freedos/autoexec.bat*** fájlt, és adjuk hozzá a fenti parancsot. Ennek hatására a *DOS* betöltésekor megtörténik az összerendelés, és attól kezdve ***T:*** meghajtóként érjük el az adott könyvtárat.

Egér

Már említettem, hogy mindenféle meghajtóprogram nélkül van egerünk, amelyet minden olyan programban használni is tudunk, amely a *DOS*-os egeret igénybe veszi. Egyetlen apró probléma adódhat ezzel kapcsolatban: a mai egerekkel



5. kép Command & Conquer – még mindig fut!

6. kép Network Q Rally a'la Linux

túl gyors a kurzor mozgása. Ezen a */etc/dosemu/dosemu.conf* fájl szerkesztésével segíthetünk. Keressük meg a `$_mouse_baud` változót, vegyük ki előle a megjegyzésre utaló jelet („kommentezzük vissza"), majd adjunk neki valamilyen kicsi egész értéket, mondjuk 1-et. Ezután újraindítva a *Dosemu*-t már jelentősen lassúbb az egér mozgása, de még mindig lehetne rajta mit javítani. Bár az 1-es a legkisebb megadható érték, ez még mindig gyors egy kissé. De sebaj: pár perc alatt hozzá lehet szokni.
Ezen a ponton talán érdemes megjegyezni, hogy a *dosemu.conf* fájlban minden lehetséges beállítás megjegyzésként szerepel („ki van kommentezve"). Amiket itt látunk, azok az alapértelmezett beállítások. Ha ezeken változtatni szeretnénk, tüntessük el a megjegyzés jelet az adott sor elejéről, majd írjuk át a változó értékét.

Billentyűzet

Ami a billentyűzetet illeti, itt is csak a billentyűzetkiosztásról érdemes szólni. Előfordulhat, hogy a *Dosemu* nem ismeri fel automatikusan az *X* billentyűzetkiosztását. Ilyenkor a már emlegetett *dosemu.conf* fájlban keressük meg a `$_layout` változót, s állítsuk `hu`-ra, különben az angol kiosztással indul a rendszerünk. Ide tartozik még a használt kódlapok kérdése is, ám ennek igazából csak akkor lesz jelentősége, ha terminál módban és unicode környezetből használjuk az emulátort. Éppen ezért ennek ismertetésétől most eltekintünk.

Botkormány

Az egérhez hasonlóan mindenféle vezérlőprogram nélkül elérhető a botkormány (joystick) is. A mozgás és annak érzékenysége a beállítási fájl `$_joy_*` változóival szabályozható.

Hangkártya

Igazi, emulált hangkártya áll rendelkezésünkre anélkül, hogy a kisujjunkat is mozdítani kellene érte. Az alapértelmezett beállítások szerint a *0x220*-as címen, *5*-ös IRQ-val, *1*-es ill. *5*-ös DMA-val rendelkezik. Természetesen van *MIDI* is, méghozzá a szabványos *0x330*-as címen.
Az összes, általam próbált program gond nélkül működött vele, számos beállítóprogram pedig magától felismerte. Egyetlen aprócska hibája, hogy az első hangtaktus előtt pukkan egyet, mintha csak bekapcsolnánk egy erősítőt. A hangkártya beállításai a `dosemu.conf` `$_sb_*` változók értékeinek megadásával változtathatók meg.

Memória

Van konvencionális, *XMS*, illetve *EMS* memóriánk is, mégpedig alapértelmezésként 640/8192/2048 Kb-os leosztásban. Ezek a méretek a beállítási állományban a `$_dosmem`, `$_xms`, `$_ems` változókkal szabályozhatók. Fontos hozzátenni, hogy ezek valójában különálló memóriaterületek, nem úgy, mint ahogy azt a *DOS*-ban megszokhattuk, ezért előfordulhat, hogy a `mem` parancs nem teljesen pontos értékeket ír ki.
Jó hír, hogy a 640kb-nyi konvencionális memória szinte teljes egészében szabad, ezért szinte bármilyen játékot elindíthatunk, amely ezt használja: nem fog memóriaproblémákkal leállni.

Nyomtató

Alapértelmezésként a *Dosemu* a rendszer nyomtatóit állítja csatasorba, s az egész folyamat úgy működik, mintha csövek (pipe-ok) segítségével irányítanánk át a nyomtatást. A *DOS*-os működés során simán használható az *LPT1* port, tehát működik például az `echo 1 > lpt1` parancs. Így a régi alkalmazások gond nélkül nyomtathatnak majd a gépünk printerére.

Beépített hangszóró (speaker)

Ezt az eszközt alapértelmezésként a *Dosemu* emulálja, mégpedig úgy, hogy működését a hangkártyán keresztül képezi le. Lehetőség van azonban a közvetlen hozzáférésre is, de ez nem ajánlott. Mellesleg azért is teljesen felesleges, mert a legújabb alaplapokon már nincs is olyan csatlakozó, amire a beépített hangszórót rádughatnánk. Helyette az alaplapra integrált hangkártya „pittyeg".

VESA/VGA

Ha az emulátort *X* ablakban futtatjuk (lásd később), szabványos *VESA 2.0* illetve *VGA* kompatibilis videokártyát szolgáltat a futó programoknak. Ehhez igazából nem is szükséges hozzányúlni, az esetek 99 százalékában tökéletes az alapbeállítás.

CD-ROM

Ez az egyetlen olyan eszköz, amely nincs beállítva alapértelmezetten. Ennek is megvan az oka: nem maga a fizikai eszköz, hanem a befűzött korong könyvtára látszik a *Dosemu*-ból. Ennek ellenére persze előfordulhat olyan eset is, amikor tényleg magára a meghajtóra van szükség.
A módszer ilyenkor pontosan ugyanaz, mint amit régen megszokhattunk. Egy ***CD-ROM*** vezérlőre van szükség, azután pedig az `mscdex.exe` nevű programra.
A ***CD-ROM*** vezérlő része a *Dosemu*-nak, így nem is kell mást tenni, mint hozzáadni a ***config.sys***-hez az alábbi sort:

```
devicehigh=c:\dosemu\cdrom.sys
```

Ezzel betöltődik a vezérlőprogram, s után már csak az

```
mscdex.exe /d:mscd0001 /l:<betűjel>
```

parancsot kell kiadni, és készen is vagyunk. Akárcsak a hőskorban, célszerű ezt az ***autoexec.bat*** fájlba beírni, hogy automatikusan végrehajtódjon.
Az `mscdex.exe` nem része a *FreeDOS*-nak, ezért javaslom, a *Google*-n keressünk rá a letöltési lehetőségre (A win95-ös változatra lesz szükségünk), majd szerezzük be.
Menet közben elvileg úgy tudunk korongot váltani, hogy meghívjuk az `eject.com` parancsot, amely kilöki a tálcát, ez azonban a gyakorlatban nem működik. Ha beteszem a CD-t, olvassa ugyan, de kivenni már csak akkor lehet, ha véget ér a *Dosemu* futása. Elképzelhető, hogy ezt a hibát a közeljövőben kijavítják a fejlesztők, ezért az sem kizárt, hogy mire a kedves olvasó kezébe veszi az újságot, már a hibamentes változat lesz tölthető a *Dosemu* honlapjáról.

Hogyan futtathatók a grafikus alkalmazások?

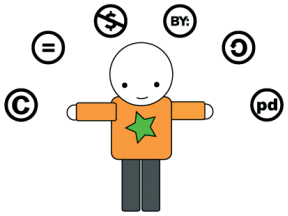
Szó esett arról, hogy a ***VESA/VGA*** mód ***X*** ablakban történő futtatás során érhető el. Ez a gyakorlatban azt jelenti, hogy a *Dosemu* nem a terminálban fut, hanem egy önálló grafikus *X* ablakban, ahol természetesen ugyanúgy a szöveges ablakot kapjuk meg elsőnek, ám grafikus formában. Ehhez kell az a bizonyos ***dosemu-vga*** nevű csomag, amely a szükséges betűtípusokat tartalmazza.
Én azt javaslom mindenkinek, hogy inkább *X* ablakban futtassa az emulátort, ugyanis a szöveges alkalmazások futása is simább, kevesebb hibát okoz, mintha terminálban tennénk mindezt.
Adjuk ki tehát a `dosemu -X` parancsot, melynek hatására egy új ablakban indul el a *DOS*-os gépünk. Most már bátran elindíthatunk bármilyen alkalmazást, az ablak átméreteződik a megfelelő felbontásra, majd megjelenik a grafikus kép. Nyugodtan próbáljuk ki az egyik régi CD-ről előszedett játékunkkal.
Természetesen nem túl jó, hogy kicsi ablakban fut a játék, mi több, ha egerezünk, kifejezetten zavaró, hogy kiszaladhatunk a képből, illetve hogy a hardveres és a szoftveres egérmutató nem mindig ugyanott van. Ezek elkerülésére adjuk ki a `dosemu -X -w` parancsot, amely teljesképernyős módban indítja az emulátort. Egy jótékony mellékhatása a dolognak az, hogy a billentyűkombinációk helyesen fognak működni. Az Alt+F4 például nem a *Dosemu* ablakát zárja be, hanem a futó program kezelheti le az eseményt.

Nagyjából a végére is érkeztünk a bemutatásnak. A futó *Dosemu* gyakorlatilag olyan, mint egy számítógép a számítógépben. Az alkalmazásainkat pontosan ugyanúgy futtassuk, mint ahogyan azt régen is tettük. Élmény lesz újra végigjátszani néhány klasszikus darabot.
Szeretném azonban megjegyezni, hogy nem csak játékra van lehetőségünk, bár én igyekeztem a cikk során mindenhol azokat a területeket kiemelni, amik leginkább a játékok esetében fontosak, s az egész emulátort egyfajta „Hogyan játszhatunk jól?” szemüvegen keresztül bemutatni.
A lehetőségek itt nem merülnek ki. Példaképp álljon itt egy mindennapos bosszantó eset: Az interneten számtalan helyen találhatók olyan állományok, amelyek önkicsomagoló ***.exe*** formájában vannak tömörítve. Ezt igen kicsi eséllyel tudjuk *Linux* alatt fájdalommentesen kitömöríteni. Hibátlan megoldást ad azonban a *Dosemu* használata. Helyezzük el a fájlt ott, ahová ki szeretnénk tömöríteni, majd az emulátoron keresztül keressük meg a fájlrendszerben és indítsuk el. Rövid műveletvégzés után előbányászhatjuk a *Linux* fából a kitömörített állományokat.
Sok ezer egyéb beállítási lehetőség is rendelkezésünkre áll a programmal kapcsolatban, ám ilyen terjedelemben csak ilyen felületes bemutatásra nyílik lehetőség. Azt hiszem, a teljes programról egész könyvet lehetne írni.
Annyit szóltam már a *Dosemu* előnyeiről, hogy gyorsan meg kell említsem: nem old meg minden problémát. Számos olyan gyengén megírt játék akad, amely nem hajlandó elindulni az újabb masinákon még valódi *DOS* alatt sem. Általában valamilyen programhibával szállnak el rögtön az indítás után. Ez természetesen az emulátorban is gondot okoz. Ezeket a játékainkat valószínűleg csak a régi vasakon tudjuk újra életre kelteni. Nem kell azonban elkeseredni, jó néhány olyan játék létezik, amely gond nélkül fut a fent részletezett környezetben. Jómagam is számtalan programot végigjátszottam újra. Ha valami nem fut elsőre, azért ne adjuk fel egyből, rengeteg testreszabási lehetőség van még a tarsolyunkban, egy kis nyomozás után bátran alkalmazhatunk néhányat, s meglátjuk: meg lesz az eredménye.
Mondanom sem kell, hogy az itt ismertetett megoldás jelenleg páratlan. Nem tudok még egy ilyen gyorsan és könnyen, valamint ilyen hatékonyan üzemelő programról, amellyel régi *DOS*-os játékokat lehet szóra bírni. S bár ennek nem sok gazdasági haszna van, a *Linux* megint csak feladta a leckét a vetélytársaknak.

Komáromi Zoltán
(komi@kiskapu.hu)
23 éves, a BME hallgatója, mellette PHP-programozóként dolgozik. Kedvenc területe a multimédia.

KAPCSOLÓDÓ CÍMEK

Az emulátor honlapja: ➲ http://www.dosemu.org



Néhány jog fenntartva – Creative Commons

Az alkotások felszabadítása az alkotás szabadságáért: hogyan teremthető folytonos átmenet a szabad felhasználás, és a teljes jogi védelem között.

Mi is az Creative Commons?

Olvasóink közül a webet böngészve bizonyára többen is találkoztak már ilyen feliratú logóval, ami egy felhasználási engedély jelöl. A ***Creative Commons (CC)*** nem könnyen fordítható magyarra. Szó szerint annyit tesz ***„Kreatív Közjavak"***, de talán a ***„Közös Alkotás"*** jobban kifejezi a jelentését. A *CC* egy amerikai non-profit szervezet, amely azt tűzte ki célul, hogy egyszerű felhasználói szerződések formájában alternatívát kínáljon a szerzői jogok jelzésére. Arra törekedtek, hogy olyan, minél könnyebben használható és értelmezhető licenceket készítsenek, amelyekre bárki hivatkozhat közzétett műveivel kapcsolatban, legyen az egy írás, zenemű, grafika, fénykép vagy film.

A szerzői jog célja ugyanis eredetileg sem pusztán a védelem és a felhasználás korlátozása. Van egy másik, legalább olyan fontos szerepe, ami a tilalmak erdejében háttérbe szorul: a szellemi alkotás ösztönzése és az egyetemes kultúra értékeinek megóvása. A törvényalkotók szándéka szerint „egyensúlyt teremt és tart fenn a szerzők és más jogosultak, valamint a felhasználók és a széles közönség érdekei között, tekintettel az oktatás, a művelődés, a tudományos kutatás és a szabad információhoz jutás igényeire is"

Mire jó a CC licenc?

A szabad szoftverek világát ismerő olvasóban joggal merülhet fel a kérdés: minek még egy licenc, mikor már így is annyi van belőlük, hogy kiigazodni is nehéz közöttük. Az a probléma, hogy szoftverektől eltérő alkotások esetében a létező felhasználási szerződéstípusok nem igazán praktikusak. Részint ezért született meg maga a *CC* is, melynek licenceit tulajdonképpen a szabad szoftverek felhasználási szerződései inspirálták, azok mintájára dolgozták ki őket.

A szoftverlicencekkel szemben a *CC* licencek előnye, hogy rugalmasan alkalmazhatók, szövegük mégis rövid, közérthető és bármilyen alkotáshoz felhasználható.

Az írásos műveknél jelenleg szinte mindenhol a hagyományosan elterjedt ©, azaz ***„Minden jog fenntartva"*** jelzéssel találkozhatunk, ami lényegében azt jelenti, hogy semmit nem tehetünk az adott művel, a szerző engedélye nélkül. Ez ugyan kissé sarkított megfogalmazás, mivel van néhány törvény által biztosított kivétel. Ilyen például az, hogy idézhetünk a művekből, ami a jóhiszemű felhasználás (*fair use*) tárgykörébe esik. A néhány meglevő kivételtől eltekintve ugyanakkor mindennemű „egyéb" felhasználáshoz engedélyt kellene kérnünk a szerzőtől, aki esetleg soha nem is akart bennünket korlátozni az adott tevékenységben, de jobb nem lévén kénytelen volt odabiggyeszti művel mellé a © jelet. Valójában nem is szükséges kiírni a mágikus jelet, mivel a művek a „jogi alapértelmezés szerint" eleve védelem alatt állnak. (Erről bővebben ***Dr. Dudás Ágnes*** rovatában olvashatunk). Az alapértelmezett jogvédelem ráadásul a szerző halála után még további 70 évig érvényben marad.

Jogos tehát a kérdés: mit tegyen az alkotó ha nem kívánja teljes mértékben korlátozni műve felhasználását? A *CC* licencek pontosan erre kínálnak megoldást. A szervezet által kidolgozott „jogi alapelemek" építőkockák módjára tetszés szerint, akár kombinálva alkalmazhatók.

A *CC* elemtípusok a következők:

- ***"Attribution"*** – engedélyezi a mű szabad másolását, terjesztését
- ***"NonCommercial"*** – kereskedelmi céllal nem használható
- ***"No Derivative Works"*** – nem módosítható, nem dolgozható át
- ***"Share Alike"*** – módosítható és átdolgozható, de az így létrejött művet csak az eredetivel azonos licenccel lehet publikálni.

A *CC* honlapján egy egyszerű űrlap segítségével, néhány lépésben ki lehet választani a számunkra legmegfelelőbb változatot. Ha például azt szeretném, ha ezt a cikket szabadon másolhassák és terjeszthessék mások is, de csak eredeti formájában, akkor az első és a harmadik feltételt jelölhetem meg. Eredményként megkapom a kódot, amelyet beszúrhatok a megfelelő oldal szövegébe. Ezután feltüntetem a cikk mellett a *CC* ikonját, ami egyben hivatkozás a licenc közérthető, egyszerűsített változatát tartalmazó oldalra. (Innen természetesen a részletes jogi szöveg is elérhető, ha valakinek szüksége lenne rá, például egy jogi eljárás során.)

A licenc kiválasztásánál megadhatok számos, a a műre jellemző információt is, amelyek metaadatként szolgálnak a könnyebb azonosításhoz. Ezeket az űrlap *RDF/XML* formátumban illeszti a kódhoz.

Miért akarná valaki „felszabadítani" művét ilyen licenccel?

Erre számos oka lehet. Ilyen például, ha azt szeretné, ha szabadon terjednének gondolatai, minél többen megismerhetnék művét, vagy esetleg át is dolgozhassák azt, illetve felhasználhassák saját alkotásukhoz.
Első hallásra talán furcsa, de e mögött akár üzleti érdek is állhat. Egy kezdő művésznek például hasznos promóciós eszköz lehet a nyilvánosság, hiszen ezzel olyan széleskörű ismertséget érhet el, amire egyébként nem volna lehetősége. Persze az is lehet, hogy egész egyszerűen azért használ *CC* licencet, mert szeretné megosztani másokkal tudását, alkotásával hozzájárulni az emberiség szellemi javainak gyarapításához. A Linuxvilág olvasóitól feltehetőleg nem áll távol ez a fajta gondolkodásmód, mivel a nyílt forrású fejlesztéseket és a szabad szoftver mozgalmat igen hasonló elvek vezérlik.
Hogy van rá igény, az nem kétséges – elég csak a *Google*-lal rákeresni a *„Creative Commons"* szavakra, vagy böngészés közben megfigyelni, milyen sokan használják már most is honlapjaikon.

Több, mint licenc

A *CC* licencek tehát megkönnyítik azoknak a szerzőknek a dolgát, akik nem szeretnék „bezárni" művüket a hagyományos copyright korlátai mögé. Ezzel elősegítik a gondolatok, ismeretek szabadabb áramlását és tágabb értelemben a kultúra terjedését.
Ám a *Creative Commons* tevékenysége nem pusztán könnyen alkalmazható licencek készítésében és közzétételében merül ki.
Bár nem kívánnak felépíteni egy központi adatbázist, mivel hisznek a web decentralizált modelljének hatékony működésében, a CC licenccel publikált alkotásokról elérhetővé tesznek egy folyamatosan bővülő gyűjteményt. És ami még ennél is fontosabb: úgy tervezték meg a licenceket, hogy ne csak a szerzők használhassák egyszerűen műveikhez, hanem a jogvédelem típusa alapján könnyen meg is lehessen találni azokat. A licencek kódjába szerkesztett metaadatok alapján azonosíthatók a megfelelő tartalmak, és fellelhetők a weben.

Icommons és a magyar CC projekt

Én körülbelül egy éve találkoztam először a *Creative Commons* névvel a weben barangolva. Külföldi oldalakon, főleg blogokon, egyre gyakrabban találkoztam a felhasználási licencet jelző logóval és merő kíváncsiságból megnéztem, mi is ez tulajdonképpen. Rögtön megtetszett és mivel éppen akkortájt dolgoztunk a Linuxvilág online archívumán, kézenfekvő ötletnek tűnt, hogy alkalmazzuk a magazin cikkeinek jogállására is. A honlapjukból az is kiderült, hogy fiatal kora ellenére a CC kezdeményezése nemzetközivé fejlődött. Az úgynevezett *icommons* projekt keretében más országok is csatlakozhatnak a *CC*-hez, és a helyi jogrendnek megfelelő, saját nyelvükre fordított licenceket tehetnek elérhetővé. Fel is vettem a kapcsolatot a nemzetközi *CC projekt*, az *iCommons* egyik képviselőjével, és felajánlottam, hogy a fordítást és a honosítást a kiadónk elvállalná. Egyrészt örömmel fogadtak, másrészt jelezték, hogy a dolog azért nem olyan egyszerű: a honosítási projektekhez létezik egy kidolgozott tervezet és bizonyos feltételeknek is teljesülniük kell.
Először is kell egy jogász végzettségű projektvezető, valamint szükség van egy „elismert" szervezet vagy intézmény közreműködésére, amely a nevét és támogatását adja az ügyhöz. Fontos ugyanis, hogy a *CC* licencek fordítása a hazai jogrendhez legyen igazítva. Néhány magyar fórumon felvetettem tehát az ötletet, de mivel nem nagyon tolongtak a jelentkezők, a kezdeti lelkesedésem kissé alábbhagyott és félretettem a dolgot.
Idén nyáron azonban amolyan váratlan fordulatként néhány hét leforgása alatt minden összeállt. ***Dudás Ági*** (a szoftverjoggal foglalkozó új cikksorozatunk szerzője) jelentkezett nálunk, hogy szívesen publikálna a lapban és mivel tudtam róla, hogy jogász és az ***FSF.hu*** egyik aktivistája, kaptam az alkalmon és elmeséltem neki a *CC* honosításának ötletét. Neki is megtetszett a dolog és hamarosan kiderült, hogy – tőlünk teljesen függetlenül – másokat is foglalkoztat a *CC* ügye. Velem csaknem egy időben kereste meg őt ***Bodó Balázs***, a ***BME Média Oktató és Kutató Központjának (MOKK)*** munkatársa.
Némiképp sorsszerű találkozásunk után nem sokkal már konkrét terveket szőttünk. Megvolt a projektvezető, hiszen Ági elvállalta ezt a szerepet, a *MOKK* „személyében" pedig volt már szervezetünk is.
Felmerült egy-két probléma a hazai szerzői joggal való harmonizációval kapcsolatban, de reményeink szerint ezek áthidalhatók. (Erről a témáról egy későbbi alkalommal számolunk be). Úgy véltük mindenképpen érdemes elkezdeni a szervezést, és elindulni a megvalósítás felé.
A fordítás már elkezdődött, halad a közös munka, aminek eredményeiről a Linuxvilág lapjain is beszámolunk.
Hamarosan elindítjuk a ***creativecommons.hu*** honlapot is, ahol az érdeklődők folyamatosan nyomon követhetik az eseményeket.
Amikor legutóbb jobban körülnéztem a weben, feltűnt, hogy már számos magyar oldalon is felbukkan a CC logója. Ez egyértelműen mutatja, hogy idehaza is van rá igény. Remélem már nem kell soká várnunk arra, hogy magyarul is egyre több helyen olvashassuk a következő feliratot:
„Néhány jog fenntartva!"

Kelényi Attila (attila@linuxvilag.hu)
A Kiskapu Kft. ügyvezető igazgatója, a Linuxvilág marketingvezetője. Menthetetlenül netfüggő, de szenvedélye az aikido, a spanyol nyelv és ha tehetné, csak utazna.

KAPCSOLÓDÓ CÍMEK

- http://creativecommons.org/
- http://www.lessig.org/blog/
- http://search.creativecommons.org/index.jsp
- http://mokk.bme.hu/

Szoftverjog – személyhez fűződő jogok

A szerzőknek, mint már korábban is emlegettem, két nagy csoportba sorolható jogosultsági vannak. Az első halmaz a személyhez fűződő jogoké, míg a második a vagyoni jogoké. Ismerkedjünk meg ezekkel kicsit közelebbről.

Elhatárolás

A személyhez fűződő jog, mint kifejezés kicsit talán félrevezető, hiszen, ha valaki ezt a szókapcsolatot említi, akkor – a szerzői jogászokon kívül – mindenkinek először a polgári törvénykönyv vonatkozó rendelkezései jutnak eszébe. Amely – bár rokon terület – mégsem pontosan egyezik a szerzői jogban használt fogalommal.
Lássuk először a polgári jogot, mint mögöttes joganyagot[1]. Tehát a személyhez fűződő jogoknak ebből a szempontból több alcsoportja lehetséges. Ilyenek a Ptk. 76.§-ban felsoroltak:

- az egyenlő bánásmód követelményének megsértése,
- a lelkiismereti szabadság sérelme,
- a személyes szabadság jogellenes korlátozása,
- a testi épség, az egészség, valamint a becsület és az emberi méltóság megsértése

Láthatjuk, hogy itt többnyire az Alkotmányból is ismert szabadságjogok köszönnek vissza, a diszkrimináció tilalma, a szabad vallásgyakorlás, az emberi méltóság védelme.
A bíróság jogértelmezése pedig még szélesebbre tárja ezeket a kapukat. Elég ha csak arra gondolunk, hogy 1995-ben a Legfelsőbb Bíróság megállapította a hátrányos megkülönböztetés tényét, mikor egy pénzintézet, illetve egy biztosító társaság bejárata nem tette lehetővé, hogy azt mozgássérült személyek is megközelítsék.
A szerzői jogban kifejezetten az egyenlő bánásmód követelményét felállító szabályt nem találunk. Erre vonatkozó következtetést vonhatunk le mégis abból, hogy a szerzősség lehetőségét kifejezetten csak a mű megalkotásától teszi a törvény függővé. A helyzet „nemszabályozással” pedig visszautal a Ptk. említett paragrafusára.

Csak erélyesen!

Ha például egy megrendelő szoftver fejlesztő munkájáért csak azért fizet kevesebbet, mert a programozó nő, az adott hölgy joggal háborodhat fel, és fordulhat a bírósághoz. 2004. januárja óta egységesebb lett a jogrendszer annyiban, hogy most már nem csak a munkaviszony, hanem a munkavégzésre irányuló egyéb jogviszonyokban is az igazságtalan megkülönböztetés esetében a munkáltatót (megbízó, foglalkoztató, stb.) terheli a bizonyítási kötelezettség, mely szerint a kisebb bér fizetésének más okai is fennállnak,... hogy a megkülönböztetés nem volt jogellenes.

Nézzük ezt a példát kicsit alaposabban kifejtve:

Klementina tizenkét éves kora óta perfekt C programozó, az évek folyamán elvégezte az egyetem informatika karát, tudását bővítette (C++ -ban is profi lett), látókörét szélesítette, és valamely rejtélyes okból kifolyólag úgy döntött, hogy a szőke haj, jobban megy a szemszínéhez. Már másfél éve dolgozik az egyik szoftveróriásnak, mikor felvesznek mellé egy új kollégát: Taszilót.
A projecten közösen munkálkodnak, Klementina tanítgatja az új jövevényt, a bérüket banki átutalással kapják, minden szép és jó, mindaddig, míg egy céges bulin ki nem derül, hogy Klementina zöldfülű munkatársa közel másfélszer annyit keres, mint a lány. Ha most hősnőnk, miután sikertelenül próbálja meg tisztázni a helyzetet a cégvezetéssel, bírósághoz fordul, a cég kell, hogy bizonyítsa mennyi egyéb – a cég számára hasznos, és ezért a magasabb fizetésében megmutatkozó – erénye van Taszilónak. Amennyiben ez nem sikerül, Klementina fizetésemelésre is feltehetően sor kerül, – a bíró személyétől illetve a kereseti kérelemtől függően – akár több hónapra visszamenőleg is.
Ha azonban a munkavállalók közötti különbségtétel alapja, hogy Klementina csak főiskolát, míg az új jövevény két egyetemet is elvégzett, a fizetés különbségnek helye van még akkor is, ha Tasziló C++ tudása (a közgazdász és gépész diplomája dacára) fabatát sem ér.

[1] Mögöttes annyiban, hogy ha nem találunk esetleg a szerzői jogban megfelelő szabályokat, akkor a jó öreg Ptk-t kell elővennünk, és abban kutatni válasz után.

A névviselés joga

Mindenkinek joga van a névviselésre. Sőt, tovább megyek – a törvény korlátai közt, mindenkinek joga van arra, hogy olyan nevet viseljen, amilyet szeretne. Ennek szabályait egy 1982-ből származó törvényerejű rendelet szabályozza. A névváltoztatást a belügyminiszter engedélyezi. Az első változtatás 10.000 minden további 20.000 forint. Ha például szeretnénk felvetetni nicknevünket az anyakönyvbe, annak várhatóan ilyen illeték költségei lesznek... és várhatóan nem fog sikerülni, mert a neveknek a magyar nyelvhez kell igazodniuk. Persze ha András néven csetelünk, miközben minket Péternek hívnak, a változás nem kudarcra ítélt. Csupán annak van kevés esélye, hogy pl: Dudás Eowen Ágnessé válhassunk az eljárást követően.

A szerzői jog ezen a ponton különleges helyzetet teremt, ugyanis lehetőséget ad – anyakönyvezés nélküli – névváltoztatásra. Felvett néven u.n. „művésznéven" vagy „írói álnéven" is dolgozhatunk. Ahogy azt már sokan előttünk megtették, elég csak P. Howard-ra gondolunk.

Ugyanakkor figyelnünk kell arra, hogy választott nevünkkel ne sértsünk meg mást. Azaz, nem szórhatjuk tele a falu postaládáit röplapokkal, melyeken a szomszédunk neve alatt, silány minőségű, és minden egyes falubelit vérig sértő verseket költöttünk, bármennyire is haragszunk rá a zöldebb füve miatt.

Külön érdekessége a dolognak, hogy ha mondjuk Kertész Áron fiaként az Imre nevet kaptuk, és valamikor 20-25 évesen íróvá cseperedünk, – kérésre – a saját nevünket leszünk kénytelenek valamilyen kiegészítő jellel, toldással vagy szóval ellátni, nehogy összetéveszthetővé váljunk a Nobel-díjas elődünkkel, persze asztalos vagy pék nyugodtan lehetünk a megkülönböztető jelzés nélkül is.

Tekintve, hogy jóhírnév sérelmének kérdését már az előző számban tisztáztuk, átevezhetünk ténylegesen a szerzői jog tengerére. Vitorlát bonts!

A mű nyilvánosságra hozatalának joga

A törvény által említett első jogosultság a mű nyilvánosságra hozatalának joga. Ez egészen pontosan azt jelenti, hogy a szerző maga döntheti el, meddig akar az asztalfiókja – winchestere – számára dolgozni, és mikor ad egy esélyt magának, hogy művével előrukkolva világhírre tegyen szert. A nyilvánosságra hozatalhoz adhatunk szóban, vagy írásban engedélyt. Megadottnak tekinthető ez akkor is, ha a programunkat, alkotásunkat közzé tesszük a neten. A nyilvánosságra hozatal lehetőség és nem kötelesség. Ugyanakkor vélelem szól amellett, hogy ha egy felhasználási szerződés keretében jogokat adunk az általunk alkotott mű többszörözésére és terjesztésére, akkor egyúttal a nyilvánosságrahozatalához is hozzájárultunk. Ez ugyanakkor fordítva nem igaz. Ha tehát egy játékprogramot átadunk kipróbálásra, az nem jelenti azt, hogy az, aki megkapta továbbadhatja vagy akár másolatokat készíthet róla.

Engedély kell nem csak a mű egészének, hanem a részletek nyilvánosságra hozatalához is. Tehát ha a játékprogram demo változatával akar a megrendelőnk kedvet csinálni a vásárlóknak, a demo változatot is csak az engedélyünkkel teheti közzé egy számítógépes lapban vagy akár a weben. Kicsit mások a körülmények, ha mondjuk a jogdíjnak egy jelentős részét már megkaptuk, a tényleges programátadás előtt.

Ha irodalmi párhuzamot keresünk, Kafka az egyik legjobb példa. Az örök elégedetlen író, aki szinte mániákusan vetette tűzre írásait, halálos ágyán is azt kérte barátaitól, hogy égessék el a be nem fejezett, illetve ki nem adott műveit. Az egyes gimnazisták szerint átkozott Max Brod azonban nem hallgatott rá, így lehet jelenleg is tananyag „A per", illetve „A kastély." Talán pont ezen a példán okulva rendelkezik jelenleg is úgy a szerzői jogi törvény, hogy az alkotó halálát követően előkerült művek vonatkozásában a szerző nyilvánosságra hozatalhoz való hozzájárulását megadottnak kell tekinteni. Az már más kérdés, hogy ezt a vélelmet jogutódja, örökösei megdönthetik, ámbár miért tennék, mikor a felhasználási szerződéseket helyette ők fogják megkötni, és a jogdíj őket illeti majd meg.

Ugyanennek a jognak van másik arca is. Ez pedig a nyilvánosságra hozatalhoz adott hozzájárulás visszavonása. Első ránézésre idegennek tűnhet ez a szabály. Való igaz, ritkán használt jogintézményről van szó. Leginkább azért, mert már megadott engedélyt visszavonni csak alapos okból és kártalanítás mellett lehet. A visszavonás hatálya ilyenkor teljes, a visszavont művet sem felhasználási szerződés keretében, sem szabad felhasználás formájában nem lehet használni. A visszavonási jog a szerződéses jogviszonyban felmondási oknak minősül, és csak akkor gyakorolható, ha a szerző a kártalanítási költségeket fedező biztosítékot ad. A további kitétel pedig az, hogy ha a szerző ismételten meggondolja magát és mégis szeretné nyilvánosságra hozni művét, a korábbi felhasználási jogosultat u.n. „előfelhasználási jog" fogja megilletni, ami az elővásárlási joghoz hasonlít.

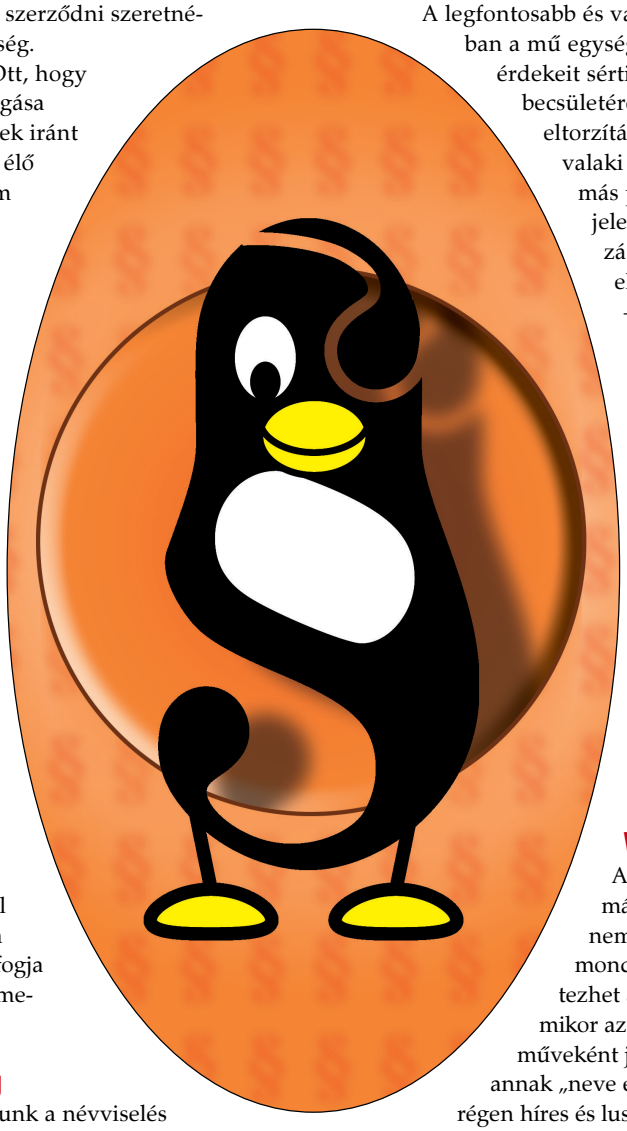
Példálódzunk...

Adott egy programozó, nevezzük Java Benedeknek, aki magányos harcosként fejlesztget a kishúgának egy estimese felolvasó programot. A program működik. Hug persze elmeséli az oviban mindenkinek, hogy neki milyen nagy és okos bátyja van. A programnak híre megy, és az Ovónők Országos Szövetsége (a továbbiakban: OOSz) is szemet vet az alkotásra, meg is keresik programozónkat. Megállapodnak a feltételekről, a fizetendő díjról és Benedek buta módon – vagy mert nem olvasott elég Linuxvilágot – kizárólagos jogokat ígér a Szövetségnek. Eltelik pár nap, és megjelenik a Babysitterek Ligája, akik lényegesen több pénzt ajánlanának a programét, ami azonban már elkelt. Benedek, miután tanulmányozta a szerzői jogi törvényt, azt a csalafinta megoldást találja ki, hogy a program nyilvánosságrahozatalához adott jogot visszavonja, ugyanis, indokolása szerint „nem helyes az, ha a gyerekeknek géppel olvastatunk fel ahelyett, hogy magunk mellé ültetnénk a kicsiket minden este, és mesélnénk nekik óriásokról és törpékről, meg sárkányokról, akik nem eszik meg a királylányokat vacsorára." Az indok – lássuk be – alapos. Az Ovónők Szövetsége bánatosan ugyan, de tudomásul veszi a döntést és a szerződés felmondását. Mivel még túl sok beruházást nem eszközöltek ezzel kapcsolatban, Benedeknek mindössze azon

levelek postaköltséget kell megtérítenie, amelyekben sajnálattal értesítik a megrendelőket, hogy a termék mégsem kerül forgalomba. Benedek ezután – mondván hogy újra megváltoztak az indokai, mert a „gépi mese még mindig jobb, mint a semmilyen” – leül tárgyalni a szoftverről a Babysitterekkel, és ha időközben már okosodott (vagy megkeresett egy ügyvédet), akkor nekik nem kínál kizárólagos jogokat. A szerződést azonban annak megkötése előtt meg kell mutatnia az OOSz-nek, és ha azok az újrafogalmazott feltételek mellett szerződni szeretnének, őket illeti meg az elsőbbség.

Hol van ebben a turpisság? Ott, hogy ha Benedek időszakos megingása nem volt őszinte, és a gyerekek iránt érzett rokonszenve – mellyel élő mesélőt kívánt nekik program helyett – csak színlelés volt, könnyen előfordulhat, hogy az OOSz bírósághoz fordul, és kéri a helyzet orvoslását, méghozzá olyan módon, hogy az eredetileg kialkudott díjért nyerjen kizárólagos felhasználási jogokat.

A helyzet még egy kicsit komplikáltabb, ha Benedek a mesefelolvasó programot munkaviszony keretében fejlesztette. Ha ilyen esetben, a szoftver elkészítését és bemutatását követően következik be életében az a lelki törés, amely miatt nem akarja, hogy a gyerekek PC mesélőt hallgassanak, akkor a folyamatot már nem tudja megállítani. Mindössze annyit érhet el, hogy a felolvasó szoftverről lemarad a neve. Azaz, ha nem árulja el senkinek, senki sem fogja megtudni, hogy ő volt a „gépmesélés” atyja.



A névfeltüntetéshez való jog

Az előbbi példával el is jutottunk a névviselés problémájához. Aki volt már valaha nagyobb cégnél programozó, vagy próbálta már megtalálni egy szoftver szerzőjét, az meglepődve olvashatta az előző sorokat. Nem, nem az történik, amire következtetnénk, hogy minden programozó mindig visszavonja a nyilvánosságra hozatalhoz adott engedélyét, és ezért kell őket eltüntetni a cd-ről. Hanem sokkal inkább az a jogsértő tendencia hatalmasodik el – nem kevés amerikai behatásra – hogy a szerzők névtelenségbe vesznek. Sokkal inkább az a helyzet, hogy nem tudják, jogukban állna szerzőként feltűnni, vagy egyszerűen csak nem merik szóvá tenni ez irányú igényeiket. Szoftverek esetében nehézség az is, hogy a szerzők nevének feltüntetésére adott esetben csak a forráskódban történik meg, amit azonban – leszámítva a nyílt forráskódú programokat – titokban tartanak.

A név feltüntetésén kívül megilleti a jog a szerzőt, hogy fellépjen azzal szemben, aki szerzői minőségét kétségbe vonja. Megilleti ez a jog azt is, aki a felhasználási jogokkal rendelkezik, aki adott esetben a „c” betű jogosultja.

A mű integritásához való jog

A legfontosabb és valóban használható jog azonban a mű egységének védelme. A szerző érdekeit sérti a mű bármiféle torzítása, becsületére vagy hírnevére sérelmes eltorzítása. Ide tartozik például, ha valaki trójai falovakkal teszi tele más programját. Ez persze nem jelenti azt, hogy bármiféle változás sérelmes, hiszen a művet előnyére változtatható persze – a magyar szabályok szerint – csak a szerző hozzájárulásával. A szerző azonban visszaélésszerűen ezen hozzájárulást nem tagadhatja meg.

Szoftver esetében például annak egy nem tetsző irányban történő továbbfejlesztése esetén az engedély megtagadásával – munkaviszonyban alkotott mű kivételével – megfékezhető a folyamat.

A kivétel esetében azonban ismételten csak a név elhallgatása kérhető.

Végszó

A személyhez fűződő jogok másik sajátossága, hogy azok nem ruházhatóak át, azokról lemondani nem lehet, már nem létezhet az a rabszolga jellegű munka, mikor az alkotást egy jó nevű ember műveként jelenik meg, csak azért, mert annak „neve eladható.” Ez előfordulhatott régen híres és lusta írókkal... ám mostanában tekintve az egyébként is névtelen programozókat, kevéssé valószínű. Le kell szögeznünk azonban, hogy mikor egy szoftvert „eladunk” egy cégnek, az említett jogaink megmaradnak. Felszólalhatunk, ha azt a nevünk nélkül akarják közzé tenni, ugyanúgy, mintha azt megcsonkítva akarják közzétenni.

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.